Máster en Ingeniería de Sistemas y Control

Desarrollo de un Laboratorio Remoto para el control por eventos de una planta de cuatro tanques acoplados

Autor:
Jesús Chacón Sombría

Directores:
Ph.D. Sebastián Dormido Bencomo
Ph.D. José Sánchez Moreno

Curso académico 2010-2011, convocatoria de Septiembre.

Master in Control and System Engineering

Development of a remote laboratory for decoupled event-based control of a quadruple-tank plant.

Author:
Jesús Chacón Sombría

Advisors:
Sebastián Dormido Bencomo
José Sánchez Moreno

Academic course 2010-2011, September call.

Autorización

Autorizamos a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado:

Firma del alumno

Abstract

In this work we focus on the development of a software tool which will allow the user to perform experiments in an easy way with event-based PI controllers in a multivariable system composed of four coupled tanks. The quadruple-tank plant is very interesting from the academic point of view, because it allows the control engineering student to experiment and obtain an intuitive knowledge about multivariable systems (systems with more than one inputs and more than one outputs), study the difference between the minimum phase and non-minimum phase behaviours and the difficulties in control that arise in the latter case, and all this with the motivation that the experiments can be done not only in simulation but with a real plant.

The application has been designed to work as a remote laboratory, in the sense that it is divided to two differents parts, on one hand the client side that the user can run from any computer with an internet connection and a Java virtual machine, and on the other hand the server side, running in the PC host connected to the plant in the real laboratory. In addition, the simulation has been included with the purpose of allowing the user to experiment with the application even when the real plant is not available.

The controllers implemented are two PI (because we are dealing with a 2x2 system) that can be configured to act as a periodic or time-based controller, in which the control action is updated with a fixed time interval, or to act as an event-based controller where the control action is updated when an event triggering condition is satisfied.

The control system provides the possibility of use a decoupling network between the controllers and the actuators, which can have a direct or inverse scheme, and which can be used to reduce the effect of the interactions between inputs and outputs.

Finally, we present a set of results illustrating the possibilities of the application to investigate the performance of the event-based controller together with a decoupling strategy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

When teaching engineering and in particular control theory, it becomes very important for a student to see in practice the theoretical results and get familiarized with the equipment and the instruments manipulation in real situations [1]. However, in many cases the acquisition and maintenance of laboratory equipment represent an excessive cost that can not be afforded for some colleges [2]. In addition, the laboratories need the presence of an instructor and therefore frequently the timetable is limited.

The use of remote laboratories is thus an alternative to be considered, to allow students to carry out experiences as if they were in the laboratory. It is common that these remote laboratories provide a graphical interface to interact with the plant, and video streaming so that students can watch what actually happens, having a most sense of presence in the laboratory.

The quadruple-tank introduced by Johansson ([3],[4]) has received a great attention because it presents interesting properties in both control education and research. The quadruple-tank exhibits complex dynamics in an elegant and simple way. Such dynamic characteristics include interactions and a transmission zero location that are tunable in operation. With adequate tuning this system presents non minimum-phase behavior that arises due to the multivariable nature of the problem. For this reason, the quadruple-tank has been used to show the results of different control strategies and as an educational tool in teaching advanced multivariable control techniques.

In other context, in recent years we have witnessed rapid progress in the field of wireless communication. Examples are systems using infra-red IrDA for short-range communication, personal area networks (WPAN), Bluetooth protocols and ZigBee, etc. Although many people have been aware of this thanks to the spectacular development of mobile telephony, there are many other possible applications for different types of existing wireless technologies, especially wireless sensors and sensor networks, military applications, home automation, agriculture, logistics, etc. Basically, a wireless sensor is a system consisting of radio frequency (RF) transceiver, sensors, microcontrollers and power sources (typically batteries). In some situations in automatic control, it is convenient

to have a wireless communication channel to send data from the data acquisition system to the control system. There are numerous examples in vehicle guidance applications, food traceability systems, or in irrigation control systems, where wiring suffers wear much faster. One of the main advantages achieved with wireless transmissions are the significant cost savings associated with instalation and system maintenance.

However, the use of wireless communications to transmit information leads to a number of drawbacks. Also, the use of a shared communication channel can introduce other problems that adversely affect the stability of systems. In addition, the waste of energy in the wireless modules must be kept to a minimum, in order to ensure the autonomy of those elements for a reasonable time. For these reasons we must optimize the delivery of data from different sensors, and a way to achieve this is through the use of sampling techniques and event-based control. For example, instead of sending data or control actions with a constant sampling period, the communications are done only when certain events are triggered.

In summary, event-based strategies are advantageous when the nature of the control problem impose restrictions over the number of control actions to be applied. This is due to the fact that, in periodic control systems, communications are done regardless of the state of the plant, and thus much of the information flow can be unnecessary. In [5] periodic and event-based sampling for first order stochastic systems are compared. Different types of event-based PID control algorithms have been proposed in recent works, as in [6] and [7] where a send-on-delta sampling is incorporated to the controller. In [8] a state-feedback approach with a disturbance estimator is investigated. From the implementation point of view, frequently the data acquisition hardware is designed to be used in periodic sampling systems. It is thus necessary in these cases to perform periodic sampling at high frequency and then to implement by software the event detection mechanism. Moreover, in many cases the energy efficiency is a key issue, as it occurs with wireless sensor network (see [9]), where the elements have autonomous power supply. In this cases, the communication must be optimized to obtain the maximum life of the batteries.

If we combine the event-based concepts mentioned before with control strategies from multivariable systems, we could take advantage of the benefits from each field. Then, it would be very useful to have a remote laboratory both for teaching control theory concepts involving multivariable, non-minimum phase behavior, etc., but also for the research in event-based control of multivariable systems.

In this work we have addressed both issues by developing a remote laboratory with the quadruple-tank and we using it to obtain some interesting results.

Chapter 2

Objectives

The main objectives of this work are to develop a software tool which will allow the user to perform experiments in an easy way with event-based PI controllers in a multivariable system composed by four coupled tanks, and to research the performance of this algorithm that combines event-based controllers with decoupling strategies.

This objectives can be decomposed into several subobjectives, which are listed below:

1. Verification of the correct working of the different components (sensors, DAQ systems, pumps, etc.) and get familiarized with the system.
2. Development of a LabVIEW Virtual Instrument to:
   (a) interact with the plant (sensor readings and pumps control),
   (b) implement the controller and the decoupler, and
   (c) simulate the plant model.
3. Development of an user interface to interact with the system, i.e., to show the state of the plant, the control signals, PI controller events, etc., and to provide a way to modify the configuration of the control system.
4. Research in a new method combining event-based controllers with decoupling strategies.

Chapter 3

Resources

This chapter contains a description of the resources used in this work, grouped into software resources - both for development process and use of the plant - and hardware resources such as the plant itself, PCs, etc.

3.1 Software resources

Several software tools have been used for developing the present work. The most important are those listed below,

EJS 4.3.2, Easy Java Simulations, also known as EJS or Ejs, is a free authoring tool written in Java that helps non-programmers to create interactive simulations in Java, mainly for teaching or learning purposes. EJS has been created by Francisco Esquembre and is part of the Open Source Physics project.

LabVIEW 8.5, is a graphical programming environment used by engineers and scientists to develop sophisticated measurement, test, and control systems using intuitive graphical icons and wires that resemble a flowchart. It offers integration with thousands of hardware devices and provides hundreds of built-in libraries for advanced analysis and data visualization – all for creating virtual instrumentation. The LabVIEW platform is scalable across multiple targets and OSs, and, since its introduction in 1986, it has become an industry leader.

LaTeX, is a document preparation system for high-quality typesetting. It is most often used for medium-to-large technical or scientific documents but it can be used for almost any form of publishing.

3.2 Hardware resources

3.2.1 Description of the plant

The plant to be controlled is composed of two Two-Tank plants from Quanser, which are used simultaneously and coupled to obtain a more complex Multi-Input-Multi-Output (MIMO) experiment, the quadruple-tank process described in [3], [4]. It can be shown that the four-interconnected-tank system has an adjustable zero, which can be moved along the real axis in the left- or right-hand-side of the s-plane. Therefore by changing the system parameters, the multivariable zero dynamics can be configured to be either minimum phase or non-minimum phase.

Figure 3.1: Two-Tank System from Quanser.

The Coupled-Tank overall frame is made of Plexiglas. The system's two water tanks are made out of Plexiglas tubes of uniform cross section. The Coupled-Tank pump is a gear pump composed of a 12-Volt DC motor with heat radiating fins. The materials that come into contact with the fluid being pumped are: two molded Delrin gears in a Delrin pump body, stainless steel shafting, a Teflon diaphragm and a Buna-N seal. It is also equipped with 3/16" ID hose fittings.

Each tank's actual liquid level is measured through a pressure sensor. Such a level sensor is located at the bottom of each tank and provides linear level readings over the complete liquid vertical level. In other words, the sensor output voltage increases proportionally to the applied pressure. Its output measurement is processed through a signal conditioning board and made

available as 0 to 5V DC signal.

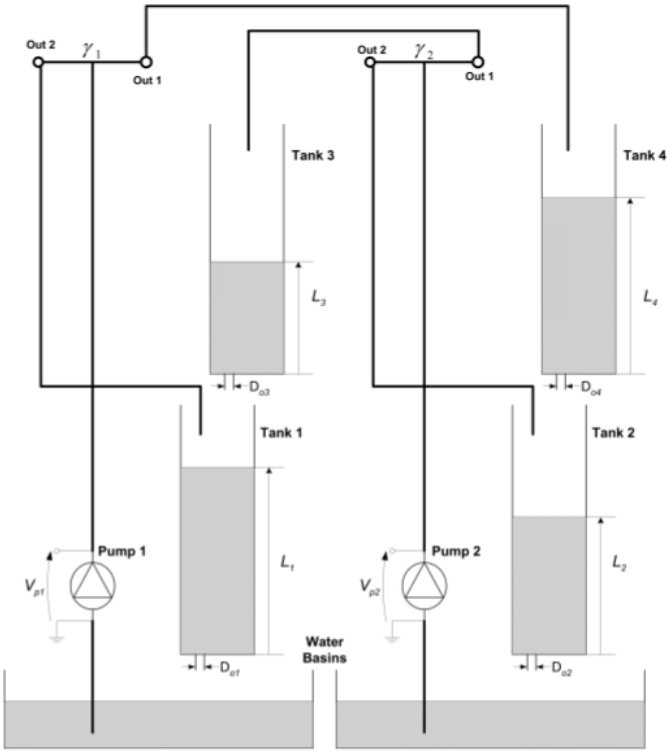

Figure 3.2: Quadruple-Tank System.

3.3 Data acquisition system

The data acquisition card consist of a Quanser Q8 control board. The Q8 is a H.I.L. control board with an extensive range of input and output support . A wide variety of devices with analog and digital sensors as well as quadrature encoders are easily connected to the Q8. This single board solution is ideal for use in control systems and complex measurement applications.

Quanser's Q8 is supported by The MathWorks xPC Target, LabVIEW RTX and RT-LAB.

The main features of this board are,

- 8 x 14-bit programmable analog inputs
- 8x 12-bit D/A voltage outputs
- 8 quadrature encoder inputs
- 32 programmable digital I/O channels
- Simultaneous sampling of both analog and encoder sections
- 2x 32-bit dedicated counter/timers
- 8x 24-bit reconfigurable encoder counter/timers
- 2x on-board PWM outputs
- 32-bit, 33 MHz PCI bus interface
- Supports Quanser real-time control software WinCon (2000/XP)
- Totem Pole digital I/O for high speed
- Easy synchronization of multiple Q8 boards

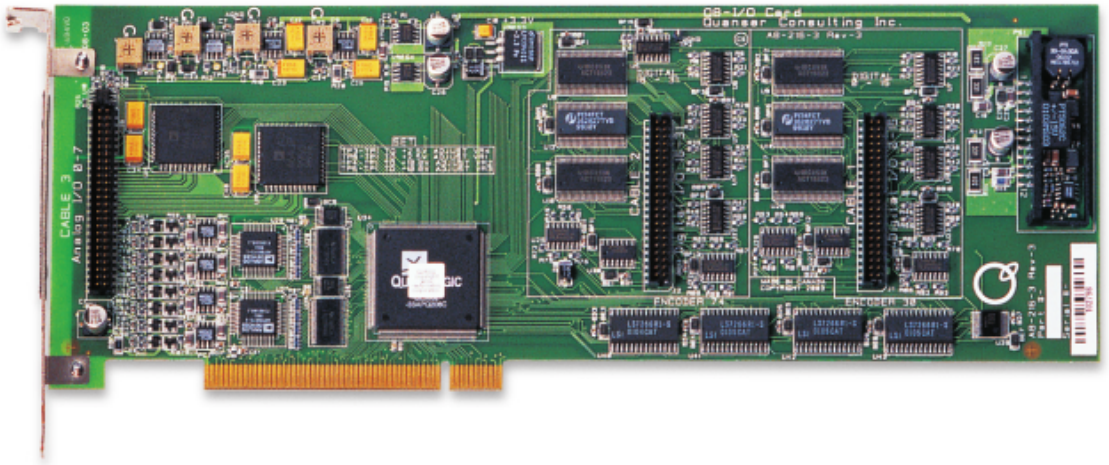

Figure 3.3: Quanser Q8 board.

3.4 Power module

The connection between the DAQ and the plant (sensors and actuators) is done by means of an Universal Power Module UPM-2405 from Quanser, which is a power amplifier that provides the adequates voltages and currents to manage the pumps and to adapt the sensor signals to the board ranges.

Figure 3.4: Quanser Universal Power Module UPM-2405.

3.5 Server PC

There is a PC which acts as a host connected to the system via the Q8 board described in the previous section, and which runs the LabVIEW application that contains the control system implementation, and the JIL server to add connectivity with the client side. Optionally, the user interface can be executed in this PC, but also in another PC with the Java Virtual Machine and with a newtwork connection to the host PC.

Chapter 4

State of the art

In this chapter we introduce several concepts related to multivariable systems and control theory, which are essential to understand the motivations and the development of the present work.

4.1 Event-based control

As opposed to periodic control systems, event-based controllers are not driven by time. Instead of having a constant sampling time, the controller only actuates when a change of the system state is detected. So an event-based controller acts when something interesting happens from a control point of view.

Although there are different approaches in the literature to develop event-based control strategies, frequently two categories are considered. In the first one, the feedback control action is computed when the process is outside a certain detection region located around the set-point; the controller employed is, in general, a PID controller with variable sampling period; and once the process is inside the detection region, no control actions are produced till the process leaves the region due to a disturbance or a set-point change. In the other group, the feedback control action is constant: it is set to the maximum; but once the process is inside the region, feedforward actions are applied to move the process output asymptotically to the reference value.

Unlike periodic sampling systems, where information is sent every period regardless of the state of the plant, in event based sampling information is only transmitted when something important occurs, which leads to a more efficient use of the transmission medium. When the communication is done by a wireless channel then the data sending scheduling must be optimized to minimize the risk of data packet losses. Moreover, wireless sensor networks usually consist of autonomous nodes (motes) which have a small battery as energy source. For these nodes, the power supply is a crucial issue. Since radio communication is often the most power-consuming activity, a reduction in data transmission is important in order to increase the life-time of the batteries.

We will focus in the event-based algorithm described in [11]. It is based on a periodic PI controller. However, the control action is not updated with a constant period, but when a triggering condition is satisfied. The proportional and integral actions are treated independently, i.e., there is an event condition associated with the proportional action and another one with the integral action.

The control law is that well-known of a PI controller

$$u(t) = u_p(t) + u_i(t) = K_p \cdot e(t) + \frac{1}{T_i} \int_0^t e(t)dt \tag{4.1}$$

where K_p is the proportional gain, and T_i the integral time. The proportional part, $K_p \cdot e\,(t)$, is updated when

$$|e\,(t) - e\,(t_{prev})| > \delta_P \tag{4.2}$$

where δ_P is the event triggering threshold for the proportional part.

And the integral part, $K_i \cdot IE\,(t)$ is updated when

$$|IE\,(t) - IE\,(t_{prev})| > \delta_I \tag{4.3}$$

where δ_I is the threshold for integral events.

For the implementation, we discretize the integral term of (4.1), obtaining the expression,

$$IE(t) = IE(t - T_s) + T_s e(t) \tag{4.4}$$

where IE is the integrated error calculated in the sampling instants. Then, the discretized control law is $u = K_p \cdot e\,(t) + \frac{1}{T_i} \cdot IE\,(t)$. There also exist the possibility of defining a deadband of width ϵ. In that case, the integrated error is not anymore the real error, but a function defined as $e_i = \begin{cases} e & |e| > \epsilon \\ 0 & |e| < \epsilon \end{cases}$.

4.2 MIMO systems

Any process can be represented as a generic block with n inputs and m outputs. When both n and m are greater than one, the process is said to be multivariable or MIMO (multiple input - multiple output).

4.2.1 Relative Gain Array (RGA)

Relative Gain Array (RGA) is an analytical tool used to determine the optimal input-output variable pairings for a multi-input-multi-output (MIMO) system. In other words, the RGA is a normalized form of the gain matrix that describes the impact of each control variable on the output, relative to each control variable's impact on other variables. The process interaction of open-loop and closed-loop control systems are measured for all possible input-output variable pairings. A

ratio of this open-loop 'gain' to this closed-loop 'gain' is determined and the results are displayed in a matrix.

The array will be a matrix with one column for each input variable and one row for each output variable in the MIMO system. This format allows a process engineer to easily compare the relative gains associated with each input-output variable pair, and ultimately to match the input and output variables that have the biggest effect on each other while also minimizing undesired side effects.

To understand the results of the RGA, it is useful to keep in mind that,

- The closer the values in the RGA are to 1 the more decoupled the system is,
- The maximum value in each row of the RGA determines which variables should be coupled or linked, and
- Also each row and each column should sum to 1

The RGA reflects how the inputs and outputs of the system are coupled. For 2x2 systems at steady state it has the form

$$M = \begin{pmatrix} m & 1-m \\ 1-m & m \end{pmatrix}$$

with

$$m = \frac{g_{11}(0) \cdot g_{22}(0)}{g_{11}(0) \cdot g_{22}(0) - g_{12}(0) \cdot g_{21}(0)}.$$

For big values of m, the dominating elements of the transfer function matrix are the diagonal elements. The output 1 is affected mostly by the input 1, and the output 2 by the input 2. If m is small, the output 1 depends mainly on the input 2, and the output 2 on the input 1. The knowledge of which input mainly affects which output is important for the design of (decentralized) PI controllers for the MIMO system.

4.2.2 Decoupling

Compensation or decoupling in multivariable processes is a method based on determining the elements d_{ij} of the compensator $D(s)$ such that the plant composed of this and the process to be controlled $Q(s) = G(s)D(s)$ have less interaction problems between variables than the original process. Depending on the characteristics of the plant and the compensator, we can have a *perfect decoupling* or a *partial decoupling*.

Perfect decoupling is obtained when, for a given MIMO system $G(S)$, a matrix $D(s)$ is found such that the compensator with the process, denoted by $Q(s)$, is a diagonal matrix

$$Q(s) = \begin{pmatrix} q_1(s) & 0 \\ 0 & q_2(s) \end{pmatrix}. \tag{4.5}$$

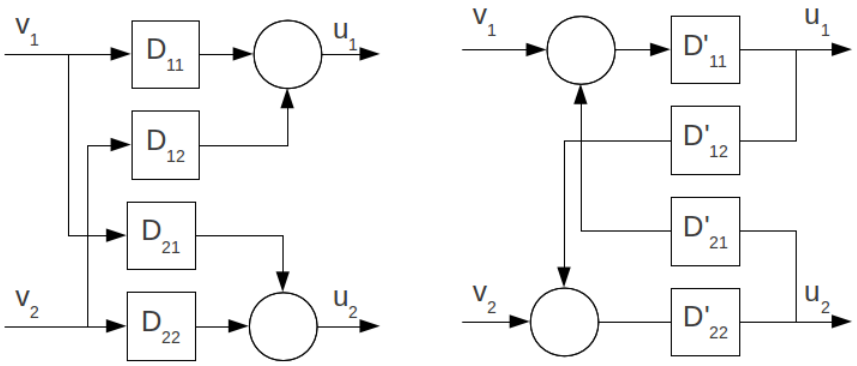

Figure 4.1: Direct decoupling scheme (left) and inverse decoupling scheme(right).

Partial decoupling is when $D(s)$ is found such that $Q(s)$ is a lower or upper triangular matrix. In both cases the interaction is only in one way and the RGA matrix of the apparent process $Q(s)$ is the identity.

$$Q_{utm}(s) = \begin{pmatrix} q_{11}(s) & q_{12}(s) \\ 0 & q_{22}(s) \end{pmatrix}, Q_{ltm}(s) = \begin{pmatrix} q_{11}(s) & 0 \\ q_{21}(s) & q_{22}(s) \end{pmatrix}. \tag{4.6}$$

The decoupling net $D(s)$ can be implemented in several ways by means of SISO blocks. The more common form is represented in Figure 4.1(left) for 2x2 systems, where the matricial structure is reproduced with fidelity. Each block is asociated to an element of the D(s) matrix. This method is known as direct decoupling since the control signals go directly through the blocks D_{ij} to the process.

However, other authors have proposed the use of a compensation with a feedback structrure known as inverse decoupling. Figure 4.1(right) shows a configuration for the 2x2 case. In this kind of net there are n blocks that form a direct path between the new control signals u_{ci} and the process inputs u_i, while the other blocks feed back positively the signals u_i. In this case there not exist a direct correspondency between the blocks D'_{ij} and the matrix elements as it occurs in the direct case.

Then the control strategy can be composed of two parts, namely, a decoupling net which will allow us to see the TITO system as two independent SISO systems, and a set of independent controllers between each output and the input chosen to control it. In the best case, the system can be considered as n non-interacting SISO subsystem, each one with its corresponding controller.

Chapter 5

Control Problem

From the point of view of control theory, the quadruple tank plant is a MIMO system with two inputs (the pumps flow), and four outputs (the tanks water level).

Though in principle there are four outputs, we are more interested in the control of the levels of the lower tanks. The reason is that if we choose as output the upper tanks and choose the correct input-output pairing, we have two independent SISO system for which there are well-known techniques to control. On the other hand, if we choose the lower tanks levels as outputs the system can have a rich dynamic behaviour.

5.1 Plant Model

A mathematical model for the plant can be derived from mass balances and Bernouilli's law.

Mass balance gives for each of the four tanks

$$\dot{V} = A \cdot \dot{h} = q_{in} - q_{out} \tag{5.1}$$

where V is the volume of water in the tank, A is the cross-section area of the tank, h is the water level, q_{in} the inflow and q_{out} the outflow.

By evaluating Bernouilli's law for incompressible liquids

$$\rho + \frac{1}{2}\rho v_w^2 + \rho g h = const. \tag{5.2}$$

at the water surface ($v_w = 0$) and at the bottom of each tank ($h = 0$) and substracting the resulting equations from each other, we obtain for the outflow

$$q_{out} = a \cdot v_w = a\sqrt{2g}\sqrt{h} \tag{5.3}$$

where a is the cross-section area of an outlet, v_w is the speed of water at the outflow, and g is the acceleration due to the gravity.

The previous expressions applied to the system yields the following differential equations

$$\frac{dh_1}{dt} = -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{A_1}v_1 \tag{5.4}$$

$$\frac{dh_2}{dt} = -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2 k_2}{A_2}v_2 \tag{5.5}$$

$$\frac{dh_3}{dt} = -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)k_2}{A_3}v_2 \tag{5.6}$$

$$\frac{dh_4}{dt} = -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)k_1}{A_4}v_1 \tag{5.7}$$

where A_i is the cross-section of Tank i, a_i is the cross-section of the outlet hole, and h_i the water level. The voltage applied to the Pump i is v_i and the corresponding flow $k_i v_i$. The parameters γ_1 and γ_2 depend on the configuration of the valves. The flow to Tank 1 is $\gamma_1 k_1 v_1$ and the flow to Tank 1 is $(1-\gamma_1)k_1 v_1$, and similarly for Tank 2 and Tank 3. The acceleration of gravity is denoted by g.

5.1.1 Linearization of the model

The mathematical model can be linearized around an operating point. Defining the variables $x_i = h_i - h_i^0$ and $u_i = v_i - v_i^0$, where h_i^0 and v_i^0 are respectively the steady state tank level and input flow corresponding to the operating point. Then the linear system can be represented in state space as

$$\frac{dx}{dt} = \begin{pmatrix} -\frac{1}{T_1} & 0 & \frac{A_3}{A_1 T_3} & 0 \\ 0 & -\frac{1}{T_2} & 0 & \frac{A_4}{A_2 T_4} \\ 0 & 0 & -\frac{1}{T_3} & 0 \\ 0 & 0 & 0 & -\frac{1}{T_4} \end{pmatrix} x + \begin{pmatrix} -\frac{1}{T_1} & 0 \\ 0 & \frac{\gamma_2 k_2}{A_2} \\ 0 & \frac{(1-\gamma_2)k_2}{A_3} \\ \frac{(1-\gamma_1)k_1}{A_4} & 0 \end{pmatrix} u \tag{5.8}$$

$$y = \begin{pmatrix} k_c & 0 & 0 & 0 \\ 0 & k_c & 0 & 0 \end{pmatrix} x. \tag{5.9}$$

The transfer functions matrix is

$$G(s) = \begin{pmatrix} \frac{\gamma_1 c_1}{1+sT_1} & \frac{(1-\gamma_2)c_1}{(1+sT_3)(1+sT_1)} \\ \frac{(1-\gamma_1)c_2}{(1+sT_4)(1+sT_2)} & \frac{\gamma_2 c_2}{1+sT_2} \end{pmatrix} \tag{5.10}$$

where $c_i = \frac{T_i K_i K_c}{A_i}$ and $T_i = \frac{A_i}{a_i}\sqrt{\frac{2h_i^0}{g}}$.

5.1.2 Minimum and Non-minimum phase

One interesting property of the four tanks system, from the academic point of view, is that the multivariable system can be of minimum or non-minimum phase depending on the configuration of the distribution valves (γ_1, $gamma_2$). Thus the system can be used for a wide variety of experiments, for example to illustrate the student the difficulty to control a nonmimimum phase.

As explained in [4], the zeros of the transfer matrix are the zeros of the numerator polynomial of the rational function

$$detG(s) = \frac{c_1 c_2}{\gamma_1 \gamma_2 \prod_{i=1}^{4}(1+sT_i)} \times \left[(1+sT_3)(1+T_4) - \frac{(1-\gamma_1)(1-\gamma_2)}{\gamma_1 \gamma_2}\right]. \tag{5.11}$$

This means that the matrix G has two finite zeros for $\gamma_1, \gamma_2 \in (0,1)$. One of them is always in the left half plane, but the location of the second can be either in the left or the right half-plane. In particular, it can be showed that the system is minimum phase for

$$1 \leq \gamma_1 + \gamma_2 \leq 2, \tag{5.12}$$

and non-minimum phase for

$$0 \leq \gamma_1 + \gamma_2 \leq 1. \tag{5.13}$$

There exists a straightforward physical interpretation. If the system is minimum phase ($\gamma_1 + \gamma_2 > 1$), then the flow to the lower tanks is greater than the sum of the flows of the upper tanks, and the system is easier to be be controlled. However, if the system is non-minimum phase ($\gamma_1 + \gamma_2 \leq 1$), the flow to the lower tanks is smaller than the sum of the flow to the upper tanks, and in this case the control of the plant is much more difficult.

5.1.3 Relative Gain Array

We can calculate the RGA from the linearized model (5.10), obtaning the following expression

$$\lambda_{11} = \frac{\gamma_1 c_1 \gamma_2 c_2}{\gamma_1 c_1 \gamma_2 c_2 - (1-\gamma_2) c_1 (1-\gamma_1) c_2} = \frac{\gamma_1 \gamma_2}{\gamma_1 + \gamma_2 - 1}. \tag{5.14}$$

This means that for $\gamma_1 = \gamma_2 = 1$ we have a completely decoupled process, or partially decoupled if only one of (γ_1, γ_2) is equal to one. For the intermediate cases, the RGA can take positive values greater than 1 or negative values. This latter case is the worst in terms of interaction between the variables.

Chapter 6

Implementation

In this chapter the sytem design and implementation are explained in detail. First we present the architecture of the system, which is divided functionally into several subsystems. Then we explain some lower level implementation aspects that cope with problems that appeared during the development phase.

6.1 Architecture

As mentioned before, it is desired to give the system the possibility to have phisically separated the control system and the user interface for configuration and/or monitoring. Keeping this idea in mind, the system has been designed to have a three-tiered architecture, where the server side lies on the PC directly connected to the pumps and sensor, and the client side can be running in the same or in another PC. The communication between client and server is done through the application JIL Server (see below). This architecture is represented more clearly in Figure (6.1).

The functions which are implemented in the LabVIEW application are divided into several areas, namely:

- *Data Acquisition* The communication with the acquisition card, in order to read the level sensors and to send the control action to the pumps.

- *Controller* In this area are included the implementation of the controller and all the components needed for switching between the different modes of the control system.

- *Data Logging* This subsystem contains the blocks which perform the writing to disk of the data logged from the system.

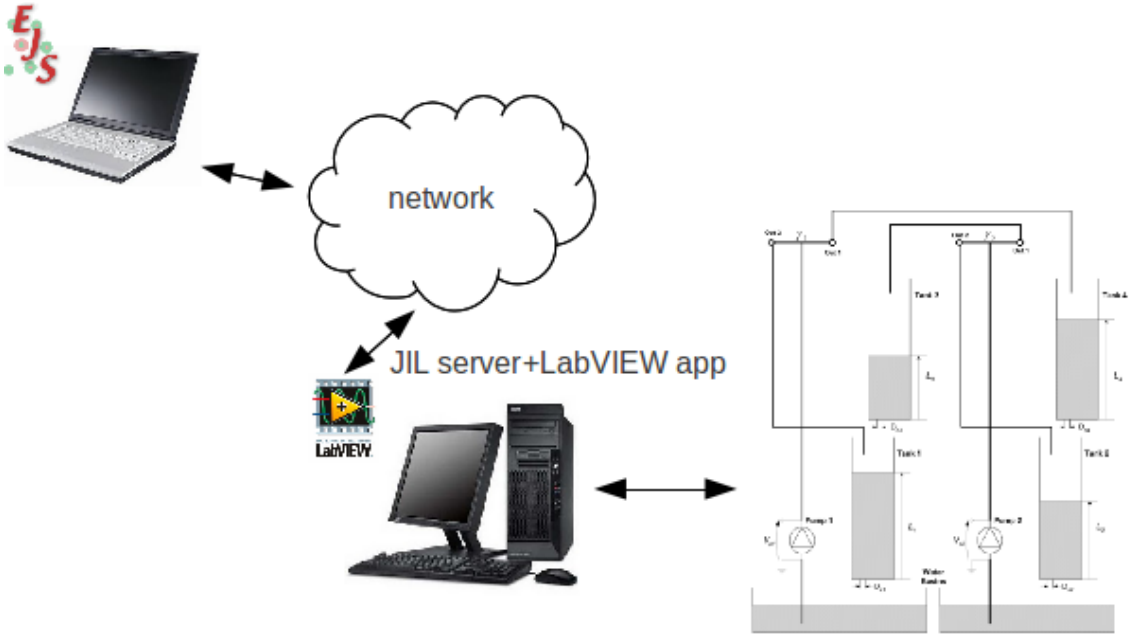

Figure 6.1: Architecture of the system.

6.2 JIL Server

The TCP/IP communication between Java applets and LabVIEW can be implemented directly with the Java API and the LabVIEW communications library. Though it can be very effective, this approach has the inconvenient that the applications developed frequently are only known by the programmer, which degrades the maintenance and scalability of the system.

This mechanism is based on a middleware layer composed of a server, named JIL Server (Java Internet LabVIEW Server), which is ubicated between the Java applets and the LabVIEW applications that performs the local real-time control in the laboratory.

6.3 Communication between EJS and LabVIEW

JIL server acts as an interface between EJS and LabVIEW. It provides an easy way to send and receive the values of variables directly without entering into low level details. JIL provides a Java API to define in EJS variables associated to the controls and indicators of LabVIEW Virtual Instruments (VIs), and to control the load and the execution of this VIs.

The variables exchanged between the LabVIEW and EJS applications are listed below. There are two group of variables, the first ones are used to send values from EJS to LabVIEW (see Table 6.1), and the second ones to send data from LabVIEW to EJS (see Table 6.2).

Variable	Type	Description
stop	boolean	Controls the execution of the LabVIEW application
clear	boolean	Restarts the application after an emergency stop
Kp1	double	Proportional gain of the first controller
Ti1	double	Integral gain of the first controller
Td1	double	Derivative gain of the first controller
deltaP1	double	Proportional threshold of the first controller
deltaI1	double	Integral threshold of the first controller
deltaD1	double	Derivative threshold of the first controller
Kp2	double	Proportional gain of the first controller
Ti2	double	Integral gain of the first controller
Td2	double	Derivative gain of the first controller
deltaP2	double	Proportional threshold of the first controller
deltaI2	double	Integral threshold of the first controller
deltaD2	double	Derivative threshold of the first controller
qPump1	double	Control action of the first pump
qPump2	double	Control action of the second pump
sp1	double	Setpoint of the first controller
sp2	double	Setpoint of the second controller
pumpControl	int	Control mode
switchPumps	int	Switch the pumps
decoupling	int	Decoupling mode
offset1	double	Offset of the first pump
offset2	double	Offset of the second pump
decString	String	Configures the decoupling matrix

Table 6.1: Values sent from EJS to LabVIEW.

Variable	Type	Description
tankLevel1	double	Tank 1 water level
tankLevel2	double	Tank 2 water level
tankLevel3	double	Tank 3 water level
tankLevel4	double	Tank 4 water level
time	double	Time measured in the server
qPump1PID	double	Control action sended to the first pump
qPump2PID	double	Control action sended to the second pump
qPump1PIDNotDec	double	Control action before decoupling
qPump2PIDNotDec	double	Control action before decoupling
PeventPID1	boolean	Proportional event in first controller
IeventPID1	boolean	Integral event in first controller
DeventPID1	boolean	Derivative event in first controller
PeventPID2	boolean	Proportional event in first controller
IeventPID2	boolean	Integral event in first controller
DeventPID2	boolean	Derivative event in first controller

Table 6.2: Values sent from LabVIEW to EJS.

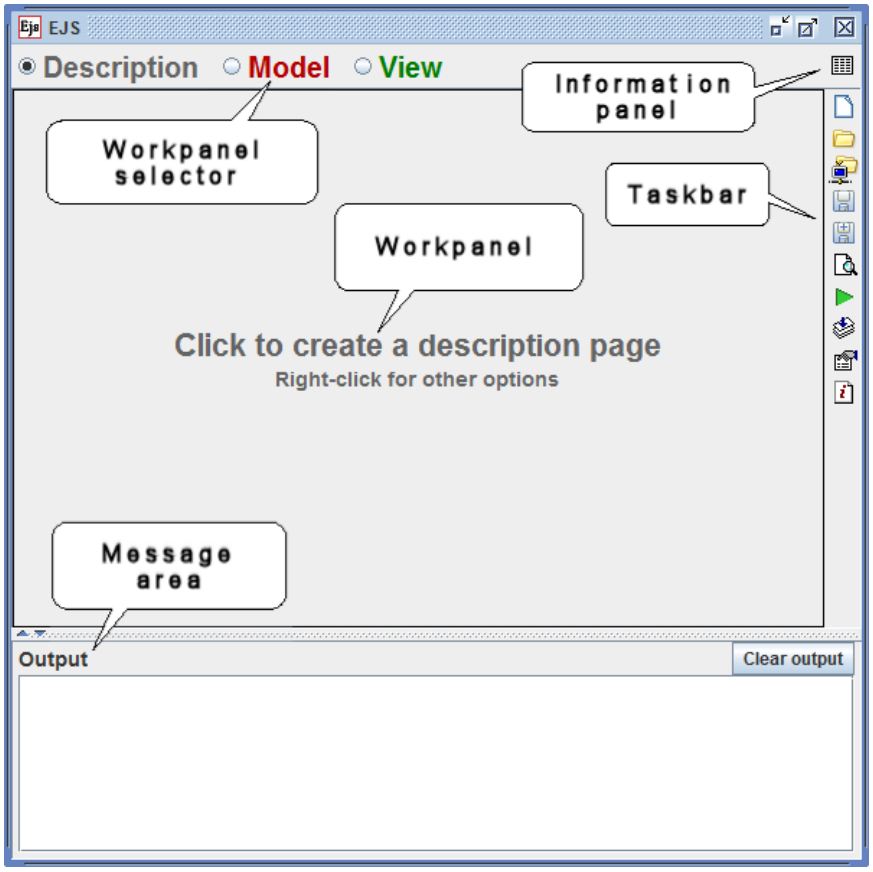

Figure 6.2: The graphical user interface of EJS.

6.4 EJS

Easy Java Simulations(EJS) is an open source (and therefore completely free) software tool designed to create simulations in Java with high-level graphical capabilities and with an increased degree of interactivity. The tool provides its own mechanism for describing models of scientific and control engineering phenomena, and, as such, can be used to create virtual laboratories on its own. Figure 6.2 shows the EJS user interface.

EJS is different from most other authoring tools in that it is not designed to make life easier for professional programmers, but has been conceived for science students and teachers. That is, for people who are more interested in the content of the simulation, the simulated phenomenon itself, and much less in the technical aspects needed to build the simulation.

The tool structures a simulation in two main panels, the **Model** and the **View**. Apart from the Model and the View, there is also an introductory part, named **Description**, to describe (using HTML files) the system to be simulated.

The Model describes the simulated system by means of variables (both state variables and parameters), that completely characterize the system, and of computer algorithms that state how

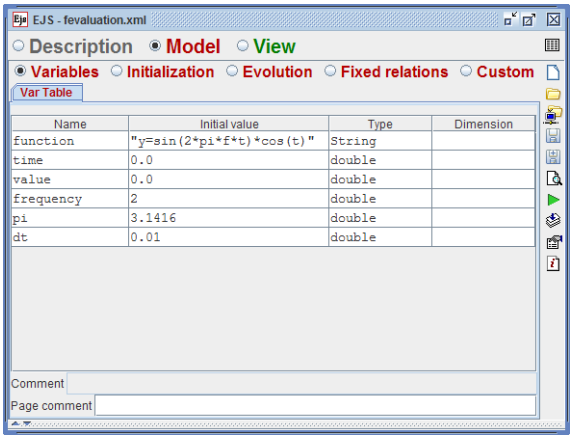

Figure 6.3: Subpanel Variables of EJS.

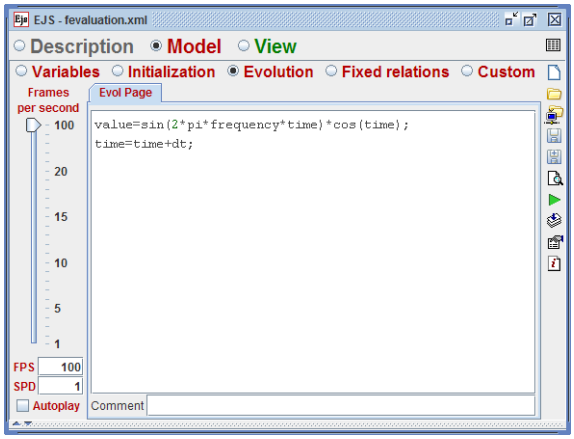

Figure 6.4: Subpanel Evolution of EJS.

the system evolves in time and how it responds to user interaction. Authors need to declare the variables using a simple table, and write the Java code needed to specify the algorithms. EJS offers specialized help to solve models based on ordinary differential equations (ODEs) by providing an editor to write these equations and automatically generating the code required using the most popular solvers.

The Model is divided in five subpanels: `Variables`, `Initialization`, `Evolution`, `Fixed relations` and `Custom`. In the Variables subpanel the global variables of the simulation are declared. The Initialization subpanel allows authors to execute code of initialization before stepping the simulation. In the Evolution subpanel authors can input two type of descriptions: pure Java code or ordinary differential equations by using the ODEs editor. Both types of descriptions are evaluated continuously while the simulation is performed. The Fixed relations subpanel provides an additional way to execute Java code when the user interacts with the view while the simulation is paused. The Custom subpanel can be used by authors to implement their own Java methods. Figure 6.3 and Figure 6.4 show the Variables and Evolution subpanels respectively.

The View provides the visualization of the simulated system, either in a realistic form or using one or several data graphs, and the user interface elements required for user interaction. These **view elements** can be chosen from a set of predefined, ready-to-use components, to build a tree-like structure in a kind of block construction game for the view. There are elements of several types. Each type specializes in a given visualization or interaction task, but can also be customized using the so-called **properties**, a set of internal values that modify the aspect and behaviour of the element on the screen. This way, the job of the author when building the view consists in choosing the right elements from those offered and in customizing them for the interaction desired for the simulation.

Both, model and view need to be interconnected. Any change in the model state must be immediately reflected by the view in order to keep a dynamic, on-the-fly visualization of the system.

In turn, any interaction of the user with the view must immediately affect the model so that the desired interactivity is achieved. This communication is based on connecting model variables and view elements properties. This connection is very easily established by typing, in the table of properties of view elements, the names of the model variables to be connected to the properties. Once the model and the view have been created and the required connections established, EJS creates the ready-to run simulation at a single mouse click, taking care of a good number of technical issues that thus becomes completely transparent to the author. The result is an independent, high performance, interactive simulation which can either be run as a stand-alone Java program, or be embedded as an applet in an HTML page. More description about using Easy Java Simulation, some examples, and the software can be obtained from *http://www.um.es/fem/Ejs/*.

6.4.1 Graphical User Interface

The design of the User Interface has been made considering the information about the state of the plant that is more interesting for the user, i.e.,

- Water levels
- Flow of the pumps
- Events in the controller

and also the configuration of the control system,

- Connection control
- Automatic (controller) or manual (user) mode
- PID parameters (gain and thresholds)

The design option finally adopted has been to divide the application window into three different areas (see Figure 6.5). At the upper left part of the window there is an interactive graphical representation of the four tank system, with the double purpose of presenting to the user the state of the plant and to change the setpoint of the water levels for the tanks in the bottom. The right half of the window is entirely dedicated to show all the states of interest of the plant and the controllers, by means of several plots. The plot on the top shows the water level of the four tanks, together with the two set points. Below that, the plots are divided into the left half, with information about the first controller, and the right half corresponding to the second controller. For each controller there is one plot showing the control action and the event times (when the control is updated). Finally, at the bottom left there are controls which allows the user to configure the controller (gains, event thresholds, decoupling, etc.) and also to perform the conection with the server.

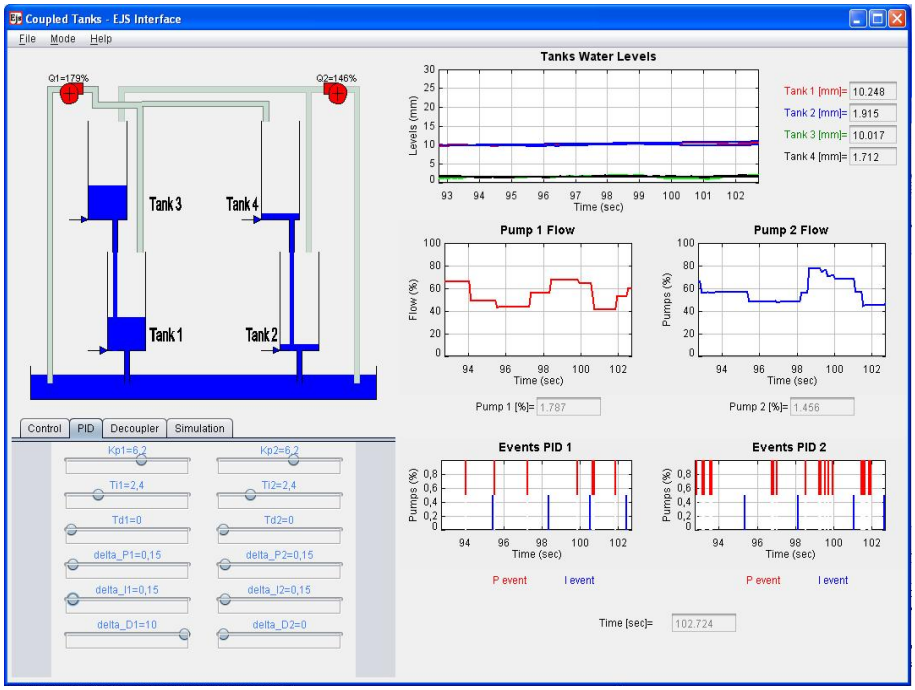

Figure 6.5: EJS Four Tanks Interface.

6.5 Main loop

The main loop of the EJS application, which is continuosly in execution, is defined within an Evolution page, and it is divided into three tasks, namely,

- Receive the state of the plant and other data from the server.
- Send the configuration commands to the server
- To write this data to disk

The code included in the Evolution page is listed below,

```
if(((jil.Jil)vi).isConnected()) {
  PeventPID1 = false;
  IeventPID1 = false;
  PeventPID2 = false;
  IeventPID2 = false;

// Get the state of the plant
  buffer = ((jil.Jil)vi).getDataAvailable();
  if(((jil.Jil)vi).isRunning()) {
    if (buffer > 0) {
      getValues();
    }
  }

// Send the control action
```

```
  setValues();
  decString = "";

// log the state
  if(lastTime < time) {
    lastTime = time;
    exp_logstate();
  }
}
```

6.6 Control modes

The system admits the selection of four control modes, namely,

- *Pump 1 Manual - Pump 2 Manual.* The two pumps are directly controlled from the EJS side.
- *Pump 1 Manual - Pump 2 Automatic.* The control signal is directly sended to the first pump from the EJS side, but the pump 2 is controlled by the PI.
- *Pump 1 Automatic - Pump 2 Manual.* The opposite to the previous case.
- *Pump 1 Automatic - Pump 2 Automatic.* The two PI controllers are active, and the EJS side only acts as a monitor.

These modes allows the user to have more flexibility in the interaction with the plant. For example, the commonest case corresponds to the automatic mode for the two tanks, where one can do several experiments with the built-in PI controllers. But thinking in a more demanding user, the manual mode can be used to obtain total control of the signal sent to the actuator, and then one can perform a wide range of experiments over the plant, such as identification of the tanks or even the substitution of the controllers with another control law, which in addition can be tested over the network.

6.7 Switching the pumps

In addition to the modes explained before, the application provides the possibility to switch the pumps, i.e., to send the control signal of the first controller to the second pump and viceversa. In the normal mode the Tank 1 is controlled by the Pump 1 and the Tank 2 by the Pump 2, and in the switched mode the pairing is Pump 1 - Tank 2 and Pump 2 - Tank 1. It must be noted at this point that the effect of switching the pumps can be also obtained by using a decoupling matrix or, from another point of view, when the decoupling matrix is being used the pumps-switch configuration also affects to the system.

6.8 Decoupler

The system admits three modes of decoupling, namely,

- *Not decoupled.* The control signal is forwarded directly to the pumps. It also can be viewed as if the decoupling matrix is set to the identity.
- *Direct decoupling.* The system is decoupled by using a direct scheme, defined by the transfer functions of the elements D_{ij}.
- *Inverse decoupling.* An inverse scheme defined by the transfer functions of the elements D_{ij} is used to decouple the system.

Due to the fact that JIL Server can not work directly with complex types such as clusters of arrays, it is needed to find another way of sending the decoupling matrix. The method used in our application is to convert the decoupling matrix into an XML string which follows the specification of the LabVIEW XML type definition. Then the string is sended via JIL and unflattened in LabVIEW to rebuild the data as a cluster.

6.9 LabVIEW implementation

6.9.1 Data acquisition

The LabVIEW top level VI is divided into three parts, *initialization code*, which contains all the code needed for the application to have a valid state, *main loop* where the control loop is implemented, and *finalization code* with all the code needed to free the resources used by the application. Though it is not actually needed, because of the data dependency forces the correct execution order of the code, a *sequenced structure* is used to emphasize the sequence of the code.

Initialization code

This first tab of the sequence contains all the code needed to initialize the different subsystems of the VI. The most important actions are two, the creation of a file to register the session data, and the connection with the driver of the data acquisition card.

Main Loop

The main loop is the core of the application. It is a periodic loop which performs the fast sampling, with a configurable sampling time (with a predefined value of $T_s = 100ms$). In each iteration of the loop the actions listed below are performed, in the same order of appearance,

1. The water levels of the tanks are read by doing a call to the DAQ system.

2. The controller updates the control action if there is an event.
3. Depending on the decoupler configuration, the control action:
   (a) is not modified,
   (b) is modified by the direct decoupler,
   (c) is modified by the inverse decoupler.
4. The watchdog verifies the safetyness of control actions.
5. The control action is sended to the pump.

Finalization code

The finalization code comprises all the actions needed to free resources and and to leave the system ready to perform either another experiments with the plant or the execution of any other application. The most important actions are to close the disk file and the connection with the card.

6.9.2 Writing data to disk

The data logging loop is a low priority loop executed in parallel with the main control loop. This is done so to avoid conflicts between both operations, control and log. Since the former is evidently more important, it should not occur that the control loop had to wait for the execution of the I/O operation. The opposite is not a serious problem, because in the worst case it can lead to the losing of data. However, considering that in practice EJS does write the same data to disk it is very odd to have such a problem.

Concerning to the format of the data, the values are written in a plain text file that can be directly opened in MATLAB or easily imported in a spreadsheet software like *OpenOffice.org Calc*. The values are stored as numbers with a precision of 4 decimal digits, using the dot as decimal point. Each row contains several fields separated with a white space, and the return of line as as separator between rows. The fields contained in a row are those listed in 6.3.

6.9.3 Real plant mode

In the real plant mode, the application reads the level sensors and sends the pumps voltage one time per each iteration of the main loop. These actions are done by calling the HIL (Hardware-In-the-Loop) VIs provided by the Quanser LabVIEW driver.

Once the sensors are read, the voltages values are transformed into water levels given in milimeters. For this operation, and as mentioned in (referencia), the sensors are supposed to be linear and thus the conversion is done with a linear gain and an offset (which is required to adjust the

column	description
1-4	Tank levels
5-6	Set points
7-8	Control signal (before the decoupler)
9-10	Control signal (after the decoupler)
11-12	P and I Events in the first controller
13-14	P and I Events in the second controller

...

14.931 14.878 2.480 2.261 15.000 15.000 9.378 9.113 0.074 -0.001 1.000 1.000 1.000 1.000
14.939 14.910 2.461 2.219 15.000 15.000 9.331 9.259 0.042 -0.023 1.000 1.000 1.000 1.000
14.902 14.961 2.416 2.282 15.000 15.000 9.216 9.236 -0.246 0.152 1.000 1.000 1.000 1.000

...

Table 6.3: Format of the data file (top) and sample lines extracted from a data file (bottom).

zero level). It is also desirable to reduce the measure noise in high frequencies, and for this reason a first-order low-pass filter have been added to each measure. The time-constant have been adjusted to filter the noise but avoiding to affect the frequencies of interest for the control of the system.

6.9.4 Watchdog

Due to physical limitations on the actuators (pumps), the control signal cannot take an arbitrary value, but have a minimum value at which the pumps stop, and also a maximum value over which the pump cannot have more power. In spite of the fact that the pumps can be saturated, demanding an excessive power to the pumps can even led to a pump fail.

The other problematic aspect that must be accounted for is that the tanks are not closed, and thus it is possible to have an overflow is the pump is not controlled properly.

In summary, it can be unsafe to let the user send directly the control actions, without any checks, and by this reason it has been added a block, which is named watchdog (but not in the sense of a watchdog timer used in other context), whose function is precisely to guarantee both that the control action sent to the pumps is within the safety range, and that the water levels never exceed the height of the tanks.

6.9.5 Simulation in LabVIEW

The simulation of the model given by equations (5.4) has been performed in LabVIEW with the Control Design and Simulation Module. This toolkit provides several tools and VIs implementing transfer functions, arithmetics operations, and the basic elements to create a simulation from the mathematic model of the plant.

The architecture of the application remains exactly the same as explained before, with the only difference that the calls to the DAQ subVIs have been substituted by the calls to the simulation

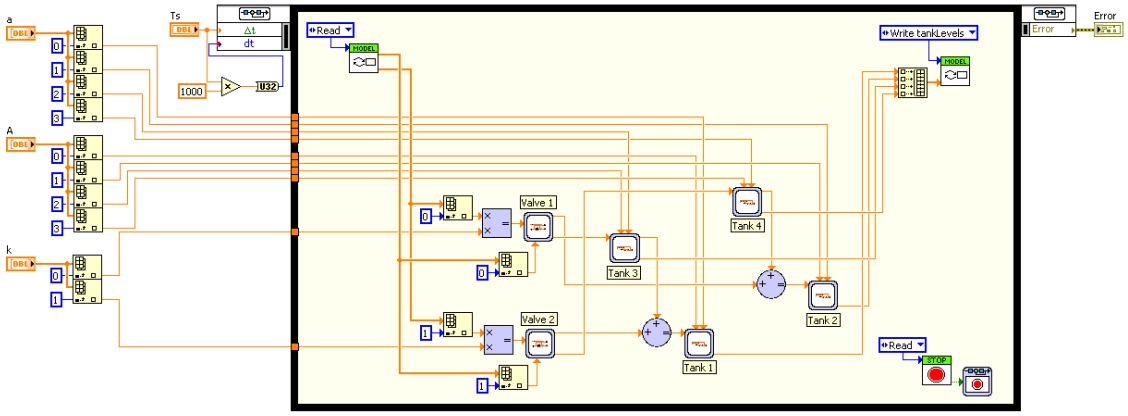

Figure 6.6: LabVIEW Simulation Model. The simulation loop contains the blocks diagram implementing the model of Equation (5.4), which is built by means of the subsystems *Tank model* and *Three-way valve*. The execution of the simulation loop is synchronized with the main control loop.

model. Therefore, the execution of an iteration of the main loop is as follows:

1. The water levels of the tanks are read (simulation values).

2. The controllers are updated.

3. The control signal is decoupled (or not, depending on the configuration).

4. The control action is sent to the simulation model.

As it can be seen, the behaviour of the main loop does not change as well as the variables needed to control the application, and thus the EJS interface still can handle the execution without the need of additional changes.

6.9.6 Model

In order to develop a modular and more readable simulation model, there are two simulation subsystems, one corresponding to a single tank model, and the other which corresponds to a three-way valve.

Pump

The pumps has been modeled as a linear gain which represents the conversion between the input voltage and the pump flow.

Tank model

The tank model is a subsystem with one input representing the total inflow to the tank and two outputs which correspond to the water level and the outflow of the tank, and two parameters representing the tank section and the diameter of the outlet (Equation 5.4).

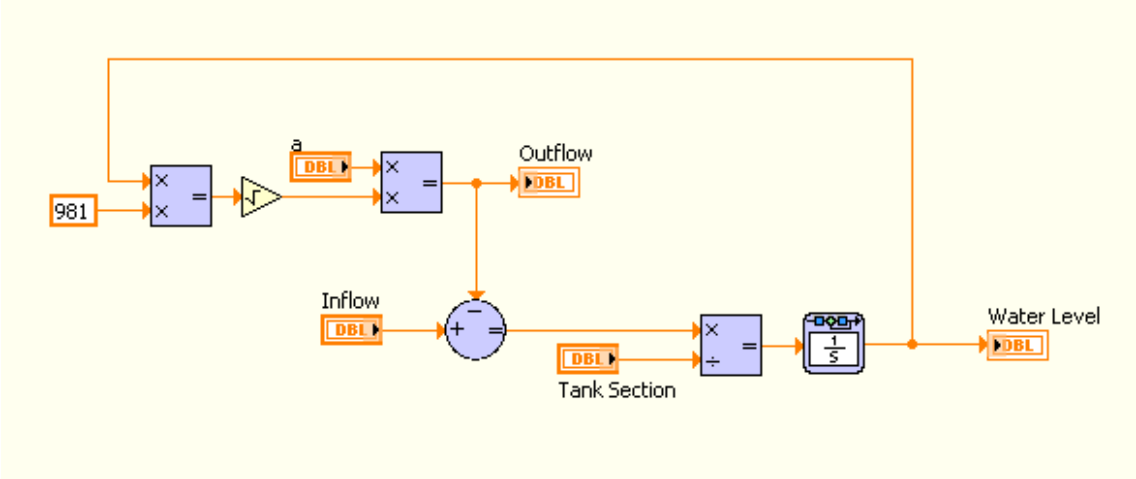

Figure 6.7: Tank Simulation Subsystem. The diagram use an integrator to implement the equation (5.4). The inputs are the parameters of the model, $TankSection$ and $OutletSection$, and the $Inflow$ of the tank. The model yields the $Outflow$ and the $WaterLevel$.

Three-way valve

The three-way valve has been modeled as a subsystem which has two inputs and two outputs. The inputs are the inflow (q_{in}) and a parameter *gamma* (γ) representing the distribution of the inflow between the two outputs. Thus, the value of the first output is γq_{in} and the second output is $(1-\gamma)q_{in}$.

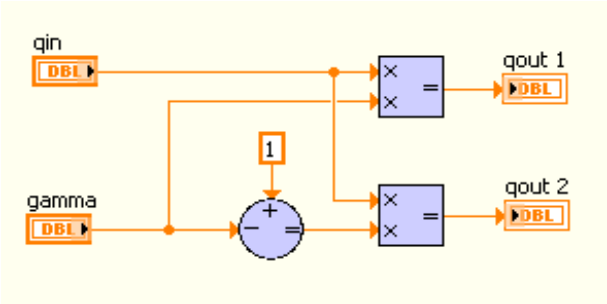

Figure 6.8: Distribution Valve Simulation Subsystem. The block has two inputs, *qin*, the flow entering the valve, and *gamma*, the distribution of the inflow. The output *qout1* is equal to $gamma \cdot qin$, and the output *qout2* is $(1 - gamma) \cdot qin$.

6.9.7 PI Controller

The controller has been implemented within a LabVIEW formula node, which evaluates mathematical formulas and C-like expressions. Therefore the implementation of the control law has been decoupled in a certain way from the other components, in the sense that we can change the controller by updating only this code and the LabVIEW diagram needs not to be modified. Apart from being useful during the development phase, this can be an interesting option for the application to be reused easily in other projects.

Two identical PI controller have been connected to the system closing the loop through the decoupling matrix.

It must be noted that the controller implementation is not purely event-based but uses fast sampling techniques. This means that the events are not detected at the very time the condition becomes true, as it is only checked at sampling times. However, we are admitting that the sampling is done with a frequency very much higher than the greater time constant of the system, so that we can neglect this aspect.

As in [11], a new proportional event is triggered every time the difference between the current and the last error exceed a threshold δ_p. When this occurs, the proportional contribution is updated with the proportional control law $u_p = K_p\delta_p$.

In each sampling time, the absolute error is integrated by using the expression $IE = IE + |e - e_{prev}| \cdot T_s$, being T_s the nominal sampling time of the controller. Then, the integral event is triggered when the absolute integrated error exceed a threshold δ_i. When this occurs, the integral contribution is updated with the control law $u_i = u_i + K_i\delta_i sign(IE)$.

Finally, a deadband can be defined in which the error being integrated is reset to zero, in order to avoid the appearance of limit cycles due to the integral action. The width of the deadband is defined by ϵ. Then the error being integrated is not anymore the real error, but a function defined as $e_i = \begin{cases} e & |e| > \epsilon \\ 0 & |e| < \epsilon \end{cases}$.

Chapter 7

Results

In this chapter we present the results obtained in simulation. Two different configurations have been tested. For the first set of tests the plant has been configured to have minimum phase behaviour, and for the second set it has been configured to have a non-minimum phase system. For each configuration we present the response to a set-point step change in the output variables and the response to disturbances obtained with differents values of the controller parameters.

In order to facilitate the visualization and comparison of the results, first we present a discussion about the results, showing plots that give an overall understanding of the ideas exposed, and we present in the appendices A and B the figures that show in detail the results of the tests.

The parameters of the model used in the simulations have been obtained from the data sheet provided with the plant, and are those listed in Table 7.1.

Parameter	Value	Description
A_i	15.517	Tank section (cm^2)
a_i	0.178	Outlet section (cm^2)
K	3.3	($cm^2/s/V$)

Table 7.1: Values of the parameters of the model used in simulation.

To compare cuantitatively the different controllers we have used as performance indexes,

- *Integrated Absolute Error*, which is defined as $IAE = \int_0^t |e(t)|dt$, and it is used to measure the control performance. The highest the IAE is the more time takes the process to stabilize around the reference. The IAE has been calculated separately in two different contexts,
  - during the set-point following task (IAE_{sp}), and
  - during the disturbances rejection task (IAE_{dr}).

  The IAE has been calculated with a sampling period equal to that of the time based PI controller, $T_s = 0.1s$.

- *Reduction in control updates*, which is calculated as $\eta = 1 - \frac{N}{N_{ref}}$ and it is used as a measure of the control effort, expressed as a percentage of the number of updates performed in a time-based PI with the same nominal period.

7.1 Minimum phase configuration

Choosing $\gamma_1 = \gamma_2 = 0.6$, we have that $\gamma_1 + \gamma_2 = 1.2 \geq 1$ and therefore it is a minimum phase configuration. The operating point has been established in $h_1 = 15cm$ and $h_2 = 15cm$, which gives values for the inputs of the pumps of $v_1 = v_2 = 9.2583$.

Substituting the values of Table 7.1 and the operating point in 5.10, the transfer function matrix for the linearized system is

$$G(s) = \begin{pmatrix} \frac{1.9441}{1+15{,}2368s} & \frac{1.2961}{(1+6.0947s)(1+15.2368s)} \\ \frac{1.2961}{(1+6.0947s)(1+15.2368s)} & \frac{1.9441}{1+15.2368s} \end{pmatrix} \tag{7.1}$$

where $T_1 = T_2 = 15,2368$, $T_3 = T_4 = 6.0947$ and $c_1 = c_2 = 3,2402$, $c_3 = c_4 = 1,2961$.

Calculating the RGA, it can be seen that the correct pairing is between input 1 and output 2, and between input 2 and output 1. Then, the PI controller parameters were tuned by using the software tool TITO [1], to have a phase margin of 60°, which yields values of $K_p = 4,1766$, $T_i = 2,1275$ for both loops.

7.1.1 Decoupling

For the linealized system around the operating point, the decoupling used in the experiments correspond to the simplified case with $\Delta_{11} = \Delta_{22} = 1$, $\Delta_{12} = -\frac{g_{12}}{g_{11}}$ and $\Delta_{21} = -\frac{g_{21}}{g_{22}}$, which gives the following matrix

$$G(s) = \begin{pmatrix} 1 & -\frac{0.1094}{s+0.1641} \\ -\frac{0.1094}{s+0.1641} & 1 \end{pmatrix}. \tag{7.2}$$

The PI parameters for the decoupled system were also be tuned by using the software tool TITO, to have a phase margin of 60°, obtaining values of $K_{p_1} = 4.7646$ and $T_{i_1} = 1.4665$ for both loop.

7.1.2 Setpoint step

With the purpose of illustrating the effect that the decoupling network has in the system, the first tests we present are the responses to a set-point step change for the system with a time based PI controller, both with the decoupling network activated and without it. These results are

[1] TITO is a software tool developed by F.Vázquez (Univ. Córdoba) and F.Morilla (UNED). This software allows to analize and design decoupling networks for TITO systems and tuning of PI controller parameters.

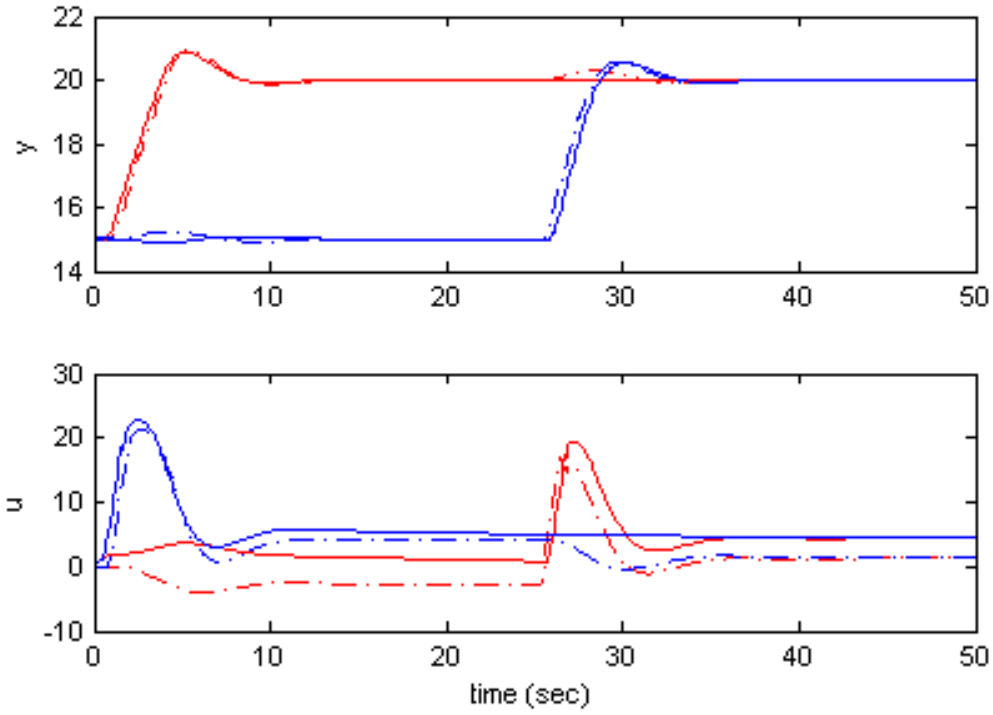

Figure 7.1: Setpoint steps for the plant with time-based PI and without decoupling (dash-dotted line) and with decoupling (continuous line). The outputs are showed in the upper plot and the inputs in the lower plot.

showed in Figure 7.1, and it allow us to verify the performance of the decoupling and to serve as a comparison for the event-based controller. After an initial phase (not showed) where the output is still reaching the operating point, a set-point step change was applied to the first output. Then, when the process have moved to the new set-point, another step is introduced in the second loop.

It can be seen how the decoupling is very effective reducing the interaction between the two loops. This will be an advantage for the event-based PI. As opposed to the previous case, where the set-point change increased the number of events for the two loops, due to the effect of the decoupling there is a reduction in the numbers of events in the loop that is not involved in the set-point change.

For the event-based case, four different set of parameters have been tested, corresponding to $\delta_p = \delta_i = 0.01$, $\delta_p = \delta_i = 0.02$, $\delta_p = \delta_i = 0.05$ and $\delta_p = 0.01,\ \delta_i = 0.05$. To compare between the responses obtained for all the controllers, the complete results are showed in Appendix A. It can be seen that, as it was expected, the performance of the controllers degrades when increasing the event thresholds. It can be observed that the events provoke oscillations around the set point. Thus an excessive value of the thresholds can lead to an important decrease in performance. However, for reasonable values the performance is good and the number of updates in the control action are

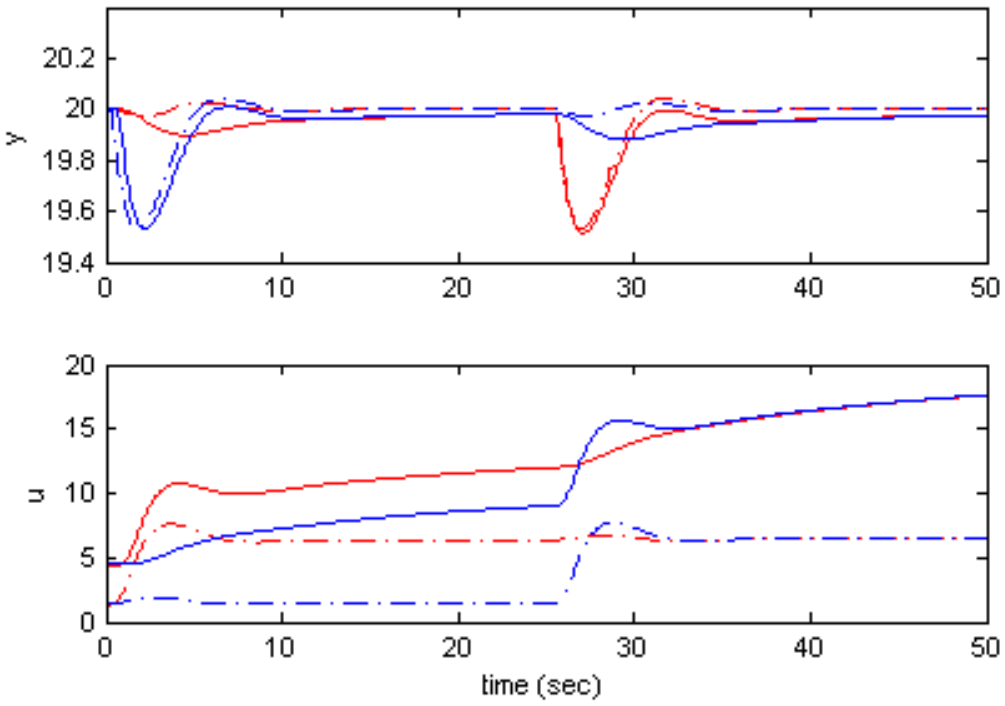

Figure 7.2: Disturbances rejection for the minimum-phase plant and time-based PI controller with decoupling (dash-dotted line) and without decoupling (continuous line). The outputs are showed in the upper plot and the inputs in the lower.

reduced significantly with respect to the periodic PI controller.

7.1.3 Disturbance rejection

In the previous section we have seen that the decoupling provides the possibility of coping with the set-point following task in an independent way for the two outputs. However, one of the disadvantages of using decoupling strategies is that even with a good response to a setpoint, the disturbance rejection is not guaranteed. Thus it is also important to check whether the controller achieves good performance when disturbances are presented in the system. The following results show the responses of the system with the PI controller. The test performed is to introduce a disturbance when the system is in steady state and wait until the process recover its state previous to the disturbance (first in one input, then in the other). The responses to the introduction of disturbances for the periodic PI controller with and without decoupling are showed in Figure 7.2. The results can also be used as a reference to compare with the performance obtained by the event-based PI controllers.

Figure 7.3 shows the response for the event-based controller with $\delta_p = 0.01$ and $\delta_i = 0.05$ without decoupling, and Figure 7.4 with decoupling. The plots show the tank levels, the control

	δ_p	δ_i	IAE_{sp} (y_1/y_2)	IAE_{dr} (y_1/y_2)	η (PI_1/PI_2)
not decoupled	0	0	12.86/10.32	1.58/1.59	0.0/0.0
	0.01	0.01	13.03/10.64	1.84/1.82	0.79/0.80
	0.02	0.02	13.02/11.18	2.04/1.95	0.80/0.83
	0.05	0.05	13.82/11.39	2.94/3.03	0.87/0.87
	0.01	0.05	12.95/10.47	2.04/2.05	0.83/0.84
decoupled	0.0	0.0	11.83/10.44	3.10/3.15	0.0/0.0
	0.01	0.01	11.87/10.58	3.13/3.16	0.85/0.81
	0.02	0.02	11.98/10.48	3.09/3.15	0.87/0.85
	0.05	0.05	12.35/11.24	3.45/3.65	0.89/0.89
	0.01	0.05	11.69/10.42	3.07/3.16	0.85/0.85

Table 7.2: Performances of the controllers with the minimum-phase plant.

signal generated by the PI controllers and the control signal after being modified by the decoupling matrix and, finally, the proportional and integral events, i.e. when there is an update for both PI controllers. The complete results obtained for each experiment are shown in Appendix A.

Finally, the comparison of the performance is showed in Table 7.2. It can be seen that the controller without decoupler obtains a better performance for the disturbances rejection task both in the IAE value as in the reduction of events, while the controllers with decoupler obtain a better performance in the set-point following task.

7.2 Non-minimum phase configuration

Choosing $\gamma_1 = \gamma_2 = 0.4$, we have that $\gamma_1 + \gamma_2 = 0.8 \leq 1$ and therefore it is a non-minimum phase configuration. Substituting the parameters values in (5.10) the transfer function matrix for the linearized system is

$$G(s) = \begin{pmatrix} \frac{1.0583}{1+12{,}4408s} & \frac{1.5874}{(1+7.4645s)(1+12.4408s)} \\ \frac{1.5874}{(1+7.4645s)(1+12.4408s)} & \frac{1.0583}{1+12.4408s} \end{pmatrix} \tag{7.3}$$

where $T_1 = T_2 = 12.4408$, $T_3 = T_4 = 7.4645$ and $c_1 = c_2 = 3.2402$, $c_3 = c_4 = 1.2961$.

From the transfer function matrix (7.3) we can obtain easily the RGA matrix, and looking it we conclude that the correct pairing is input 1 with output 1 and input 2 with output 2.

For the event-based case, three different set of parameters have been tested, corresponding to $\delta_p = \delta_i = 0.01$, $\delta_p = \delta_i = 0.02$ and $\delta_p = \delta_i = 0.05$.

For the linealized system around the operating point, the decoupling used in the experiments corresponds to the simplified case with $\Delta_{11} = \Delta_{22} = 1$, $\Delta_{12} = -\frac{g_{12}}{g_{11}}$ and $\Delta_{21} = -\frac{g_{21}}{g_{22}}$, which gives the following matrix

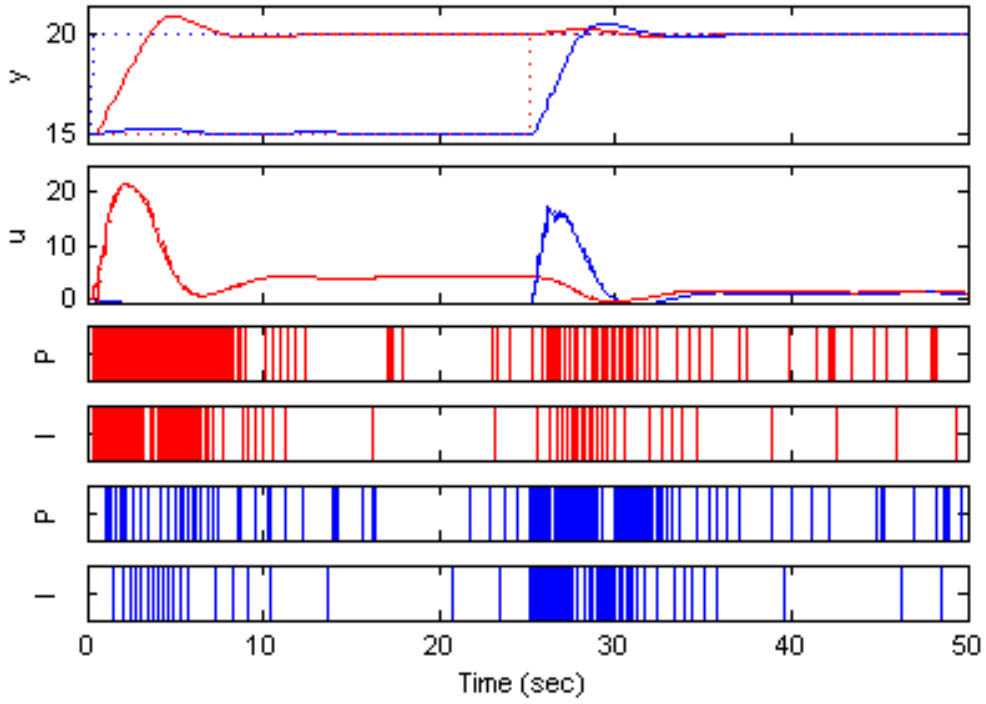

Figure 7.3: Set-point steps for the minimum-phase plant without decoupling, event thresholds $\delta_p = 0.01$, $\delta_i = 0.05$. The first plot shows the two outputs and the setpoints, and the second plot contains the control signals. The other plots show the event times of both controllers, i.e. when the proportional or integral action are updated.

$$G(s) = \begin{pmatrix} 1 & \frac{-0.0072s+0.0109}{s^2+0.2296s+0.0108} \\ \frac{-0.0072s+0.0109}{s^2+0.2296s+0.0108} & 1 \end{pmatrix}. \tag{7.4}$$

As the configuration has non-minimum phase behaviour, the system is much more difficult to control. This can be seen in Figure 7.5, where the time scale is longer than in the previous case. As before, the the PI parameters were tuned with the TITO software tool.

All the results obtained with this configuration are showed in Appendix B. For each test the plots show the water levels in the tanks, the set points, the control action generated by the PI and after the decoupling matrix, and the instant when the events are triggered in the controllers.

7.2.1 Setpoint step

The results are showed in Figure 7.5. As before, when the process is in the operating point, a set-point step change is applied to the first output. Then, when the process has moved to the new set-point, another step is introduced in the second loop.

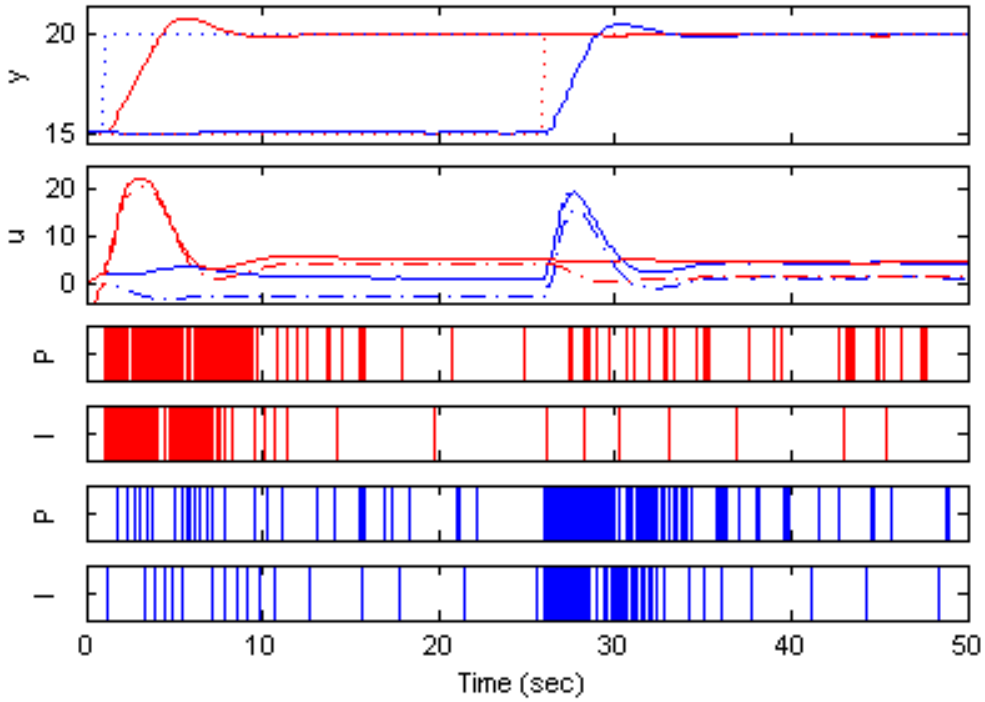

Figure 7.4: Set-point steps for the minimum-phase plant with decoupling, event thresholds $\delta_p = 0.01$, $\delta_i = 0.05$. The first plot shows the two outputs and the setpoints, and the second plot contains the control signals. The other plots show the event times of both controllers, i.e. when the proportional or integral action are updated.

7.2.2 Disturbance rejection

The responses to the introduction of disturbances for the periodic PI controller with and without decoupling are showed in figure 7.6. The results can also be used as a reference to compare with the performance obtained by the event-based PI controllers.

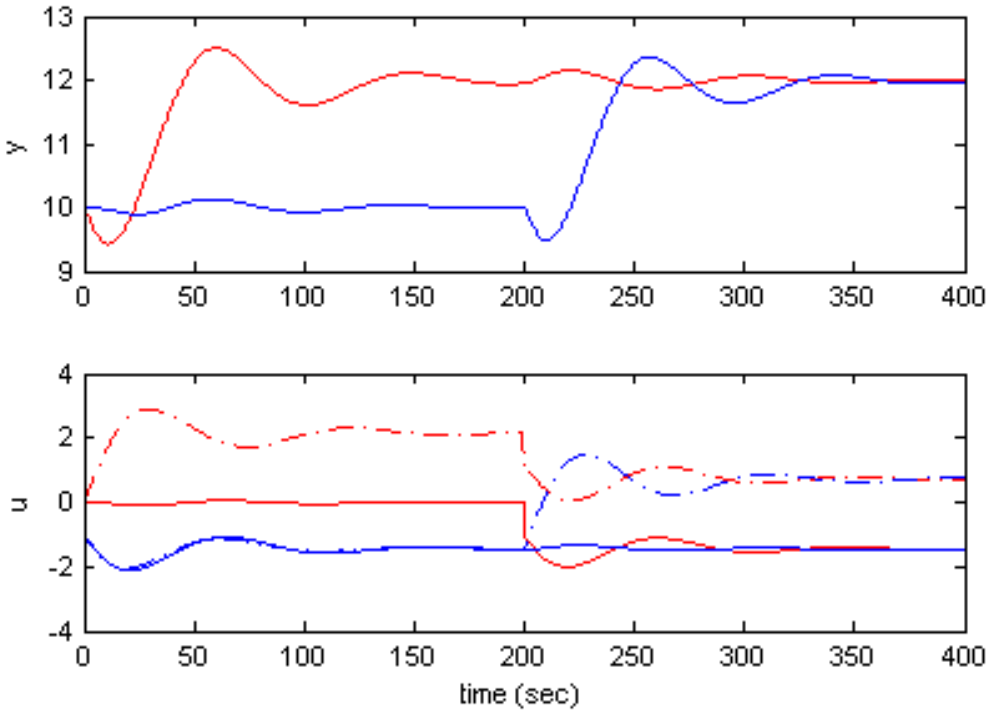

Figure 7.5: Set-point steps for the non minimum-phase plant and time-based PI controller with decoupling. The outputs are showed in the upper plot and the inputs in the lower plot.

Finally, the comparison of the performance is showed in Table 7.3. The better result was obtained by the event-based PI controller with $\delta_p = 0.05$, $\delta_i = 0.15$, which achieved a control performance comparable to the time-based PI, with a number of control action significantly reduced.

	δ_p	δ_i	IAE_{sp} (y_1/y_2)	IAE_{dr} (y_1/y_2)	η (y_1/y_2)
decoupled	0.0	0.0	113.20/100.81	107.69/106.49	0
	0.01	0.05	116.48/104.67	110.55/108.93	0.38/0.40
	0.05	0.05	118.04/109.92	120.65/121.69	0.68/0.68
	0.05	0.15	110.36/105.93	121.56/122.83	0.84/0.84

Table 7.3: Performances of the controllers with the non minimum-phase plant.

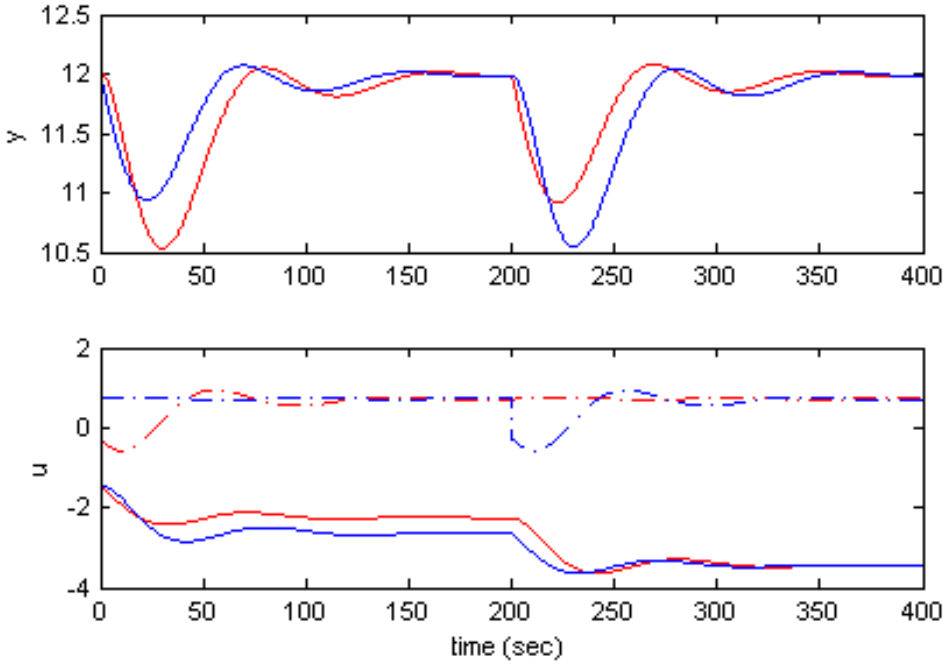

Figure 7.6: Disturbances rejection for the non minimum-phase plant and time-based PI controller with decoupling. The outputs are showed in the upper plot and the inputs in the lower plot.

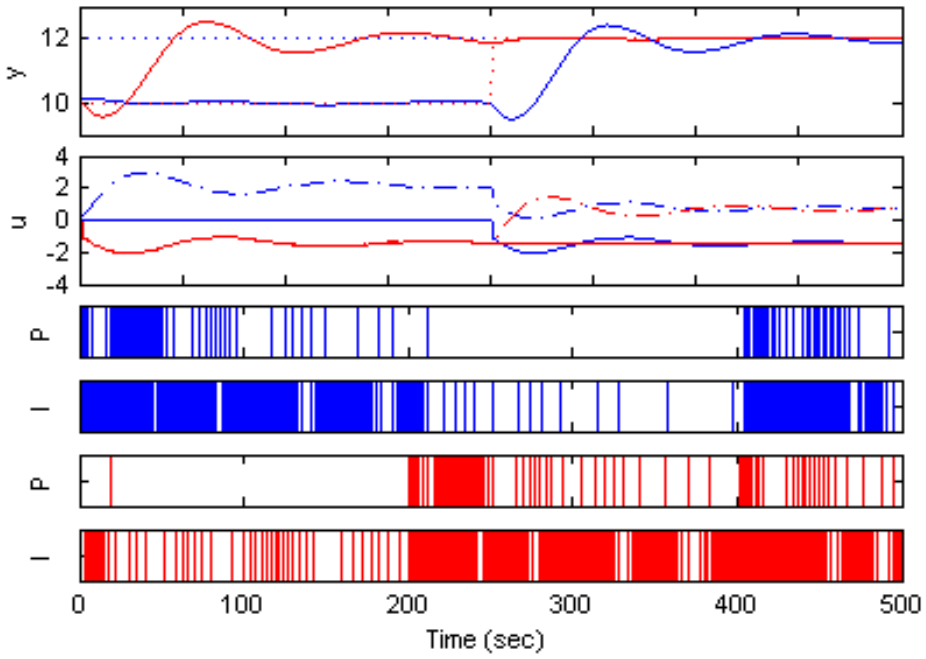

Figure 7.7: Setpoint steps for the non-minimum phase plant with decoupling, event thresholds $\delta_p = 0.01$, $\delta_i = 0.05$.

Chapter 8

Conclusions

As the main source of satisfaction, it is remarkable the success in the development of the experimental framework for testing new control methods based on events, which was one of the target fixed at the begining of this project.

To do this succesfully, we had to overcome each one of the secondary objectives, solving several practical problems that appeared during the development.

A lot of time and effort was dedicated to the LabVIEW, an essential part of the system, since it contains the implementation of the control system, the model simulation, interaction with the plant, etc. With respect to the EJS, it was important to have an intuitive and easy to use graphical interface. The communication between LabVIEW and EJS, while it is not a trivial aspect, was not a problem thanks to the use of JIL software, which provides high-level tools to perform this difficult task.

Related to the other main objectives, we have obtained good results with the algorithm based on the combination of event-driven control and decoupling strategies. The use of a decentralized control scheme with event-based PI controllers and the decoupling network obtained a good performance, and led to a reduction in the number of communications between the controller and the actuator. There was a trade-off between the performance and the number of communication, adjustable by the event triggering thresholds. Due to the interaction presented in the system, a disturbance or set-point change introduced in one loop provokes the increase of the number of events of all controllers. With the decoupling network a set-point change in one control loop can be managed by its associate controller without affecting the other loop, and thus leading to a reduction in the number of events of this second loop. It must be noted however that the results were not as good in all the circunstances. One of the disadvantages of decoupling is that though when the set-point following task achieves a good performance, the disturbance rejection task is not guaranteed. In particular, for the non-minimum phase system this was a problem that in the worst case led to unstabilities. This problem can be aggravated with the introduction of event-based

mechanisms.

As future lines of work, it is desirable to add a webcam so that the user can see directly what is happening in the plant, since the application can be used as a remote laboratory. This is straightforward to do because it was already considered in the design. However, during the development phase the webcam was not available. With respect to the control system, the results presented for the event-based PI controller together with the decoupling strategy are promising, but still much work has to be done.

Appendix A

Minimum-phase plots

In this appendix the results of all the controllers tested with the minimum-phase plant are showed. Each plot shows the two output, the control signal generated by the PI controllers and after the decoupler, and the events triggered.

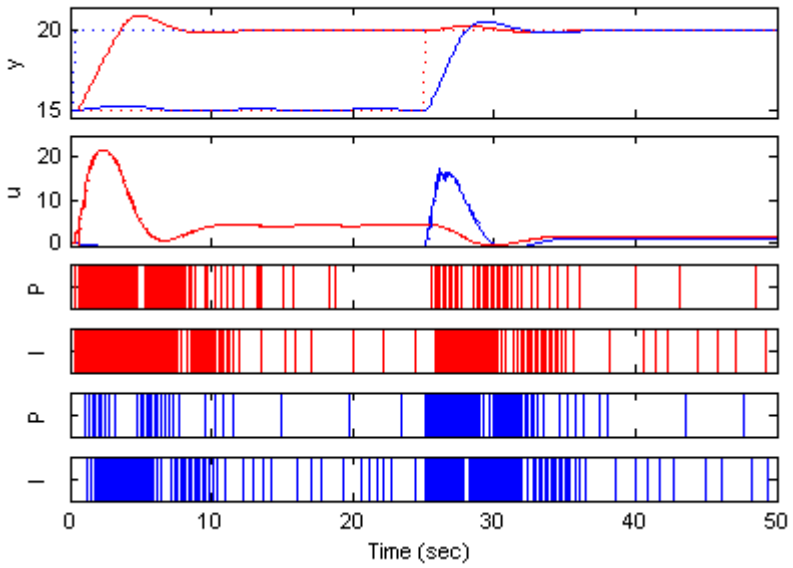

Figure A.1: Setpoint steps without decoupling, event thresholds $\delta_p = \delta_i = 0.01$.

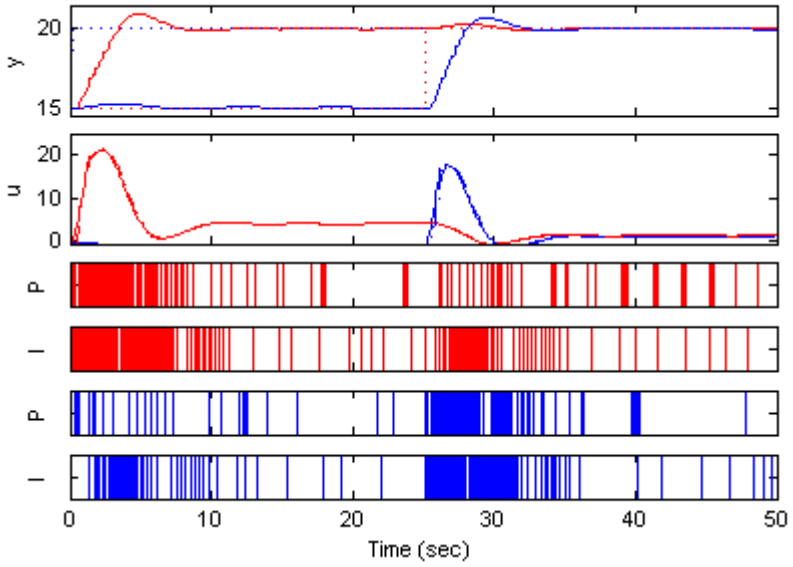

Figure A.2: Setpoint steps without decoupling, event thresholds $\delta_p = \delta_i = 0.02$.

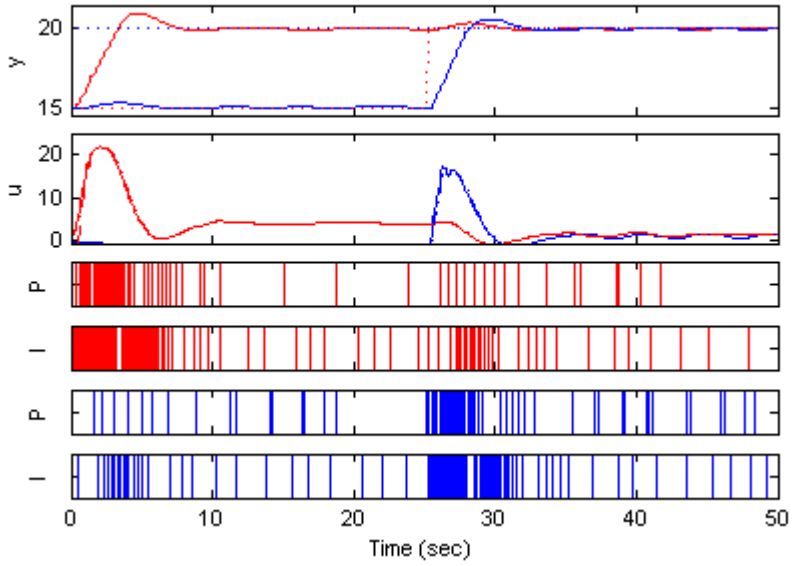

Figure A.3: Setpoint steps without decoupling, event thresholds $\delta_p = \delta_i = 0.05$.

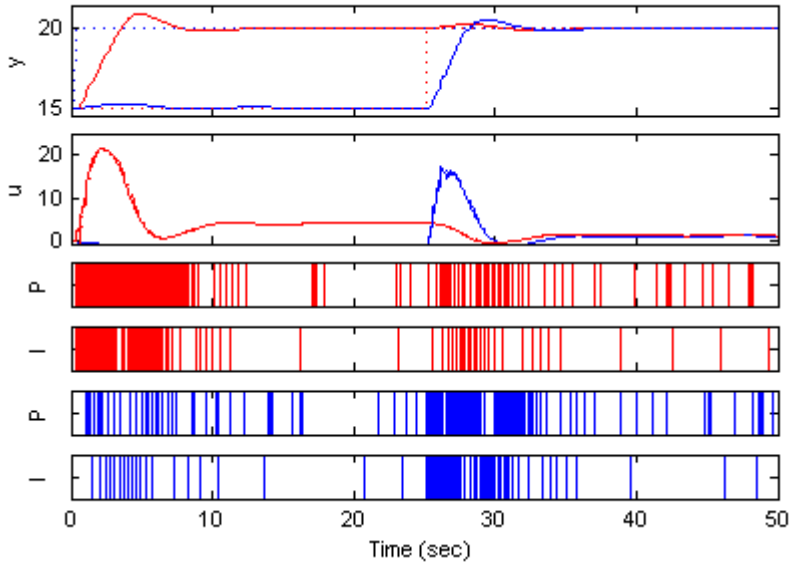

Figure A.4: Setpoint steps without decoupling, event thresholds $\delta_p = 0.01$, $\delta_i = 0.05$.

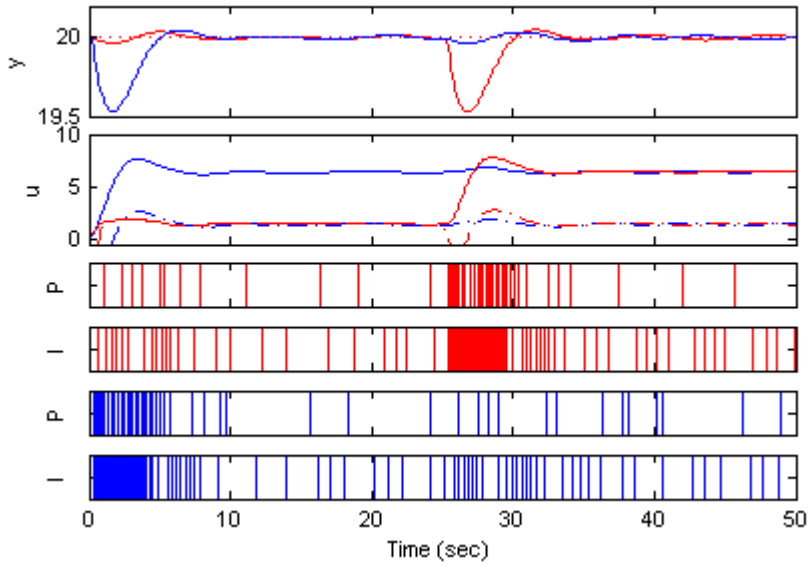

Figure A.5: Disturbance for the minimum-phase plant without decoupling and event-based PI with event thresholds $\delta_p = \delta_i = 0.01$.

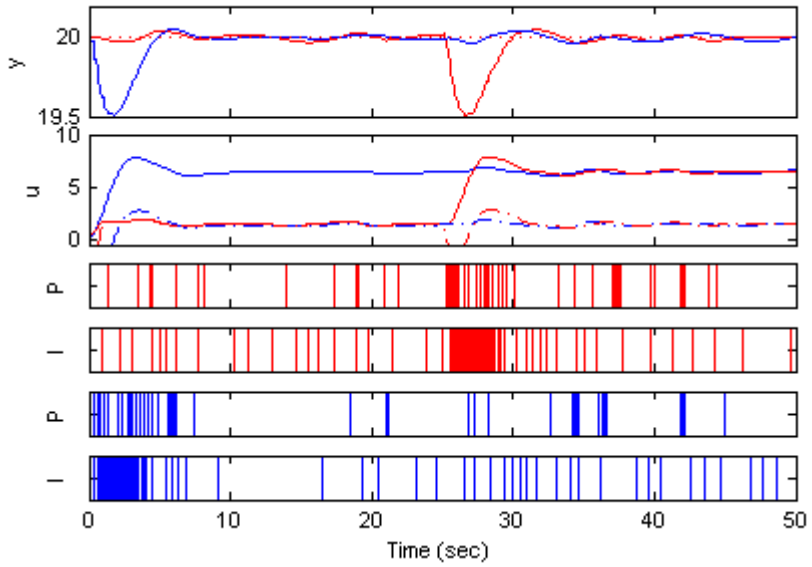

Figure A.6: Disturbance for the minimum-phase plant without decoupling and event-based PI with event thresholds $\delta_p = \delta_i = 0.02$.

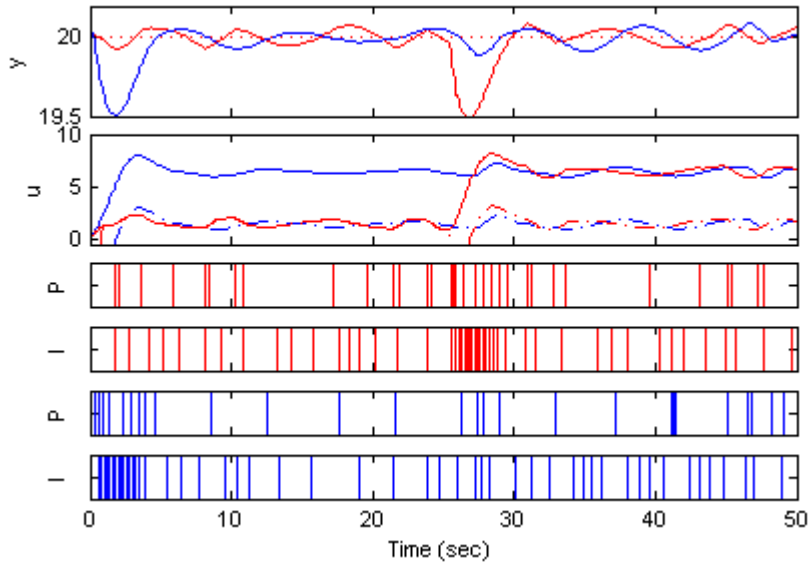

Figure A.7: Disturbance for the minimum-phase plant without decoupling and event-based PI with event thresholds $\delta_p = \delta_i = 0.05$.

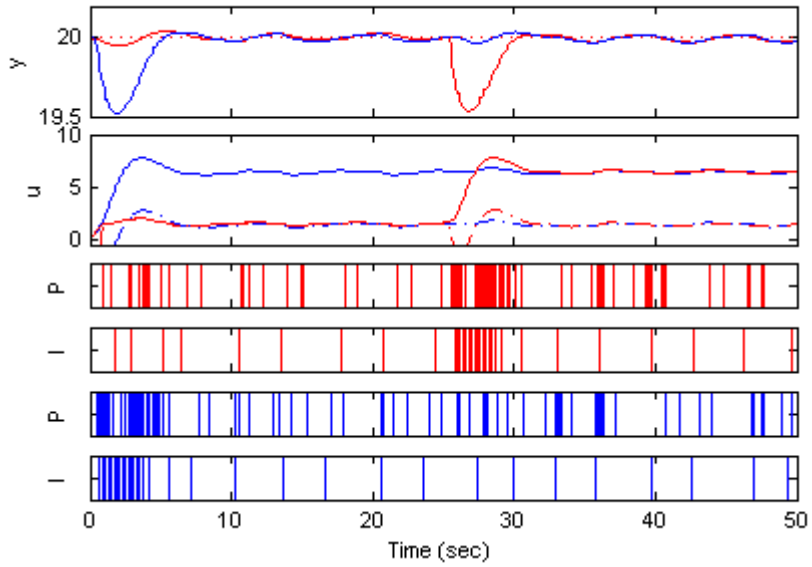

Figure A.8: Disturbance for the minimum-phase plant without decoupling and event-based PI with event thresholds $\delta_p = 0.01$,$\delta_i = 0.05$.

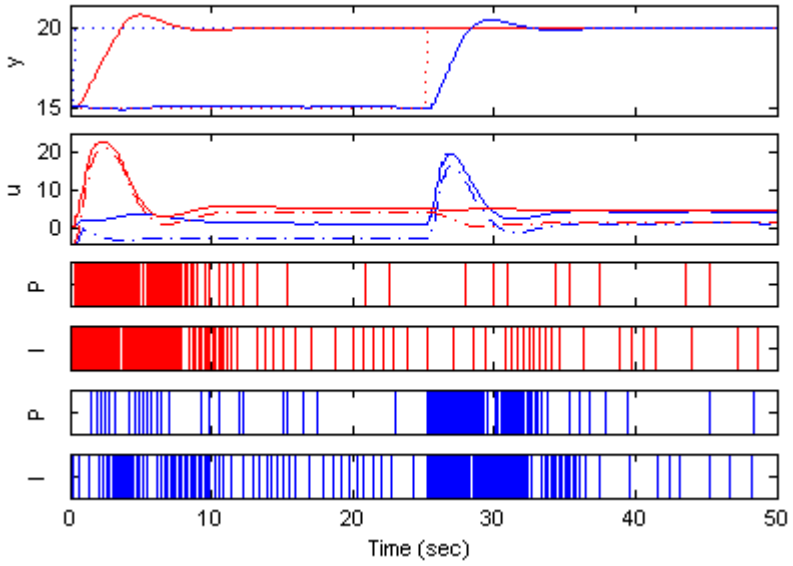

Figure A.9: Setpoint steps with decoupling, event thresholds $\delta_p = \delta_i = 0.01$.

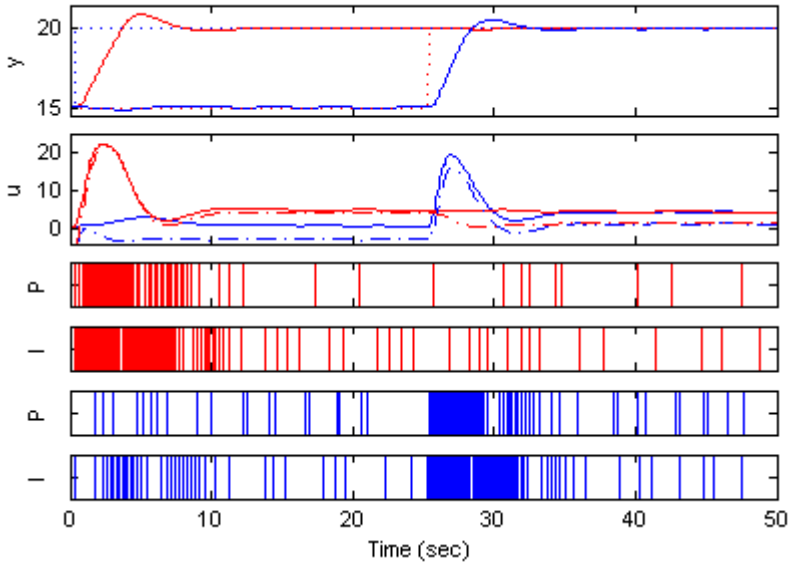

Figure A.10: Setpoint steps with decoupling, event thresholds $\delta_p = \delta_i = 0.02$.

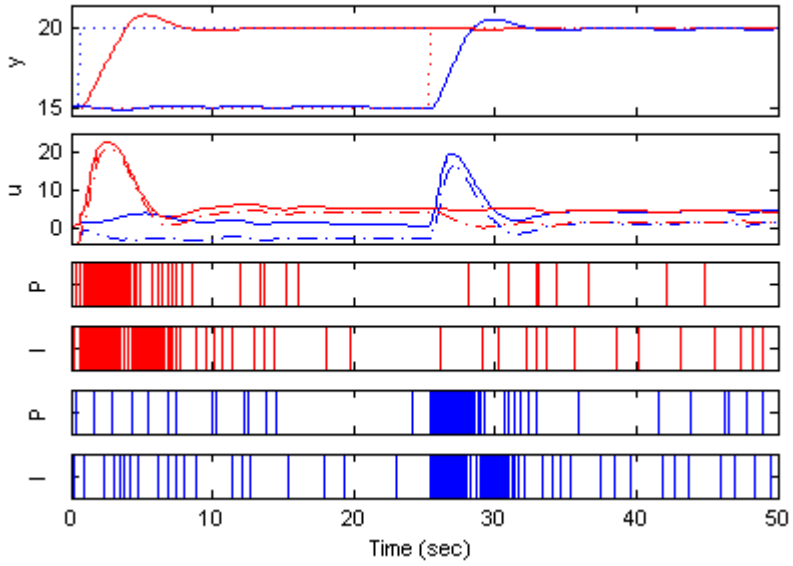

Figure A.11: Setpoint steps with decoupling, event thresholds $\delta_p = \delta_i = 0.05$.

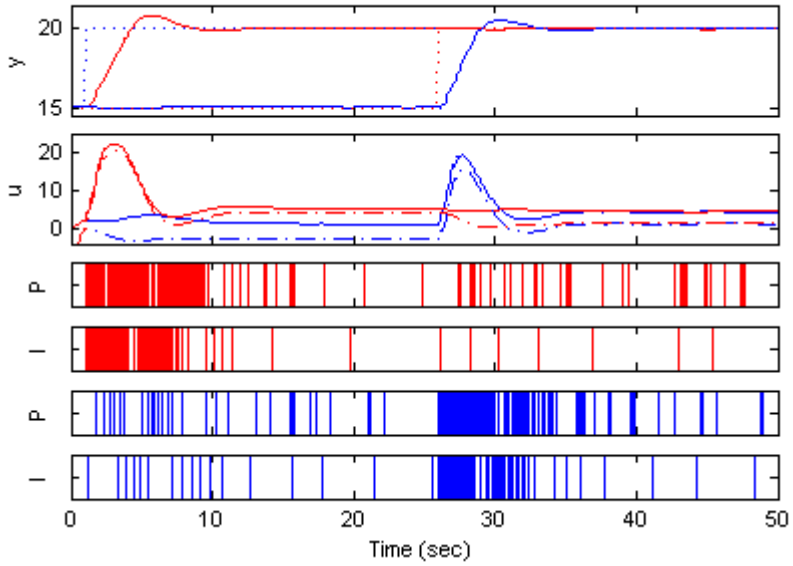

Figure A.12: Setpoint steps with decoupling, event thresholds $\delta_p = 0.01, \delta_i = 0.05$.

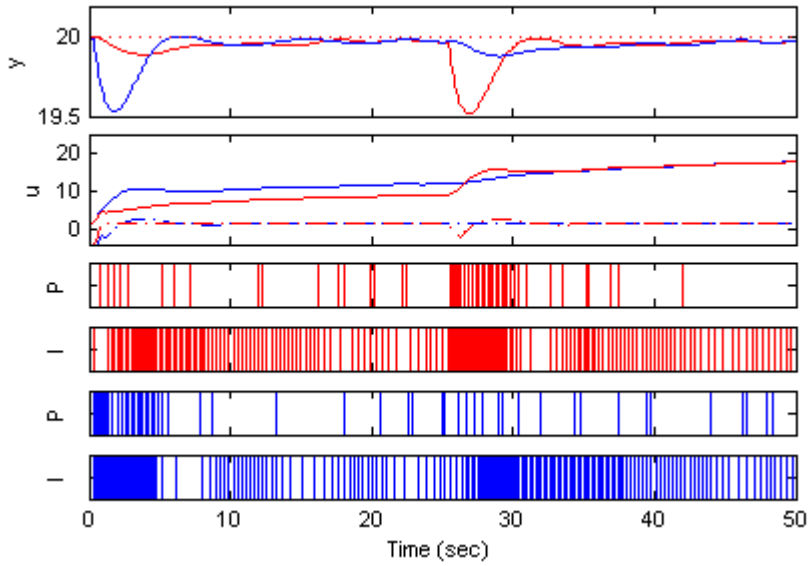

Figure A.13: Disturbance for the minimum-phase plant with decoupling and event-based PI with event thresholds $\delta_p = \delta_i = 0.01$.

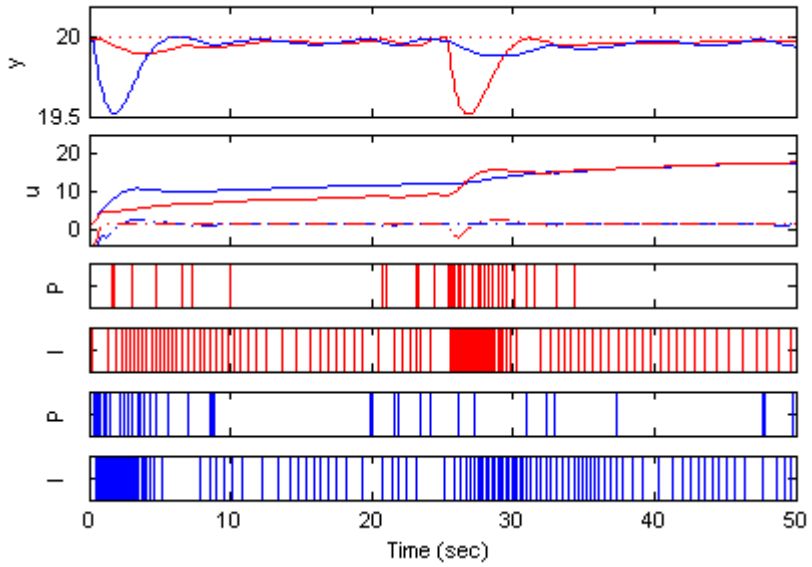

Figure A.14: Disturbance for the minimum-phase plant with decoupling and event-based PI with event thresholds $\delta_p = \delta_i = 0.02$.

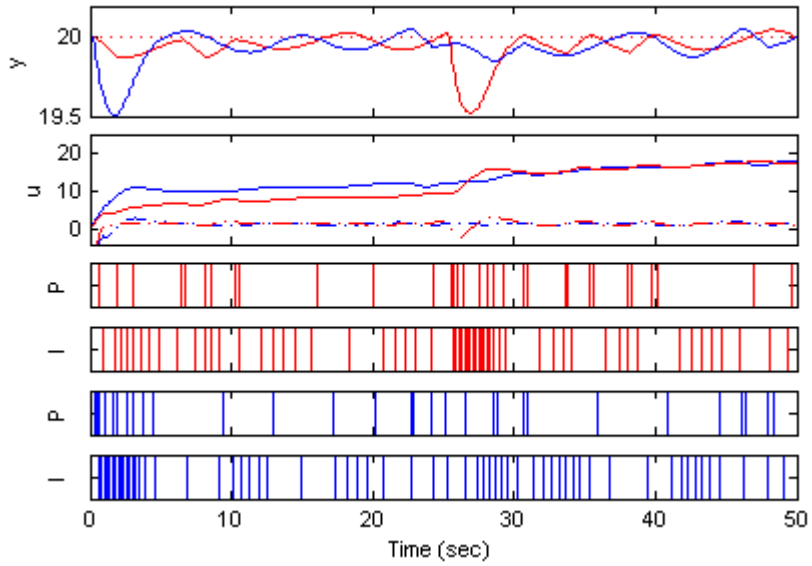

Figure A.15: Disturbance for the minimum-phase plant with decoupling and event-based PI with event thresholds $\delta_p = \delta_i = 0.05$.

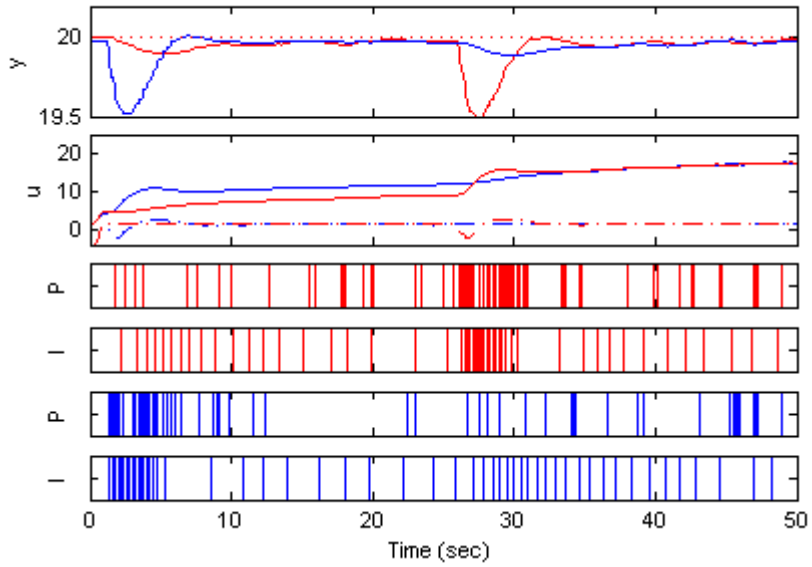

Figure A.16: Disturbance for the minimum-phase plant with decoupling and event-based PI with event thresholds $\delta_p = 0.01$,$\delta_i = 0.05$.

Appendix B

Non minimum-phase plots

In this appendix the results of all the controllers tested with the non minimum-phase plant are showed. Each plot shows the two output, the control signal generated by the PI controllers and after the decoupler, and the events triggered.

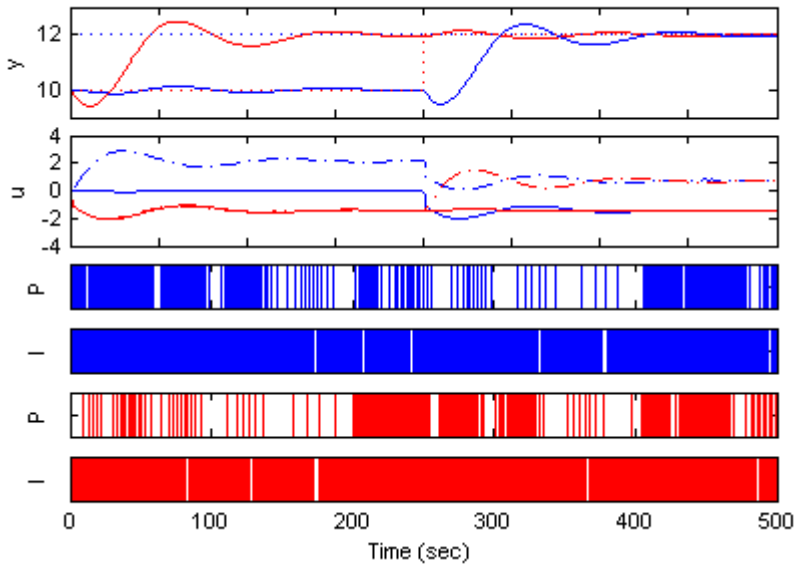

Figure B.1: Setpoint steps without decoupling, event thresholds $\delta_p = \delta_i = 0.01$.

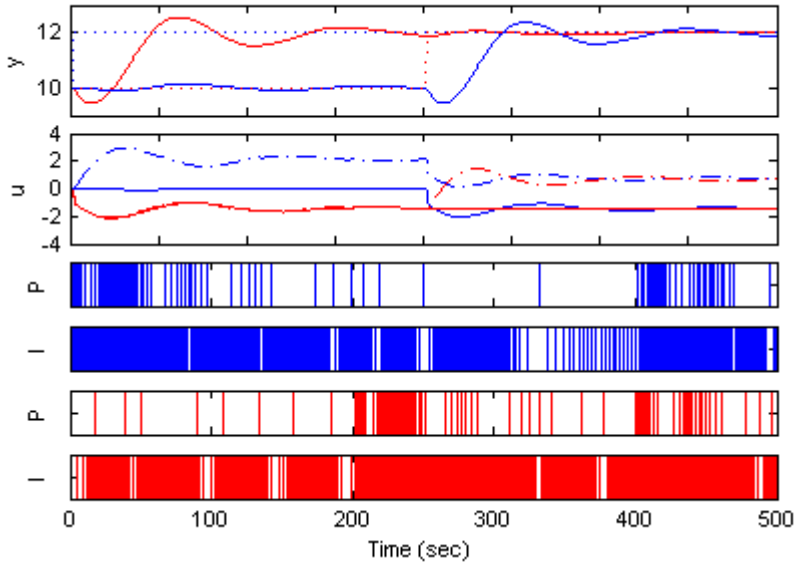

Figure B.2: Setpoint steps without decoupling, event thresholds $\delta_p = \delta_i = 0.02$.

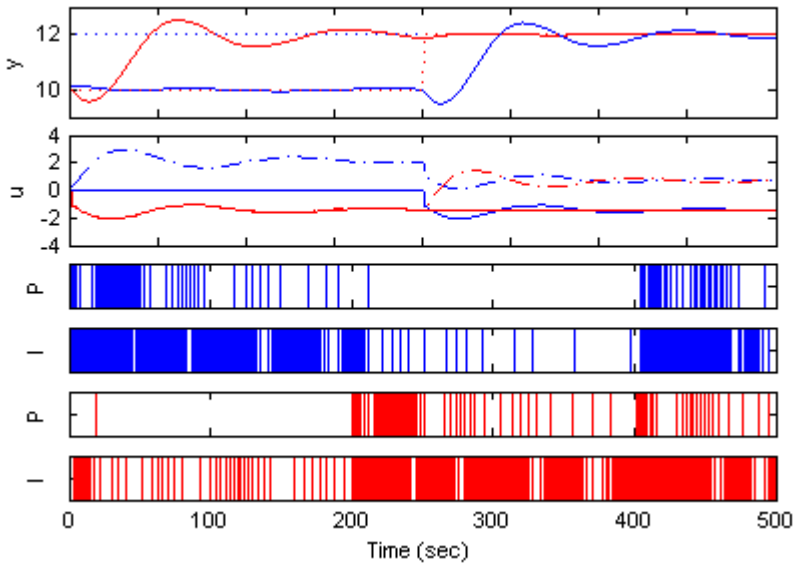

Figure B.3: Setpoint steps without decoupling, event thresholds $\delta_p = \delta_i = 0.05$.

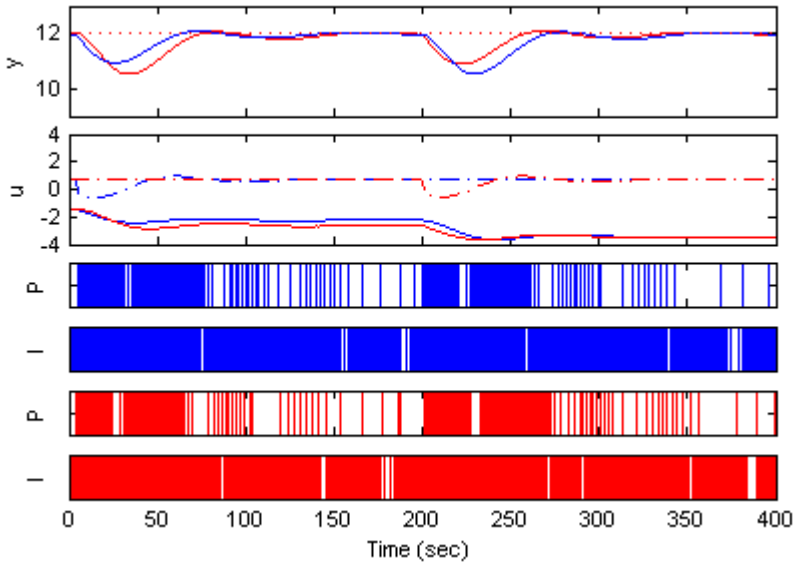

Figure B.4: Disturbance for the minimum-phase plant without decoupling and event-based PI with event thresholds $\delta_p = \delta_i = 0.01$.

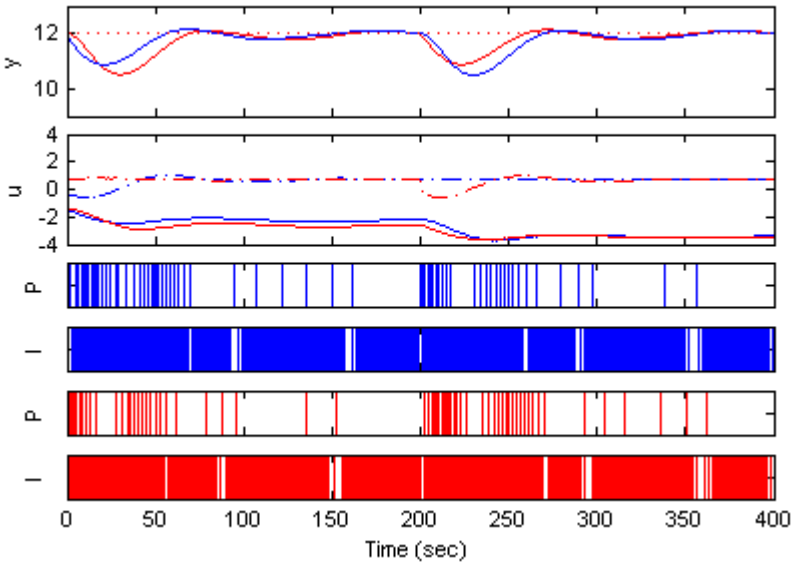

Figure B.5: Disturbance for the minimum-phase plant without decoupling and event-based PI with event thresholds $\delta_p = \delta_i = 0.02$.

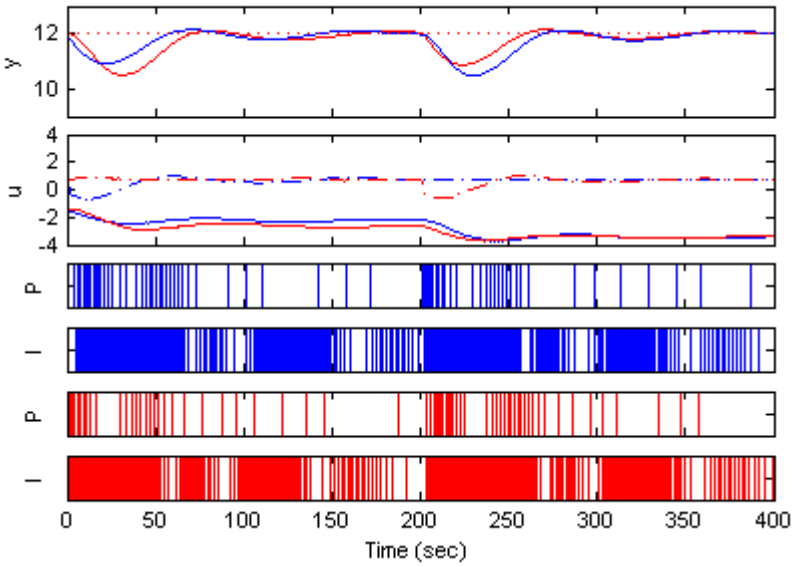

Figure B.6: Disturbance for the minimum-phase plant without decoupling and event-based PI with event thresholds $\delta_p = \delta_i = 0.05$.

Bibliography

[1] Candelas, F.A. and Sánchez, J., "Didactic resources on Internet based in system engineering and automation", Latinamerican Journal on Automation and Industrial Informatics, vol. 2, no. 2, pp. 93-101, April 2005.

[2] Candelas, F.A., Torres, F., Gil, P., Ortiz, F., Puente, S. and Pomares, J., "Virtual remote laboratory and its impact evaluation in Academics", Latinamerican Journal on Automation and Industrial Informatics, vol. 1, no. 2, pp. 49-57, July 2004.

[3] Johansson, K. H., "Relay feedback and multivariable control". PhD thesis TFRT-1048, Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1997.

[4] Johansson, K. H., "The quadruple-tank process: A multivariable laboratory process with an adjustable zero". IEEE Trans. on Control Systems Technology, vol 8, no 3, may, 2000, pp. 456-465.

[5] K. Åström, and B. Bernhardsson, "Comparison of periodic and event based sampling for first-order stochastic system", *in Proc. of the 14th IFAC World Congress*, Beijing, P.R. China, 1999.

[6] A. Capponi, L. Shi, C. Pilotto, and R.M. Murray, "Estimating Sensor Lifetime using an Event Based Contol Strategy", *Third International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks*, Annapolis, Maryland, USA, 2008

[7] V. Vasyutynskyy, and K. Kabitzsch, "Implementation of PID controller with send-on-delta sampling", in *Proc. ICC'2006, International Conference Control 2006*, August 2006, Glasgow, Scotland.

[8] K.-E. Arzén, "A simple event-based PID Controller", in *Proc. of the 14th IFAC World Congress*, Beijing, P.R. China, 1999.

[9] J. Lunze, and D. Lehmann, "A state-feedback approach to event-based control", Automatica, 2010, pp. 211-215.

[10] J. Sánchez, A. Visioli, and S. Dormido, "An event based PI controller based on feedback and feedforward action", *35th Annual Conference of the IEEE Industrial Electronics Society (IECON 2009)*, Porto (Portugal), 2009, pp. 1464-1469.

[11] J. Chacón, J. Sánchez, and S. Dormido, "Experimental study of an event-based PI controller in a wireless control system", *9th Portuguese Conference on Automatic Control (Controlo 2010)*, Coimbra (Portugal), 8-10 September, 2010.