ISSN 0280–5316
ISRN LUTFD2/TFRT--5588--SE

Modeling and Control of the Quadruple-Tank Process

José Luís Nunes

Department of Automatic Control
Lund Institute of Technology
March 1998

Department of Automatic Control
Lund Institute of Technology
Box 118
S-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
March 1998

Document Number
ISRN LUTFD2/TFRT--5588--SE

Author(s)
José Luís Nunes

Supervisor
Karl Henrik Johansson

Sponsoring organisation

Title and subtitle
Modeling and Control of the Quadruple-Tank Process

Abstract

In this master thesis a new quadruple-tank process is investigated and a user interface is developed.

The quadruple-tank process is a coupled control system with two inputs and two outputs. It is built in a way that it is easy to have both a minimum and nonminimum phase system. These two systems settings are investigated, and the results of the physical modeling, identification and control are presented.

A real-time graphical user interface for the process is developed. It includes a simulation of the real process, MIMO controllers, and facilities to log data. This interface is extensively used in the experiments performed in this master thesis.

Key words
Multivariable systems; Multivariable zero; Decentralized control; Performance limitations; Laboratory process; Physical Modeling; Identification; Process control; Real-Time Control Systems; User Interface

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
47

Recipient's notes

Security classification

Contents

1. Introduction

In many industrial processes there are hundreds of control loops, which interact more or less. The control design has often been performed using single-input single-output (SISO) methods. However, modern control theory provides design approaches that take care of the interactions.

In this master thesis project, a coupled control system consisting of two loops is investigated. A new quadruple-tank process is built and a real-time control system is developed using the real-time kernel [6] developed at the department, and the man-machine interface software InTouch. The process connected with a PC running the graphical user interface is shown in Figure 1.

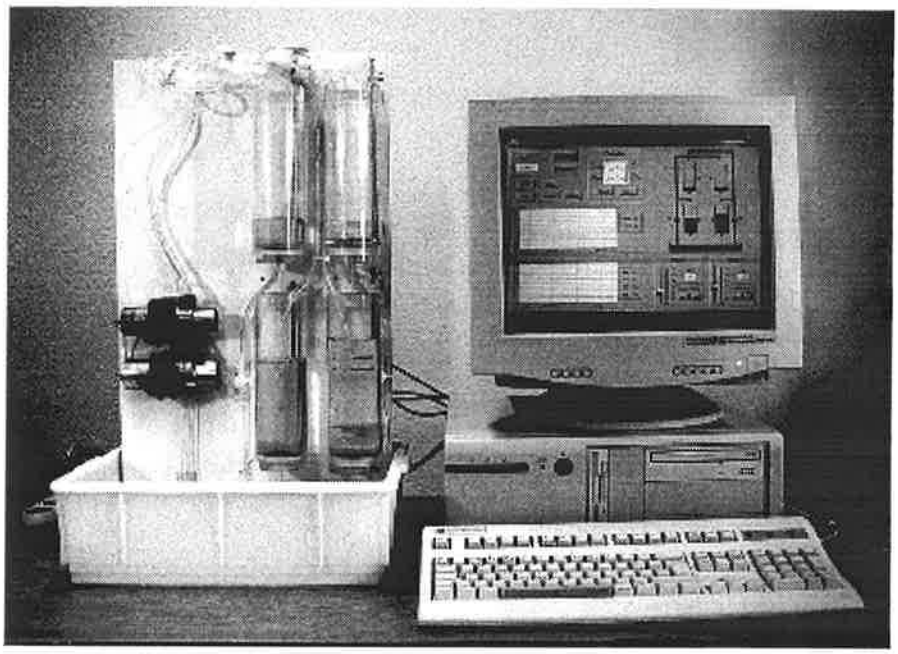

Figure 1 Photograph of the quadruple-tank process connected with a PC running the graphical user interface.

The linearized dynamics of the quadruple-tank process contains a zero, whose location can be easily change. This feature is used to illustrate the difference in controlling the process when it is minimum and nonminimum phase. For each system setting a system modeling and identification is performed, and the performance of the decentralized PID control structure is examined.

This work originated a paper [16] that was submitted to a conference, and part of it is also found in a doctoral thesis [13].

The outline of the report is as follows. In Section 2 we describe the quadruple-tank process and derive a mathematical model of the system, based on physical data. Some system properties are also investigated, and physical as well as mathematical interpretations are given. In Section 3 we explain the procedure on how to identify the system based on experimental data. In Section 4 we test some controllers on the real process. Decentralized PID control is used and the results for both system settings are compared. In Section 5 we describe the user interface and explain how some relevant parts of it were implemented. In the conclusions, in Section 6, we leave some guidelines

for further developments of the user interface. In Appendix A we include a simple and short guide on how to setup the process and configure the interface. A sample session is included in the appendix.

2. The Quadruple-Tank Process

2.1 Introduction

The quadruple-tank process was built in order to study various behaviors in interacting control systems. The process is a MIMO system with two inputs and two outputs, consisting of four tanks and two pumps, see Figure 2. The voltages fed to the pumps correspond to the inputs of the system and the levels of the lower tanks to the outputs.

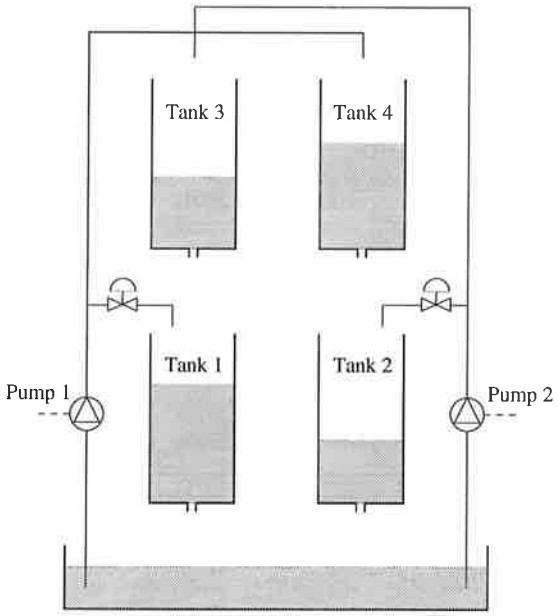

Figure 2 The quadruple-tank process consists of four tanks, two pumps and two valves. The inputs are the voltages fed to the pumps and the outputs the levels of the lower tanks.

By the adjustment of two valves, it is possible to set the relative water flow in the tubes. As we will see in the next subsection, this will change the position of the transmission zero of the linearized model of the system, giving a more or less difficult process to control. The influence of the zero position on the identification and controller performance is investigated latter in Sections 3 and 4. The derivations seen here are also done in [16] and [13].

2.2 Physical Modeling

Here we derive a mathematical model from physical data. We start by deriving the nonlinear differential equations that describe the tanks from Bernoulli's law. Then we linearize these equations and give the linear input-output map both in a state-space representation and as a transfer function matrix. Some properties considering multivariable zeros are derived, and finally the physical parameter values of the quadruple-tank process are given.

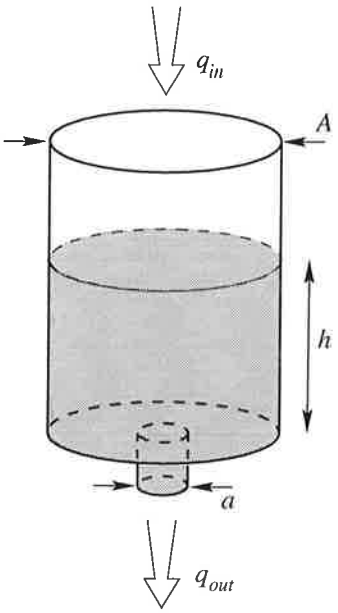

Figure 3 A simple tank where h represents the water level, a and A the hole and tank cross-section, respectively, and q_{in} and q_{out} the inflow and the outflow.

Derivation

Assuming mass balance for a tank like the one show in Figure 3, we have

$$A\frac{dh}{dt} = -q_{out} + q_{in}$$

where A denotes the cross-section of the tank, q_{in} and q_{out} the inflow and outflow of the tank, respectively, and h the water level. Bernoulli's law gives $q_{out} = a\sqrt{2gh}$ where a is the cross-section of the hole and g is the acceleration of gravity. The flow through each pump is split proportional to how the valves are adjusted, see Figure 2. If we assume that the flow generated by the each pump is proportional to the applied voltage, we get

$$q_L = \gamma k u, \qquad q_U = (1-\gamma)ku, \qquad \gamma \in [0,1]$$

where q_L is the flow going to the lower tank and q_U flow going to the upper tank. The parameter γ is determined from how the valves are set. It follows that the nonlinear dynamics of the quadruple-tank process is given by

$$\begin{aligned}
\frac{dh_1}{dt} &= -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{A_1}u_1 \\
\frac{dh_2}{dt} &= -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2 k_2}{A_2}u_2 \\
\frac{dh_3}{dt} &= -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)k_2}{A_3}u_2 \\
\frac{dh_4}{dt} &= -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)k_1}{A_4}u_1
\end{aligned} \tag{1}$$

where subscript i of a_i, A_i and h_i represents Tank i, k_i corresponds to Pump i, and γ_i to the flow through Pump i.

Linearization

Linearizing around an operating point $(h_1^0, h_2^0; u_1^0, u_2^0)$ and introducing $\Delta h_i = h_i - h_i^0$ and $\Delta u_i = u_i - u_i^0$, yield the linear equations

$$\begin{aligned}
\frac{dh_1}{dt} &= -\frac{a_1}{A_1}\sqrt{\frac{g}{2h_1^0}}\Delta h_1 + \frac{a_3}{A_1}\sqrt{\frac{g}{2h_3^0}}\Delta h_3 + \frac{\gamma_1 k_1}{A_1}\Delta u_1 \\
\frac{dh_2}{dt} &= -\frac{a_2}{A_2}\sqrt{\frac{g}{2h_2^0}}\Delta h_2 + \frac{a_4}{A_2}\sqrt{\frac{g}{2h_4^0}}\Delta h_4 + \frac{\gamma_2 k_2}{A_2}\Delta u_2 \\
\frac{dh_3}{dt} &= -\frac{a_3}{A_3}\sqrt{\frac{g}{2h_3^0}}\Delta h_3 + \frac{(1-\gamma_2)k_2}{A_2}\Delta u_2 \\
\frac{dh_4}{dt} &= -\frac{a_4}{A_4}\sqrt{\frac{g}{2h_4^0}}\Delta h_4 + \frac{(1-\gamma_1)k_1}{A_4}\Delta u_1
\end{aligned}$$

We now assume that the measured signal y_i is proportional to the true level h_i, i.e.,

$$y_i = k_c h_i$$

Let the time constants $T_i = \frac{A_i}{a_i}\sqrt{\frac{2h_i^0}{g}}$ and rename states and inputs as $x = \Delta h$ and $u = \Delta u$, respectively. Then we get the following state-space representation

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

which corresponds to the transfer function matrix

$$\begin{aligned}
G(s) &= \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} \\
&= \begin{bmatrix} \frac{\gamma_1 k_1 c_1}{1+sT_1} & \frac{(1-\gamma_2)k_2 c_1}{(1+sT_3)(1+sT_1)} \\ \frac{(1-\gamma_1)k_1 c_2}{(1+sT_4)(1+sT_2)} & \frac{\gamma_2 k_2 c_2}{1+sT_2} \end{bmatrix}
\end{aligned} \tag{2}$$

with

$$c_1 = \frac{T_1 k_c}{A_1}, \qquad c_2 = \frac{T_2 k_c}{A_2}$$

System Properties

We are going to establish relations between some physical settings in the system and the properties of the system. In particular we will show the influence of the valve positions on the stationary point of (1) and on the transmission zero of (2).

Stationary point For a stationary operating point $(h_1^0, h_2^0; u_1^0, u_2^0)$ with $h_3 = h_3^0$ and $h_4 = h_4^0$, (1) gives that

$$\begin{aligned} \frac{a_3}{A_3}\sqrt{2gh_3^0} &= \frac{(1-\gamma_2)k_2}{A_3}u_2^0 \\ \frac{a_4}{A_4}\sqrt{2gh_4^0} &= \frac{(1-\gamma_1)k_1}{A_4}u_1^0 \end{aligned}$$

and thus

$$\begin{aligned} \frac{a_1}{A_1}\sqrt{2gh_1^0} &= \frac{(1-\gamma_2)k_2}{A_1}u_2^0 + \frac{\gamma_1 k_1}{A_1}u_1^0 \\ \frac{a_2}{A_2}\sqrt{2gh_2^0} &= \frac{(1-\gamma_1)k_1}{A_2}u_1^0 + \frac{\gamma_2 k_2}{A_2}u_2^0 \end{aligned}$$

it follows that there always exists a unique input (u_1^0, u_2^0) giving the stationary point (h_1^0, h_2^0) if and only if the matrix

$$M = \begin{bmatrix} \gamma_1 k_1 & (1-\gamma_2)k_2 \\ (1-\gamma_1)k_1 & \gamma_2 k_2 \end{bmatrix}$$

is non-singular, i.e., $k_1 k_2(\gamma_1+\gamma_2-1) \neq 0$. This happens if and only if $\gamma_1+\gamma_2 \neq 1$. The singularity is natural. For simplicity assume $k_1 = k_2 = 1$. Then the flow through Tank 1 is $\gamma_1 q_1 + (1-\gamma_2)q_2$ and the flow through Tank 2 is $\gamma_2 q_2 + (1-\gamma_1)q_1$. If $\gamma_1+\gamma_2 = 1$, these flows equal $\gamma_1(q_1+q_2)$ and $(1-\gamma_1)(q_1+q_2)$, respectively. The flows through Tank 1 and Tank 2 are hence dependent, and so must then the levels also be.

Zeros Briefly we recall the notion of zeros in SISO and MIMO systems. For further reference see [1]. The zeros of a SISO system are the roots of the numerator polynomial of its transfer function. The system has different properties, depending on the zero locations. We have two types of systems:

minimum phase systems, for which all zeros are located in the left half-plane (having a negative real part); and

nonminimum phase systems, when at least one zero is in the right half-plane (having a real part greater than zero).

The location of the zeros influence the difficulty in controlling the system.

There are many ways of defining zeros of MIMO systems. We only need the simplest one and therefore define a MIMO zero z as a point in the complex plane for which the rank of $G(s)$ drops below its normal rank (full). These zeros are often called transmission zeros. It is possible to show that many of the properties of SISO zeros also hold for transmission zeros, for example, transmission zeros in the right half-plane impose restrictions on the achievable control performance.

From the transfer function matrix (2) we get,

$$\det G(s) = \frac{k_1 k_2 c_1 c_2}{\prod_{i=1}^{4}(1+sT_i)}\left[(1+sT_3)(1+sT_4) - \frac{(1-\gamma_1)(1-\gamma_2)}{\gamma_1\gamma_2}\right]$$

Hence, the system has two zeros. One of these is always in the left half-plane. The location of the other zero depends however on the sign of

$$\eta = 1 - \frac{(1-\gamma_1)(1-\gamma_2)}{\gamma_1\gamma_2}$$

It follows that the zero is in the right half-plane if $\eta < 0$, in the left half-plane if $\eta > 0$, and in the origin if $\eta = 0$. Then the system is nonminimum phase for

$$0 < \gamma_1 + \gamma_2 \leq 1$$

and minimum phase for

$$1 < \gamma_1 + \gamma_2 \leq 2$$

Note that the zero in the origin corresponds to $\gamma_1 + \gamma_2 = 1$, which equals the singularity discussed when deriving the stationary point.

Parameter Values

To determine the values of the physical constants in the derived model, we have to measure some components of the real process (like the diameter of the holes and the tanks) and perform some simple experiments in order to find the values of the pump constants and valve positions.

The quadruple-tank process consists of two types of tanks which have slightly different dimensions. Tank 1 and Tank 3 have dimensions

$$\begin{aligned} A &= 28 \text{ cm}^2 \\ a &= 0.071 \text{ cm}^2 \end{aligned}$$

The dimensions of Tank 2 and Tank 4 are

$$\begin{aligned} A &= 32 \text{ cm}^2 \\ a &= 0.057 \text{ cm}^2 \end{aligned}$$

The constant k_c used in the level sensors is

$$k_c = 0.50 \text{ V/cm}$$

To determine the constants associated with the pumps, we check the time it takes to fill a tank when the input voltage fed to the pump is constant and known, and the outlet of the tank is stopped. The values we get are

$$\begin{aligned} k_1 &= 3.1 \text{ cm}^3/\text{s} \\ k_2 &= 3.3 \text{ cm}^3/\text{s} \end{aligned}$$

The values for the these constants could vary a little if the voltage fed to the pumps differs much from the ones used for the calculations. For that, to get a physical model that behaves as much as possible as the real one, it necessary to do some other experiments to calibrate these constants and to determine the two parameters γ_1 and γ_2. The procedure used to determine and fine tune these parameters is explained next.

If we are interested in finding a particular setting for the valves all that has to be done is to run the process until the steady-state is reached, and then from the nonlinear equations for Tank 3 and Tank 4 extract the values of γ_1 and γ_2. For that we need to know the stationary level of the upper tanks – (h_3^0, h_4^0). Since γ_1 and γ_2 are derived from h_3^0 and h_4^0, it is almost certain that the level of the lower tanks extracted from the model (1) doesn't exactly match the experimental values. To overcome this and because these levels correspond to the model outputs, we have to do some adjustments to have approximately the same stationary levels. For this, small changes is the model parameters are done while checking the model outputs for the best match.

Minimum phase system By setting the position of the valves so that we have more flow going to the lower tanks, i.e., $\gamma_1 + \gamma_2 > 1$, we reach an operating point of $(h_1^0, h_2^0; u_1^0, u_2^0) \approx (12.4, 12.7; 3.0, 3.0)$. And using the expressions derived in this section knowing that the stationary level of the upper tanks are $(h_3^0, h_4^0) \approx (1.8, 1.4)$, we obtain the following values for the time constants, T_i, associated with each tank,

$$\begin{aligned} T_1 &= 62 \text{ s} \\ T_2 &= 90 \text{ s} \\ T_3 &= 23 \text{ s} \\ T_4 &= 30 \text{ s} \end{aligned}$$

and for the other parameters

$$\begin{aligned} k_1 &= 3.3 \text{ cm}^3/\text{s} \\ k_2 &= 3.3 \text{ cm}^3/\text{s} \\ \gamma_1 &= 0.70 \\ \gamma_2 &= 0.60 \end{aligned}$$

With these parameters we can compute the transfer function matrix of the system

$$G(s) = \begin{bmatrix} \frac{2.6}{1+62s} & \frac{1.5}{(1+23s)(1+62s)} \\ \frac{1.4}{(1+30s)(1+90s)} & \frac{2.8}{1+90s} \end{bmatrix}$$

and the transmission zeros $z = -0.018$ and $z = -0.060$. A plot showing the output of the real process and the output of this model can be seen in Figure 4. The data used for the comparison is the MIMO data that we collect in Section 3.2 for the minimum phase system setting.

Nonminimum phase system If we instead have less flow going to the lower tanks, by setting the valves position so that $\gamma_1+\gamma_2 < 1$ but trying to keep the operating point close to the one we got for the minimum phase setting, we come to an operating point of $(h_1^0, h_2^0; u_1^0, u_2^0) \approx (12.6, 13.0; 3.15, 3.15)$ and an upper tanks stationary level of $(h_3^0, h_4^0) \approx (4.8, 4.9)$. For this setting the system time constants have the following values

$$\begin{aligned} T_1 &= 63 \text{ s} \\ T_2 &= 91 \text{ s} \\ T_3 &= 39 \text{ s} \\ T_4 &= 56 \text{ s} \end{aligned}$$

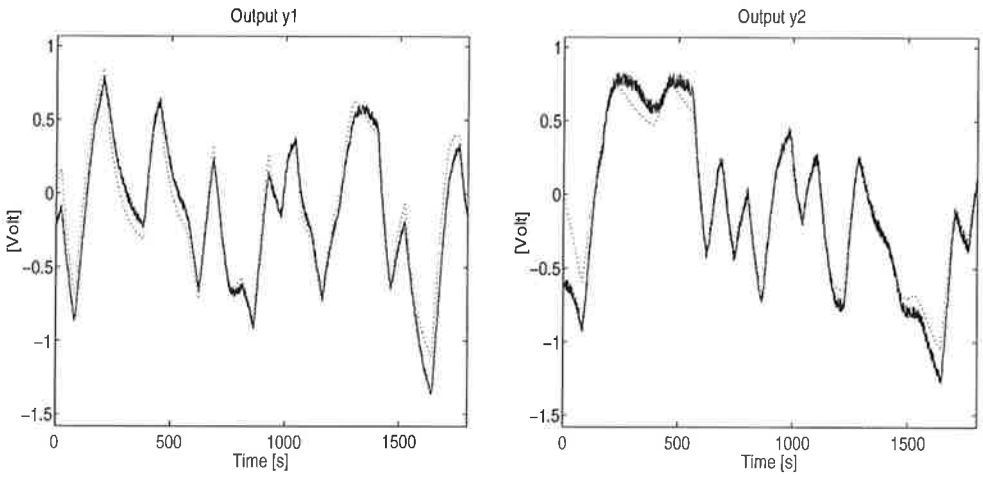

Figure 4 Measured (solid) and simulated (dotted) outputs for the physical model (minimum phase setting).

and the other parameters

$$\begin{aligned} k_1 &= 3.1 \text{ cm}^3/\text{s} \\ k_2 &= 3.2 \text{ cm}^3/\text{s} \\ \gamma_1 &= 0.43 \\ \gamma_2 &= 0.34 \end{aligned}$$

Then the system transfer function matrix derived by means of physical modeling is

$$G(s) = \begin{bmatrix} \frac{1.5}{1+63s} & \frac{2.5}{(1+39s)(1+63s)} \\ \frac{2.5}{(1+56s)(1+91s)} & \frac{1.6}{1+91s} \end{bmatrix}$$

with transmission zeros in $z = 0.013$ and $z = -0.057$. Note the positive transmission zero for this setting. The output of this model and the real output of the process are shown in Figure 5. The data used is the MIMO data that we collect in Section 3.2 for the nonminimum phase system setting.

2.3 Summary

The output of the models we derived by means of physical modeling, both for the minimum and nonminimum phase system settings, agree very well with the output of the real process (see Figures 4 and 5). In these figures we can see a disparity between the model and the process outputs, more noticeable for the nonminimum phase setting, in the initial phase. This is due to the fact that when we start the simulations we don't take in consideration the state of the process.

These models are compared with the models we get from the experimental data, in the Section 3. From now on, every time we mention the minimum and nonminimum phase systems we are referring to these particular settings.

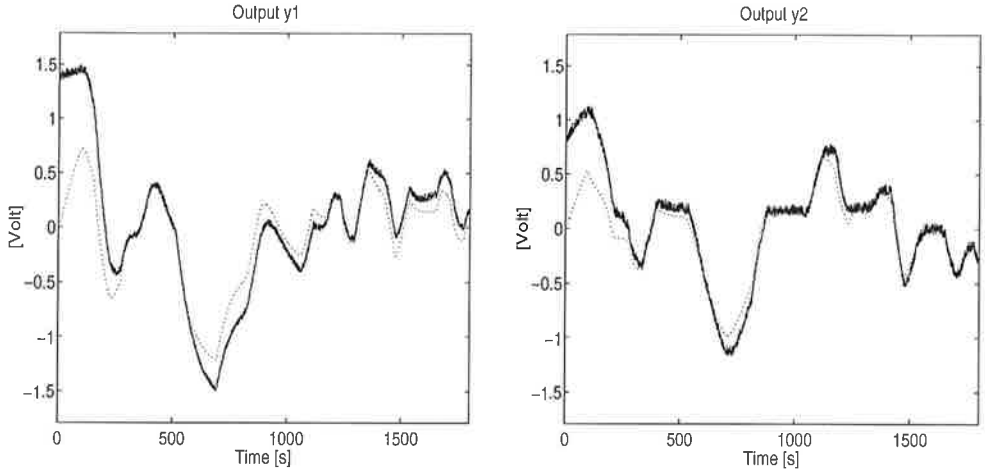

Figure 5 Measured (solid) and simulated (dotted) outputs for the physical model (nonminimum phase setting).

3. System Identification

3.1 Introduction

The goal of system identification consists in finding a description of the system from the observed data. Model complexity will depend on the purpose of the identification. In our case the aim is to find a model well suited for simulation of the real process and for control design. Model categories used were ARX and ARMAX as well as state-space structures.

The class of ARMAX models are given by difference equations on the form

$$A(z^{-1})y_k = z^{-d}B(z^{-1})u_k + C(z^{-1})w_k$$

where d is a time delay, A, B and C are polynomials with unknown parameters, and w_k is white noise. A special case of this type of models is the ARX models, where C is equal to 1. To estimate these models we use the ARX and ARMAX functions from the System Identification Toolbox [14] included in Matlab.

For state-space subspace identification the estimated models are of the form

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k \\ y_k &= Cx_k + Du_k + e_k \end{aligned}$$

where A, B, C, D and K are matrices to be identified, and w_k and e_k are white noise. To estimate these models we use a function called N4SID from the System Identification Toolbox, which implements Van Overschee's and De Moor's method [8] [14].

The use of state-space models was motivated by the fact that for multivariable system identification we could experience some difficulties when applying ARMAX models.

As it was said when we derived the physical model of the process, we have a two-input two-output process, so what will be done with the identification is to get a transfer function matrix that relates all the system inputs with all the

outputs. This matrix, $G(s)$ will be a 2-by-2 matrix, were the element $G_{ij}(s)$ correspond to the transfer function between input j and output i.

$$G(s) = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix}$$

To get these transfer functions, we can either use SIMO data and identify each of them individually, or we can use MIMO data and perform a multivariable system identification. The different types of data used in the identification, and the experiments done in order to collect it are presented in Section 3.2. A block diagram showing the interactions in the process is shown in Figure 6.

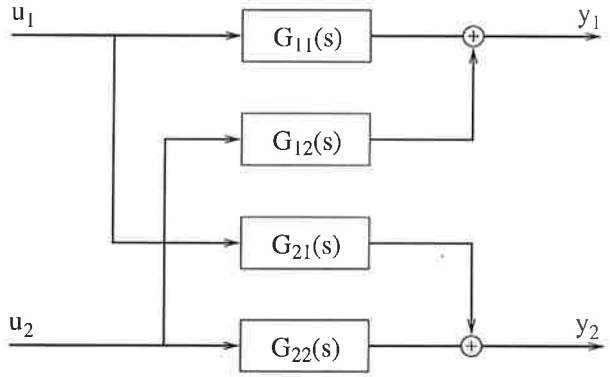

Figure 6 System's block diagram where G_{ij} represents the transfer function between the input u_i and the output y_j.

The procedure used to find a model that corresponds to a map between an input-output pair, was to divide each data set in two parts, one for the identification and the other for the validation of the model. In SISO identification we use the data collected in the first two experiments, so the transfer functions derived from each of these data sets are $G_{11}(s)$ and $G_{21}(s)$ from the first experiment, and $G_{12}(s)$ and $G_{22}(s)$ from the second. Concerning the MISO and MIMO identification, the data collected in the third experiment is used.

To test the validity of the models we use residual analysis, and cross-validation with the real data. In the residual analysis we take in consideration that if the model structure represents the data adequately, the residuals will be uncorrelated with each other. Also, the inputs and the residuals will be uncorrelated, indicating that the model extracted all the useful information from the data. With respect to the cross-validation, we test the models using the second half of the data set, which we also use for the identification. Latter, in order to compare the overall model derived using SISO methods with the MIMO model, we use the validation part of the MIMO data. In this way we can compare the ability of different models to track the outputs of the real process when we have two input signals acting on the system which are not constant all the time, even if some of the models (SISO) were not derived from such type of data.

The results of the identification are presented in Sections 3.3 and 3.4, where only the best model is presented. For further information on these and other models derived using different identification methods, specially the state-space subspace method, refer to [12].

To perform the identification we use extensively Matlab and the System Identification Toolbox, and latter Simulink for validation and test of the models.

3.2 Experiments

To have a good enough data to perform the identification we have to do some preliminary experiments, in order to get a first knowledge of the process. These are simple experiments, most of them consist on see how the process behaves to a determined input without collecting any data, and some other ones, like step responses or more generally, the system's response when the input has the form of a square wave, will let us prepare the continued experiments where we collect the data for the identification. Being in the presence of a MIMO system, more precisely a two-input two-output system, and as we wish to try different methods of perform the identification, we decide to do three continued experiments. These experiments will consist, basically, in feed a pseudo-random binary sequence (PRBS) through one input, while holding the other one constant (SIMO experiments), or through both of the inputs simultaneously (MIMO experiments). The data collected allow us to identify the four transfer functions that describe the dynamics of the quadruple-tank process using SISO, SIMO and MIMO identification methods. To do the first stage experiments and latter the continued experiments, we use a graphical user interface built for this specific process (see Section 5), that let us follow the evolution of the experiments and at the same time, collect the data we need.

Prior to the use of data for identification of the system transfer functions, we should concern about the quality of the data for identification purposes. At this stage, artifacts like outliers, aliasing or lost signals should be detected. Another valuable test is the coherence spectrum, as it serves as a measure of dependency between two signals. An important use is its application as a test of signal-to-noise ratio and linearity between two variables. The estimated coherence function is

$$\gamma_{xy}(\omega) = \frac{|S_{xy}(i\omega)|}{\sqrt{S_{xx}(i\omega)S_{yy}(i\omega)}}$$

A value of coherence close to one indicates the frequency range where we can expect a good approximation with a linear model. It also tells that a successful identification can be expected. The coherence function estimate plot of an input-output pair is presented whenever applicable to the data in question. For each SISO data set only two plots are presented, showing the dependency between the two system outputs and the input in question. For the MIMO data four plots are presented as both inputs are relevant.

Minimum phase system As we want to control the level of the lower tanks around 13 centimeters high, we start by setting the mean of the input signal so that the these tanks have a little bit more water than half of their capacity. By setting the mean to 3 volts we reach an operation point of approximately $(h_1, h_2, u_1, u_2) = (12.4, 12.7; 3.0, 3.0)$. To adjust the amplitude of the signal we had to make a compromise between the signal-to-noise ratio and the system nonlinearities. After looking at the system's response to a square wave of different amplitudes, an amplitude of 0.15 volts proved to be a good choice as it enable us to position the lower tank levels between 10 and 16 centimeters high while maintaining a certain level of linearity. For the sampling

period, and based on a rule of thumb that says for control purposes it should be chosen such that $\omega_r h = 0.2 - 0.6$ [4], we thought that a value ten times smaller would be well suited for the identification experiments. So a sampling time of 1s is chosen. Concerning the PRBS period, we thought of a 60s period since we are dealing with a system with time constants that vary between 20 and 80 seconds.

To be able to identify the system using SISO methods we performed two experiments (SIMO experiments) where we excited the system by sending to the inputs, one at a time, a PRBS wave with the characteristics stated before. The amount of data collected was 3600 points, and is used both for model identification and validation. Two plots showing the data collected can be seen in Figures 7 and 8.

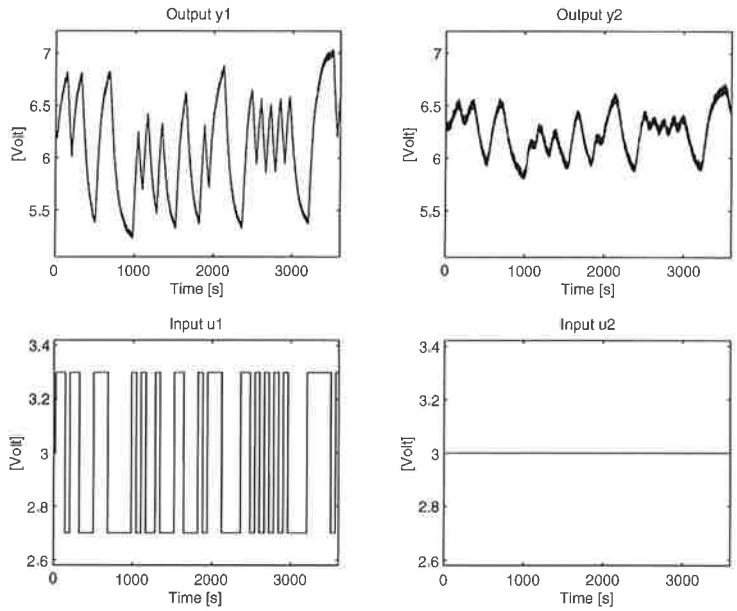

Figure 7 SIMO experiment using a PRBS signal on input u_1 while input u_2 is kept constant (minimum phase setting).

In the MIMO experiments we excited the system with two uncorrelated PRBS signals with the same characteristics of the ones used in the previous experiments. This data is used for two purposes, first to perform MISO and MIMO identification using ARMAX and state-space models, and second to compare the ability of the different models we've got in predicting the system's response. The data is shown in Figure 9.

Concerning the quality of the data collected in these three experiments, the coherence function estimates for the SIMO and MIMO data are presented, see Figures 10 and 11. From the upper plots in Figure 10 (coherence function estimates for the data collected in the first experiment) we notice a more stronger dependency between input u_1 and output y_1 than with output y_2. This result is not surprising as for this specific setting, we have values of γ_1 and γ_2 greater than 0.5, meaning that more than half of the water pulled by each pump goes to the lower tanks. Thus, influencing more the level of the lower tank with which the pump is connected (y_1), than the other one (y_2). The same result can be seen in the lower plots in the same figure, but this time the influence is bigger in output y_2 rather than in y_1, as in this case input u_2 is considered. The coherence function estimates for the data collected in

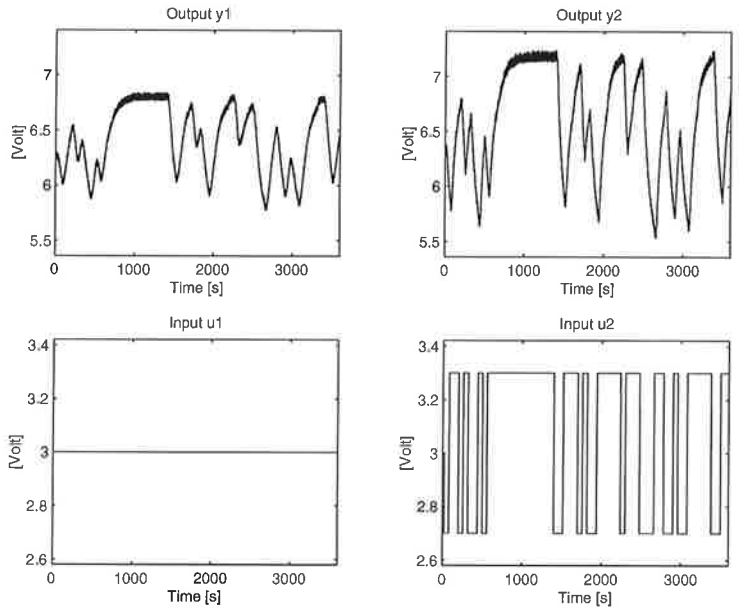

Figure 8 SIMO experiment using a PRBS signal on input u_2 while input u_1 is kept constant (minimum phase setting).

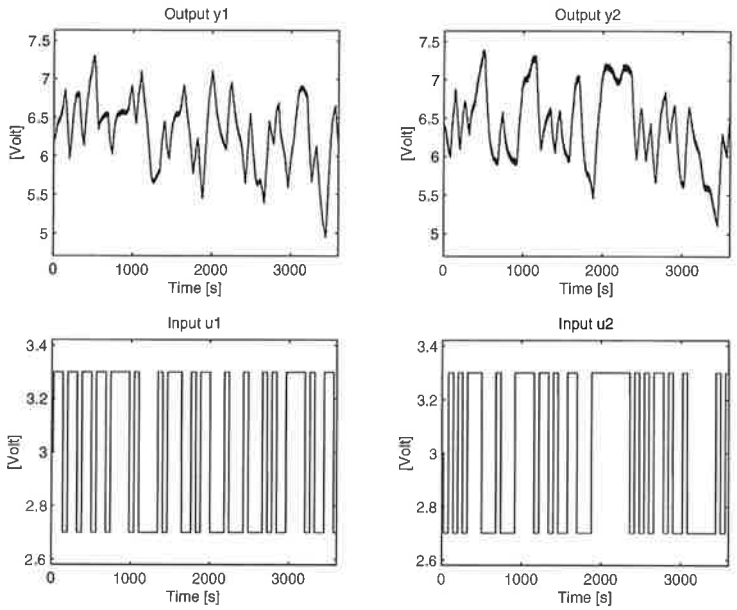

Figure 9 MIMO experiment using two uncorrelated PRBS signals as inputs (minimum phase setting).

the third experiment is shown in Figure 11. Here we notice that the coherence function estimate for the cross-couplings, u_1/y_2 and u_2/y_1, are much worse for the MIMO data. This result is not surprising since when we excite both system inputs at the same time, there will be loss of information respecting the dependency between each input and output. What can be said from the analysis of these plots is that a linear model extracted from this data sets is able to well approximate the real system for frequencies between 0Hz and 0.1Hz.

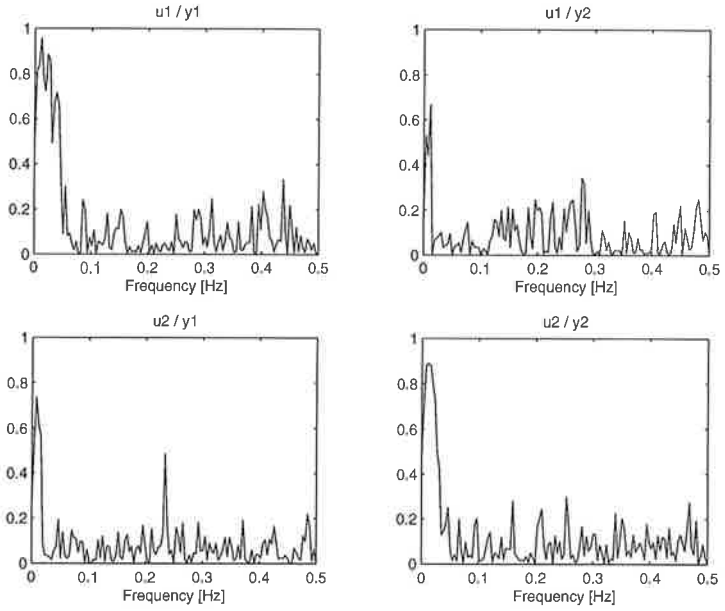

Figure 10 Coherence function estimates for the SIMO data (minimum phase setting). The upper graphs are related with the first SIMO data set, and the lower with the second.

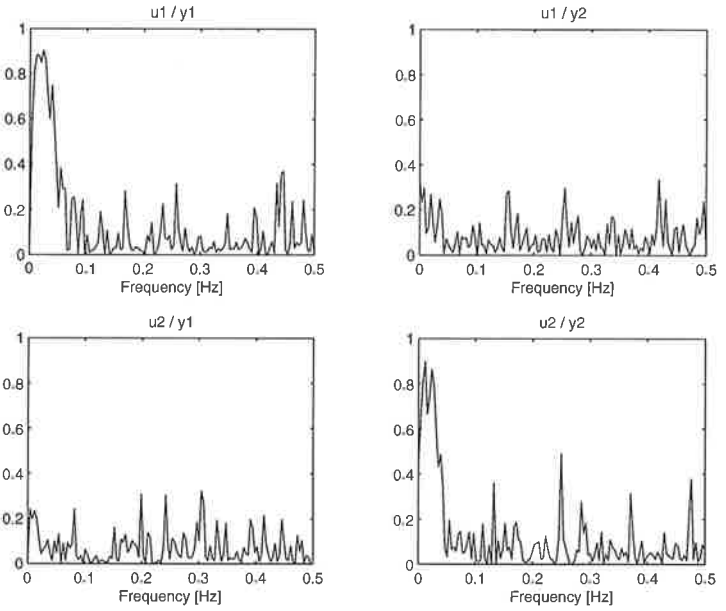

Figure 11 Coherence function estimates for the MIMO data (minimum phase setting).

Nonminimum phase system For this system setting we tried to have an operating point as close as possible to the one we got in the previous setting. After changing the position of the valves in order to have less flow going to the lower tanks and thus a nonminimum phase system, we adjust the mean of the input signal to 3.15 volts and get an operating point of $(h_1, h_2, u_1, u_2) = (12.6, 13.0; 3.15, 3.15)$. The amplitude of the wave is set to 0.3 volts so the changes in the level of the lower tanks be more or less 6 centimeters. The sampling period as well as the PRBS period used are the same as the ones

used for the minimum phase setting, 1s and 60s, respectively.

The SIMO experiments are similar to the experiments done for the minimum phase setting with the exception of the input signal, which has different characteristics. The amount of data collected is also 3600 points, and is shown in Figures 12 and 13.

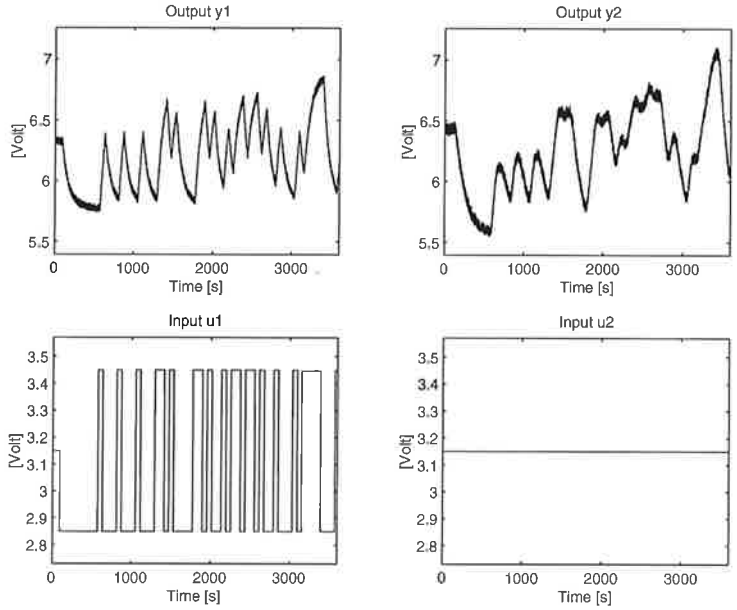

Figure 12 SIMO experiment using a PRBS signal on input u_1 while input u_2 is kept constant (nonminimum phase setting).

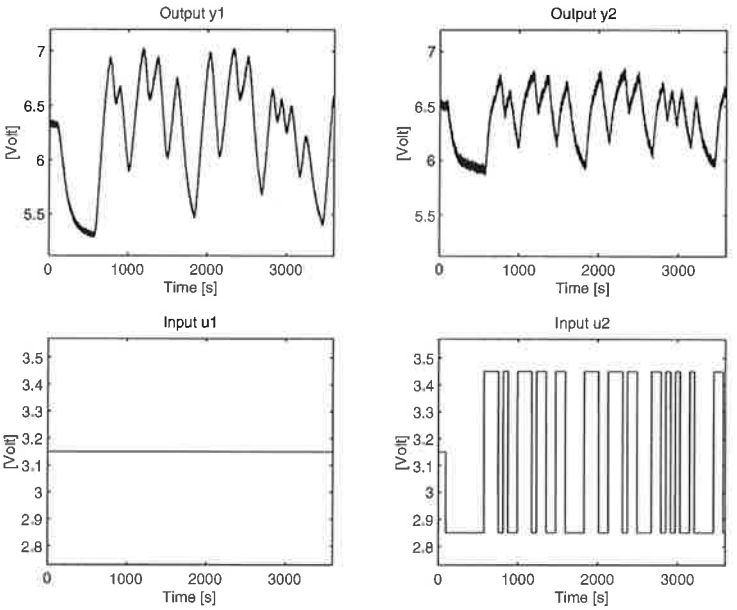

Figure 13 SIMO experiment using a PRBS signal on input u_2 while input u_1 is kept constant (nonminimum phase setting).

With respect to the MIMO experiment, it is also similar to the experiment done for the minimum phase setting. The data collected is shown in Figure 14.

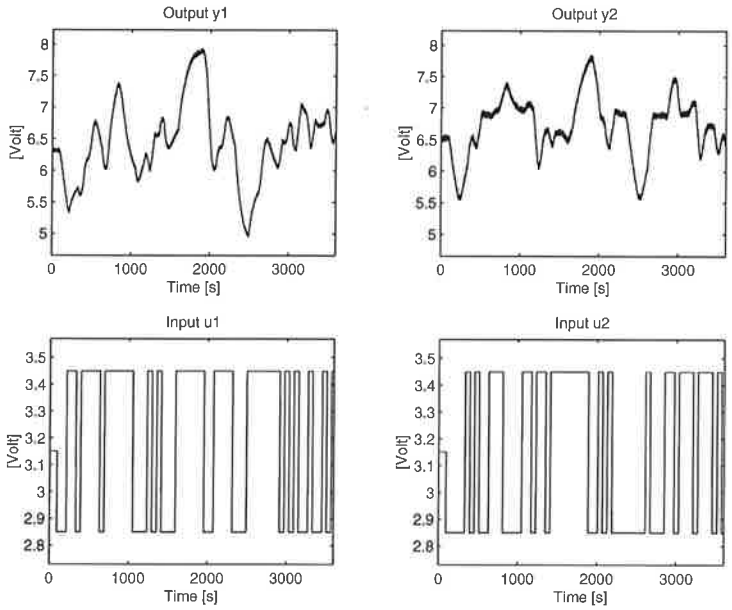

Figure 14 MIMO experiment using two uncorrelated PRBS signals as inputs (nonminimum phase setting).

Concerning the quality of the data it seems that there are no anomalies. From the coherence function plots for these data set, shown in Figures 15 and 16, we can say that the data is good enough to identify a linear model that can well approximate the real system for frequencies between 0Hz and 0.05Hz. In the MIMO data coherence estimate we notice the same effect for the cross-coupling pairs that was seen in the MIMO data collected from the minimum phase system. Comparing the minimum phase setting results with these, we notice that a model derived from those data sets is able to approximate the real system in a broader frequency range than one derived from these data sets.

3.3 Minimum Phase System

Before attempting to identify the system using parametric models (ARMAX and state-space), we used spectral estimation based on Fourier transform operations to get an estimated frequency response of the system. We used the SIMO data to do the spectral analysis and obtain the estimated frequency responses for each of the four transfer function involved. The length of the Hamming window used was 720 points. These responses were used to have an idea of the models frequency response derived from the data sets, and also to check the SISO parametric models.

In the ARMAX models category we started by identify some ARX models from the different types of data collected. We used three methods based on SISO, MISO and MIMO identification. For the models with an ARX structure the SISO identification method proved to be the best one. The influence of the delays in these models is minimal, and all of them have a delay of zero. These models extract the process dynamics but they are generally worse than the models with an ARMAX structure.

For the ARMAX structure models we are limited by the Matlab's function used, to two identification methods: SISO and MISO. The responses of these models are better than the ARX models. The match between the measured

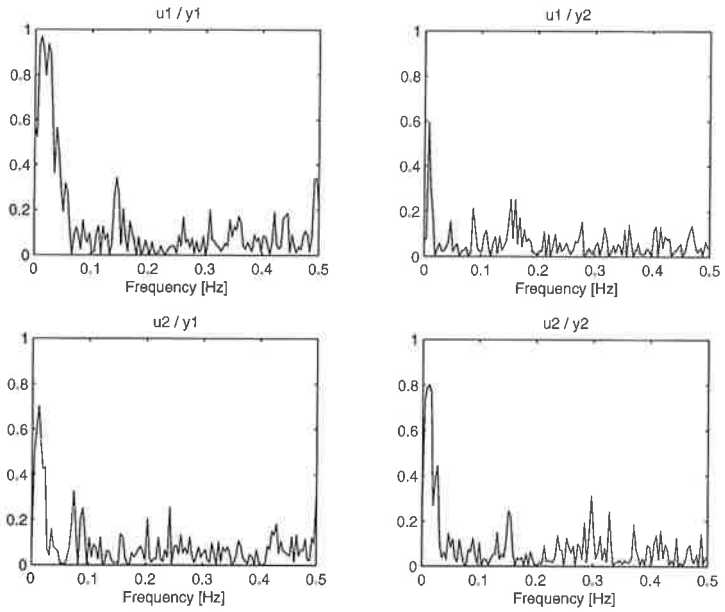

Figure 15 Coherence function estimates for the SIMO data (nonminimum phase setting). The upper graphs are related with the first SIMO data set, and the lower with the second.

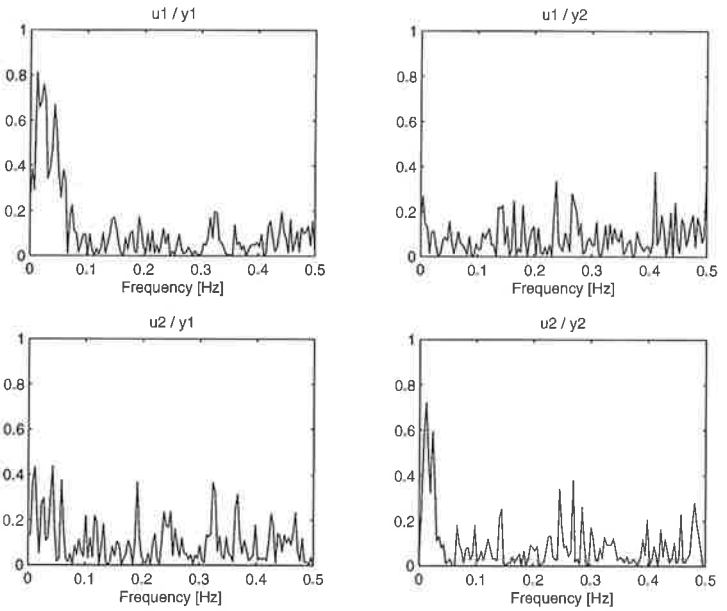

Figure 16 Coherence function estimates for the MIMO data (nonminimum phase setting).

and the model output is almost perfect for the SISO model, and a little bit worse for the MISO model. As the ARMAX model identified using the SISO data is the best of all the models, including the state-space subspace models, we present this model by showing the model transfer function matrix and some plots of its outputs. Some properties of the model are also stated and whenever possible the order of the model is reduced and the new transfer function matrix shown.

The identified transfer function matrix for the ARMAX model is

$$G(s) = \begin{bmatrix} \frac{3.3(1+1.0s)}{(1+93s)} & \frac{2.0(0.88s^2+1.4s+1)}{(1+43s)(1+58s)} \\ \frac{-1.6(1+1.8s)(1-4.1s)(0.27s^2-0.16s+1)}{(1+43s)(1+86s)(0.15s^2+0.0255s+1)} & \frac{3.1(0.88s^2+1.5s+1)}{(1+4.2s)(1+89s)} \end{bmatrix}$$

The correlation functions of the residuals and the cross correlation functions between the residuals and inputs have almost every point inside the 99% confidence interval. The loss function and Akaike's FPE are always less than $1.0E-5$. As there is a considerable difference in the time constants of some of these transfer functions, is reasonable to look for a reduced order model that doesn't affect seriously the input-output behavior. Because some of the heuristic model reduction methods have serious shortcomings, the method used is based on the Grammian-based balanced realization. The reduced order model transfer function matrix is

$$G(s) = \begin{bmatrix} \frac{3.3(1+0.21s)}{(1+91s)} & \frac{2.0(30s^2-2.8s+1)}{(1+26s)(1+77s)} \\ \frac{-1.0(1-14s)(1+40s)}{(5000s^2+122s+1)} & \frac{-3.1(1-3.0s)}{(1+88s)} \end{bmatrix}$$

with transmission zeros in $z = 0.32 \pm 0.70i$ and $z = -0.038$. Note that we have a pair of complex conjugate transmission zeros with positive real parts. The response of this model is shown in Figure 17.

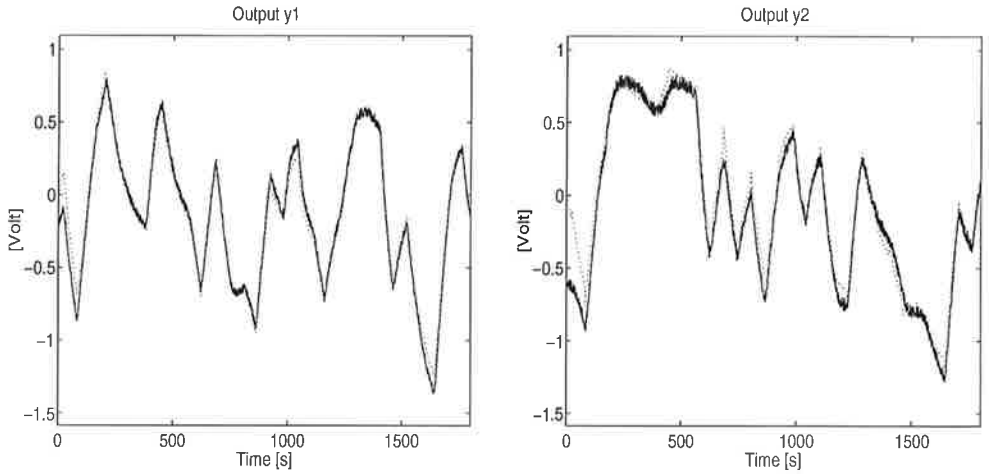

Figure 17 Measured (solid) and simulated (dotted) outputs for the ARMAX reduced order model (minimum phase setting).

For the state-space models the identification was done using subspace methods, using the matlab function N4SID. Because the function doesn't take in consideration the delays, we had to write some code in order to include them. Generally no improvements were obtained by including delays in the models. The best model identified using this method has a degree of three and is almost as good as the ARMAX model. The cross-validation with the real data was the only criteria used to select the best model of this type. For further information on the state-space subspace model and its response see [12].

3.4 Nonminimum Phase System

For this system setting, we also used spectral analysis to have and idea of the frequency response of each transfer function. The Hamming window used had the same size of the one used for the other setting, 720 points.

For ARX models we notice that they roughly capture the dynamics of the system, and when compared to the models we identified for the minimum phase setting they are much worse. Again, the MIMO models tend to be worse than the SISO models.

The ARMAX models identified are much better than the ARX models, and a little bit better than the state-space subspace models. For the ARMAX model presented the correlation functions of the residuals and the cross correlation functions between the inputs and residuals are very good. Most of the points from these curves are inside the 99% confidence interval. The value of the loss function and Akaike's FPE is always lower than $1.0E-5$. The identified transfer function matrix is

$$G(s) = \begin{bmatrix} \frac{1.9(1+1.0s)}{(1+88s)} & \frac{4.5(s^2+1.4s+1)}{(1+36s)(1+138s)} \\ \frac{2.9(1+0.072s)(1+1.0s)(0.42s^2+0.46s+1)}{(1+40s)(1+167s)(0.14s^2+0.021s+1)} & \frac{2.0(1+0.99s)}{(1+132s)} \end{bmatrix}$$

By calculating a Grammian-based realization and then reducing the model, we got the model shown below, with transmission zeros in $z = -0.52$, $z = 0.12$ and $z = 0.013$. This system has two zeros located in the right half-plane.

$$G(s) = \begin{bmatrix} \frac{-1.9(1-1.3s)}{(1+84s)} & \frac{-3.5(1-9.0s)(1+18s)}{(1+51s)(1+99s)} \\ \frac{-4.0(1-20s)(1+32s)}{(1+38s)(1+263s)} & \frac{-2.0(1-0.63s)}{(1+128s)} \end{bmatrix}$$

The response of the model is shown in Figure 18.

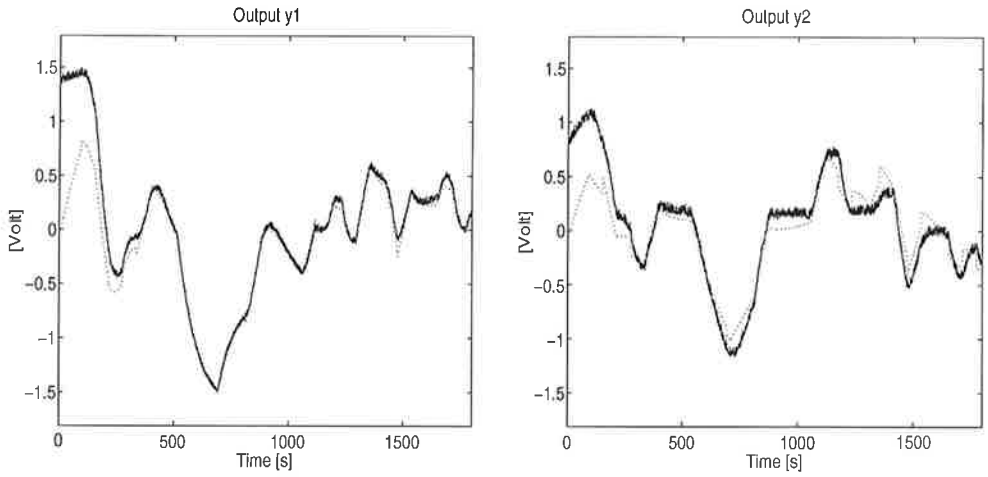

Figure 18 Measured (solid) and simulated (dotted) outputs for the ARMAX reduced order model (nonminimum phase setting).

For the state-space subspace identification different order models with different delays were tried. Using the cross-validation with the real data as the criteria for choosing the best one, we come up with a fourth order model with a delay of two. The output of this model is similar to the ARMAX model presented. The model and its response can be seen in [12].

3.5 Summary

In general, we notice when identifying the system using an ARMAX structure that the models derived from SISO data are often a better representation of the input-output behavior than the ones derived from MIMO data. This could have

to do with the quality of the MIMO data and probably with interferences when we excite both of the inputs at the same time. For the MIMO identification using subspace methods the results were almost the same as the ones we got for the ARMAX models, which were very good.

Comparing the ARMAX models identified with the physical models derived in Section 2 we notice, has expected, that the identified models are better. But the difference between them is small.

About the two different system settings, for which we performed two system identifications, it was noted that in the nonminimum phase case the approximation of the model output to the output of the real system was always worse than for the other case. This result is not surprising since from the analysis of the coherence spectrum we were expecting a less accurate model in a narrow frequency range.

4. PID Control

4.1 Implementation

Decentralized PID control is one of the most common control schemes for interacting MIMO plants. The main reason for this is its relatively simple structure, which is easy to implement. For these type of controllers the number of tuning parameters is $3n$, where n is the number of inputs and outputs. Despite the wide popularity of decentralized PID control the number of tuning methods is limited.

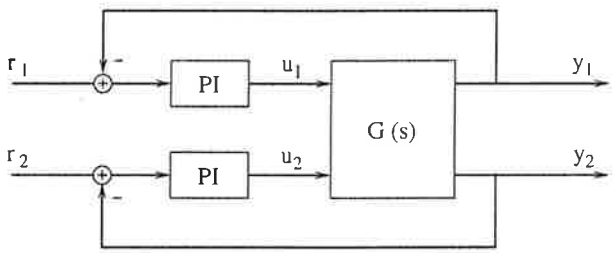

Figure 19 Block diagram of the decentralized PI control system.

In the quadruple-tank process we are about to use decentralized PI control in order to control the level of the lower tanks for the two different system settings we have. The scheme used is depicted in Figure 19, and as we can see the pattern used on how the signals should be paired is $1-1$, $2-2$.

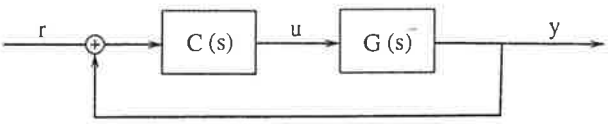

Figure 20 Compact block diagram of the control system, in matrix form. C(s) is a diagonal matrix.

It is clearly seen that the control signal u_i $(i = 1, 2)$ depends only on the error e_i which is the characteristic of decentralized control. If we instead use the matrix form shown in Figure 20, the control matrix $C(s)$ is a diagonal

matrix given by,

$$C(s) = \begin{bmatrix} C_1(s) & 0 \\ 0 & C_2(s) \end{bmatrix}$$

where $C_1(s)$ and $C_2(s)$ are PI controllers parameterized as

$$K_p(1 + \frac{1}{T_i s})$$

The number of controller parameters is four, two for each PI controller. The manual tuning method we use is basically pole placement followed by some refinements in the parameter values as we test the controller against the simulations. Before testing the controllers on the real system, we test them using the nonlinear and the ARMAX models derived in Sections 2 and 3, respectively. The close-loop response and the control signals of these models are shown together in order to compare the similarity between them.

4.2 Minimum Phase System

For the minimum phase system we notice, by analyzing its open loop step response, that the interactions between the signals are significant and that the pattern we are using for the pairing of the signals is probably the best one.

The controller parameters we get for the PI controller in the first control loop $(u_1 - y_1)$ are,

$$\begin{aligned} K_p &= 3.0 \\ T_i &= 30 \end{aligned}$$

and for the second control loop $(u_2 - y_2)$,

$$\begin{aligned} K_p &= 2.7 \\ T_i &= 40 \end{aligned}$$

The step responses of the closed-loop system and the control signals, using the nonlinear and linear ARMAX models for this matrix controller, are shown in Figure 21 for a step in reference signal one, and in Figure 22 for a step in reference signal two. The amplitude of these steps is 1 volt, which represents a change of approximately 2 cm in the tanks level.

From these plots we see that there is always a small disturbance in output signal y_2 for a step change in reference one, and in output signal y_1 for a step change in reference two. The performance of this controller is very good. The system is well damped, with an overshoot less than 10%, and a settling time of about 60 seconds.

When we test the controller on the real system, applying again a step with the same amplitude (1 volt), we get the results depicted in Figures 23 and 24.

Comparing the results obtained using the real system with the ones we got for the models, we notice a great similarity between them. Apart from a small increase in the overshoot we can say that the responses are identical.

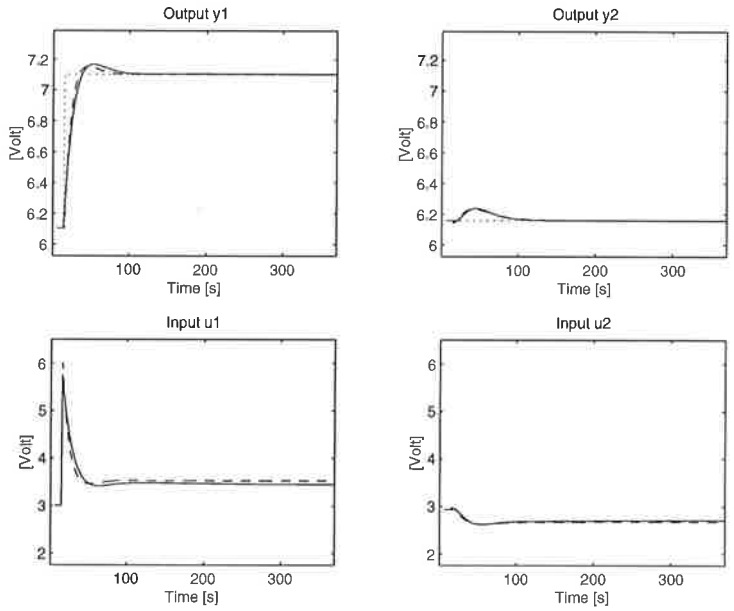

Figure 21 Closed-loop response for a step change of amplitude 1 volt in reference signal r_1, using the nonlinear (dashed line) and the linear ARMAX (solid line) models as a representation of the real process (minimum phase setting).

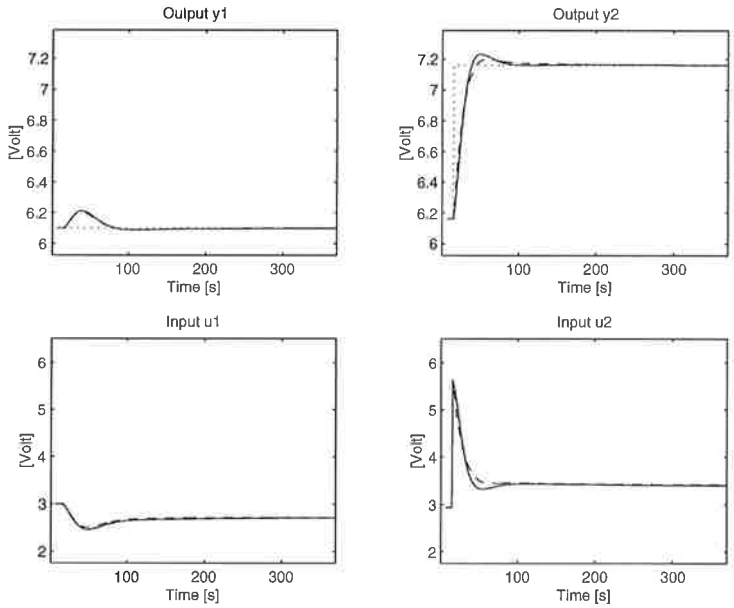

Figure 22 Closed-loop response for a step change of amplitude 1 volt in reference signal r_2, using the nonlinear (dashed line) and the linear ARMAX (solid line) models as a representation of the real process (minimum phase setting).

4.3 Nonminimum Phase System

By doing some simple experiments with the nonminimum phase system and analyzing its open loop step response, we can notice a even bigger interaction between the signals than for minimum phase system. If we look at the pattern we are about to use for pairing the signals, we could say that probably this is not the best choice. We reach this conclusion because the influence of a step change in control signal u_1 is more noticeable in output y_2 rather than in

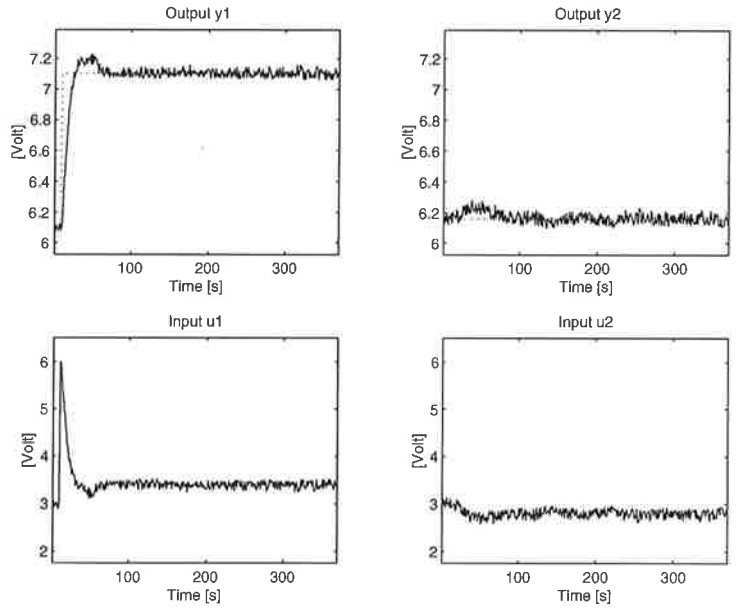

Figure 23 Closed-loop response for a step change of amplitude 1 volt in reference signal r_1, using the real process (minimum phase setting).

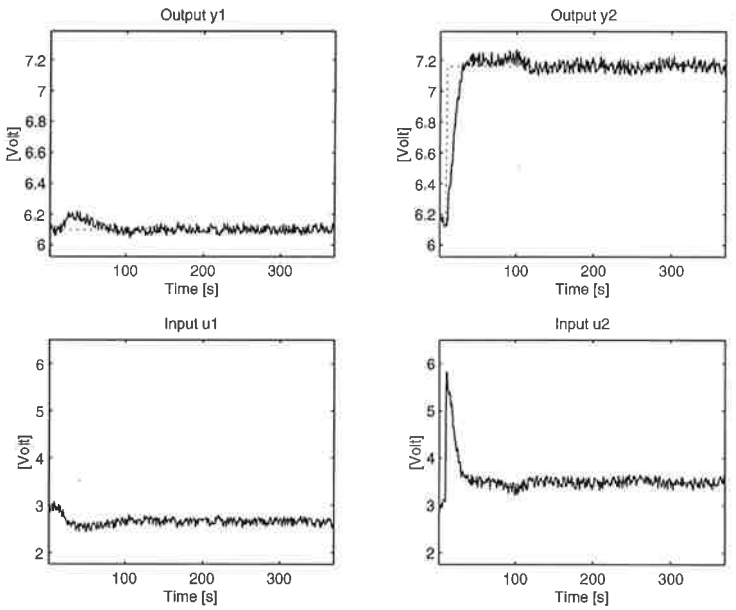

Figure 24 Closed-loop response for a step change of amplitude 1 volt in reference signal r_2, using the real process (minimum phase setting).

output y_1 (the correlation between u_1 and y_2 is higher than between u_1 and y_1), and vice-versa for control signal u_2.

The controller parameters we get for the PI controller in the first loop $(u_1 - y_1)$ are,

$$\begin{aligned} K_p &= 1.5 \\ T_i &= 110 \end{aligned}$$

and for the second loop $(u_2 - y_2)$,

$$\begin{aligned} K_p &= -0.12 \\ T_i &= 220 \end{aligned}$$

The step responses of the closed-loop system and the control signals, using the nonlinear and linear ARMAX models for this matrix controller, are shown in Figure 25 for a step in reference signal one, and in Figure 26 for a step in reference signal two. Again, the amplitude of the steps is equal to 1 volt (≈ 2 cm). Note that the time scale in the nonminimum phase system plots is 10 times bigger than the one used in the minimum phase system plots.

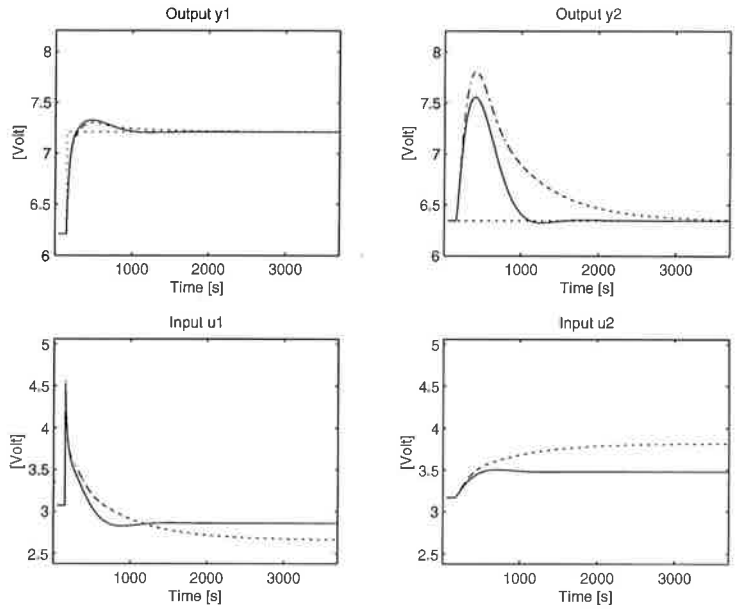

Figure 25 Closed-loop response for a step change of amplitude 1 volt in reference signal r_1, using the nonlinear (dashed line) and the linear ARMAX (solid line) models as a representation of the real process (nonminimum phase setting).

We experience a great difficulty to control the nonminimum phase system. Difficulty which was even higher than we were expecting due to the nonminimum phase properties. The stability margin for the controller parameters is narrow and, in order to have a stable closed-loop system, we had to use a negative gain for the PI controller in the second loop. Looking at Figure 25 we notice a big disturbance in output y_2 when there is a step change in reference one, as opposed to the relatively small overshoot in output y_1, about 15%. For a step change in reference two, we notice a disturbance on the form of an undershoot in output y_1, which is much small in magnitude than the disturbance in y_2 for a the step change in reference one. We also notice an inverse response (y_2) to a step input in the initial phase due to the PI controller negative gain in the second loop. The response of the closed-loop system is very slow, it takes about 800 seconds to reach the steady-state. One other thing that is evidenced by the plots is the disparity between the two models closed-loop response for output y_2.

If instead of using the process models we use the real system, the closed-loop response will look like the one shown in Figures 27 and 28. Comparing these plots with the models response, we notice that the real system closed-loop response is very similar to the ARMAX model response. As a mater of

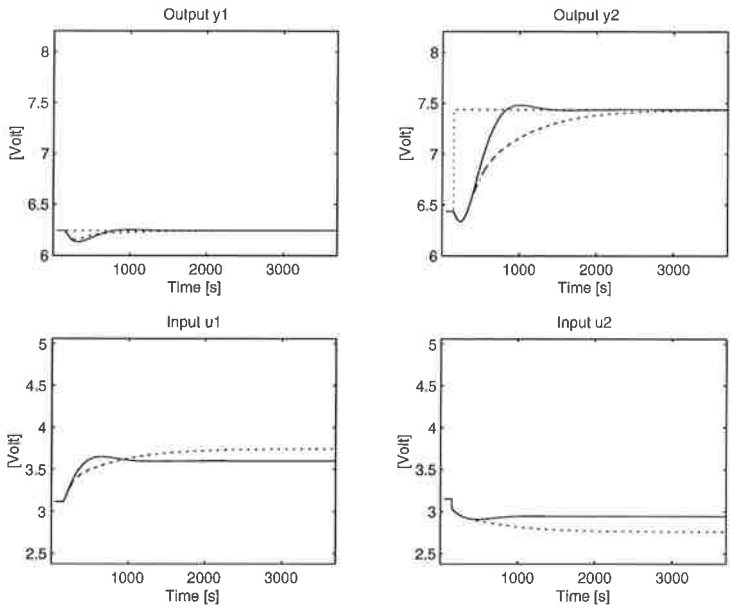

Figure 26 Closed-loop response for a step change of amplitude 1 volt in reference signal r_2, using the nonlinear (dashed line) and the linear ARMAX (solid line) models as a representation of the real process (nonminimum phase setting).

fact, they are equal except for the disturbance in output y_2 when we add a step change in reference one, which is bigger in the real system closed-loop response. With respect to the nonlinear model, the overall results are worse than for the ARMAX model and for the real system. Again, it is more noticed in output y_2.

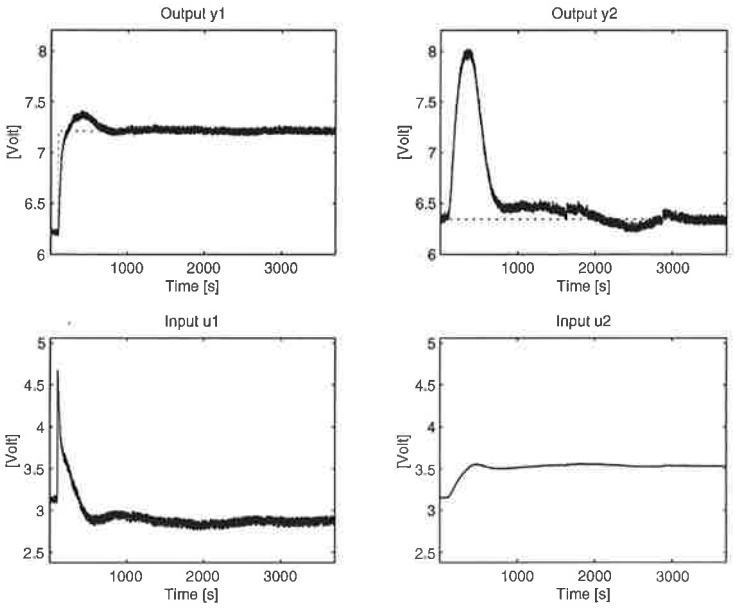

Figure 27 Closed-loop response for a step change of amplitude 1 volt in reference signal r_1, using the real process (nonminimum phase setting).

The not so good step responses for this controller were not surprising at all as we are in presence of a MIMO system with a RHPT zero located in

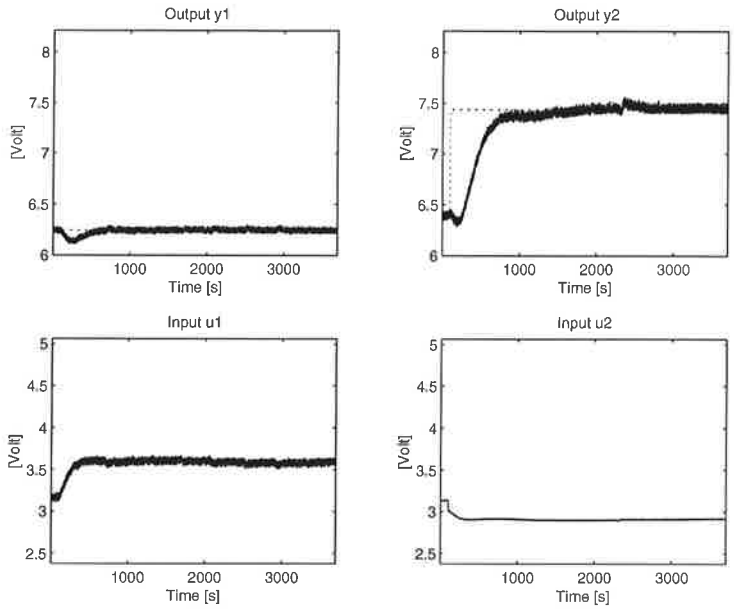

Figure 28 Closed-loop response for a step change of amplitude 1 volt in reference signal r_2, using the real process (nonminimum phase setting).

0.0133. With a RHPT zero so close to the origin the difficulty to control the system is much higher. As we only used decentralized PI control assuming that the two system loops don't interact, we could expect better results if we take advantage of interactive type control using for the effect a full matrix controller.

4.4 Summary

In this section, using decentralized PI control in the two different system settings, the minimum and the nonminimum phase, we experienced as expected much more difficulties in controlling the second system. These difficulties were even bigger than expected at the beginning as our nonminimum phase system has a RHPT zero very close to the origin, and the interaction between the two system loops are much bigger in the nonminimum phase system. As only decentralized PI control was tried, and knowing the effects of the RHPT zeros and the interaction between loops in such closed-loop systems, we could expect an improvement in controller performance if we instead take advantage of a full matrix controller that could handle more properly the interactions present in the system. The controllers used are by no means the best or the optimal controllers, as our main goal was to prove the difficulties in controlling the process when in presence of a nonminimum phase system.

5. The User Interface

In order to simplify the interaction between the user and the real process a graphical user-friendly interface was developed. As we were looking for a powerful but at the same time easy to use tool, we think that the right thing to do is to spend the most part of this section explaining the possibilities of the interface and the way to do things, instead of trying to explain all the in-depth technical aspects. Of course, we will mention how things are

implemented internally, whenever we think that is important for the user to know.

This section starts with a brief introduction to some of the tools used in the building of the user interface and the major goals we had prior to its building. The following three subsections explain what the user can and can not do with the interface, and the way to do it. There, we introduce the user with the basics of using the graphical interface and then we describe the operations the user can perform in the Main and Setup windows. By the end of the section, we refer some technical aspects of the implementation that could be relevant for the user. As there were some ideas for the interface that were not implemented, in the final section of this report – Section 6, we state some developments that, in the future, could be made to the interface.

5.1 Introduction

The user interface was developed using an Intel PC platform running Microsoft Windows NT 4.0 operating system (see Appendix A.1 for hardware and software requirements). The machine was connected with the real plant by means of an AD/DA converter local to the machine, with four inputs and two outputs. A scheme on how to connect the cables can be seen in Appendix A.2. The software tools used were a Modula2 compiler and a common used tool to monitor and control industrial processes - InTouch 6.0, from Wonderware corporation.

The interface is composed of two separate parts: a Modula2 program that runs over the real-time kernel [6] developed at the department and which implements all the algorithms used (the process mathematical model, controllers, etc.), and an InTouch program used in the design of the graphical user interface. These two programs take advantage of the Dynamic Data Exchange (DDE) communication protocol to interact with each other. In order to synchronize and enable the effective communication a protocol was defined, but it's specifications are beyond the scope of this report.

We could not finish this introduction without mention the initial goals we wanted to address when we start to build this man-machine interface. Among other things that we were discovering over its development, the main goals were:

- the design of an intuitive and easy to use interface;
- the use of a modular structure in a way that further developments and the add of new features could be done with simplicity;
- let the user act on the real plant or on a simulation;
- allow the use of manual and automatic control; and
- add the possibility of collecting data for identification purposes and to analyze the controllers performance and the process behavior.

In the first two no big things changed as we were concerned with the simplicity for both the end user and the people who will add new features to the interface. On the other hand, in its functionality some other features were added. For example, the possibility to exchange data with Matlab 5.0 (both process data and controller information), the possibility to simulate some real disturbances when running against the simulation of the process (add water to the tanks, change the cross-section of the tanks by introducing objects in them, etc.).

5.2 The Environment

When the user starts up the application by selecting the *Tank* application from the windows Start button, or by running the batch file `tank.bat` from the application directory, two main action take place. First the Modula2 program (*Tank*) is started in background, as you can notice by the icon in Windows NT taskbar, and then the InTouch run-time viewer (*View*) follows which runs the code that enables the graphical interface. After this initial startup phase the user is automatically dropped into one of the two application windows, in this case the Main window.

The user interface is composed by two windows: the Main window and the Setup window. The first is used more often as it allows the user to perform the basic functionalities included in the interface. As opposed to this one, the Setup window is used only to set parameters that the user doesn't need to change every time, like the sampling time. To improve the functionality, some parameters can be set in both of the windows, as it is the case of some controller parameters.

To improve even more the functionality, the user is also provided with the ability of moving back and forth by means of scheme where the confirmation, setting and restore of the default values is possible in all occasions. Because of that, some assumption were made about closing windows, which follows:

- almost every window that opens by clicking in an object, usually a button, can be closed by clicking in the same object for a second time;
- when clicking in the popup window title, that window is automatically closed; and
- in some windows the use of a button that usually means accepting the values just set, automatically closes it.

These methods are common to all windows, with the exception of the Exit window due to consistency reasons. This scheme lets the user cancel the changes made by closing the window using one of the first two methods described above, or to actually confirm them by using another button, usually a Set button, that automatically closes the window in question (third method).

Another thing that worth to be mention is the use of default values for all the parameters that can be changed by the user, and the possibility to return to these values every time the user wishes to. This can be achieved in two different ways: globally, for all the interface parameters by rereading the `default.par` file; and locally, for the parameters of a controller, input signal wave, etc., using the Default button. In order to know when the parameter values are the default ones, they are shown in two different colors. For default values the color used is black, while for the values that the user changed and that differ from the default ones, the color is gray.

The default values mentioned are the values with which the interface is initialized. Which could be the values included in the Modula2 code , referred as *preset* values, or the values contained in the `default.par` file. The application will always use the values from this file as default, except when the file doesn't exist.

As a final remark, an advise on how to close the application. The user should not use the Windows NT close application method in the InTouch View program, as using it the Modula2 program that runs in background isn't stopped and has to be stopped manually by the user. The appropriate way to

close the application is to used the Exit button, present in both the Main and Setup windows.

5.3 The Main Screen

The Main window will be for sure the screen with which the user will interact more often. In this window the user is presented with different types of objects, some of them are merely to show information (status) about the real/simulated process, while others allow the user to act directly or indirectly on it. In the first group we include the two plots and the interactive image that represents the process, located respectively in the lower-left and upper-right regions of the window. In the other group we have the mode selection buttons and the controller setting in the upper-left region, and finally the input signal setting in the lower-right region of the window. A captured image of the Main window can be seen in Figure 29. For a better understanding of the possible actions that can be performed within this window, a short description of each object will be made and the procedure to do some usual actions is referred.

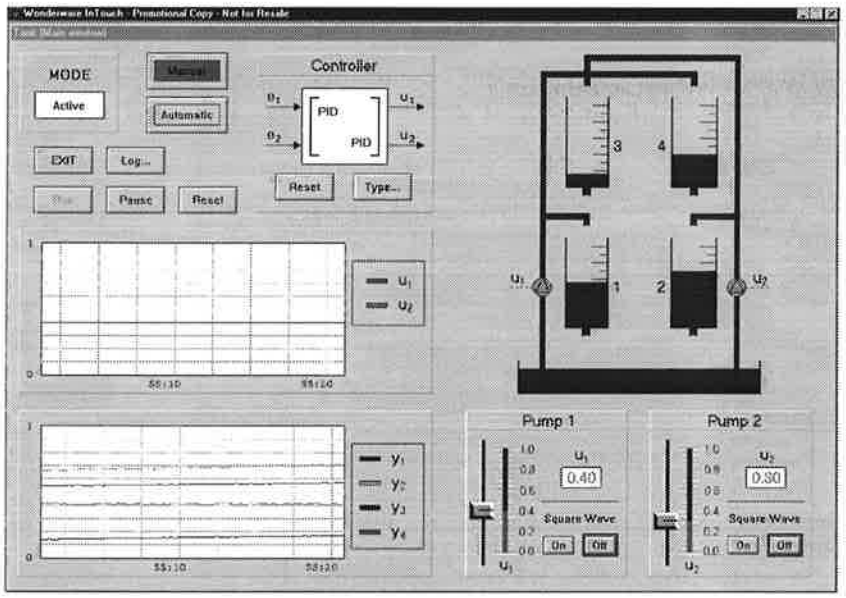

Figure 29 The Main window can be used to act on either the real plant or the simulation.

Modes In the interface we have three different mode types: the mode that the user specifies inside the Mode box, allowing him to choose whether he wants to act on the real process or on a simulation of it, which we call *operation* mode; the *state* mode that corresponds to the actual state of the application, either running or stopped; and the *control* mode.

To change the operation mode the user just has to click in the Mode box which will present a pulldown menu with three different operation modes, instead of two as it was mentioned.n This is because there was the need to add a Setup window where the user could set some parameters that are not changed so frequently. But truly, when the user selects the Setup mode, he is still acting in the same "process" as he was before, whether it is the real process or the simulation. So, this Setup mode only presents the user with another window without actually changing anything else. The three options for the operation mode are:

Simul, which stands for simulation mode and specifies that the user is acting on a simulation of the process;

Active, this mode specifies that the user is acting on the real process; and

Setup, shows the Setup window without changing the previous operation mode.

Another mode type is what we call the state mode which has three possible states: Run, Pause and Reset. The Run state means than any changes made in the interface parameters will have immediate repercussion on the real process or the simulation. The Pause state have different meanings depending in which operation mode is the user running. When in Active mode it just keeps the value of the input signals, while in Simul mode the simulation is paused which means that all the simulation states (input signals and tank levels) are kept. The Reset state just reset all the states and then enters the Pause state. To select the different states the user just has to click in the appropriate button, located above the plots.

The last mode type used is designated by control mode and can be set to one of two possible choices: Manual or Automatic. In Manual mode the input signal specified by the user (see Input Signals in this subsection) corresponds to the control signal – u, while in Automatic mode it corresponds to the reference signal – ref. A scheme for bumpless changes between Manual and Automatic modes is implemented. For more information see Internal Implementations subsection.

The default values for the operation, state and control modes are respectively: Simul, Pause and Manual. These are the modes in which the interface is started.

Input signals The signals fed to the process, directly or indirectly through the controller, can be specified in the lower-right region of the window. The labels that the user sees vary according to the selected control mode. In Manual mode the signals specified are the control signals, u_1 and u_2 (the process inputs), while in Automatic mode they are the reference signals, ref_1 and ref_2 (the controller references). In the Main window the user can only specify the wave parameters but can not change the wave type. See The Setup Screen subsection for the different wave types and how to switch between them.

The parameters that define the input signal wave are: amplitude, period and mean. The last one can be set using either the slide bar or the text box, depending on whether the user wants to progressively or abruptly change (step change) between the initial and the final value. The other two parameters, and the mean also, can be set in a popup window that is shown by clicking in the wave type. This window is shown in Figure 30. The period has different meanings according to the selected wave type, see Reference Configuration in the next subsection.

For simplicity, the user is allowed to turn the wave on and off by clicking in the appropriate button. This way he can switch between a constant input signal equal to the mean, or the wave itself. The input signal waves one and two are completely independent, thus allowing the user to use different wave types on each of them.

Sometimes the input signal means are changed automatically. This happens when the user changes the control mode and is due to the scheme used to implement the bumpless changes between Manual and Automatic modes, already mentioned.

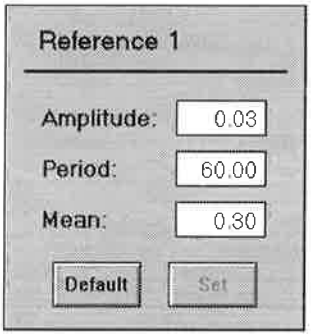

Figure 30 The Reference 1 parameters window.

Controllers In a box designated by Controller, located in the upper region of the Main window, the user can define the controller he wants to use. The controller in represented by a 2-by-2 matrix where the element ij represents the SISO controller between input j and output i. This notation is the same that has been used across this report. This matrix changes with the selected controller type and is used to change the parameters of each controller. There is also a Reset button, whose purpose is to reset the internal states of all the four controllers, and a Type button that allows the user to select the controller type.

The controller types implemented are decentralized and full-matrix PID controllers, and the same for general transfer function controllers (designated by General controllers). The selection of the controller type is done using the Type button that shows a popup window from where the user can select the appropriate one, see Figure 31. One assumption that was taken related with the switch between different controller types is that, the controller internal states are only kept when changing between decentralized and full-matrix controllers of the same type, otherwise all the internal states are reset.

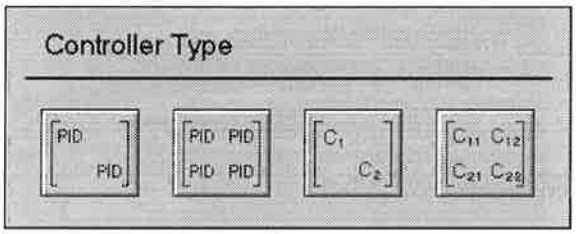

Figure 31 The Controller Type window.

To set the parameters of each SISO controller the user just has to click in that particular controller, located inside the controller matrix. By doing this, he is presented with a popup window that varies depending on the controller type, PID or General. The two possible windows are shown in Figure 32. For a PID controller the parameters are the usual ones for this type of controller: K, T_i and T_d. Other parameters like maximum derivative and integral reset time, can only be set in Setup window. The check boxes are used to activate or deactivate the controller integral and derivative parts. The Reset button as a similar effect to the Reset button mentioned above, but the scope of this one is different as it only resets the internal states of this particular controller and not all the four controllers. For a General controller the parameters are the coefficients of the transfer function involved plus the degree of the controller,

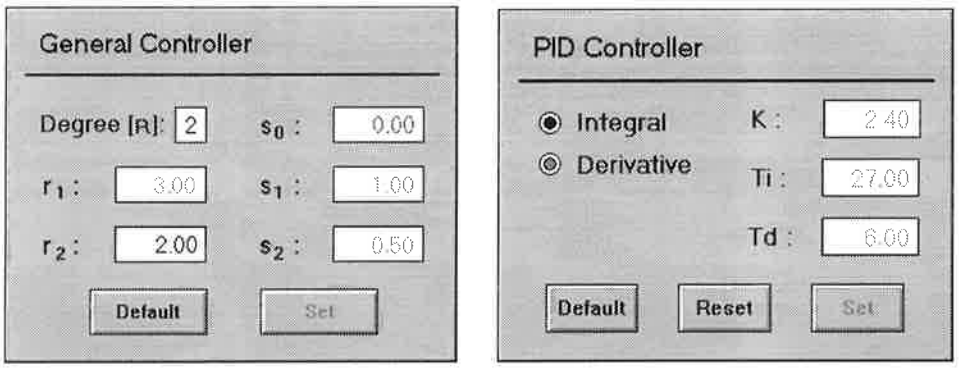

Figure 32 The General (left) and PID (right) Controller windows in Main window. The parameter values written in black are default ones while the others, in gray, were modified.

which can vary between 0 and 4. When the controller parameters change on-line an algorithm for bumpless parameter changes is responsible for providing a smooth transition. The algorithms implemented for the PID and General controllers are discussed in Internal Implementations subsection.

Process monitoring Some objects – an image of the process and two plots – were included in the interface to monitor the process state.

The process image is a thrust representation of the real process, and has some particularities that allow the user to see how the process state evolves. The image is composed of four tanks, two pumps and the tubes that represent the water path. The tanks are used to show, graphically, the amount of water in each of them and can be used to display some additional information by only clicking on them. The information provided, as it can be seen in Figure 33, is the operating point, the tank diameter, the time constant associated with the tank and the height (level). The height field in this popup window is also used, when running in Simul mode, to simulate disturbances equivalent to the act of adding water to the tanks on the real process. The value introduced in this field corresponds to the amount of water the user wishes to add. The size of the tanks shown in this image are proportional to their size, specified by the diameter value in Setup window. The size of the tanks only has implications on the simulation of the process. The pumps are used in the same way as the tanks to show the value of its input, see Figure 34. Finally, the tubes are used to provide information on whether there is or there is not water flowing in them, and to show the water path from each pump.

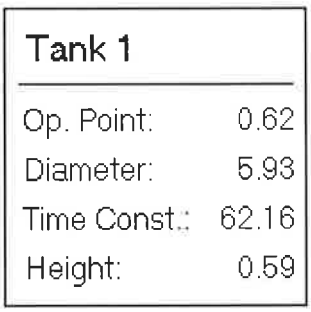

Figure 33 The Tank 1 information window (Main window).

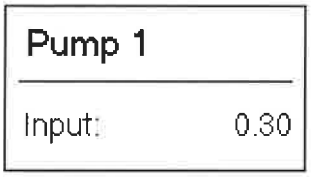

Figure 34 The Pump 1 information window.

The plots serve as a way of following the evolution of some process variables. The variables displayed in each of these plots can be selected by a plot type from the list of predefined ones. To change the plot the user only has to click on the variable label box, located on the right of the plot displaying area, and then select one of the plot types shown, see Figure 35. The current selection for the plot type is represented by a shaded box, while the other plot shown is written in gray, which also means that this isn't a valid option. The default plots used are u_1u_2 and $y_1y_2y_{1ref}y_{2ref}$ for the upper and lower plots, respectively. The sampling time (refresh rate) used is 50ms and is not affected by any changes in the sampling time value, in the Setup window.

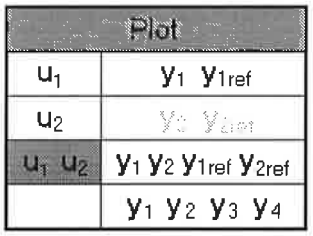

Figure 35 The Plot type window. The plot u_1u_2 is selected and the plot y_2y_{2ref} is being shown on the other graph image.

Logging In the interface we included some data logging facilities so the user can collect whatever process data he needs. All the parameters related with the logging are specified in the Logging window, shown in Figure 36, that can be seen by clicking in the Log button. Prior to the logging of data, some parameter values must be set. These include: the number of points to collect, with a maximum of 5000; the name of the file to write data to, which will have the extension `.log` and the variables the user wishes to log, that are specified by enabling the set boxes for those variables. By default, the logged variables are the inputs and the outputs of the process.

To start and stop the logging process the user uses the buttons designated by LogOn and LogOff, respectively. The state of the logging is shown by means of a status bar, in which a percentage is also shown. After the logging is finished, either by user will or because all data points were collected, the data can be saved to a file using the Save button. The file is of ASCII type and besides the data itself, it includes an header that gives some information about the meaning of the numbers stored in it. This file can be loaded into Matlab using the function `load4t.m`, included with the interface, or by using the *load* command, but this is not recommended since, for versions of Matlab under 5.0, an error occurs due to the presence of the header. A simple way to solve the problem, in case the user doesn't want to use the *load4t* function, is to remove the header using a text editor. Included with the interface there are

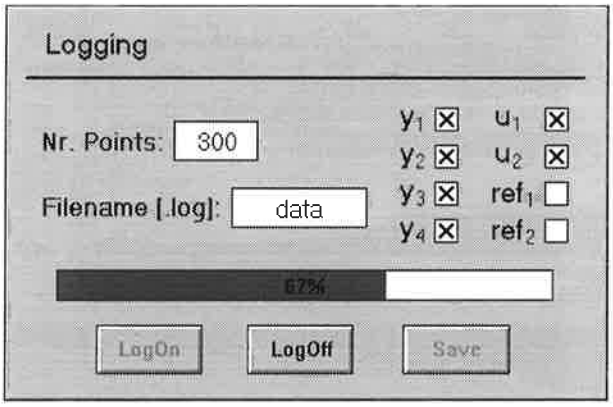

Figure 36 The Logging window is used to monitor the state of the logging and to set some parameters associated with it.

also other Matlab functions that simplify the treatment of this type of data. A sample of a log file is show below.

```
% TANK LOG-file
% Created on: Jun 20 00:30:50 1997
% Logged variables: y1 y2 u1 u2   TSamp: 1s

 0.622070  0.623535  0.300000  0.300000
 0.625000  0.623047  0.300000  0.300000
 0.623535  0.626465  0.300000  0.300000
 0.620117  0.628418  0.300000  0.300000
 0.623535  0.628906  0.300000  0.300000
 0.619629  0.627930  0.300000  0.300000
 0.619629  0.627930  0.300000  0.300000
 0.621094  0.624023  0.330000  0.300000
 0.620117  0.626465  0.330000  0.300000
 0.623535  0.626465  0.330000  0.300000
 0.625488  0.628906  0.330000  0.300000
 0.628418  0.628418  0.330000  0.300000
 0.625977  0.630371  0.330000  0.300000
 0.625488  0.624512  0.330000  0.300000
 0.626465  0.631348  0.330000  0.300000
 0.627930  0.630371  0.330000  0.300000
```

5.4 The Setup Screen

In the Setup window it is possible to set parameters that don't need to be changed so often, as it is the case of the sampling time. With a brief look at this window, shown in Figure 37, it's noticeable the division into small regions. It's these different parts that compose the Setup window that will be describe further on. But before that, it should be born in mind that by changing to this window the interface will still be running in either Simul or Active mode and that any changes will take effect immediately as if they were made in the Main window.

Sampling time The value of the sampling time is set in a text box in the top-left corner of the window. The lower bound for this parameter is 50ms

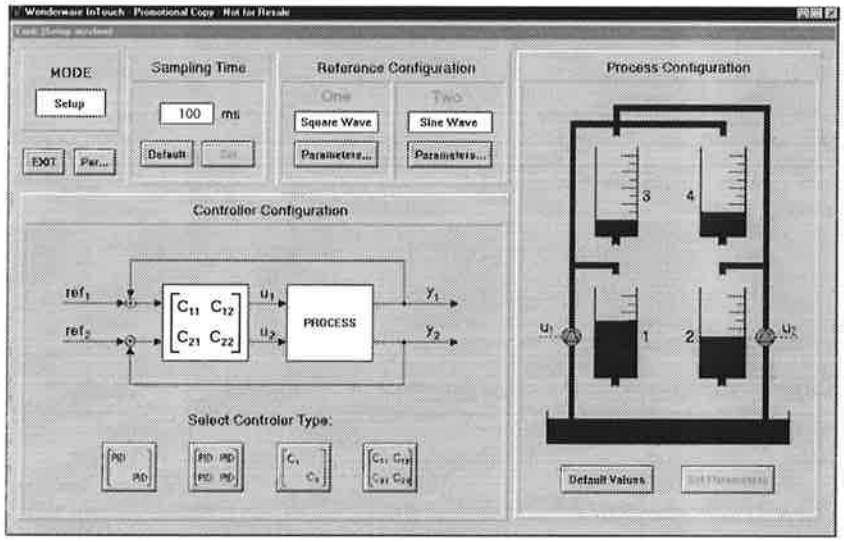

Figure 37 The Setup window is used to perform tasks that aren't done very often.

while the upper bound can be a few seconds. As it was mention, the value of this parameter doesn't affect the sampling time (refresh rate) used in the plots in the Main window.

Reference configuration The reference is specified by a wave type and its parameters, that include: the amplitude, period and mean. The wave type is selected using a pull down box, and the parameters are specified in a window presented when the user clicks in the Parameters button. The wave type can be set to one of the following values:

Square wave;

Sine wave; and

PRBS wave.

This is achieved by first clicking in the wave type box in order to view the pulldown menu, and then by selecting the appropriate wave from this menu. The parameters that specify the wave are set in a popup window that is presented when the user clicks in the Parameters button. This window is similar to the one used in the Main window for the same purpose, see Input Signals in the previous subsection. The meaning of the period parameter is different for a PRBS wave than it is for the Square and Sine waves. For the PRBS wave it represents the minimum number of samples between changes, while for the other wave types it represents the period of the wave as it is known, i.e. the time after which the signal will repeat itself. The user should have special attention to the value of this parameter because it should be at least twice as large as the sampling time, otherwise the wave generated will be a constant signal.

Controller configuration To select the controller type and set its parameters the procedure is identical to the one explained for doing it in the Main window, see Controllers in the previous subsection. The only difference is that, in Setup window it is also possible to set other parameters for the PID controllers. These parameters are: N (maximum derivative gain), Tr (integral reset time), a (setpoint weighting for the derivative part) and b (setpoint

weighting for the proportional part). The range for a and b parameters is between 0 and 1. The popup window for a PID controller is shown in Figure 38. This window is similar to the PID window shown in Figure 32 with the exceptions mentioned. With respect to the General controller window, it is equal to the one already shown in Figure 32.

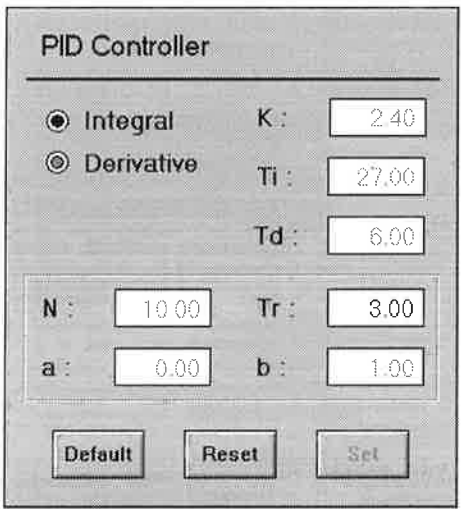

Figure 38 The PID Controller window (Setup window). The parameter values written in black are the default ones while the others, in gray, were modified.

Process configuration To set some parameters related with the simulation a representation of the process, identical graphically but not functionally to the one shown in the Main window, is also included in Setup window. As opposed to the image described in The Main Screen subsection, in this one the tubes represent only the water path, and the water in the tanks show the operating point specified by the user for that particular tank. Another difference comes from the actions that can be performed in the image, as clicking in a tank will open a popup window with additional information on that tank, but the same is not true for the pumps. The window shown, which can be seen in Figure 39, differ from the one shown in Figure 33 in the parameters presented and in that it's possible to set their values. The valve position parameter (γ_i) is only included in the lower tanks popup window, because we assume that the process simulation only has two valves. The values introduced in this field range between 0 and 1, and represent the amount of water that goes to the lower tank, while the amount of water going to the upper tank connected to the same pump is given by $1 - \gamma_i$. The meaning of the valve position parameter is the same we used in the beginning of this report. The other parameters included in this window are the same presented when performing the same action in the Main window process image, but now it's possible to change their values. There is only one thing left that should be mentioned, when the user change the value of one of these parameters some of the others will change automatically. This happens because of the relations between them, that were introduced in Section 2.

The buttons included in the bottom of the process image are used to reload the default values and to confirm the values just set. The actions performed by these buttons influence all the parameters that can be changed in the image.

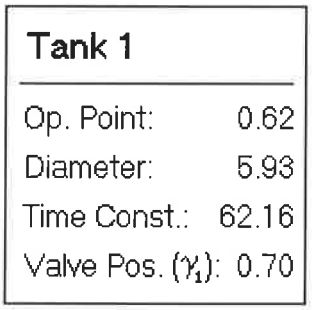

Figure 39 The Tank 1 information window (Setup window).

Parameter values The possibility of saving all parameters values for latter use or to be used by other programs like Matlab, is included in Setup window. The scheme used also allow the user to load parameter values previously stored in a file, and to reload the *default* or the *preset* parameter values. The meaning of these two types was introduced in The Environment in the previous subsection.

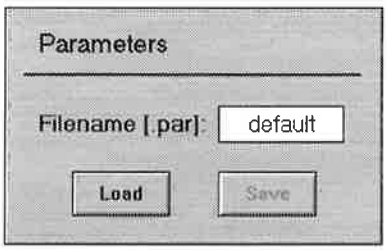

Figure 40 The Parameters window is used to load/save the interface parameters in a file.

To save or load all the interface parameters to or from a file, the user just as to open the popup window shown in Figure 40, by clicking in the Par button. In this window the user has to specify the name of the file, and then press the appropriate button. There are two names that cannot be used when saving the interface parameters to a file, which are the *default* and *preset* reserved names. These are used to reload the default and the preset parameter values. All the files are loaded from and written to the application directory in ASCII format. Some parts of a sample parameter file are shown below.

```
% TANK PAR-file
% Created on: Aug 05 14:01:47 1997

SAMPLING TIME
TSamp  100.0000

REFERENCE ONE
Mean  0.300000
Period  6.000000
Amplitude  0.150000

LOGGING
Points  120
LogFile data
```

```
SIMULATION
Diameter1  5.930000
OpPoint1  0.620000
Valve1  0.700000
Valve2  0.600000

CONTROLLER ONE
K  3.000000
Ti  30.00000
Td  0.000000
Tr  1.000000
r1  1.000000
s0  1.000000
```

The intention to not allow the user to override the file that contains the default parameter values using the method described, was to protect the common user from accidently override it. However, it is possible to change the interface default values by renaming any other parameter file as `default.par`. There was an idea on how to implement this function inside the interface by asking the user to login the interface, and having different privileges for distinct users.

5.5 Internal Implementations

During the interface development, decisions related with the implementation of some features were made. Because of that, an explanation on how some of them were implemented is imposed. We only mention the ones that the user should be aware of to understand what happens in certain situations. Among these we have the structure of the Modula2 and InTouch programs, the process simulation and the controllers.

Modula2 The program is structured in modules following some ideas introduced in the Real-Time Control Systems course [9]. It's divided in modules, some of them having an associated process, as it is the case of OpCom, RefGen and Regul, while other don't, like Simul and Control modules. The application is composed of four processes: OpCom, which handles the communication with the InTouch program and of course the interaction with the user; RefGen, that generates the reference signal waves; Regul, which communicates with the real process and uses other module functions to implement the features presented in the previous subsections; and Main, responsible for the launching and termination of all the other processes. In Figure 41, we can see the different modules in a graph that shows the dependency between them.

By using a structure like this it's possible to add new features to the interface, like other controller types, simple by adding modules similar to the Simul and Control.

Simulation The process simulation is done by means of the nonlinear equations that correspond to the process dynamics (physical model). The equations used (1) were discretized using the forward differences approximation with a sampling time defined by the user, that can vary between 50ms and a few seconds.

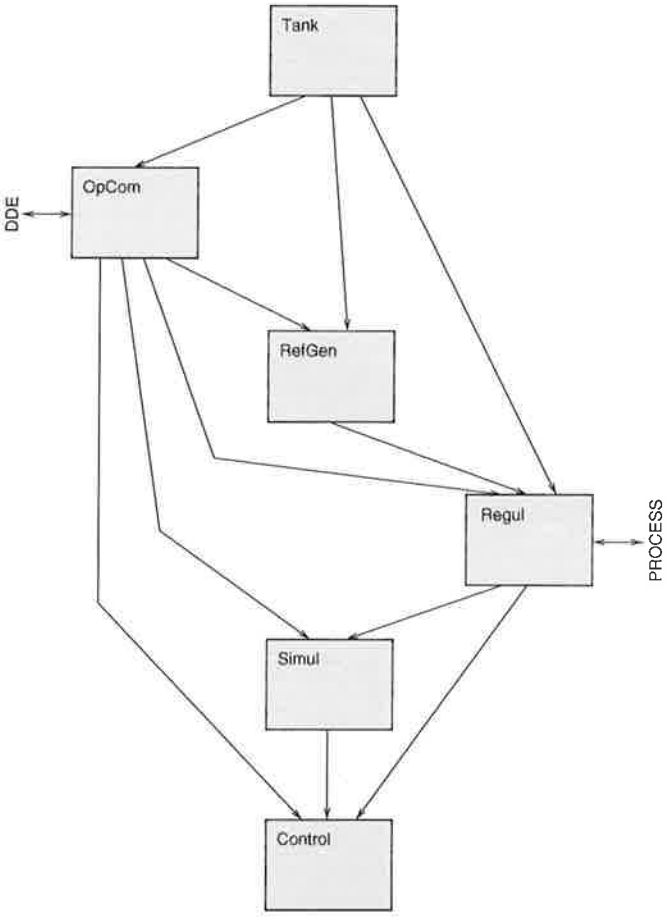

Figure 41 The modules dependency graph, in which the communication with InTouch and the real process is also shown.

Controllers The PID controllers implemented are described by the algorithm

$$U(s) = K_c\Big[bY_{sp}(s) - Y(s) \quad + \quad \frac{1}{sT_i}\big(Y_{sp}(s) - Y(s)\big) \\ + \quad \frac{sT_d}{1 + sT_d/N}\big(aY_{sp}(s) - Y(s)\big)\Big]$$

These controllers include setpoint weighting for both the proportional and the derivative parts, plus a limitation in the derivative gain. An anti-windup method to avoid integrator windup (reset windup) was implemented using the back-calculation, also called tracking method. We decide to implement this scheme only on the main diagonal PID controllers. Also implemented are methods for bumpless transfer between manual and automatic modes and bumpless parameter changes, in order to provide smooth transitions between manual and automatic modes and in cases where the controller parameters change. Concerning the discretization, the methods used were the forward and backward differences to approximate the derivative and the integral parts, respectively. The PID controller algorithm used and the way to implement the features mentioned, are described in [5].

The syntax of the general transfer-function controller implemented is given

below.

$$U(s) = \frac{s_0 s^4 + s_1 s^3 + s_2 s^2 + s_3 s + s_4}{s^4 + r_1 s^3 + r_2 s^2 + r_3 s + r_4} E(s)$$

These controllers have a maximum degree of 4 and the discretization method used was the Tustin approximation.

For both controller types, the sampling time used is always equal to the simulation sampling time, which has a minimum value of 50ms and a maximum of a few seconds.

InTouch Because the InTouch programs need much system resources when they use plots (real-time trends), it was necessary to made a compromise between faster plot refresh rates and adequate response times to user actions. For that, the plots sampling time was set to a constant value of 50ms. This value doesn't have any relation with the simulation/controller sampling time beyond the fact that it equals its lower bound.

6. Conclusions

In this project we studied a quadruple-tank laboratory process with an adjustable zero, built at the department by Rolf Braun. This study followed two parallel paths, one for the minimum phase case and the other for the non-minimum phase case. We did the physical modeling of the process, and the experimental identification using black box models for simulation and control purposes. This study was finished with the implementation and analysis of a control scheme where decentralized PID control was used and where the expected properties of the two system settings were evidenced.

As part of this project too, and as one of the main goals, we developed a graphical user interface for the real process which also includes a nonlinear simulation of the process. This interface was used in the study of the process and will be used, in the future, for further investigations on it. The interface, which from the begin was intended to have an open structure in order to be possible to, in an easy way, add new features, includes a considerable number of functionalities. As there were some other functionalities that we thought of but, because of the prototype characteristics of the process and time constrains we couldn't address, we finish leaving some guidelines for further developments of the user interface:

- implementation of other controller types and other anti-windup schemes;
- include the possibility to control the physical valves by software;
- development of other Matlab functions to treat the data created by the interface, as is the case of controllers data, so the user can take advantage of the advanced functions one can find in Matlab's environment;
- include the possibility of a user to login, in order to allow different privileges for distinct users (administrator and common user).

7. Bibliography

[1] Bradley R. Holt and Manfred Morari. *Design of Resilient Processing Plants-VI. The Effect of Right-Half-Plane Zeros on Dynamic Resilience*, Chemical Engineering Science, Vol 40, No. 1, pp. 59-74, 1985.

[2] Eric Astor. *Fran Pascal till Modula-2*, Studentlitteratur, 1986.

[3] L. Ljung *System Identification–Theory for the User*, Prentice Hall, Englewood Cliffs, NJ, 1987.

[4] Karl J. Åström and Bjorn Wittenmark. *Computer-Controlled Systems*, Prentice Hall, 1990.

[5] Karl Johan Åström and Tore Hägglund. *PID Control: Theory, Design, and Tuning*, Prentice Hall, 1990.

[6] L. Andersson and A. Blomdell. *A Real-Time Programming Environment and a Real-Time Kernel*, Technical Report No 30 1991-06-21, Dept. of Computer Systems, Uppsala University, Uppsala, Sweden, 1991.

[7] Rolf Johansson. *System Modeling & Identification*, Prentice Hall, 1993.

[8] P. Van Overschee and B. De Moor. *N4SID: Subspace Algorithms for the Identification of Combined Deterministic-Stochastic Systems*, Automatica, 30:1, pp. 75-93, 1994.

[9] Karl-Erik Arzen. *Real-Time Control Systems*, Department of Automatic Control, 1996.

[10] Youbin Peng, Damir Vrancic and Raymond Hanus. *A Review of Anti-Windup, Bumpless and Conditioned Tranfer*, 13th Triennial World Congress, San Francisco, USA, 1996.

[11] *Intouch Manuals*, Wonderware Corporation, 1996.

[12] José Luís Nunes. *Quadruple-Tank Process Identification*, System Identification Report, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1997.

[13] K. H. Johansson. *Relay Feedback and Multivariable Control*, PhD Thesis ISRN LUTFD2/TFRT–1048–SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1997.

[14] L. Ljung *System Identification Toolbox, Version 4.0.3*, The Mathworks, Inc, 1997.

[15] Karl Henrik Johansson. *Identification of a Double Fan and Plate Process Using Subspace Methods*, System Identification Report, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

[16] K. H. Johansson and J. L. R. Nunes. *A Multivariable Laboratory Process with an Adjustable Zero*, Submitted to 17th American Control Conference.

[17] L. Andersson, U. Jönsson, and K. H. Johansson. *A Manual For System Identification*, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

A. Operation Manual

A.1 Machine Specifications

Hardware The user interface was developed for PC machines running Windows NT 4.0 operating system. We could think that the hardware specifications for running Windows NT 4.0 should be enough, but this is not completely true, as we use another application to create man-machine interfaces – InTouch 6.0, that in certain situations needs much system resources. Because of that, we think that the best suited machine to run the interface has the following specifications:

- a Pentium processor;
- at least 16Mb of RAM; and
- an AD/DA converter.

The interface was developed using a PC with a Pentium processor running at 133Mhz, with 24Mb of RAM.

Software The only software package needed to run the interface is the InTouch 6.0 View program, for Windows NT 4.0. This program is a run-time viewer for man-machine interfaces created with InTouch 6.0, and is distributed with it. The user only has to be sure that it is installed in the machine, because the Tank program starts it automatically at the beginning.

A.2 Connecting The Cables

To communicate with the process the user interface uses an AD/DA converter, located in the back of the machine. The map between the inputs/outputs of the AD/DA converter and the process inputs/outputs is shown in Table 1. In Figure 42 a representation of the connections present in the back of the real process is shown. This figure is included because the names (numbers) given to the tanks and pumps in the real process differ from the ones we used in this report.

Converter	Process	
in 0	tank 1	(tank 1b)
in 1	tank 2	(tank 2b)
in 2	tank 3	(tank 1a)
in 3	tank 4	(tank 2a)
out 0	pump 1	(pump 2)
out 1	pump 2	(pump 1)
ground	ground	

Table 1 Map between AD/DA converter inputs/outputs and the process inputs/outputs. In the right column we have the names we gave to the tanks (left side), and the ones we find in the real process (right side).

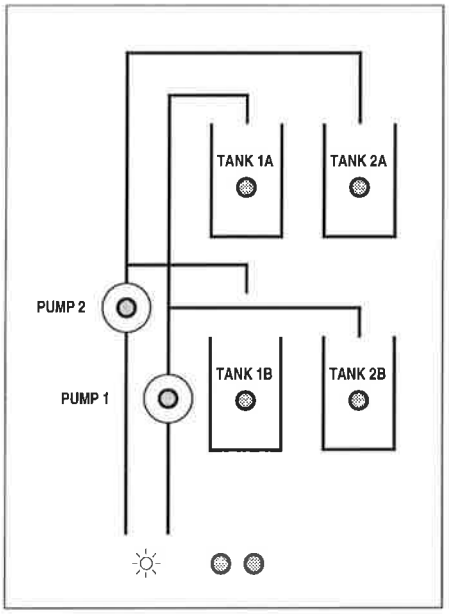

Figure 42 Image with the connections in the back panel of the quadruple-tank process.

A.3 Installing The Program

The procedure to install the interface application is very simple an it only requires that you have already installed InTouch 6.0 in your machine. The interface application is distributed in two 3.5" HD disks. To install it just run `setup.exe` and follow the instructions that are presented. During the installation procedure you will be asked to select among three different types of installations: Typical, Compact and Custom. The typical installation installs everything (application files and Matlab files), the compact installation only installs the application files, and the custom let you select the packages that you want to install. After the installation is finished, in the directory where you decided to install the application and in a subdirectory `intouch` you will find the executable files, the parameter files and the intouch files. Inside `matlab` subdirectory you will find some Matlab function to handle the data logged by the application.

Because the interface has to run the InTouch 6.0 View program, the directory were it is installed must be provided. It's assumed that InTouch is installed in `c:\programs\intouch`. If this is the case, the interface is properly installed and you can start it by selecting the *Tank* application from the Start button, or by executing the batch file `tank.bat`. If not, an error message is shown and you will have to edit this file using a text editor and change the last line so it matches the directory where you have installed InTouch. As an example, if you have InTouch installed in directory `d:\apps\intouch` then the last line of `tank.bat` should be changed to

```
start d:\apps\intouch\view intouch
```

To uninstall the application you have to go to the *Control Panel* and select icon *Add/Remove Programs*. This way all the components of the application will be deleted from the machine.

A.4 Sample Session

We'll go through a session using the interface, where the most commonly used features are introduced. Here we assume that the process setting is the minimum phase, and that we are acting on the real plant. All the parameters used are the same that were used to test the controller performance for the minimum phase setting, in Section 4. Thus this session is identical to the ones we performed in order to test the controllers presented in that section.

1. First of all we need to start the *Tank* application, after what we are dropped in the Main window. In here, select the ACTIVE mode, so the interface acts on the real plant rather than on the simulation, and switch to MANUAL control mode.

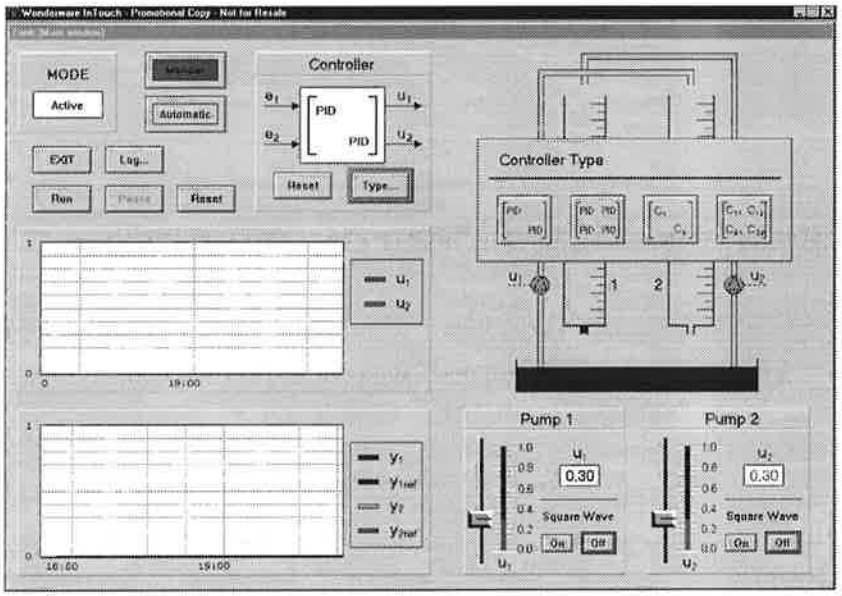

Figure 43 Interface image with the Controller Type window enabled. The controller type selected is decentralized PID, the leftmost.

2. The next step is to set the controller we want to test on the plant. We start by selecting decentralized PID control scheme in the Controller Type window, which is opened using the Type button (see Figure 43). As you can see, the controller matrix change so that it represents the selected controller type. Then we open the parameters window for each PID controller by clicking on it, see Figure 44. Choose a PI controllers by activating only the integral part, and set the parameters to: $K = 3.0$ and $T_i = 30$, for the upper-left controller; and $K = 2.7$ and $T_i = 40$, for the lower-right. Always press the Set button to confirm the values just entered.

3. After the controller parameters have been specified, set the control signal to a constant value of 0.3 (3 volts) for both process inputs and start running the interface against the real process using the Run button. When finally the process reaches the steady-state, after a few minutes, we decide to collect some data for latter analysis on controller performance. For that, we open the Logging window (see Figure 45) in order to set the parameters related with the logging. The amount of data that we are about to log is 10min (600 points with a sampling time of 1 second),

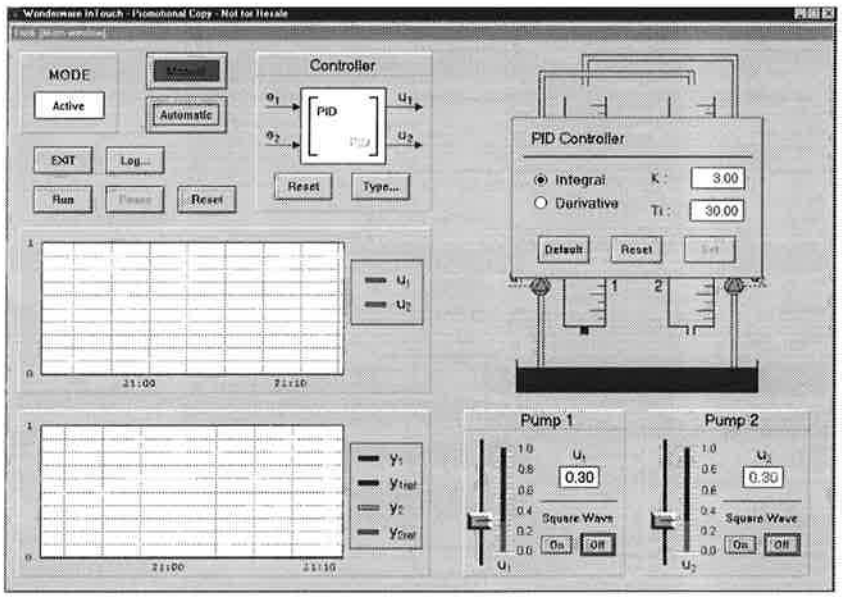

Figure 44 Interface image with the PID Controller window enabled. The controller specified is a PI.

and it will be saved in a file named `data.par`. The logging is started using the LogOn button, and then we close the Logging window.

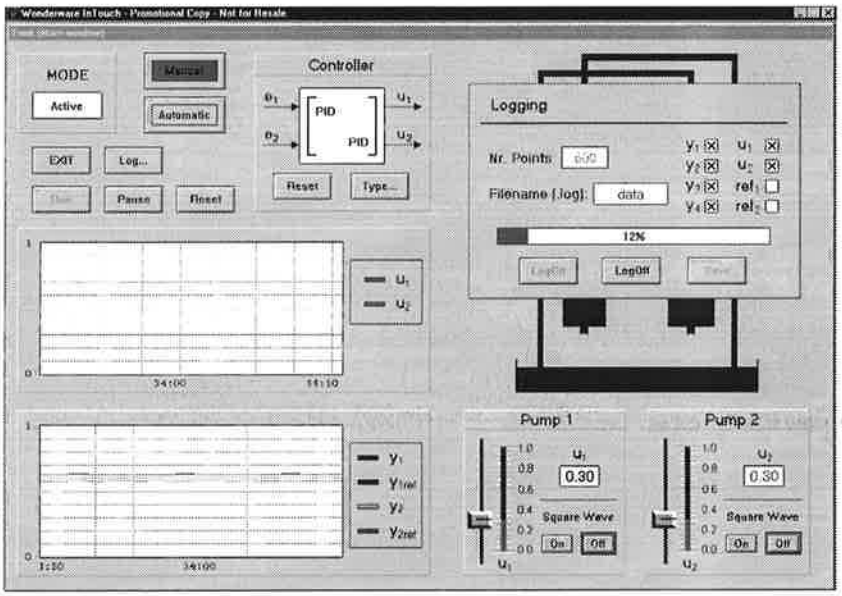

Figure 45 Interface image with the Logging window enabled.

4. Now that we are collecting data, we can test the controller we specified, by switching to AUTOMATIC control mode and adding a step change of amplitude 0.1 ($\approx$ 2 cm) in reference one. When the control mode is changed the controls used to set the control signals (u_1 and u_2) will change so the reference signals (ref_1 and ref_2) can now be set. To add a step in reference one we set its value to the actual value plus 0.1, using the text box for the effect. The process evolution can be followed in the plots and the process image (see Figure 46).

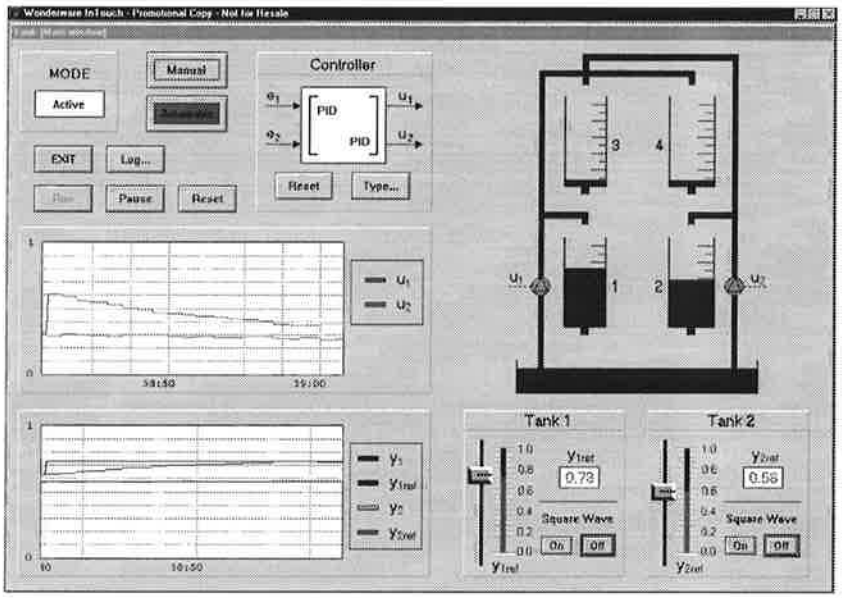

Figure 46 Interface image showing the process to a step change of amplitude 0.1 ($\approx$ 2 cm) in reference one.

5. We let the process run until all the data is collected. In order to check the logging state we reopen the Logging window and, after all the data had been logged, we save it to the specified file, using the Save button.

6. Now that we have ran the process in Manual mode, tested the controller (Automatic mode) and collected some data, we close the interface using the Exit button.

If you wish, you can start Matlab and take a look at the data we have just collected. To load it into Matlab's workspace use the Matlab function *load4t*. See Matlab's help on that function to know how to use it.