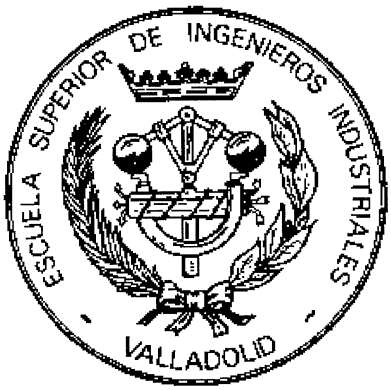
Identification and Robust Control of a MIMO Process

Ed Bras



Department of Automatic Control & System Engineering
University of Valladolid, Spain
Valladolid, November '97

Preface

In order to obtain an Electrical Engineering Degree at Eindhoven University of Technology. a final project was undertaken at the following University in Spain:
Escuela técnica Superior de Ingenieros Industriales. Universidad de Valladolid.

The difficult part with choosing a project is that there always has to be a compromise between the *number* of *subjects* to explore and the *zn-depth treatment* of these subjects
The choice I made. with the help of my supervisors of course. was an identification of a laboratory-located process and the development of several controllers The original choice has been changed during the progress of the project. keeping in my mind; my personal interests and the above compromise
Looking back. one might have doubt about the "in-depth" treatment
I definitely developed a good overview and understanding of some important trends in the identification and controller development This might make it easier to choose a possible next project that can be treated more in-depth
My personal interests are Identification and his link with (Robust) Control. while in my opinion both can't be seen as *separate*, as is often done in practice For example. why should we spend a lor of effort in identifying a very accurate model, while the controller that is to be applied is just a simple PID controller!

The project begun with intensive literature research. during which I was motivated to apply some of the new identification techniques, which are able to estimate model uncertainty regions
Unfortunately these techniques are still too "new" to put in a simple master thesis project
The result is that I used conventional identification techniques, which seemed to be difficult enough anyway, as a result of Murphy's law!

A thesis project can never be brought successful to an end by only one person. Therefore some acknowledgements are certainly necessary.
First of all I like to thank Professor P. v.d. Bosch for giving me the opportunity to fulfil my final project in a foreign country.
Before attending the University of Eindhoven I received my bachelor of science degree in physics at another school. Here: I also wanted to fulfil my final project in a foreign country, but it didn't seem to be possible.
So you can imagine that I was a bit surprised when I first walked into the office of Prof. v.d. Bosch and he asked me " Where do you want to go!".
My supervisor in Spain: Enrique Baeyeus Lazaro for helping me to overcome all the obstacles. and for being a source of inspiration.
The people from the department of Ingeniería de Sistemas y Automatics: Lorenzo, Felis. Eusebio, Gregorio and of course Alberto. They gave me the opportunity to do my work in a relaxed environment.
Last. but certainly not least, my Parents. Without their support and endless patience I would never have been able to experience such an excellent time in Spain. Of course I was not *always* working at my project !!

Valladolid. November '97
Ed Bras

Abstract

The project deals with the control of a laboratory-located MIMO process.
Two controllers are developed, namely a LQR controller (Linear Quadratic Regulator) and a Robust controller
To be able to develop a controller, a process model is estimated, which covers the main part of this essay

To perform any system identification and to implement any controller, a flexible-graphical software program has been written, that easily perfonns the above tests.

Physical laws are used to derive a inodel structure, to overcome the problem of choosing the correct model structure for inodel identification. This method is also known as *physical parameterized modelling*. The model structure contains known and unknown parts that are identified using a one-step-ahead prediction. A regularization technique is used, to restore the ill-conditioned Jacobian, due to the poor influence of the identification parameters on the model output.
An open-loop identification is performed of the final closed-loop system (including the controller).
An open-loop identification of the closed-loop has been chosen, to take a closer look at closed-loop identification and because the process is poorly damped.
A non-linear and a linear discrete process inodel are estimated. Because the non-linear inodel cannot be used to develop our controllers, we use it as a n-ay to test the derived inodel structure.
The non-linear inodel is obtained by performing a direct identification. That means that the process inputs are used as the identification inputs, instead of the reference inputs of the closed-loop system.
The identification parameters of the linear model are retrieved from the identified closed-loop model, by making use of the known controller inodel.
A Binary and Generalized binary noise signal is used as test-input signal. The a-priori information, needed to design these signals, is extracted from a calculated process model, were a good guess is made of the identification paraineters.
The estimated non-linear and linear model are validated by simulation; residual analysis and a scalar error measurement, all of them indicating a small model error.

The LQR controller. used to perform the "closed-loop" identification is based on the above calculated inodel
With the estimated process model, a (new) LQR controller and a Robust controller are developed. such that the closed-loop is asymptotically stable and that the performance requirements are met as good as possible
The performance requirements consist of a (m ell-known) zero tracking error. small overshoot and a "fast" step response with no actuator saturation
The measured process outputs correspond to the states of our process model. such that no observ er is used This results in a robust stability margin

The identification parameters have a physical significance, which can easily change through changing, for example a process valve The possible parameter uncertainties are well defined. such that a robust controller is developed, that can deal with these uncertainties in an explicit manner

The μ/H_∞ theory is used to develop our controller. such that we can guarantee pre-defined controller objectives. regardless of the parameter uncertainties.

The controller is calculated by assuming a full complex uncertainty matrix (unstructured uncertainty), such that "simple" algorithms can be used to calculate the controller. This leads to conservative results, as we are dealing with structured uncertainty, leading to a diagonal uncertainty matrix. The conservatism results are reduced by keeping the dimensions of stnrctured uncertainty matrix as small as "possible". This can be achieved by choosing only uncertain parameters (efficient parameters) that have a considerable influence on the output. The regularization technique: used during identifications, is used to select the efficient parameters.

An uncertainty state-spacc presentation is introduced. to formulate our robust controller problem in a H_∞ framework It handles uncertain parameters that appear in a state-spacc presentation in an easy way, and can easily be implemented in Matlab by using the μ-Analysis and Synthesis Toolbox

It is shown that the closed-loop performance deterioration of both controllers is small in case all uncertain parameters undergo a +20% parameter change. This is not that surprising, as during identification we already noticed that most identification parameters have only little influence on the process output, which was our motivation for using a regularization technique

Contents

1

Introduction

The project deals with the *developmenf* of *several controllers* for a laboratory MIMO process Before any controller can be developed, *identification* is performed to extract process information. which is needed for controller development

This chapter states some basic decisions and their motivation; that are necessary before we can start with our project. The ideas used are clear and simple, but not to say unimportant.
Identification consists of the making of several choices, like inodel structure and input signal. After these choices have been made, a model can "easily" be estimated by using some mathematical tools.
These choices need to be considered carefully, as the quality of the inodel depends highly on them. The choices are based on prior knowledge of the process. See the thesis of De Vries [Vri94b] for a good discussion on the use of prior knowledge.
The trend these days is to develop general flexible models where fewer prior knowledge is needed, because it can often be difficult to obtain.
With the above reasoning it can be quite easily understood, why Identification Engineers often speak about identification as being more an *art* than a *science*. This is exactly why this master thesis isn't filled with a lot of mathematical equations to make our decisions. Rather; we often use logical reasoning to come up with decisions whose accuracy, we can check later, using our estimation results.

1.1 Modelling and Controlling

To dev clop a controller. some knowledge of the process to be controlled is needed There are many ways to represent this knowledge The one most often used is the so called "mathematical representation"
In this thesis. the knowledge is stored in a discrete dynamical mathematical model. which is used to dev elop several controllers. which results will be compared

The quality of the model depends highly on the demands put on the controlled process. on the process characteristics and on the specific type of controller applied
The more accurate one wants to control the process, the more accurate one needs to store the process knowledge
Therefore it's important to specify the *identification procedure* from a controller point of view, by first specifying the controller demands. or. in other words:

Let the modelling effort reflects the intended use of the model

Our controller demands are quite "simple"
√ a zero steady state error of the output signals. with a step response as input signal.
√ a "small" settling time,
√ an overshoot of no more than about 5 %:
√ actuator saturation is allowed

All this asks for an "accurate" model

The "knowledge representation" needed for the controller depends on the type of controller
For example. with a robust controller some uncertainty description may have to be determined

There are two basic approaches to mathematical modelling white box modelling and black box modelling
White box models are solely constructed from prior knowledge and physical principles without any use of measurements from the system For obvious reasons this methods is also referred to as physical modelling
Black box models are designed entirely from data using no physical insight whatsoever The model structure is chosen from families that are known to be very flexible and successful in past applications This means that the parameters do not have a physical meaning; they are tuned just to fit observed data as well as possible

Black box identification is sometimes used as a synonym to system identification However. a much more convenient definition. and the one most commonly used, is that system identification is the theory of designing mathematical models of dynamical systems from observed data We can use white box modelling techniques and tune some of the physical parameters This method is also known as *physical parameterized modelling* and explained in-depth in the thesis of Lindskog [Lin96] Physical parameterized modelling also implies

that all identification methods can be categorised to be on a scale. ranging from pure black box to white box
The motivation of Lindskog for physical parameterized modelling is that *real engineering applications are never that black*". which means that ne are always able to collect some process information. which can be used in our identification procedure
In practice this process information is often discarded with the thought "'Re won't loose that much" By using general flexible structures it's hoped that such information is captured by tuning the parameters Howeyer the price paid for flexibility is usually that *many* parameters must be estimated. hereby violating another basic identification principle

Do not estimate what is already known !

The abo\e discussion on modelling is our main reason for using phjsical parameterized modelling Use is made of physical insight to come up with an appropriate inodel structure in state-space structure In this way a compact MIMO state space inodel is put forward, with only a fen parameters to be estimated Because we only need to estimate a fen parameters, these will show a smaller variance than when a general flexible black box structure is chosen

The process under stud: is non-linear Physical parameterized modelling is able to estimate non-linear models in an "easy" way The only problem we have is that inost of the control methods. like the one n e are going to use. are only able to cope with linear models

Tliere is chosen for an open-loop ideiltification of the closed-loop. to
- investigate "close-loop identification". as "most" industrial processes already use feedback.
- the process is poorly damped, as it has it's poles "close" to the imaginary axis. which makes it difficult to manage the process during open-loop identification (a practical issue)

It may sound somewhat confusing when we speak about an open loop-loop identification of the closed-loop. to estimate a process model While in fact. we are performing an closed-loop identification So we if we speak about the closed-loop identification. we mean that we are identifying a process model by performing an open-loop identification of the closed loop

After the identification is performed and the required models are estimated. two controllers will be deyeloped, namely a robust and a LQR controller
There is chosen for a robust controller because the position of the valves. as part of the process (fig2 *1*), can easily be changed and cause uncertainty in the parameters
The LQR controller shows some robustness properties (chapter 4). when no use is made of an observer to estimate the states For this reason a LQR controller is developed. because the process states form the measured process outputs

The software tool Matlab 4.2c, with the required toolboxes and Maple V 2.0a, will be used to perform our ideiltification and to develop the controllers.

1.2 Thesis Outline

In the next chapter we will start nith stating our objectives and a description of our process under study
In the chapters three and four a comprehensive treatment of some identification. and controller design subjects are outlined.
In chapter five and six. the outlined theory is used to perform our identification and develop several controllers

2

Objectives and Process Description

We will begin this chapter by formulating our original objectives.
Next, a description of the process will be given, and a white box model of the process will be derived. A non-linear and a linearized model structure will be derived and used during our identification procedure, which is described in the next chapter.

2.1 Problem Formulation

In the introduction we already pointed out our final objectives and some basic choices, concerning these objectives. However, in this section, we will formulate our original objectives

The process n e wish to control, is described in detail in the next section The process is approximately time-invariant and non-linear It has two control signals (the inputs) and three outputs, which arc the water levels. see *figure* 2 *1* in the next section.

The problem formulation is as follows:
Identify a process model with conventional techniques, like the ones described in [Lju87] and use this to design two controllers, namely a LQR and a Robust controller.
The controllers need to control the water levels in the outer columns.

2.2 Process Description

The laboratory-located process was developed by the German company Amira, who also has developed a software program, with Borland C++, to control the process
The acquisition card for collecting data, which is mounted in the computer, is also made by Amira The acquisition card, DAC6214 uses 12 bits for D/A and AID conversion Amira also offers an adapter card, which is an extension to the converter card, concerning interrupt generation for real-time operation However, this extension is not present in our configuration, which means that all real-time functions have to be implemented with software

Figure 2.1 shows a clear view of the inputs (u_1 and u_2) and outputs (h_1, h_2 and h_3) of the process
We will derive a non-linear and a linear model structure, which describe the input-output behaviour
In appendix A *1* a mathematical process model is derived In this section, only the final results are given, that is, a non-linear and a linearized model structure Also, an interpretation of the parameters can be found in the same appendix

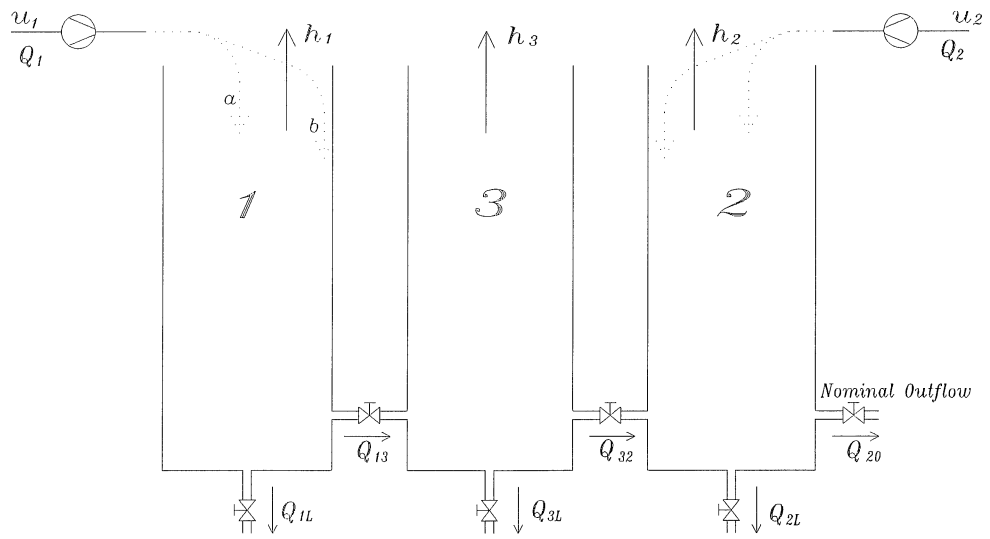


Figure 2.1: the laboratory-located process under study.

The *non-linear model* (appendix *A.1)*

$$h(k) = \Theta_N \cdot \Omega_N(k-1) \qquad (2.1)$$

$\Omega_N(k-1)$ contains the manipulated output and input samples up to time index $k-1$. the regression vector
The parameters in θ_L are easy to estimate and *keep* their physical meaning That is. usuallj we have to include a reparameterization, through which we cannot uniquely substitute back and determine the original parameters which contain the physical meaning See. for example [Lju87], example 5 1 on page 130

The *linear model* (appendix A.1):

$$h(k) = \Theta_L \cdot \Omega_L(k-1), \tag{2.2}$$

where:

$$\Omega_L(k-1) = \begin{bmatrix} h_1(k-1) \\ h_2(k-1) \\ h_3(k-1) \end{bmatrix} \tag{2.3}$$

Or in state space:

$$\begin{aligned} x_p(k+1) &= A_{p,d}x_p(k) + B_{p,d}u(k) \\ h(k) &= C_{p,d}x_p(k) \end{aligned}, \tag{2.4}$$

where:

$$A_{p,d} = \begin{bmatrix} \psi_{11} & 0 & W,? \\ 0 & \psi_{22} & , \\ \psi_{31} & w32 & w33 \end{bmatrix} = \begin{bmatrix} \psi_{14} & 0 \\ 0 & \psi_{25} \\ 0 & 0 \end{bmatrix}; C_{p,d} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.5}$$

The indexes p and d refer to the Discrete Process model.
The interpretation of the symbols can be found in appendix A.*1*

The values of the parameters in the matrices have to be estimated by performing an identification
The position of the zeros (in the matrix entries) and the parameters are easily verified by a closer look at the derived process model, using our knowledge of the process For example, if we look at the first element in the vector $x_p(k+1)$, which is equal to $h_1(k+1)$, we can see that this only depends on $h_1(k)$ (the integration). $h_3(k)$ (direct connected column) and $u_1(k)$ (the water input)

The non-linear model has to give us a better description of the real process The only problem is that we cannot use it for developing a controller with the techniques we are going to use But the non-linear model can help us to validate (give more confidence) our chosen model structure. Also can it be used in the final stage of controller design. to validate its performance

3

System Identification

This chapter gives a brief introduction to the field of parametric system identification. We start by stating the problem in section 3.1 and focus mainly on the choice of model structure in section 3.2. Section 3.3 and 3.4 address basic techniques for parameter estimation and calculation. Some general methods for model validation are presented in section 3.5.
A special section, section 3.6, is dedicated to the input design, because we can only see proper system dynamics if we apply a careful chosen input signal (you don't get what you don't ask).
We are not treating any "special" close-loop concepts, as we perform an open-loop identification of the closed-loop *system*, and extract a process model with knowledge of the controller.
The presentation is far from complete, but the purpose is merely to introduce concepts, ideas and algorithms, which are used in subsequent chapters.

More comprehensive treatments of system identification are given, for example in Soderstroin and Stoica [Söd89] and in Ljung [Lju87]. Furthermore one can find a detailed treatment of data pre-processing in the PhD thesis of Backx [Bac87].

3.1 Ingredients of System Identification

System identification is highly iterative in nature and is made up from three main ingredients: data, the model *structure* and the selection criterion, all of which include choices that are subject to personal judgements.

The data Z_N : To estimate models we first need to collect data. Let Z_N be a data column vector, containing all inputs and outputs in one row, used for identification:

$$Z_N = \left[z(1)\ z(2)\ \ldots\ z(N) \right]^T \in \Re^{N\times(p+m)}, \qquad (3.1)$$

where T indicates tlie transpose operation.
For a system with p outputs and m inputs:

$$z(k) = \left[y_1(k) \ldots\ y_p(\mathrm{k})\ u_1(\mathrm{k}) \ldots\ u_m(\mathrm{k}) \right] \in \Re^{1\times(p+m)}, \qquad (3.2)$$

where time index k is equal to iT with $\mathrm{I} = 1..\infty$.

It is of course crucial that the data reflects all important features of the underlying system. The excitation signals u(t) must be carefully chosen so that the system dynamics clearly show up in the outputs y(t).

The model structure $\mathcal{M}$ **:** One of the most difficult decisions in identification, if not the most difficult, is the model structure selection. Roughly speaking. the problem can be divided into three subproblems:

- the first one is to specify the *type* of model to use. This involves the selection betmeen linear and non-linear structures. between black box, grey box and physically parameterized approaches, and so forth;
- tlie next issue is to decide the *size* of the model set. This includes the choice of possible variables and combination of variables to use in the model. It also involves fixing orders and degrees of the chosen model types, usually to some interval. Once these two issues are settled one has in principle determined a model set * over which the search for a model can be carried out. However, one problem is that $\mathcal{M}^*$ can be much too large, although by using prior structural information, it can often be reduced significantly;
- tlie last item to consider is how to *parameterize* tlie model set. so that the estimation algorithms have a good chance of finding reasonable parameter values.

Assuming that the members of * can be parameterized by a finite-dimensional parameter vector $\theta \in D \subset \Re^{d\times 1}$, one particular model corresponding to θ is denoted (θ). The *model structure* to which such a model belongs is defined by tlie mapping

$$\mathcal{M}: D_{\mathcal{M}} \ni \theta \rightarrow \mathcal{M}(\theta) \in \mathcal{M}^* . \qquad (3.3)$$

We will usually not use this theoretic system notation, but instead denote the family of candidate structures by

$$\hat{y}(k,\theta) = g(k,\theta,\varphi(k,\theta)), \qquad (3.4)$$

where $\hat{y}(\cdot)$ accentuates that the function $g(\cdot)$ is a predictor, that is, it is based on signals that are known at time k. The predictor structure is ensured by the regressor $\varphi(k,\theta)$, which maps output signals up to the index k-I ($y(k-1)$) and input signals up to index k (u(k)) to an r-dimensional regression vector:

$$\varphi(k,\theta) = \varphi(y(k-1), u(k), \theta) \in \Re^{r\times 1}. \qquad (3.5)$$

The selection criterion $V_N(\theta, Z_N)$**:** measured and model outputs never match perfectly in practice, but differ as:

$$\varepsilon(k,\theta) = y(k) - \hat{y}(k,\theta), \qquad (3.6)$$

where $\varepsilon(k,\theta)$ is an error term reflecting unmodeled dynamics on one hand and noise on the other hand. An obvious modelling goal must be that this discrepancy is "small" in some sense. The purpose of the selection criterion is precisely to give "small" a meaning by ranking different models according to some pre-determined cost function (hence each model is assigned a quality mark). The selection criterion can come in several forms, although me shall adopt here a scalar loss function:

$$V_N(\theta, Z_N) = \frac{1}{N}\sum_{t=1}^{N} l(\varepsilon(k,\theta)), \qquad (3.7)$$

where $l(\cdot)$ is a positive scalar-valued function, typically chosen to be quadratic. In the case of SISO systems this becomes $0.5\varepsilon^2(k,\theta)$ and in the case of MIMO systems: $0.5\varepsilon^T\Lambda\varepsilon$.
With the matrix A one can weight the relative importance of the components in $\varepsilon(\cdot)$.

Once these three items are settled, we have implicitly defined the model search. It then "only" remains to estimate the parameters θ and to decide upon whether the model is good enough or not. If the model cannot be accepted, some, or even all, of the entities above have to be reconsidered; in the worst case one must start from the very beginning and collect new data. Thus system identification is iterative and the (model acceptance) criterion shows *personal* taste characteristics.

3.2 Model Structures

This section gives an overview of some common model structures.
The section only contains a brief overview, while a lot of the theory is very well known and can be found in the books pointed out at the beginning of this chapter.

Linear Black Box Structures: linear black box modelling is based on the assumption that the data originates from a system:

$$y(k) = G^0(q,\theta^0)u(k) + H^0(q,\theta^0)e^0(k), \qquad (3.8)$$

where q is the shift operator, and $G^0(q,\theta^0)$ and $H^0(q,\theta^0)$ are rational transfer functions, both assumed to be proper and $H^0(q,\theta^0)$ additionally assumed to be monic and inversely stable. Furthermore, the sequence $\{e^0(t)\}$, $t=1,2,\ldots,N$, denotes one particular realisation, which for

analysis purposes often is assumed to be white noise. In practice $e^0(t)$ is often taken as Gaussian white noise.
By simply replacing $e^0(t)$ by *(3.6)* we get the one-step-ahead predictor (the predictor structure follows as $H^0(\cdot)$ is monic):

$$\hat{y}(k,\theta^0) = (1 - H^0(q,\theta^0)^{-1})y(k) + H^0(q,\theta^0)^{-1}G^0(q,\theta^0)u(k)\,. \qquad (3.9)$$

Because neither the parameters θ^0 nor the structures $G^0(\cdot)$ and $H^0(\cdot)$ are known, these must be searched for in the model structure:

$$g(k,\theta,\varphi(k,\theta)) = (1 - H(q,\theta)^{-1})y(k) + H(q,\theta)^{-1}G(q,\theta)u(k)\,. \qquad (3.10)$$

What this leaves us with is the parameterization of $G(\cdot)$ and $H(\cdot)$, which is usually done by using "some" process information. Several special cases of the structure of *(3.10)* (depending on the choice of $G(\cdot)$ and $H(\cdot)$) have been so successful in past applications that they have been given their own names. The most common are: FIR and ARX. which form a linear regression $\theta^T\varphi(t)$, and OE, ARMAX and BJ which form a pseudo-linear regression $\theta^T\varphi(t,\theta)$.

The general MIMO case, with m inputs and p outputs, can be covered by working with $p\times$ m pseudo-linear parallel regressions. However. for such multi-variable systems, it is often more convenient to work with model structures with state-space innovation-form:

$$\begin{aligned} x(k+1) &= A(\theta)x(k) + B(\theta)u(k) + K(\theta)e(k) \\ y(k) &= C(\theta)x(k) + D(\theta)u(k) + e(k) \end{aligned}\,, \qquad (3.11)$$

with $e(k)$ discrete white noise:
The corresponding one step predictor is given by (replacing $e(k)$ by *(3.6)*):

$$\begin{aligned} \hat{x}(k+1) &= (A(\theta) - K(\theta))C(\theta)\hat{x}(k) + (B(\theta) - K(\theta)D(\theta))u(k) + K(\theta)y(k) \\ \hat{y}(k) &= C(\theta)\hat{x}(k) + D(\theta)u(k) \end{aligned} \qquad (3.12)$$

Notice that *(3.11)* is a more restricted structure than the one in [Lju87] (page 86), for where the Kalman applies, but it is the same structure that Ljung uses in the identification Toolbox of Matlab.
If all these matrices are filled with parameters it is rather easy to show that the above state space structure is over-parametrized. That is, it involves more parameters than necessary to describe the input-output behaviour of the true system, leading to more than one global minimum. This has motivated the development of special *structures* (for example; controller and observer canonical forms), having fewer parameters.
McKelvey has shown that there are black box situations where a fully parametrized state space structure is to be preferred to these tailed structured, mainly because of numerical problems. See for example [Kel93].

Non-linear Black Box Structures: a lot of things become more complicated when me turn to non-linear black box modelling.
This area isn't of direct importance to our work, but it cannot be left out of our small overview of model structures.
The possibilities are enormous in this area and we restrict our overview to just the basic structure, from which many black box models follow. See [Lin96] and references therein for a more complete overview.

Suppose that we have a non-linear MIMO system like *(3.8):*

$$y(k) = f^0(t,\theta^0, y(k-1), u(k)) + v(k), \qquad (3.13)$$

where $\mathbf{f}^0(\cdot)$ is some unknown nail-linear function and $v(t)$ an additive disturbance term.
Suppose further that the regression vector $\varphi(t)$ has a predetermined number of terms. A "natural" identification approach then is to try predictor structures of the following form:

$$g(k,\theta,\varphi(k)) = \sum_{j=1}^{n} \alpha_j g_j(\varphi(t), \beta_j, \gamma_j), \qquad (3.14)$$

where, sometimes with tlie abuse of notation, we call $g_j(\cdot)$ a basis function. These are usually rather simple and they all are typically of one single type. Furthermore, the basis functions mean that each $g_j(\cdot)$ essentially covers a certain part of the total regression space. This part is specified by the parameters β_j and γ_j, where β_j is related to the scale or direction of the basis function and γ_j specify the position or translation of it. The remaining α_j parameter is a "coordinate" parameter (weight), giving the basis function its final amplitude shape.
Depending on the basis function there are several classifications that can be made.
As an example. we would like to point out a well known non-linear black box model, the so called *neural network* (*fig 3.1)*, which have a basis function of the so called *ridge* type:

$$g_j(\varphi(k), \beta_j, \gamma_j) = k(\beta_j^T \varphi(k) - \gamma_j), \qquad (3.15)$$

where $k(\cdot)$ is a function of one variable. Well known is tlie *threshold* function $k(x)=max(0,sign(x))$.

Physically Parameterized Structures: physically paraineterized model structures are solely designed from physical insight and are customized for one specific application or for some classes of applications.
The model description is fixed. often through quite a laborious procedure, it contains known constants as well as unknown parameters. This also means that all introduced variables $x(k)$ have physical significance.
Since most laws of physical are time-continuous it is natural here to adopt the framework of differential equations.

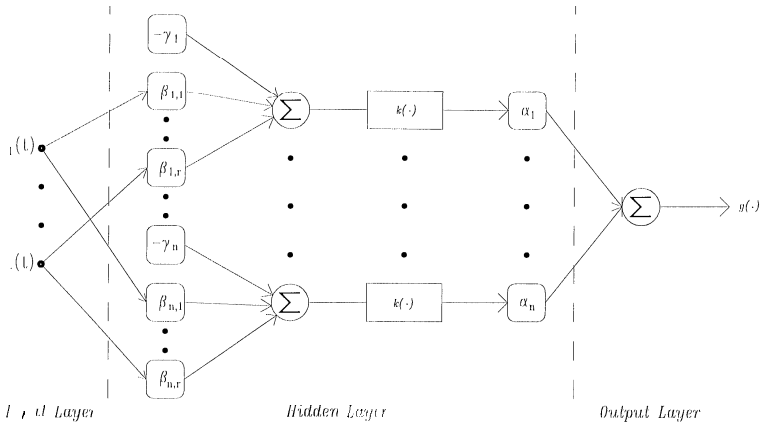


***Figure** 3.1: neural network*

Although we often start our work in this domain, the resulting model is sooner or later converted into the time-discrete world. Before this, the gained relationships are often summarized in a time-continuous state space form like:

$$\begin{aligned} x(t) &= f(t, x(t), \theta, u(t), v(t)) \\ y(t) &= h(t, x(t), \theta, u(t), e(t)), \end{aligned} \qquad (3.16)$$

where $f(\cdot)$ and $g(\cdot)$ are non-linear functions that, except for the parameter θ, are known.

3.3 Parameter Estimation

Having determined what kind of model structure will be used, the next step is to use input-output data to estimate the unknown parameters. These should be adjusted so that the performance of the accepted model is optimal in some sense. To be able to make this decision we must define a performance criterion. As a criterion function we use *(3.7)* (scalar loss function) and minimize this function. which ***indirect,*** results in a small *average model error* :

$$\hat{\theta}_N = \arg \min_{\theta \in D_M} V_N(\theta, Z_N), \qquad (3.17)$$

where "*arg min*" is the operator that returns the argument which minimises the loss function. Equation *(3.17)* is a very important and well-known problem formulation leading to *prediction error minimization (PEM)* methods.
We can consider two type of PEM algorithms. The first one deals with model structures nere the parameters enter the prediction model linearly and where *(3.17)* becomes a *linear least-squares problem.* This problem can be solved analytically.

The second type deals with model structures where the paraineters enter the prediction structure non-linearly, were *(3.17)* can be solved by a *non-linear* least- square^ *technique.*

Non-linear Constrained Least-Squares: for many grey box problems it is known *a priori* from the application which set of paraineters values are *feusible.* Hence it is most natural to use this insight to provide reasonable initial parameter values for the algorithms above. However. nothing hampers this knowledge from being violated once the search is started. To ensure an estimate with some physical meaning, we can include a parameter constraint.
The idea is as follows. The loss function *(3.7)* is expanded with an extra term, which contains the parameter constraints:

$$V_N(\theta, Z_N) = \frac{1}{N}\sum_{t=1}^{N} l(\varepsilon(k,\theta)) + \rho\sum_{k=1}^{l}\sigma(c_k(\theta)),, \qquad (3.18)$$

where p ^{l} is a positive scalar and $\sigma(\cdot)$ is a *barrier function.* The barrier function is chosen so that an increasingly larger value is added to the objective function as the boundary of the feasibility function is approached from the interior; at the boundary itself this quantity is infinity. $c_k(\cdot)$ contains the constraints of all the l parameters.

Spurious and efficient parameters: ill-conditioning of the Jacobian, which is propagated to the Hessian approximation, is the main numerical difficulty in Newton type of search algorithms. In identification this problem arises, for example, when the data Z_N is not sufficiently informative or when the applied model structure $g(\cdot)$ is "too" flexible. i.e., it is over-parainetrized.
A rank deficient Jacobian matrix is obtained if one or more of the columns of the J_N contain zero or almost zero entries only. It also occurs when some of the columns are linearly or near to linearly dependent.
Zero columns occurs when the criterion function may exhibit flat valleys, which means that these parameters hardly influence the model output in this search area. Linearly or near to linearly columns of the Jacobian usually means that several parameters try to reflect similar system properties.
Both cases mean that we are dealing with parameters that don't influence the criterion fit that much. This observation suggests that the parameters should be divided into two sets: the set of *spurious* and the set of *efficient* paraineters. Since the spurious paraineters do not improve the fit considerably it is intuitively reasonable to treat those as constant that are not to be estimated. Notice that this paraineter decomposition often can be made quite arbitrarily. For example, if the model structure is $g(\cdot)=(\theta_1+\theta_2)u(k-1)$, then either θ_1 or θ_2 can be regarded as efficient.
The above situation can easily occur (less obvious) when using high flexible models. Such excessive flexibility is found in most black box approaches, e.g., in structure *(3.11)*, in neural networks, in radial basis function networks, etc., while the problem is rarely encountered when using physically parameterized structures. In general, physical insight can be used to reduce the number of spurious paraineters. For example, parameters in "flat.' regions can be replaced by constants.
The way to overcome the ill-conditioning problem and automatically unveil an efficient parameterization are known as *regularization* techniques.
A regularizing effect can be imposed by adding a penalty term to the criterion *(3.7)*:

$$V_N(\theta, Z_N) = \frac{1}{N}\sum_{t=1}^{N} l(\varepsilon(k,\theta)) + \mu\theta^2, \qquad (3.19)$$

where $\mu > 0$ is a small user-tuneable row vector. ensuring a positive definite Hessian.
The regularizing effect is that a parainеter, not affecting the first term that much will be kept close to zero by the second term. This means that μ can be seen as a threshold that labels the parameters to be either efficient or spurious. A large μ simply means that the number of efficient parameters d becomes small.
Altering the criterion to be:

$$V_N(\theta, Z_N) = \frac{1}{N}\sum_{t=1}^{N} l(\varepsilon(k,\theta)) + \mu(\theta - \theta')^2, \qquad (3.20)$$

yields regularization towards Q'. This is interesting, as Q' can represent prior parameter knowledge.
There can be shown [Lju87] that, under mild assumption, the asymptotic misfit essentially depends on tno factors, that can be affected by the choice of the model structure. First we have a *bias* error, which reflects the misfit between the true system and the best possible approximation of it. Typically, this error decreases when the number of parameters d increases. The other term is the parainеter *variance* error, which usually grows with d but decreases **m** ith N.
Thus there is a clear trade-off between the bias and the variance contributions. Suppose that a flexible enough model structure has been decided upon. Decreasing the number of parameters that are actually updated, by increasing μ is beneficial for the total misfit as long as the decrease in variance error is larger than the increase in bias error. In other mords. the purpose of regularization is to decrease the variance error contribution to a level where it balances the bias misfit.
A statistical analysis, like an asyinptotic expression for the covariance matrix of the parameters, can be found in [Lju92] and [Kel93].

3.4 Parameter Calculation

This section describes how the linear least-square and non-linear least-square problem can be solved, using fixed model structures.
A more in-depth treatment of the described methods, can be find in [Lin96], [Lju87], [Gra92] and references therein.

Linear Least-Squares: when all parameters occur linearly one usually talks about a linear *least-squares* problem. This is the case for the structures like FIR. ARX. In fact all structures of the form $\theta^T\varphi(t)$, allowing linear as well as non-linear regression vector elements, belong to this category.

The optimal solution of *(3.17)* is found, when the gradient of *(3.17)*, with respect to θ, equals zero, that is:

$$\frac{\partial}{\partial \theta} V_N(\theta) = 0 .$$

In case of a MIMO system with a quadratic criterion, *(3.21)* becomes (Lju87 page 179):

$$\left[\sum_{k=1}^{N} \varphi(k)\Lambda^{-1}\varphi^T(k)\right]\hat{\theta}_N = \sum_{k=1}^{N} \varphi(k)\Lambda^{-1}y(k) \qquad (3.22)$$

For numerical reasons, the inverse of $[\cdot]$ is rarely formed, but instead the estimate is computed via QR-factorization or singular value decomposition (SVD), which both are able to handle rank deficient regression matrices.

Non-Linear Least-Squares: when the parameters to be estimated appear in a non-linear fashion, in the function to minimize. Typically all structures of the form $0^T\varphi(t,\theta)$ belong to this category.
Methods for numerical minimization of a function $V(\theta)$, update the estimate of the minimising point iteratively. This is usually done according to:

$$\hat{\theta}^{(i+1)} = \hat{\theta}^{(i)} + \alpha f^{(i)}, \qquad (3.23)$$

where $\mathbf{f}^{(i)}$ is a search direction based on information about $V(\theta)$ acquired at previous iteration, and α is a positive constant determined so that an appropriate decrease of $V(\theta)$ is obtained. Let $0, \in \Re^{d\times 1}$.
In this section we will pay especial attention to the determination off $^{(i)}$. Depending on the information supplied to determine $\mathbf{f}^{(i)}$, numerical minimization methods can be divided into three groups:

1 methods using function values only;
2. methods using values of the function V as well as its gradient;
3. methods using values of the function, of its gradient, and of its Hessian (the second derivative matrix).

The typical member of group 3 corresponds to *Newton algorithms*, where the correction in *(3.23)* is chosen in tlie "Newton" direction:

$$f^{(i)} = -\left[V_N''(\hat{\theta}_N^{(i)})\right]^{-1} V'(\hat{\theta}_N^{(i)}), \qquad (3.24)$$

with the gradient as:

$$V_N'(\hat{\theta}_N^{(i)}) = -\frac{1}{N}\sum_{k=1}^{N} J(k,\hat{\theta}_N^{(i)})\varepsilon(k,\hat{\theta}_N^{(i)}) \tag{3.25}$$

With $J(\cdot) \in \Re^{d\times m}$ being the Jacobian vector:

$$J(k,\hat{\theta}_N^{(i)}) = \left[\frac{\partial \hat{y}(k,\hat{\theta}_N^{(i)})}{\partial \theta_1^{(i)}} \quad \cdots \quad \frac{\partial \hat{y}(k,\hat{\theta}_N^{(i)})}{\partial \theta_d}\right]^T, \tag{3.26}$$

where m is the number of outputs.
Differentiating the gradient with respect to the parameters yields the Hessian:

$$V_N''(\hat{\theta}_N^{(i)}) = \frac{1}{N}\sum_{k=1}^{N} J(k,\hat{\theta}_N^{(i)})J(k,\hat{\theta}_N^{(i)})^T - \frac{\partial J(k,\hat{\theta}_N^{(i)})}{\partial \hat{\theta}_N^{(i)}}\varepsilon(k,\hat{\theta}_N^{(i)}). \tag{3.27}$$

The major computational burden lies often in the calculation of the Jacobian $J(\cdot)$. One way around this difficulty is to numerically approximate the derivatives by finite differences. For example by a forward difference approximation with a perturbation vector $h_k e_k$:

$$J_k(k,\hat{q}_N^{(i)}) = \frac{\partial \hat{y}(k,\hat{\theta}_N^{(i)})}{\partial \hat{\theta}_k^{(i)}} \approx \frac{\hat{y}(k,\hat{\theta}_N^{(i)} + h_k e_k) - \hat{y}(\hat{\theta}_N^{(i)})}{h_k}, \tag{3.28}$$

with e_k being a column vector with a one in the kth position and zeros elsewhere and with h_k being a small positive scalar perturbation.

As the first part of the Hessian can be formed directly from the Jacobian, the part involving the derivative of $J(\cdot)$ is generally rather expensive to compute. Therefore two classes of methods can be recognised: those that try to approximate the second derivative and those ignoring it. Here we focus on two methods, which fit in the latter category and hence second derivative approximations will not be needed.
First we will point out the most simplest way of approximating the Hessian, used by the so called gradient method.

Gradient method (steepest -descent): since the gradient itself points in a descent direction a simple and robust idea is to replace the Hessian by an identity matrix of appropriate dimension, thus avoiding the inverse completely. Hence $f^{(i)}$ will become:

$$f^{(i)} = -I \cdot V_N'(\hat{\theta}_N^{(i)}) \tag{3.29}$$

Although simple, one major drawback with the gradient method is that the convergence rate is rather slow, close to the minimum when the function to be minimized has long narrow valleys. A. Grace shows this in [Gra92], through an example.

Gauss-Newton method: by neglecting the second derivative term of the Hessian, we get:

$$f^{(i)} = -\left[\sum_{k=1}^{N} J(\cdot)J(\cdot)^T\right]^{-1} V_N'(\hat{\theta}_N^{(i)}) . \qquad (3.30)$$

Note that we left out the constant N in *(3.30)*, which appears in *(3.27)*, as this is just a constant and can easily be included in α in *(3.23)*
Now suppose there exists a global minimum, such that $\varepsilon(k,\theta)=e_0(k)$, such that $E\{\varepsilon(k,\theta)\}=0$. Then this value yields to a global minimum of $E\{V_N(\theta)\}$. Close to the global minimum the second term of Hessian will then be close to zero, since:

$$E\frac{\partial J(\cdot)}{\partial \theta_0} e_0(k) = 0 \quad \forall k , \qquad (3.31)$$

where θ_0 indicates the parameters in the global minimum. As a consequence: leaving out the second term, will not take away too much of our 'good' convergence *near* the minimum, like the gradient method does.
Moreover. by omitting the second term, the approximated Hessian is always assured to be positive semidefinite. This makes the numerical procedure a descent algorithm and guarantees convergence to a stationary point.
Even though the expression $[\cdot]$ in *(3.30)* is assured to be positive semidefinite, it may be singular or close to singular. This is the case when the model is over-parametsized or the data not informative enough. Then some numerical problems arise when calculating the inverse in *(3.30)*. Various ways to overcome this problem exists and are known as *regularization* techniques. A well known technique is the *Levenberg-Marquardt* procedure. Other techniques are the singular value decomposition (SVD) and QR-factorization, which were already mentioned as a tool for solving the Linear Least-Squares.

Levenberg-Marquardt method: is able to deal with the singularity problems through:

$$f^{(i)} = -\left[\left(\sum_{k=1}^{N} J(\cdot)J(\cdot)^T\right) + \delta \cdot I\right]^{-1} V_N'(\hat{\theta}_N^{(i)}) , \qquad (3.32)$$

where the approximated Hessian is guaranteed to be positive definite since $\delta > 0$. Just like in the case of the Gauss-Newton algorithm, the update is in a descent direction.

Notice that:
- the Gauss-Newton method is also know as *modified Newton-Raphson* and *Quasi-Linearization*;

often the term *damped* is used, to indicate that a adjusted step size is used ($\alpha^{(i)}$ in *(3.23)*). For example the *damped Gauss-Newton* method. Adjusting the step size is done by a so called line-search method. which we did not point out in this section.

3.5 Model Validation

After estimation, the obvious question is whether the derived model is adequate for its intended use or not. This is a *subjective* and overall hard problem of *model validation*.
To gain confidence in a model. the general advice is to employ as many validation tools as possible. by using *all* available process information.
So it might be better to speak about model invalidation, as we are doing our best, searching for some test. which our model will not pass successfully. If it passes the test. me will obtain more confidence in our estimated model.
For example if the presented parameter represents the length of a rod it must at least be positive. Test of this kind belong to the prior knowledge category, and are especially important when the parameters have a physical meaning.
The overwhelming majority of methods, are based on experimental data. A basic test is to investigate the variance of the estimated parameters. A high variance compared to the parameters value indicates that something is wrong. Another useful method is to estimate several models simultaneously parallel. The frequency response of a parametric model can for example be compared to a non-parametric spectral estimate. This non-parametric model is not very useful for developing a controller, but for validating in the frequency domain it serves as a nice tool. Especially while it has a small bias and hardly depends on any prior knowledge. which gives a high confidence level.
The most versatile validation tool for all categories is *simulation*. The true system and the derived model are then fed with the same input signals. whereupon the measured outputs are compared to the ones computed from the model. For a *fair* comparison it is desirable here that the experiments are based on fresh data, that is. data not used for estimation. This is known as *cross validation*.
It is also often worthwhile to investigate the residual sequence $\varepsilon(\cdot)$ *(3.6)*, especially on new data.
The often assumed "whiteness" (all process information is captured by the chosen model structure) of the residual sequence can be tested through the sample covariance

$$R_N^{\varepsilon}(\tau) = \frac{1}{N}\sum_{k=max(1,1+\tau)}^{N+\tau}\varepsilon(k,\hat{\theta}_N)\varepsilon(k-\tau,\hat{\theta}_N), \qquad (3.33)$$

which for $\tau > 0$ should be "small". at least if N is large. Furthermore, the assumed independence between the input and residual signals can be inspected by plotting

$$R_N^{\varepsilon,u}(\tau) = \frac{1}{N}\sum_{k=max(1,1+\tau)}^{N+\tau}\varepsilon(k,\hat{\theta}_N)u(k-\tau), \qquad (3.34)$$

for various time lags τ. These sliould also be "small", since otherwise there is more information in tlie output originating from the input, which means that there is still unmodeled dynamics present.

The significance of these tests can be seen by substituting *(3.8)* and *(3.10)* in *(3.6)*, so that we get:

$$\varepsilon(k,\hat{\theta}) = H^{-1}(q,\hat{\theta})\left\{(G(q,\hat{\theta}) - G^0(q,\theta^0))u(k) - H^0(q,\theta^0)e(k)\right\} \qquad (3.35)$$

It can easily be seen that *both* $G^0(q,\theta^0)$ and $H^0(q,\theta^0)$ are estimated correctly, if *(3.33)* is small. Whereas *(3.34)*, will only tell us if $G^0(q,\theta^0)$ is estimated correctly, that is, *(3.34)* will be small, when the *bias* term $G(\cdot) - G^0(\cdot)$ is small.

3.6 Input Design

Because we haven't any restrictions on our input design, as with an industrial process, we are left with a lot of freedom to carefully construct an input signal which is informative enough to estimate a reliably model.
In this section we take a closer look at two methods for the construction of our input signal, namely the *Binary Noise* and *Generalized Biriary Noise*, which can easily be used in practice.

Input design has been a research topic for several decades, and various mathematically based procedures have been developed to design an input signal. See the thesis of v.d. Klauw [Kla95] and references herein for an overview.
Most mathematical ideas are based on reduction of the parameter variance.
Let us not forget that the model error consists of a *bias* part and a *variance* part, like equation *(3.35)* indicates. The bias part is caused by wrong, or under-modelling (the true model cannot be estimated in the chosen model set) and in practice, by the finiteness of tlie data. The variance part or model uncertainty is caused by noise, acting on the process. It can be influenced by changing tlie parameterization and by the test-input signal.
The drawback of most of the input design methods are that they are often not easy to use within a reasonable amount of time in practice and depend on a lot of prior information.

The amount of information in the data depends on the experimental situation. which mainly is determined by the sampling time, the acquisition equipment and the input signal.
An *experiment is informative enough* if the data allows discrimination between any two models in the chosen set.
Ljung shows that, in case of linear model structures [Lju87]:

- √ an open loop experiment is *infornzative* if the test-input signal is *persistently exciting;*
- √ a quasi-stationary signal *{u(k)}* is *persistently exciting* of order *n,* if the frequency spectrum $\Phi_u(\omega)$ is different from zero at at least *n* points in the interval $-\pi < \omega \leq \pi$.

  As a consequence: a maximum number of *n* parameters can be estimated, when we use an

input signal $u(k)$, that is *persistently exciting* of order n. It is thus sufficient to use n sinusoids to identify an 11th-order system.

With the above two statements we can construct an *experiment* that is "informative enough" to generate *data sets* that are "informative enough", which still leaves us with a lot of *freedom*. This can, for example, be used to construct $\Phi_u(\omega)$ so that the variance in the parameters is minimised by choosing some appropriate cost function.
The construction of the input signals in this section scarcely uses this freedom, but surely satisfies the above requirements needed for an informative experiment and doesn't require a lot of a *priori* knowledge.

Binary Noise (BN): a sequence that switches between two values (e.g., -1 and 1) with a non-switching probability $p = 0.5$. The signal has equal power over the full frequency range up to half the basic switching frequency, which is usually chosen so that we have equal power over the whole frequency region of interest in which the system is located. So, every mode is activated with the same amount of energy. The basic switching frequency is the fastest possible switching frequency.
The amplitude of the input signal, often depends on the linearity of the process around the working point.
As the basic sampling frequency determines an upper limit, the length of the experiment determines the lower limit of the frequency area, which can be identified.
As discussed in [Bac89], the length of the identification data has to be chosen 5–10 times the largest relevant time constant to allow reliable estimation of the process eigenvalues.

Advantages of a binary noise test signal in *comparison to other test signals such as a multisine:*
- from an operational point of view, a BN signal is much more acceptable than a test signal which has non-constant amplitude, such as an ensemble of sine wave signals or white noise.
  In the process industry it is preferred that control engineers perform *BN* or step response experiments since this type of disturbance is familiar to plant operators:
- the approach doesn't call for detailed *a priori* knowledge like an optimal multisine wave;
- optimal multisine wave signals concentrate their energy in, or in the neighbourhood of, the natural frequencies, as this is most informative. However plant operators do not easily accept such a concentration of energy at such "critical" frequencies.

BN is also know as Pseudo Random Binary Noise Sequence *(PRBNS)* and among others is used in [Bac89] and [Lju87].

Generalised Binary Noise (GBN): a binary noise signal with a non-switching probability $0 < p < 1$. The sequence has an extra free parameter p, which results in more required *a priori* knowledge than a *BN* to determine p.
This signal is introduced by Tulleken [Tul90] to overcome the serious shortcomings of *BN* in conjunction with one-step-ahead predictor error estimators. Such estimators appear to give an unfair advantage to higher-frequency components, as they only look one step ahead. By

changing the non-switching probability we can manipulate the frequency spectrum of tlie test-input signal. such that most power is concentrated in the lower frequencies.
For controller design this is usually the most important frequency area.

Alternatively tlie *BN* can also be used to put more energy in the lower frequencies by enlarging the basic switching time T'= MT with M=2,3,... The basic switching time is defined as the smallest possible switching time.
An important drawback is hidden in the fact that we are not able to properly excite frequencies higher than w $=2\pi/(MT)$, while the basic switching time is $T'=MT$. The continuous-time spectrum of the M-fold conventional *BN* is $\Phi(\omega)=MTsinc(\omega MT/2)^2$ [Tul90]. We are confronted with *gaps* in our spectrums at frequencies: $\omega = 2k\pi/(MT)$ with k=1,2,3,...On the other hand any *GBN* has it its gaps at more irrelevant frequencies $\omega=2k\pi/T$.

Tn [Tul90] a method is developed to estimate the optimal p for a general SIMO ARMAX system, where it is assumed: the number of observations are "large", so that $\{u_k\}$ and $\{y_k\}$ become almost stationary stochastic signals, and an *asymptotically unbiased* estimator is used. The optimal non-switching probability is estimated by maximising a cost function, which contains the ARMAX parameters, which are not *a priori* known. After a process model has been estimated, one can use this cost function to find a "better" input design, to estimate a more accurate process model.

Tulleken developed guidelines for estimating a sub-optimal p, which he derives from Monte-Carlo experiments with a second order process, with white noise added at the input and output. Of course this is a restrictive choice but a significant amount of processes can be very well approximated with second order dynamics. The guidelines are summarised in *table 3.1* and are calculated by solving a cost function numerically

Process type		T/τ_s	p	$E\{T\}/\tau_s$
First Order		1/32	0.94	1/2
Sec. order *osc.* damped	min phase	1/25	0.8	1/5
	non-min phase	1/15	0.8	1/3
Sec. order *over.* damped	min phase	1/20 to 1/10	0.9-0.95	1
	non-min phase	1/20 to 1/10	0.9-0.95	1

***Table** 3.1: guidelines for sub-optimal p.*

In table 3.1 we use τ_s for the 99 % settling time and $E\{T\}$ for the expected switching time [Tul90]:

$$E\{T\} = \frac{T}{1-p} \qquad (3.36)$$

If another basic switching time T has been chosen than tlie one corresponding to the third column (T/τ_s), p should be adjusted by using the last column together with *(3.36).*
Note that we "only" need some knowledge about a step response (settling time, osc. damp) *a priori* to estimate *p*.

3.7 Pre-Processing of Data

This section describes techniques that have been developed to carry out various signal manipulations to improve the modelling of a process after the input-output data has been collected.
411 polishing actions carried out on the signals, are directed to increase the ratio of relevant information on the process dynamics to disturbances, blurring tliat information.
Just like the other sections it contains only an overview and a more detailed account can been found in [Bac87] and [Lju87].

Pre-processing of the collected process data involves the following processing steps:
- peak shaving;
- trend determination and correction:
- scaling and offset correction;
- filtering:
- delay time correction;
- sample rate reduction.

Peak shaving: is required to reduce the effect of spikes (pealts due to e.g. loose contacts. power failure, cross-talk between cables etc.) on measured process signals. In industrial practice. spikes are often induced in the sensors and the long leads from the sensors to the measuring equipment.
The amplitudes of the noise spikes may be very large coinpared to tlie actual signal range.
If these spikes are not removed from the signals. they may form an important part of the noise energy. As a consequence the spikes can have a considerable influence on the ultimate model, although they have no relation with the process itself. This *a priori* know ledge has to be used for seducing the influence of the spikes which can be accomplishecl by using the following signal processing procedure:
- clip the signal amplitudes to values never to be reached by the real process signals;
- compute a trend signal from the clipped signal;
- compute the standard deviation of the trend-corrected, clipped signal:
- interpolate all samples of the original signal that are outside a band defined by the trend signal plus and minus a times the computed standard deviation, where a is chosen such that no signal value exceeds the permitted signal range.

Trend correction: is generally required as measured process signals often show drifts or variations of which the cause in most cases is known. In general not ali process variables that influence tlie output are selected as input for modelling purposes. These inputs that haven't been selected can however. contribute to the changes in the output, which can be considered as coloured white noise. They don't average out because of their low frequency behaviour and the relative 'short' data length. As a consequence they have a bad influence on the estimation.

Mostly it is impossible to prevent drifts and slow variations during open-loop experiments, but these drifts will eventually be compensated for, by the MIMO control system to be designed.
The trend can be removed by subtracting it from the data. It can be estiinated with a symmetric, non-causal, low-pass filter, as such a filter has no phase shift. The idea is to filter twice: once with a causal low-pass filter, which has negative phase shift, and a second time with the corresponding anti-causal low-pass filter, which has a positive phase shift. The average of the sum of the two filtered signals obtained, will not be shifted in phase. Mathematically this can be expressed by

$$y_{tr} = \sum_{i=0}^{\infty} h_i^c y(k-i) + \sum_{i=-\infty}^{-1} h_i^{ac} y(k-i) = \sum_{i=-\infty}^{\infty} h_i y(k-i) \ , \qquad (3.37)$$

where h^c and h^{ac} are the impulse responses of the causal and anti-causal filters respectively.

Scaling and Offset correction: we want to describe the dynamic behaviour of the process in its working point. so the offsets need to be subtracted. Especially in practice, these offsets can be rather large compared to the signal changes, which could lead to loss of computational accuracy.
In general not all inputs and outputs have the same order of magnitude. The numerical values obtained are related to physical quantities which in general do not even have the same dimension. The signal with the largest numerical value will automatically get the highest weight in the quadratic criterion which is minimised in the estimation phase. As a consequence, if no scaling takes place, the model obtained for the description of the process dynamics, will be good for the input/output combination with the largest numerical values and bad for the input/output combination with the smallest numerical value.
Scaling can best be done with respect to the power contents of the signals.
To make the power of all signals equal, first the average power of each signal is determined. The average power of the dynamic part of the signal is equal to the variance of the signal.
To make the average power of each signal equal to one, each sample, after tlie offset has been subtracted. is divided by its standard deviation, i.e. the square root of the estimated variance. As a consequence all outputs will be equally important for the identification algorithms and the relative accuracy of the various transfers will no longer depend on tlie signal amplitudes.
Let C,, and C, be the diagonal scaling matrices:

$$y_s(t) = \Sigma_y y(t); \quad u_s(t) = \Sigma_u u(t) \ , \qquad (3.38)$$

and the chosen model structure:

$$\mathrm{Y}(s) = G(s)u(s) \qquad (3.39)$$

After scaling, we will estimate the transfer function $G^*(s)$:

$$y\ (s) = G^*(s)u\ (s) \qquad \text{with} \quad G(s) = \Sigma_y^{-1} G^*(s) \Sigma_u \ . \qquad (3.40)$$

So the original transfer function can be recovered by multiplying the estimated transfer with the scaling matrices. as indicated in (3.40).

In case of a state-space presentation:

$$\begin{aligned} x(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \qquad (3.41)$$

the estimated state-space model will be:

$$\begin{aligned} x(t) &= Ax(t) + B^{*}u(t) \\ y(t) &= C^{*}x(t) + D^{*}u(t) \end{aligned} \qquad (3.42)$$

with

$$B = B^{*}\Sigma_{u}\,; \quad C = \Sigma_{y}^{-1}C^{*} \quad \text{and} \quad D = \Sigma_{y}^{-1}D^{*}\Sigma_{u}\,. \qquad (3.43)$$

Filtering: has to prohibit aliasing effects caused bp sampling and has to reduce tlie influences of tlie disturbances on the measured process signals. On the other hand filtering on the data may give rise to undesirable loss of relevant process data. Therefore the design of appropriate filters is an important step.
The design of a filter is mainly governed by the following constraints:

- √ In the pass-band the filters are not allowed to have noticeable influence on the signal characteristics. Therefore they need to have a flat frequency response.
- √ The cut-off frequency of the filters has to be chosen high enough so that the filters don't introduce a significant attenuation and phase shift at frequencies covered by tlie process dynamics.
- √ Disturbances at frequencies beyond the bandwidth of the process have to be removed as much as possible in order to prevent problems caused by folding effects and to reduce the influence of the disturbances.

Considering these constraints, we choose for a Lowpass Butterworth Filter (LBF). as it guarantees a *maximally flat passband* and it has an easy electric realization.
A *digital* LBF can easily be determined with. for example Matlab. The design of an *analog* LBF asks for some extra work and will be treated in this section.
We will point out some of his properties and a design procedure, which makes it easy to develop them. A more detailed discussing can be found in [Val82].

An analog LBF is based on the connection of several second order transfers of the form

(3.44)

with the same ω_0's and different Q's, to get the desired rolloff in the transition band. as *(3.34)* only has a rolloff of +20 dB per decade.
The frequency response of *(3.44)* is maximal flat in the passband with a DC-gain equal to one.

We then have to determine the ω_0's and Q's to meet our specifications. This can be accomplished with the following *design procedure*:
1. specify the filter;
2. find the order n, which has to be an integer:
3. determine ω_0;
4. determine Q;
5. realisation of an electric circuit.

1. Filter specification: The filter is often specified as done in *figure* 3.2

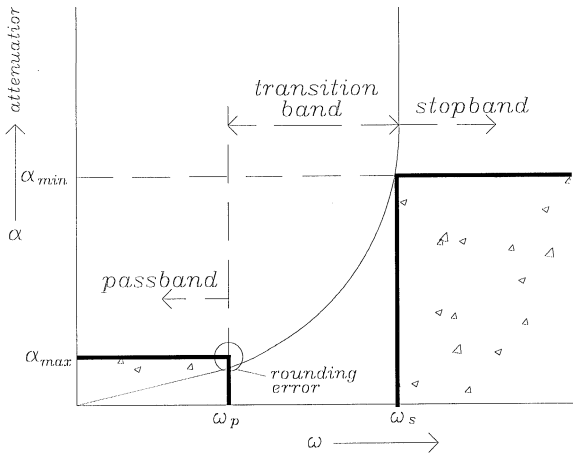


Figure *3.2:* filter *specification.*

Where ω_p, $a_{,,,,,}$ and $m_{,,}$ $a_{,,,,,}$ are used to specify respectively the passband and sperband.
Notice that a is calculated by [dB] = $-20\cdot\log|H(s)|$.

2. The order *n*: is determined with the following formula [Val82]:

$$n = \frac{\log \dfrac{10^{\alpha_{min}/10} - 1}{10^{\alpha_{max}/10} - 1}}{2\log \dfrac{\Omega_s}{\omega_p}} \qquad (3.45)$$

Because n will in general be a non-integer, it has to be round up, to make sure we meet our specifications.

3. Determination of ω_0: A maximally flat passband is realised by making sure that all derivatives of $|H(s)|$, near $\omega=0$, are zero. This is accomplished by choosing $|H(s)|$ such that it yields [Val82]:

$$|H(s)_n| = \frac{1}{\sqrt{1+\left(\frac{\omega}{\omega_0}\right)^{2n}}}, \qquad (3.46)$$

where n indicates the amount of LBF elements of the form *(3.44)*.
From *(3.46)* we can obtain an expression for ω_0:

$$\omega_0 = \frac{\omega}{(10^{\alpha/10}-1)^{1/2n}} \qquad (3.47)$$

Depending on which frequency has to be met exactly (*fig 3.2*), one uses ω_s or ω_p in *(3.47)*. This difference in ω_0 is due to the rounding error in the order n . In *figure 3.2* the effect of the rounding error is indicated, when we are taking ω_s as exact point, that is, the filter response has to go exactly throw ω_s.

4. Determination of Q: it can easily be shown that all n poles [Val82]:
lie in the complex. left half-plane, on a circle with radius ω_0;
- are symmetric with respect to the real axis;
- are separated by the *same* angle.

As a consequences; the position of the poles are exactly know, when the order is given. Considering *(3.44)*, the following expression for Q can be obtained:

$$Q_i = \frac{1}{2\cos\psi_i}, \qquad (3.48)$$

with r=1..n and ψ_i the angle between the negative real axis and the poles in the upper left half-plane.
The parameters Q, only depend on the order and are listed in textbooks like in [Val82].
Notice that *(3.48)* can also, easily be determined when we use the damping parameter ς in *(3.44)*, instead of *Q*. which is more familiar in control theory. We know that $\varsigma_i=cos\psi_i$ and $Q=1/(2\varsigma)$.

5. Realisation of an electric circuit: when ω_0 and all Q_i's have been determined. an analog circuit have to be determined to implement our parameters.
There are several possibilities, which all use one op-amp (operational-amplifier) to realize *(3.44)*. The circuit we will use is shown in *figure 3.3*.

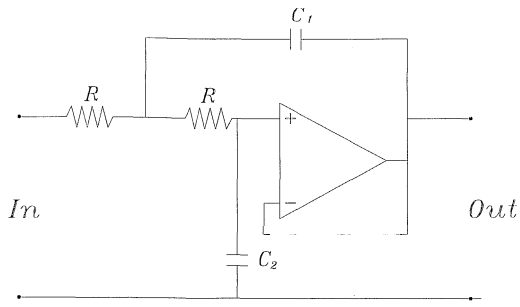


Figure 3.3: *realisation of a LBF element*

With:
$$C_1 = \frac{2Q_i}{\omega_0 R} \; ; \quad C_2 = \frac{1}{2\omega_0 Q_i R} \tag{3.49}$$

Delay time correction: one is often confronted with the problem that the signals to be used for process analysis and process modelling can only be measured with, sometimes significant, time delays. For example when a temperature cannot be ineasured directly, but only from a certain distance.
Time delays can easily be compensated by shifting the input and output relative to each other. For a process with m inputs and p outputs, each output can show a delay in the response to each input, so that finally $m \cdot p$ different time delays may be found. However. the number of delays that can be compensated by shifting the signals against each other is only $m+p-1$.
Time delays remaining after compensation, have to be estimated as part of the model, which will increase the number of parameters that need to be estimated. More parameters will decrease the expected accuracy of the parameters (increase of variance), as they have to be estimated from the same data.
Estimation of the delays can be done with correlation techniques. One method is based on the analysis of the cross-correlation between process input and output signals. For this purpose the process under investigation is assumed to be ergodic. The input signal applied to the process is assumed to be a stationary, white, inter-channel independent. zero mean noise sequence with variance:

$$E\{uu^T\} = \sigma^2 I_p \; . \tag{3.50}$$

Furthermore the input signal is assumed, not to be correlated with the output noise signal $n(k)$. The process response to an arbitrary input signal may be written as:

$$y(k) = \sum_{i=0}^{\infty} M_i u(k-i) + n(k) \, , \tag{3.51}$$

where M_i form the Marlcov parameters.

The final cross-correlation between process output and input is then estimated to be [Bac87]:

$$\psi_{yu}(\tau) = \sigma^2 M_\tau \qquad (3.52)$$

Consequently tlie cross-correlation function obtained will have the form of the impulse response.
If the input signal doesn't match the required assumptions, the cross-correlation function is the convolution of the impulse response with the auto-correlation function of the input. One can also first filter the data by whitening the input as much as possible, as done by Matlab.

The time delaqs can be found from the cross-correlation function by looking for tlie beginning of a sequence of values that differ significantly from zero. Due to the finite length of the data sequence used for the computation of the cross-correlation function, the first elements of the computed correlation function mill not be equal to zero even if they present a pure time delaq.

Sample rate reduction: to build in redundancy that may be used for signal processing, the data is sampled at a frequency 5-10 times higher than the bandwidth of the process. For example, filtering the data analog to overcome aliasing can be done better at a frequency much higher than the process bandwidth. More accurate filtering can better be carried out digitally.
After all pre-processing has been carried out, one has to reduce the sample rate to the frequencq used for building the model. This operation is called *decimation*. Before decimation. one probably has to filter the data (digitally) again to prevent aliasing.

3.8 Final Remarks

After the basic concepts of system identification have been described, which are needed to estimate an "good" model. we can finally start with our identification in chapter 5.

A lot of literature has been devoted to the subjects described in this chapter. Therefore system identification has often been described as an area crowed with seemingly unrelated ad-boc methods and tricks. However, as Ljung points out in his book [Lju87]; the underlying basic ideas are really quite small, and that it is indeed quite possible to orient oneself in the area of system identification with these basic ideas as a starting point. This is the reason why only identification concepts were described in this chapter, which will be used as a starting point in chapter 5.

4

Controller Design

After an appropriate process model has been estimated, several controllers can be developed. The basic concepts of the controllers, that need to be developed, are outlined in this chapter. The first section explains the ideas of the LQR controller, which is also used to perform a closed-loop identification.
In section 4.2 we explain the development of a robust controller based on the minimisation of the systems ∞-norm, which gives us the possibility to incorporate process uncertainties in an explicit way.
Only an overview will be given, and in the corresponding sections, one can find literature references for a more in-depth discussion.

4.1 LQR Design

A brief overview of the LQR concepts will be given. A more detailed treatment can be found in [Dam96a], [And90] and [Fran94].

The LQR (Linear Quadratic Regulator) controller belongs to the family of the *state-feedback* controllers. where the states of the process form past of the inputs of the controller.

A well known feature of these controllers is that they use more process information (the states). Therefore, they can in general perform better then when only the process inputs and outputs are used.
State-space design consists of four *independent* steps:
1. determination of the control law;
2. estimation of the states;

3. combining the estimator and the control law;
4. introducing the reference input.

The outputs of our process are equal to the process states, so step 2 and 3 are left out.

Determination of the control law: consider the following process:

$$\begin{aligned} \dot{x}(t) &= A_p x(t) + B_p u(t) \\ y(t) &= C_p x(t) \end{aligned}, \qquad (4.1.a)$$

or in discrete form:

$$\begin{aligned} x(k+1) &= A_p x(k) + B_p u(k) \\ y(k) &= C_p x(k) \end{aligned}, \qquad (4.1.b)$$

where the time index k is equal to iT with $i = 1..\infty$, with T indicating the sample time. Notice that *(4 1.17)* doesn't indicate the discrete counterpart of *(4.1.a)*, as they are using the same symbols A. B, C, D to indicate *a* process.

The control law has the following form:

$$u(d) = K(d) \cdot x(d) \quad \textbf{with} \quad K(d) = [K_1(d) \;\; K_2(d) \;\; \cdots \;\; K_n(d)] \text{ and } x(t) = \begin{bmatrix} x_1(d) \\ x_2(d) \\ \cdots \\ x_n(d) \end{bmatrix}, \qquad (4.2)$$

were the time index d can be either the continuous time index t or the discrete time index k. Combining *(4.1)* with (4.2) yields:

$$\begin{aligned} \dot{x}(t) &= (A_p - B_p K)x(t) \\ y(t) &= C_p x(t) \end{aligned}, \qquad (4.3.a)$$

$$\begin{aligned} x(k+1) &= (A_p - B_p K)x(k) \\ y(k) &= C_p x(k) \end{aligned} \qquad (4.3.b)$$

Note that no reference signal appears in *(4.3)*, as the *original* regulator problem is to apply a control signal $u(t)$ such that the states $x(t)$ are returned to zero as quickly as possible. In step four, the reference input will be introduced.

For an nth-order SISO system, there will be n feedback gains, $K_1,..,K_n$. Since the system has n poles, it is possible that there are enough degrees of freedom to select *arbitrarily* any

desired pole location by choosing tlie proper values of K, (pole-placement problem), provided that the state-space realisation of the process is controllable.
How do we choose the closed-loop pole locations ?, or in other words; how do we find tlie correct value for K, in *(4.2)*. One approach is called *optimal* control, which minimises a cost-function. LQR makes use of the following cost function (performance index):

$$J = \frac{1}{2}\int_{t_o}^{t_f}\left(x^T(t)Qx(t) + u^T(t)Ru(t)\right)dt + \frac{1}{2}x^T(t_,)P_f x(t_f) \qquad (4.4)$$
$$\text{with } Q \geq 0,\ R > 0,\ P, \geq 0$$

The matrices Q and P_f are symmetric, non-negative definite; and R is symmetric, positive definite and are used to "tune" the *performance requirements*.

Note that the cost function is considered for continuous process inodels . However if we consider discrete process models, like the ones given in *(4.1.b)*, the integral will become a summation and the time index t will indicate the discretization of the continuous time interval.

By means of R, Q, t_f and P_f we can give different weightings to the cost of control and tlie cost of deviation from the desired state. The choice of these quantities is more an art than a science.
We will restrict ourselves to linear time-invariant feedback control, where K isn't a function of *t*. The so called time-invariant solution can be found by letting t_f go to infinity. In this case the matrix P_f no longer makes sense, while $x(t_f)=0$ for $t_f \to \infty$, such that it can be omitted.

Solving (4.4) in case $t_f \to \infty$, when dealing with a *continuous* process model [Dam96a]:

$$K = -R^{-1}B_,^T P, \qquad (4.5.a)$$

where P is the (unique) symmetric positive definite solution of the Algebraic Riccati *Equation*:

$$PA_, + A_p^T P - PB_p R^{-1} B_p^T P + Q = 0. \qquad (4.5.b)$$

Solving (4.4) in case $t_f \to \infty$, when dealing with a discrete process inodel (Dam96a):

$$K = -\left[R + B_p^T P B_p\right]^{-1} B_p^T P A_p, \qquad (4.6.a)$$

m here P is tlie (unique) symmetric positive definite solution of the *Discrete* Algebraic Riccati *equation*:

$$P = A_p^T P A_p - A_p^T P B_p [R + B_p^T P B_p]^{-1} B_p^T P A_p + Q. \qquad (4.6.b)$$

This leads to an asymptotically stable closed-loop system, while $x(t_f)=0$ for $t_f \to \infty$, provided that the process is stabilizable and detectable.

In comparison with other techniques that find an appropriate K, LQR offers a convenient way of including the control effort that is allowed. However, it doesn't allow explicit control of the transient response.

Introducing the reference input: the state feedback K, as determined in the preceding part, brings the states as soon as possible to zero. However, in practice, the output, which is a function of the states. has to follow a certain reference signal, with a zero steady state error.
In [Fran94] two approaches are discussed for introducing the command input signal.
One method re-defines the old states. The LQR controller will then be developed to force the *state-error* to become zero, instead of the *states* themselves. The state-error is the difference between the measured, and the desired sate. The desired states reflect the desired outputs.
Knowledge of the desired states has to be available to ensure a zero steady-state error between the output and the reference signal. which involves exact knowledge of the matrices A_p, B_p, C_p, D_p, which is seldom available. As a consequence, we can say that the method is not very robust, as a small process change will cause the steady-state error to be 11011-zero.
The above method has a *feedforward* control structure, whereas the following method has a *feedback* structure and it doesn't make use of "any" process knowledge.

A common method for tracking signals is by introducing integral action into the controller. *New* states are introduced to include the integral action, which will cause the steady state error to become zero. without the need for process knowledge. In this section we will introduce direct integral action, which causes the steady state error to become zero for step-wise inputs, which are most common in practice.

Consider again *(4.1.a)* and introduce:

$$x_I(t) = y(t) - r(t). \qquad (4.7)$$

Thus the augmented process becomes:

$$\begin{bmatrix} \dot{x}_I(t) \\ \dot{x}(t) \end{bmatrix} = \begin{bmatrix} 0 & C_p \\ 0 & A_p \end{bmatrix} \begin{bmatrix} x_I(t) \\ x(t) \end{bmatrix} + \begin{bmatrix} 0 \\ B_p \end{bmatrix} u(t) + \begin{bmatrix} -I \\ 0 \end{bmatrix} r(t), \qquad (4.8)$$

and the feedback law:

$$u(t) = \begin{bmatrix} K_1 & K_0 \end{bmatrix} \begin{bmatrix} x_I(t) \\ x(t) \end{bmatrix}. \qquad (4.9)$$

They will result in a control configuration as shown in *figure 4.1*

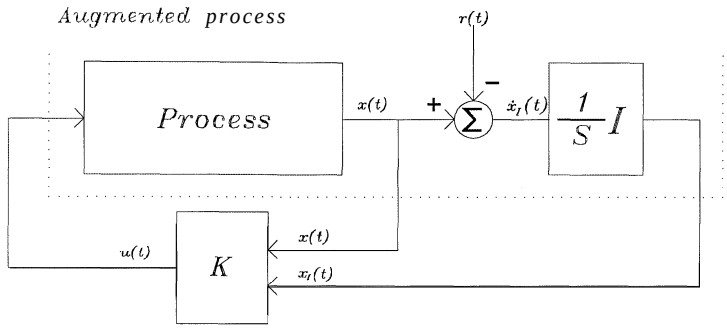


Figure 4.1: the *augmented* process.

Figure *4.1* corresponds (again) to the original problem formulation, for where the LQR controller was developed, and *(4.5)* and *(4.6)* were derived.
The LQR controller is determined for the augmented process. However, the total controller exists of the feedback gain K and the (included) integrator.
The signal *e(t)* can be seen as a state vector containing disturbances, reflecting the reference signal, which need to be brought to zero.
The performance matrices R and Q are used to "tune" the performance requirements, such that we choose the weighting matrix Q so, that only the states $x_I(t)$ are weighted in *(4.4)* and *not* $x(t)$, as these are not the ones we wish to bring to zero !

When we are dealing with a discrete process model, we have to approximate the derivatives in *14.7)* This can be done with difference equations. For example by a ZOH approximation. like also was used in appendix A.*1:*

$$\dot{h}(k) = \frac{h(k+1) - h(k)}{T}, \qquad (4.10)$$

Equation *(4.7)* then becomes:

$$x_I(k+1) = x_I(k) + T(y(k) - r(k)) \qquad (4.11)$$

Thus the augmented process yields:

$$\begin{bmatrix} x_I(k+1) \\ x(k+1) \end{bmatrix} = \begin{bmatrix} I & TC_p \\ 0 & A_p \end{bmatrix} \begin{bmatrix} x_I(k) \\ x(k) \end{bmatrix} + \begin{bmatrix} 0 \\ B_p \end{bmatrix} u(k) + \begin{bmatrix} -TI \\ 0 \end{bmatrix} r(k), \qquad (4.12)$$

and the feedback law:

$$u(k) = \begin{bmatrix} K_1 & K_0 \end{bmatrix} \cdot \begin{bmatrix} x_I(k) \\ x(k) \end{bmatrix} \tag{4.13}$$

In [And90] some interesting features, concerning the *robustness* of the optimal *time-invariant* regulator are dealt with, in case *no* estimator is used. We would like to point out two of these features that concern the gain and the phase margin:

- when *no* estimator is used, to reconstruct the states, we are assured to have an infinity gain margin in *each* loop . with a downside limit of ½ ;
- a phase margin of at least 60 degrees.

However, as they share the same properties, we shall only examine them for SISO systems. as MIMO systems are more awkward to consider.
A more in-depth discussion of these properties and more. can be found in [And90].

We shall consider our fundamental open-loop system as given in *(4 1 a)*, that is completely controllable, observable and time-invariant. Let *(4.4)* be our performance index (cost function) with $t_f \to \infty$.
Then, the following will hold [And90]:

$$\begin{aligned} R + B_p^T(-j\omega I - A_p^T)^{-1} Q (j\omega I - A_p)^{-1} B_p = \\ \left[I - B_p^T(-j\omega I - A_p^T)^{-1} K^T \right] R \left[I - K(j\omega I - A_p^T)^{-1} B_p \right], \end{aligned} \tag{4.14}$$

and

$$\left[I - B_p^T(-j\omega I - A_p^T)^{-1} K^T \right] R \left[I - K(j\omega I - A_p)^{-1} B_p \right] \geq R . \tag{4.15}$$

The inequality *(4.15)* is a simple consequence of *(4 14)*, once it is recognised that $N^* QN$ is non-negative definite, with Q non-negative definite and N arbitrary. The operator $(\cdot)^*$ taltes the transpose and the complex conjugate of the matrix $(\cdot)$. Equation *(4.14)* is known as the *return difference equality,* and a proof is given in appendix B.*I*.

For SISO systems *(4.15)* becomes:

$$\left| 1 - K(j\omega I - A_p)^{-1} B_p \right|^2 \geq 1 . \tag{4.16}$$

Or in terms of the sensitivity function:

$$|S| \leq 1, \tag{4.17}$$

for all frequencies, as:

$$S = \left[I - K(j\omega I - A_p)^{-1} B_p \right]^{-1} \qquad (4.18)$$

The left side of *(4.18)* is known as the *return difference,* when the system is given as in *figure* 4.2.

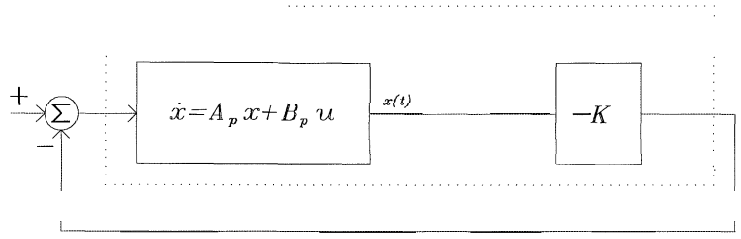


Figure 4.2: *the closed-loop part of the process with LQR control.*

Gain margin: we recall that the gain margin of a closed-loop system is the amount by which the loop gain can be changed until the system becomes unstable.
The Nyquist plot is a curve in the complex plane, obtained from the complex values of $-K(j\omega I-A_p)^{-1}B_p$, as ω varies through the real numbers from minus to plus infinity.
Because of *(4.16),* the Nyquist plot of the closed-loop system avoids a certain region of the complex plane. which is equal to a circle of unit radius centered at -1+j0. That is, the distance of any point on the Nyquist plot from the point -1+j0 is at least unity (*fig.* 4.3).
It is known that when a closed-loop system is multiplied by a constant factor β , it will continue to be asymptotically stable if the Nyquist diagram $-K(j\omega I-A_p)^{-1}B_p$ encircles the point $-1/\beta+j0$ a number of tiines counterclockwise, which is equal to the amount of unstable *process poles.*
Consequently, asymptotic stability is guaranteed for all real $\beta > ½$, as in this case the point $1/\beta+j0$ lies in the avoided circle.

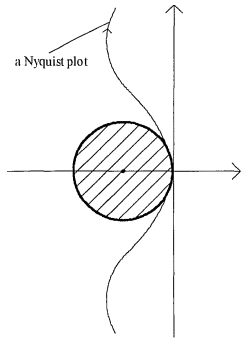


Figure 4.3: *the avoided region by the Nyquist plot for a system with LQR control*

Notice that we already showed that the LQR controller results in a stable system. In this case. the Nyquist plot always encircled the avoided circle counterclockwise, a number of times equal to the amount of unstable *process poles.*

Phase margin: we recall that the phase margin is the amount of negative pliase shift that must be introduced (without gain increase) to make that part of the Nyquist plot. corresponding to $\omega \geq 0$, pass through the point $-1+j0$.
The phase margin is determined from that point, or those points on the $\omega \geq 0$ part of the Nyquist plot. which are at unit distance from the origin.
As the system avoids the circle with unit radius centered at $-1+j0$, the smallest phase margin that can be constructed is 60 degrees. So the phase margin is at least 60 degrees.

The same results. referring to the gain margin and phase margin, hold for the MIMO case, in *each* loop, provided that the weighting matrix R is *diagonal.* The only difference is, that instead of *(4.17),* we obtain:

$$\overline{\sigma}(R^{1/2} S R^{-1/2}) \leq 1, \qquad (4.19)$$

for all frequencies. where $\overline{\sigma}$ is the largest singular value and S and R can be found in respectively *(4.18)* and *(4.4).*

Notice that:
- the robustness properties still hold for tlie augmented process. This can easily be seen by taking a closer look at the proof of the return difference equation in appendix B.1. Hereby keeping in mind that, the reference signal can be seen as an disturbance, influencing some of the states and that the state matrices are expanded, due to the new introduced states:
- in the case when an estimator is used, these features are lost. The more the process differs from our estimated observer model, for reconstructing the states. the smaller the critical circle becomes:
- the robustness properties were derived in case of a *continuous* process model with a *analog* controller.
  Anderson also derives almost the same robustness propel-ties in case of a *discrete* process model with a *discrete* controller. which are about the same, depending on the sample time. For example, there also exists an avoided circle, centered at $-1+j0$, with a radius, that is equal to one when the sampletime approaches zero. However. the gain margin can never approach infinity, as some system poles will be located outside the unit circle. centered at zero, which forms the stability boundary.

4.2 Robust Controller Design

With Robust control we mean that we want to develop a controller which can deal with pre-defined process uncertainties in an explicit manner. That is, we can still guarantee certain performance objectives, regardless of these uncertainties. The performance objectives can be tested by analysing the systems w-norin.
We will begin by defining our process under study and some of his properties. Next we will look how these properties can be used to define our performance objectives and how they can be realised by making use of the systems ∞-norm. We will focus our self on the basic ideas and the problem formulation. A more detailed background can be found in books as [Zho96], [Cla95]. and [Dam96b].

4.2.1 Nominal Performance

Lets us consider the control configuration as shown in *figure* 4.4.

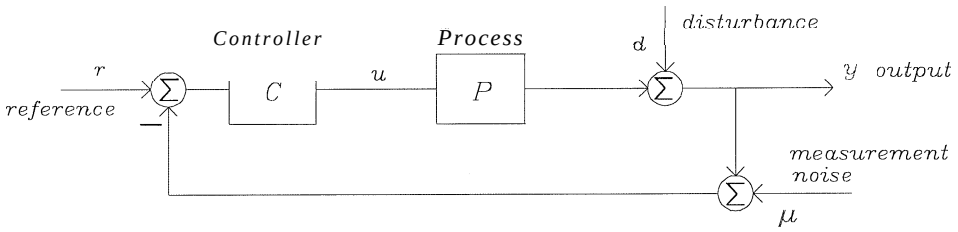


Figure 4.4: *Simple* control *configuration*

we can derive the following:

$$\begin{aligned} y &= T\cdot(r-\mu)+S\cdot d \\ e &\hat{=} r-y = S\cdot(r-d)+T\cdot\mu \\ u &= R\cdot(r-\mu-d) \end{aligned} \qquad (4.20)$$

with

$$\begin{aligned} T &= PC(I+PC)^{-1} \\ S &= (I+PC)^{-1} \\ R &= C(I+PC)^{-1} \end{aligned} \qquad (4.21)$$

Where T, S and R are known as respectively the *complementary sensitivity*, the *sensitivity* and the *control* sensitivity functions.

Notice that:
- for reasons of conveniently, we left out the function argument of the symbols in *(4.20)* and *(4.21)*;
- the above expressions are also known as the *output* complementary sensitivity, the output sensitivity and the *output* control sensitivity functions, as they are measured at the plant *output*. We will. however, not make this difference.

From *(4.20)* it follows that:
- for a good disturbance error reduction, that is for $r - d$ to affect e to the least extent, S should be small;
- for a good measurement noise error reduction, that is for μ to affect e to the least extent. T should be small:
- for the "disturbances" $r - d$ and the noise μ to affect the control input u to the least extent. R should be small.

From above it follows that T, S and R should *all* be made small. However it can easily be seen that the following holds:

$$S + T = 1, \qquad (4.22)$$

which indicates that T. S and R can't be made small in the same frequency area. such that there should be a sort of *trade-off*. Fortunately the spectra of the disturbances d are usually concentrated at low frequencies, whereas the spectra of the measurement noise μ is concentrated at higher frequencies. Thus one may shape Tand S, such that $S(j\omega)$ is "small" at low frequencies and $T(j\omega)$ is "small" at high frequencies.
If both T and S should be small in the same frequency area, we could think of a two-degree of freedom controller, making use of the feedback controller C' *(fig.* 4.4) and a pre-compensator, located at the reference input.
We would like to have a scalar measurement of the systems T, S and R to be able to speak about a "small " T (or S, R). For SISO systems we can easily use the maximum absolute value of the complex frequency responses $|T(j\omega)|$, $|S(j\omega)|$ and $|R(j\omega)|$. For the MIMO case this is not that straightforward. We could think about using the *spectral radius* p as a scalar measurement:

$$\rho(A) = \max_i |\lambda_i|, \qquad (4.23)$$

where λ_i are the eigenvalues of the matrix A. However the eigenvalues may give poor indication of the "gain" of a transfer function, if the gain is measured as the 2-norm ratio of the output y to the input u. See [Cla95] for a nice example, indicating this drawback..
We will use the $\mathcal{L}_2$- induced norm as a scalar measurement, defined as:

$$\|H\|_{(2,2)} \triangleq \sup \frac{\|H(u)\|_2}{\|u\|_2} \quad \text{with } u \in \mathcal{L}_2 \qquad (4.24)$$

"Sup" stands for supremum. $H(j\omega)$ is a stable transfer function matrix and $\mathcal{L}_2$ is a normed space known a the Lebesgue *space* for continuous signals, defined as:

$$\mathcal{L}_2 = \left\{ u(t) \;\middle|\; \|u(t)\|_2 = \sqrt{\int_{-\infty}^{\infty} u(t)^T u(t) \cdot dt} < \infty \right\}, \qquad (4.25)$$

with $u(t) \in \Re^{i \times 1}$ a vector as function of time.

Let the $\mathbf{H}_\infty$ norm of a stable transfer be defined as:

$$\|H\|_\infty \triangleq \sup_{\omega \in \Re} \bar{\sigma}(H(j\omega)), \qquad (4.26)$$

where $\bar{\sigma}(\cdot)$ is the largest singular value.
It then can be shown that [Dam96b]:

$$\|H\|_{(2,2)} = \|H\|_\infty \qquad (4.27)$$

So a small $T(j\omega)$ now means that $\|T(j\omega)\|_\infty$ should be small, or in other words we need to minimise $\mathbf{H}_\infty$ norm, by calculating a correct stable controller $C(s)$.

Weighting functions can be used to reflect the frequency area of interest. A typical performance specification for robust control is given as a weighted sensitivity function:

$$\sup_\omega \bar{\sigma}(W_{p2}(j\omega)S(j\omega)W_{p1}(j\omega)) = \|W_{p2}(j\omega)S(j\omega)W_{p1}(j\omega)\|_\infty \le 1 \qquad (4.28)$$

where W_{p1} and W_{p2} denote the input and output weight respectively, as shown in *figure 4.5*.
The *normalised* input *vector* d' is assumed to belong to the Lebesgue space, which we defined in *(4.25)* with their norm bounded by 1.
The input weight W_{p1} is used to transfer the normalised inputs to the physical inputs. For example, disturbances $\mathbf{d}$ are usually expected to have small amplitude at high frequencies. Thus, if disturbance rejection is of primary interest a low pass filter would be a possible choice for W_{p1}.
The output weight W_{p2} is used to trade-off the relative importance of the individual errors in e and to weight the frequency range of primary interest.
So the weighting function offers us a tool for specifying our input and output signals in a physical set-up. However, in practice, the filters are often designed without any physical background, concerning the input and output signals. They are adjusted until the desired performance is met.

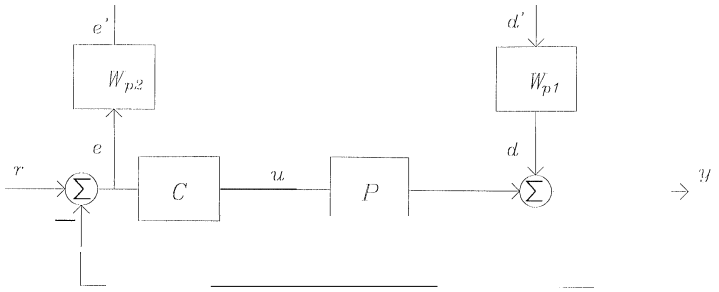


Figure 4.5: *Nominal performance problem.*

The nominal performance objective is defined as follows:

Nominal Performance Problem: the nominal performance problem is, given the weighting function W_{p1} and W_{p2}, to design a stabilising controller $C(s)$ such that the cost function:

$$J = \left\| W_{p2}(s) S(s) W_{p1}(s) \right\|_\infty \tag{4.29}$$

is minimised. Thus:

$$C(s) = \arg \min_{C(s) \text{ stabilizes}} \left\| W_{p2}(s) S(s) W_{p1}(s) \right\|_\infty , \tag{4.30}$$

where $\mathbf{S}$, denotes the set of all stabilising controller. If a controller can achieve $\| W_{p2} S W_{p1} \|_\infty < 1$, we say that the closed-loop system has **nominal performance.**
Problem *(4.30)* is a *standard* $\mathbf{H}_\infty$ *problem* which can be solved with well known techniques, that are implemented in the μ-Analysis and Synthesis Toolbox [Bal93].

A very convenient way of formulating the nominal performance problem is by use of the 2x2 *Block Problem Formulation*, as shown in *figure* 4 6. The *generalized plant N(s)* contains the nominal plant *P(s)* as well as the weighting functions to reflect the nominal performance objectives. If set-point changes are of primary importance, d' may be replaces by a normalised reference r'.

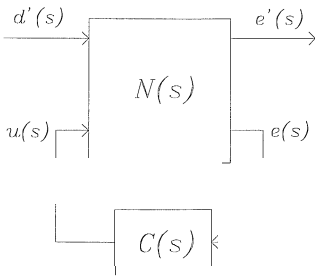


Figure 4.6: *the 2x2 block nominal performance problem.*

The transfer function from d' to e' is given by the *linear fractional transformation* (LFT):

$$\begin{aligned} e' &= (N_{11} + N_{12}C(I - N_{22}C)^{-1}N_{21}) \cdot d' \\ &= W_{p2}(s)S(s)W_{p1}(s) \cdot d' \end{aligned}, \qquad (4.31)$$

where N_{ij} are the matrix entries of the transfer function matrix $N(s)$ in *figure* 4.6.
The above formulation of performance objectives is not restricted to "ordinary" sensitivity problems. If, e.g. disturbances enter the loop not only on the output y but also on the input. or even as an additional input to $P(s)$, this will merely change the transfer matrix expression in *(4.28)*.

4.2.2 Robust Stability

We will now come to the main reason for choosing a H, controller strategy. That is. the possibility to incorporate pre-defined process uncertainties and to guarantee closed-loop stability, as long as the process lies in the model set defined by the uncertainties.
The closed-loop stability under above circumstances is known as *Robust stability* and will be explained in this section.

Let us assume that the process in *figure* 4.4 has an output *multiplicative* uncertainty, as shown in *figure* 3.7. Tlie uncertainty $\Delta(s)$ is any stable transfer function.

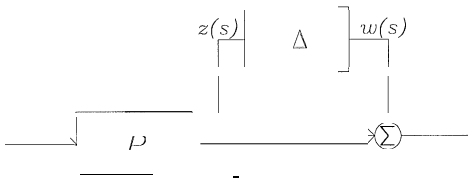


Figure 4.7: *process with multiplicative uncertainty.*

We can *pull* out the uncertainty block, with reference zero, as is done in *figure* 4.8. In our example $M(s)$ equals the complementary sensitivity $T(s)$, but in general it depends on the type of process uncertainty.

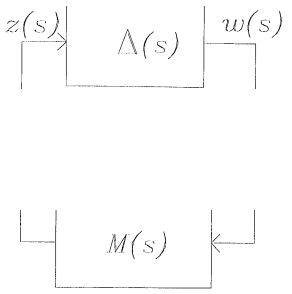


Figure 4.8: the closed loop, *with* pulled *out uncertainty.*

We can guarantee robust stability, bp making use of the famous small gain theory:
Small gain theory [Dam96b]: Suppose that the systems *M(s)* and *Δ(s)* are both stable transfer functions. Then the autonomous system determined by the feedback interconnection of *figure* 4 *8* is asymptotically stable *if*:

$$\|M(j\omega)\Delta(j\omega)\|_\infty < 1 \qquad \forall\omega \qquad (4.32)$$

Using Schwartz equality it follows that:

$$\|M(j\omega)\Delta(j\omega)\|_\infty \le \|M(j\omega)\|_\infty \|\Delta(j\omega)\|_\infty, \quad \forall\omega \qquad (4.33)$$

It then can be proven that [Zho96], when:

$$\|\Delta(j\omega)\|_\infty \le \frac{1}{\gamma}, \qquad \forall\omega \qquad (4.34)$$

where γ is a positive constant, the feedback connection of *figure* 4.8 is asymptotically stable *if, and only if*:

$$\|M(j\omega)\|_\infty < \gamma \qquad \forall\omega \qquad (4.35)$$

Usually two digital weighting matrices $W_{u1}(s)$ and $W_{u2}(s)$ are introduced, such that:

$$\tilde{\Delta}(s) = W_{u2}(s)\Delta(s)W_{u1}(s), \qquad (4.36)$$

and $\| \tilde{\Delta}(s) \|_\infty \le 1 \ \forall\omega$.

Usually the input weight W_{u1}(s) is used to perform any necessary scaling and $W_{u2}(s)$ is used as a frequency weight to approximate the scaled $\bar{\sigma}(\Delta(j\omega))$. There is seldom any reason not to choose $W_{u1}(s)$ and $W_{u2}(s)$ as diagonal matrices.

The *robust stability objective* is thus, given e.g. an output multiplicative uncertainty specification W_{u2} A W_{u1}, to design a stabilising controller $C(s)$ such that the cost function:

$$J = \left\| W_{u2}(s)T(s)W_{u1}(s) \right\|_{\infty} \qquad (4.37)$$

is minimised. Thus:

$$C(s) = \arg \min_{C(s) \in \Im_s} \left\| W_{u2}(s)T(s)W_{u1}(s) \right\|_{\infty}, \qquad (4.38)$$

where $\Im_s$ denotes the set of all stabilising controller. If a controller can achieve $\| W_{u2} T W_{u1} \|_{\infty} < 1$. we say that the closed-loop system is robustly stable Notice how the structure of the robust stability problem (4.38) equals the structure of the nominal performance problem (4 30) Consequently the robust stability problem may be formulated as a 2x2 block problem as mell. as is shown in *figure* 4 *9*

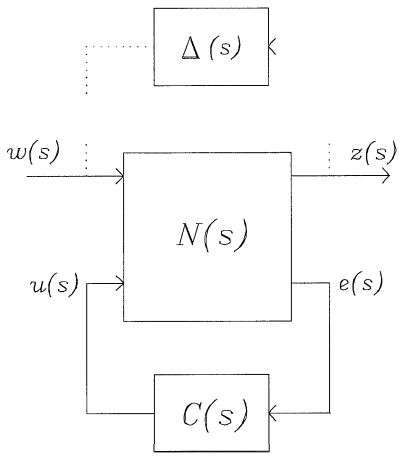


***Figure** 4.9: the 2x2 block robust stability problem.*

4.2.3 Robust Performance

The *robust performance objective* is derived from *(4.29)*, with the nominal sensitivity $S(s)$ function replaced by the perturbed one $\tilde{S}(s)$ due to process uncertainty:

$$J = \left\| W_{p2}(s)\tilde{S}(s)W_{p1}(s) \right\|_{\infty} \qquad (4.39)$$

The *robust performance problem* in *figure* 4 *10*. is obtained by combining *figure* 4 6 and *figure* 4 *9* and know as the $N\Delta K$ formulation, where K is the stabilising controller

The robust performance problem can be formulated as:

$$C(s) = \arg \min_{C(s)\ \text{stable}} \sup_{\|\tilde{\Delta}\|_\infty \le 1} \left\| W_{p2}(s)\tilde{S}(s)W_{p1}(s) \right\|_\infty , \qquad (4.40)$$

If:

$$\left\| W_{p2}(s)\tilde{S}(s)W_{p1}(s) \right\|_\infty < 1 \qquad (4.41)$$

for all $\tilde{\Delta}(s)$ with $\| \tilde{\Delta}(s) \|_\infty \le 1$, we say that the closed loop system has **robust performance.**

As was already mentioned with the nominal performance problem, the above formulation of robust performance is not only restricted to "ordinary" sensitivity problems.
If we include the controller in *figure 4.10* in *N(s)* (see *jig.* 4.11), we can easily formulate the robust performance problem as an 2x2 problem and determine the *linear fractional transformation (LFT)* from *d'* to *e'*

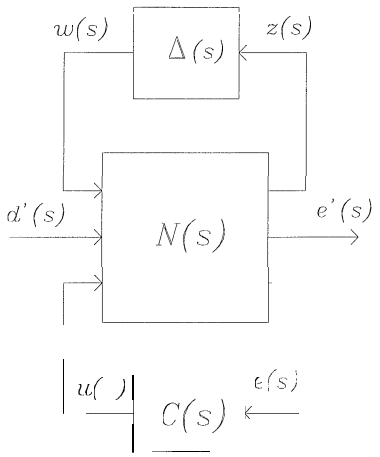


Figure 4.10: *the robust performance problem.*

Notice that the robust performance condition *(4 41)* is similar to the robust stability condition *(4.35)*, when using a normalised process uncertainty $\| \tilde{\Delta}(s) \|_\infty \le 1$. Hence we conclude: the system $W_{p2} S W_{p1}$ satisfies the robust performance condition *(4 41) if and only if* it is stable for a norm bounded matrix uncertainty A, with $\|\Delta_p\|_\infty \le 1$. Thus by augmenting the uncertainty structure with a full complex *performance block* $\Delta_p(s)$, the robust performance condition can be equivalenced with a robust stability condition. This is shown in *figure 4.11.*

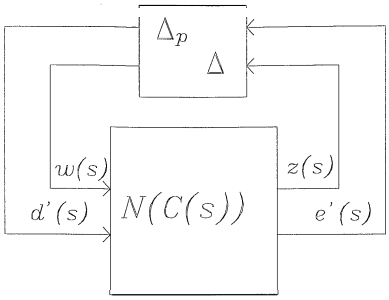


Figure *4.11: the* 2x2 *block robust performance problem.*

Let $\Delta'(s) = \mathrm{diag}\{\tilde{\Delta}(s), \Delta_p(s)\}$ denote the augmented uncertainty matrix. Then the following holds [Cla95]:
Assume that *N(C(s))* and $\Delta'(s)$ are stable transfer functions with $\|\Delta'(s)\|_\infty \leq 1$. Then the system N will satisfy the robust performance criterion *if*:

$$\|N(C(s))\|_\infty < 1, \tag{4.42}$$

which is easily understood from the small gain theory ((4.34) and (4.35)). However. because the structure $\Delta'(s)$ is more restrictive, it is a *sufficient* condition only. That is Δ' has a diagonal structure, which is an element of the set that consists of all full complex uncertainty matrices, for where the *necessary* condition applies.

Clearly robust performance implies both nominal performance and robust stability, such that the latter ones imply a *necessary* condition for robust performance.
We may now formulate a H, problem as:

$$C(s) = \arg \min_{C(s) \in \tilde{\Im}} \|N(C(s))\|_\infty \tag{4.43}$$

If (4.42) holds, the closed-loop system will have robust performance. However since *(4.42)* is a sufficient condition only, it may be *arbitrarily conservative.*
Figure 4.11 can be formulated as a 2x2 block problem:

$$\begin{bmatrix} z(s) \\ e'(s) \end{bmatrix} = \begin{bmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \end{bmatrix} \begin{bmatrix} w(s) \\ d'(s) \end{bmatrix} \tag{4.44}$$

An convenient way to check robust stability, nominal performance and robust performance is the following:

- *robust stability:* let $\|\Delta(s)\|_\infty \leq 1$, then the closed-loop plant is robustly stable *if, and only if* (follows directly from *(4.35))*:

$$\|N_{11}\|_\infty < 1 \tag{4.45}$$

- *nominal performance:* without taking model errors into account ($w = 0$), nominal performance can be guaranteed *if*:

$$\|N_{22}\|_{\infty} < 1 \qquad (4.46)$$

- *robust performance:* taking into account the model errors, the system is said to have robust performance *if*:

$$\|N\|_{\infty} < 1 \qquad (4.47)$$

As mentioned before, we encounter two main limitations in connection with the H_∞ control theory. Probably tlie most important one, is that we can only handle full complex uncertainty structures $\Delta(s) \in C^{m\times m}$ non-conservatively in an H_∞ robustness test. The other main limitation is that tlie robust performance problem can only be considered conservatively *even for full complex uncertainty blocks,* because performance and robustness *cannot* be separated in the H_∞ framework. The conservatism introduced, depends on the applied uncertainty model and the condition number of the plant [Cla95]. We shall now give a short introduction to the *structured singular value* μ, to overcoine above limitations. That is, we will discuss a method that take into account the diagonal structure of $\Delta(s)$ and that can deal with (real) parameter uncertainties.

4.2.4 Structured Singular Value.

In [Cla95] it is shown that *(4.35)* turns into a *sufficient condition* only, when we are dealing with parameter uncertainties, which is easily understood when we realise tliat the scaled parameter uncertainty set [-1;+1], is a more restrictive set than the earlier defined set $\|\Delta'(s)\|_{\infty} \le 1$, for were the small gain theory applies. Because we are dealing with a more restrictive uncertainty set, in general the sufficient condition *(4.35)* can be arbitrarily conservative.
So we have to use some measure, which takes into account the diagonal structure of the uncertainty, such that the "*necessity condition*" is "recovered".

Definition of the structured singular value:
Let M be a matrix $M \in C^{m\times m}$, then the positive real-valued function μ_Δ is defined as:

$$\mu_{\Delta}(M) \triangleq \frac{1}{\min\{\overline{\sigma}(\Delta) \mid \Delta \in \Psi, \det(I - M\Delta) = 0}, \qquad (4.48)$$

where the structured set ψ is defined as:

$$\Psi = \{ diag(\alpha_1 I_{r1},...,\alpha_s I_{rs},\beta_1 I_{rp},...,\beta_p I_{rp},\Delta_1,...,\Delta_f) \mid \alpha_i \in \Re,\ \beta_j \in \mathbb{C}, \Delta \in \mathbb{C}^{k\times k}\}, \quad (4.49)$$

which is the set of possible structured uncertainties. Notice that the full complex block Δ (which can be a performance block) is of course just a special element of the above set ψ. The positive integers *ri, rj* and *k* are used to specify repeated blocks and the full complex blocks respectively.

When **M** is a transfer matrix. we can interpret $1/\mu_{\Delta}(M(j\omega))$ as the size of the smallest uncertainty $\Delta(j\omega)$ which shifts the Nyquist plot of the transfer matrix $M(j\omega)$ to the Nyquist point -1 at the frequency ω.

Notice that:

√ the maximum singular value can be defined in the same way as *(4.48)* [Zho96]:

$$\bar{\sigma}(M) \hat{=} \frac{1}{\min\{\bar{\sigma}(\Delta) \mid \text{A is } \textit{unstructured}, \det(I - \text{MA}) = 0} \qquad (4.50)$$

√ Equation *(4.48)* is equivalent to:

$$\mu_{\Delta}(M) = \max_{\Delta \in \Psi} \rho(M\Delta), \qquad (4.51)$$

where p was defined in *14.23);*

From *(4.48)* we can directly derive expressions for the robust stability problem and the performance problem:

Let us define $F_u(N(s),C(s))$ (the upper linear fractional transformation) as being the transfer from w(s) to z(s) in *figure 4.9* and $\Delta(s) \in \psi$ be the normalised uncertainty with $\|\Delta(j\omega)\|_{\infty} < 1$. The closed-loop system is *robustly* stable *if, and only if*:

$$\left\|\mu_{\Delta}(F_u(N(j\omega),C(j\omega)))\right\|_{\infty} \le 1, \qquad \forall\omega \qquad (4.52)$$

where

$$\left\|\mu_{\Delta}(F_u)\right\|_{\infty} \hat{=} \sup_{\omega} \mu_{\Delta}(F_u) \qquad (4.53)$$

Notice that the left side of *(4.53)* is also is known as $\|\mathbf{F}_{\mathbf{l}}\|_{\mu}$.

We can formulate the robust performance as follows:

Let an $\mathbf{H}_{\infty}$ performance specification (normalised) be given on the transfer function from *d'* to *e'* (typically a weighted sensitivity function), of the form:

$$\left\|F_u(N(s),\Delta(s),C(s))\right\|_{\infty} < 1 \qquad (4.54)$$

Then $F_u(N,\Delta,C)$ is stable (robustly stable) and $\|F_u(N,\Delta,C)\|_\infty < 1$ (*having robust performance)* $\forall\ \Delta(s) \in \psi$ with $\|\Delta(j\omega)\|_\infty < 1$ *if, and only if*:

$$\left\|\mu_\Delta(N(j\omega))\right\|_\infty \le 1, \qquad \forall\omega \qquad (4.55)$$

where *N(s)* is the transfer function as indicated in *figure 4.11*, and the performance block is just an element of $\Delta(s)$.

With *(4.55)* we can test *both* robust stability and robust performance in a non-conservative manner. So performance and stability conditions are now separated. much tighter uncertainty descriptions may be given due to the diagonal structure on ψ and non-conservative results are provided for all uncertainty models.

As we have shown above, μ is a powerful tool for assessing robust stability and robust performance under structured and unstructured uncertainties. Unfortunately the computation of μ is very difficult and yet an unsolved mathematical problem. However tight upper and lower bounds for μ may be effectively computed for both complex and mixed uncertainty sets. Algorithms for calculating the upper bound and lower bound are nowadays commercially available in the Matlab Toolbox p-Analysis and Synthesis [Bal93].
We will not explain how the p-bounds are used to analyse and develop a controller. See [Zho96] and [Cla95] for a detailed discussing. However, referring to the upper and lower bound we like to notice that from *(4.50)* and *(4.51)* it is not difficult to see that:

$$\rho(\Delta P) \le \mu_\Delta(P) \le \bar{\sigma}(P) \qquad (4.56)$$

However these bounds are insufficient since the gap between the upper an lower bound can be arbitrarily large. Thus these must be tightened, which can be done through transformation on P that *do not* effect $\mu_\Delta(P)$.

4.3 Final Remarks

The LQR controller has some useful robustness properties, when no estimator is used to reconstruct the states. We gave an overview of two of those propel-ties; the gain and phase margin. An in-depth treatment was avoided, as we only want to make the reader aware of these "useful" properties, which gives us motivation for the development of a LQR controller, as will be done in the next chapter.

Tlie robustness properties were derived for continuous process model with an analog controller and for discrete process model with a discrete controller. However, liow will these robustness properties look like, when the *discrete* controller is implemented, that is controlling a *continuous* process ?

If the sample time will be small enough, compared to the systein dynamics, we can probably consider the whole system in the continuous domain (s-plane), such that the robustness properties, like mentioned in section 4.1 will apply.

The LQR controller may show some robustness properties, but can't deal with pre-defined uncertainties in an *explicit* manner, such as the H_∞ theory can. This was our main motivation for applying the H_∞ theory to develop a robust controller.

5

Identification of the Process

The theory outlined in chapter 3 will be used to identify a linear and non-linear process model, of a non-linear process, which was described in chapter 2.

An open-loop identification is performed of the final closed-loop system (including the controller).
A non-linear and a linear discrete process inodel are estimated. Because the non-linear model can't be used to develop our controllers, we use it as a way to test the derived model structure.
The non-linear model is obtained by performing a direct identification. That means that the process inputs are used as the identification inputs, instead of the reference inputs of the closed-loop system.
The identification parameters of the linear model are retrieved from the identified closed-loop, by making use of the known controller model.
The estimated non-linear and linear inodel are validated by simulation, residual analysis and a scalar error measurement.

There is chosen for a open-loop identification of the closed-loop, to
- investigate "closed-loop" identification, as a lot of industrial processes already use feedback;
- the process is poorly damped, as it has its poles "close" to the imaginary axis, which makes it difficult to manage the process during open-loop identification (a practical issue).

It may sound somewhat confusing when we speak about an open loop-loop identification of the closed-loop, to estimate a process model. While in fact, we are performing an closed-loop identification. So we if we speak about the closed-loop identification, we mean that we are identifying a process model by performing an open-loop identification of the closed loop.

Before we collect any data and estimate a process model, which is done in section 5.4, we first need to do some preparation, to be able to perform our identification. Basically, they consist of

- the development of a flexible environment, to be able to collect data (section 5.1);
- the development of a controller, to perform a "closed-loop identification" (section 5.2);
- some final identification preparations (section 5.3):
  - √ choice of the frequencies for model building, collecting data and controller calculation;
  - √ design of the anti-aliasing filter:
  - √ input design.

The chosen working-point (point of linearization) is $h_1 = h_2 = 30\ cm$, which is half of the total output range (0-60 cm). In industrial practice, the point of linearization depends on the use of the process: that is, identification should be done in that part of the output range, which is most interesting. However, because this information isn't available, we simply choose the working point as half of the total output range. For the same reason, all valves (fig 2.1), are set approximately half open.

5.1 Flexible Environment

To perform a closed-loop identification and to control the process, it is necessary to nrite some software that can:

- √ save some system data at a pre-defined sample rate, which will be used to estimate our model:
- √ use a pre-defined input signal:
- √ control the process "properly" at a pre-defined sample rate, so that a closed-loop identification can be performed;
- √ show "some" process information on the screen.

The laborator>-located process was developed by the German company Amira, which has also written some software with Borland C++. to control the process.
The acquisition card, for collecting data, which is mounted in the computer. is made by Amira. The converter card DAC6214 uses 12 bits for D/A and AID conversion. Amira offers an adapter card, which is an extension to the converter card. However, this extension isn't available, which mains that all real-time functions have to be implemented with software.

Unfortunately, tlie software written by Amira, can't be used to perform the above tasks. Mainly this is due to badly organised documentation, which is especially important with the pre-compiled parts of the software. Therefore we can't even use parts of the software, apart from the drivers.

New software has been developed with Borland, C++, version 3.1. The main features are
- saving system data at a pre-defined sample rate;
- use of a pre-defined input signal;
- control of the process at a pre-defined sample rate, such that a closed-loop identification can be performed;
- graphical representation of the input and output signals, with an automatic scaling option, such that data trends can be observed in detail;
- possibility to choose different controllers.

The software has the following structure:
- a main file (*.c);
- √ header file (*..h). Contains among others, all function definitions and declaration, used by the functions present in the function file;
- √ function file (*.c). Contains all functions written by the developer;
- √ Borland library files;
- √ CoolTimer file (*.c), responsible for the real-time functions. It handles the PC-timer programming and is know under the name CoolTimer, written by Bobby Z;
- √ driver file (*.c), which contains the acquisition drivers, necessary for communicating with the acquisition card, written by Ainira.

In [Bra97b] one can find a listing of the program.
It will go to far to discuss the software in detail. However we like to point out some "important" details:
- the software is written with the ANSI-C norm in graphical mode. So no use has been made of the object-orientated possibilities that Borland offers;
- tlie software is written for PC's, that are able to support 640x480 pixels with 16 colors (VGAHI);
- the real-time functions, which are called in the interrupt service routines. cannot contain all possible C-functions when we are using CoolTimer. For example the command *printf* cannot be used in a real-time function. For this reason we used arrays to realise our buffers. as the C-function *malloc* (and more) isn't allowed. This function is used when working with *dynamic* data structures, to allocate memory:
- the buffers are used to:
  - √ save data to file;
  - √ plot data on screen;
  - √ read tlie input data from file.

  The buffers are used by the real-time functions, in the interrupt service routines, because:
  1. we cannot include all C-function in the real-time functions;
  2. to obtain flexibility, such that no data is lost;
  3. to allow an interaction between user and software.

  The last one decides tlie length of the buffers, which corresponds to the "allowed" interaction time. An example of an interaction is a setpoint change. In this case tlie program *waits* for an input from the user, which means that no data is written from the buffer into a file. However, the data is still being processed by the real-time functions in the interrupt service routine, that is, that data is still being written in tlie buffers.

As a consequence, when the buffers are entirely filled, the real-time functions fill the buffers at positions that still need to be read. in case of saving data to file. So *wrong* data will be read from the buffers, which is due to a too long *interaction* time, which depend on tlie fastest sample rate that is used.
The buffers are chosen, that we have a interaction time of about 30 seconds, when the fastest sample frequency is 5 Hz. This seems to be more than enough.
A remedy for the above "problem" would be, to collect the keyboard-input by making use of the PC-keyboard buffer. However, the program with the buffers, seems to work well enough to perform our work and it is not in our interest to develop a piece of software that works "perfect";

- the following files need to be located in the same directory. as from were the program is started:
  - √ egavga.bgi, mhich is a display driver:
  - √ input.ed, when using pre-defined input signals. Tt is an ASCII file, with a column for every setpoint-*offset*. That is, the new setpoint is the sum, of the setpoint before tlie input file is used, and the offset defined in the input file. Each row-entry is a function of time, namely the sample frequency *SampleFreqCon*, which is a value, defined in the Header file;
  - √ fil_ini.ed, when using the digital filters.
  - √ sim_ini.ed, when using the intern simulation model.
  - √ lqr_ini.ed, when using the LQR controller.
  - √ rob_ini.ed, when using the Robust controller.

  The abov e *.ini file (ASCII files) contain a model description in state-space. The state-space matrices A. B, C and D appear in the *.ini file. in the same corresponding order.
  Notice that fil_ini.ed contains the SISO filter configuration . which corresponds to equation *(5.9)*.
  The size of tlie A matrix and the number of inputs (number of columns *B)* and outputs (number of columns *C)* need to be defined in the header file.
  The *.ini files introduce a lot of flexibility. For example, the calculated robust control (with Matlab), can directly be saved in the corresponding rob_ini.ed file.
  - √ The format is such that the non-zero matrix entries of a matrix A, are listed in either lines or columns. The order of the matrix entries: $A_{1\,1}$, A, $_2$, ..., A,„?,, where n and m indicate the number of rows and columns respectively. In case of system matrices *(A, B, C. D)*. they appear in the corresponding alphabetical order.
- the following file is created in the same directory, as from were the program is started:
  - √ datai.ed (i: file index). This is an ASCII file in which data is stored. that can be used for identification. Each row-entry is a function of time, namely tlie sample frequency *SampleFreqHigh*, which is a value, defined in the Header file. Tlie saved data consists of an index: the 2 setpoints; the 2 pumpvalues; the 2 LQR controller states and the 3 process outputs.

To get an impression of tlie software layout on the screen, some screen dumps can be found in appendix A.*2*.

5.2 Controller Design

It is not in our interest to develop a high performance controller *yet,* as we only need a colltroller to be able to perform a closed-loop identification. The tasks it needs to perform are:

√ overcome actuator saturation during identification;
√ a "fast enough" step response.

With the idea "try simple things first", two equal analog controllers are implemented (after discretization), to pull away the process poles (3 poles), as they are located 'near' the imaginary axis (integration behaviour). That's why this colltroller is called the Pull controller to distinguish the different types of controllers. A pole is placed in the origin to ensure a zero steady state error.
Because all states of the process are available (levels h_1, h_2, h_3), and the Pull controller doesn't work satisfactory in practice, we also develop a LQR colltroller in this section.

To test and compare the controllers, we use a step response of the real process with a controller calculation frequency of 5 Hz . The choices of these frequencies are explained in the next section.
Before data is collected and controller calculation takes place, analog and digital filtering is performed to prevent aliasing . The analog and digital anti-aliasing filters will be described in section 5.3.2.

Pull control: two equal SISO controllers of the form:

$$C_c = K_c \frac{(s+n)^2}{s(s+p)}, \qquad (5.1)$$

are implemented with Tustin's approximation (bilinear approximation)

$$s = \frac{2}{T}\frac{z-1}{z+1}. \qquad (5.2)$$

Where $n < p$, and n, p are the zero and pole respectively in *(5.1).*
Note that this way of developing and implementing an analog colltroller makes it analytically independent of the sample time T.

Through eye inspection, the values of the Pull-controller are set to: $n = 0.025$. $p = 0.05$ and $K_c = 0.4$ (in accordance with above conditions).
Both SISO controllers operate parallel and use *only* the error signal as input: the difference between the reference signal and the output.
In appendix *A.3,* step responses *(fig. a.3.1)* of the controller, with the simulation model and the real process, are shown. In the same appendix, one can find a bode plot *(fig. a.3.2)* from input 1 to all outputs, of the simulation model.

The simulation model is implemented in Matlab and consists of the SISO controllers together with the continuous state-space model, as was derived in appendix A.1. The matrix entries of the linearized model are calculated by assuming that all the process valves are half open. In the next section, more details are given on the calculation of the closed-loop simulation model.

However, like the step response appears to have a reasonable settling time (figure a *3.1.a)*, in practice this seems to be much larger. This is probably due to a long interaction between the two outer columns. as they are controlled independently. For example, lets assume that the setpoint in the first column is kept constant and that the setpoint in the second column is changed (see *fig.* 2.1 for the corresponding numbers of the columns). The controller for the first column will not react, *until* it notices an error in the first column. It would be better to use a controller that also looks to the changes in the other columns, such that it will be able to react sooner/better.
This is exactly what the LQR controller does.

LQR control
In the pull controller we did not use all available *information*, like the level of the third column (h_3). We will develop a controller; a LQR controller, which uses all available output information,
A brief overview of the LQR controller was given in the previous chapter. In section 4.1, independent steps were pointed out to develop an LQR controller.
Because we are measuring all the states, no estimator needs to be developed, which leaves us only with two steps.
In this section we will go through this steps to develop our controller, that is. determining the control law and introduction of the reference signal.

Introduction of the reference input: this is done as indicated in *figure 4.1.* This leaves us with an augmented process with five states: three states from the original process. and two new states. introduced by the integrator.
The state-space presentation of the augmented process is given by equation *(4.8).*

Determination of the control law: is determined by minimising *(4.4)*, for $t_f \rightarrow \infty$, with the continuous augmented process as argument. The time-invariant feedback gain can easily be determined with Matlab. The Matlab function *lqrd* is used, which calculates a discrete feedback gain, from a continuous cost-function. Before the function minimises *(4.4)*, it first discretizes the continuous process and the continuous weighting matrices, using the sample time and the zero order hold approximation (ZOH). See the manual of the Control Toolbox for more information on this function.

Before Q and R are determined, we need to be aware of the non-linearity in the actuator transfer.
Non-linearity is a well know property of most actuators and their static transfer is shown in *figure 5.1.* Notice that this is the static non-linearity of the actuator transfer. The dynamic behaviour of the actuator transfer is much faster than the process dynamics and therefore not of our interest.

Before Q and X are determined, we need to be aware of the non-linearity in the actuator transfer.
Non-linearity is a well know property of most actuators and their static transfer is shown in *figure 5.1*. Notice that this is the *static* non-linearity of the actuator transfer. The dynamic behaviour of the actuator transfer is much faster than the process dynamics and therefore not of our interest.
We have to try that, during identification, we find ourselves in the most linear part of the actuator transfer. such that an accurate *linear* model can be fitted from the data.

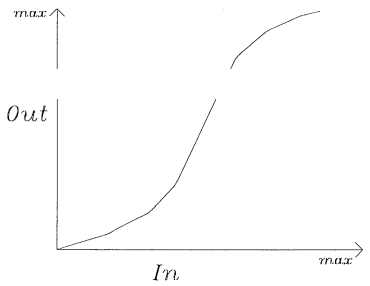


***Figure** 5.1: static-transfer of a family of actuators.*

The working point was chosen as 30 cm, such that the process output is half of the total output range (0-60 cm), in which we find ourselves in the most linear part of the actuator transfer (valves half open).
The weighting matrices are chosen diagonal; Q is fixed and R is changed until no saturation takes place with a step input of 0–>30 cm on both inputs. In this case we are almost sure to find ourselves in the most linear part of the actuator transfer during identification. where we use a step input-change of about 5 cm.

The augmented process is simulated with simulink to verify the estimated feedback gain, before the controller is implemented. Within some iterations, the following feedback gain and weighting matrices were obtained (with a sample frequency of 5 Hz):

$$R = 10^3\begin{bmatrix} 20.5 & 0 \\ 0 & 16.5 \end{bmatrix};\quad Q = \begin{bmatrix} 1.5 & 0 & 0 & 0 & 0 \\ 0 & 1.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}; \tag{5.3}$$

$$K = -\begin{bmatrix} 8.536\cdot10^{-2} & -2.823\cdot10^{-6} & 6.511\cdot10^{-1} & 2.715\cdot10^{-3} & 4.601\cdot10^{-2} \\ 2.959\cdot10^{-6} & 9.514\cdot10^{-3} & 3.372\cdot10^{-3} & 6.733\cdot10^{-1} & 4.605\cdot10^{-2} \end{bmatrix}$$

Notice that:

- the weighting matrices needed to be chosen higher than those originally found using simulink, because of the difference between the simulation model and the real process, like the non-linearity;
- we choose the weighting matrix Q such that, only the states $x_I(t)$ *(4.8)* are weighted in *(4* 4) and *not x(t)*, as these are not the ones we wish to bring to zero;
- the integrators are implemented in software by making use of the Zero Order Hold approximation:

$$s = \frac{z-1}{T}, \qquad (5.4)$$

where *(5.4)* represents a transformation from the s-domain (continue) to the z-domain (discrete):

besides the *static* non-linearity in the *actuator*, the ***process*** also shows ***.static*** non-linearities.

The static non-linearity of the *whole system* will be determined further on in this chapter. which will eventually determine the amplitude of our input signal, used for identification. Note that the actuator and the process mere mentioned separately here. However, unlike noticed otherwise, we will never make this distinction and always mean that the actuator is included in the process, like already was done in the preceding part of our project:

In appendix ***A.3***.a step response ***(fig.****a.3.1*) of the controller, with the simulation model and the real process. is shown. In the same appendix, one can find a bode plot *(fig* ***a*** 3.2) from input 1 to all outputs, of the simulation model. The responses are calculated in the same way as was done with the Pull controller.

The most Important differences between the two controllers are that:

- √ because the LQR controller uses more process information than the Pull controller, the output in an outer column is less sensitive to the input in the other outer column. This can be seen in *figure* ***u.3.l.c***and by comparing the lower frequency areas of their bode plots in ***fig.****a* 3.2 in appendix A.3;
- √ the LQR controller has a smaller bandwidth (*fig. a.3* 2), such that it is less sensitive to high frequency disturbances at the output/input. With the real process, the LQR controller does indeed. show a more calm steady state behaviour. This so called calm behaviour is notable at low water levels (around 20 cm), when the water falls into the column, causing "high" frequency noise (flow ***a*** in *fig* 2.1). Normally the water enters the col1111111 by flowing along the wall (flow *b* in fig. 2.1);
- √ the LQR controller ensures that the bode plots from input 1 and 2 to all outputs are almost the same for the simulation model. This is due to the same weight put on the states by the Q matrix. This means that the step responses for both outer columns are almost the same. whereas both columns have *different* dynamic behaviour (the leakage's are different). The Pull controllers are, however, exactly the same. As a consequence the step responses for both outer columns are different. as well as both their bode plots:
- √ the Pull controller is faster (first intersection with the desired setpoint), but shows an higher overshoot and longer 99%-settling time;

√ The LQR controller shows satisfactory results, and will therefore be used for identification.

5.3 Final Identification Preparations

In this section, the final identification preparations are described, such that in the next section data can be collected and a non-linear and linear model can be estimated.
The preparations consist of input design, development of the anti-aliasing filters and the choice of the "identification" frequencies. That is, the frequencies for calculating the controller and collecting data.

5.3.1 The Identification Frequencies

In the previous chapter we already used a simulation model of the process and a frequency to calculate our controller. In this section we will take a closer look on how they are determined.

A closed-loop model will be calculated, with help of the linear process model. as derived in appendix A.I. to extract information. This information is needed to choose the following frequencies:.
- model building frequency V„„);
- collecting data, used for pre-processing (f_i);
- calculate the controller outputs (f_c);
- collecting data from the process (f_p);

The above frequencies are shown in *figure* 5.2.

Model building frequency V;,): defines the frequency used for final model building (discrete model).
We have to consider [Lju87]:
√ a "small" f_m allows much noise reduction, with possible loss of system dynamics:
√ a "high" f_m could give numerical problems, as all poles cluster around the point 1 in the z-plane;
√ a "high" f_m gives a model fit, which is mainly concentrated to the high-frequency band;
√ a "high" f_m can result in a non-minimum phase model, while the original system is minimum phase.

Ljung determines that the optimal f_m lies near the time constants of the system. This is , however, not exactly known and overestimating them may lead to very bad results.

Therefore he advises to use a f_m which is about ten times the bandwidth of the system to be identified.

Collecting data, used for pre-processing V;.): we choose this 5-10 times the frequency, used for model building to perform some data polishing. before parameter estimation takes place. This so called polishing was briefly explained in section 3.7.

Calculating the controller outputs (f_c): defines the frequency used for calculating the controller outputs

We have to consider [Fran94]:
- when implementing an analog controller in a digital computer, we need to find a digital equivalence by using approximation techniques. A good choice. for getting a good digital approximations, is a frequency of 20-30 times the bandwidth of the total system:
- the frequency must be chosen high enough to reduce the high frequency disturbances in the process output:
- a fast enough response to setpoint changes is often required;

Collecting data from the process V;,): this frequency must be chosen high enough, such that the filter characteristics doesn't have any important influence on the system characteristics.

From the above discussion it follows that we have to obtain an idea of the system bandwidth. This is determined with help of the mathematical (white) model, derived in appendix *A.1*.

In appendix A.*1* the following state-space process model is constructed:

$$\begin{aligned} x_p(t) &= A_{pc}\, x_p(t) + B_{pc}\, u(t) \\ h(t) &= C_{pc}\, x_p(t) \end{aligned} \qquad (5.5)$$

Note that we have include the index 'c', to indicate the continuous model. Index '*d*' will be used to indicate the discrete model.
The parameters of the matrices A_{pc}; B_{pc}; C_{pc} are calculated by assuming that all valves (fig 2.1) are half open. This is a reasonable suggestion to get an "idea" of the process characteristics. as it strongly corresponds to the situation under which identification takes place.

Note that the *dynamic* non-linearity of the process is influenced by changing the position of the valves. For example, when the valve in column 1 is opened more, that simulates leakage, the non-linearit) "increases". This can be understood, as the level in column 1 will increase more slowly. but decreases faster. With a linear model. however, there is no difference between the speed of decreasing and increasing.
The degree of dynamic non-linearity will not be investigated, but can mean that we are not able to fit a linear model to our collected data with only a small error.

With the valves half open we get (appendix *A.1*):

$$A_{p,c} = 10^{-3} \cdot \begin{bmatrix} -2.658 & 0 & 1.608 \\ 0 & -3.315 & 1.608 \\ 1.608 & 1.608 & -4.367 \end{bmatrix}; \; B_{p,c} = 10^{-2} \begin{bmatrix} 3.247 & 0 \\ 0 & 3.247 \\ 0 & 0 \end{bmatrix}$$

$$C_{p,c} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad (5.6)$$

For identification we use the LQR controller, with the configuration as discussed in the previous chapter **(fig I.1)**. The state-space representation of the closed-loop system, is found by combining *(4.8)* and (4.9):

$$\begin{aligned} \dot{x}_s(t) &= A_{s,c} x_s(t) + B_{s,c} r(t) \\ y_s &= C_{s,c} x_s(t) \end{aligned} \qquad (5.7)$$

with

$$A_{s,c} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -2.777 \cdot 10^{-4} & 8.911 \cdot 10^{-8} & -2.382 \cdot 10^{-2} & -8.832 \cdot 10^{-5} & 1.136 \cdot 10^{-4} \\ -9.932 \cdot 10^{-8} & -3.096 \cdot 10^{-4} & -1.097 \cdot 10^{-4} & -2.520 \cdot 10^{-2} & 1.123 \cdot 10^{-4} \\ 0 & 0 & 1.608 \cdot 10^{-3} & 1.608 \cdot 10^{-3} & -4.366 \cdot 10^{-3} \end{bmatrix};$$

$$B_{s,c} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}; \; C_{s,c} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The index '*s*' is used to indicate the closed-loop system.
We used the Matlab function *lqr* to calculate the continuous feedback gain

It can "easily" be seen that the "maximum" bandwidth is determined by the transfer function between input 1 and output 1 (see *fig* 2.I for an explanation of the input and output numbers).
From the bode plot between input 1 and output 1, as shown in *figure a.3.2* in appendix A.3, we can determine:

$$\omega_{3dB} \approx 2 \cdot 10^{-2} rad/s \; ; \; \omega_{40dB} \approx 0.3 \; rad/s \quad . \qquad (5.8)$$

Hereby using [dB]=20·log[..].
Ljung's advises a $\omega_m = 10 \cdot \omega_{BW,\,system}$, which yields $f_m \approx 0.032$ Hz (frequency for model building).

We choose $f_m = 0.1$ Hz, which corresponds to the basic switching time of tlie (generalized) binary noise test signal, which will be designed in section 5.3.3. In this case tlie input test signal doesn't need to be filtered. before parameter estimation takes place.
Notice that this corresponds to a T_m, which is about 18 % of tlie 99 %-settling time, which is about 548 s (section 5.3.3). In [Söd89], T_m is taken as 10 % of the 99 %-settling time, as rule of thumb.

Considering the arguments of Franklin [Fran94], given in this section, we choose $f_c = 5$ Hz (Calculating frequency, used to calculate the controller outputs).

The frequency for collecting data. used for identification; f_i, equals f_c. This means: $f_i \approx 50 f_m$, such that we have a lot of redundancy for pre-processing, which won't cause any problem.

To neglect the influence of the anti-aliasing filters, we choose $f_p = 25$ Hz (frequency for collecting data from the process).
Notice that we need an extra digital anti-aliasing filter, before we lower f_p of 25 Hz to f_c of 5 Hz.This combination of a digital and analog filter, to avoid aliasing effects, is well known in practice, and avoids the use of one expensive analog filter, when not using a digital anti-aliasing filter

Figure 5.2 gives a clear overview of the chosen frequencies and the role of the anti-aliasing filters, which will be developed in the following section.

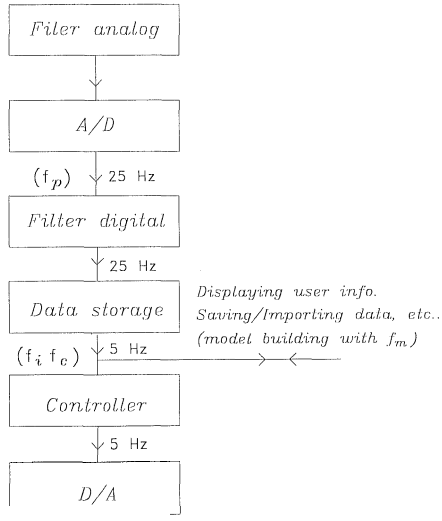


***Figure** 5.2: implementation structure of the filters and their frequencies.*

5.3.2 Anti-Aliasing Filters

In this section we will develop two Lowpass Butterworth Filters (LBF). as outlined in section 3.7.
As shown in *figure 5.2* we need two lowpass Butterworth filters to prevent aliasing effects: a digital LBF before calculating the controller output and an *analog* LBF before collecting the data.
An anti-aliasing filter is acquired to remove (sufficient attenuation) all frequency components higher than half the sample frequency, also known as the Nyquist frequency.

We will go through the design *procedure*, as outlined in section *3.7.* to design our analog lowpass butterworth filter.

Analog LBF: which has to remove all frequency components higher than 12.5Hz (f_p=25Hz).

1. Filter specification: defining tlie parameters values in figure 3.2:
α_{min} = 40 dB; α_{max} = 3 dB; ω_p = 6.5 Hz; ω_s = 12Hz;

2. The order n: with *(3.45)* we find n_a=4 (after rounding). Order four means that we get a cascade circuit with two LBF components.

3. Determination of ω_0: if we choose ω_s as exact point, *(3.47)* yields ω_0=42.42rad/s.

4. Determination of Q: with *(3.48)* we get: Q_1=0.541 and Q_2=1.307

5. Realisation of an electric circuit: if we choose R=10MΩ we find with (3 *49)*: C, $_1$=2.55·10^{-9}F; C, $_2$=2.22·10^{-9}F; $C_{2\,1}$=6.16·10^{-9}F: $C_{2,2}$=0.90·10^{-9}F. However, we have to use normalised values. By using the E-12 norm, which has a 10% tolerance, we get: $C_{1\,1}$=2.7nF; $C_{1\,2}$=2.2nF; $C_{2\,1}$=6.8nF; $C_{2\,2}$=1nF.

In *figure 5.3* the filter frequency response is shown, for the calculated components, and the normalised components. Notice that the error made, by using normalised values, can be neglected.
Don't forget that the components have a tolerance of 10%. This influence isn't investigated in detail, but the frequency response is tested with a spectrum analyser and tlie filter gives satisfactory results.

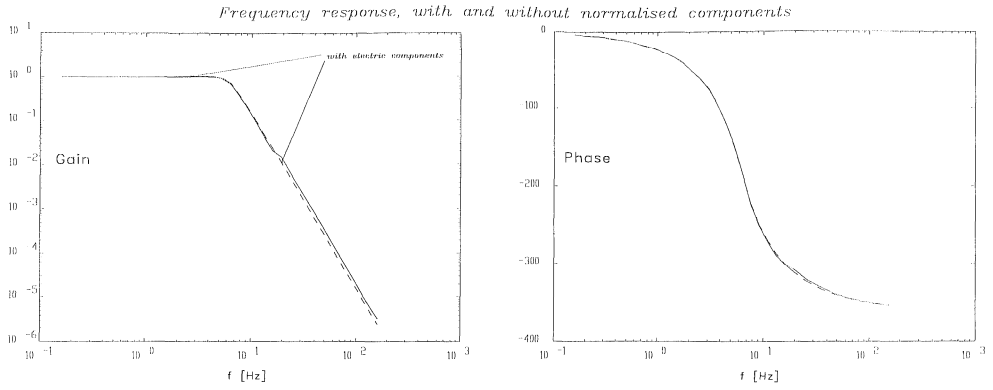


Figure 5.3: *frequency response of the analog* LBF.

The final electric circuit is given in *figure 5.4*

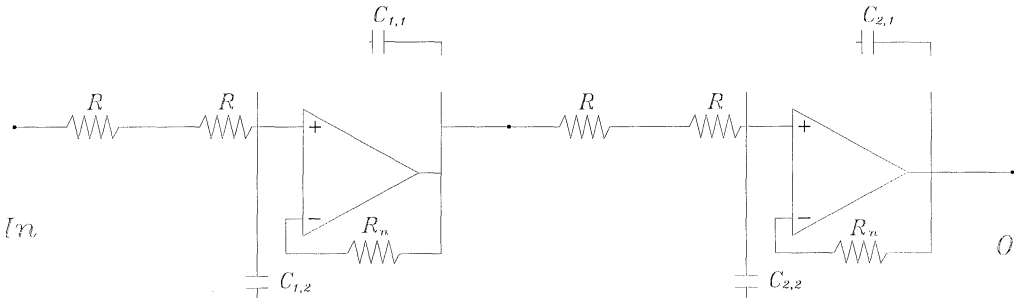


Figure 5.4: *analog lowpass Butterworth filter* to *prevent aliasing effect*

Notice that:

- the circuit in *figure* 5.4 is only for one process output, while the total process has three outputs;
- an extra resistance is placed in the op-amp feedback DC-loop: $R_n = 20$ MΩ, to reduce the DC-offset at the op-amp outputs. This offset can't be ignored, as the DC-resistance at the input is quite large.
  We are using op-amps of the TTL family: LM324, which have a bias-offset current at both inputs of 5nA. The non-inverting input, which has the *same* voltage as the op-amp output (in steady state, without R_n), will then have an offset of $20 \cdot 10^6 \times 5 \cdot 10^6 = 0.1$ V for one op-amp (for one process output: 0.2V). This offset can't be ignored, as the output range of a sensor is only about two Volts. The offset can be reduced by putting a resistance in the DC-feedback loop (R_n). This resistance has to be equal to the total DC input resistance. seen from the non-inverting input: $R_n = 2R = 20$ MΩ. In this case, the offset that appears at the non-inverting input, which is the same as the voltage at the inverting input, will be compensated by R_n. As a consequence the output of one op-amp is "almost" offset free. In practice we still have an offset of about 0.01 V, which can be neglected and can easily be compensated by the software.

Note that one channel has an offset of about 0.2 V, as it contains two op-amp, which corresponds to a level offset of about 10 cm. This can easily be compensated by the software. However. the process will automatically stop a pump, when a level passes the 60 cm limit, which will then already take place at a level of about 50 cm.
A remedy would be to use other op-ainps. For example op-amps of the CMOS family. which hardly has any offset current. However, these were not available;

- m, was chosen as exact point. We could have chosen ω_p as exact point, as these cut-off frequencies are far removed from the real process characteristics. Even so. in our case, a little amount of aliasing can be allowed, as this will be rejected by our digital filter. The combination of an analog and digital filter, to form an anti-aliasing filter. is well know in the field of (practical) signal processing (audio) and is discussed in [Ver95]

Digital LBF: which has to remove all frequency components higher than 2.5 Hz (f_c = 5Hz).
The design is a bit different than that of tlie analog filter, as we make use of the Matlab function butter. This function takes as arguments; the order and the -3dB cut-off frequency. We have chosen for a fifth order LBF and a -3dB cutt-off frequency of 1 Hz, such that $a_{\text{stop}} \geq$ 40dB at m = 2.5Hz.
The state-space presentation (controller canonical form) of the digital LBF, with a sample frequency of 25 Hz (f_p), then becomes:

$$x_f(k+1) = \begin{bmatrix} 4.187 & -7.070 & 6.010 & -2.5704 & 0.442 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} x_f(k) + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} in(k)$$

$$out(k) = 1 \cdot 10^{-3} \begin{bmatrix} 0.197 & 0.063 & 0.343 & 0.052 & 0.031 \end{bmatrix} x_f(k) + 2.140 \cdot 10^{-5}\ in(k) \qquad (5.9)$$

The frequency response of the digital filter is shown in *figure 5.5:*

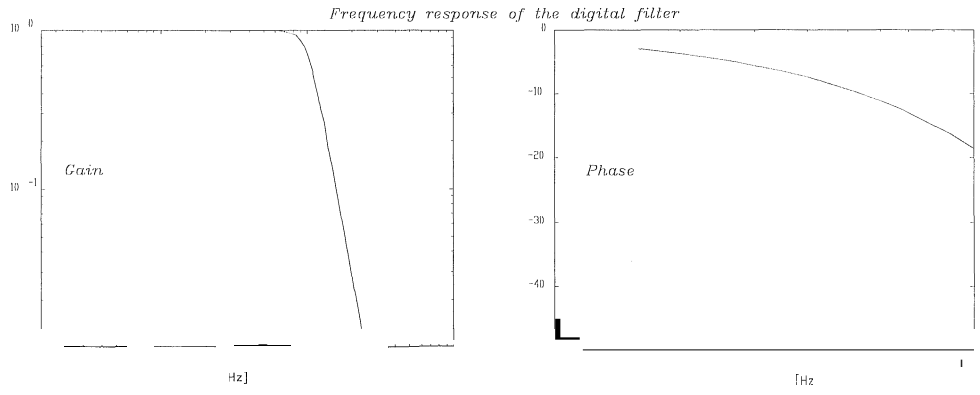


Figure 5.5: *the frequency response of the digital LBF.*

Notice that the values of *(5.9)* are *implemented* with an accuracy of *15 numbers*, while in *(5.9)* only the rounded numbers are shown. In this case, we can neglect the rounding errors. If we use the given values in *(5 9)*, the filter has a step response like in *figure 5* 6.6, instead of that in *figure 5* 6 *a*. It seems that *this* controller-canonical representation is very sensitive to parameter disturbances (not very robust).

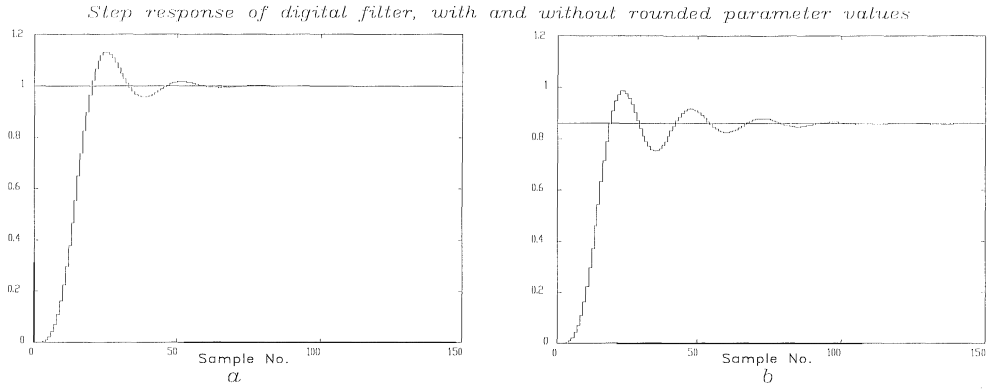


Figure 5.6: *step response of digital LBF, without (5 6 b) and with rounded values (5 6 . 4*

5.3.3 Input Design

In this section we design two kinds of input signals: a Binary Noise (BN) and a Generalized Binary Noise (GBN). Both input signals were discussed in section 3.6.
Two GBN signals are designed, to estimate respectively a non-linear and linear inodel.
First the input signals for the linear model will be developed. At the end of this section some changes are made to the conventional GBN signal. which will then be used to identify a non-linear model.
During the input design. some prior system information is needed, which is extracted from the calculated simulation model. which was determined in section 5.3.1.

The whole system (controller and process), which will be identified. consists of two inputs and two outputs, such that we apply two *independent* input signals at the *same* time.

Binary Noise (BN): a sequence that switches between two values (e.g., -1 and 1) with a non-switching probability $p = 0.5$. The signal has equal power over the full frequency range up to half the basic sampling frequency, which is usually chosen so that we have equal power over the whole frequency region of interest in which the system is located. The basic switching frequency is the fastest possible switching frequency.
What we need to determine is:
1. the basic sampling frequency;

2. tlie length of the input signal;
3. the amplitude.

1. Basic sampling frequency: in section 5.3.1 we obtained a system bandwidth ω_{-40dB} = 0.048 Hz.
We choose a basic sampling frequency of 0.1 Hz to determine an input signal that has equal power in the frequency range of 0–0.048 Hz.

2. Length of input signal: as the basic sampling frequency determines an upper limit, the length of the data used for identification, determines tlie lower limit of the frequency area. which can be identified.
As discussed in section 3.6, we choose the length of the identification data as 5–10 times the largest relevant time constant to allow reliable estimation of tlie process eigenvalues. Tlie largest time constant of the *simulation model* can easily be calculated with Matlab: $\tau_{big} \approx$ 229sec and $\tau_{small} \approx$ 80sec. We then choose an identification data length of $10 \cdot 10^{+3}$ samples. Including data, used for validation, we obtain a final input signal of $20 \cdot 10^{+3}$ samples. Let us remind that the input signal is added to the process at a frequency of 5 Hz and the data is collected with tlie same frequency.

3. Amplitude: we use the following constraints:
- √ tlie process is assumed to be linear. To be able to fit a linear model, tlie amplitude ranges of tlie applied test signals have to be small enough. such that this assumption isn't violated;
- √ tlie output signals should have a sufficient signal to noise ratio. A S/N of more than 20 dB is desirable.

Step one refers to the *static non-linearity*. Notice that the *dynamic* non-linearity will not be investigated.
The static non-linearity of the whole system in the surroundings of tlie working point, can be tested by applying a staircase signal, as shown in *figure 5.7*. This signal is applied to each process input *separately*. The time interval of each step should be chosen in accordance with the response time of the process.

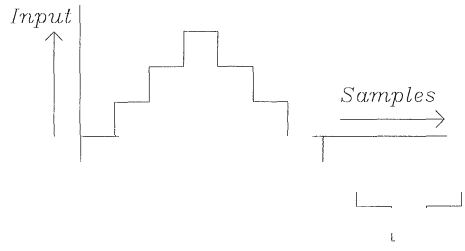


Figure *5.7:* a staircase signal *for* testing the static non-linearity.

Notice that we have a closed loop configuration, while the staircase signal as described above, should be applied in open-loop configuration. The linearity test can then be

performed by taking the *setpoints* as input and the *process inputs* as output. The modified test will show the same static non-linearity of the process. as the controller is linear.

Applying the linearity test to our system, with a step of 3 cm and a working point of 30cm; results in the static transfers that are shown in *figure 5.8* and *figure 5.9* (with an offset correction).

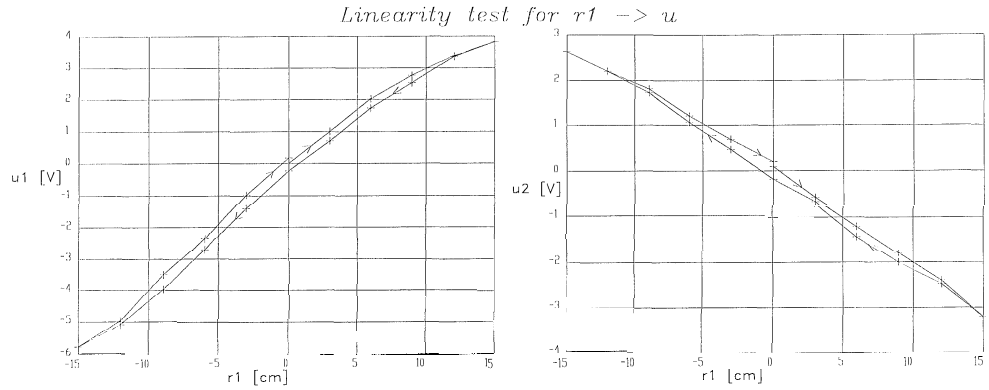


Figure 5.8: static non-linearity test for setpoint 1

The setpoints and the process inputs are indicated as r_i and u_i respectively.

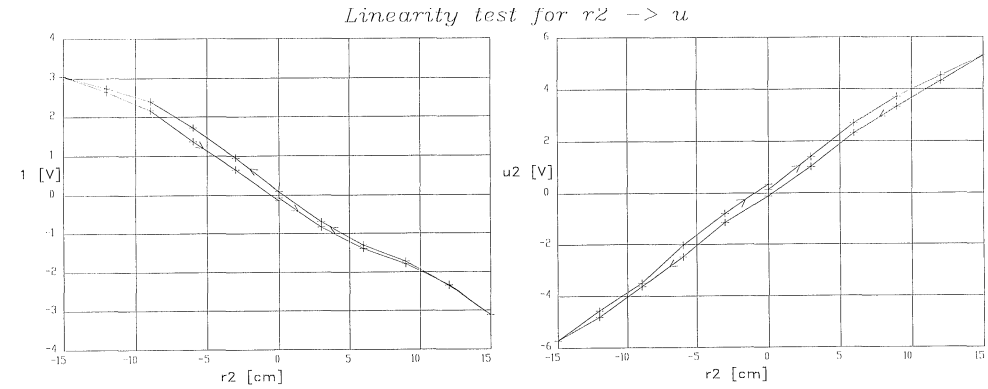


***Figure** 5.9: Static non-linearity test for setpoint 2.*

To make sure tliat we find ourselves in the linear steady-state transfer. we choose an amplitude of 4 cm. The signal to noise ratios of the outputs are given *table* 5.1, together with the standard deviation (Std) of the noise.
The S/N are computed in steady state, from 8 data sets of 1000 samples each, with a sampling frequency of 5 Hz.

	S/N ([dB]=20log(·))	Std.
output 1 (h_1)	88.360 dB	$2.472 \cdot 10^{-2}$
output 2 (h_2)	98.807 dB	$2.148 \cdot 10^{-2}$

***Table** 5.1: the signal to noise ratios and the standard deviations of the process outputs.*

It is not surprising that the staiidard deviations are very low, as we deal with a laboratory-located process. which hardly shows any disturbance influences like an industrial process . Therefore an amplitude of 4 cm should be more than enough.
In *figure* 5.10 the final BN signal for input 1 is shown, together with his power spectrum.

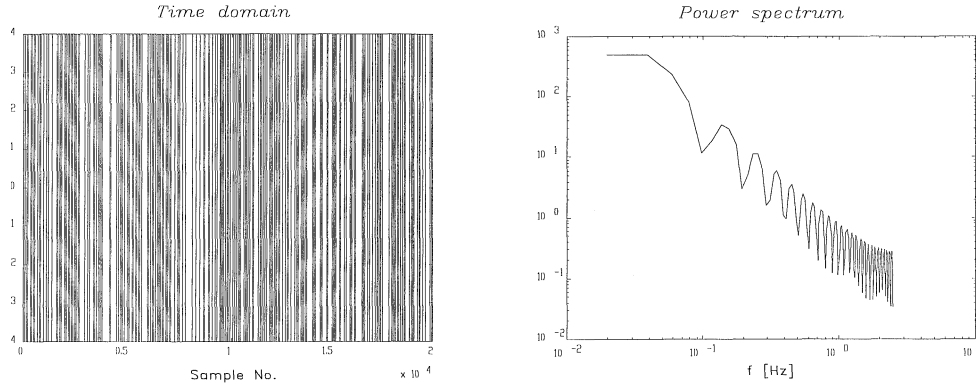


***Figure** 5.10: binary noise test-input signal for input 1.*

The BN signals are constructed with Matlab, by using the uniform random generator.
We have chosen a basic switching frequency of 5/50 Hz, such that the basic switching time is 50 times as big as the sample time, used for controller calculation. In section 3.6 we noticed that this way of constructing an input signal, will cause gaps in the power spectrum at frequencies $f=k/(MT_c)$ with k=1,2,3,... In our case, the gaps appear at 0.1Hz, 0.2Hz,...(M=50), as can be seen in *figure 5.10*. However these gaps are outside our frequency area of interest and therefore irrelevant.

Generalized Binary Noise (GBN): will be constructed as a BN signal. The only difference is that the non-switching probability (p) won't be set to 0.5, but will be determined with the help of some prior system knowledge. For the determination of p. we will make use of Tullekens guidelines, which are summarised in *table* 3.1, section 3.6.

Our system shows dynamics which "look like" that of an over-damped second order process. such that together with *(3.36)* and the last column in table 3.1 we get:

$$p = 1 - \frac{T_{sw}}{\tau_s}, \qquad (5.10)$$

where τ_s is the 99 % settling time and T_{sw} is the basic switching time of 0.2 s.
The settling time is 548sec and is the average of the settling times of h_1 and h_2, measured from there step response: 30 –> 40cm (with the *real* process). Substituting this values in *(5.10)* yields $p = 0.9996$.
The rest of the parameter values, like the amplitude, are the same as for the BN signal.
In *figure 5 11* the final GBN signal for input 1 is shown, together with his power spectrum.

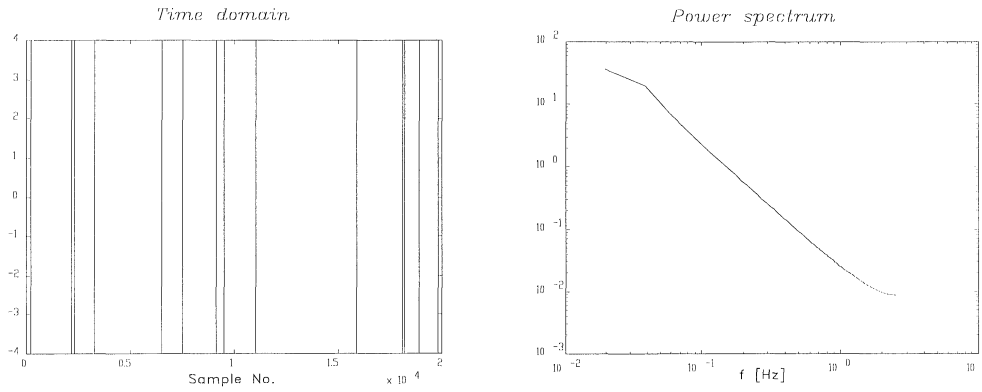


***Figure** 5.11: generalised binary noise test-input ,signalfor input I*

To estimate a *non-linear model*, we have to make sure to activate the static non-linearities of our process as our model is able to capture these non-linearities. In other words; in section 3.6 it was discussed that the input signal need to be *informative enough*, to allow discrimination between any two models in the chosen set. When estimating a linear model this comes down to designing an input signal that is *persistently exciting* of order n. We can say that, concerning the model discrimination, we at *least* have to make sure that the system non-linearit) 's are activated.

The non-linear model can cover more information, so the input signal have to be riches than in the linear case.
We take the same GBN signal as designed earlier and choose an amplitude which switches between 5 and 10 cm.
The non-linear model structure that is derived in appendix *A* I will be used for identification of a non-linear process model. Notice that this model is able to capture *process non-linearity's*. but not *actuator non-lirzearities*, like the one discussed in section 5.2, because they were not encountered in the derived model. For this reason, and to overcome actuator saturation, we chose a maximum amplitude of 10 cm. Whereas *figure 5 8* and *figure 5* 9 indicate that in this area, the static non-linearity just begins to appear. So we would be motivated to choose the maximum amplitude higher than 10 cm.
In *figure* 5.12 the final test-input signal for input 1 is shown, together with his power spectrum.

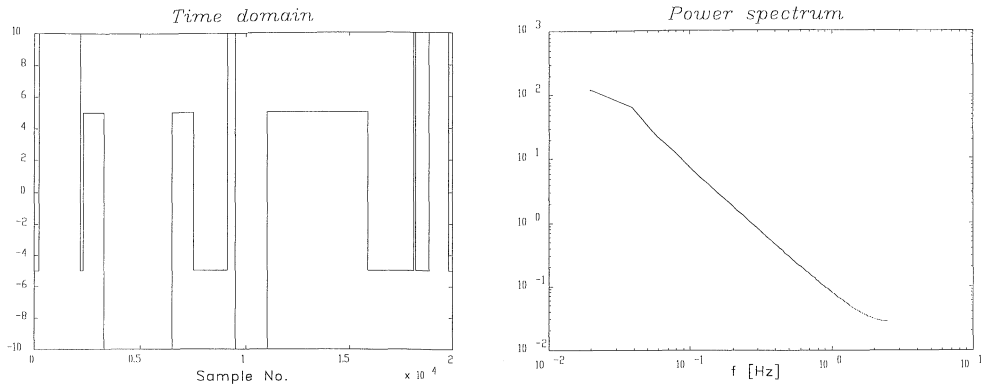


***Figure** 5.12: generalised binary* noise *test-input signal, for input 1 with a variable amplitude.*

5.4 Model Estimation

After the input signals, as developed in the previous section, are applied and data is collected, we can almost start with our model estimation
We will start with some polishing operations to improve the data contents, before the final models (non-linear and linear) are calculated in section 5.4.2. In the same section a validation, of the estimated models, will be carried out.

5.4.1 Pre-Processing

Pre-processing of data concerns the improvement of the ratio of relevant information on the process dynamics to disturbances. blurring that information.
Pre-processing was explained in section 3.7 and is carried out with the help of Matlab.
The pre-processing actions are performed on the data collected with:
1. the GBN input signal, with variable amplitude. We call this the *GBNv data set*;
2. the *BN* input signal (constant amplitude), we call this the BN *data set*;
3. the GBN input signal (constant amplitude), we call this the *GBN data set.*

Notice that the *BN data* and the *GBN data* are used to estimate a linear model. while the *GBNv data* is used to estimate a non-linear model.

Pre-processing involves the following operations (section 3.7):
√ peak shaving:
√ trend determination and correction;
√ scaling and offset correction:

√ filtering:
√ delay time correction;
√ sample rate reduction.

Peak shaving, detrending and scaling will not be carried out. as we are identifying a closed-loop system, such tliat possible trends are corrected by the controller. The system is located in a laboratory. which hardly has any influences of power failures, loose contacts, etc.... causing possible peaks.
Furthermore do the system outputs have tlie same dimension (cm) and about the same variance, such tliat scaling is superfluous.
So this leaves us with the following polishing action:

Offset correction: offset correction is carried out by subtracting the data with the chosen working points. The working points are: $h_{12} = 30$cm and $h_3 \approx 28$cm.

Filtering and sample rate reduction: we have to filter to (section 3.7):
- avoid aliasing effects. as we are performing a sample rate reduction ($f_i \rightarrow f_m$);
- to reduce the influences of the disturbances on the measured process signals.

Step two forms an tighter upper limit than step one. for the filter attenuation.
The lower limit of the cut-off frequency depends on the system characteristics, such that the system will not be influenced too much by the filter characteristics.
The filter is determined by using information of the spectrum ratios, of the output–input signals. which are shown in *figure 5.13. In figure* 5.13 we can distinguish the process characteristics (lower frequency area) and the noise characteristics (higher frequency area). which we try to separate by choosing a correct filter.

Notice that:
√ we can both use the *BN* or GBN data to calculate the ratios, which give the "same" results:
√ tlie Matlab function *spectrum* is used to plot the power spectrum. To plot reliable frequency information, make sure tliat the fourier length is chosen correctly, as it is related to tlie frequency resolutioii. See tlie Signal Processing Toolbox manual for more information. concerning this function.

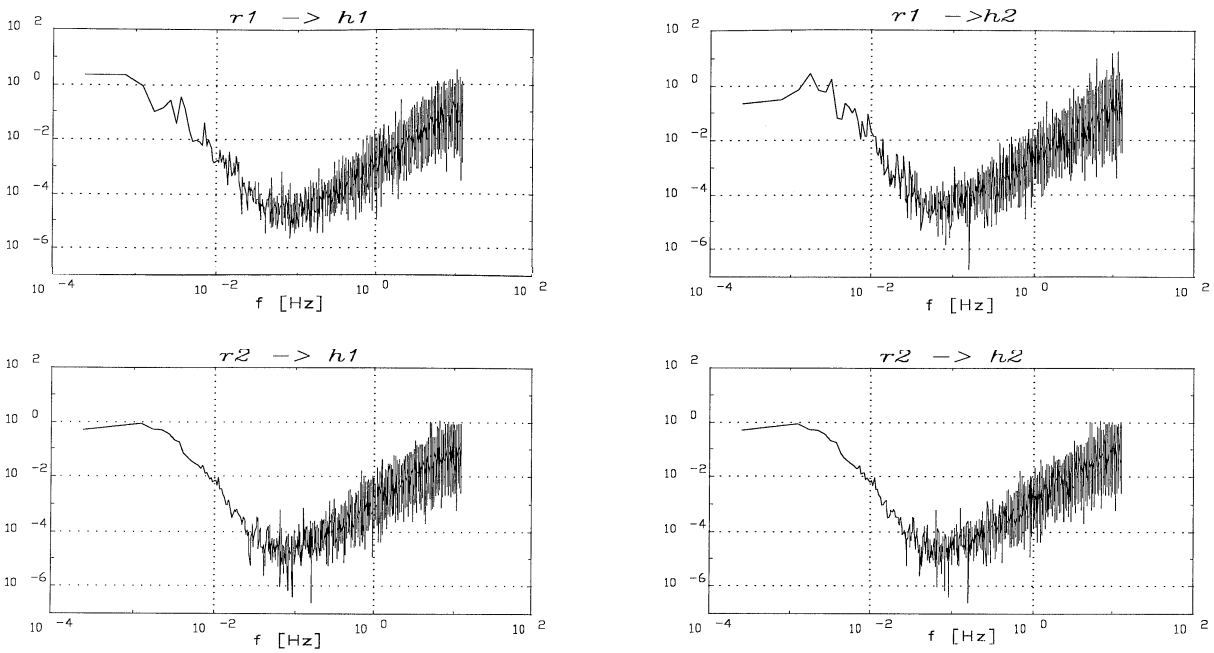


***Figure** 5.13: ratio of the output-input spectrum.*

In *figure 5.14* the frequency response of the filter is shown, which is a sixth order Butterworth with a cut-off frequency of 0.02 Hz, such that $a_{,,,,} \geq 40$ dB at $\omega_s = 0.05$ Hz. The data is filtered by using the Matlab function *idfilt.*

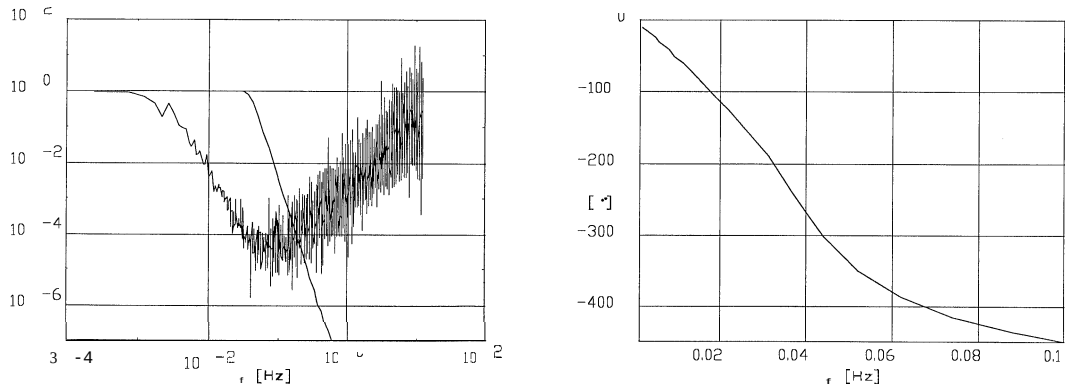


***Figure** 5.14: the frequency response of the digital filter, used for pre-processing.*

In Appendix *A.4; ,figure a.4.1*, the data is plotted after offset correction, filtering and sample rate reduction. Only the data, used for the linear model, is plotted. The data, used for the non-linear model, undergoes, apart from the offset correction, the same polishing actions.
Notice that:

– the GBN *input* signal is filtered as well, before sample rate reduction is performed, as it has a basic switching frequency of 5 Hz, whereas $f_m = 0.1$ Hz;

- the phase shift in *figure 5.14* is almost linear in the area of the system frequency response. Such that the influence of the filter on the system characteristics can be neglected, when *both* the input and output are filtered.
- After filtering, but before the sample rate reduction takes place, the first and last 50 samples are thrown away. This is done, because these samples are contaminated by tlie influences of improper initial conditions of the filters.

Delay time correction: we *measure* the process output without any delaq. such that we shouldn't expect a delay correction. However we are performing a closed-loop identification. and the controller used for model building works with a frequency (f_m = 0.1 Hz) that is different than that of the controller, used for collecting data (f_c = 5 Hz).The controllers have one time delay (the controller exists of an integrator and a feedback gain; *fig. 4.1)*. The controller. used for model building, will therefore have a delay of 10 s, while the delay of the implemented controller is only 0.2 s. We therefore have to shift our output data 9.8 s (also the process input), relativelq to our system inputs (the setpoints), to compensate the difference.
Notice that:

√ the parameters of both controller are different, due to different sample times. However, we won't expect any problems, as the controller is just a discretization of our continuous controller, calculated in section 5.2 and that the process dynamics are relatively slow. compared to a f_m of 0.1 Hz:

√ if no shifting is performed, the cross-correlation between the input and the residuals indicates a large value for small positive lags of τ *(3.34)*. This can be explained as follows: Lets assume that we have a large value for $\tau = 1$. This means that the residual and the input *u(k-1)* are strongly related, such that the residual *takes over the role* of the influence of *u(k-1)* on the model output. as the model isn't able to fulfil this task. This is obvious, as our model structure can only produce an output after about 10 s. while our data already shows an *input influence* after about one second.

5.4.2 Parameter Estimation

Two models will be calculated in this section, a non-linear and a linear model.
The models are calculated as described in section 3.3, that is by minimising a cost function, which reflects the error between the output of the calculated model and the collected data.

For validation we use: an error indication; simulation results and residual analysis. which were explained in section 3.5.
The residual analysis consits of the cross-correlation between the inputs and the residuals. present in the data used for estimation. In this way we can verify if the model structure is able to capture all dynamics, present in the data.

The Error indication is defined as:

$$E = \frac{1}{\text{validation length}} \sum_{j=1}^{l} \sum_{i\,(\text{validation set})} (\hat{h}_j(i) - h_j(i))^2 , \qquad (5.11)$$

which is a summation of the squared-error outputs calculated with the validation set.
With the non-linear model the outputs are formed by h_1, h_2, h_3 and with the linear model by h_1 *and* h_2.

5.4.2.1 Non-linear model estimation

The parameters of the non-linear model structure *(a.1.12) - (a.1.14)* in appendix *A.1* will be estimated, when use is made of the GBNv input signal. The input signal was designed in section 5.3.3
The linear regression model *(a.1.*14):

$$h(k) = \Theta \,.\, \Omega(k - 1) , \qquad (5.12)$$

can directly be used as prediction model, when the noise, acting on the system is assumed to be white.
We calculate the parameters by writing separate regression equations for every output (h_1, h_2, h_3), that can hold the collected samples.
For h_1 this yields *(a.1.12):*

$$\begin{bmatrix} \Omega_1(0) & Q^*(0) & Q,(0) & Q,(0) \\ Q,(1) & Q^*(1) & \Omega_3(1) & \Omega_4(1) \\ & & & \\ \Omega_1(N-1) & \Omega_2(N-1) & (\quad l \quad) & \Omega_4(N-1) \end{bmatrix} \begin{bmatrix} P_{11} \\ \varphi_{12} \\ \varphi_{13} \\ \varphi_{14} \end{bmatrix} = \begin{bmatrix} h_1(1) \\ h_1(2) \\ \\ h_1(N) \end{bmatrix} . \qquad (5.13)$$

Notice that we used $\Omega_i(k)$ to indicate the i th element of the vector Ω, which can be found in (a.*1.12)*.
We solve our *linear problem formulation* with the Matlab operator '\', which uses a QR factorisation to calculate the inverse:

$$x = A \backslash \mathrm{B} \text{ is the solution of } A \,.\, x = B \,. \qquad (5.14)$$

We perform a *direct identification,* that means that we use the process inputs (u_1, u_2) as identification inputs and the process outputs (h_1, h_2, h_3) as identification outputs. In this case we *can't* guarantee that we haven't got a bias when the data length approaches infinity and

the real process model lies in the chosen model structure [Kla95]. This is due to the fact that the identification input is correlated with the noise, acting on the process outputs.
The proof given in [Kla95] counts for linear models, while we have a non-linear model. However it can easily be seen. that, because our model is only non-linear in the samples, it can be written as a linear model by renaming the samples (our linear regression model), for which the proof holds.

We perform a direct identification, because we can't derive *one direct* model betmeen our system in- and outputs, *without* re-constructing tlie process input as an auxiliary signal. This is due to tlie non-linear part of our system function.
Even if we would use our system in- and outputs as identification in- and outputs, we still need to construct our auxiliary signal, such that we are *still* performing a (hidden) direct identification.
We shall see that for the estimation of our linear model, we make use of *one direct* system model, which is tlie discrete version of *(5.7)*. 111 this case we don't use any auxiliary signal. The only signals we use are formed by the system in- and outputs, which we use for identification. The input isn't influenced by the output, so neither by the noise, acting on the output, such that we haven't got the bias problem as discussed above.

The identification data exists of the first half of the total data length and the rest is used for validation. In *table* 5.3 the *estimated* parameter values are given and in *figure a 5 2* (appendix *A.5)* tlie validation by simulation is shown.
The *calculated* parameter values are given in *table 5* 2 and in *figure a.5* I tlie validation by simulation is shown. In this case the valves are assumed half open, as was done in the previous sections to extract some system information.

Parameters	Parameter value	Parameters	Parameter value
φ_{11}	1	φ_{27}	$7.191 \cdot 10^{-2}$
φ_{12}	$-1.151 \cdot 10^{-1}$	φ_{28}	$3.247 \cdot 10^{-1}$
φ_{13}	$-7.191 \cdot 10^{-2}$	φ_{39}	1
φ_{14}	$3.247 \cdot 10^{-1}$	$\varphi_{3,10}$	$-1.151 \cdot 10^{-1}$
φ_{25}	1	$\varphi_{3,11}$	$-7.191 \cdot 10^{-2}$
φ_{26}	$-1.870 \cdot 10^{-1}$	$\varphi_{3,12}$	$7.191 \cdot 10^{-2}$
Error		13.095 cm^2	

Table 5.2: *parameters of the calculated non-linear model,*

Parameters	Parameter value	Parameters	Parameter value
φ_{11}	1 (fixed)	φ_{27}	$4.151 \cdot 10^{-1}$
φ_{12}	$-3.912 \cdot 10^{-1}$	φ_{28}	$3.257 \cdot 10^{-1}$
φ_{13}	$-5.172 \cdot 10^{-1}$	φ_{39}	1 (fixed)
φ_{14}	$3.486 \cdot 10^{-1}$	$\varphi_{3,10}$	$-2.967 \cdot 10^{-1}$
φ_{25}	1 (fixed)	$\varphi_{3,11}$	$-3.941 \cdot 10^{-1}$
φ_{26}	$-6.104 \cdot 10^{-1}$	$\varphi_{3,12}$	$4.947 \cdot 10^{-1}$
Error		$3.100 \cdot 10^{-1}$ cm'	

Table 5.3: *parameters* of the estimated non-linear model.

Notice that:

√ The data, used for the 11011-linearmodel, undergoes, apart from the offset correction, the same polishing actions. No offset correction has been performed, as the estimated model structure isn't only valid in a certain working point. Even so, if we would perform offset correction, we would have to take the root of negative data points, which gives use coinplex parameter values.
We won't loose any accuracy due to no offset correction, as the output variation is about 10% of the working point.;

√ some parameters are fixed, as can be seen in table 5.3;

√ validation is performed with the second half of the original data set;

√ the model input range is: [-10,10] volt and the output range: [0,60] cm.

We like to know if our chosen model structure can capture our data, without lost of physical meaning. That is, do our parameters retain their physical meaning.
The nine identification parameters that we estimated earlier (table *5.3)* consist of eight uncertain parameters, as can be seen from *(a.1.16)* in appendix *A.1*, which are caring our physical significance. Because we have more identification parameters than uncertain parameters, we can't calculate an unique solution for these latter ones, given the identification parameters. It would then be more obvious to directly estimate our eight uncertain parameters, which turns our linear problem into a non-linear problem. Because we want to restrict our parameter search to values with physical meaning we begin by defining parameter constraints, as was discussed in section 3.3.

The uncertain parameters are formed by the section areas of our process valves and the static pump gains (a.*1.16* appendix A.*1*). We define feasibility *regions* of these parameters, as knowledge about them is available.
A feasibility region for every parameter is defined as in *figure 5.15*.

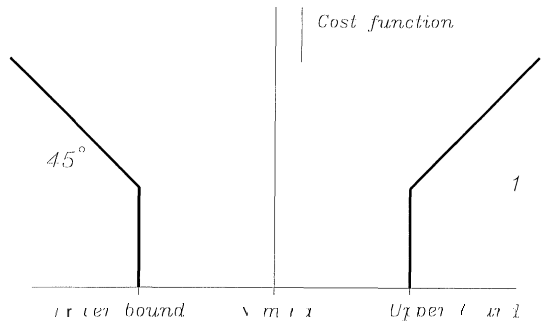


***Figure** 5.15: cost function of a feasibility region*

During identification, the process valves are about half open, such that the me define the following regions *(table* 5.5):

Uncertainty Parameters	Upper bound	Nominal	Lower bound
Leakage	0.7 cm^2	0.4 cm^2	0.1 cm^2
Interconnection	0.4 cm^2	0.25 cm^2	0.1 cnz^2
Pump Gain	8 $cm^3/(sV)$	5 $cm^3/(sV)$	2 $cm^3/(sV)$

***Table** 5.4: uncertainty regions of the parameters*

The obtained *non-linear least square problem* is solved with the Matlab function *leastsq* (optimization toolbox).
The *Levenberg-Marquardt* method, as described in section 3.4, has been used to search for a minimum.
Tlie nominal values *(table 5.3)* are used as initial parameter values (valves half open).
Because the function is quite *general,* we notice that the parameter constraints cannot be implemented as in *(3 18)*, but instead, all constraints are added to *all* prediction errors (all vector elements). So tlie second summation operator in *(3.18)* has been left out.

The estimated parameter values are given in *table* 5.5.

Leakage		Interconnection		Pump Gain	
a_{L1}	$7.000 \cdot 10^{-1}$	a_{I13}	$5.421 \cdot 10^{-1}$	c_1	4.825
a_{L2}	8.42010^{-1}	a_{I32}	$4.523 \cdot 10^{-1}$	c_2	4.771
a_{L3}	$1.488 \cdot 10^{-1}$	a_{I20}	$6.933 \cdot 10^{-1}$		
Error			$4.146 \cdot 10^{-1}$ cm^2		

***Table** 5.5: the physical parameters of the non-linear model, with parameters constraints.*

The meaning of the symbols in *table* 5.5 can be found at the end in appendix *A.*I.

From the Error in *table 5.5* it can been seen that tlie parameters, found with our earlier linear problem formulation *(table 5.3)*, have lost their physical meaning. That is, the (local) minimum can't be found in our *constraint search area.*
Notice that:

- √ there is a small possibility that our earlier found minimum (linear formulation) lies in our defined feasibility region, as we dealing with local minima. However this possibility was made small by starting the search procedure with different initial values;
- √ severai warnings were shown by Matlab due to a ill-conditioned Jacobian. This is caused by a "too flexible" model structure, as was discussed at the end of section 3.3. As long as they doesn't appear to "often", it won't give any problems. If the warnings appear at the end of the search, they will be responsible for a high parameter variance for one or more parameters. That is, we find our self in a minimum plain, rather than in a minimum point;
- √ apart from the estimated pump gains, which apparently have little influence, most of the estimated parameter values are near the upper bounds defined in *table 5.4;*
- √ we tried to find an estimate of the parameter variance, which is a function of the Jacobian. However. because of lack of information about the function *leastsq*, this seemed to be rather difficult.

In *table 5.6* and *tuble 5.7* the results are shown, when we are *not* using our defined *feasibility regions.* That is. the search area doesn't has any restrictions. The Matlab function *leastsq* is used to search for a minimum.

Leakage		Interconnection		Pump Gain	
a_{L1}	1.373	a_{I13}	$1.623 \cdot 10^{-1}$	c_1	5.172
a_{L2}	1.129	a_{I32}	$1.302 \cdot 10^{-1}$	c_2	4.883
a_{L3}	$9.633 \cdot 10^{-1}$	a_{I20}	$9.788 \cdot 10^{-1}$		
Error			$2.898 \cdot 10^{-1}$ cm^2		

***Table** 5.6: the physical parameters of the non-linear model, without parameters constraints.*

Parameter s	Parameter value	Parameter s	Parameter value
φ_{11}	1	φ_{27}	$3.744 \cdot 10^{-2}$
φ_{12}	$-3.949 \cdot 10^{-1}$	φ_{28}	$3.171 \cdot 10^{-1}$
φ_{13}	$-4.667 \cdot 10^{-2}$	φ_{39}	1
φ_{14}	$3.359 \cdot 10^{-1}$	$\varphi_{3,10}$	$-6.062 \cdot 10^{-1}$
φ_{25}	1	$\varphi_{3,11}$	$-3.744 \cdot 10^{-2}$
φ_{26}	$-6.062 \cdot 10^{-1}$	$\varphi_{3,12}$	$4.667 \cdot 10^{-2}$
Error			$2.898 \cdot 10^{-1}$ cm"

***Table** 5.7: parameters (matrix entries) of the estimated non-linear model, without parameter constraints.*

Apparently we are able to find an even smaller Error, when the problem is formulated as an non-linear problem without parameter constraints, than tlie linear problem formulation.
In figure *a.*5.3 and figure *a.5.4* (appendix *A.5)* the cross-correlation between the residuals and inputs are shown, which is a validation method that was explained in section 3.5. Notice that the cross-correlation function is further removed from the x-axis. in case of the parameter constraints than without parameter constraints. This indicates that more system information, present in the data, is included in case of the search without parameter constraints (see also the end of section 3.5). This is an alternative way of verifying that the (local) minimum, found for modelling, doesn't lie in the feasibility region.
The horizontal lines in *figure a.5* 3 and *figure a.5.4* indicate the 99% confidence levels, when the cross-correlation is assumed to be normal distributed (Lju87). These confidence levels are a function of the residual auto-correlation function. Because the residuals of the model with parameter constraints is "less white" (not shown here), the confidence levels are further removed from the x-axes. That the residuals are less white is easily understood when we assume that our noise. acting on the process outputs, is white, like assumed in our chosen model structure. Then the "less whiteness" of the residuals is caused by process information. which couldn't be included in the model structure. such that it is present in the residuals, causing a relation between the residual samples.

5.4.2.2 Linear model estimation

The parameters of the linear model structure *(a* 1 9) - *(a 1.10)* in appendix *A.1* will be estimated, when use is made of the BN and GBN input signals. These input signals were designed in section 5.3.3
The parameters will be estimated with the Matlab functions *mf2th* and *pem*, from the Identification toolbox [Lju91]. Or with the Matlab function *leastsq* from the Optimization toolbox [Gra92].
As validation. we use the last half part of the *GBN* data.

We begin by implementing the model structure *(3.11)* with the Matlab function *mf2th*, such that the parameters can be estimated with the function *pem.*
The matrices $A(\theta)$, $B(\theta)$, $C(\theta)$ in *(3.11)* correspond to the closed-loop matrices $A_{sd}(\theta)$, $B_{sd}(\theta)$, $C_{sd}(\theta)$, which form the discrete counter part of the matrices in *(5.7).* The matrices consist of known parts, due to the known controller and unknown parts due to the *uncertainty* in the *physical* parameters (table *5.1).*
Matrix $D(\theta)$ equals zero and $e(k)$ is a $\Re^{2\times 1}$ vector, which produces an independent white noise sequence on every output. The covariance matrix of $e(k)$ is diagonal, with the same value for every matrix entry. Notice that the value of this matrix entry is of minor importance as they all have the same value, and therefore has the effect of a scaling factor on the final matrix $K(\theta)$.

$K(\theta)$ consists of a $\Re^{5\times 2}$ matrix, that indicates the noise influence on the system states.

After a first run of our identification with the function *pem*, $K(\theta)$ is chosen with zero values for all matrix entries, except for the entries $K_{3\,1}$ and $K_{4.2}$ (to be estimated). as these are the only parameters that give a significant contribution when all matrix entries are chosen as free parameters (in the first run).

The initial values are the nine calculated parameter values (matrix entries), in case the valves are assumed to be half open (table 5.8).

The Matlab function *pem* uses the *Gauss-Newton* method (section 3.4) to estimate the paralneters (to find a local minimum).
We seemed to have an ill-conditioned problem (for both the GBN and BN data-sets), as Matlab let us know that we only have a rank ten, while we are estimating eleven parameters: nine of the process and two of the Kalman gain. This problem was discussed in section 3.3. The problem occurs when we are estimating too many parameters to capture the process information, such that some parameters have a gradient of almost zeros, or when there exist parameters that are a function of the others.
Our nine process parameters, to be estimated, are formed by eight uncertain parameters (a.*1.16*), such that there are parameters which are related. The parameters we are estimating are given in *(a* 1.15). It can easily be seen that we have the following relations:

$$\begin{aligned} \psi_{11} &= f_{11}(\psi_{13}, \theta_{12}) \\ \psi_{22} &= f_{22}(\psi_{23}, \theta_{22}) \\ \psi_{33} &= f_{33}(\psi_{31}, \psi_{32}, \theta_{32}) \end{aligned} \qquad (5.15)$$

Frosn *(5.15)* it follows that the estiinated parameters are strongly related, resulting in an ill-conditioning of the Jacobian. Possible ways to overcoine this problems are:
- fixing one or more parameters;
- estimating the eight parameters that are caring the uncertainties, as done with the non-linear model:
- performing regularization.

The above methods will be applied in the above order:

Fixing parameters: when we are fixing any parameter, the Jacobian seem to be well-conditioned. However we derived in (5.15, that more parameters are related, such that fixing only one, wouldn't solve the problem. Apparently these relations are hardly present in our collected data. which can occur when the parameters are only "weakly" related.
We seem to obtain best results when fixing the parameters ψ_{13} and ψ_{23}. They are fixed to their initial values.

The *estimated* parameter values are given in table 5.19 and in *figure* a.5 6 and *figure* a 5.7 (appendix *A.5)* the validation by simulation, and the residuals analysis are shown. Only the cross-correlation between the inputs and the residuals are used for residual analysis, as we are more interested if "all" process knowledge is/can be included.

The *calculated* parameter values are given in *table 5.10* and in *figure a.5.5* tlie validation by simulation is shown. In this case the valves are assumed to be half open.

Parameters	Parameter value	Parameters	Parameter value
ψ_{11}	$9.641 \cdot 10^{-1}$	ψ_{24}	$3.247 \cdot 10^{-1}$
ψ_{13}	$2.542 \cdot 10^{-2}$	ψ_{31}	$2.542 \cdot 10^{-2}$
ψ_{14}	$3.247 \cdot 10^{-1}$	ψ_{32}	$2.542 \cdot 10^{-2}$
ψ_{21}	$9.575 \cdot 10^{-1}$	ψ_{33}	$9.383 \cdot 10^{-1}$
ψ_{23}	$2.542 \cdot 10^{-2}$		
Error		$9.467 \cdot 10^{-1}$ cm'	

Table 5.8: *the calculated parameters of the linear model.*

BN parameters				GBN parameters			
ψ_{11}	$8.955 \cdot 10^{-1}$	ψ_{24}	$2.200 \cdot 10^{-1}$	ψ_{11}	$8.506 \cdot 10^{-1}$	ψ_{24}	$2.682 \cdot 10^{-1}$
ψ_{13}	$2.542 \cdot 10^{-2}$	ψ_{31}	3.831		$2.542 \cdot 10^{-2}$	ψ_{31}	4.081
ψ_{14}	$2.144 \cdot 10^{-1}$	ψ_{32}	5.441	ψ_{14}	$2.840 \cdot 10^{-1}$	ψ_{32}	3.937
ψ_{21}	$8.690 \cdot 10^{-1}$	ψ_{33}	$-7.282 \cdot 10^{-1}$	ψ_{21}	$8.411 \cdot 10^{-1}$	ψ_{33}	$3.472 \cdot 10^{-1}$
ψ_{23}	$2.542 \cdot 10^{-2}$			ψ_{23}	$2.542 \cdot 10^{-2}$		
Error_{gbn}		$5.922 \cdot 10^{-2}$ cm^2		Error_{gbn}		$3.906 \cdot 10^{-2}$ cm^2	
Error_{bn}		$1.319 \cdot 10^{-2}$ cm^2		Error_{bn}		$1.780 \cdot 10^{-2}$ cm^2	

Table 5.10: *the estimated parameters of the linear model with 2 parameters fixed.*

Notice tliat:

- also tlie *BN* validation data is used for simulation. indicated by Error_{bn}. (without index. the *GBN* validation data is used). From the Error difference it follows tliat with the *GBN* input test signal, the model has most power concentrated in the lower frequencies. That is, the Error with the GBN parameters, is "significant" smaller than that with the B S parameters, using the GBN validation data. Note that with the *GBN* validation data, tlie lower frequency components are dominating;
- the 99 % confidence levels of the *GBN* residuals are further removed from tlie x-axes than those of the *BN* residuals. The confidence levels are a function of the residual auto-correlation. as was already mentioned with the estimation of the non-linear model. The residuals of tlie *BN* data are caring more high frequency components. improving this white noise characteristics, corresponding to a smaller value for the confidence level. Because the *BN* residuals are caring more high frequency components, his cross-correlation is more scattered than that of tlie *GBN* residuals:
- we find a large value fo tlie *BN* cross-correlation of r_1 *to* h_1 and r_2 *to* h_2 for small positive lags of τ *(figure a.5.6)*. It indicates tliat the estimated model isn't able to include the

influence of $u(k-\alpha)$ sufficient, for small values of a. This can be caused by an overestimated time delay, or by high frequency modes, which can't be included in our model. The cross-correlation of the *GBN* residuals doesn't show the large cross-correlation peaks, and the *BN* input test-data shows more high frequency components. So most probably, the real process shows high frequency modes, which can't be included with our derived model structure. It is obvious that the *GBN* input test-data isn't activating these high frequency modes, as it was designed with our derived model structure.

The filter used for pre-processing introduces a phase shift, which is almost linear in the frequency region of interest, resulting in an almost pure time delay. With the *BN* data this time delay is *only* introduced in the output signal, as the input isn't filtered. Thus, me assumed a smaller time delay as is present in the final *BN* data. However, this is exactly the opposite operation than the one, causing our cross-correlation peak. and therefore can't be the cause. We verified above. by filtering also the input data with the same filter as the output data and the peaks, indeed still appeared;

- we put more weight on the lower frequency area, when we also filter the *BN* input data [Lju87]. We then obtain smaller Errors for both validation sets ($Error_{gbn}$ and $Error_{bn}$), which are almost the same as the Errors measured with the *GBN* data.

estimating the eight parameters that are caring the uncertainties: the nine matrix entries of our model structure, which we are estimating, are a function of eight independent parameters, that are caring the uncertainties. To prevent an ill-conditioned problem, due to the relations between the matrix entries *(5.15)*, we directly estimate these eight independent parameters.

We performed the estimation with the Matlab function *pem*, and still came up with an ill-conditioned problem. This is due to the little influence that some paraineters have on the model output, causing matrix columns of the Jacobian to become almost zero. This was easily checked by fixing some parameters. When fixing a parameter, causing the ill-conditioned problem, we can do with seven parameters to be estimated. However, when leaving this parameter as to be estimated, it can be that we first have to fix three or more parameters. before its influence becomes significant.

performing regularization: a method to get grip on an ill-conditioned problem, as was explained in section 3.3.

We use only the *GBN* data, as we noticed that the difference with the *BN* results were rather small, but still a bit better (the Error). Even so, with the *GBN* input signal, more weight is put on the more important (in control) low frequency area of our model.

Equation (3.20) is implemented using the Matlab function *leastsq*. We started with a "large" regularization value μ to prevent the Matlab warnings, that appear with our previous problem formulation. Next, μ is decreased, such that no warnings appear and that the Error decreases. In this way we try to find a balance between the increase in bias and the decrease in variance. We found a $\mu = 5.10^{-5}$, which is the same for *all* parameters.

The estimated parameter values are given in *table* 5.10 and in *figure* *a.5.8* tlie validation by simulation. and the residuals analysis are shown.

GBN parameters					
Leakage		Interconnection		Pump Gain	
a_{L1}	1.109	a_{I13}	1.070	c_1	4.272
a_{L2}	$6.150 \cdot 10^{-1}$	a_{I32}	1.663	c_2	4.469
a_{L7}	2.091	a_{I20}	$4.5 \cdot 10^{-1}$		
Error				$2.800 \cdot 10^{-1}$ cm^2	

Table 5.10: *the physical parameters (uncertainties) of the linear model, with regularization.*

Notice that:

√ from *table* 5.10 it follows tliat some of our parameters lost their physical meaning (let us remind us that when the valves are entirely opened: $a_{\mathrm{Ii}} = 0.4$ cm^2 and $a_{\mathrm{Li}} = 0.8$ cm^2). We can easily increase the regularization parameter μ to retain our physical meaning. However this will result in a bigger Error:

√ we found a smaller error than with fixing the parameters;

√ Like discussed in section 3.3. the *efficient* parameters are formed by a_{L1}, a_{L3}, a_{I13}, a_{I32}. and the *spurious* parameters by a_{L2} a_{I20}, c_1 and c_2. That is, the spurious parameters. on tlie contrary to the efficient ones. have ***hardly*** any influence on the model output, such tliat they could be fixed. So regularization can be used to examine which parameters can be fixed;

√ the estimated parameter values of the non-linear model, without the parameter constraints. are quite different tlian those of the linear model. with regularization (compare *table* *5.6* and *table* 5.10). This isn't that surprising when we remind ourselves tliat with the non-linear model, we were dealing with a (local) minimum plain rather than a minimum point.

During our experiment design, we extract some system information from the derived white model (appendix *A* 1). For the choice of the sample frequencies we made use of the system bandwidth and for the total data length we used tlie biggest time constant. A wrong estimation of those values won't influence the final results that much. Still. with the final estimated model (with regularization) we calculate: $\tau_{big} \approx 108$ s and $\omega_{-3dB} \approx 1.2 \cdot 10^{-2}$ rad/s. The values original used: $\tau_{big} \approx 229$ s and $\omega_{-3dB} \approx 2 \cdot 10^{-2}$ rad/s. The time constant seems to be overestimated. which means that we have collected more data than necessary and thus should improve the estimation results.

Notice that for the design of the *GBN* signal we used the 99%-settling time, which we obtained directly from the real process.

5.5 Conclusion and Final Remarks

We developed some software to perform our identification and to implement our controllers. The real-time function are implemented in software, which asked for some special care. as the use of buffers, such that no data is lost and an accurate order of the source code.
The software was proven to be flexible and therefore already in use by other students to improve the software (buffer problem and change of language), perform experiments and implement other controllers.

We estimated several linear and non-linear models, based on a model structure, derived from a mathematical process model (appendix *A.1*). Such a model structure is also known as a physically parameterized model structure, as was explained in section 3.2.

The estimated non-linear model can't be used for our controller design, but surely gives us more confidence in the chosen model structure. We found two non-linear models of which one retained his physical meaning in the parameters *(table 5.5)*. The physical meaning of the parameters is retained by defining parameter constraints. However we showed that a lower value of the cost function, which corresponds to the model output error, can be found *outside* the constraint search area *(table 5.6)*.
Apparently there is more dynamics present in the data than can be captured by the derived model structure. retaining his physical meaning. Possible causes are, that:

1. we haven't captured all "possible" *process* dynamics, with the white model, derived in appendix *A.1*;
2. we can't guarantee that we haven't got a bias, under certain weak conditions, as was mentioned in section 5.4. This means that it is possible that we don't converge to our true process model, *even* if it lies in our chosen model set. While it is possible that this true process model retains his physical meaning:
3. a combination of 1 and 2.

A way of un-correlating the process inputs, used as identification inputs, is by using a two stage identification strategy [Hof93]. In this way, first a model between the setpoints and the process inputs is estimated, which is used to reconstruct the *un-correlated* process inputs. These are then used to estimate our process model. Because the identification inputs are uncorrelated with the noise, acting on the process outputs, we can guarantee a zero bias. under certain weak condition (section 5.4).
However the model used to reconstruct our process inputs would be a linear one, which could cause loss of information, as we want to estimate a non-linear process model.

We found a linear model, which is able to re-produce the validation data with a small error. Also the residual analysis seemed to be acceptable, which means that almost all process information, available in the data, was captured in the estimated model.
The small error indicates that we can "easily" simulate our non-linear proces with a linear model, in surroundings of the working point in which we performed our identification. Let us remind us that the process non-linearity highly depends on the positions of the valves, simulating the leakages.

Just like the estimated non-linear model, tlie linear model did not retain his physical meaning, due to the approximation of a non-linear process with a linear model and/or uncaptured process dynamics in our derived process model.

We showed that our chosen linear model structure seemed to be "too" flexible, which leads to an ill-conditioning of the Jacobian. We used two techniques to overcome this problem, namely regularization and fixing parameters.
Regularization seems to be an easy and flexible way to overcoine ill-condtioned problems and gives us more insight in the influence of the estimated parameters.
Fixing parameters, also gives good results, but isn't able to give us insight in the parameter influence. like regularization. Also it can be difficult to obtain correct values for the fixed parameters, without some process knowledge. Regularization can even help us finding better values for the fixed parameters.

The improvement we obtained, by using the *GBN* input signal, instead of tlie *BN* input signal, is hardly visible as we already obtained an accurate process model with the *BN* data. The *GBN* input signal puts more weight on the lower frequency area, as was discussed in section 3.6. We showed that we can also put more weight on the lower frequency components by filtering the *BN* input signal, which results in smaller Errors for the validation data. The Errors. then appear to be almost the same as the ones obtained with the *GBN* data.

In the introduction (chapter 1), we motivated the use of a physically paraineterized model structure. We almost, only looked at his benefits. Our major motivation was that real engineering applications are never that "black". We were able to find a "good" model, but let us not forget the extra work (problems) we went through. Basically this was due to the time put in deriving the model structure and the need of re-tuning (tricks) the model structure and his unknown parameters (e.g.: regularization), due to the chosen model structure.

6

Control of the Process

We will develop two controllers which make use of the process model. estimated in the previous chapter through regularization. In the first section. a LQR controller will be developed, which should perform better than the LQR developed in chapter 5. as it makes use of a more accurate process model.

Next, a Robust controller will be developed. as the process valves *(fig.* 2.1) can easily be changed, causing parameter perturbation.
Because the parameter uncertainties are well defined a robust controller is developed. that can deal with these uncertainties in an explicit manner.
The H_∞ theory is used to develop our controller; such that we can guarantee pre-defined controller objectives, regardless of the parameter uncertainties.

Theoretical background of both controllers was given in chapter 4 and references therein

6.1 LQR Development

We will develop our *LQR* controller in the same way as we did in section 5 2
The only difference is that we use the process model, estimated in the previous chapter to calculate our controller We use the parameters. that were estimated with a regularization technique and are summarised in *table 5* 10
Because we use another process model. we have to re-tune our weighting matrices
Using the function *lqrd*, we obtain

$$R = 10^3 \begin{bmatrix} 17 & 0 \\ 0 & 14 \end{bmatrix}; \quad Q = \begin{bmatrix} 1.5 & 0 & 0 & 0 & 0 \\ 0 & 1.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}; \tag{6.1}$$

$$K = -\begin{bmatrix} 9.790 \cdot 10^{-3} & -1.217 \cdot 10^{-4} & 5.215 \cdot 10^{-1} & 7.088 \cdot 10^{-2} & 1.541 \cdot 10^{-1} \\ 1.294 \cdot 10^{-4} & 1.033 \cdot 10^{-2} & 9.002 \cdot 10^{-2} & 5.129 \cdot 10^{-1} & 2.185 \cdot 10^{-1} \end{bmatrix}$$

See section 5.2 for more detailed information about the development conditions

To compare both *LQR* controllers. based on different process models (a calculated and estimated one). we compare their step responses The step responses are given in *figure* h 2 **1** in appendix B 2

Our new *LQR* controller shows the following differences with the old controller
√ has little overshoot (**'g** h 2 *1 a).*
√ is faster (*fig* h 2 *1 a*);
√ an outer column has less influence of a setpoint change in tlie other outer column. when his setpoint is kept constant (*fig b* 2 *1* h and jig h 2 **I** c).
√ the transfer from input 1 to output 2 and from input 2 to output 1 (cross-transfers) show non-minimum phase behaviour (*fig* h 2 **1** *h)*

We can say that we find a new *LQR* controller that performs "better". which isn't that surprising as we used a more accurate process model

6.2 Robust Controller Development

The development of our robust controller basically consist of three steps:
- problem formulation This consist of the specification of the:
  - √ model uncertainty;
  - √ performance objectives.
- controller calculation:
- closed-loop analysis.

In the next section we will begin by formulating our problem. and include our parameter uncertainties. Next we will calculate our controller and analysis the closed-loop results

6.2.1 Problem Formulation

We begin by specifying our problem. The uncertain parameters are directly related to the interconnections between the three tanks (see *fig. 2.1*), the tank leakage's and the pump transfers (see *a.1.16*). As a consequence of their uncertainties, there also appears an uncertainty in the working point value of tank 3 (see *Jig. 2.1*), which appear in *(u.* **1.***15).* So the *structured uncertainties* are formed *by:*

$$a_{L1},\ a_{L2},\ a_{L3},\ a_{I13},\ a_{I23},\ a_{I20},\ c_1,\ c_2,\ h_{13},\ h_{23}, \tag{6.2}$$

with

$$h_{13} = \frac{1}{\sqrt{\overset{\circ}{h}_1 - \overset{\circ}{h}_3}} \quad \text{and} \quad h_{23} = \frac{1}{\sqrt{\overset{\circ}{h}_2 - \overset{\circ}{h}_3}} \tag{6.3}$$

Notice that we have included the uncertainty of $\overset{\circ}{h}_3$ (level of tank 3 in the working point) as an *independent* uncertainty which is incorrect as it is a function of the other uncertainties However. as this relationship is not exactly known and not included in our process model, we assume it as an independent uncertainty
As the working point of tank 1 equals that of tank 2, me get that h_{13} equals h_{23}, such that h_{23} can be removed from (6 2)

Every uncertain parameter represents a *set* of parameter values. which we represent by

$$\begin{aligned}
a_{l1} &= \overset{\circ}{a}_{l1}(1+\delta_{aL1}\Delta_{aL1}) \\
a_{l2} &= \overset{\circ}{a}_{l2}(1+\delta_{aL2}\Delta_{aL2}) \\
a_{l3} &= \overset{\circ}{a}_{l3}(1+\delta_{aL3}\Delta_{aL3}) \\
a_{I13} &= \overset{\circ}{a}_{I13}(1+\delta_{aI13}\Delta_{aI13}) \\
a_{I23} &= \overset{\circ}{a}_{I23}(1+\delta_{aI23}\Delta_{aI23}) \\
a_{I20} &= \overset{\circ}{a}_{I20}(1+\delta_{aI20}\Delta_{aI20}) \\
c_1 &= \overset{\circ}{c}_1(1+\delta_{c1}\Delta_{c1}) \\
c_2 &= \overset{\circ}{c}_2(1+\delta_{c2}\Delta_{c2}) \\
h_{13} &= \overset{\circ}{h}_{13}(1+\delta_{h13}\Delta_{h13})
\end{aligned} \tag{6.4}$$

with

$$\Delta_{aL1},\ \mathrm{A} \quad .\ ,\ \Delta_{h13} \in B\Delta \quad \text{and} \quad B\Delta = [-1,+1] \tag{6.5}$$

Above means that, for example c_1 has a uniform distribution with an average value $\overset{\circ}{c}_1$ and an interval length of $\delta_{c1}\cdot\Delta_{c1}\cdot\overset{\circ}{c}_1$

In appendix B *3.* the graphical state-space representation is given of the process *(u* **1** *6)* with the above uncertain parameter

In appendix *B* 3 we use the following macros

$$h_{13} = \frac{1}{\sqrt{h_1^\circ - h_3^\circ}}; \quad h_{10} = \frac{1}{\sqrt{h_1^\circ}}; \quad h_{20} = \frac{1}{\sqrt{h_2^\circ}}; \quad h_{30} = \frac{1}{\sqrt{h_3^\circ}}$$
$$g = \frac{\sqrt{2g}}{2A}; \quad b = \frac{1}{A} \qquad (6.6)$$

We have to *pull* out the uncertainties A (as done in *figure 4* S) to determine the final $N\Delta K$ presentation, from which the controller can be calculated To perform this task, we introduce the following state-space uncertainty representation with (D, =*0*)

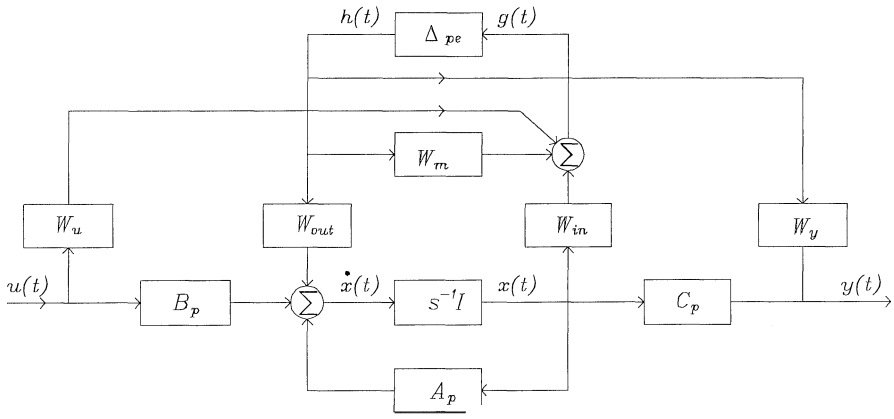


***Figure** 6.1: Uncertainty state-space presentation*

Assume that we have a system with m inputs and p process outputs, s states and q uncertainties

Then $h, g \in \Re^{q\times 1}$, $x \in \Re^{s\ 1}$, $u \in \Re^{m\times 1}$, $y \in \Re^{p\times 1}$, such that $W_u \in \Re^{q\times m}$, $W_{out} \in \Re^{s\times q}$, $W_{in} \in \Re^{q\times s}$, $W_m \in \Re^{q\times q}$, $W_1 \in \Re^{p\times q}$ and $\Delta_{pe} \in \Re^{q\times q}$

With

A_p, B_p, C_p, the process state-space matrices ($D_{pp} = 0$), which correspond to *(a* I *7)*;

W_u, W_{out} W_{in} W_{pp} used to incorporate the state-space parameter uncertainties A. which are incorporated in the diagonal matrix A,

Notice that g is the *fotal* number of uncertainties that exists of *independent* and *dependent* (repeated) uncertainties

It is easy to see that W_{pp} and W_u are used to handle uncertainties that appear in the process matrices B_{pp} and C, respectively So W_1 equals zero, as the uncertain parameters are located in A_{pp} and B_p

The presentation of *figure 6 1* can easily be implemented in Matlab, by making use of the Matlab command *sysic* This is an interconnection program, to connect different matrices, which can be a constant or a system matrix For more information about this program, see the μ-Analysis and Synthesis Toolbox manual [Bal93]

For our graphical state-space model in appendix B.3 we define:

$$h = \begin{bmatrix} i_{L1} & i_{L2} & i_{L3} & i_{I13} & i^2_{I23} & i^3_{I23} & i_{I20} & i^1_{13} & i^2_{13} & i^3_{13} & i_1 & i_2 \end{bmatrix}^T$$
$$g = \begin{bmatrix} o_{L1} & o_{L2} & o_{L3} & o_{I13} & o^2_{I23} & o^3_{I23} & o_{I20} & o^1_{13} & o^2_{13} & o^3_{13} & o_1 & o_2 \end{bmatrix}^T \quad (6.7)$$

Notice that the vectors in (6 7) represent 12 uncertainties. while we only have 9 uncertain parameters The difference is due to the repeated uncertainties (see *figure* h 3 *1 -h* 3 *3*), which are indicated by a SLIP-index

The matrix entries of W_u, W_{out}, $W_{,,,}$, W_m aid Δ_{pe} can easily be determined with the help of the state space presentation in appendix B.3 and are summarised in the same appendix.

The next step in formulating our problem is the *performance specification.*
As we want a good tracking behaviour with a zero steady state error; we minimise a sensitivity function, which is the transfer function between the error e and the reference v: see *figure 6.2.* To overcome actuator saturation, we minimise the control sensitivity function. which is the transfer between the input u and the disturbance d (see *fig.* 6.2). The frequency areas of interest, over which we minimise, are specified with the filters V, and V„:

$$V_{e,u}(s) = v_{e,u}(s) \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (6.8)$$

For $v_{e,u}(s)$ we choose a simple first order filter, as the final controller order is proportional to the order of the generalized plant:

$$v_{e,u}(s) = K \frac{\tau_1 s + 1}{\tau_2 s + 1} \quad (6.9)$$

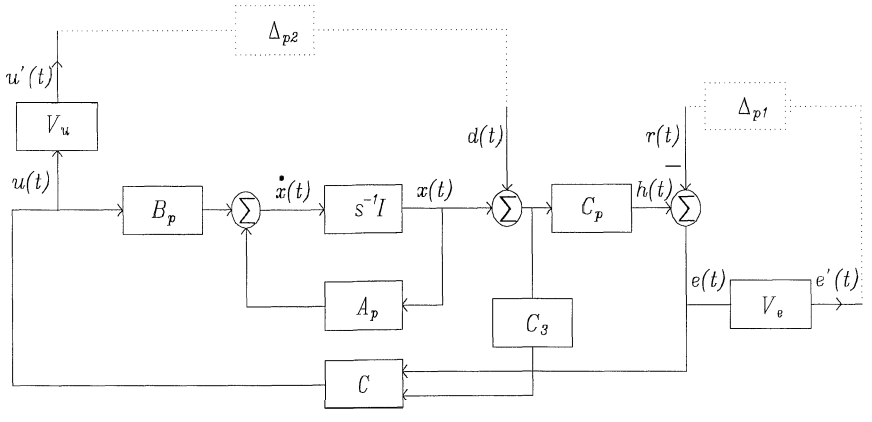


Figure 6.2: *Performance specification.*

The generalised plant is the open loop configuration as in *figure* 6 1 with all filters. for performance and uncertainty specification included
Having defined the parameter uncertainties and the performance objectives. we construct the following NAK structure

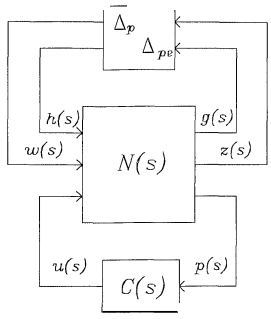


Figure 6.3: The final NAK *structure.*

In which:

$$
\begin{aligned}
z &= \begin{bmatrix} e_1' & e_2' & u_2' & u_2' \end{bmatrix}^T \\
w &= \begin{bmatrix} r_1 & r_2 & d_1 & d_2 & d, \end{bmatrix} \\
p &= \begin{bmatrix} e_1 & e_2 & h_3 \end{bmatrix}^T \\
\mathrm{A} &= \mathrm{diag}\{\Delta_p, \mathrm{A}_{pe}\};\ \mathrm{A}_{,,} = \mathrm{diag}\{\Delta_{p1}, \mathrm{A}_{,,}\ \}
\end{aligned}
\qquad (6.10)
$$

Where e,. u_i, r_i, d_i and h, are components of the corresponding vectors, which can be find in *figure* 6 2
A diagonal matrix is indicated by $diag\{\cdot\}$
The matrix C, and C_3 are used to couple out h_1, h_2 and h_3 respectively
The diagonal uncertainty matrix A $\Delta_{pe} \in \Re^{12\times12}$ (appendix B 3) and the performance blocks $\Delta_{p1} \in \mathbb{C}^{2\times2}$; $\Delta_{p2} \in \mathbb{C}^{3\times2}$, with $\|\Delta\|_\infty \leq 1$

We noticed that we want a good tracking behaviour, with a zero steady state error This can be accomplished by including an integrator in the controller One way to fulfill this task is to include an integrator in the filter V,. In this way the controller must have a pole at $s = 0$ to make the ∞-norm finite The problem with this approach is that the pole $s = 0$ of V_e becomes an uncontrollable pole of the generalised plant, which violates the assumption that all states have to be detectable and stabilizable This is one of the assumptions made, to be able to calculate an internally stabilising controller [Dam96b]

A possible way out. is to approximate the integrator by a first order process with a "small" pole This will correspond to a small pole in the controller. which can be approximated by an integrator when implementing the controller
For this reason we choose $1/\tau_{e2}$ in *(6 9)* sufficiently small with $\tau_{e2} > 0$. to approximate the integrator action in V_e

6.2.2 Controller Calculation and Validation

A controller is calculated by minimising the "gains" of tlie transfer functions between *h(s)* to *g(s)* and *w(s)* to *z(s)* *(fig 6 3)*, such that the influence of the parameter uncertainties and disturbances is minimised
The "gain" is measured as the 2-norm ratio of the outputs *g(s)* and *z(s)* to the inputs *h(s)* and *z(s)* respective1y The gain is minimised by minimising the ∞-norm of the generalized plant

We find a "reasonable" H,-coiltroller with the Matlab function *hinfsyn* Our process model, is the estimated model in the previous chapter. which was estimated through regularization (*table 5 10*) Furthermore we chose for all parameters a maximum parameter uncertainty of 20% (in *(6 4)* 1/15,= *20*)

By choosing a smaller value for $1/\tau_{e2}$ we are moving $|v_e(j\omega)|$ to the left.
It seems that. by moving $|v_e(j\omega)|$ to the left, the closed-loop system becomes rather slow. referring to his step response. This can be explained in the following way. When n-c are moving $|v_e(j\omega)|$ to the left, we are also moving in the same direction the allowed frequency area in which the sensitivity function *S(s)* may appear, (see section 4.2 about the use of the weighting filters). Due to $S(s) + T(s) = 1$, it is easy to see that the complementary sensitivity function *T(s)* will move to the left as well, causing a slower step response. Notice that by moving *S(s)* to the left, his,*form* won't change that much, and due to $S(s) + T(s) = 1$ *neither* does *T(s)*.
Another "simple" way of examining tlie relation between *T(s)* and *S(s)* is by presenting them as vectors in the complex plane, related by $S(s) + T(s) = 1$.

As was mentioned in section 4.2. the ∞-norm of the generalized plant is minimised by assuming a *full complex* uncertainty matrix $A \in \mathbb{C}$ This can lead to a conservative indication about the robust stability and robust performance, when we are dealing with structured uncertainties as was explained in section 4 2 Therefore, we also calculated the structured singular value μ_Δ, which is shown in *figure* 6 *4*. together with the largest singular value The μ_Δ value is calculated for the *mrxed* uncertainty matrix A in (6 *10)*, with the p-Analysis an Synthesis Toolbox

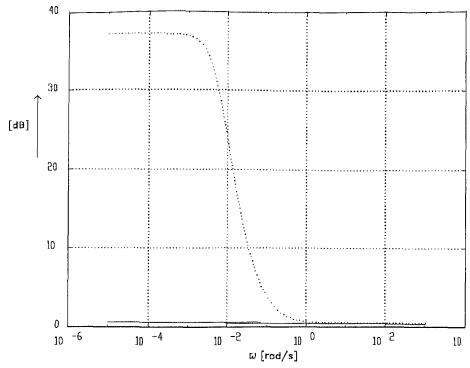


Figure 6.4: h e lnvgest singular value (dotted) and the p-bounds (solid), with 9 independent uncertain parameters

From *figure* 6.*4* we can conclude that we introduced quite some conservatism by assuming a full complex uncertainty matrix to calculate the H,-controller.
Less conservative results can be obtained by:

1. reducing the number of uncertain parameters, which results in a smaller uncertainty matrix with relatively less zero matrix entries (whenever that is possible);
2. using a μ-synthesis technique, which is able to deal with structured uncertainties, as was explained in section 4.2.4;
3. transforming the *structured* uncertainty matrix *A,* in (6.10) to a *unstructured* uncertainty matrix by determining the process uncertainty as a function of the frequency.

We will calculate a robust controller by reducing the number of uncertain parameters, which only asks for some small changes in our original problem formulation Furthermore can we use our identification results in the previous chapter to reduce our set of uncertain parameter
The μ-synthesis techniques have as disadvantage that. they lead to high order controllers In section 4 2 4 we briefly outlined that the μ-upper and lower bounds are tightened by correct matrix transformations (filters) However these filters cause the order of the generalized plant to increase and therefor as well the order of the controller
Before we calculate our controller. by reducing the number of uncertain parameters. we like to outline point 3

Another interesting n a y to incorporate process uncertainties and to come up with a controller of relatively lon order. is by defining the process uncertainty in the frequency domain We like to give an small outline of how this m orks
Assume that we like to represent our process uncertainty as an output multiplicative uncertainty as was shown in *figure 3 7*:

$$P'(s) = (I + \Delta(s)W(s))P(s), \qquad (6.11)$$

where P ' is a set of possible process models. due to the perturbation A $A \in C^{p\,p}$ with $\|\Delta\|_\infty \leq 1$ and p the number of process outputs P presents the nominal process and W is the well known weighting filter

Rewriting *(6.*11**)** yields:

$$(\mathrm{P}'(\mathrm{s}) - P(s))P^{-1}(\mathrm{s}) = \Delta(s)W(s)\,, \tag{6.12}$$

where the left side presents the relative process uncertainty.
We are only considering robust stability. With $\|\mathrm{A}\|_\infty < 1$. it is then sufficiently that the folloning holds. to guarantee robust stability (4.33):

$$\left\|(P'(s) - P(s))P^{-1}(s)\right\|_\infty \le \|W(s)\|_\infty \tag{6.13}$$

We then choose a filter W of the fonn (6 S), with w equal to the left side of (6 *13)* The left side of (6 *13)* can be calculated by varying the uncertain parameters We define the uncertainty interval of every uncertain parameter and calculate the relative process uncertainty for every pre-defined discrete parameter value in the uncertainty interval For example, if we have 8 uncertain parameters and we discretisize our defined uncertainty interval with 10 points, me will perform a total of 10^8 process uncertainty calculations
During calculating, we can easily register the maximum relative process uncertainty. that is a function of the frequency With the Matlab function *fitmag* (there are more that perform the same. see [Bal93]) we fit a rational stable transfer function to the determined upper bound. that consists of amplitude points as a function of the frequency This stable transfer function serves as our final filter W
Apart from a long calculation time. needed to determine our upper-bound. we determine a relatively tight bound The number of calculations can be reduced by choosing for example, a normal distribution for the discretized uncertainty interval
The square uncertainty matrix A has "only" the dimension of the number of outputs, which yields to less conservatism. when calculating the H,-coiltroller of the final uncertainty matrix (augmented with the performance blocks)
We could label the above procedure as a way of transforming a large-real uncertainty matrix (structured uncertainty) to a smaller-complex uncertainty matrix (unstructured uncertainty)

To reduce the number of parameters, to decrease the uncertainty matrix dimensions. we make use of the identification results in the previous chapter In *table 5 10* we found an accurate process model through a regularization technique, mere the estimated parameters were divided in so called *efficient* and *spurious* ones The spurious parameters. on the contrary to the efficient ones, hardly have any influence on the model output, when dealing with parameter perturbation We therefore fix the spurious parameters and only choose the efficient ones. as the uncertain parameters
The efficient parameters are a_{I1}, a_{I3}, a_{I13}, a_{I32} leading to a smaller diagonal process uncertainty matrix $\mathrm{A}, \in \Re^{5\times 5}$
Our new uncertainty vectors *(6 7)* become

$$\begin{aligned} h &= \begin{bmatrix} i_{L1} & i_{L3} & i_{I13} & i_{I23}^2 & i_{I23}^3 \end{bmatrix}^T \\ g &= \begin{bmatrix} o_{L1} & o_{L3} & o_{I13} & o_{I23}^2 & o_{I23}^3 \end{bmatrix}^T \end{aligned}, \tag{6.14}$$

and the new formed matrices W, and Δ_{pe} in *figure* 6.1 are summarised in appendix ***B.3***. Notice that our performance specification slays the same.

The largest singular value and the p-bounds of the final closed-loop system arc shown in *figure 6.5,* when the H_∞-controller is calculated with the Matlab function *hinfsyn*, under the same conditions as before. That is, a chosen parameter uncertainty of 20% for all parameters.

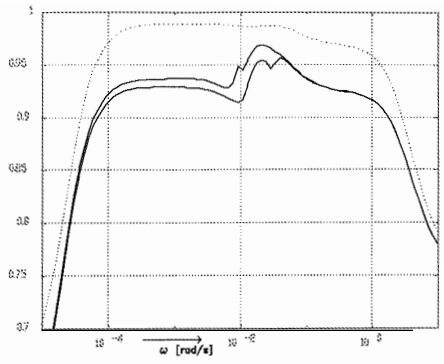


Figure 6.5: the largest *singular value (dotted)* and the μ-*bounds (solid), with* 4 *(independent) uncertain parameters.*

The p-bounds are calculated for the *mixed* uncertainty matrix A in (6.10) and the difference between the ∞-norm and the p-upper bound of our generalised system, certainly reduced considerable.
It seems that we find a robust controller that has *robust performance*, as defined in section 4.2. As explained in section 4.3, we can investigate the *robust stability* and the *nominal performance* separately, which is done in *figure* 6.6.

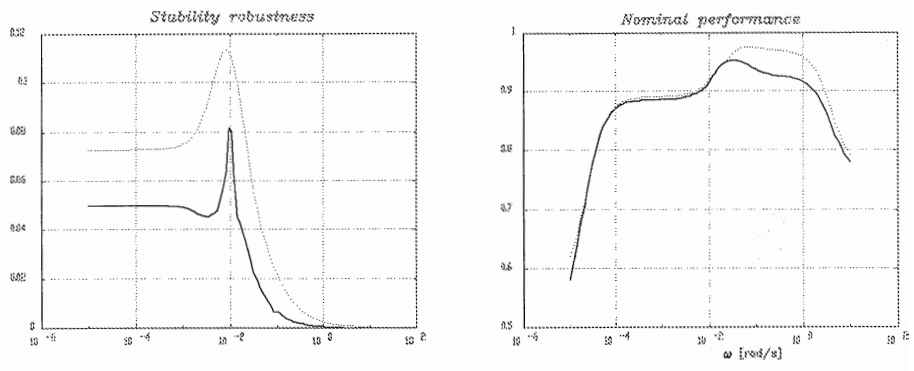


Figure 6.6: *the largest singular value* (dotted) *and* p *bounds (solid)* for- *the stability robustness and the nominal performance.*

In *figure 6.6* only the p-upper bound for the stability robustness is shown, as the p-lower bound is zero for some frequencies, which results in numerical problems.

The filter parameters of V, and V„ *(figure 6.2* and *(6.9))* are summarised in *table 6.1* and their frequency response is shown in *figure* 6.7.

filters:	τ_1	τ_2	K
V_e	$1/1.265 \cdot 10^{-2}$	$1/2 \cdot 10^{-5}$	200
V_u	$1/2 \cdot 10^{-2}$	114.428	0.7

Table 6.1: *the parameter values, of the performance filters.*

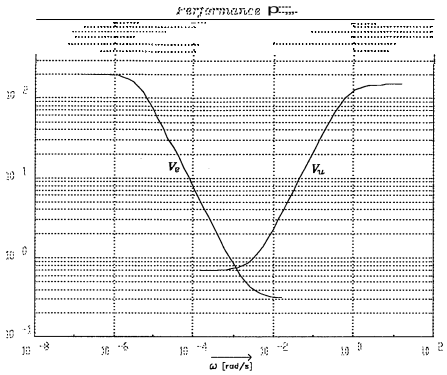


Figure 6.7: *the frequency response of the performance filters.*

We found a H,-controller of order seven, which we reduced to order five, without hardly any increase in the m-norm: The final fifth order continue controller is summarised in appendix *B.4*

In *figure b.4.1* (appendix *B.4)* a step response with the real process is shown for input 1, for both the LQR controller, developed in the previous section and the robust controller.
If we compare both controllers, it is difficult to speak of a "better controller" because of two reasons:

√ a "better controller" is decided, based on the control objectives, which we hardly have, which gave us a lot of freedom;

√ the maximum value of the actuator output is 4 Volt and 7 Volt with respectively the robust and LQR controller, with a step of 0 –> 30 cm. That means that the performance filters could be adjusted such that better use is made of the total actuator range (-10 – 10 Volt) . This will certainly results in a faster closed-loop system.

Notice that:

– we approximate the two small poles $1/\tau_{e2}$ by zero values, which did not seem to give any virtual difference in the step response. The final discrete controller is calculated with the tustin transformation;

- the step response of *figure* b.4.1 differs from tliat in *figure h.*2.1, while we use the same LQR controller, but under different process conditions;
- At first, the filter parameters were chosen such that we were sure to obtain stability robustness. Next, through fine-tuning the filter parameters, actuator saturation was avoided and the perfonnance specifications were met as good as possible.
- The first order filter V, (see *(6.8))* is used to approximate integration behaviour in the controller, by choosing a sufficient small value $1/\tau_{e2}$. However; this results in a small bandwidth for V_e, and thus of the upper bound of tlie sensitivity function *S(s)*. This can be unacceptable when for example, the disturbances are also present at "higher" frequencies.A more accurate V_e can be developed by using a second order filter: giving us more degree of freedoms, which can be used to improve the frequency bandwidth.

We non come to the final part of our analysis were we like to compare the performance deterioration of the closed-loop. due to parameter perturbations, for the different controllers developed in this essay

In *figure* b *4* 2 – h 4 4 (appendix B 4) the closed-loop step responses are shown. with the estimated model and the controllers developed in the previous sections. when all four uncertain parameters undergo a non-time varying +20% paraneter change

We can see that tlie influence of paraneter perturbations on the process output is rather small for *all* three controllers That is. for the Pull. LQR and Robust controller This also can be seen from *figure 6 5* and *figure* 6 6 iii case of the robust controller. as there appears to be a relative small difference between the nominal performance and the robust performance

The result isn't that surprising, as we. during identification. already saw tliat most parameters have only little influence on the process output. which was our motivation for using a regularization technique

6.3 Conclusion and Final Remarks

We developed a (new) LQR colltroller and a robust colltroller. which are based on the process model. estimated at the end of the previous chapter
The new LQR colltroller seems to perform "better" *(figure* h 2 *1)* than the LQR controller developed in the previous chapter. which was used to perform our closed-loop ideiltification This results is not that surprising as the new LQR colltroller is based on a more accurate process model

We developed a robust colltroller by minimising the ∞-norm of the generalized plant, to deal with the real-structured parameter uncertainties in an *explicit* way
The regularization technique, used during identification in chapter five. helps us to select the "correct" uncertain parameters That is. the identification parameters that have a considerable influence on the process model output
The final largest singular value of the closed-loop system doesn't differ that much from the p-upper bound (structured singular value). such that it was not necessary to use one of those 'hen" synthesis techniques These so called 'hen" synthesis techniques cm deal with structured uncertainties. leading to a less conservative controller Their major drawback is that they lead to high order controller Therefore we discussed an easy way to determine a "small" unstructured uncertainty matrix from a "large" structured uncertainty matrix, such that "conventional" algorithms can used to calculate the controller

As our uncertain parameters appear in a MIMO state-space description. we introduce an easy way to incorporate their uncertainty, such that the derived generalised plant can easily be implemented in Matlab. bp making use of the μ-Analysis and Synthesis Toolbox.

The final Robust controller is compared with the earlier mentioned LQR controller and the differences arc rather "small". even when the uncertain parameter are perturbed
Let us remind that our LQR colltroller doesn't use an observer. to reconstruct the process states, and that an accurate process model was estimated, such that it was expected to perform "well"

It is shown that the closed-loop performance deterioration of the 3 coiltrollers Pull. LQR and the Robust controller is small in case all uncertain parameters undergo a non-time vaning +20% parameter change (*Figure* h 4 *2 - b* 4 *4)* This is not that surprising. as we during ideiltification already saw that most parameters have only little influence on the process output, which was our motivation for using a regularization technique
The difference in performance deterioration, for the different controllers can probably be increased by letting the parameter perturbations vary in time However. due to lack of time we were not able to investigate this validation method Let us remind that the H, controller guarantees robust performance for *any* stable transfer $\|A\|_\infty < 1$ (scaled). which allows time-varying uncertainties (dynamic uncertainties)

It is well known that the weighting filters. to specify performance, are of major importance. as the closed-loop performance depends highly on it (just as the choice of the model structure in system identification). It offers us a lot of freedom in directly shaping our system transfers

in the frequency domain Because of this large amount of freedom. care must be taken in developing the filters, as wrong choices can easily lead to bad performance or prevent us of finding tlie (sub)optimal solution We used simple first order weighting filters to reduce the controller order However, as was mentioned at the end of section 6 2 2. wc probably good perform better by using a second order filter, which gives us more degrees of freedom to incorporate the controller integration behaviour Due to lack of time. we were not able to investigate this possible improvement

Bibliography

[And90] B. D. O. Anderson and J. B. Moore. "Optimal control: linear quadratic methods".
Prentice-Hall, Englewood Cliffs, N.J, '90.

[Bac87] T. Backx. "Identification of an Industrial Process: A Markov Parameter Approach". PhD Thesis, Measurement & Control Group, Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands, '87.

[Bal93] G. J. Balas; J. C. Doyle; K. Clover; A. Packard and R. Smith. "p-Analysis and Synthesis Toolbox". User's guide version 2.0. The MathWorks, Inc, USA, '93

[Bra97b] Ed Bras "Program Listing. for identifying and controlling a MIMO process" Department of Automatic Coiltrol & System Engineering. University of Valladolid. Spain, '97

[Cla95] S. Toffner-Clausen. "System identification and robust control. A synergistic approach". PhD thesis. Department of control engineering, Aalborg university, Denmark.

[Dam96a] A. Damen. "State control". Course: Modem Control Theory, second part (5N050). Measurement & Control Group, Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands, April '96.

[Dam96b] A. Damen and S. Weiland. "Robust control". Course: robuuste regelingen (5P430). Measurement & Control Group, Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands, August '96.

[Fran94] G F Franklin; J D Ponell and A Emami-Naeini "Feedback control of dynamic systems". third edition Addison-Wesley Publishing Company, '94

[Gra92] A. Grace. "Optimization toolbox", User's guide. The MathWorks, Inc, USA. '92.

[Hof93] P. M. J. v.d. Hof and R. J. P. Schrama. "An indirect method for transfer function estimation from closed loop data". Automatica '93 Vol: 29 Iss: 6

[Hof95b] P. M. J. v.d. Hof and R. J. P. Schrama. "Identification and Coiltrol - Closed Loop Issues". Automatica '95 Vol: 31 Iss: 12.

[Kel93] T. Mckelvey. "System ideiltification using Overparametrized state-spacc models". Intern rapport, Department of Electrical Engineering Linköping University, Sweden, '93.

[Kla95] A C v d Klauw " Closed-loop Identification issues in the process industry" PhD Thesis, Mechanical Engineering System and Control Group. Delft University, The Netherlands, '95

[Lin96] P Lindskog "Methods, Algorithms and Tools for Sj stem Identification Based on Prior Knowledge" PhD Thesis, Department of Electrical Engineering Linköping University. Sweden, '96

[Lju87] L Ljung "System identification Theory for the User" Preiltice-Hall. Englewood Cliffs, N J. '87.

[Lju91] L Ljung "System ideiltification toolbox", User's guide The MathWorks, Inc. USA. '91

[Lju92] L Ljung; J Sjöberg and T Mckelvey "On the use of regularization in system ideiltification" Intern rapport. Department of Electrical Engineering Linköping University. Sweden, '93

[Mak95] P. M. Makila. J. R. Partington and T. K.Gustafsson." Worst-case control-relevant idelltification". Automatica '95 Vol: 31 Iss: 12.

[Nin95] B Ninness and G C Goodwin "Estimation of model quality" Automatica '95 Vol 31 Iss 12

[Söd89] T Söderström and P Stoica "System identification' Prentice-Hall, Englewood Cliffs, N J. '89

[Tul90] H J A F Tulleken "Generalized binary noise test-signal concept for improved identification-experiment design" Automatica '90 Vol 26 Iss 1

[Val82] M.E. v. Valkenburg. "Analog Filter Design". Holt. Rinehart and Winston, '82

[Ver95] G Verkroost "Digital signaalbewerking" Course Digitale signaalbewerking (5J040) Electronic Signal Processing Sj steins Group, Department of Electrical Engineering. Eindhoven University of Technology. The Netherlands, April '96

[Vri94b] D de Vries. "Identification of model uncertainty for control design". PhD thesis, Mechanical Engineering System and Control Group, Delft University, The Netherlands. '94.

[Zh096] K Zhou; J. C Doyle and K Glover "Robust control and optimal control" Preiltice-Hall. Information and Science Series, New Yersey, '96

Appendix A

Identification results

This appendix contains preparation and identification results of the laboratory-located process. The identification is performed in chapter 6 and the theoretical background c m be found in chapter 3.

A.l Mathematical Process Model

In this section a mathematical process model will be derived, of which the final results are given in section 2.2.

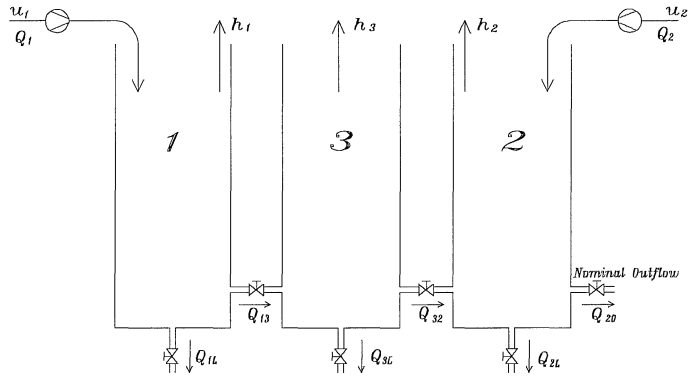


***Figure** a1.**1:** The process*

If we make a balance for every tank, we obtain the following 3 equations:

$$\begin{aligned} Q_1(t) - Q_{13}(t) - Q_{1L}(t) &= A \cdot \dot{h}_1(t) \\ Q_2(t) + Q_{32}(t) - Q_{20}(t) - Q_{2L}(t) &= A \cdot \dot{h}_2(t) \\ Q_{13}(t) - Q_{32}(t) - Q_{3L}(t) &= A \cdot \dot{h}_3(t) \end{aligned} \quad , \qquad (a.1.1)$$

where:

Q_{ij} (t) The water flow, as indicated in the fig. a.1.1 [cm^3/s].
A The section area of a tank [cm^2].
h_i(t) The water level in the tank (bottom of tank represents the zero level)[cm].

The following expression describes the relation between the tank level and the velocity (m/s) of a liquid in an opening of a tank (*Torricelli*)

$$v(t) = \sqrt{2gh(t)}\,, \tag{a.1.2}$$

where g is the gravity "constant"
With *(a* **1** 2). *Q(t)* in the interconnections valves becomes

$$Q_{ij}(t) = a \cdot \mathrm{sgn}(h_i(t) - h_j(t)) \cdot \sqrt{2 \cdot g \cdot |h_i(t) - h_j(t)|} \tag{a.1.3}$$

In which a is the section area of the interconnection or a leakage [cm^2]. This parameter is influenced by the position of the valve.
Combining (N.1.I) and *(a.*1.3) yields:

$$\begin{aligned}
\dot{h}_1(t) &= \frac{1}{A}(Q_1(t) - a_{I13}\,\mathrm{sgn}(h_1(t) - h_3(t))\sqrt{2g|h_1(t) - h_3(t)|} - a_{L1}\sqrt{2gh_1(t)}) \\
\dot{h}_2(t) &= \frac{1}{A}(Q_2(t) + a_{I32}\,\mathrm{sgn}(h_3(t) - h_2(t))\sqrt{2g|h_3(t) - h_2(t)|} - a_{L2}\sqrt{2gh_2(t)}) \\
\dot{h}_3(t) &= \frac{1}{A}(a_{I13}\,\mathrm{sgn}(h_1(t) - h_3(t))\sqrt{2g|h_1(t) - h_3(t)|} - a_{I32}\,\mathrm{sgn}(h_3(t) - h_2(t))\sqrt{2g|h_3(t) - h_2(t)|} - a_{L3}\sqrt{2gh_3(t)})
\end{aligned} \tag{a.1.4}$$

Notice that the section area in the interconnection between tank 1 and tank 3 is indicated as a_{I13}

We can linearize equation *(a.*1 4). using a first order approximation

$$h_t(k) \approx h_i + \sum_{i=1} \frac{\partial h_t(k)}{\partial h_i(k)}_{h_i\ u_s} \Delta h_i(k) + \sum_{j=1}^{2} \frac{\partial h_t(k)}{\partial u_j(k)}_{h_i\ u_s} \Delta u_j(k)\,. \tag{a.1.5}$$

with r,t=1..3 and s=1..2.

Where h_i indicates the value of h, in the working point in which we linearize
Before we linearize. we have to decide what our working point will be. to get rid of the *sign* and *absolute* functions:

In the working point we choose $h_1 = h_2$

We control h_1 and h_2, and h_3 is simply following the levels in these tanks Therefore the level in h_3 will *always* be a bit lower than the one in h_1 and h_2, *in* the working point!
Using *(a* **1** *5)* to linearize *(a* **1** *4)*, gives us the final linearized process model

$$\begin{aligned} \dot{x}_p(t) &= A_{p,c}x_p(t) + B_{p,c}u(t) \\ h(t) &= C_{p,c}x_p(t) \end{aligned}, \qquad (a.1.6)$$

with:

$$A_{p,c} = \begin{bmatrix} \psi_{11} & 0 & \psi_{13} \\ 0 & \psi_{22} & \psi_{23} \\ \psi_{31} & \psi_{32} & \psi_{33} \end{bmatrix}; \; B_{p,c} = \begin{bmatrix} \psi_{14} & 0 \\ 0 & \psi_{25} \\ 0 & 0 \end{bmatrix}; \; C_{p,c} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \qquad (a.1.7)$$

Notice that

- √ me are only interested in the process model in the working point, so that we left out the working point $\dot{h}_r$ and also use h_i instead of Δh_i;
- √ we have include the index 'c', to indicate the continuous model. Index 'd' will be used to indicate the discrete model;
- √ the meaning of the parameters in the matrices $A_{p,c}$; $B_{p,c}$; $C_{p,c}$ are summarised the end of this section.

We use a computer to identify our process with **A/D** converters which contain Zero Order Hold (*ZOH*) components, to convert the signal.
So the most natural way to approximate the derivatives in *(a. 1.6)*, is by a *ZOH* approximation:

$$\dot{h}(k) = \frac{h(k+1) - h(k)}{T}, \qquad (a.1.8)$$

where T is the sampletime and the time index k is equal to iT with $i = 1..\infty$.

$Q_i(t)$ is controlled by the input signal $u_i(t)$, with $i = 1,2$, by low-level servo systems The dynamic of these controllers are relative fast. compared with our process dynamics, that we can approximate the transfer between Q, (t) and u_i (t) by a static gain, which can be found in the documentation of the process

If a e use the *ZOH* approximation and we write the equations directly in matrix form, we get

$$\begin{aligned} x_p(k+1) &= A_{p,d}x_p(k) + B_{p,d}u(k) \\ h(k) &= C_{p,d}x_p(k) \end{aligned}, \qquad (a.1.9)$$

with

$$A_{p,d} = \begin{bmatrix} \tau_{11} & 0 & \tau_{13} \\ 0 & \tau_{22} & \tau_{23} \\ \tau_{31} & \tau_{32} & \tau_{33} \end{bmatrix}; \; B_{p,d} = \begin{bmatrix} \tau_{14} & 0 \\ 0 & \tau_{25} \\ 0 & 0 \end{bmatrix}; \; C_{p,d} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \qquad (a.1.10)$$

where:

$$\begin{aligned}
&\tau_{11} = (1 + T\psi_{11}) \;;\; \tau_{13} = T\psi_{13} \;;\; \tau_{14} = T\psi_{14} \\
&\tau_{22} = (1 + T\psi_{22}) \;;\; \tau_{23} = T\psi_{23} \;;\; \tau_{25} = T\psi_{25} \\
&\tau_{33} = (1 + T\psi_{33}) \;;\; \tau_{31} = T\psi_{31} \;;\; \tau_{32} = T\psi_{32}
\end{aligned} \qquad (a.1.11)$$

To determine a *non-linear model*, we return to equation *(a.1 4)* and use a *ZOH* approximation to obtain a discrete model. which yields

$$\begin{bmatrix} h_1(k) \\ h_2(k) \\ h_3(k) \end{bmatrix} = \begin{bmatrix} \varphi_{11} & \varphi_{12} & \varphi_{13} & \varphi_{14} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \varphi_{25} & \varphi_{26} & \varphi_{27} & \varphi_{28} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \varphi_{39} & \varphi_{3.10} & \varphi_{3.11} & \varphi_{3.12} \end{bmatrix} \begin{bmatrix} \dfrac{h_1(k-1)}{\sqrt{h_1(k-1)}} \\ \operatorname{sgn}(h_1(k-1) - h_3(k-1))\sqrt{|h_1(k-1) - h_3(k-1)|} \\ u_1(k-1) + 10 \\ \dfrac{h_2(k-1)}{\sqrt{h_2(k-1)}} \\ \operatorname{sgn}(h_3(k-1) - h_2(k-1))\sqrt{|h_3(k-1) - h_2(k-1)|} \\ u_2(k-1) + 10 \\ \dfrac{h_3(k-1)}{\sqrt{h_3(k-1)}} \\ \operatorname{sgn}(h_3(k-1) - h_2(k-1))\sqrt{|h_3(k-1) - h_2(k-1)|} \\ \operatorname{sgn}(h_1(k-1) - h_3(k-1))\sqrt{|h_1(k-1) - h_3(k-1)|} \end{bmatrix},$$

(a.1.12)

where:

$$\begin{aligned}
&\varphi_{11} = \theta_{11} \;;\; \varphi_{12} = T\theta_{12} \;;\; \varphi_{13} = T\theta_{13} \;;\; \varphi_{14} = T\theta_{14} \\
&\varphi_{25} = \theta_{21} \;;\; \varphi_{26} = T\theta_{22} \;;\; \varphi_{27} = T\theta_{23} \;;\; \varphi_{28} = T\theta_{24} \\
&\varphi_{39} = \theta_{31} \;;\; \varphi_{3,10} = T\theta_{32} \;;\; \varphi_{3,11} = T\theta_{33} \;;\; \varphi_{3,12} = T\theta_{34}
\end{aligned} \qquad (a.1.13)$$

Notice that we perform an offset correction on the input *u(k-1)* of 10 Volt. as the process input range is I-10.101 Volt and the output range [0,60] cm..

We can write *(a* 1.*12)* as a linear regression structure:

$$h(k) = \Theta_N \cdot \Omega_N(k-1) \qquad (a.1.14)$$

The meaning of the matrix entries in *(a.1.7)* and **(a.I.12)**:

$$\Psi_{11}=\frac{\theta_{13}}{2\sqrt{\overset{\circ}{h}_1-\overset{\circ}{h}_3}}+\frac{\theta_{12}}{2\sqrt{\overset{\circ}{h}_1}}\;;\;\Psi_{13}=\frac{-\theta_{13}}{2\sqrt{\overset{\circ}{h}_1-\overset{\circ}{h}_3}}\;;\;\Psi_{14}=\theta_{14}$$

$$\Psi_{22}=\frac{\theta_{23}}{2\sqrt{\overset{\circ}{h}_2-\overset{\circ}{h}_3}}+\frac{\theta_{22}}{2\sqrt{\overset{\circ}{h}_2}}\;;\;\Psi_{23}=\frac{\theta_{23}}{2\sqrt{\overset{\circ}{h}_2-\overset{\circ}{h}_3}}\;;\;\Psi_{24}=\theta_{24} \qquad (a.1.15)$$

$$\Psi_{31}=\frac{\theta_{34}}{2\sqrt{\overset{\circ}{h}_1-\overset{\circ}{h}_3}}\;;\;\Psi_{32}=\frac{-\theta_{33}}{2\sqrt{\overset{\circ}{h}_2-\overset{\circ}{h}_3}}\;;\;\Psi_{33}=\frac{\theta_{34}}{2\sqrt{\overset{\circ}{h}_1-\overset{\circ}{h}_3}}+\frac{\theta_{33}}{2\sqrt{\overset{\circ}{h}_2-\overset{\circ}{h}_3}}+\frac{\theta_{32}}{2\sqrt{\overset{\circ}{h}_3}}$$

and:

$$\theta_{12}=1\;;\;\theta_{12}=-\frac{a_{L1}}{A}\sqrt{2g}\;;\;\theta_{13}=-\frac{a_{I13}}{A}\sqrt{2g}\;;\;\theta_{14}=\frac{c_1}{A}$$

$$\theta_{21}=1\;;\;\theta_{22}=-\frac{(a_{I20}+a_{L2})}{A}\sqrt{2g}\;;\;\theta_{23}=\frac{a_{I23}}{A}\sqrt{2g}\;;\;\theta_{24}=\frac{c_2}{A} \qquad (a.1.16)$$

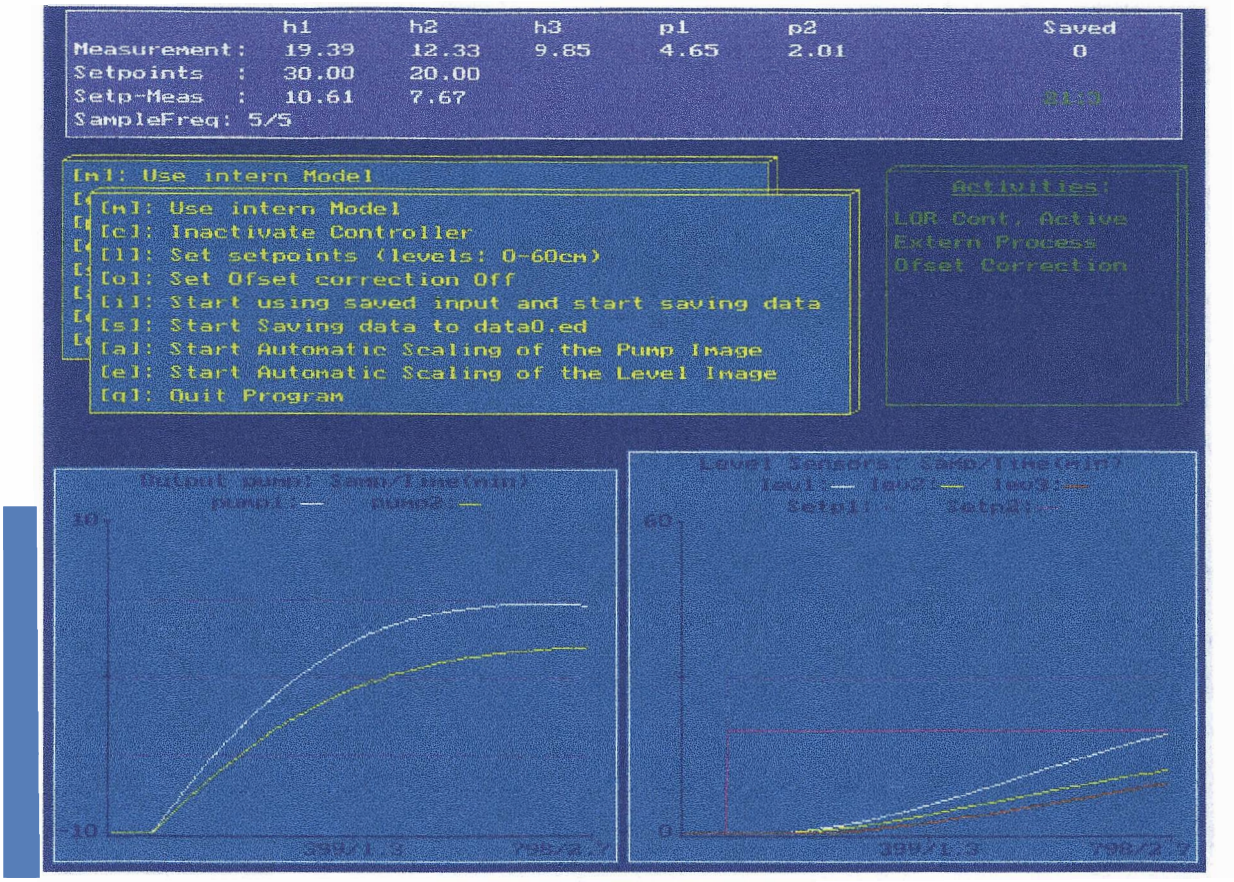
$$\theta_{31}=1\;;\;\theta_{32}=-\frac{a_{L3}}{A}\sqrt{2g}\;;\;\theta_{33}=-\frac{a_{I23}}{A}\sqrt{2g}\;;\;\theta_{34}=\frac{a_{I13}}{A}\sqrt{2g}$$

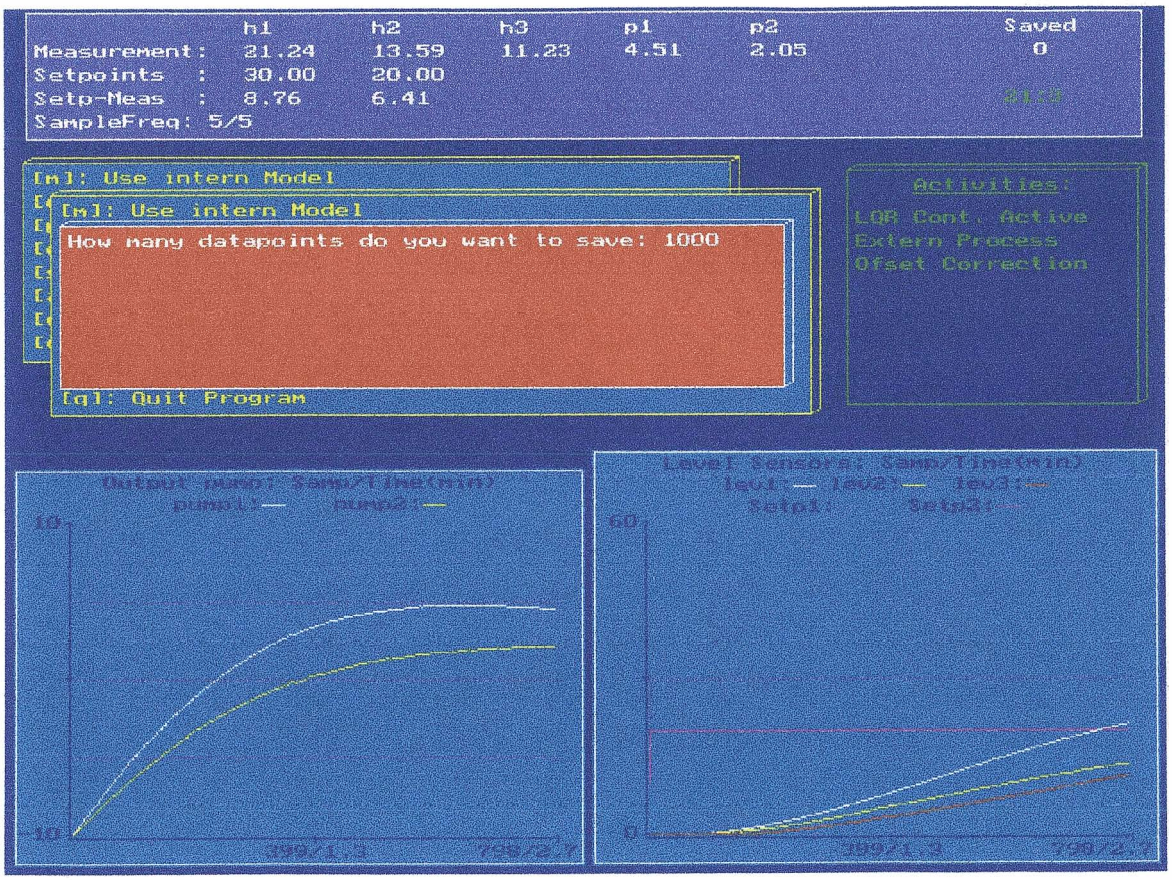
All that left us is to give the values of the constants $c_{,}$, which is the static gain of the transfer from u_i to Q,. the section area of tank (A) and those of the interconnections (a_{Iij}) and leakage (a_{Li}), when the valves are *entirely* opened These values can be found in the documentation of the process

$$A=154\ \text{cm}^2\ .\ a_{Iij}=0\ 4\ \text{cm}^2\ ,\ a_{Li}=0\ \text{S}\ \text{cm}^2\ ,\ c_i=5\ \text{cm}^3/(\text{sV})$$

A.2 Flexible Environment

Two screen-dumps are shown, of the software that makes it possible to control the laboratory-located process. The software is described in section 2.1




h1 h2 h3 p1 p2 Saved
Measurement: 31.24 13.59 11.23 4.51 2.05 0
Setpoints : 30.00 20.00
Setp-Meas : 8.76 6.41
SampleFreq: 5/5
[m]: Use intern Model
[m]: Use intern Model
How many datapoints do you want to save: 1000
[q]: Quit Program
Activities:
LQR Cont. Active
Extern Process
Ofset Correction
10
-10
60
0

A.3 Results of the Closed-Loop Controllers

This section contains some results of the two controllers, which were designed to perform a close-loop identification.
The controllers are a Pull and LQR controller, of which the LQR controller is finally used for identification
The section contain a step response, with the real process and with the simulation model, and bode plots of the simulation model.
The step response of the real process is performed with a controller calculation frequency of 5 Hz (section 5.3.1). Before data is collected and controller calculation takes place, analog and digital filtering is performed to prevent aliasing. The analog and digital anti-aliasing filters are described in section 5.3.2.

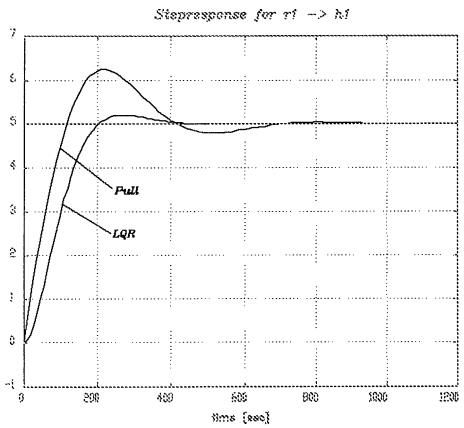


***Figure a.3.1.a**: a step response of the calculated simulation model with the Pull and LQR controller from input 1 to output 1.*

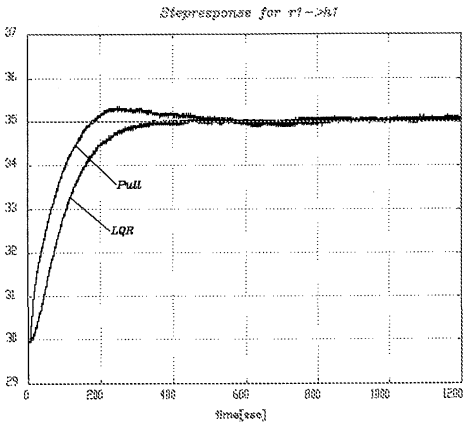


***Figure a.3.1.b:** a step response of the real process with the Pull and LQR controller from input 1 to output 1.*

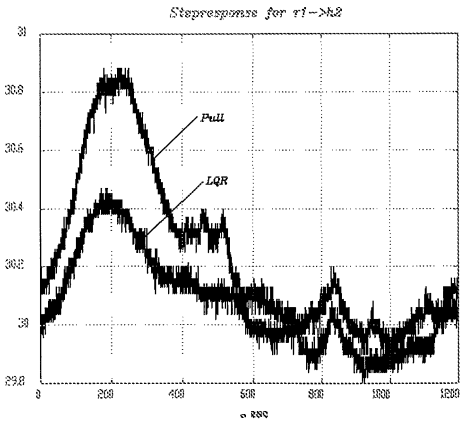


***Figure n.3.1.c: a** step response of the real process with the Pull and LQR controller from input 1 to output 2.*

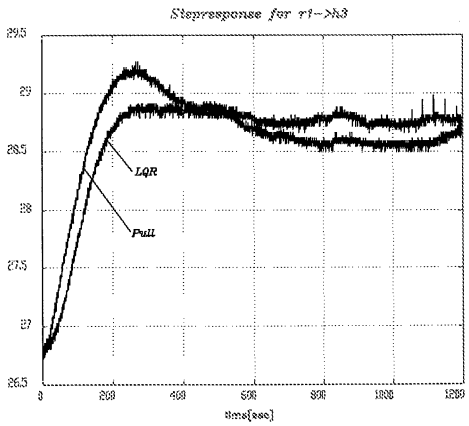


***Figure a.3.1.d:** a step response of the real process with the Pull and LQR controller from input 1 to output 3.*

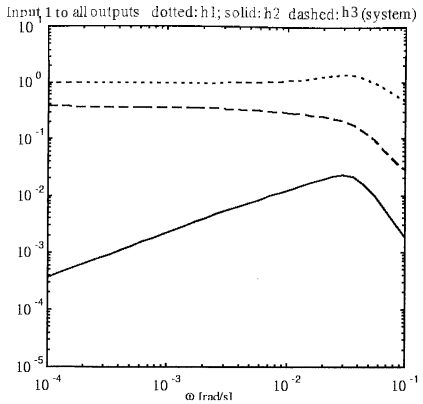


***Figure** n.3.2.a: bode plot of the Pull controller from input 1 to all outputs with the calculated simulation model.*

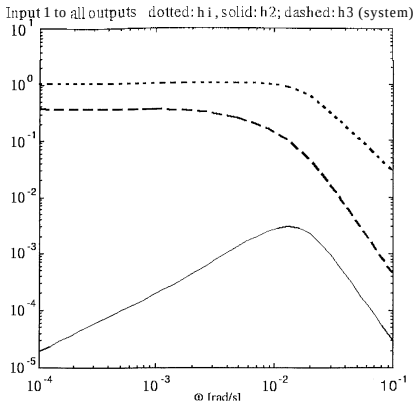


***Figure** n.3.2.b: bode plot of the LQR controller from input 1 to all outputs with the calculated simulation model.*

A.4 Pre-Processing Results

The meaning of the used symbols in the figures:

GBNv	generalized binary noise input signal, with a variable amplitude;
GBN	generalized binary noise input signal, with a constant amplitude;
BN	binary noise input signal, with a constant amplitude;
r_i	setpoint i, with i=1,2 (input);
h_i	water level i, with i=1,2 (output).

– **Data after scaling; offset correction, filtering and sample rate reduction:**

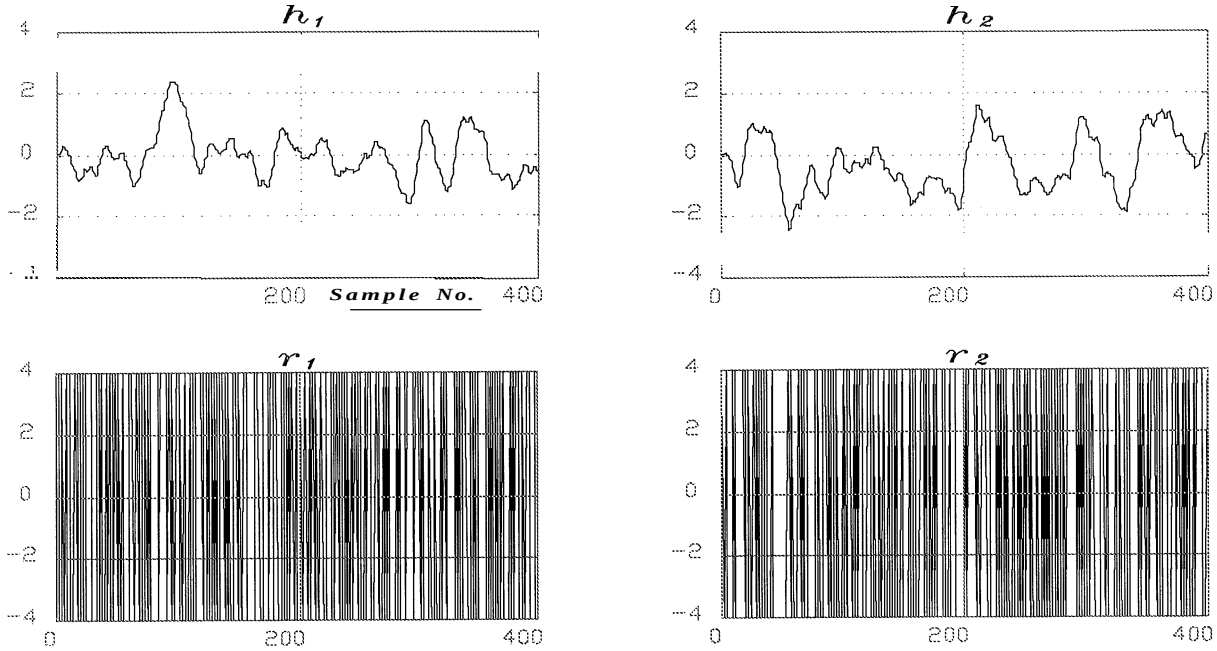


Figure ***a.4.1.a:*** collected *data after, offset correction, filtering and sample rate reduction, with BN* as *input signal.*

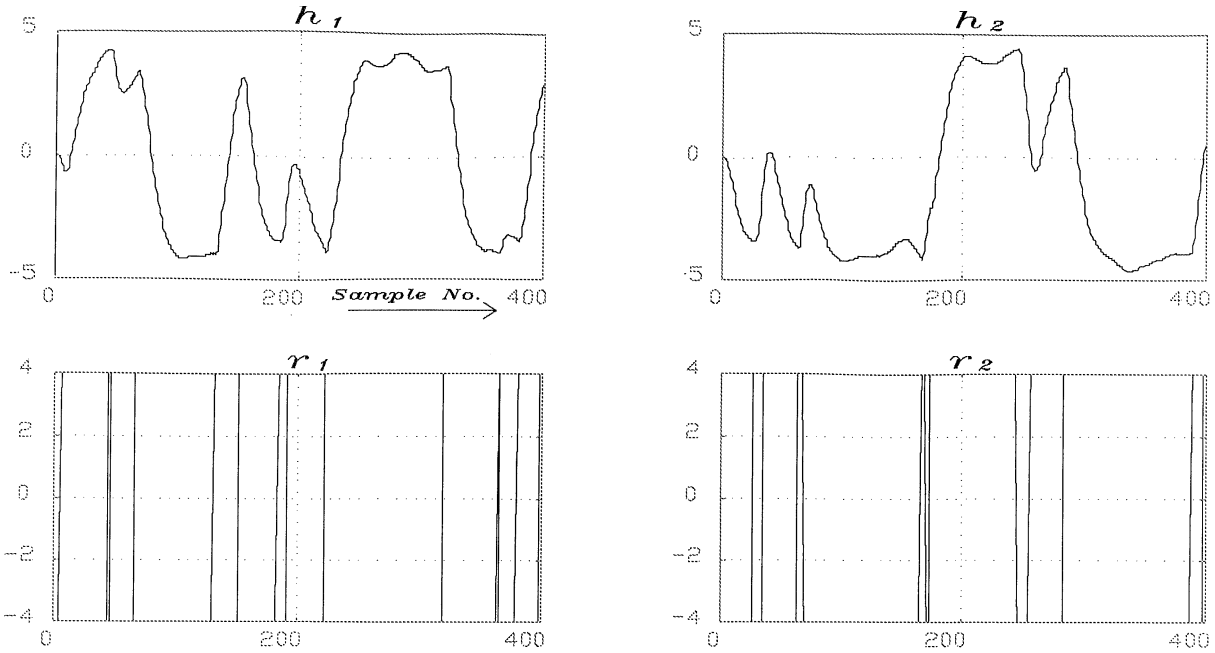


Figure n.4.3.b: *collected data after, offset correction, filtering and sample rate reduction, with GBN as input signal.*

A.5 Validation results

The validation results of the calculated non-linear model and linear model.

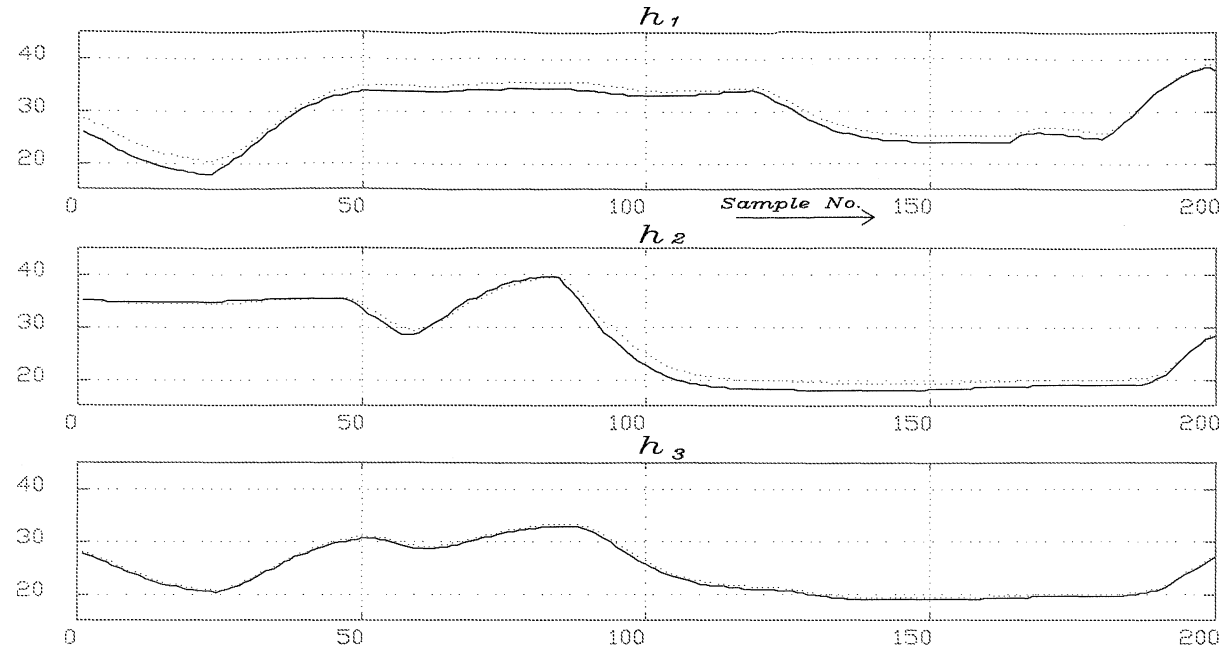


Figure a.5.1: *validation by simulation of the calculated iron-linear-model (solid) with the collected data (dotted).*

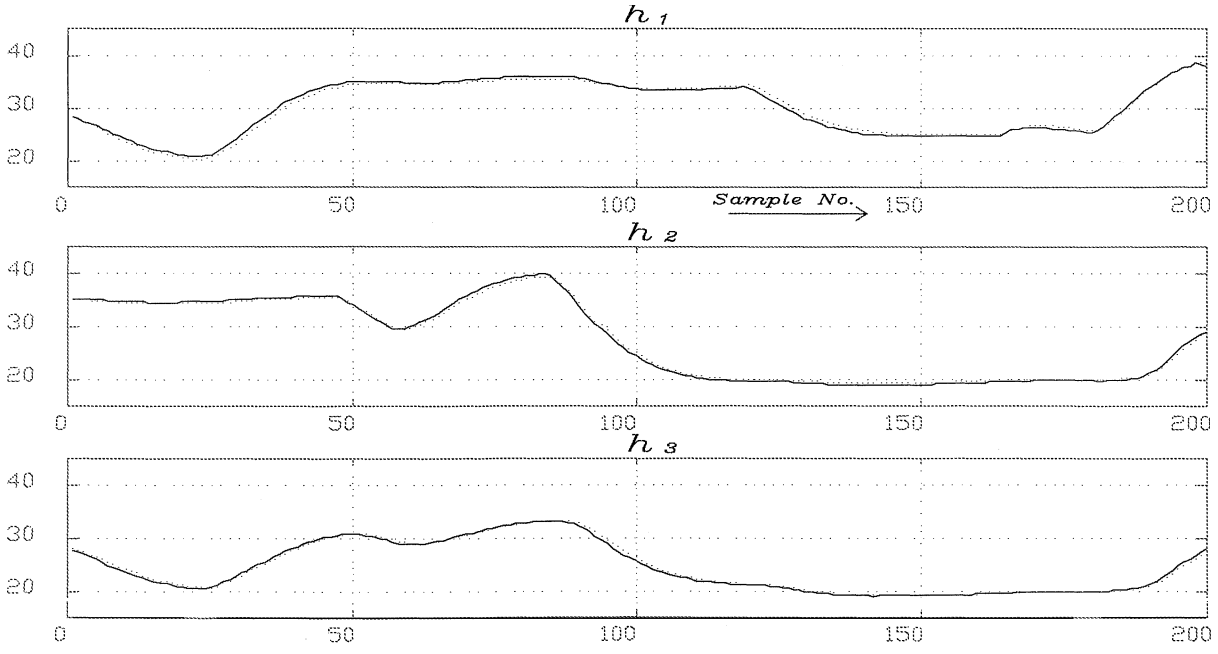


Figure a.5.2: *validation by simulation of the estimated rrorr-linear model (solid) with the collected data (dotted).*

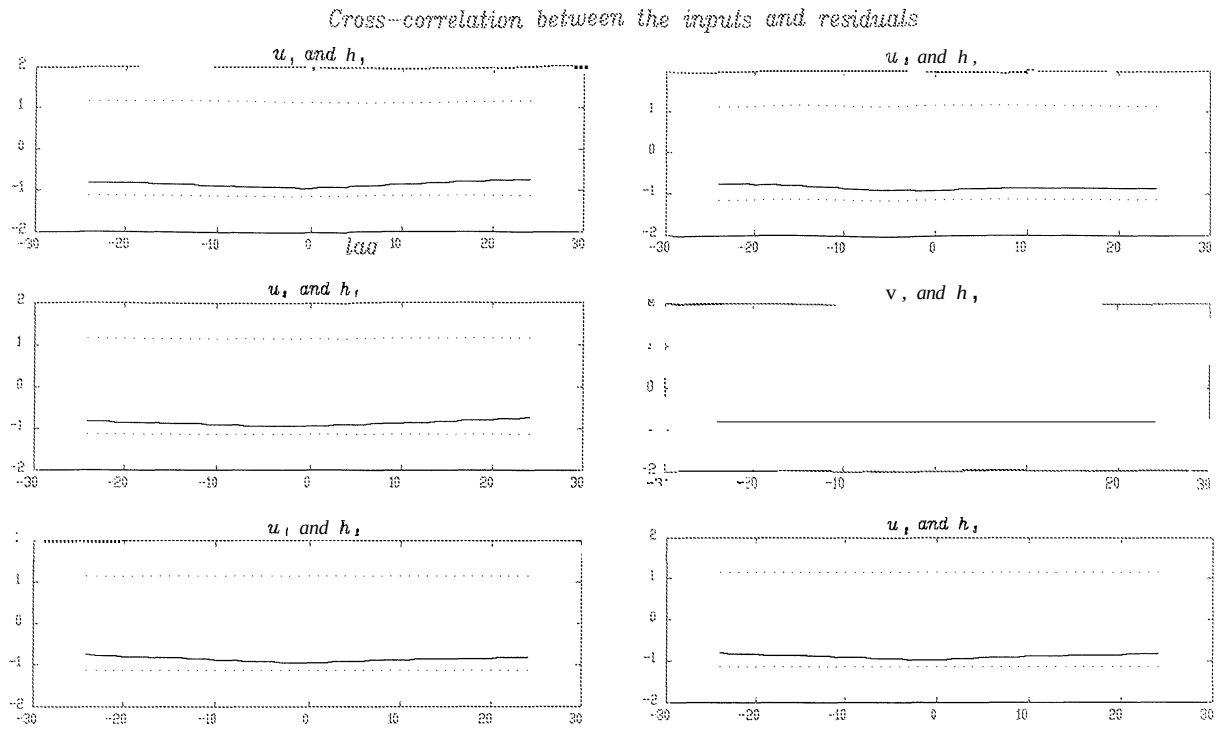


Figure *a.5.3: cross-correlation between the inputs and the residuals for the non-linear model, with parameter constraints.*

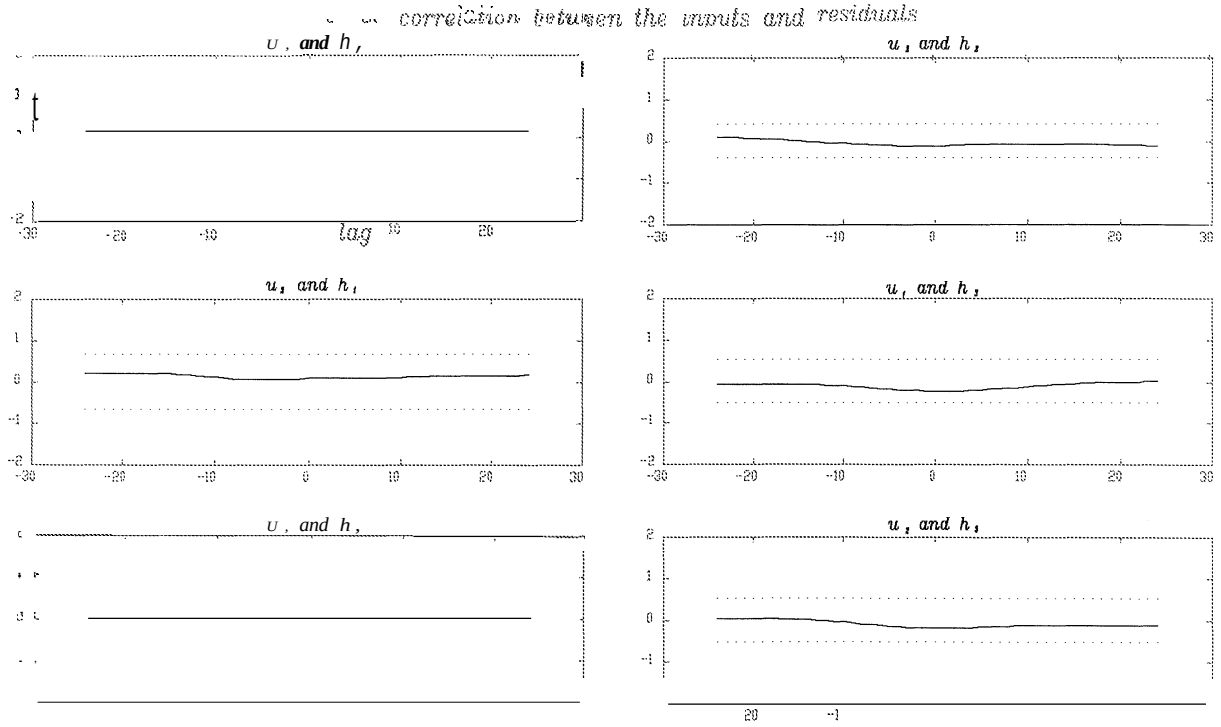


Figure a.5.4: *cross-correlation between the inputs and the residuals for the non-linear model, without parameter constraints.*

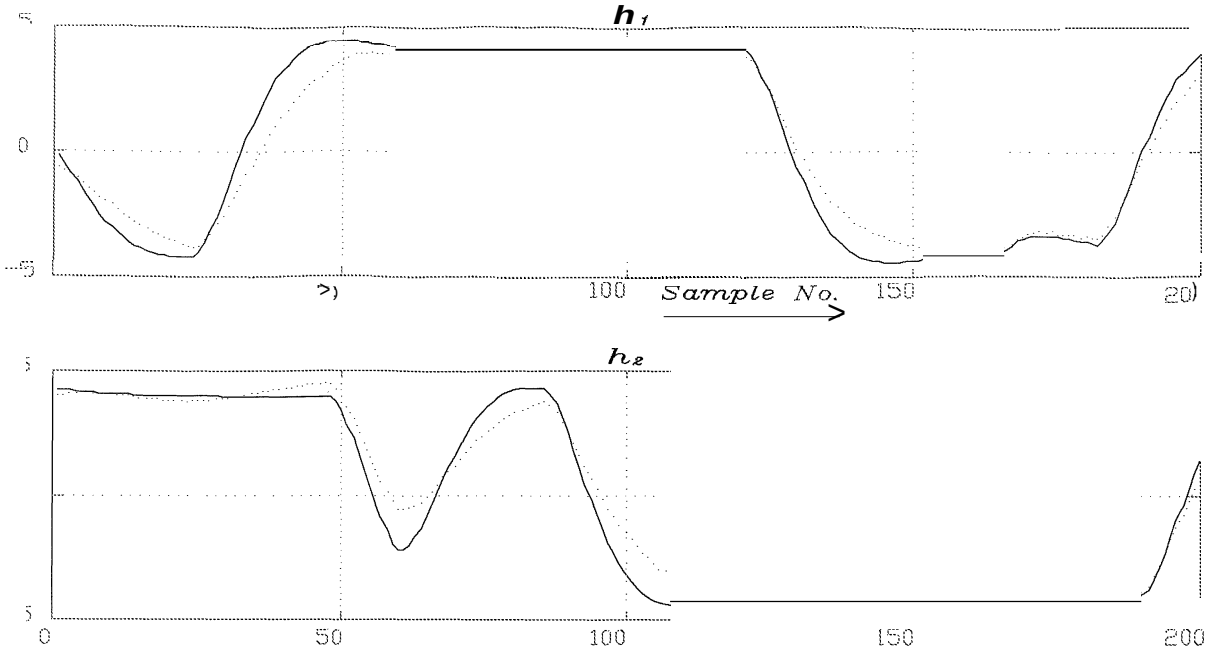


Figure a.5.5: *validation by simulation of the calculated linear. model (solid) with the collected data (clotted).*

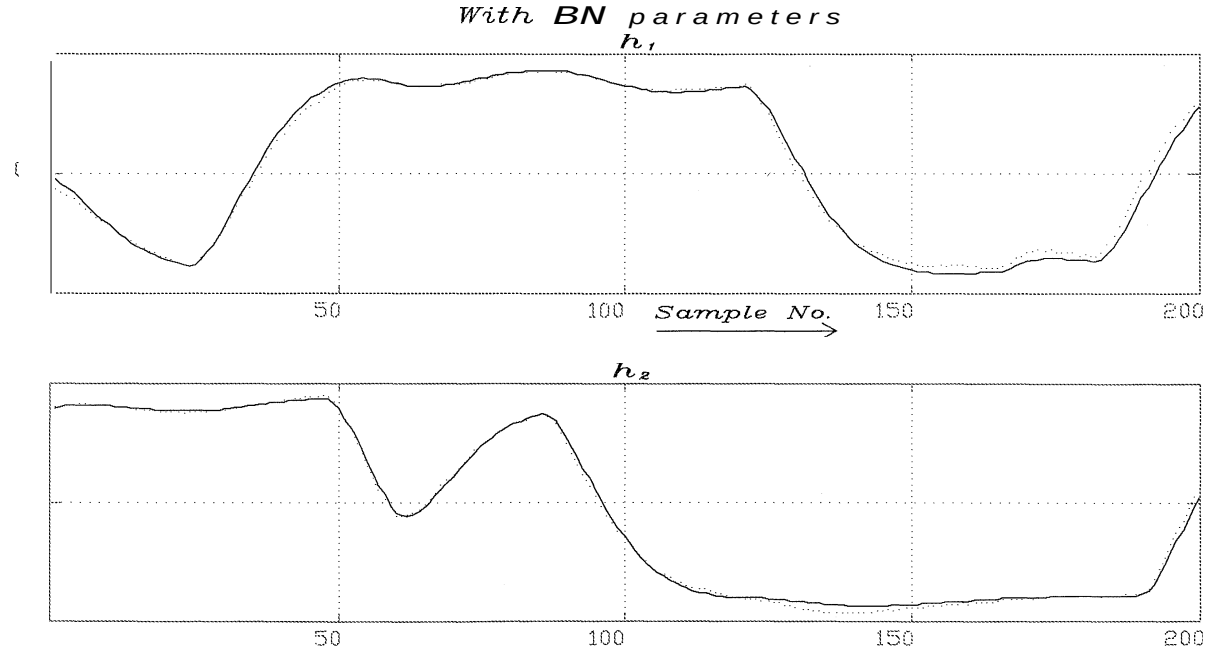


Figure a.5.6.a: *validation by simulation of the estimated linear-model (solid), when two parameters are fixed, with thr collected data (dotted).*

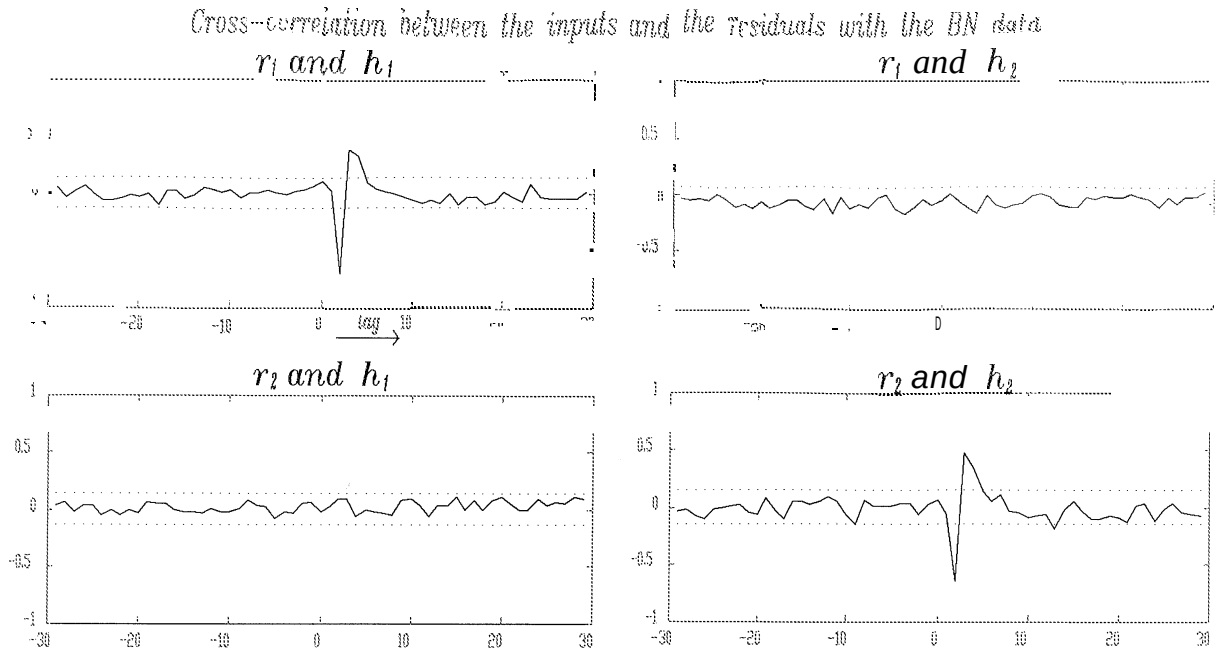


Figure n.5.6.b: *cross-correlation between the inputs and the residuals for the linear model, when two parameters are fixed.*

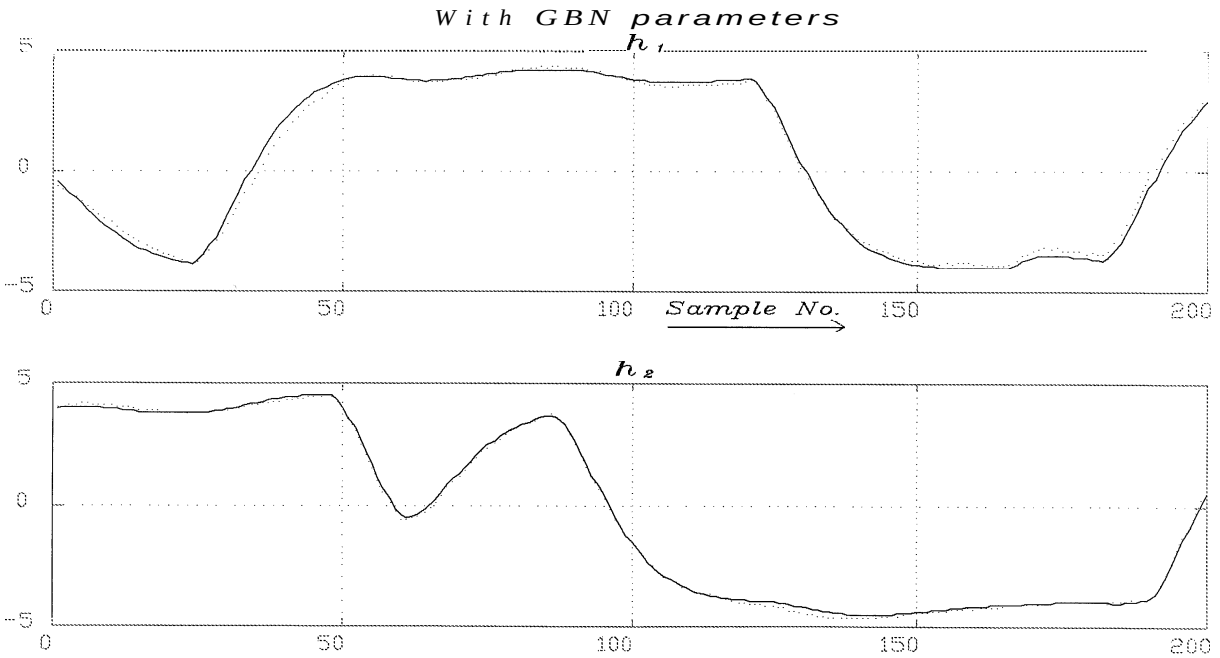


Figure a.5.7.a: *validation by simulation of the estimated linear model (solicl), when two parameters are fixed, with the collected data (dotted).*

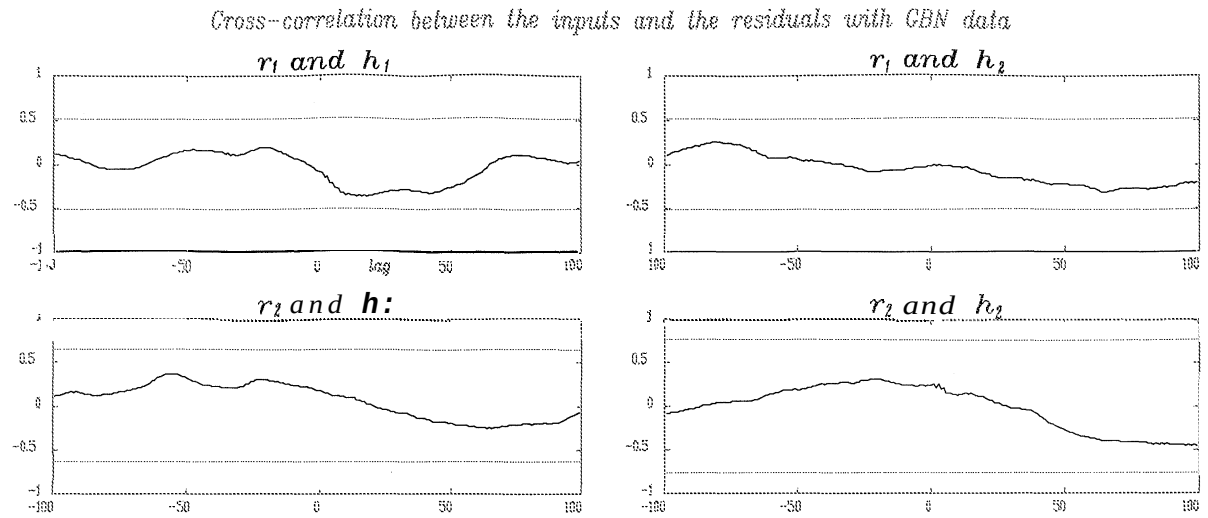


Figure a.5.7.b: *cross-correlation between the inputs and the residuals for the linear model, when two parameters are fixed.*

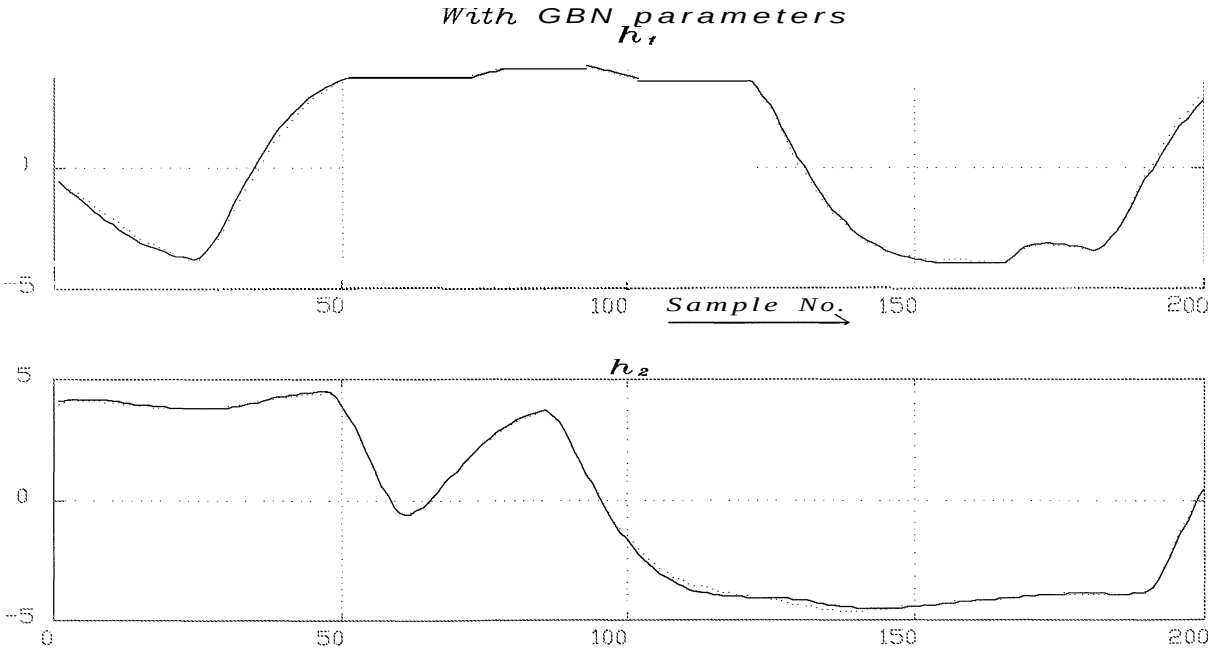


Figure a.5.8.a: *validation* **by** *simulation of the estimated linear model (solid), with regularization, with the collected data (dotted).*

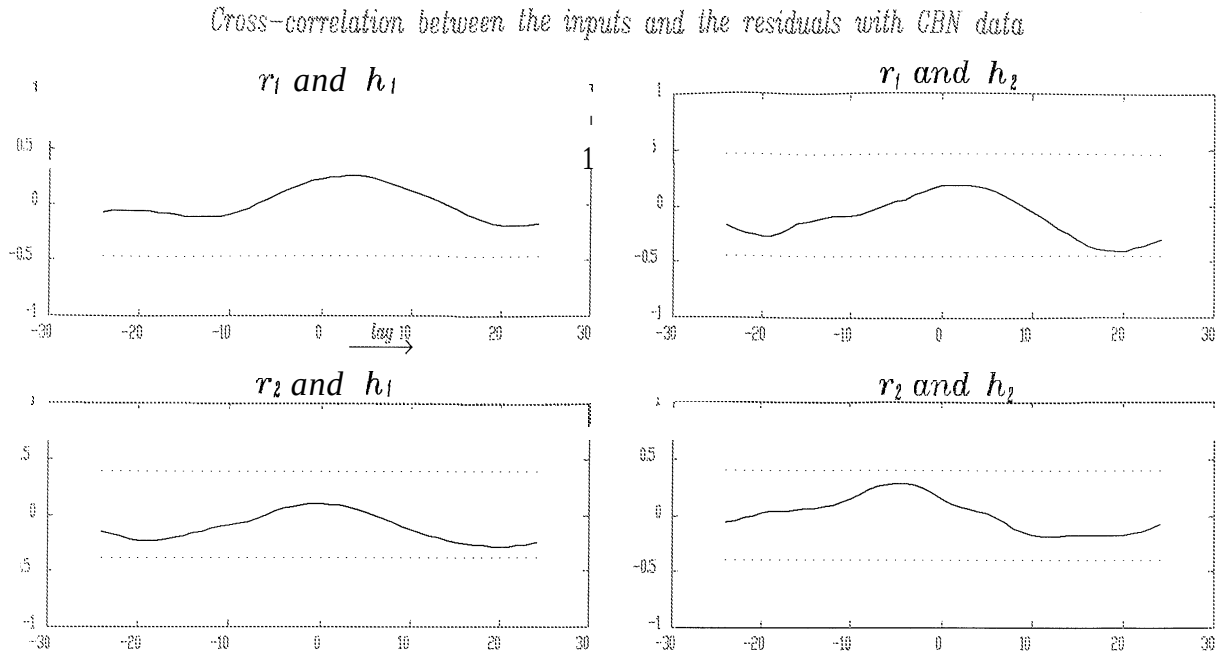


Figure a.5.8.b: *cross-correlation between the inputs and the residuals for the linear model, with regularization.*

Appendix B

Controller results

This appendix contains results, and preparations of the design of the LQR controller and the Robust controller. Both controllers are developed in chapter 6 and their theoretical background is given in chapter 4.

B.1 Return Difference Equality

The optimal LQR feedback gain *(4.5.a)* was given as:

$$K = -R^{-1}B_p{}^T P\,, \qquad (b.1.1)$$

and the steady state Riccati equation *(4.j.b)* as:

$$PA_p + A_p{}^T P - PB_p R^{-1} B_p{}^T P + Q = 0\,. \qquad (b.1.2)$$

From *(b.1.1) it follow,* keeping in mind that Q, R and P are symmetric, that:

$$PB, = -K^T R\,. \qquad (b.1.3)$$

With *(b.1.1), (b.1.2)* can be rewritten as

$$P(j\omega I - A_p) + (-j\omega I - A_p^T)P + K^T RK = Q\,. \qquad (b.1.4)$$

Multiplying on the left by $B^T{}_p(-j\omega I\text{-}A^T{}_p)^{-1}$ and on the right by $(j\omega I\text{-}A_p)^{-1}B_p$ yields

$$-B_p^T(-j\omega I - A_p^T)^{-1}K^T R - RK(j\omega I - A_p)^{-1}B_p + B_p^T(-j\omega I - A_p^T)^{-1}K^T RK(j\omega I - A_p)^{-1}B_p = B_p^T(-j\omega I - A_p^T)^{-1}Q(j\omega I - A_p)^{-1}B_p \quad . \quad (b.1.5)$$

Reorganising (b. *1.5)* yields the *return difference equality (4.14):*

$$R + B_p^1(-j\omega I - A\ )^{-1}Q \quad - A_p)^{-1}B_{\text{„}} = [I - B_p^{'1}(-j\omega I - A_p^{'1})^{-1}K^{'1}]R[I - K(j\omega I - A_p)^{-1}B_{\text{„}} \quad . \quad (b.1\ 6)$$

B.2 LQR controller step responses

In this section we compare two LQR controllers. One was based on our calculated model, which we used to perform our closed-loop identification. The second LQR controller is based on our estimated model, as discussed in section 6.1.

The step responses of the real process are performed with a controller calculation frequency of 5 Hz (section 5.3.1). Before data is collected and controller calculation takes place, analog and digital filtering is performed to prevent aliasing. The analog and digital anti-aliasing filters are described in section 5.3.2.

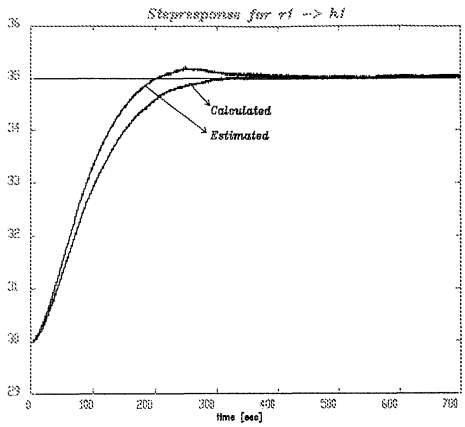


Figure b.2.1.a: *a step response of the estimated closecl-loop model with 2 different LQR controllers from input 1 to output 1.*

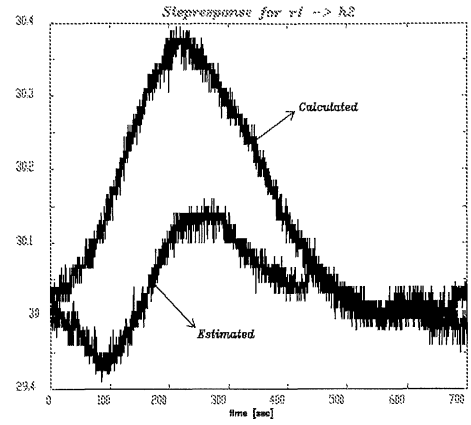


Figure b.2.1.b: *a step response of the estimated closed-loop model with 2 different LQR controllers from input 1 to output 2.*

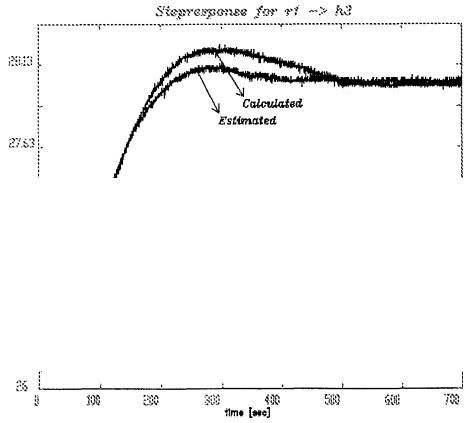


*Figure **b.2.1.c:** a step response of the estimated closed-loop model with 2 different LQR controllers from input 1 to output 3.*

B.3 State-Space Uncertainty Presentation

The graphical state-space presentation of the process *(a.1.6)*, with the uncertain parameters defined as in (6.3). For an explanation of the used symbols, see sectioii 6.2.

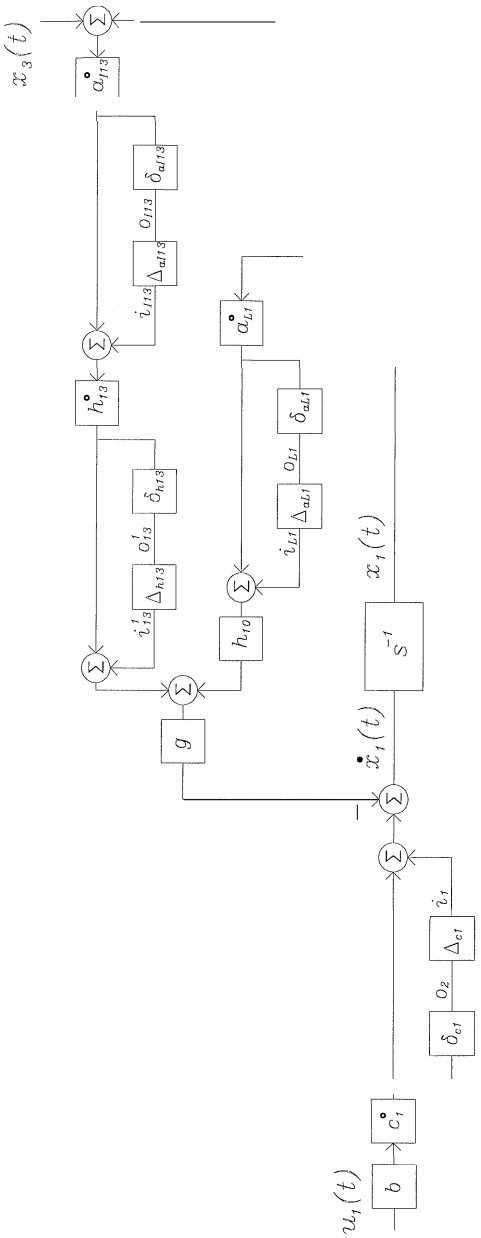


Figure ***b.3.1:*** The state space presentation of $x_1(t)$.

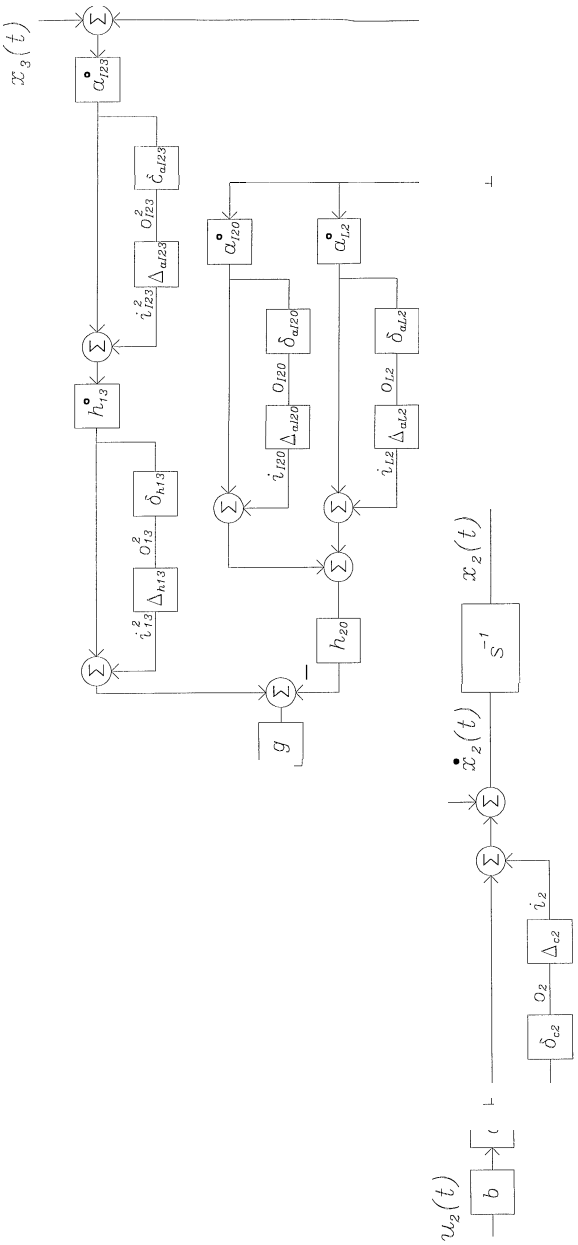


Figure b.3.2: *The state space presentation of $x_2(t)$.*

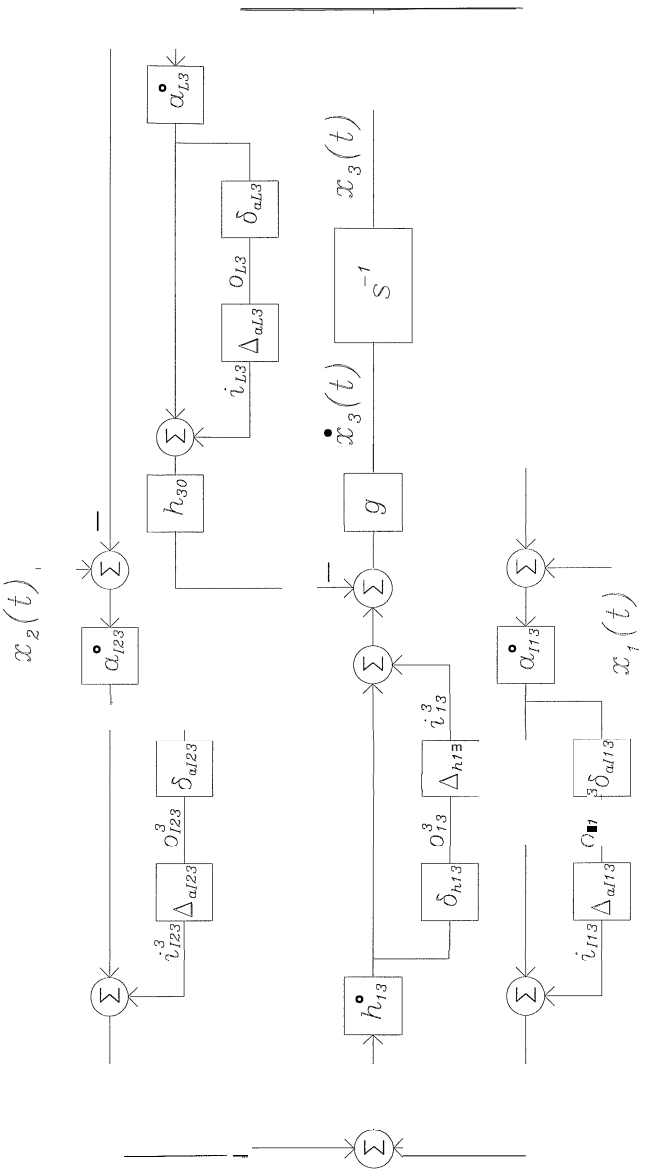


Figure b.3.3: *The state space presentation of $x_3(t)$.*

The matrix entries of W, $W_{,}$, , W_{in}, $W_{,}$ [redacted] Δ_{pe}

This section contains the matrices used to specify the parameter uncertainties, in the state space uncertainty presentation. showed in *figure 6.1*. The matrices are summarised in case we are assuming nine, and four (independent) uncertain parameters.
To spare paper and for reasons of conveniently. we only summarise the matrix entries. The remainder entries are *zero*.
Our process model has 2 inputs and 2 process outputs, 3 states and an explanation of the symbols (parameters) can be find in *(a.1.15)*, *(a.I.16)*, *(6.3)*, *(6.5)* and *(6.6)*.

– W_i **and** Δ_{pe} **with nine (independent) uncertain parameters** :

$W_u \in \mathfrak{R}^{12\times 2}$.

$w_{11,1}$	$\delta_{c1} \cdot c_{1} \cdot b$
$w_{12,2}$	$\delta_{c2} \cdot c_{2} \cdot b$

$W_{out} \in \mathfrak{R}^{3\times 12}$:

w_{11}	$-g \cdot h_{10}$	w_{22}	$-g \cdot h_{20}$	w_{33}	$-g \cdot h_{30}$
w_{14}	$-g \cdot h_{13}$	w_{25}	$-g \cdot h_{13}$	w_{34}	$g \cdot h_{13}$
w_{18}	$-g$	w_{27}	$-g \cdot h_{20}$	w_{36}	$-g$
$w_{1,11}$	1	w_{29}	g	$w_{3,10}$	$g \cdot h_{13}$
		$w_{2,12}$	1		

$W_{in} \in \mathfrak{R}^{12\times 3}$:

w_{11}	$\delta_{aL1} \cdot a_{L1}$	w_{22}	$\delta_{aL2} \cdot a_{I2}$	w_{33}	$\delta_{aL3} \cdot a_{L3}$
w_{41}	$\delta_{aI13} \cdot a_{I13}$	w_{52}	$\delta_{aI23} \cdot a_{I23}$	w_{43}	$\delta_{aI13} \cdot a_{I13}$
w_{81}	$\delta_{h13} \cdot h_{13}\, a_{I13}$	w_{62}	$\delta_{aI23} \cdot a_{I23}$	w_{53}	$\delta_{aI23} \cdot a_{I23}$
$w_{10,1}$	$\delta_{h13} \cdot h_{13}\, a_{I13}$	w_{72}	$\delta_{aI20} \cdot a_{I20}$	w_{63}	$-\delta_{aI23} \cdot a_{I23}$
		w_{92}	$\delta_{h13} \cdot h_{13}\, a_{I23}$	w_{83}	$\delta_{h13} \cdot h_{13}\, a_{I13}$
		$w_{10,2}$	$\delta_{h13} \cdot h_{13}\, a_{I23}$	w_{93}	$\delta_{h13} \cdot h_{13}\, a_{I23}$
				$w_{10,3}$	$\delta_{h13} \cdot h_{13}\, (a_{I13} - a_{I23})$

$W_m \in \Re^{12\times 12}$:

w_{84}	$\delta_{h13}\cdot h_{13}$
w_{95}	$\delta_{h13}\cdot h_{13}$
$w_{10,4}$	$\delta_{h13}\cdot h_{13}$
$w_{10,6}$	$\delta_{h13}\cdot h_{13}$

$\Delta_{pe} \in \Re^{12x12}$:

Is a diagonal matrix, which we indicate with diag{·}:

$\Delta_{pe} - \text{diag}\{\Delta_{aL1}, \Delta_{aL2}, \Delta_{aL3}, \Delta_{aI13}, \Delta_{aI23}, \Delta_{aI23}, \Delta_{aI20}, \Delta_{h13}, \Delta_{h13}, \Delta_{h13}, \Delta_{c1}, \Delta_{c2}\}$

– W_i **and A„ with four (independent) uncertain parameters:**

$W_{out} \in \Re^{3\times 5}$:

w_{11}	$-g\cdot h_{10}$	w_{24}	$-g\cdot h_{13}$	w_{32}	$-g\cdot h_{30}$
w_{13}	$-g\cdot h_{13}$			w_{33}	$g\cdot h_{13}$
				w_{35}	$-g\cdot$

$W_{in} \in \Re^{5\times 3}$:

w_{11}	$\delta_{aL1}\cdot a_{L1}$	w_{42}	$\delta_{aI23}\cdot a_{I23}$	w_{23}	$\delta_{aL3}\cdot a_{L3}$
w_{31}	$\delta_{aI13}\cdot a_{I13}$	w_{52}	$\delta_{aI23}\cdot a_{I23}$	w_{33}	$\delta_{aI13}\cdot a_{I13}$
				w_{43}	$\delta_{aI23}\cdot a_{I23}$
				w_{53}	$-\delta_{aI23}\cdot a_{I23}$

W, and W, become zeros, as can easily be seen from *figure b.3.1-3.*

A*, $\in \Re^{5\times 5}$:

$\Delta_{pe} - \text{diag}\{\Delta_{aL1}, \Delta_{aL3}, \Delta_{aI13}, \Delta_{aI23}, \Delta_{aI23}\}$

B.4 Robust Controller Results

This section contains results of the final robust controller with four uncertain parameters, which was developed in section *6.2.* The robust controller is also compared with the LQR controller developed in section 6.1.

The final fifth order continue robust controller:

$$\begin{aligned} \dot{x}_c(t) &= A_c x_c(t) + B_c in(t) \\ out(t) &= C_c x_c(t) + D_c in(t) \end{aligned}, \qquad (b.4.1)$$

with:

$$A_c = 10^{-2} \cdot \begin{bmatrix} -8.446 & 0 & 0 & 0 & 0 \\ 0 & -5.436 & 0 & 0 & 0 \\ 0 & 0 & -3.874 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}; \quad B_c = \begin{bmatrix} 3.824 \cdot 10^{-1} & 1.971 \cdot 10^{-2} & 4.907 \cdot 10^{-2} \\ -3.365 \cdot 10^{-1} & 5.805 \cdot 10^{-2} & -1.112 \cdot 10^{-2} \\ -5.871 \cdot 10^{-2} & -1.207 \cdot 10^{-1} & -4.920 \cdot 10^{-2} \\ -1.995 \cdot 10^{-2} & -5.162 \cdot 10^{-2} & -1.805 \cdot 10^{-9} \\ 5.724 \cdot 10^{-2} & -5.143 \cdot 10^{-2} & 0 \end{bmatrix}$$

$$C_c = \begin{bmatrix} 1.162 \cdot 10^{-1} & 1.098 \cdot 10^{-1} & 6.773 \cdot 10^{-2} & 2.613 \cdot 10^{-2} & -5.910 \cdot 10^{-2} \\ 1.205 \cdot 10^{-2} & -7.562 \cdot 10^{-3} & 9.734 \cdot 10^{-2} & 4.614 \cdot 10^{-2} & 4.632 \cdot 10^{-2} \end{bmatrix}$$

$$D_c = \begin{bmatrix} -1.041 \cdot 10^{-3} & 2.082 \cdot 10^{-3} & -1.850 \cdot 10^{-3} \\ 6.820 \cdot 10^{-4} & -8.780 \cdot 10^{-4} & 1.717 \cdot 10^{-3} \end{bmatrix}$$

The step response of input 1, with the real process, with both the LQR controller and Robust controller, developed in chapter 6:

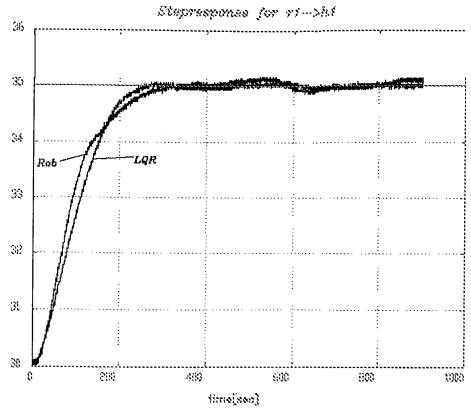


Figure 0.4.1.n: *step response of the real process with the LQR controller and Robust controller from input 1 to output 1.*

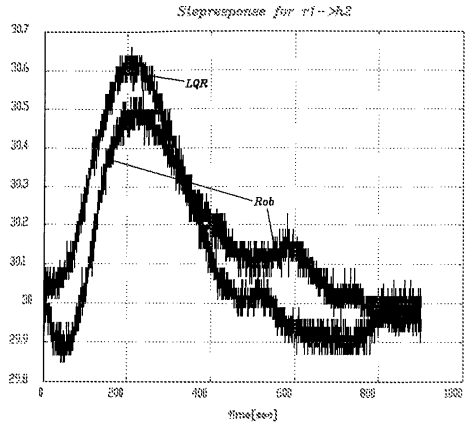


Figure b.4.1.b: *step response of the real process with the LQR controller and Robust controller from input 1 to output 2.*

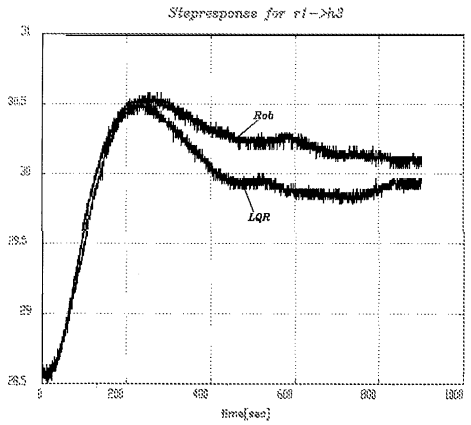


Figure b.4.1.c: *step response of the real process with the LQR controller and Robust controller from input 1 to output 3.*

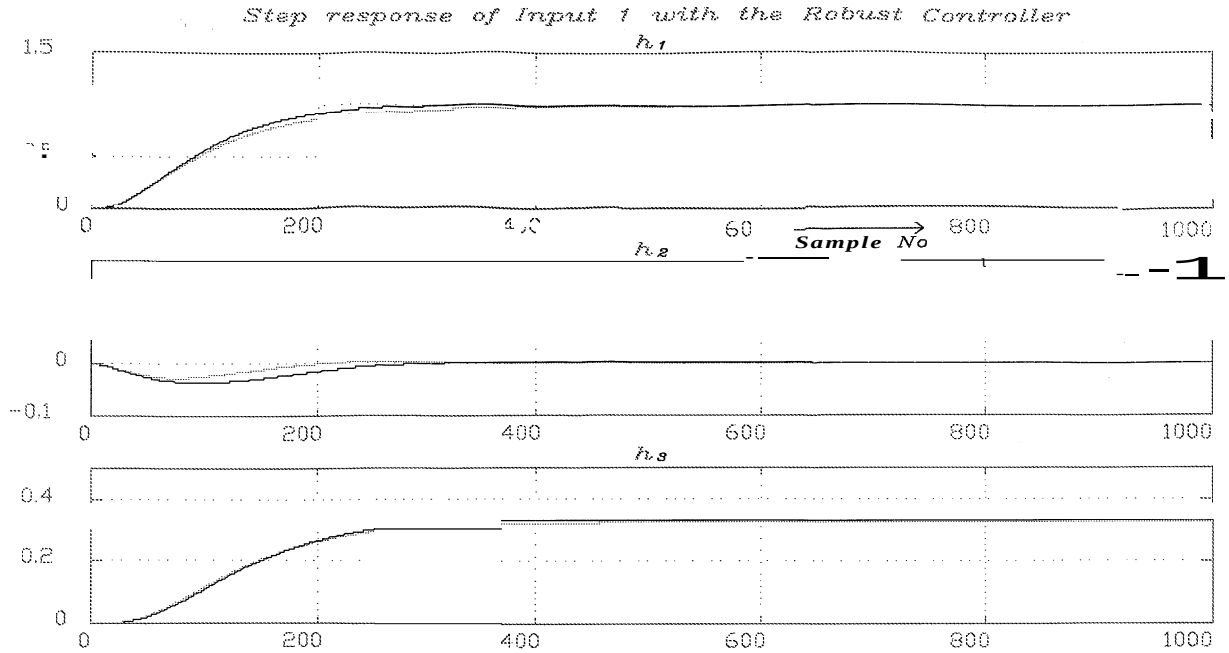


Figure b.4.2: *the step response of input 1 without (solid) and with (dotted) a non-time varying parameter change of +20% , in case of 4 uncertain parameters.*

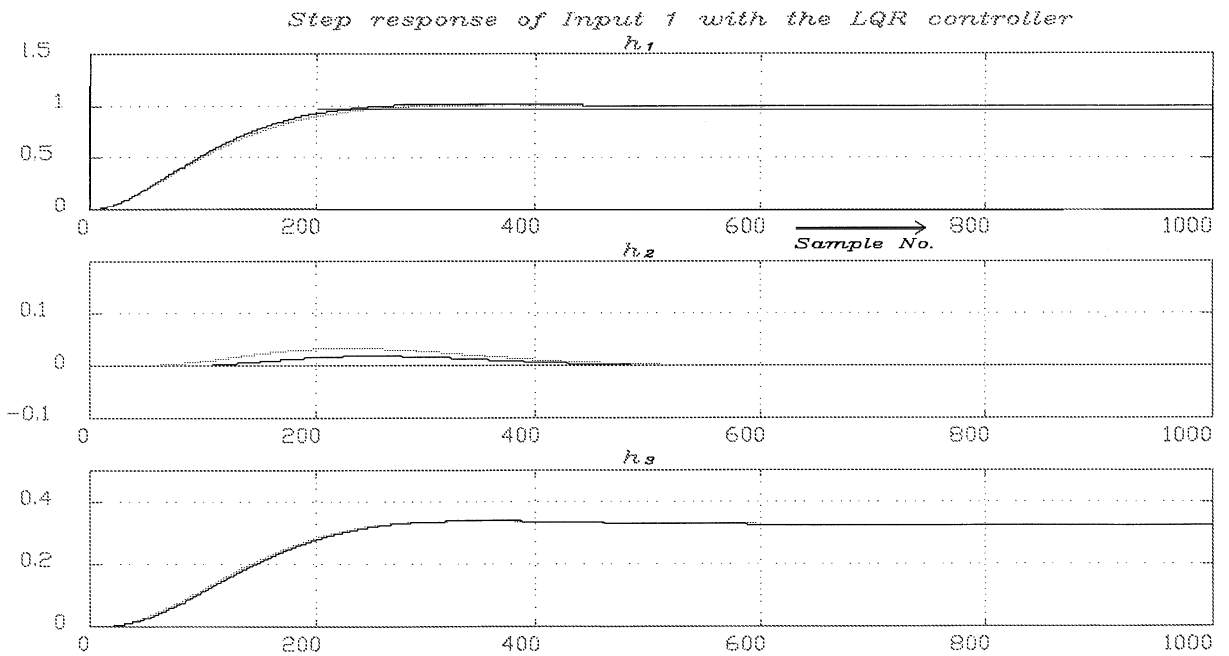


Figure 8.4.3: *the step response of input 1 without (solid) and with (dotted) a non-time varying parameter change of +20% , in case of 4 uncertain parameters.*

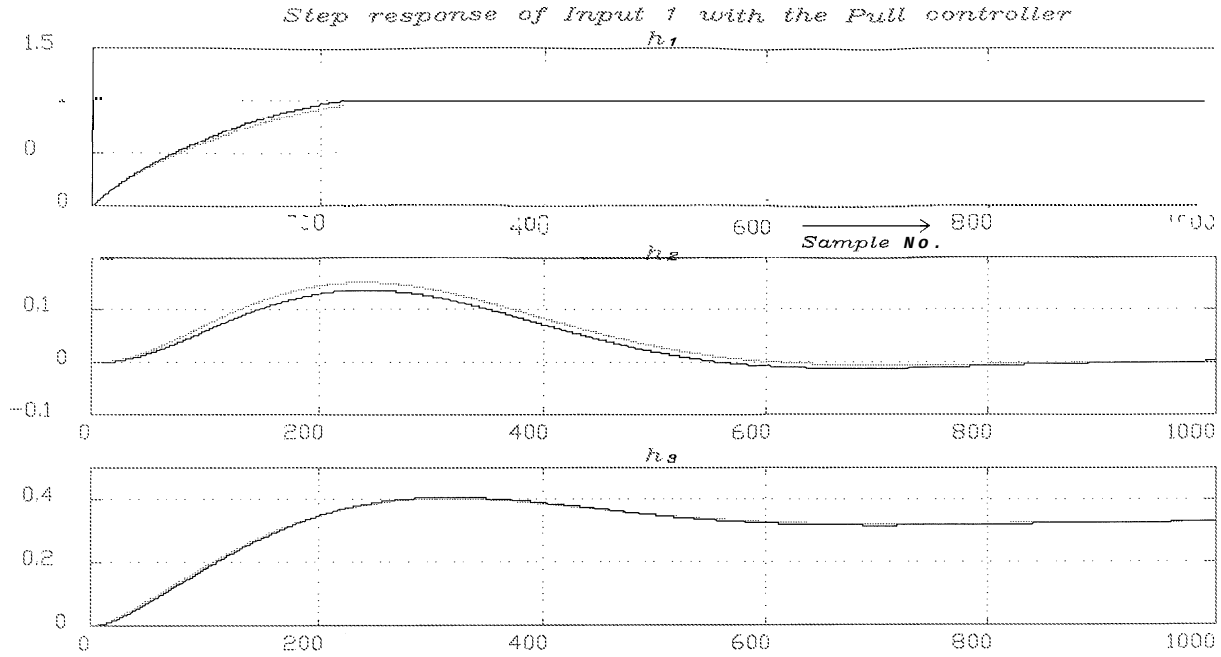


***Figure** 6.4.4: the step response of input I without (solid) and with (doffed)a non-time varying parameter change of +20% , in case of 4 uncertain parameters.*