Process Estimation with Relay Feedback Method

Adis Kurjak
Faris Mustajbasic

Department of Automatic Control
Lund University
February 2006
Abstract
Manual tuning of PID controllers can be very time-consuming and that is why automatic tuning was developed. The conventional automatic tuners use a relay instead of controller in a closed loop system to obtain stable oscillations. The information from a relay experiment is later used to calculate PID control parameters with different tuning rules and there also exist systems that, apart from the relay experiment, use a step response experiment to obtain even better control. The aim with this thesis is to investigate if there is enough information in a relay experiment to estimate an unknown process as a first order process with delay and then use AMIGO design to obtain satisfying PI/PID controller parameters. The calculation of the estimation parameters of the model is done with Gauss-Newton optimization algorithm. The algorithm minimizes the square of the output error between the unknown process output and estimation model output and calculates the optimal model parameters. The algorithm is dependant of good initial Svalues so a method for initializing good values is developed.
Preface

This master thesis is one of the requirements for the Master of Science in Electrical Engineering degree at Lund Institute of Technology (LTH) in Lund, Sweden. The quantity of this work corresponds to 40 weeks of full time work, i.e. 20 weeks per student.

We would like to thank our supervisors, professor Tore Hägglund, LTH, and professor Karl Johan Åström, LTH and University of California Santa Barbara, for helping us through this thesis and always having time when we needed help. Their hints and suggestions were very helpful when we got stuck in our work.

We are grateful to the research assistant, Ph.D. Anders Robertsson at Department of Automatic Control, LTH, for helping us with the practical experiment and we would like to thank the secretaries at the Department of Automatic Control for their kindness.

Last, but not the least, we would like to thank our families and our soul mates Jasmina and Amina for their support and being there for us.

Lund
Beginning of year 2006

Authors
Contents

Preface

1 Introduction ......................................................................................................... 1

2 Theory .................................................................................................................. 2
   2.1 PID Control ................................................................................................ 2
   2.2 Automatic Tuning of PID Controller ......................................................... 3
       2.2.1 The Ziegler-Nichols Step Response Method .................................... 3
       2.2.2 The Ziegler-Nichols Closed-Loop Method ....................................... 4
       2.2.3 Relay Feedback Method ................................................................. 5
       2.2.4 Describing Function Analysis ........................................................... 7
   2.3 Amigo Design Method ............................................................................... 8
   2.4 The Gauss-Newton Optimization Method ................................................ 9
       2.4.1 The Newton’s Method .................................................................... 9
       2.4.2 The Gauss-Newton Method ............................................................ 10

3 Modelling of First Order Process .................................................................... 12
   3.1 Estimation of First Order Process ............................................................ 12
   3.2 Implementation of Estimation .................................................................. 15
   3.3 First Order Results ................................................................................... 18

4 Estimation of Higher Order Processes ............................................................ 19
   4.1 High Order Process Results ..................................................................... 19
   4.2 Control Parameters .................................................................................. 30

5 Method for the Initiation of $K_p$, $T$ and $L$ ................................................ 32
   5.1 Backward AMIGO Method ..................................................................... 32
   5.2 Results of the Backwards AMIGO Method ............................................. 35

6 Real Process Test .............................................................................................. 37
   6.1 Estimation of a Tank Process ................................................................... 37
   6.2 Control of a Tank Process ........................................................................ 39
   6.3 Comment .................................................................................................. 41
7 Conclusions

7.1 Summary

7.2 Future Work

Appendix A

Appendix B

Appendix C

Bibliography
Chapter 1

Introduction

Since manual tuning of a PID controller is a very time-consuming process, automatic tuning was developed. Ziegler-Nichols tuning rules have been used for more than 50 years and today, there are several modifications of these tuning rules available. Automatic tuning is implemented in many industrial controllers, like the commercial PID controller ECA600 from ABB that uses a relay method to calculate PID-controller parameters. There are also controllers that use a step response method for this purpose.

The objective of this master’s thesis is to examine if there is enough information in a relay experiment so that there is no need for additional experiments, like the step response experiment. In this way, many disadvantages with additional methods would be avoided. The basic idea is to approximate the unknown process as a first order process with delay using Gauss-Newton optimizations method and find some proper initial values for the method. The control parameters can than be obtained using AMIGO\(^1\) design method.

\(^1\) Approximate M-constrained integral gain optimization
Chapter 2

Theory

Tuning of PID-controller parameters can be very time consuming if it is done manually. That is why automatic tuning was developed. There are several methods that can be used to auto-tune and in this chapter, the theory behind automatic tuning is treated. First, in section 2.1, PID-control is explained concisely. In section 2.2, the theory behind some simple, widely used, automatic tuning rules is explained. Section 2.3 treats the AMIGO design method and in the section 2.4 Gauss-Newton optimization method is treated.

2.1 PID Control

A PID-controller is a simple controller but a very effective one. Due to the flexibility of this controller, it is the most spread controller. It is used in the majority off all applications that use automatic control where it is provided that the performance requirements are not too high [4]. The mathematical expression for the PID-controller is given by

\[
G_{PID}(s) = K \left( 1 + \frac{1}{sT_i} + sT_d \right)
\]  

(2.1)

where \( K \) is gain, \( T_i \) is called integral time and \( T_d \) derivative time [8]. A PID controller can have serial or parallel structure. A simple parallel structure can be seen in figure 1.1.
Action control of the PID-controller is based on past and present error values and prediction of future control errors, where integral part acts on the average of past errors, the proportional part acts on present error value and derivative part acts as prediction of future errors based on linear extrapolation.

### 2.2 Automatic Tuning of PID Controller

Today, there exist several methods for tuning a PID-controller, like Ziegler-Nichols step response method and Ziegler-Nichols oscillation method. Manual tuning can be very time consuming, so it is very good and timesaving if it is done automatically and that is why automatic tuning has been introduced. Åström-Hägglund developed a method for auto tuning of PID-controllers that is related to Ziegler-Nichols oscillation method [4].

#### 2.2.1 The Ziegler-Nichols Step Response Method

In 1942, Ziegler and Nichols published a method where the PID parameters are determined with step response data [4]. The method uses two parameters, $a$ and $L$, that can be seen in figure 2.2. These parameters can be determined by looking at a maximum slope tangent of system’s step response. As figure 2.2 shows, parameter $a$ is the difference between the value where the tangent is intercepting the
horizontal axis and the initial value. The time difference between the time point when the step starts and the time point when the maximum slope tangent is intercepting the horizontal axis, is the time delay $L$. Time $T$ in figure 2.2 is time from when delay time ends to the time point when 63% of steady state value is reached. To calculate parameter $T$, gain $k$ has to be known. Relation between mentioned parameters is given by

$$a = \frac{k \cdot L}{T}$$  \hspace{1cm} (2.2)

The closed loop system that is received with this method is often poorly damped and the systems with better damping can be obtained by modifying these parameters [4]. Two essential drawbacks with this method are that it is difficult to know if steady state has been reached and how large step should be applied. The step should not be so large that the production is disturbed and it should be clearly above the noise level. For example if a step is applied on a process that has large rise time, it is hard to see in a step response if it is the step response of the process that is examined or if it is, for example, play in some valves in the process. Controller parameters obtained with this method are given in table 2.1.

<table>
<thead>
<tr>
<th>Type of controller</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$K_c$</td>
</tr>
<tr>
<td>P-controller</td>
<td>$1/a$</td>
</tr>
<tr>
<td>PI-controller</td>
<td>$0.9/a$</td>
</tr>
<tr>
<td>PID-controller</td>
<td>$1.2/a$</td>
</tr>
</tbody>
</table>

Table 2.1 Controller parameters obtained with Ziegler-Nichols step response method.

2.2.2 The Ziegler-Nichols Closed-Loop Method

By removing the integral and derivative part of controller we will receive pure proportional control and then by increasing the gain, oscillations of the output signal are observed. To calculate the controller gain $K_0$ and the period time of output oscillations, $T_0$, which this method uses in calculation of controller parameters, the gain has to be increased until stable oscillations are observed [2]. These values of the controller gain and the period time of the output oscillation is then used to calculate controller parameter values according to the table 2.2:
Chapter 2. Theory

<table>
<thead>
<tr>
<th>Type of controller</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$K_c$</td>
</tr>
<tr>
<td>P-controller</td>
<td>0.5 $K_0$</td>
</tr>
<tr>
<td>PI-controller</td>
<td>0.45 $K_0$</td>
</tr>
<tr>
<td>PID-controller</td>
<td>0.6 $K_0$</td>
</tr>
</tbody>
</table>

Table 2.2 Controller parameters obtained with Ziegler-Nichols frequency method.

The Ziegler-Nichols oscillation method has been used for more than 50 years although it has several drawbacks like poor robustness. One disadvantage with this method is that, when getting the parameters $K_c$, $T_i$ and $T_d$, one has to work with the closed loop system on the stability boundary with the risk of having uncontrollable oscillations that can harm the process. To avoid this, Åström-Hägglund developed so called relay experiment method where the controller is replaced by a relay. Since a relay has a limited output, stable oscillations are obtained [2].

### 2.2.3 Relay Feedback Method

To avoid the requirements, that when calculating controller parameter values with Ziegler-Nichols oscillation method, one has to work on the stability boarder with risk of having unstable oscillations, a relay method has been developed by Åström and Hägglund. In this method, a PID controller is replaced with a relay and stable and controllable oscillations are obtained, see figure 2.3 [4, 7].

![Figure 2.3 Schematics of a relay experiment.](image)

The relay works in this case as a constant gain that gives constant oscillations independent of the input signal, i.e. the amplitude and frequency in the output from the relay, a square wave, is determined by the relay’s hysteresis and its output level. An example of relay output signal and the process output can be seen in figure 2.5. If relay has no hysteresis, the working point will be on the
negative real axis in the Nyquist diagram (see fig. 2.4). By introducing hysteresis $h$, the working point will move away from the real axis as figure 1.4 shows with parameters $a$ and $b$ defined in figure 2.5 [2]. This leads to a system that is less sensitive to noise and disturbances.

Movement of the working point can be described with theory of describing functions explained in section 2.2.4. The phase lag is still 180° as in Ziegler-Nichols methods, but the great difference is that the phase lag in the relay method includes phase lag property of a relay and phase lag of the process. Since the sum of the phase lags of the process and the relay is 180°, the phase lag of the process will be lower than the one in Ziegler-Nichols methods.

To explain simply how this works, a process with low pass characteristics can be imagined. A square wave that enters a process will exit the process as a sine wave. By looking at the amplitude and the period time, $K_0$ and $T_0$, PID-parameters can be calculated according to table 2.2. It has been shown that this method will give low values at the stability margin and that is why Åström-Hägglund suggested a
modification of table 2.2, more than fifteen years ago. $K_c = 0.35K_u$, $T_i = 0.77T_u$ and $T_d = 0.19T_u$ were suggested as new PID parameters [2].

### 2.2.4 Describing Function Analysis

Describing function analysis is a method that can be used to predict and approximately analyze nonlinear behaviour. The basic idea is to approximate a nonlinearity by a linear equivalent and then use different frequency domain techniques to analyze the resulting system [3]. Since a relay is a nonlinearity, this analysis is used.

![ Relay feedback circuit.](image)

**Figure 2.6** Relay feedback circuit.

Signal $u(t)$ in figure 2.6 can be written as $u(t) = N(A, \omega) \cdot (-x)$ where $N(A, \omega)$ is a frequency response function of the relay, also called a describing function that is amplitude and frequency dependant. A relay’s describing function without hysteresis is given by

$$N(A, \omega) = \frac{4 \cdot a}{\pi \cdot b}, \quad (2.3)$$

where $a$ and $b$ are defined in figure 2.5.

If hysteresis $h$ is introduced in a relay, the describing function is than given by

$$N(A, \omega) = \frac{4a}{\pi b} \left( \sqrt{1 - \left( \frac{h}{b} \right)^2} - i \frac{h}{b} \right). \quad (2.4)$$

If it is assumed that the reference value is zero and that there are oscillations with amplitude $A$ and frequency $\omega_u$ in the system, output $y(t)$ can be written as

$$y = e \cdot N(A, \omega_u) \cdot G(j \omega_u)$$
and with the reference value set to zero, this can be rewritten as

\[ y = -y \cdot N(A, \omega_n) \cdot G(j \omega_n) \]

which gives

\[ N(A, \omega_n) \cdot G(j \omega_n) = -1 \Leftrightarrow G(j \omega_n) = -\frac{1}{N(A, \omega_n)} \quad (2.5) \]

This means that the intersections between \( G(j \omega_n) \) and \(-\frac{1}{N(A, \omega_n)} \) gives the ultimate frequency, \( \omega_n \), and amplitude, \( A \), for the oscillation. Absolute value of \( G(j \omega_n) \) is called ultimate gain and can be written, using a relay’s describing function, as \( K_u = \frac{\pi \cdot b}{4 \cdot a} \) [3, 6].

### 2.3 Amigo Design Method

Instead of using two parameters to calculate controller parameters as in Ziegler-Nichols methods, there is a method called AMIGO, proposed in [5], that uses three parameters \( K_p, T \) and \( L \) that correspond to the parameters of a first order model with time delay

\[ Y_u(s) = K_p \frac{e^{-st}}{1 + sT} U(s). \]

The AMIGO method suggests that the controller parameters for a PI-controller should be calculated as follow:

\[ K_c = \frac{0.15}{K_p} + \left( 0.35 - \frac{LT}{(L + T)^2} \right) \frac{T}{K_p L} \]

\[ T_i = 0.35L + \frac{13LT^2}{T^2 + 12LT + 7L^2} \quad (2.6) \]

PID controller parameters are calculated in the following way:
AMIGO design method will not be described in detail in this master thesis. For those interested in AMIGO, a detailed description can be found in [5].

### 2.4 The Gauss-Newton Optimization Method

Today, there exist many optimization methods that are used to find a minimum of a function of several variables. Multi-dimensional optimization is difficult because several local minima may exist and neither of them might be a global one. It is only in simple cases that it is possible to determine minimum exactly and that is why one has to resort to numerical methods. These numerical methods consist of some scheme for wandering around in the domain of the function to be minimized, and trying to find smaller and smaller function values [1].

There is no general optimization method that will solve all problems because most of the methods are designed to work in certain applications and might be useless in other applications.

Since one of our problems is to minimize a loss function given by a quadratic function, a modified Newton’s method, so called the Gauss Newton method is suggested for this kind of problems. That is why this method has been chosen, described and implemented in this master thesis.

#### 2.4.1 The Newton’s Method

Assume that the function of \( n \) variables \( f \) is a class \( C^2 \). Then, by Taylor’s formula, near a point \( x_k \), we can approximate \( f(x) \) by the quadratic function
where \( H(x_k) \) denotes the Hessian of \( f \) at \( x_k \). If this matrix is positive definite, this Taylor expansion has a minimum where \( \nabla q(x) = 0 \). By differentiation of \( q \) we obtain

\[
\nabla q(x) = 0 \\
\iff \\
\nabla f(x_k) + H(x_k)(x - x_k) = 0 \\
\iff \\
x = x_k - H(x_k)^{-1}\nabla f(x_k)
\]

This formula is used as a basis for an iteration step since \( x_k \) is supposed to be close to a minimum point \( \overline{x} \) of function \( f \). This means that the iteration formula is given by

\[
x_{k+1} = x_k - H(x_k)^{-1}\nabla f(x_k)
\]

The iteration of this formula starting from some initial point is defined as Newton’s method with search direction

\[
d_k = -H(x_k)^{-1}\nabla f(x_k)
\]

where the unit step is taken in that direction [1]. It is possible to calculate better step lengths in every iteration step, called line search, which could give better convergence of function variables but this will not be treated here.

### 2.4.2 The Gauss-Newton Method

The Gauss-Newton method is based on the Newton’s method applied to a least squares problem with an approximation. If there is a function \( f \) in a sum of squares

\[
f(x) = \sum_{i=1}^{m} (r_i(x))^2 \quad x \in \mathbb{R}
\]
then the differentiation of $f$ is given by

$$\nabla f(x) = \sum_{i=1}^{m} 2r_i(x) \nabla r_i(x).$$

By introducing the Jacobian of $r = (r_1, \ldots, r_m)^T$, i.e. the $m \times n$-matrix $J(x) = \left( \frac{\partial r_i}{\partial x_k} \right)$, this can be written as

$$\nabla f(x) = 2J(x)^T r(x).$$

Hessian of $f$ is given by

$$H(x) = 2J(x)^T J(x) + 2 \sum_{i=1}^{m} r_i(x) \nabla^2 r_i(x) \quad (2.11)$$

By assuming, by the nature of the problem, that all $r_i(x)$ are close to zero or not to large near the minimum, following approximation can be made

$$H(x) \approx 2J(x)^T J(x). \quad (2.12)$$

This modification of Newton’s method is called Gauss-Newton’s method. With this approximation it is now only necessary to calculate first derivatives of $r_i$.

The iteration scheme for this method is given by

$$x_{k+1} = x_k - H^{-1}(x_k) J(x_k)^T r(x_k) \quad (2.13)$$

For the convergence of this method, it is required that the initial point is sufficiently near the minimum and also that the neglected terms in Hessian are small enough. Line search can also be used in this method by multiplying direction

$$d_k = -H(x_k)^{-1} J(x_k)^T r(x_k) \quad (2.14)$$

with step length $\lambda_k$ that is calculated with line search at every iteration step [1].
Chapter 3

Modelling of First Order Process

To receive good controller parameters based on AMIGO-design method, the process to be controlled is estimated as a first order process and parameters $K$, $T$ and $L$ are calculated. Section 3.1 describes how Gauss-Newton method is used to calculate $K$, $T$ and $L$. In the section 3.2, the implementation of estimation of first order process is treated. Finally, a result that proves that the implementation works is presented in section 3.3.

3.1 Estimation of First Order Process

To estimate higher order process as a first order process with time delay, a model in Matlab was developed using Gauss-Newton minimization method.

The first order model, i.e. the estimation of higher order models, is given by

$$Y_m(s) = K_p \frac{e^{-Ls}}{1 + sT} U(s)$$

where $K_p$ is gain, $L$ is time delay and $T$ is time constant. Estimation of an unknown process, as a first order process is done by minimizing a loss function given by

$$J = \frac{1}{2} \int_0^T (y(t) - y_m(t))^2 dt$$
where \( y(t) \) is the process output and \( y_m(t) \) is the output from the estimated process model. This means that the error between model output and process output is minimized.

The structure for Gauss-Newton algorithm for minimizing the loss function is made in the following order:

1. Starting guessing parameters \( K_p, T \) and \( L \), i.e. 
2. Calculating \( J, \frac{\partial J}{\partial \theta} \) and \( \frac{\partial^2 J}{\partial \theta^2} \).
3. Updating the value of \( \theta \)

where \( \theta = [K \ T \ L]^T \).

First, a function evaluation has to be done and then, the gradients are used for minimizing the loss function. A general expression for the loss function can be written as

\[
J(\theta) = \frac{1}{2} \int_0^T (y(t) - y_m(t))^2 \, dt.
\]

The derivative of \( J \) with respect to \( \theta \) is then calculated with the following expression:

\[
\frac{\partial J}{\partial \theta} = -\int_0^T (y(t) - y_m(t)) \left( \frac{\partial y_m(t)}{\partial \theta} \right) dt
\]

where the derivatives \( \frac{\partial y_m(t)}{\partial \theta} \) with respect to \( K_p, T \) and \( L \) in s-plane are

\[
\frac{\partial Y_m(s)}{\partial K_p} = \frac{1}{1 + sT} e^{-sL} U(s) = \frac{1}{K_p} Y_m(s)
\]

\[
\frac{\partial Y_m(s)}{\partial T} = -\frac{K_p \cdot s}{(1 + sT)^2} e^{-sL} U(s) = \frac{-s}{1 + sT} Y_m(s)
\]

\[
\frac{\partial Y_m(s)}{\partial L} = -\frac{K_p \cdot s}{1 + sT} e^{-sL} U(s) = -s Y_m(s)
\]
Every step in the minimization of the loss function $J$ includes new calculations of $\frac{\partial J}{\partial \theta}$ and $\frac{\partial^2 J}{\partial \theta^2}$ with value on $\theta$, where $\theta$ is calculated in an update law. The update law is obtained from the Taylor expression of $J$ with some initial value $\theta_0$ on $\theta$:

$$J = J(\theta_0) + \frac{\partial J^T}{\partial \theta}(\theta - \theta_0) + \frac{1}{2}(\theta - \theta_0)^T \frac{\partial^2 J}{\partial \theta^2} (\theta - \theta_0)$$

(3.1)

$$\frac{\partial J^T}{\partial \theta} + \frac{\partial^2 J}{\partial \theta^2} (\theta - \theta_0) = 0$$

This can be written as an update law for $\theta$ where $\theta_k$ is $\theta_0$ in the first step:

$$\theta_{k+1} = \theta_k - \left( \frac{\partial^2 J}{\partial \theta_k^2} \right)^{-1} \left( \frac{\partial J^T}{\partial \theta_k} \right)$$

(3.2)

The second derivative that is included in the update law is given by

$$\frac{\partial^2 J}{\partial \theta^2} = \int_0^T (y(t) - y_m(t)) \left( - \frac{\partial^2 y_m}{\partial \theta^2} \right) dt + \int \left( \frac{\partial y_m}{\partial \theta} \right)^T \left( \frac{\partial y_m}{\partial \theta} \right) dt \approx \int_0^T \left( \frac{\partial y_m}{\partial \theta} \right)^2 dt \geq 0$$

where $(y(t) - y_m(t))$ in the expression goes towards zero and thus can be neglected according to Gauss Newton algorithm. Hence the approximation can be made.

Since $\frac{\partial y_m(s)}{\partial L}$ has a factor $s$ in its expression, it is hard to implement it in the MATLAB® since MATLAB®'s Simulink has no block that supports functions with only zeros and no poles. To make the implementation of $\frac{\partial J}{\partial \theta}$ in MATLAB® easy, the expression of it is made in state space form.

Two state variables $x_1$ and $x_2$ are introduced as
Chapter 3. Modelling of First Order Process

\[ x_1 = \frac{K_p}{1 + sT} U_d \]
\[ x_2 = \frac{K_p}{(1 + sT)^2} U_d \]  \hspace{1cm} (3.3)

where \( U_d = e^{-sd} U_d \)

This now gives a system in the state space form:

\[ (1 + sT)x_1 = K_p \cdot U_d \Rightarrow \frac{dx_1}{dt} = \frac{1}{T} (-x_1 + K_p u_d) \]

\[ (1 + sT)x_2 = x_1 \Rightarrow \frac{dx_2}{dt} = \frac{1}{T} (x_1 - x_2) \]

With this state space expression, \( \frac{\partial J}{\partial K_p}, \frac{\partial J}{\partial T} \) and \( \frac{\partial J}{\partial L} \) can be rewritten in a way that makes it easy to implement in Simulink.

\[ \frac{\partial Y_m}{\partial K_p} = \frac{1}{K_p} x_1 \]
\[ \frac{\partial Y_m}{\partial T} = -\frac{dx_2}{dt} = -\frac{1}{T} (x_1 - x_2) \]
\[ \frac{\partial Y_m}{\partial L} = -\frac{dx_1}{dt} = -\frac{1}{T} (-x_1 + K_p u_d) \]

3.2 Implementation of Estimation

The implementation of the estimation model is implemented, as mentioned earlier, in MATLAB® and its Simulink that is a software package for modeling, simulating and analyzing dynamical systems. To make it easy to verify that the program works properly, we started by estimating a first order process where parameters \( K_p, T \) and \( L \) are initiated by us, i.e. are known. Major part of the program is implemented in Simulink where the first and second derivatives of the loss function are calculated. The Simulink model is executed by an m-file where
Chapter 3. Modelling of First Order Process

the update law is implemented. The schema of the implementation can be seen in figure 3.1.

![Figure 3.1 Schematics of implementation of estimation](image)

When the new values of $K_p$, $T$, and $L$ are calculated, they are used as new values in Simulink model. This can be done while the difference between current parameters and the parameters calculated in the step before exceeds some value initiated by the user. It can also be done in a finite number of steps as it is done in this master thesis where the algorithm is repeated 8 times. The reason why the first alternative was discarded is that it can take very long time until the difference between the parameters gets below the initiated value. The simulation time is chosen by looking at the signals in a relay experiment and chose the time when 4 periods of process output oscillation have occurred. The m-files and the Simulink model with its subsystems can be seen in Appendix A, B and C.

Since we know the values of the process parameters, it is easy to verify that the estimation works, simply by comparing the calculated values and process values. After countless executions of the program, we realized how important the initial values are. It was important to find good values, especially for the parameter $L$. To find these values, we first had to make a Simulink model that was a pure relay experiment. To calculate good initial values with help of the model, we followed a few rules that follows.

The initial gain $K$ could easily be determined simply by looking at the relation between the process output and input. This gain corresponds to gain when the phase is $180^\circ$, i.e. it is smaller than the static gain.

If $G(s)$ is given by

$$G(s) = K_p \frac{e^{-sL}}{1+sT},$$

then $K = K_p e^{-sL}$ and $T = \frac{1}{s}$.
then the absolute value of $G(s)$ is

$$|G| = \left| \frac{K_p}{\sqrt{1 + \omega^2 T^2}} \right|.$$ 

Finally, with equation (2.5), following expression is obtained for initial $K_p$,

$$K = \frac{K_p}{\sqrt{1 + \omega^2 T^2}} = \frac{\pi \cdot b}{4 \cdot a}.$$ 

To determine the time constant $T$ a step was used as an input to our system and the rise time was examined and used as an initial value. Determination of the time delay $L$ is more complicated.

To determine an initial value of $L$, the relation between time delay and time period $T_p$ was investigated. This relation can be expressed in following way with argument of the transfer function $(sG)$:

$$-\arg G(i\omega_0) = \omega_0 L + \arctan \omega_0 T = \pi$$

To determine the interval where parameter $L$ is assumed to have a value, and with facts that $\omega_0 T \geq 0$, i.e that $0 \leq \arctan (\omega_0 T) \leq \pi/2$, two cases are studied.

$$\omega_0 L = \pi \Leftrightarrow L = \frac{\pi}{\omega_0} = \frac{\pi}{2\pi} \cdot T_p = \frac{1}{2} T_p$$

$$\omega_0 L + \frac{\pi}{2} = \pi \Leftrightarrow L = \frac{\pi/2}{\omega_0} = \frac{\pi/2}{2\pi} \cdot T_p = \frac{1}{4} T_p$$

which gives the interval for $L$ as

$$\frac{1}{4} T_p \leq L \leq \frac{1}{2} T_p.$$
This parameter initialization method was done only to verify that the program works correctly and it is not a general method for finding initial values. In this case a step response of the process was investigated to find $T$ and the goal with this thesis is to avoid step response experiments. That is why other methods for finding initial values had to be found.

### 3.3 First Order Results

<table>
<thead>
<tr>
<th>Process Parameters</th>
<th>Initial value</th>
<th>Final value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_p$ = $\frac{2}{(1+4s)}e^{-2s}$</td>
<td>$K$</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>$T$</td>
<td>3.5</td>
</tr>
<tr>
<td></td>
<td>$L$</td>
<td>2.6</td>
</tr>
</tbody>
</table>

![Bode and Nyquist plot showing the “unknown” process and the first order estimation.](image)

**Figure 3.2** Bode and Nyquist plot showing the “unknown” process and the first order estimation.

![Open loop step response of the “unknown” process and its first order estimation.](image)

**Figure 3.3** Open loop step response of the “unknown” process and its first order estimation.

The figures 3.2 and 3.3 show how good estimation of a first order process is obtained. The parameters in the first order process are estimated successfully. The initial values for the optimization algorithm were calculated with rules presented in section 3.2.
Chapter 4

Estimation of Higher Order Processes

This chapter presents the results that confirm the functionality of the implemented process estimator for higher order processes. In section 4.1, the results of the estimation are presented in different plots together with PI- and PID control. In section 4.2, the calculated values of the controller parameters are presented and compared with ones of the step response method.

4.1 High Order Process Results

Since the program for estimation of the first order process worked as wished, the next step was to use the program for estimation of higher order processes. Representing ten different processes, the initial values of parameters for the optimization algorithm were obtained from the Department of Automatic Control in Lund. These values were used at the department to calculate the controller parameters, but in our case, they were used only as initial values since it was assumed that they were not far from the optimal values. This was done to affirm that the Gauss-Newton method works even for more complicated processes and gives acceptable values of the parameters $K$, $T$ and $L$. Applied to AMIGO design rules, these parameters are expected to give a good control.

In the following plots, a wide range of processes of different order are estimated and compared with the estimation. The PI- and PID control of every process is also investigated and compared to control obtained with the values that were used as initial values in the Gauss-Newton method, i.e. the values from the department.
Chapter 4. Estimation of Higher Order Process

\[ G_p = \frac{1}{(1 + 0.1s)^2} e^{-x} \]

<table>
<thead>
<tr>
<th>Process</th>
<th>Parameters</th>
<th>Initial value</th>
<th>Final value</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>1</td>
<td>1.0123</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>0.1864</td>
<td>0.1617</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>1.0282</td>
<td>1.0482</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.1 Bode, Nyquist and step response plot of the open loop systems. The dashed line represents the true process and the solid line represents the estimation.

Figure 4.2 Figures showing PI control (above) and PID control with load disturbance. The dashed line represents the control obtained with the parameters calculated with a step response experiment and the solid line represents the control optimized with Gauss-Newton method.
Chapter 4. Estimation of Higher Order Process

\[ G_p = \frac{1}{(1 + s)^2} e^{-s} \]

<table>
<thead>
<tr>
<th>Process</th>
<th>Parameters</th>
<th>Initial value</th>
<th>Final value</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>1</td>
<td>1.3507</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>1.8643</td>
<td>2.6754</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>1.2817</td>
<td>1.3732</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.3 Bode, Nyquist and step response plot of the open loop systems. The dashed line represents the true process and the solid line represents the estimation.

Figure 4.4 Figures showing PI control (above) and PID control with load disturbance. The dashed line represents the control obtained with the parameters calculated with a step response experiment and the solid line represents the control optimized with Gauss-Newton method.
Chapter 4. Estimation of Higher Order Process

<table>
<thead>
<tr>
<th>Process Parameters</th>
<th>Initial value</th>
<th>Final value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K$</td>
<td>1</td>
<td>2.2231</td>
</tr>
<tr>
<td>$T$</td>
<td>18.6430</td>
<td>62.2210</td>
</tr>
<tr>
<td>$L$</td>
<td>3.8170</td>
<td>4.4476</td>
</tr>
</tbody>
</table>

$$G_p = \frac{1}{(1+10s)^2 e^{-s}}$$

**Figure 4.5** Bode, Nyquist and step response plot of the open loop systems. The dashed line represents the true process and the solid line represents the estimation.

**Figure 4.6** Figures showing PI control (above) and PID control with load disturbance. The dashed line represents the control obtained with the parameters calculated with a step response experiment and the solid line represents the control optimized with Gauss-Newton method.
Chapter 4. Estimation of Higher Order Process

\[
G_p = \frac{1}{(1 + s)(1 + 0.1s)(1 + 0.1^2s)(1 + 0.1^3s)}
\]

<table>
<thead>
<tr>
<th>Process Parameters</th>
<th>Initial value</th>
<th>Final value</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>1</td>
<td>1.4223</td>
</tr>
<tr>
<td>T</td>
<td>1.04</td>
<td>1.8551</td>
</tr>
<tr>
<td>L</td>
<td>0.075</td>
<td>0.0852</td>
</tr>
</tbody>
</table>

**Figure 4.7** Bode, Nyquist and step response plot of the open loop systems. The dashed line represents the true process and the solid line represents the estimation.

**Figure 4.8** Figures showing PI control (above) and PID control with load disturbance. The dashed line represents the control obtained with the parameters calculated with a step response experiment and the solid line represents the control optimized with Gauss-Newton method.
Chapter 4. Estimation of Higher Order Process

\[
G_p = \frac{1}{(1+s)(1 + 0.5s)(1 + 0.5^2s)(1 + 0.5^3s)}
\]

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Initial value</th>
<th>Final value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(K)</td>
<td>1</td>
<td>1.6887</td>
</tr>
<tr>
<td>(T)</td>
<td>1.48</td>
<td>3.3363</td>
</tr>
<tr>
<td>(L)</td>
<td>0.5</td>
<td>0.5912</td>
</tr>
</tbody>
</table>

Figure 4.9 Bode, Nyquist and step response plot of the open loop systems. The dashed line represents the true process and the solid line represents the estimation.

Figure 4.10 Figures showing PI control (above) and PID control with load disturbance. The dashed line represents the control obtained with the parameters calculated with a step response experiment and the solid line represents the control optimized with Gauss-Newton method.
Chapter 4. Estimation of Higher Order Process

<table>
<thead>
<tr>
<th>Process</th>
<th>Parameters</th>
<th>Initial value</th>
<th>Final value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_p = \frac{1}{(1 + s)(1 + 0.005s)^2}$</td>
<td>K</td>
<td>1</td>
<td>0.9088</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>1.005</td>
<td>0.9122</td>
</tr>
<tr>
<td></td>
<td>L</td>
<td>0.0095</td>
<td>0.0096</td>
</tr>
</tbody>
</table>

Figure 4.11 Bode, Nyquist and step response plot of the open loop systems. The dashed line represents the true process and the solid line represents the estimation.

Figure 4.12 Figures showing PI control (above) and PID control with load disturbance. The dashed line represents the control obtained with the parameters calculated with a step response experiment and the solid line represents the control optimized with Gauss-Newton method.
Chapter 4. Estimation of Higher Order Process

\[ G_p = \frac{1}{(1+s)(1+0.1s)^2} \]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Initial Value</th>
<th>Final Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>3</td>
<td>1.4726</td>
</tr>
<tr>
<td>T</td>
<td>1.0662</td>
<td>2.0099</td>
</tr>
<tr>
<td>L</td>
<td>0.1445</td>
<td>0.1598</td>
</tr>
</tbody>
</table>

Figure 4.13 Bode, Nyquist and step response plot of the open loop systems. The dashed line represents the true process and the solid line represents the estimation.

Figure 4.14 Figures showing PI control (above) and PID control with load disturbance. The dashed line represents the control obtained with the parameters calculated with a step response experiment and the solid line represents the control optimized with Gauss-Newton method.
Chapter 4. Estimation of Higher Order Process

\[
G_p = \frac{1}{(1+s)(1+10s)^2}
\]

<table>
<thead>
<tr>
<th>Process Parameters</th>
<th>Initial value</th>
<th>Final value</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>1</td>
<td>0.7726</td>
</tr>
<tr>
<td>T</td>
<td>18.717</td>
<td>21.9609</td>
</tr>
<tr>
<td>L</td>
<td>3.7716</td>
<td>4.4933</td>
</tr>
</tbody>
</table>

Figure 4.15 Bode, Nyquist and step response plot of the open loop systems. The dashed line represents the true process and the solid line represents the estimation.

Figure 4.16 Figures showing PI control (above) and PID control with load disturbance. The dashed line represents the control obtained with the parameters calculated with a step response experiment and the solid line represents the control optimized with Gauss-Newton method.
Chapter 4. Estimation of Higher Order Process

<table>
<thead>
<tr>
<th>Process</th>
<th>Parameters</th>
<th>Initial value</th>
<th>Final value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_p$</td>
<td>K</td>
<td>1</td>
<td>1.5222</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>1.4250</td>
<td>5.4915</td>
</tr>
<tr>
<td></td>
<td>L</td>
<td>1.6292</td>
<td>1.6970</td>
</tr>
</tbody>
</table>

$$G_p = \frac{1}{(1 + s)^4}$$

Figure 4.17 Bode, Nyquist and step response plot of the open loop systems. The dashed line represents the true process and the solid line represents the estimation.

Figure 4.18 Figures showing PI control (above) and PID control with load disturbance. The dashed line represents the control obtained with the parameters calculated with a step response experiment and the solid line represents the control optimized with Gauss-Newton method.
Chapter 4. Estimation of Higher Order Process

<table>
<thead>
<tr>
<th>Process</th>
<th>Parameters</th>
<th>Initial value</th>
<th>Final value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G_p = \frac{1}{(1 + s)^8} )</td>
<td>( K )</td>
<td>1</td>
<td>1.4307</td>
</tr>
<tr>
<td></td>
<td>( T )</td>
<td>4.3300</td>
<td>6.1636</td>
</tr>
<tr>
<td></td>
<td>( L )</td>
<td>4.3100</td>
<td>4.6453</td>
</tr>
</tbody>
</table>

Figure 4.19 Bode, Nyquist and step response plot of the open loop systems. The dashed line represents the true process and the solid line represents the estimation.

Figure 4.20 Figures showing PI control (above) and PID control with load disturbance. The dashed line represents the control obtained with the parameters calculated with a step response experiment and the solid line represents the control optimized with Gauss-Newton method.
As it can be seen in the figures, in almost all cases, there is a good control with the parameters calculated in the Gauss-Newton algorithm. The process estimation works well in the third quadrant of the Nyquist plot. In all cases, there is a load disturbance applied and it can be seen that the calculated controllers work well even in those situations. Calculated controller parameters can be seen in section 4.2.

### 4.2 Control Parameters

<table>
<thead>
<tr>
<th>Process</th>
<th>Method</th>
<th>$K_c$</th>
<th>$T_i$</th>
<th>$K_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_p = \frac{1}{(1+0.1s)^2 e^{-s}}$</td>
<td>Relay</td>
<td>0.184</td>
<td>0.403</td>
<td>0.457</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>0.190</td>
<td>0.408</td>
<td>0.466</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1+s)^2 e^{-s}}$</td>
<td>Relay</td>
<td>0.293</td>
<td>2.46</td>
<td>0.120</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>0.308</td>
<td>1.78</td>
<td>0.173</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1+10s)^2 e^{-s}}$</td>
<td>Relay</td>
<td>1.88</td>
<td>32.1</td>
<td>0.059</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>1.17</td>
<td>14.6</td>
<td>0.081</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1+s)(1+0.005s)^2}$</td>
<td>Relay</td>
<td>35.7</td>
<td>0.114</td>
<td>313</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>36.0</td>
<td>0.114</td>
<td>316</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1+s)(1+0.1s)^2}$</td>
<td>Relay</td>
<td>2.51</td>
<td>1.10</td>
<td>2.28</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>1.96</td>
<td>0.732</td>
<td>2.68</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1+s)(1+10s)^2}$</td>
<td>Relay</td>
<td>1.52</td>
<td>17.2</td>
<td>0.0884</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>1.19</td>
<td>14.6</td>
<td>0.0815</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1+s)(1+0.1s)(1+0.1^2s)(1+0.1^3s)}$</td>
<td>Relay</td>
<td>4.82</td>
<td>0.737</td>
<td>6.54</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>4.13</td>
<td>0.539</td>
<td>7.67</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1+s)(1+0.5s)(1+0.5^2s)(1+0.5^3s)}$</td>
<td>Relay</td>
<td>0.831</td>
<td>2.50</td>
<td>0.332</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>0.627</td>
<td>1.29</td>
<td>0.488</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1+s)^4}$</td>
<td>Relay</td>
<td>0.4592</td>
<td>4.6970</td>
<td>0.098</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>0.2384</td>
<td>1.4575</td>
<td>0.163</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1+s)^6}$</td>
<td>Relay</td>
<td>0.202</td>
<td>5.93</td>
<td>0.0341</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>0.251</td>
<td>4.33</td>
<td>0.0580</td>
</tr>
</tbody>
</table>

Table 4.1 Calculated PI-controller parameters and integral gain $K_i$ in all ten cases.

Table 4.1 shows the calculated PI-controller values. As it can be seen, the optimized parameter values, denoted as “Relay” are not far away from the ones calculated with the values from the department of automatic control, denoted as SR, which stands for Step Response. In section 4.1, it is shown that the control is good with “Relay” values. The integral gain is also calculated and the values shows that the “Relay” method has lower integral gain in all cases except one,
but they are close to the values of the SR-method. This confirms that the control obtained with the values optimized in Gauss-Newton algorithm is satisfying.

The values for the PID controller can be seen in table 4.2.

<table>
<thead>
<tr>
<th>Process</th>
<th>Method</th>
<th>$K_c$</th>
<th>$T_i$</th>
<th>$T_d$</th>
<th>$K_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_p = \frac{1}{(1 + 0.1s)^2} e^{-s}$</td>
<td>Relay</td>
<td>0.2661</td>
<td>0.5403</td>
<td>0.1780</td>
<td>0.493</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>0.2816</td>
<td>0.5504</td>
<td>0.1936</td>
<td>0.513</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1 + s)^2} e^{-s}$</td>
<td>Relay</td>
<td>0.797</td>
<td>2.25</td>
<td>0.600</td>
<td>0.354</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>0.855</td>
<td>1.75</td>
<td>0.531</td>
<td>0.489</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1 + 10s)^2} e^{-s}$</td>
<td>Relay</td>
<td>2.9218</td>
<td>21.4907</td>
<td>2.1771</td>
<td>0.136</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>2.3979</td>
<td>11.0461</td>
<td>1.7981</td>
<td>0.216</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1 + s)(1 + 0.005s)^2}$</td>
<td>Relay</td>
<td>47.2704</td>
<td>0.0699</td>
<td>0.0048</td>
<td>676</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>47.5921</td>
<td>0.0697</td>
<td>0.0047</td>
<td>680</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1 + s)(1 + 0.1s)^2}$</td>
<td>Relay</td>
<td>3.98</td>
<td>0.741</td>
<td>0.0780</td>
<td>5.37</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>3.52</td>
<td>0.524</td>
<td>0.0694</td>
<td>6.72</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1 + s)(1 + 10s)^2}$</td>
<td>Relay</td>
<td>3.12</td>
<td>13.0</td>
<td>2.12</td>
<td>0.239</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>2.43</td>
<td>11.0</td>
<td>1.78</td>
<td>0.221</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1 + s)(1 + 0.1s)(1 + 0.1^2s)(1 + 10^2s)}$</td>
<td>Relay</td>
<td>7.0295</td>
<td>0.4778</td>
<td>0.0420</td>
<td>14.7</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>6.4400</td>
<td>0.3612</td>
<td>0.0367</td>
<td>17.8</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1 + s)(1 + 0.5s)(1 + 0.5^2s)(1 + 0.5^3s)}$</td>
<td>Relay</td>
<td>1.62</td>
<td>1.86</td>
<td>0.281</td>
<td>0.871</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>1.53</td>
<td>1.07</td>
<td>0.227</td>
<td>1.43</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1 + s)^4}$</td>
<td>Relay</td>
<td>1.09</td>
<td>3.83</td>
<td>0.777</td>
<td>0.285</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>0.594</td>
<td>1.65</td>
<td>0.607</td>
<td>0.360</td>
</tr>
<tr>
<td>$G_p = \frac{1}{(1 + s)^8}$</td>
<td>Relay</td>
<td>0.557</td>
<td>5.99</td>
<td>1.89</td>
<td>0.0930</td>
</tr>
<tr>
<td></td>
<td>SR</td>
<td>0.652</td>
<td>4.71</td>
<td>1.66</td>
<td>0.138</td>
</tr>
</tbody>
</table>

Table 4.2 Calculated PID-controller parameters and integral gain $K_i$ in all ten cases.

As it can be seen in table 4.2, the same results are obtained with PID control as in PI control. In all cases except for one, when the sixth process from above is estimated, the integral gain is lower. This means that the area between the reference value and the process output is higher, i.e. the same result is obtained as in the case with the PI-control.
Chapter 5

Method for the Initiation of $K_p$, $T$ and $L$

The method of initialization of parameters $K_p$, $T$ and $L$, in this master thesis referred as backward AMIGO is presented in section 5.1. In section 5.2, the results of PID control received with parameters calculated with method presented in 5.1 is compared with control obtained with the step response method.

5.1 Backward AMIGO Method

As mentioned in chapter 3.2 and shown in chapter 4, the Gauss-Newton’s method needs good initial values of parameters $K_p$, $T$ and $L$. For this reason, it is important to find a method that gives satisfying initial parameters in the general case. These three parameters have to be obtained from a simple relay experiment by looking at interesting parts of the input and output signal.

The delay parameter $L$ was assumed to be a critical parameter, since it was noticed in previous tests that it was important to have a very good approximation of it. A small change in the value of $L$ could lead to completely different results. By investigating relay experiment signals from all tested processes in chapter 4 and looking at obtained approximated values of $L$ in those tests, it was noticed that a very good approximation of $L$ can be obtained directly from a relay experiment as figure 5.1 shows.
Chapter 5. Method for the Initiation of $K_p$, $T$ and $L$

The parameters $K_p$ and $T$ could not be obtained in the same way as the parameter $L$. Because of that, a method for finding these parameters was developed.

The Ziegler-Nichols frequency method gives fairly satisfying controller parameters. Since the Gauss-Newton method needs fairly good initial values that converge to optimal parameters, it is assumed that the controller parameters obtained with these optimal parameters will be relatively near to the parameters obtained with Ziegler-Nichols frequency method. The idea is to use fairly good controller parameters and then go backwards in AMIGO design rules to obtain initial values for the optimization algorithm. By rewriting the AMIGO rules (2.7) for PID controller with respect to the process parameters, following relations are calculated:

\[
L = \frac{2.5 \cdot T_i + 0.8 \cdot T_d}{2} - \sqrt{\left(\frac{2.5 \cdot T_i + 0.8 \cdot T_d}{2}\right)^2 - 4.85 \cdot T_d \cdot T_i}
\]

\[
T = \frac{0.3 \cdot T_d \cdot L}{0.5 \cdot L - T_d}
\]

\[
K_p = \frac{1}{K_c} \left(0.2 + 0.45 \frac{T}{L}\right)
\]

By using the Ziegler-Nichols frequency method, parameters $K_c$, $T_i$ and $T_d$ are calculated and applied to (5.1) to obtain initial values of $K_p$, $T$. Notice that $L$ in (5.1) is only used in calculation of $K$ and $T$ and is not used as initial value since good initial value of it is obtained as mentioned earlier.

Instead of using Ziegler-Nichols method and its modification to find a fairly good controller for the backward AMIGO, another method could be used as proposed in [5]. Following relations are suggested:
Chapter 5. Method for the Initiation of $K_p$, $T$, and $L$

\[ K_c = \frac{0.3 - 0.1\kappa^4}{K_{180}} \]
\[ T_i = \frac{0.6}{1 + 2\kappa} T_{180} \]
\[ T_d = \frac{0.15(1 - \kappa)}{1 - 0.95\kappa} T_{180} \]  

(5.2)

where $\kappa$ is the gain ratio between $K_{180}$ and $K_c$.

The gain ratio is chosen in figure (5.2) where the upper right plot shows $K, K_{180}$ vs $\kappa$. As it can be seen, there is a linear region up to $\kappa \approx 0.4$ and this inserted in (5.2) leads to following relations:

\[ K_c = 0.3K_u \]
\[ T_i = 0.33T_u \]
\[ T_d = 0.14T_u \]  

(5.3)

To avoid imaginary values in the expression for the parameter $L$ in (5.1), the factor of 0.33 in (5.3) was set a little higher. The expression of $K_c$ was also changed a little bit to get higher values of it so the value of $K$ in (5.1) decreases little. The new values are
Chapter 5. Method for the Initiation of $K_p$, $T$ and $L$

\[ K_r = 0.35K_u \]
\[ T_i = 0.4T_u \]
\[ T_d = 0.14T_u \]  \hspace{1cm} (5.4)

In this master thesis, these rules are used to calculate new parameters for the backward AMIGO since it was noticed that the Zigler-Nichols method gave poor initial values.

5.2 Results of the Backwards AMIGO Method

The results showing the values obtained with help of backward AMIGO method are presented in table 5.1

<table>
<thead>
<tr>
<th>Process</th>
<th>Parameter</th>
<th>$K_p$</th>
<th>$T$</th>
<th>$L$</th>
<th>$G_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initial</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>$1/(1+0.1s)^2$</td>
</tr>
<tr>
<td></td>
<td>Final</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>$e^{-s}$</td>
</tr>
<tr>
<td></td>
<td>Initial</td>
<td>4.33</td>
<td>1.92</td>
<td>1.15</td>
<td>$1/(1+s)^2$</td>
</tr>
<tr>
<td></td>
<td>Final</td>
<td>1.35</td>
<td>2.67</td>
<td>1.38</td>
<td>$e^{-s}$</td>
</tr>
<tr>
<td></td>
<td>Initial</td>
<td>13.0</td>
<td>9.71</td>
<td>4.57</td>
<td>$1/(1+10s)^2$</td>
</tr>
<tr>
<td></td>
<td>Final</td>
<td>1.74</td>
<td>48.57</td>
<td>4.61</td>
<td>$e^{-s}$</td>
</tr>
<tr>
<td></td>
<td>Initial</td>
<td>23.16</td>
<td>0.161</td>
<td>0.0098</td>
<td>$1/(1+s)(1+0.005s)^2$</td>
</tr>
<tr>
<td></td>
<td>Final</td>
<td>0.84</td>
<td>0.85</td>
<td>0.0097</td>
<td>$e^{-s}$</td>
</tr>
<tr>
<td></td>
<td>Initial</td>
<td>14.33</td>
<td>0.403</td>
<td>0.138</td>
<td>$1/(1+s)(1+0.1s)^2$</td>
</tr>
<tr>
<td></td>
<td>Final</td>
<td>1.18</td>
<td>1.61</td>
<td>0.165</td>
<td>$e^{-s}$</td>
</tr>
<tr>
<td></td>
<td>Initial</td>
<td>12.41</td>
<td>9.91</td>
<td>3.95</td>
<td>$1/(1+s)(1+10s)^2$</td>
</tr>
<tr>
<td></td>
<td>Final</td>
<td>1.63</td>
<td>47.0</td>
<td>4.38</td>
<td>$e^{-s}$</td>
</tr>
<tr>
<td></td>
<td>Initial</td>
<td>18.46</td>
<td>0.292</td>
<td>0.0669</td>
<td>$1/(1+s)(1+0.1s)(1+0.1^2s)(1+0.1^3s)$</td>
</tr>
<tr>
<td></td>
<td>Final</td>
<td>1.35</td>
<td>1.76</td>
<td>0.0854</td>
<td>$e^{-s}$</td>
</tr>
<tr>
<td></td>
<td>Initial</td>
<td>8.05</td>
<td>1.05</td>
<td>0.488</td>
<td>$1/(1+s)(1+0.5s)(1+0.5^2s)(1+0.5^3s)$</td>
</tr>
<tr>
<td></td>
<td>Final</td>
<td>2.06</td>
<td>4.09</td>
<td>0.576</td>
<td>$e^{-s}$</td>
</tr>
<tr>
<td></td>
<td>Initial</td>
<td>5.85</td>
<td>2.66</td>
<td>1.60</td>
<td>$1/(1+s)^4$</td>
</tr>
<tr>
<td></td>
<td>Final</td>
<td>1.57</td>
<td>5.67</td>
<td>1.71</td>
<td>$e^{-s}$</td>
</tr>
<tr>
<td></td>
<td>Initial</td>
<td>3.34</td>
<td>5.88</td>
<td>3.87</td>
<td>$1/(1+s)^8$</td>
</tr>
<tr>
<td></td>
<td>Final</td>
<td>1.30</td>
<td>5.54</td>
<td>4.67</td>
<td>$e^{-s}$</td>
</tr>
</tbody>
</table>

Table 5.1 Ten process estimations made with the backward AMIGO method.

The table 5.1 shows that the initial values are not so close to the final values but the algorithm generates very good values that are close to the values in section
4.1. As it is mentioned earlier, good initial values are necessary in the Gauss-Newton algorithm.

The explanation to the large difference between the values can be that the backward AMIGO method generates a right combination of the parameters and since the parameters are dependant on each other, the right combination of them can lead to good final values. Unfortunately, the first process could not be estimated with success since the initial values, obviously were not good enough. It could not be determined why the initiation of this process parameters didn’t work, this could possibly be avoided if the line search (see section 6.2) is implemented. Despite the problems with the first process, other estimations give satisfying results which means that it can be concluded that there seems to be enough information in a relay experiment to make a good estimation of processes with no need of step response experiment.
Chapter 6

Real Process Test

In this chapter, the method of estimation of an unknown process is tested on a real process. In section 6.1 the description and estimation of the process is done. In section 6.2, the obtained control with AMIGO design is compared to the ECA600 industrial controller.

6.1 Estimation of a Tank Process

The real process to be tested is a tank process consisting of two cascaded tanks as seen in figure 6.1. In the test, only the upper tank is used and the control is later done only on the water level in this tank. The water level is measured in 0-10 V where empty and full tank is represented by 0 Volt and 10 Volt respectively.

Figure 6.1 The tank process
Chapter 6. Real Process Test

The short relay experiment was done consisting of four periods on the tank process. This data was then used in our Simulink model to make an estimation of the process and the following estimation was obtained

\[
G(s) = 0.72 \frac{e^{-1.6s}}{1 + 9.7s}
\]

The signals from relay, real process and its estimation can be seen in figure 6.2.

![Figure 6.2 The relay experiment signals. The solid line represents the tank process and the dashed line represents the estimation](image)

When the estimation was obtained, a step response experiment was made to see how good the estimation is by comparing it to the real process step response. The result can be seen in figure 6.3.

![Figure 6.3 The step responses of the tank process and its estimation. The solid line represents the real process and the dashed line represents the estimation.](image)
As it is seen, the same results are obtained as in chapter 4 when ten different processes were estimated. The estimation of the static gain is not correct but the estimation follows the process well at high frequencies.

### 6.2 Control of a Tank Process

To see how good the obtained control is using the AMIGO design, the ECA600 controller was used as a comparison. The ECA600 is a serial PI/PID-controller that uses relay experiment for auto tuning. It is an advanced controller that does not estimate any first order model with time delay of the process to be controlled. Instead, it uses other methods to obtain the controller parameters presented in [7].

When the auto tuning with ECA600 was done, the obtained controller was in PI form. To make a fair comparison, the AMIGO design for PI control, see (2.6), was used to calculate a PI controller parameters. Later, the comparison between the parameters done using a PI controller in Simulink. The values of ECA600 controller and controller using AMIGO are presented in table 6.1.

<table>
<thead>
<tr>
<th>Process</th>
<th>Parameters</th>
<th>ECA600</th>
<th>AMIGO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tank process</td>
<td>$K_c$</td>
<td>9.46</td>
<td>2.13</td>
</tr>
<tr>
<td></td>
<td>$T_i$</td>
<td>3.0</td>
<td>7.24</td>
</tr>
</tbody>
</table>

Table 6.1 Controller parameters of ECA600 controller and AMIGO

To compare the controllers, a step was applied as reference value to the tank process and, at approximately 200 seconds, a load disturbance was added. The load disturbance was made by opening a valve that can bee seen in figure 6.1 beside the lower tank. The results are shown in figures 6.4 and 6.5.

![Figure 6.4 Step response and load disturbance using a controller with ECA600 parameters](image)
Chapter 6. Real Process Test

Figure 6.5 Step response and load disturbance with controller obtained with AMIGO.

Note that the timescale starts at 50 seconds since the water level had to stabilise first before a step change in reference value, to obtain nice plots. As expected, the controller parameters calculated by ECA600 give a faster response to the changes in reference value due to higher controller gain. Comparing the controllers, only looking at the changes in reference value, it seems that the controller with ECA600 parameters has a better control but to a certain cost. When looking at the control signals shown in figure 6.6 and 6.7, it can be seen that AMIGO controller generates a lower signal.

Figure 6.6 Control signal from a controller using ECA600 parameters. The straight line in figure represents the maximal signal value.

Figure 6.7 Control signal from a controller obtained with AMIGO design method. The straight line in figure represents the maximal signal value.
6.3 Comment

As shown in previous sections, the control obtained with AMIGO design is satisfying. However, there are few comments on the experiment.

Since only the control with control parameters in table 6.1 are compared, the reference value change of an ECA600 controller shown in figure 6.4 is not completely correct. In reality, there are filters and other improvements in ECA600 that makes the reference value change smoother.

It was noticed that the working point of ECA600 was at higher frequency when the relay experiment was done. If our working point was at even higher frequency, maybe a better estimation could be done.

The results are however satisfying even if the process was a very simple one. We believe that the method of estimation presented in this master thesis is more suitable for higher order processes.
Chapter 7

Conclusions

7.1 Summary

The aim of this thesis was to investigate if it is possible to approximate an unknown process as a first order process with delay only by obtaining information from a relay experiment. If there is a satisfying first order process approximation, there are rules, called AMIGO rules, that gives good PI-/PID controller parameters. By only using the relay experiment, many disadvantages with a step response experiment used in many commercial automatic tuners based on relay feedback, could be avoided. The algorithm used for the optimization of the approximated process parameters is the Gauss-Newton method. To determine the initial values of $K_p$, $T$ and $L$ for the optimization algorithm, the backward AMIGO design rules are used. The controller parameters $K_c$, $T_i$ and $T_d$ used in backward AMIGO are obtained with modified Ziegler-Nichols update rules.

It is difficult to say if it is completely possible to do the approximation well for wide range of processes, but in the tested processes it is shown that 9 of 10 process estimations work well. Although the 10 tested processes are carefully chosen to cover a wide range of process varieties, there have to be done more tests to confirm that.
7.2 Future Work

There are a lot of things that could possibly improve the process of finding the initial values and optimizing them. We had a lot of ideas that could work but due to lack of time, i.e. time range of the master thesis, all the ideas could not be evaluated. Some of them and few other, not implemented, ideas are:

- Introduce a method in Gauss-Newton algorithm, called *line search*, which optimizes the step length in the algorithm where the parameters are updated. This could make the algorithm less sensitive to bad initial values.
- Since a very good estimation of the parameter $L$ can be made, the first order process that approximates the unknown process can be seen as a process with only two unknown parameters, $K_p$ and $T$. It may be easier to find better estimations of these two parameters if the algorithm optimizes only them instead of all three parameters as it is done in this thesis.
Appendix B

The relay feedback with real process

The estimation in state space form
Appendix C

m-file intialvaluegenerator.m for calculation of the initial parameters

a=0.05; %relay output
b=0.0359; %relay input
time=68; %simulation time
Tperiod=15.9; %period time of relay input
L=3.87; %observed initial L

% Calculation of the control parameters
K_0=0.35*((pi*b)/(4*a));
Ti=0.40*Tperiod;
Td=0.14*Tperiod;

% Calculation of the initial parameters using backwards AMIGO
const=((2.5*Ti+0.8*Td)/2);
L1=const-sqrt(const^2-4.85*Td*Ti);
T=(0.3*Td*L1)/(0.5*L1-Td);
K=(1/K_0)*(0.2+0.45*T/L1);

m-file paramitr.m for iteration of the parameters

% Initialization of the process parameters
x=[K,T,L]'; %K,T,L calculated in intialvaluegenerator.m

% Iteration of the parameters
%==================================================================
for i=1:8
    sim('model1',[0 time]);
    n1=size(Jx1);
    n1=n1(1);
    Jx1=Jx1(n1);
    n2=size(Jx2);
    Jx2=Jx2(n2(1));
    n3=size(Jx3);
    Jx3=Jx3(n3(1));
    n11=size(Jxx11);
```
Jxx11=Jxx11(n11(1));

n12=size(Jxx12);
Jxx12=Jxx12(n12(1));
n13=size(Jxx13);
Jxx13=Jxx13(n13(1));

n22=size(Jxx22);
Jxx22=Jxx22(n22(1));

n23=size(Jxx23);
Jxx23=Jxx23(n23(1));

n33=size(Jxx33);
Jxx33=Jxx33(n33(1));

Jx=[Jx1 Jx2 Jx3]';
Jxx=[Jxx11 Jxx12 Jxx13 ; Jxx12 Jxx22 Jxx23 ; Jxx13 Jxx23 Jxx33];

x=x-Jxx\Jx;  %updating the parameters
K=x(1);
T=x(2);
L=x(3);
end

SimulationBodNyqStep  %plotting of bode, nyquist and stepresponse

m-file SimulationBodNyqStep.m for plotting

hold off;
cf;
s=tf('s');

%Initialization of the parameters of the proces to be estimated
a=0.5;
T0=10;
n=8;

process=3;

%First order model
systm=K/(1+s*T);
set(systm,'InputDelay',L);

%Different tested processes
if process==1
    systp=K0/(1+s*T0);
    set(systp,'InputDelay',L0);
elseif process==2
    systp=1/(1+s*T0)^2;
set(systp,'InputDelay',1)
elseif process==3
    systp=1/((1+s)*(1+s*T0)^2);
elseif process==4
    systp=1/((1+s)*(1+a*s)*(1+a^2*s)*(1+a^3*s));
elseif process==5
    systp=1/(s+1)^n;
end

% Plot of step response
figure(1);
t=0:0.1:300;
yp=plot(t,step(systm,t));
axis([0 10 -0.005 1.5]);
set(yp,'Linewidth',1);
xlabel('Time','FontSize',10);
ylabel('Amplitude','FontSize',10);
hold on

t=0:0.1:300;
Yp=plot(t,step(systp,t),'-.');
axis([0 10 -0.005 1.5]);
set(Yp,'Linewidth',1);
set(Yp,'Color','mag');
xlabel('Time','FontSize',10);
ylabel('Amplitude','FontSize',10);
hold off;

%Plot of Nyquist
figure(2);
[re,im,w]=nyquist(systm,logspace(-3, 4, 10000));
yp=plot(re(:),im(:));
set(yp,'Linewidth',1);
hold on
axis([-0.1 1 -0.55 0.11])
plot([-2 2.4],[0 0])
plot([0 0],[-3 1.1])
yp=plot(0.15,-0.98,'k<');
set(yp,'Linewidth',1);
[mag,phase,w]=bode(systm);
hold on;

[Re,lm,W]=nyquist(systp,logspace(-3, 4, 10000));
Yp=plot(Re(:),lm(:),'-.');
set(Yp,'Linewidth',1);
set(Yp,'Color','mag');
hold on
axis([-0.1 1 -0.55 0.11])
plot([-2 2.4],[0 0])
plot([0 0],[-3 1.1])
Yp=plot(0.15,-0.98,'k<');
set(Yp,'Linewidth',1);
set(Yp,'Color','mag');
[Mag,Phase,W]=bode(systp);
hold off;

%Plot of Bodediagram
figure(3)
subplot(2,1,1);
yp=loglog(w,mag(:));
set(yp,'Linewidth',1);
ylabel('Magnitude','FontSize',10);
tmpa=gca;
set(tmpa,'Ytick',[0.01 0.1 1]);
set(tmpa,'Xtick',[0.1 1 10 100]);
axis([0 100 0.1 2]);
hold on
Yp=loglog(W,Mag(:),'-.');
set(Yp,'Linewidth',1);
set(Yp,'Color','mag');
ylabel('Magnitude','FontSize',10);
Tmpa=Gca;
set(Tmpa,'Ytick',[0.01 0.1 1]);
set(Tmpa,'Xtick',[0.1 1 10 100]);
axis([0 100 0.1 2]);
hold off

subplot(2,1,2);
yp=semilogx(w,phase(:));
axis([0 100 -200 0]);
set(yp,'Linewidth',1);
xlabel('Frequency','FontSize',10);
ylabel('Phase','FontSize',10);
tmpa=gca;
set(tmpa,'Xtick',[0.1 1 10 100]);
hold on
Yp=semilogx(W,Phase(:),'-.');
axis([0 100 -200 0]);
set(Yp,'Linewidth',1);
set(Yp,'Color','mag');
Tmpa=Gca;
set(Tmpa,'Xtick',[0.1 1 10 100]);
xlabel('Frequency','FontSize',10);
ylabel('Phase','FontSize',10);
hold off
Bibliography


