A new, fast, and efficient automatic tuner for the ABB AC 800M family of controllers

Magnus Lundh

LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6146
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2021

Abstract

The most common controller in the process industry today is the PID controller due to its simple structure. A regular factory may have thousands of different PID controllers in use where each of them has to be tuned to work satisfactory. Doing this manually can become both a difficult and time consuming task where some knowledge for each process is required. Therefore, it is highly beneficial to have a way to automatically tune the controllers, which was the motivation behind the relay autotuner first introduced in the 1980's. Since then, many alterations of the original relay autotuner have been proposed as technology development have progressed, concerning both available computing power as well as PID controllers in general.

This master thesis is proposing a new autotuner, based on a short asymmetrical relay experiment, that is running at sufficiently low cost to be executed directly in an industrial controller unit. This is highly beneficial for end customers who are to control a process with unknown dynamics.

An extensive simulation study is conducted using a test batch of a wide range of processes representative for the process industry to ensure that the approach is adequate. A suitable first order model with time delay (FOTD) is found from which a controller structure is based, following a set of well-known tuning rules.

The method is then implemented in the ABB Ability™ System 800xA where the proposed method was shown to successfully estimate a model and control a process with unknown dynamics in a robust way.

Acknowledgements

I would like to extend my deepest gratitude to my supervisors Tore Hägglund and Kristian Soltesz at the department of Automatic Control and Alfred Theorin at ABB Process Automation for guiding me through this project. Your knowledge on the subject has been an inexhaustible source of information and has lead to many interesting and insightful discussions throughout the extent of this project. The completion of this study would not have been possible without your expertise and constant eagerness to help.

To my family, thank you for your constant love and support. Thank you for always inspiring me and encouraging me to achieve the best I can. In particular, I would like to thank my father for all your support this past semester. With your expertise in the field, you have been able to give me valuable input and suggestions on new things to try out which has been of great help to me toward the completion of this thesis.

Contents

Notation

Notation	Description
C	Controller transfer function
e	Control error
d	Load disturbance
K_p	Static gain of process
K_c	Proportional gain of controller
L	Time delay of process
n	Measurement noise
P	Process transfer function
r	Reference value, setpoint
T	Time constant of process
T_d	Derivative time of PID controller
T_f	Filter time constant
T_i	Integral time of PID controller
τ	Normalized time delay
u	Control signal, relay output
y	Process value

Abbrevation	Description
ARX	AutoRegressive with eXogenous input model
BJ	Box-Jenkins model
FOTD	First order model with time delay
LS	Least squares
N4SID	Numerical algorithms for subspace state space system identification
OE	Output-error model
PAC	Programmable automation controller
PI	Proportional integral (controller)
PID	Proportional integral derivative (controller)
RMSE	Root mean square error
SISO	Single-input single-output (system)
SOTD	Second order model with time delay
SOZTD	Second order model with zero and time delay
ZOH	Zero-order hold (discretization)

1

Introduction

This thesis introduces a new generation of the autotuner used to find suitable PID-parameters to control a process with unknown dynamics. The work has been done for ABB Process Automation in Malmö Sweden.

1.1 Motivation

The PID controller is by far the most used controller in the process industry with more than 95% of all controllers being of this type [Åström and Hägglund, 1995]. A typical factory can have thousands of PID controllers. Despite being of low complexity, manually tuning these controllers can be a tedious and time-consuming process requiring a lot of knowledge [Berner, 2017]. It is therefore desirable to have a method of doing this automatically with a so-called autotuner.

In the master thesis by [Hansson and Svensson, 2020] a novel autotuner based on the method of [Berner, 2017] was proposed and implemented at ABB. The method that they are suggesting is able to find an appropriate low-order model and from that estimate the proper PID parameters to give the desired process behavior by using only a short experiment.

1.2 Aim of the thesis

The autotuner suggested by [Hansson and Svensson, 2020] requires an external computer to solve a complex optimization problem. Additional computing power allows for much more complex computations to be done. Such computational power is normally not available on the controller level but could be obtained using e.g. industrial PCs. However, [Hansson and Svensson, 2020] highlight a major downside with requiring another device to tune a process, namely the need for a secure connection between the controller and external device. Additionally, requiring another computer also implies a higher cost and more equipment that can fail. This master thesis, therefore, aims to find a strategy that can make use of the results of [Hansson and Svensson, 2020] in a more lightweight manner such that a new autotuner

can be implemented directly in the controller, thus avoiding the additional device altogether.

1.3 Strategy

The approach for this thesis project is as follows. First, a proper model to describe the dynamics of the system must be found. Using only a few switches of an asymmetric relay, the goal is to attain a short but sufficient experiment to properly describe the dynamics of any system sought to be identified. A few different identification methods are examined using simulated data from relay experiments on a large number of different processes. The best model is then chosen based on a few different criteria, including the resemblance of the identified system to the original signal in the time domain, the frequency response, and the complexity of the implementation.

Second, a controller is designed in such a way that it creates a robust closed-loop behavior. This should be done in a straightforward way in order to allow for a re-visit of the design in the future, in case the controller would need any adjustments. Here, a set of well-known tuning methods have been used for simplification. Last, the closed-loop dynamics are examined using load disturbances over a certain period of time in order to see how well the controller is able to return to the desired value.

1.4 Structure

The structure of this thesis is as follows. In Chapter 1, an introduction to this thesis is given, along with the aim and strategy. Chapter 2 includes some preliminaries related to control theory, PID-control, and automatic tuning of controllers as well as give some insight into the ABB control system in order to facilitate understanding for readers without any prior knowledge on the subject. Then, in Chapter 3 the process models used in this thesis are described as well as the benchmarking processes. Here, the experiment design is also described. Chapter 4 explains the model estimation part of this thesis. Three different methods are considered and followed by a motivation for the final choice. Chapter 5 deals with the controller design and evaluation thereof. Chapter 6 describes the implementation in the ABB system. The thesis is then summarized in Chapter 7, where some discussions, conclusions, and major takeaways are presented. An appendix describes the different benchmark processes used.

2

Preliminaries

The purpose of this chapter is to give an introduction to a control system and the PID controller, as well as to provide a basic understanding and give intuition into the concept of automatic tuning.

2.1 Process control

The foundation of process control is about automatically controlling a process based on its current state. Typically, a signal is measured, and based on its value a control signal is computed by a controller. The control signal is then fed to the process. Many processes such as keeping the speed of your car or the temperature of your shower can be controlled by the human where we are constantly evaluating the current state and adjust our control tools accordingly. However, a controller generally does a better job at keeping to an exact level. Moreover, there are processes that are too complex for a human to control. For example, a process with a long dead-time, i.e. where the time between a change in the input to the time it affects the output is long and this is where a controller comes in handy. There are numerous different types of controllers available with different features and consequences and in this thesis, the focus is on the *PID controller*.

The closed-loop system in figure 2.1 describes the relationship between the different signals in a typical process control setting. The main idea is as previously mentioned to keep a process value y as close as possible to a desired setpoint reference value r by manipulating the process $P(s)$ with a controller $C(s)$.

A control error e is found by subtracting the process value y from the reference value r. This control error e is then used as an input to the controller $C(s)$ to compute the control signal u to be the input to the process $P(s)$. In real processes, it is also common to experience different types of load disturbances d or measurement noise n that adds constraints to the control.

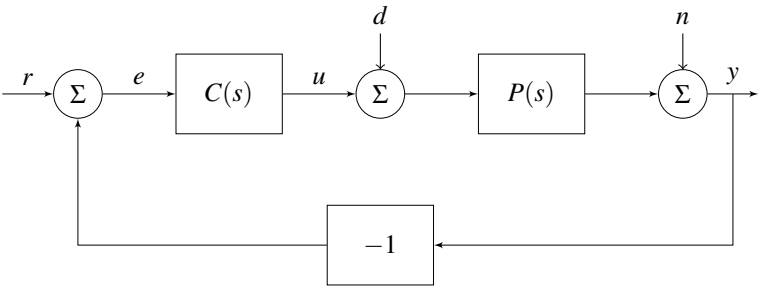

Figure 2.1 A closed-loop system with process $P(s)$ and controller $C(s)$.

The PID-controller

As mentioned previously, the by far most used controller today is the PID-controller which is said to include almost all of the controllers used in the process industry [Åström and Hägglund, 1995]. The PID-controller gets its name from its three components; the *proportional* part, the *integral* part, and the *derivative* part. The proportional part is looking at the current control error. The integral part is looking at the sum of all previous control errors, i.e. the integral of the control error from the start up until now. Lastly, the derivative part is looking at the derivative of the control signal, i.e. how fast and in what direction the error is heading. Typically, the derivative contains a lot of noise which is why a filter often is applied to the process value.

The degree of how much these three parts should affect the control signal u is determined by three coefficients K_c, T_i, and T_d and we attain the control signal as follows,

$$u(t) = K_c\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right). \tag{2.1}$$

When using a controller with derivative action, it is necessary to first filter the process value to avoid a jerky control signal which could bring unnecessary wear and tear on the actuators. In this thesis, the signal is filtered using a low-pass filter as in (2.2) with a time constant of one eighth of the controller's derivative time. When using this in a real setting, this filter is for the user to define and is something important to keep in mind whenever using a PID controller.

$$F(s) = \frac{1}{sT_d/8+1} \tag{2.2}$$

2.2 Automatic tuning

To properly tune the parameters is a crucial step when it comes to designing a controller. A badly tuned controller can not only fail to keep the desired set point value but also cause stress and tear on the equipment from an oscillating process value or with large and fast switching control signals.

Ziegler and Nichols presented in 1942 two methods for tuning of PID-controllers which have gained popularity due to their simplicity to implement and ability to attain a sufficiently good control. Because of this, they are still widely used, either in their original form or with some modifications [Åström and Hägglund, 1995]. The first method, the *step response method* is realized via a simple experiment using a unit step response of the system from which the process's gain, time constant and time delay can be estimated. From a certain set of rules the different parameters K_c, T_i, and T_d can then easily be determined [Åström and Hägglund, 1995].

The other method is known as the *frequency response method* where a proportional controller is connected to the process and its gain K_c increased gradually until the system starts to oscillate. When this occurs, the critical gain k_c is found, and the period of the oscillations is known as the critical period, T_c [Åström and Hägglund, 1995].

However, there are a few issues with these approaches. While the *step response method* could easily be implemented automatically, it tends to give a more aggressive control signal compared to the *frequency response method*, meaning a higher gain K_c. This can in the worst case cause the system to become unstable. The *frequency response method*, on the other hand, is more difficult to automatize as the gain has to be increased until the system starts to oscillate [Åström and Hägglund, 1995]. Another way to instead find these parameters, the critical gain and the critical period, is by introducing a relay function in the control loop that causes the process to oscillate and this is the basic idea behind the relay autotuner [Åström and Hägglund, 1984].

Relay tuning

Introduced by [Åström and Hägglund, 1984] they are suggesting an automated way to find the critical gain k_c and the critical period T_c based on the observation that a system with a phase lag of at least π will start to oscillate with the period T_c. With the relay amplitude d, they show that the critical gain then can be found via a Fourier series expansion as

$$k_c = \frac{4d}{\pi a} \tag{2.3}$$

where a is the amplitude of the process value [Åström and Hägglund, 1984].

The experiment is set up as seen in figure 2.2 where the controller is detached from the process during the identification phase. When the experiment is finished

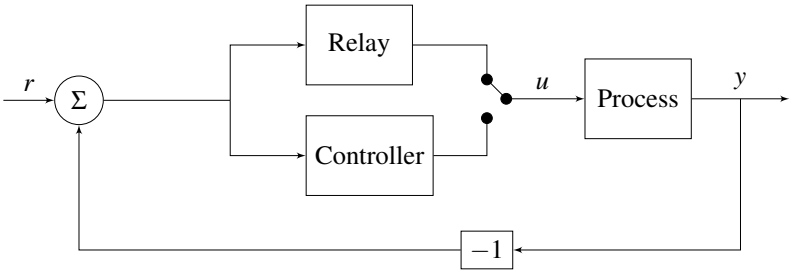

Figure 2.2 The relay feedback loop.

and the PID parameters have been tuned, the controller is switched back to control the process.

Since its introduction, variations of the relay autotuner have in effect become a standard for commercial autotuners used in industry today [Berner, 2017]. One common modification is by the use of an asymmetrical relay instead of the original symmetric relay autotuner presented by [Åström and Hägglund, 1984]. By using an asymmetrical relay, it is possible to excite a wider range of frequencies. This gives more information about the system and allows for a more precise model to better match the process dynamics. This has for example been used in the dissertation of [Berner, 2017] and in the master thesis of [Hansson and Svensson, 2020], and has proven to produce a good model estimate to be used for controller design using a specific set of rules. An asymmetric relay is therefore the focus also for this thesis.

3

Process types and experiment

Here, the process models that have been considered for this thesis are described. The test batch used to evaluate the methods as well as the relay experiment are also explained.

3.1 Process models and classification

The different models that are considered for this thesis are the *First-order model with time delay* (FOTD), the *Second-order model with time delay* (SOTD) as well as the *Second-order model with zero and time delay* (SOZTD). The reason for this is that most processes found in the process industry can be adequately described by these low-order models although the processes themselves might physically have more complex or non-linear dynamics. The phase loss from approximating a process with a lower order model than the actual process is compensated with an increased time delay L [Berner, 2017].

An FOTD model is described as

$$P(s) = \frac{K_p}{sT+1}e^{-sL}, \tag{3.1}$$

where K_p is the process's static gain, T is its time constant, and L is the time delay. For an SOTD model, the parametrization is

$$P(s) = \frac{K_p}{(sT_1+1)(sT_2+1)}e^{-sL}, \tag{3.2}$$

where there are two time constants T_1 and T_2. At last, there is the SOZTD model similar to the SOTD model, but with a zero as well, expressed as

$$P(s) = \frac{K_p(sT_z+1)}{(sT_1+1)(sT_2+1)}e^{-sL}, \tag{3.3}$$

where T_z defines the placement of the zero.

3.2 Benchmark models for the process industry

To evaluate and benchmark this approach for automatic tuning, it will be necessary to apply this to the different types of processes the method is intended to be used for. A test batch of 134 processes introduced by [Åström and Hägglund, 2004] was used throughout the project to apply the design procedure on. The processes are representative of many of the processes encountered in process control and include both delay-dominated, lag-dominated, and integrating processes and can be found in appendix A.

3.3 Relay experiment

Each experiment is initiated by estimating the variance of the process output. This was done following the procedure of [Hansson and Svensson, 2020] using *Welford's online algorithm for iterative variance*. The hysteresis band was then set based on this estimated variance, which was used to determine when the relay switches should occur.

Figure 3.1 is showing the output from a typical asymmetric relay experiment of a process with a positive gain. For a process with negative gain, the opposite will happen with the control signal, and the opposite relay switching rules are applied.

The hysteresis band is shown as the two dashed lines. The first relay is started slowly with an exponential start. When the process value reaches the high hysteresis level, the relay is switched. The process value will start decreasing after time L, where L is the process's time delay. The process value will eventually re-enter the level between the hysteresis band and when it crosses the lower hysteresis level, the second relay is turned off and the control signal is returned to the initial level as before the experiment was initiated. Some time is then given for the transient to settle, after which the experiment is complete.

To find out whether this short experiment was sufficient to capture the most important dynamics of the processes, it was evaluated using some tools in the system identification toolbox in MATLAB, such as ARX, OE, and n4sid. Comparing the simulated data from the identified models to the real process value it could then be concluded that this short experiment de facto was enough to capture the process and reproduce a good model. The advantage of such a short experiment is that it decreases the risk of error during the experiment, such as load disturbances or noise causing the measured signal to deviate from the real value. Once the relay experiment is complete, the next step is to estimate a model.

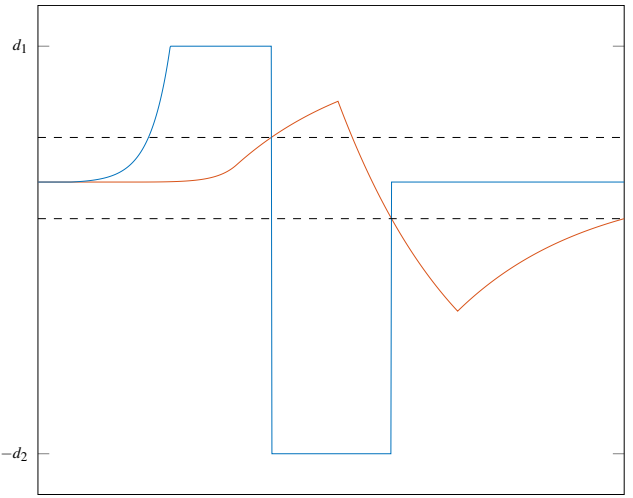

Figure 3.1 Output from a typical asymmetric relay feedback experiment with two relay switches. The control signal (blue) is switched when the process value (red) reaches either of the hysteresis levels (dashed black lines).

4

Model estimation

The original idea for this master thesis was to investigate whether the initialization step in the thesis by [Hansson and Svensson, 2020] by the use of cubic splines would be sufficient to describe the process properly and therefore the first method evaluated for the model estimation part. Two different methods, the ARX model and an FOTD fit, were then also introduced and evaluated, where the latter became the final choice as this proved to be the best option with respect to performance and computational cost as well as ease of implementation.

4.1 Continuous time model estimation

One approach to finding a good model of a system is by having good estimates of the functions for the output signal $y(t)$, the control signal $u(t)$, and their respective n^{th} and m^{th} derivatives, depending on the order of the system.

To find these desired functions, an interpolation method can be used where a continuous function is fitted to all known data points as well as in-between. A common method of doing this is by estimating a polynomial to intersect all the desired points. By allowing this continuous function to be of a sufficiently high order, the needed derivatives can easily be found. A problem with this method, however, is that a very high-order polynomial is needed to match a function that does not correspond well to a polynomial function. This could make the function not resembling the original function very well for values that do not exactly correspond to a predefined point in the data set. One way to avoid this unwanted behavior and to better estimate the original function is by using splines.

A spline is a continuous function that consists of several piece-wise polynomial functions to describe a set of known data points. Depending on the order of said polynomials, the spline curve has various orders of differentiability. The most commonly used splines are cubic splines, meaning that the spline function consists of several third-order polynomials joined together. Hence, the resulting spline is of C^2 continuity, i.e. twice differentiable. This is also what has been used for this thesis and what is further explained here.

Cubic splines

Assume $(\boldsymbol{t},\boldsymbol{y})$ are the set of points of the function to be estimated, where $\boldsymbol{t} = [t_1,\ldots,t_n]^\top$ and $\boldsymbol{y} = [y_1,\ldots,y_n]^\top$. A cubic spline is a function that fulfills:

1. $s(t)$ is a cubic polynomial on $[t_i,t_{i+1}],\ i = 1,\ldots,n-1$;
2. $s(t_i) = y_i$ at each node $t_i,\ i = 1,\ldots,n$;
3. $s''(t)$ exists and is continuous on $[t_1,t_n]$;
4. $s''(t_1) = s''(t_n) = 0$.

First, by introducing the notation in (4.1) where Δ_i represents the time between two neighboring nodes t_i and t_{i+1}, γ_i represents the second-derivative of the spline in the node t_i, and $s_i(t)$ the spline segment for all times t between two nodes t_i and t_{i+1}.

$$\begin{aligned} \Delta_i &= t_{i+1} - t_i && , \ i = 1\ldots n-1, \\ \gamma_i &= s''(t_i) && , \ i = 1\ldots n, \\ s_i(t) &= s(t), t \in [t_i,t_{i+1}] && , \ i = 1\ldots n-1 \end{aligned} \tag{4.1}$$

The following can then be proven to hold for all $t \in [t_1,t_n]$.

$$\begin{aligned} s_i(t_i) = {} & \frac{\gamma_i}{6\Delta_i}(t_{i+1}-t)^3 + \frac{\gamma_{i+1}}{6\Delta_i}(t-t_i)^3 \\ & + \left(\frac{y_{i+1}}{\Delta_i} - \frac{\gamma_{i+1}}{6}\Delta_i\right)(t-t_i) + \left(\frac{y_i}{\Delta_i} - \frac{\gamma_i}{6}\Delta_i\right)(t_{i+1}-t), \end{aligned} \tag{4.2}$$

$$s_i'(t_i) = -\frac{\gamma_i}{2\Delta_i}(t_{i+1}-t)^2 + \frac{\gamma_{i+1}}{2\Delta_i}(t-t_i)^2 + \frac{y_{i+1}-y_i}{\Delta_i} + \frac{\Delta_i}{6}(\gamma_i - \gamma_{i+1}), \tag{4.3}$$

$$s_i''(t) = \gamma_i\frac{t_{i+1}-t}{\Delta_i} + \gamma_{i+1}\frac{t-t_i}{\Delta_i} \tag{4.4}$$

For the n points $(\boldsymbol{t},\boldsymbol{y})$ the fitted spline $s(t)$ that is aimed to be found is the one that minimizes the cost function

$$J_\alpha(s) = \alpha\sum_{i=1}^{n}(y_i - s(t_i)) + (1-\alpha)\int_{t_1}^{t_n} s''(t)dt \tag{4.5}$$

where $\alpha \in [0,1]$ is a tuning parameter. The first term in expression (4.5) rewards model complexity to keep the modeled values as close to the real values as possible, whereas the second term is a penalty that punishes non-linearity in order to attain a function as smooth as possible.

Cross-validation

In order to find the α that results in the best possible fit, the spline function is evaluated with a cross-validation method. The data points are separated into folds of 150 samples where every 10^{th} sample was left out. These folds are overlapping each other with 60%, meaning a pre-defined share of the data points is present in two different sets in order not to miss any important dynamics. The spline values are evaluated at the times of the left-out samples and then compared to the actual sampled value at that time and its squared error is used to evaluate how well the spline function is fitting the data points. This α is then re-calculated for each new model estimate.

Least squares

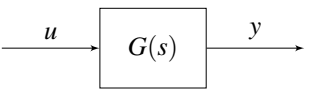

Figure 4.1 Transfer function from $u(t)$ to $y(t)$.

Once the functions have been estimated, a proper model can be found using a least squares method. A continuous system with a time delay L and transfer function $Y(s) = G(s)U(s)$ as in figure 4.1, can in the frequency domain be written as

$$G(s) = \frac{b_m s^m + \cdots + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_0} e^{-sL}. \tag{4.6}$$

In the time domain, equation (4.6) translates to equation (4.7) where the time-delay is affecting the input signal $u(t)$.

$$y^{(n)}(t) + a_{n-1}y^{(n-1)}(t) + \cdots + a_0 y(t) = b_m u^{(m)}(t-L) + \cdots + b_0 u(t-L) \tag{4.7}$$

By re-writing equation (4.7), the system can be represented as an over-fitted system of equations as in (4.8) by using arrays with the sampled input and output values, $\boldsymbol{u}$ and $\boldsymbol{y}$, respectively.

$$\boldsymbol{y}^{(n)}(t) = \begin{bmatrix} \boldsymbol{y}^{(n-1)}(t) & \cdots & \boldsymbol{y}(t) & \boldsymbol{u}^{(m)}(t-L) & \cdots & \boldsymbol{u}(t-L) \end{bmatrix} \begin{bmatrix} -a_{n-1} \\ \vdots \\ -a_0 \\ b_m \\ \vdots \\ b_0 \end{bmatrix} \tag{4.8}$$

For clarity, equation (4.8) is re-written as

$$Y = X\theta, \tag{4.9}$$

where $Y = \boldsymbol{y}^{(n)}(t)$, $X = [\boldsymbol{y}^{(n-1)}(t) \ \cdots \ \boldsymbol{y}(t) \ \boldsymbol{u}^{(m)}(t-L) \ \cdots \ \boldsymbol{u}(t-L)]$, and $\theta = [-a_{n-1} \ \cdots \ -a_0 \ b_m \ \cdots \ b_0]^\top$. Using a least squares method, the best parameters θ can be found if and only if the matrix $X^\top X$ is invertible, i.e. if it has full rank. The parameters are then expressed as

$$\theta = [X^\top X]^{-1} X^\top Y. \tag{4.10}$$

Going back to equation (4.6), the optimal parameters θ are used to describe the system $G(s)$. By repeating this process for a range of different time delays L, the best model candidate can be found by choosing the one that minimizes the output-error norm $||y-\hat{y}||_2$, where y is the process value and $\hat{y}$ is the simulated output using the parameters θ and control signal u.

Model evaluation

Below are some simulation results and frequency responses for a sample of four different types of processes as well as their FOTD, SOTD, and SOZTD estimates for the cubic spline estimation method. In figure 4.2, process 3 in the batch is evaluated, which has the transfer function

$$G(s) = \frac{10}{s+10} e^{-1s}. \tag{4.11}$$

Here, the FOTD is able to capture the dynamics quite well whereas the higher-order models are resulting in some oscillations. Instead, looking at figure 4.3 where we have process 63 from the batch with transfer function

$$G(s) = \frac{64}{(s+8)(s+4)(s+2)(s+1)}, \tag{4.12}$$

the FOTD is not at all capturing the dynamics while the SOTD and SOZTD models are better at describing the system. Also, the frequency responses for these two estimates are good and manages to capture the important dynamics.

In figure 4.4 the estimated results are shown for process 71 in the batch, which is an integrating process with transfer function

$$G(s) = \frac{10}{s(s+10)} e^{-0.9s}. \tag{4.13}$$

Again, the FOTD estimate is not able to capture the dynamics in a good way, but the SOTD and SOZTD estimates seem to better describe the desired behavior. At last, in figure 4.5 process 85 in the batch with transfer function

$$G(s) = \frac{10}{(s+10)(s+1)} e^{-0.9s} \tag{4.14}$$

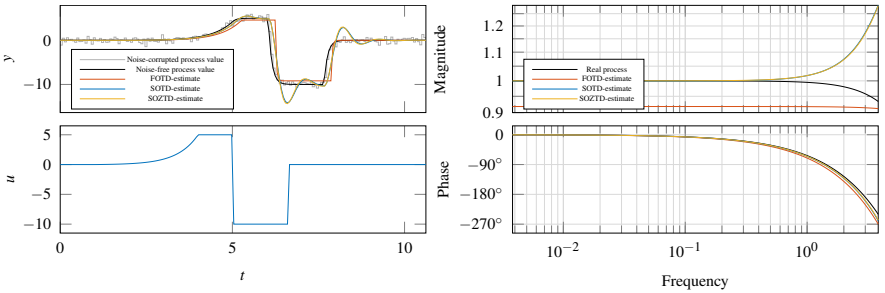

Figure 4.2 The estimated FOTD, SOTD, and SOZTD models from the cubic spline estimation method for process 3 in the test batch, with transfer function (4.11). To the left, the time response for the three models driven by the relay experiment control signal, and to the right, their frequency response.

is shown. Similar to previous simulations, the SOTD and SOZTD manages to capture the dynamics quite well whereas the FOTD does not.

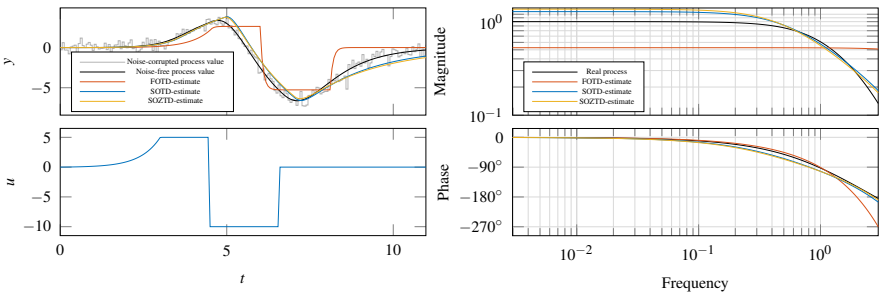

Figure 4.3 The estimated FOTD, SOTD, and SOZTD models from the cubic spline estimation method for process 63 in the test batch, with transfer function (4.12). To the left, the time response for the three models driven by the relay experiment control signal, and to the right, their frequency response.

In figure 4.6 the L_2-norm can be seen for the different processes estimated with an FOTD, an SOTD, and an SOZTD model using the cubic spline estimation model. For the FOTD model estimate, the average L_2-norm for the error compared to the real process was 2.0654 with a variance of $\sigma^2 = 4.3700$. For the SOTD estimate, the average root mean square error (RMSE) was 1.0492, and the variance $\sigma^2 = 0.3209$. At last, for the SOZTD model estimate, the mean of the RMSE was 1.2630, and the variance $\sigma^2 = 0.5428$. From these numbers and figure 4.6, the SOTD is the best option.

One reason as to why this method does not perform as desired could be that the relay switches of the control signal are not possible to differentiate. This leaves

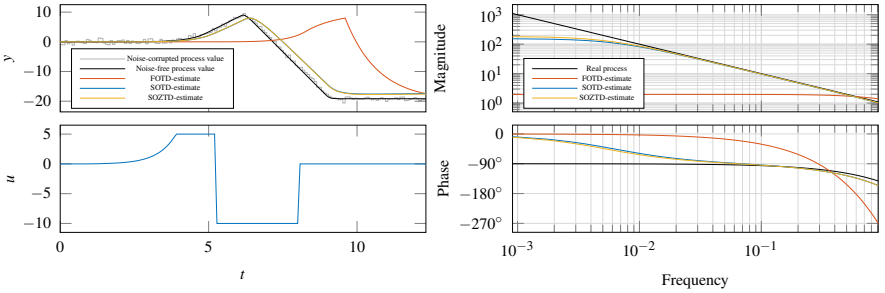

Figure 4.4 The estimated FOTD, SOTD, and SOZTD models from the cubic spline estimation method for process 71 in the test batch, with transfer function (4.13). To the left, the time response for the three models driven by the relay experiment control signal, and to the right, their frequency response.

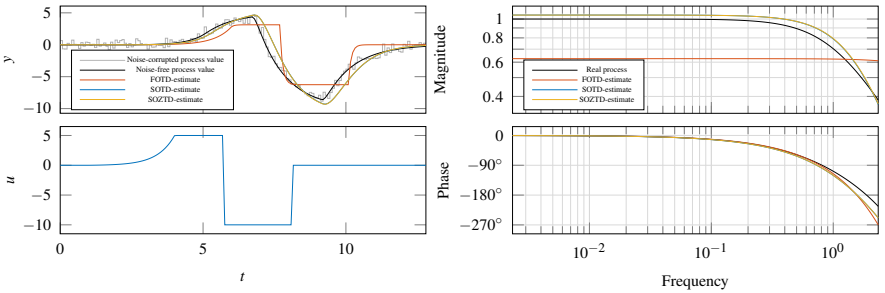

Figure 4.5 The estimated FOTD, SOTD, and SOZTD models from the cubic spline estimation method for process 85 in the test batch, with transfer function (4.14). To the left, the time response for the three models driven by the relay experiment control signal, and to the right, their frequency response.

room for an ambiguous interpretation of the first and second derivative of the control signal for the times when the relay switches occur.

4.2 ARX model estimation

The second model estimation was the *AutoRegressive with eXogenous input model* (ARX) which is a well-known discrete estimation model. The general, time-discrete model can be expressed by

$$y(t) = \eta(t) + w(t), \tag{4.15}$$

where $\eta(t)$ is the disturbance-free output and $w(t)$ the noise. Both these terms can in turn be written as transfer functions $G(q,\theta)$ and $H(q,\theta)$ from $u(t)$ and $e(t)$ re-

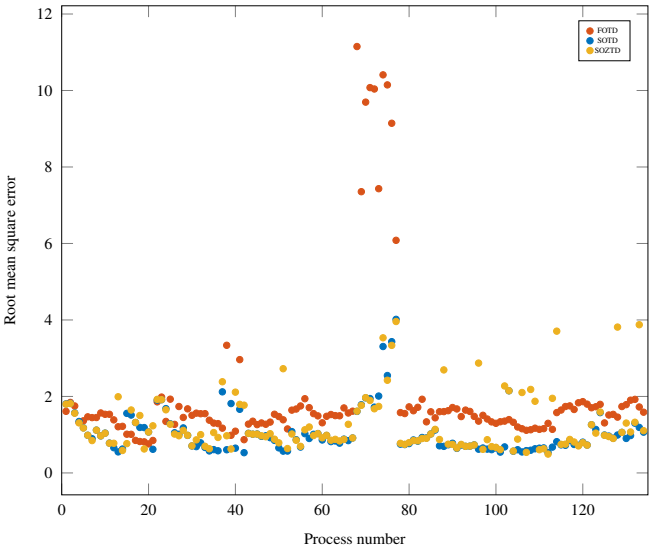

Figure 4.6 The RMSE for the cubic spline estimation model for the different processes in the batch.

spectively, where $u(t)$ is the control signal and $e(t)$ represents white noise. With

$$G(q,\theta) = \frac{B(q)}{F(q)} = \frac{b_1 + b_2 q^{-1} + \cdots + b_{nb} q^{-nb+1}}{1 + f_1 q^{-1} + \cdots + f_{nf} q^{-nf}} q^{-nk} \tag{4.16}$$

and

$$H(q,\theta) = \frac{C(q)}{D(q)} = \frac{1 + c_1 q^{-1} + \cdots + c_{nc} q^{-nc}}{1 + d_1 q^{-1} + \cdots + d_{nd} q^{-nd}} \tag{4.17}$$

where nk is the time delay in samples, the output $y(t)$ can be expressed as

$$y(t) = \frac{B(q)}{F(q)} u(t) + \frac{C(q)}{D(q)} e(t). \tag{4.18}$$

This model is known as *Box-Jenkins* (BJ) model and the aim is to find the optimal parameters $\theta = [b_i, c_i, d_i, f_i]$ given the five structural parameters nb, nc, nd, nf, and nk [Ljung and Glad, 2016].

A special case of this, where $F(q) = D(q) = A(q)$ and $C(q) \equiv 1$ is the ARX model, and its model structure can be seen in figure 4.7. The advantage of the ARX-model over other black-box models derived from the BJ-model is that it can be transformed into a least squares problem and solved rather easily at low computational cost [Ljung and Glad, 2016].
The transfer function 4.18 re-written for the special case of the ARX-model can be expanded to the following.

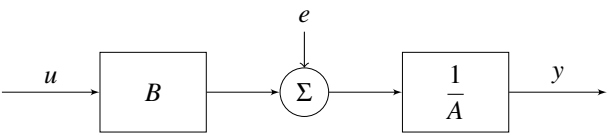

Figure 4.7 ARX-model structure.

$$y(t)+a_1y(t-1)+\cdots+a_{na}y(t-na)=b_1u(t-nk)+\cdots+b_{nb}u(t-nk-nb+1)+e(t) \tag{4.19}$$

Bilinear transformation

In order to represent the attained discrete-time model from the ARX-estimation as a continuous-time system, a transformation between the two domains is needed. One such method is the Bilinear transform, also known as Tustin's approximation [Wittenmark et al., 2002]. The relationship between a discrete-time transfer function $H(z)$ and a continuous-time transfer function $G(s)$ is represented as $z=e^{sh}$. Three common approximations for this transformation are:

$$z=e^{sh}\approx 1+sh \qquad \text{(Forward difference)}$$

$$z=e^{sh}\approx \frac{1}{1-sh} \qquad \text{(Backward difference)}$$

$$z=e^{sh}=\frac{e^{sh/2}}{e^{-sh/2}}\approx\frac{1+sh/2}{1-sh/2} \qquad \text{(Trapezoidal method)}$$

where the latter is used for Tustin's approximation. The continuous-time transfer function is then analogously found by replacing z with the desired approximate. For Tustin's approximation, this results in

$$G(s)=H(z)\Big|_{z=\frac{1+sh/2}{1-sh/2}}. \tag{4.20}$$

Model evaluation

Here some results for the same models as in section 4.1 are shown. In figure 4.8 the process with transfer function (4.11) is shown. As can be seen, all three model estimates manage to recreate the real process dynamics quite well, both for the time- and the frequency response.

In figure 4.9 the process with transfer function (4.12) can be seen. The FOTD does not completely manage to capture the process dynamics, but both the SOTD and the SOZTD model estimates do a good job in resembling the real process.

The dynamics for (4.13) are shown in figure 4.10. Again, the FOTD does not completely manage to describe the true process dynamics, while the SOTD and the SOZTD do a much better job.

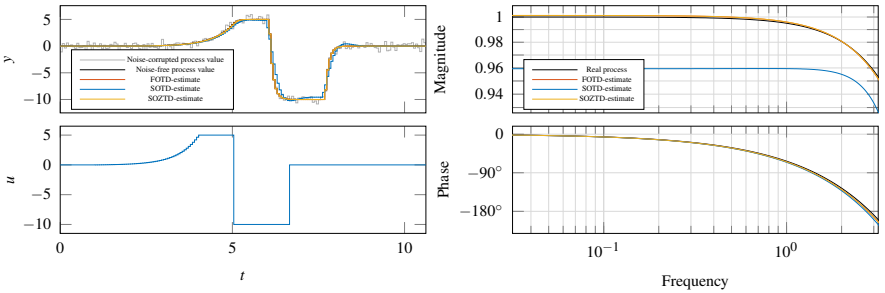

Figure 4.8 The estimated FOTD, SOTD, and SOZTD models from the ARX estimation method for process 3 in the test batch, with transfer function (4.11). To the left, the time domain response for the three models driven by the relay experiment control signal, and to the right, their frequency response.

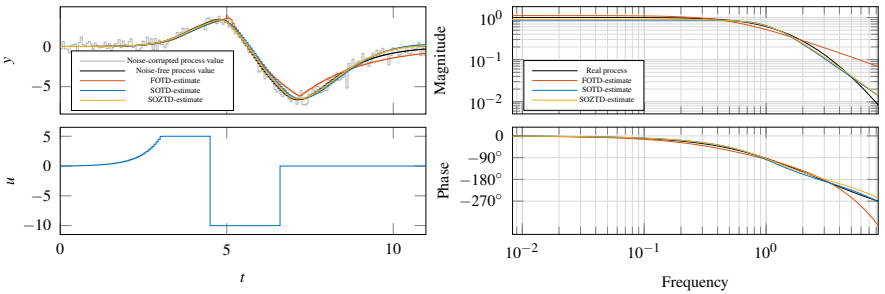

Figure 4.9 The estimated FOTD, SOTD, and SOZTD models from the ARX estimation method for process 63 in the test batch, with transfer function (4.12). To the left, the time domain response for the three models driven by the relay experiment control signal, and to the right, their frequency response.

At last, in figure 4.11 the process with transfer function (4.14) is shown. Here, all three models manage very well to describe the true dynamics.

In figure 4.12, the RMSE is shown for each model estimation. The mean RMSE for the FOTD estimate is 0.3047 with a variance of $\sigma^2 = 0.0746$. For the SOTD estimate the similar value is 0.2207 as well as $\sigma^2 = 0.0219$ and for the SOZTD, 0.0885 and $\sigma^2 = 0.0103$. As can be seen, the SOZTD model estimate generally performs the best, followed by the SOTD and at last the FOTD estimate.

4.3 Time efficient FOTD model estimation

Recall that an FOTD model can be described using three parameters, L, T, and K_p, and be written on the form

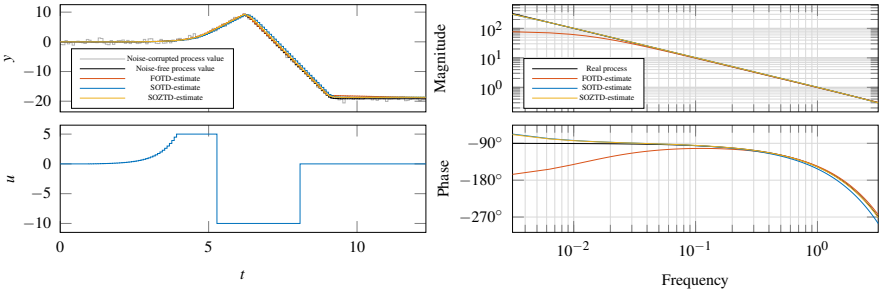

Figure 4.10 The estimated FOTD, SOTD, and SOZTD models from the ARX estimation method for process 71 in the test batch, with transfer function (4.13). To the left, the time domain response for the three models driven by the relay experiment control signal, and to the right, their frequency response.

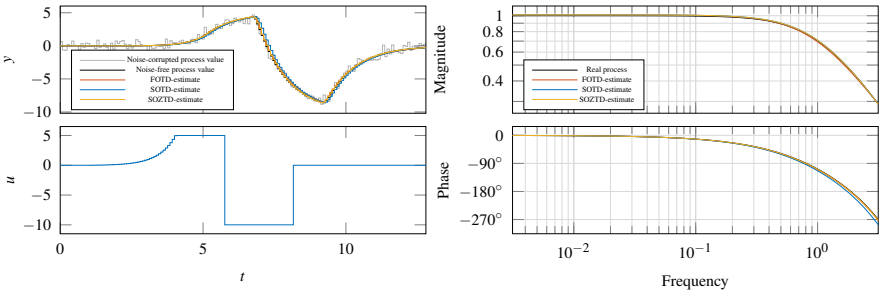

Figure 4.11 The estimated FOTD, SOTD, and SOZTD models from the ARX estimation method for process 85 in the test batch, with transfer function (4.14). To the left, the time domain response for the three models driven by the relay experiment control signal, and to the right, their frequency response.

$$P(s) = \frac{K_p}{sT+1} e^{-sL}. \tag{4.21}$$

A process can be classified based on the normalized time delay

$$\tau = \frac{L}{L+T} \tag{4.22}$$

for an FOTD. τ is a value ranging from 0 to 1 where for low τ-values, the process is defined as lag-dominated, and for higher τ-values as delay-dominated. This can later play a major role in the decision of an appropriate controller design, but here it will also be used during the estimation part.

To find the best FOTD model to match the process dynamics, the objective is to find the optimal parameters $\theta = [K_p \;\; T \;\; L]^\top$ that minimize the output-error norm

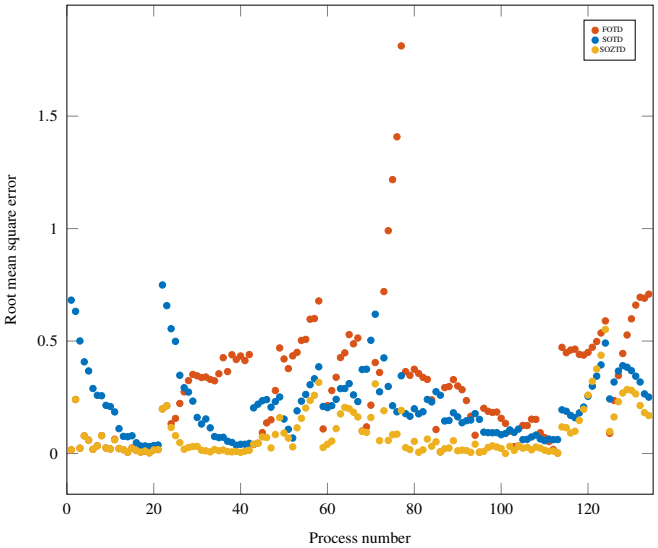

Figure 4.12 The RMSE for the different processes in the batch.

$$V(\theta|u,y) = ||\hat{y}(\theta) - y||_2^2 = (\hat{y}(\theta) - y)^\top (\hat{y}(\theta) - y). \tag{4.23}$$

Using the normalized time-delay τ described in (4.22), a finite range in the parameters describing the FOTD model can be defined — with $\tau \in [0 \;\; 1]$ and $L \in [0 \;\; L_{max}]$, where L_{max} is the longest time between two relay switches. However, due to implementational reasons, the actual lower limit used for τ is in fact ε_τ, meaning the range is $\tau \in [\varepsilon_\tau \;\; 1]$. For each pair τ and L, the system was first zero-order hold (ZOH) discretized and simulated using $K_p = 1$ to attain $\hat{y}$. The last parameter, the static gain K_p of the system was then found by minimizing the cost

$$V(K_p|\tau,L) = ||y - K_p\hat{y}||_2, \tag{4.24}$$

which is the same as minimizing

$$V' = [y - K_p\hat{y}]^\top [y - K_p\hat{y}] = y^\top y - 2K_p y^\top \hat{y} + K_p^2 \hat{y}^\top \hat{y}. \tag{4.25}$$

Differentiating (4.25) by K_p and equating to zero, the following expression for K_p is attained if and only if $\hat{y}^\top \hat{y} \neq 0$

$$K_p = \frac{y^\top \hat{y}}{\hat{y}^\top \hat{y}}. \tag{4.26}$$

At last, the model candidate using this τ and L can then be computed and evaluated by their RMSE. Repeating this process for 100 equidistantly spaced points in both

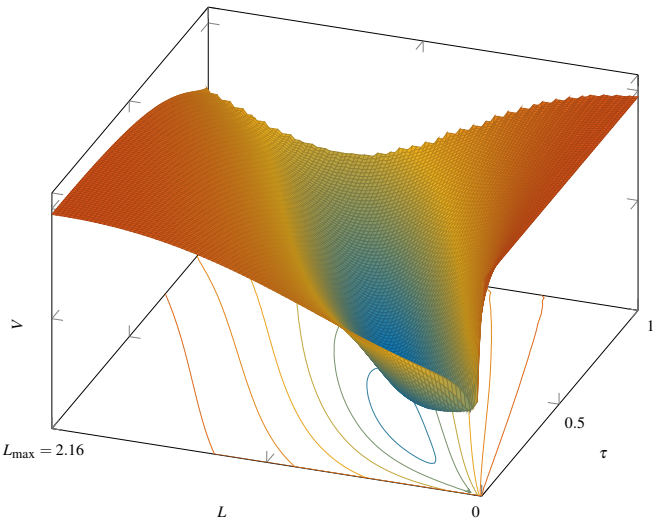

Figure 4.13 Cost for different values of τ and L for process 63 in the batch, with transfer function (4.12).

τ and L results in a total of 10000 evaluation points for which each respective cost can be seen in figure 4.13 for the process with transfer function (4.12). It can be seen that there exists a global minimum at around $\tau \approx 0.25$ and $L \approx 0.7$.

Using 100 points each for both τ and L corresponds to a resolution of 1% of their value range and requires as previously stated a total of 10000 different evaluations. Given the somewhat limited computing power in the controller, it is therefore desirable to find a more efficient identification method where not all points are evaluated. By first verifying that no local minima existed in close proximity to the global minimum for any of the processes in the batch, a much more elegant solution could then be implemented by only evaluating points in the areas of interest.

The idea behind this optimized search method is as follows. Five equidistantly spaced points in L are initially evaluated. For each point L, the optimal τ and K_p are found as the ones that minimize the cost function (4.23). The L that produced the lowest cost is then chosen for further analysis by evaluating a smaller interval around this point in five new points the same way as just described. This procedure is then repeated until the desired tolerance is reached. The search for an optimal τ is done in the same way as for L by incrementally decreasing the interval until desired tolerance is met. Below, an example of finding the optimal τ for a fixed L is described.

In figure 4.14 again the cost function for batch process 63 with transfer function in (4.12) can be seen, but here with fixed $L = 0.6327$. In the first step of the algorithm, the five points shown in black are evaluated. Of these, $\tau \approx 0.25$ was the

one resulting in the lowest cost. It is then known that the lowest cost will be in the range of $\tau = [\varepsilon_\tau, 0.5]$, and these will serve as the lowest and highest bound respectively for the next iteration. This means that half of the interval can be discarded right away. If the point corresponding to the lowest cost instead would have been one of the end-points, an even bigger interval could be discarded as the new interval then is defined between that end-point and its only neighbor, one fourth of the previous interval. For the next iteration, $\tau \approx 0.25$ will be the middle point and two new points, $\tau \approx 0.125$, and $\tau \approx 0.375$, shown in gray, are evaluated and compared with the known costs of the already evaluated points. Looking at figure 4.14, the point corresponding to the lowest cost is still the one at $\tau \approx 0.25$ and will therefore serve as the middle point for the following iteration with $\tau \approx 0.125$ being the lower limit for the interval and $\tau \approx 0.375$ being the higher limit. This process is then repeated until the size of the interval is within the desired tolerance. In this thesis a tolerance of 0.01 of the initial interval size has been considered, meaning $\varepsilon_\tau = 0.01$ and $\varepsilon_L = 0.01 L_{max}$.

It can be proven that the optimal number of points to evaluate in each iteration, for both τ and L, is 5. This means that at least half of the interval is discarded each iteration by only evaluating two new points as the values of the two edge points as well as the middle point are known from the previous iteration. If the value corresponding to the lowest cost is one of the end-points, three quarters of the interval is discarded and three new points are evaluated for the next iteration.

Theoretically, $4-7$ iterations are required to reach the desired tolerance. This means that the worst-case scenario would then be to evaluate a total of 17 points in each dimension, three initial points plus 14 from the 7 iterations. This equates to evaluating a total of $17 \times 17 = 289$ points compared to 10000 to attain a precision better than evaluating every grid-point corresponding to the desired tolerance.

Model evaluation

For this model estimation, the same processes as for the previous models are shown. As can be seen, the models can quite well describe the dynamics of the system. Figure 4.15 is showing the true process dynamics (4.11) as well as the corresponding estimated FOTD model. As can be seen, the estimated model manages very well to capture and describe the true dynamics. In figure 4.16 the dynamics for process (4.12) are shown. For the frequency region shown, the estimated model is managing to mimic the dynamics of the true process quite well. For higher frequencies, the phase shift of the true process will converge to -360°, while the time delay of the FOTD estimate will cause the phase shift to go towards $-\infty$. However, we are only interested in the frequency region up to where the phase shift is -180° to design an adequate PID controller. As can be seen in the right of figure 4.16, the two processes are in fact quite correlated around this frequency.

In figure 4.17 the dynamics for process (4.13) can be seen. For both the time response and the frequency response, the FOTD estimate is managing very well to

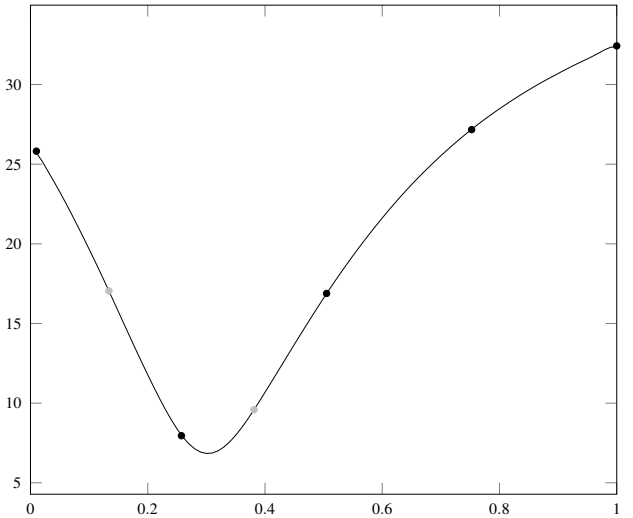

Figure 4.14 Cost for different values of τ and $L = 0.6327$ for process 63 in the batch, with transfer function (4.12). The black dots mark the evaluated points for the first iteration. The gray dots mark the two new evaluated points for the second iteration.

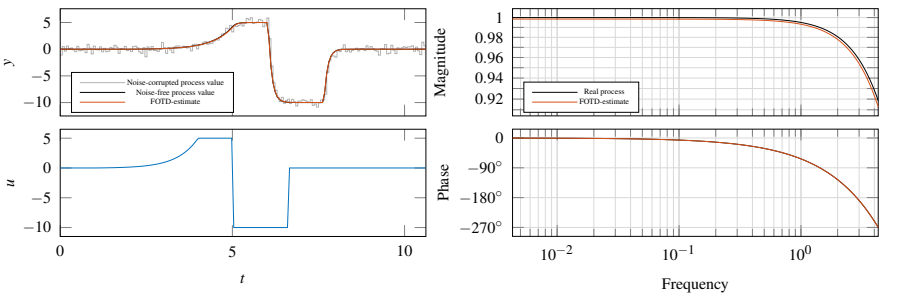

Figure 4.15 The estimated FOTD model from the parameter search method for process 3 in the test batch, with transfer function (4.11). To the left, the time domain response for the three models driven by the relay experiment control signal, and to the right, their frequency response.

capture the dynamics. The same goes for the process with transfer function (4.14) which is found in figure 4.18.

In figure 4.19, the RMSE of each model can be seen. The average RMSE for each process estimated using this method is 0.5514 with a variance $\sigma = 0.0098$.

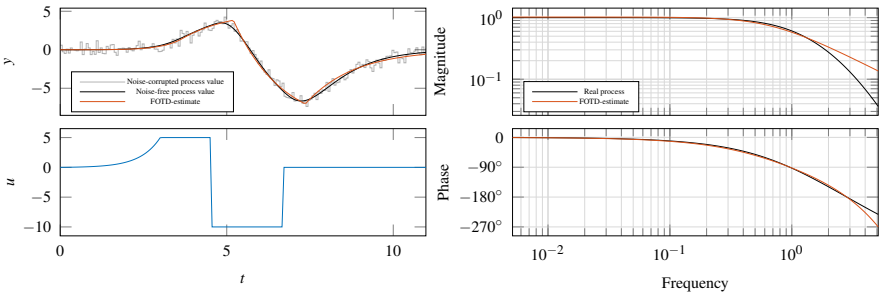

Figure 4.16 The estimated FOTD model from the parameter search method for process 63 in the test batch, with transfer function (4.12). To the left, the time domain response for the three models driven by the relay experiment control signal, and to the right, their frequency response.

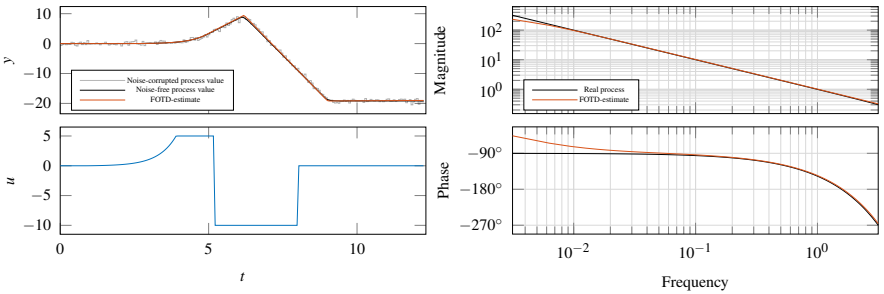

Figure 4.17 The estimated FOTD model from the parameter search method for process 71 in the test batch, with transfer function (4.13). To the left, the time domain response for the three models driven by the relay experiment control signal, and to the right, their frequency response.

4.4 Conclusion

Looking at the numerics of the time-domain simulations, the SOZTD model estimate using the ARX model is clearly the best option. Something to keep in mind, however, is that only looking at the RMSE is not the most important or relevant comparison. This gives an estimation of how well the system responds to a certain control signal, in this case, the output from a relay experiment. The more important aspect is how the estimated model represents the real process in the frequency domain from looking at the frequency response. This can not be represented numerically in the same way as for the simulations in the time-domain, but instead by analyzing all frequency responses in the most important region, i.e. up to the frequency where the phase shift is -180 degrees. Here, it could although be concluded

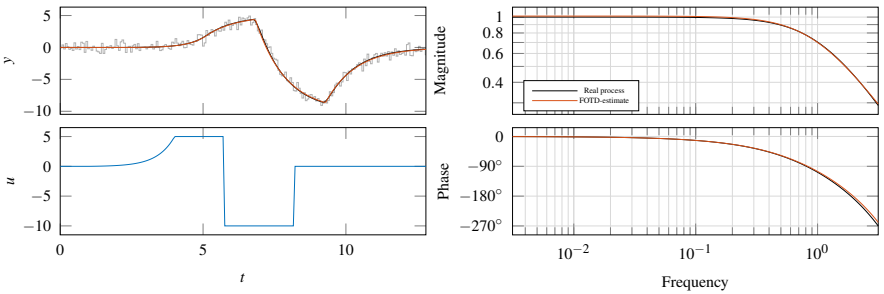

Figure 4.18 The estimated FOTD model from the parameter search method for process 85 in the test batch, with transfer function (4.14). To the left, the time domain response for the three models driven by the relay experiment control signal, and to the right, their frequency response.

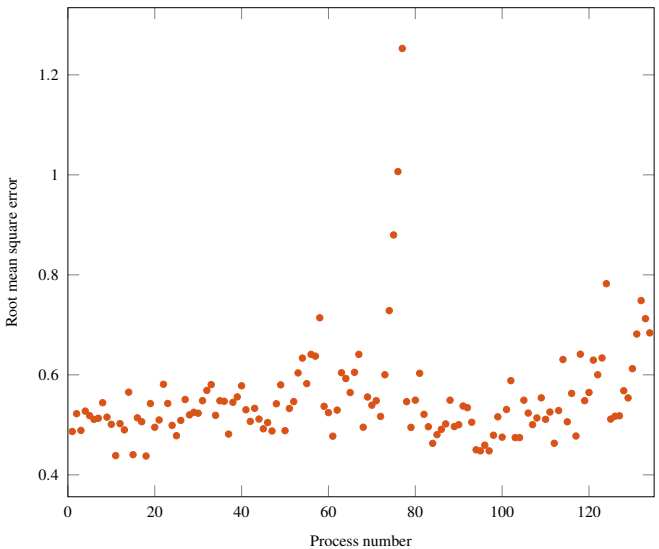

Figure 4.19 The RMSE for the different processes in the batch.

that the model estimate to best describe the true dynamics was in fact the SOZTD estimate from the ARX model estimation.

However, considering the execution time of the FOTD model as well as the abundance of controller design methods available, it was concluded that the model attained from the FOTD parameter search method would be the best option. Furthermore, for an ARX-estimated model, it is necessary to allow for some matrix operations in the controller that neither of the five languages available in the controller unit supports in a straightforward way. It is also worth mentioning that matrix operations generally require more computational power in terms of memory and CPU power. It would therefore be necessary to keep the sizes of the matrices low which is corresponding to a lower number of data points. To reach a data length where such computations are possible to perform in a controller, it is possible that the data has been downsampled to the extent that it is no longer managing to resemble the unknown process dynamics.

5

Controller design

In this chapter, the procedure for computing the PI or PID controller parameters is described. The controllers are then evaluated by a simulation study using a step response from a load disturbance step.

5.1 Controller design

Once the FOTD model has been found, the PID tuning parameters can be calculated according to table 5.1 for PI controller parameters and 5.2 for PID controller parameters.

Two different controller design methods have been considered, the AMIGO method as well as the lambda method [Åström and Hägglund, 2006]. For the lambda method, the parameter λ is determining how fast the controller should be. For this thesis, $\lambda = T$ has been used. For lag-dominated processes, the AMIGO method has from previous research proven to not be so effective and the choice for controlling such processes, therefore, landed on the lambda method. The cut-off for when the lambda method should be used was for $\tau < 0.25$ and for processes with a higher τ, the AMIGO method was used. For integrating processes the lambda method is known to not perform very well, and the AMIGO method was therefore used for these processes, despite having a low τ. As the models are all approximative it can be difficult to decide what process would qualify as an integrating one. A process is in this thesis defined as integrating if the estimated time constant is ten times greater than the experiment time, $T > 10t_n$.

5.2 Closed-loop simulation

The performance of a controller can quite well be assessed using a step in the load disturbance and see how well the process value returns to the desired setpoint. From figure 2.1, the output y can be computed as the following,

$$Y(s) = \frac{1}{1+C(s)P(s)}\Big(N + P(s)D + C(s)P(s)R\Big). \tag{5.1}$$

Table 5.1 Tuning parameters for the AMIGO- and λ-method for a PI controller.

	AMIGO	Lambda
K_c	$\frac{1}{K_p}\left(0.15+0.35\frac{T}{L}-\frac{T^2}{(L+T)^2}\right)$	$\frac{1}{K_p}\frac{T}{L+\lambda}$
T_i	$0.35\,\mathrm{L} + \frac{13LT^2}{T^2+12LT+7L^2}$	T

Table 5.2 Tuning parameters for the AMIGO- and λ-method for a PID controller.

	AMIGO	Lambda
K_c	$\frac{1}{K_p}\left(0.2+0.45\frac{T}{L}\right)$	$\frac{1}{K_p}\frac{0.5L+T}{0.5L+\lambda}$
T_i	$\frac{0.4L+0.8T}{L+0.1T}L$	$T+L/2$
T_d	$\frac{0.5LT}{0.3L+T}$	$\frac{LT}{L+2T}$

Only looking at the load disturbance d, the transfer function from d to y can be expressed as

$$G_{dy}(s) = \frac{P(s)}{1+C(s)P(s)}. \tag{5.2}$$

By taking a step on (5.2), the following results is attained for the same processes as previously evaluated in chapter 4, which can be seen in figure 5.1, 5.2, 5.3, and 5.4.

For some of the processes, both the control signal as well as the process value for the PID controller is a bit jagged and not following the same smooth path as for the PI controller as seen in figure 5.1. This is most likely an effect of using a controller with a long derivative time T_d, meaning that the derivative part will have a great impact on the control signal.

The odd behavior for the step response of the uncontrolled system in figure 5.3 is because it is a process with an integrator, and without a controller, the process value will therefore never return to a steady-state from a load disturbance.

5.3 Conclusions

From the results in section 5.2 it can be concluded that a good controller can be designed based on the estimated FOTD models described in section 4.3. Moreover, using the two proposed tuning rules, the lambda method and the AMIGO method, an adequate closed-loop behavior has been demonstrated. Looking at the uncontrolled

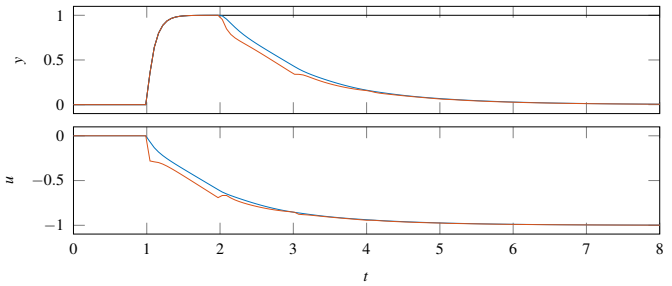

Figure 5.1 Load disturbance step on process 3 in the batch with transfer function (4.11) controlled with a PI controller (blue), a PID controller (red), and the uncontrolled system (black).

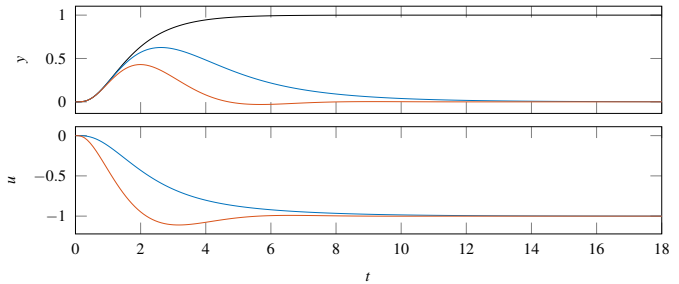

Figure 5.2 Load disturbance step on process 63 in the batch with transfer function (4.12) controlled with a PI controller (blue), a PID controller (red), and the uncontrolled system (black).

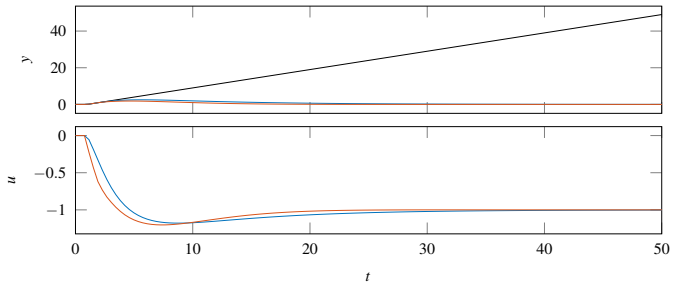

Figure 5.3 Load disturbance step on process 71 in the batch with transfer function (4.13) controlled with a PI controller (blue), a PID controller (red), and the uncontrolled system (black).

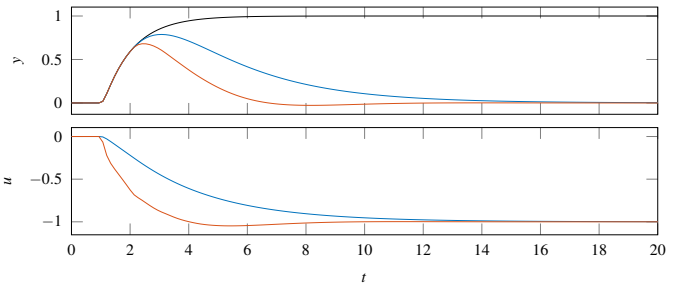

Figure 5.4 Load disturbance step on process 85 in the batch with transfer function (4.14) controlled with a PI controller (blue), a PID controller (red), and the uncontrolled system (black).

(true) system G_p plotted in black, and comparing it to the process values of the same system controlled with a PI and a PID controller, it can furthermore be seen that the controller is quite quick when it comes to getting the process value back to its setpoint. From this, it has been demonstrated that this proposed method theoretically manages to design a PI or a PID controller to be used in the process industry.

6

Implementation in the ABB control system

In this chapter, some background, as well as some insight into the implementation in ABB Ability™ System 800xA, is described. An evaluation on a physical process is also carried out and the results from using the proposed controller in comparison to the existing ABB autotuner are presented.

6.1 Backgound to the system

ABB's flagship Distributed Control System is found in the ABB Ability™ System 800xA. The heart of the ABB Ability™ System 800xA is the ABB AC 800M Programmable Automation Controller (PAC) which is a family of modules consisting of CPUs, communication modules, I/O modules, power supply modules and various accessories. AC 800M supports the open international IEC 61131-3 standard which includes five different programming languages used in industrial automation; Sequential Function Charts (SFC), Ladder Diagram (LD), Function Block Diagram (FBD), Instruction List (IL), and Structured Text (ST) [IEC, 2003]. The first three languages, SFC, LD, and FBD are graphical while the two others, IL and ST are textual [IEC, 2003].

To configure the code and hardware layout of a control application, the ABB Control Builder has been used for this thesis, which is a powerful tool that supports all five IEC 61131-3 programming languages. This has an extensive out-of-the-box function block library ranging from logical operations to autotuning PID controllers [*Control Builder* 2021]. Apart from the IEC 61131-3 programming languages, ABB Control Builder also supports two other languages, Function Diagram (FD) and Control Module Diagram (CMD) [Pernebo and Hansson, 2002]. For the implementation in this thesis, a combination of ST, FD, and CMD has been used.

6.2 Procedure

The code execution in an ABB AC 800M PAC is based on scans where one scan is referring to an entire block of code being executed. This occurs once each sample, where the sample time is explicitly defined beforehand.

Due to the way this is implemented it becomes crucial that computations are not too complex and manage to finish in less than the duration of one sample. This was an issue that arose during the implementation of the new autotuner. Although only including simple arithmetic, many deeply nested functions being called multiple times caused the controller to time out when using a sample time that was too short. By dividing the different functions into smaller portions that executed in several consecutive scans it was, however, possible to estimate an FOTD model in the controller.

The experiment was then carried out in the following way. First, the controller is set to manual mode, meaning a manual control signal is being fed to the process. The system is then given some time to stabilize before the experiment is started. The relay experiment is then started followed by the model estimation based on this outcome. The corresponding controller parameters are then sent to the controller which was then put in automatic mode.

6.3 Experiment on laboratory process

The autotuner was then tested on a physical process, namely the dual tank process used in a few different laboratory sessions in the department of automatic control at Lund university. Using two tanks as seen in figure 6.1, there are essentially two different processes available for the single-input single-output (SISO) scenario. The first of these two different processes is by looking only at the upper tank, thus controlling level l_1 and the other is controlling level l_2. Physically, these processes are known to be nonlinear FOTD and SOTD processes for the upper and lower tank, respectively. Here we are investigating if a linear FOTD model estimate can be sufficient to capture the important dynamics and from that design an appropriate controller.

Below, the outcome of the water tank experiments are shown. Both the upper and the lower tank processes were evaluated using both the new, proposed autotuner as well as the existing autotuner for a PI and a PID controller. The estimated model for the upper tank using this approach is the following,

$$\hat{G}(s) = \frac{1.6214}{9.7339s+1} e^{-0.20384s}. \tag{6.1}$$

From this model, with transfer function (6.1), the controller parameters for the corresponding PI and PID controller types can be found in tables 6.1 and 6.2, respectively. Here, the parameters found by the existing autotuner are also shown. The

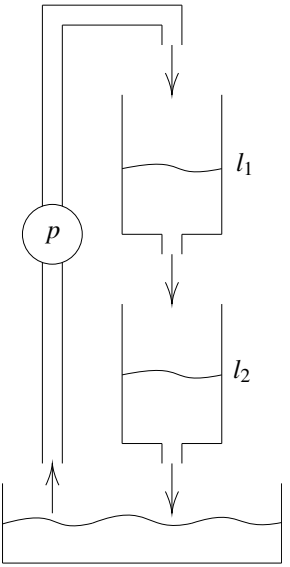

Figure 6.1 Tank process used for the simulation. l_1 denotes the level in the upper tank, and l_2 the level in the lower tank. The control signal is controlling the pump p.

Table 6.1 Upper PI

	K_c	T_i
Proposed autotuner	0.61	9.73
Existing autotuner	2.37	8.66

existing autotuner has three settings; *fast*, *normal*, and *slow*, where here, the parameters for the normal mode are shown. Using these parameters, the controller was evaluated using a step in the reference value as well as using a load disturbance. For the PI controlled system, this is shown in figure 6.2. For the PID controlled system, figure 6.3 is showing the equivalent values.

For the lower tank, the estimated model is as seen in (6.2). The corresponding PI and PID controller parameters have from this model resulted in the ones in table 6.3 and 6.4, respectively. Evaluation of this controller for the lower process is found in figure 6.4 for the PI controller, and 6.5 for the PID controller.

$$\hat{G}(s) = \frac{1.6558}{14.915s+1} e^{-1.915s} \tag{6.2}$$

From the figures, one can see that for these processes, the existing autotuner is faster in terms of how the process value is adjusting to the desired setpoint value,

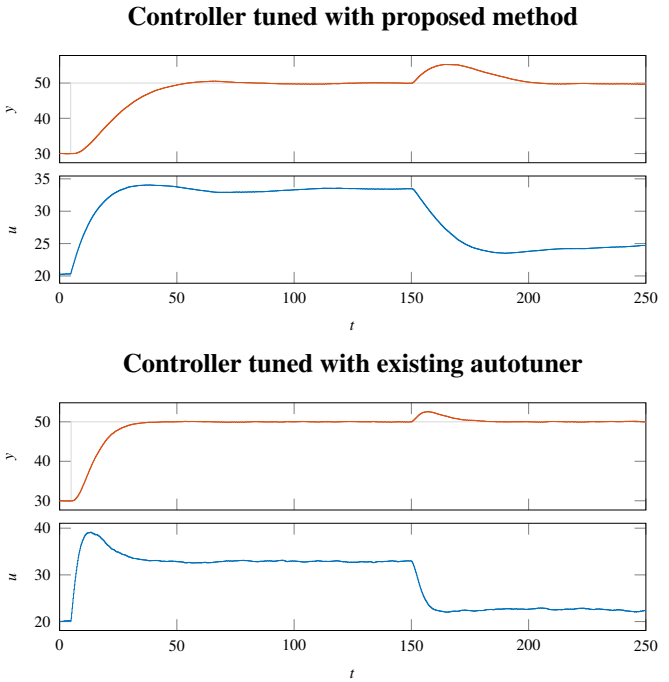

Figure 6.2 The upper tank level controlled with a PI Controller using a setpoint change from 30 to 50 at time $t = 5s$ and a load disturbance of 10 on the control signal at time $t = 150s$ for the proposed autotuner as well as the existing autotuner.

Table 6.2 Upper PID

	K_c	T_i	T_d
Proposed autotuner	0.62	9.84	0.10
Existing autotuner	2.37	5.11	0.82

compared to the proposed autotuner. Whether this is good or bad is hard to tell and does not have a definite answer. It depends on the process itself as well as the setting it is used in. Some processes need to be controlled quickly whereas others might need slower, less aggressive control signals in order to not damage the actuators of the process. Looking at the oscillatory behavior in figure 6.4, this is typically something unwanted in the process industry and where the proposed method is providing better control. However, this is not necessarily a problem with the tuning itself but could be an effect of noisy signals or other types of disturbances. Furthermore, as only a few physical processes are evaluated, this should not be seen

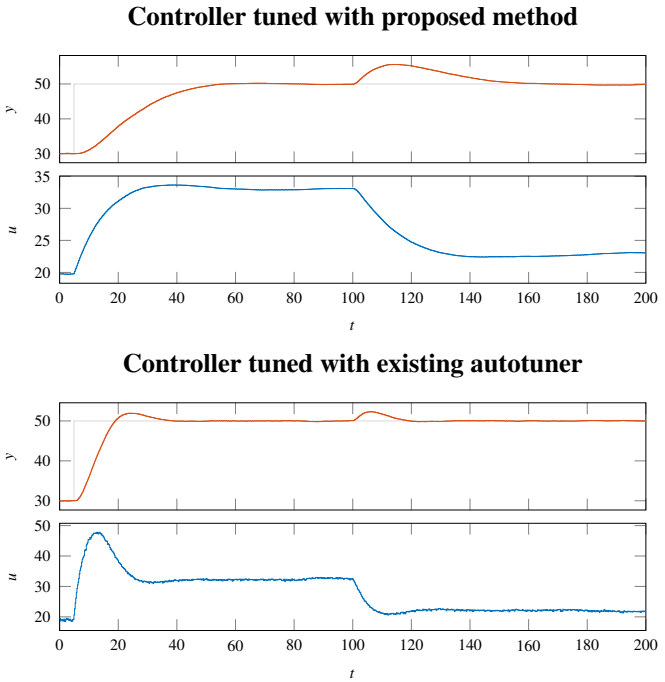

Figure 6.3 The upper tank level controlled with a PID controller using a setpoint change from 30 to 50 at time $t = 5s$ and a load disturbance of 10 on the control signal at time $t = 100s$ for the proposed autotuner as well as the existing autotuner.

Table 6.3 Lower PI

	K_c	T_i
Proposed autotuner	0.54	14.915
Existing autotuner	2.59	21.86

as a fault in the existing autotuner.

While the results are inconclusive as to whether the proposed autotuner is better than the existing autotuner or vice versa it is, however, possible to conclude that this approach can be applied both for the upper and the lower tank to design a PI or a PID controller to control either system adequately.

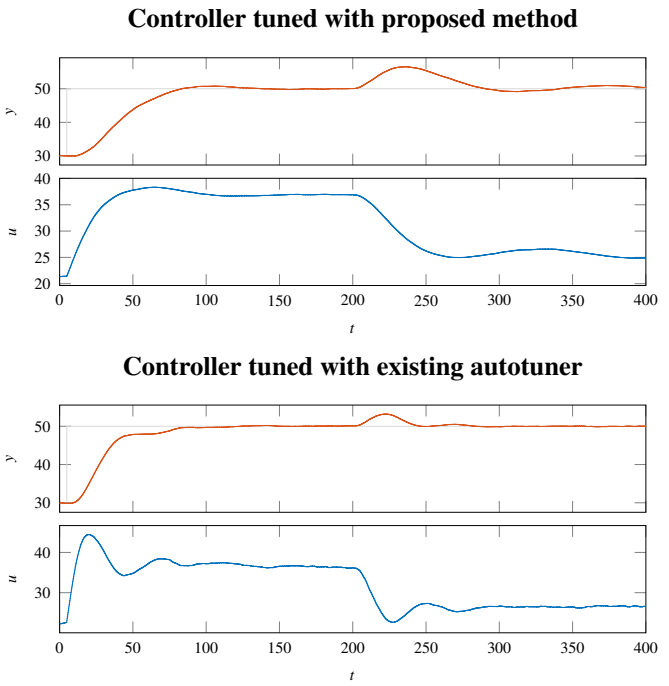

Figure 6.4 The lower tank level controlled with a PI controller using a setpoint change from 30 to 50 at time $t = 5s$ and a load disturbance of 10 on the control signal at time $t = 200s$ for the proposed autotuner as well as the existing autotuner.

Table 6.4 Lower PID

	K_c	T_i	T_d
Proposed autotuner	0.60	17.82	1.25
Existing autotuner	2.37	21.86	3.5

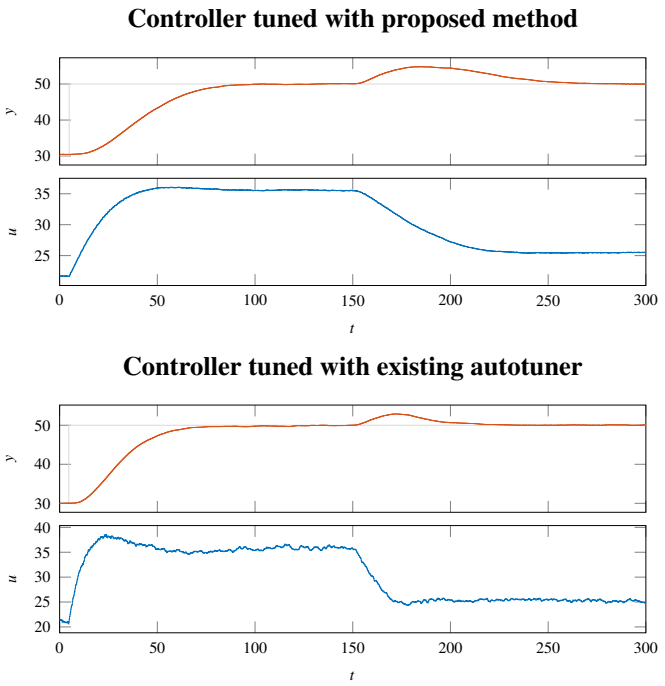

Figure 6.5 The lower tank level controlled with a PID controller using a setpoint change from 30 to 50 at time $t = 5s$ and a load disturbance of 10 on the control signal at time $t = 150s$ for the proposed autotuner as well as the existing autotuner.

7

Conclusions

In this chapter, the observations made in this thesis are summarized from which some conclusions are drawn. Future work to be done in the area with these results as a basis is then also explained.

7.1 Conclusions

In this thesis, the possibility of estimating an FOTD model in an industrial controller has been demonstrated. With the goal to develop and implement a new automatic tuner that is of sufficiently low complexity such that it could easily be running directly in an ABB AC 800M controller, it can be deemed successful.

From a short experiment using an asymmetric relay, very little memory in terms of saving the experiment data is used. Then, using a novel, lightweight optimization procedure, an FOTD model is found that is capable of capturing the important dynamics of the system. This system has then been used together with the Lambda and AMIGO tuning rules to design an adequate controller.

From simulation results, it has been proven that the proposed controller de facto is capable of controlling all processes it has been evaluated on. One question, however, could be the aspect of how fast the controller is reacting. Comparing to the existing autotuner used in the ABB controllers today, one can see that the proposed one is not responding as fast for the physical process tested on. The existing autotuner used today has, as previously mentioned, three settings; *fast*, *normal*, and *slow*. For this proposed autotuner, a similar option could easily be implemented to attain a quicker control if desired, so this should not be a concern.

7.2 Future work

Mainly, this thesis has circled around the ability to produce a framework for how the important dynamics in a process with unknown dynamics easily can be modeled. This was here used for designing a controller used to control said process. However, the low-order knowledge for a process with unknown dynamics is not only useful

for tuning the parameters of PI and PID controllers. Having this simple and "easy to find" model structure allows for an overview of the dynamics of the process which has a range of different applications in the process industry. Some future work could therefore lie in applying this approach of estimating an FOTD model to be used for more advanced controller structures such as model predictive control or to design an accurate filter. If more advanced controller structures are used, this method can, as previously mentioned, be used to provide initial insight into the process dynamics. This initial knowledge about the process with unknown dynamics can also be used as an initialization point for more advanced optimization methods where one runs the risk of getting caught in a non-global minimum.

Bibliography

Åström, K. and T. Hägglund (1984). "Automatic tuning of simple regulators with specifications on phase and amplitude margins". English. *Automatica* **20**:5.

Åström, K. and T. Hägglund (1995). *PID Controllers: Theory, Design, and Tuning*. English. ISA - The Instrumentation, Systems and Automation Society. ISBN: 1-55617-516-7.

Åström, K. and T. Hägglund (2004). "Revisiting the Ziegler–Nichols step response method for PID control". *Journal of Process Control* **14**:6.

Åström, K. and T. Hägglund (2006). *Advanced PID Control*. ISA-The Instrumentation, Systems, and Automation Society. ISBN: 9781556179426.

Berner, J. (2017). *Automatic Controller Tuning using Relay-based Model Identification*. English. PhD thesis. Department of Automatic Control. ISBN: 978-91-7753-446-4.

Control Builder (2021). URL: `https://new.abb.com/control-systems/system-800xa/800xa-dcs/hardware-controllers-io/control-builder-engineering-software` (visited on 2021-05-19).

Hansson, J. and M. Svensson (2020). *Next Generation Relay Autotuners – Analysis and Implementation at ABB*. English. MA thesis. Department of Automatic Control.

IEC (2003). *IEC 61131-3: Programmable controllers – Part 3: Programming Languages Ed2.0*. Tech. rep. International Electrotechnical Commission.

Ljung, L. and T. Glad (2016). *Modeling and identification of dynamic systems*. English. Studentlitteratur. ISBN: 9789144116884.

Pernebo, L. and B. Hansson (2002). "Plug and play in control loop design". *Preprints Reglermöte 2002*.

Wittenmark, B., K.-E. Årzén, and K. J. Åström (2002). *Computer Control: An Overview*. English. IFAC PROFESSIONAL BRIEF. International Federation of Automatic Control.

A

Test batch of processes

Model 1-21: $P_1(s) = \dfrac{e^{-s}}{sT+1}$

$T = 0.02, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 1, 1.3, 1.5,$
$2, 4, 6, 8, 10, 20, 50, 100, 200, 500, 1000$

Models 22-42: $P_2(s) = \dfrac{e^{-s}}{(sT+1)^2}$

$T = 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 1, 1.3,$
$1.5, 2, 4, 6, 8, 10, 20, 50, 100, 200, 500$

Models 43-52: $P_3(s) = \dfrac{1}{(s+1)(sT+1)^2}$

$T = 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 2, 5, 10$

Models 53-58: $P_4(s) = \dfrac{1}{(s+1)^n}$

$n = 3, 4, 5, 6, 7, 8$

Models 59-67: $P_5(s) = \dfrac{1}{(s+1)(\alpha s+1)(\alpha^2 s+1)(\alpha^3 s+1)}$

$\alpha = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$

Models 68-77: $P_6(s) = \dfrac{1}{s(sT_1+1)} e^{-sL_1}$

$L_1 = 0.01, 0.02, 0.05, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0,$
$T_1 + L_1 = 1$

Models 78-113: $P_7(s) = \dfrac{T}{(sT+1)(sT_1+1)} e^{-sL_1}$

$T = 1, 2, 5, 10,$
$L_1 = 0.01, 0.02, 0.05, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0,$
$T_1 + L_1 = 1$

Models 114-124: $P_8(s) = \dfrac{1-\alpha s}{(s+1)^3}$

$$\alpha = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1$$

Models 125-134: $P_9(s) = \dfrac{1}{(s+1)((sT)^2+1.4sT+1)}$

$$T = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0$$

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
July 2021

Document Number
TFRT-6146

Author(s)
Magnus Lundh

Supervisor
Alfred Theorin, ABB Process Automation, Sweden
Kristian Soltesz, Dept. of Automatic Control, Lund University, Sweden
Tore Hägglund, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
A new, fast, and efficient automatic tuner for the ABB AC 800M family of controllers

Abstract

The most common controller in the process industry today is the PID controller due to its simple structure. A regular factory may have thousands of different PID controllers in use where each of them has to be tuned to work satisfactory. Doing this manually can become both a difficult and time consuming task where some knowledge for each process is required. Therefore, it is highly beneficial to have a way to automatically tune the controllers, which was the motivation behind the relay autotuner first introduced in the 1980's. Since then, many alterations of the original relay autotuner have been proposed as technology development have progressed, concerning both available computing power as well as PID controllers in general.

This master thesis is proposing a new autotuner, based on a short asymmetrical relay experiment, that is running at sufficiently low cost to be executed directly in an industrial controller unit. This is highly beneficial for end customers who are to control a process with unknown dynamics.

An extensive simulation study is conducted using a test batch of a wide range of processes representative for the process industry to ensure that the approach is adequate. A suitable first order model with time delay (FOTD) is found from which a controller structure is based, following a set of well-known tuning rules.

The method is then implemented in the ABB Ability™ System 800xA where the proposed method was shown to successfully estimate a model and control a process with unknown dynamics in a robust way.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-53

Recipient's notes

Security classification