Next Generation Relay Autotuners – Analysis and Implementation at ABB

Jonas Hansson

Magnus Svensson

LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6107
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 2020

Abstract

For a long time, a method which tunes a robust controller automatically has been researched. In this thesis, a method to estimate processes using a short experiment and low order time delayed models is described. The goal was to achieve a model that could be used for PI- and PID-tuning in an autotuner. Through a simulation study, various experiment designs were tested to find an optimal experiment for model identification. The method was implemented in the ABB AC 800M controller. Using a simulation study and ABB's AC 800M controller the proposed method was shown to be successful in estimating models for a variety of processes. Furthermore, the proposed method was shown to be more robust and have a shorter experiment time than the autotuner that is currently used.

Acknowledgements

We would like to send a big thanks to all of our supervisors at the Department of Automatic Control, LTH, Kristian Soltesz, and Tore Hägglund who have helped us with everything ranging from theoretical questions to acquire the water tank process. From the Department of Automatic Control, we would also like to thank Karl-Johan Åström, who was not even supposed to be our supervisor but he did it anyway which we are very grateful for. We would also like to thank the people at ABB automation in Malmö where we have worked during this master thesis and always felt welcome. A special thanks to Alfred Theorin who was our supervisor at ABB and someone that we could always talk to and one of the biggest reasons why we always felt very welcome there. We also thank Sumit Shinde and Ulf Thomsen who have always been helpful and eager to teach us about the control systems at ABB and also helped us with the implementation of the experiment in their controller.

Contents

Nomenclature

FOTD **F**irst **O**rder model with **T**ime **D**elay

LS **L**east **S**quare

MSE **M**ean **S**quare **E**rror

SOTD **S**econd **O**rder model with **T**ime **D**elay

SOZTD **S**econd **O**rder model with **Z**ero and **T**ime **D**elay

1

Introduction

1.1 Background

ABB's current product for process automation is the ABB Ability System 800xA which includes the AC 800M controller family. This product supports a few automatic tuning methods and through new research, it is believed that the tuning methods can be improved to be more efficient and robust.

For a long time the goal of finding a robust and quick autotuner method for simple controllers has been a practical and relevant research area, see [Åström and Hägglund, 1984] where the currently used method at ABB is described. There are many reasons why this area is relevant. To begin with, the most used controllers in the world are the simple ones such as P, PI, and PID controllers where a regular factory can have thousands of controllers [Berner, 2017]. Manually tuning a controller is, in general, a hard and time-consuming procedure, even for a control engineer. Therefore, it is desirable to have a method which tunes the controllers automatically in the shortest time possible. Back when the algorithm in [Åström and Hägglund, 1984] was implemented in ABB's control library the available computing power was very limited. To cope with the limited data problem, they sought to only find the critical gain and critical period. This point is where the transfer function intersects the negative real axis in the Nyquist diagram. If this is measured accurately, it gives some information regarding the amplitude margin. To find this point they used a relay algorithm which works as an on-off controller that makes the system converge towards a steady state oscillation. From the period and amplitude of the oscillation, it was then possible to find the point of interest. A block diagram of the current autotuner is shown in Figure 1.1. However, the algorithm has some limitations. First, the system has to be stationary before it starts, so how do you define that a system is stationary? Second, the experiment time can vary even on the same system. This depends on how it is initialised. With longer experiments, the risk for disturbances in the system increases. Lastly, as the method only gives one point on the Nyquist curve and a lot of information in the experiment is not used.

As more computing power has become available, the potential to overcome the outlined limitations has become feasible, as outlined in [Berner, 2017]. In this the-

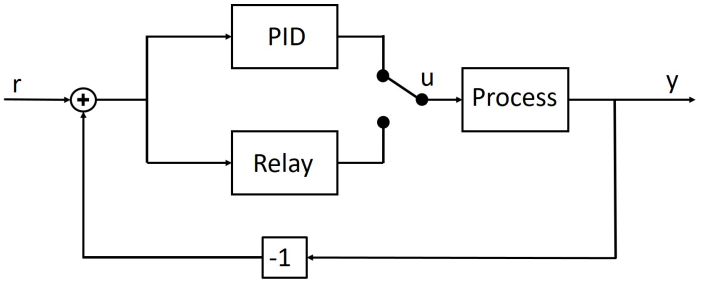


Figure 1.1: Block diagram of how the current ABB autotuner works. When it is turned on it switched the control signal from the PID to the relay signal. After the experiment has finished, the control signal is switched back to the PID controller which is tuned.

sis, a short relay experiment is described. It can be started from a non-stationary point and uses an asymmetric relay for better excitation. Here, optimisation was used to fit second order time delayed models to experimental data, which was then used to tune simple PI and PID controllers by known tuning methods. It was shown that with only three relay switches a good model could be estimated and then used to design controllers. Such a method is however currently not used.

1.2 Aim of master thesis

Because ABB is using the aforementioned autotuner in [Åström and Hägglund, 1984], this master thesis aims to implement, analyse and modify the method which was described in [Berner, 2017], and implementing it at ABB for use in their next autotuner. The goal is to have a method, with a short experiment time, which is easy to use, and provides system models which can be used for more complex control structures such as MPC in the future. To achieve these goals the plan was to do a simulation study on a large test batch of transfer function models where we try the experiment and then compare the estimated models to the correct model. By implementing the experiment in ABB's Compact Control Builder for AC 800M we want to compare the current method with the new one on real processes and see how they differ in performance.

1.3 Strategy

To surpass the performance of the current ABB autotuner, the strategy was to follow the methods outlined in [Berner, 2017] with some modifications. To begin with, the overall goal was to identify the best possible low order models. The three chosen models to identify were a first order model with time delay (FOTD), a second order

model with time delay (SOTD), and a second order model with zero and time delay (SOZTD). The identification depends on both the experiment design and on the algorithm used. To optimise the experiment, the plan was to use an asymmetrical relay to excite a wider spectrum of frequencies in comparison to the current method, which uses a symmetric relay. To determine how the experiment should look, the number of relay switches were varied to show the impact on the identified models. Furthermore, a chirp was added with varying length to test if a chirp improved the model identification. To estimate models using the data from the experiment an optimisation method was used, however, from [Berner, 2017] it was known that a problem that occurs when doing low-order model identification is bad initialisations of the optimisations. A bad initialisation can lead to a local minimum in the optimisation, which may represent the real system poorly. To solve this problem the strategy was to use smoothed local cubic splines on the sampled data and then perform a least-squares based parameter estimation to find an approximate solution. From this solution, local gradient-based optimisation methods were used to find a final optimal model. To assess the performance the Vinnicombe metric, a proposed weighted mean square error (MSE), the Bode plot, and the Nyquist diagram were used.

2

Theory

To understand the method used in this master thesis some theory is needed. This chapter is divided into several sections and presented in the same order as it is used, starting with the explanation of what a relay and a chirp are in section *Input signal*. The asymmetry in the relay is also explained here, together with the exponential start of the relay. This is followed by the section *Variance estimation and filter design*. The third section *Optimisation* explains the theory behind estimation of the low order models with focus on the initialisation of the optimisation. The last section *Comparison* covers the theory used to compare the estimated models to each other and the process model the experiment was performed on.

2.1 Input signal

Relay and chirp

One of the most important parts of this thesis was to evaluate what control signal to use. In Figure 2.1 an example of what an experiment can look like is shown. The basic idea of a relay experiment is to use a control signal to make the system oscillate. This is achieved by switching between two input signal levels. For example, in Figure 2.1 the two input signal levels are 5 and -10. When noise is present in the measurement signal it can be hard to determine when the relay should switch. To increase robustness a hysteresis level is introduced, which is a threshold used to determine whether the relays should switch or not. Looking at Figure 2.1 it is clear that the control signal u switches when the measurement y reaches the green hysteresis lines during the first half of the experiment. With this in mind a more formal definition of a relay experiment can be formed, namely:

Definition 2.1.1. *Relay: The relay is started when the control signal is set to a constant value, added to the starting value, to push a measurement signal past a hysteresis level. When the signal has reached the hysteresis the relay switches the control signal to push the measurement signal past a lower hysteresis level. Then, the relay switches the control signal back to its previous value and so on.*

With the relay defined, let us also define what is meant by a chirp and why it is used. The simplest form of chirp is a sinusoidal signal with a time-vary frequency:

$$u_{Chirp}(t) = A\ \sin(t\ \omega(t)). \tag{2.1}$$

Where A is a constant amplitude, t is the time, and $\omega(t)$ is a time dependant angular frequency. A chirp is often used within system identification as it excites a range of frequencies. The relay experiment primarily excites frequencies around its self-oscillation which is called the critical frequency where the phase is -180°. With a chirp, the intention is to excite lower frequencies to make the process output display underlying dynamics better. To incorporate this into a live experiment, a simple method for identifying the critical frequency is needed. In this project, and in particular for the experiment which is shown in Figure 2.1, this is identified by looking at the longest time between two subsequent relay switches during the relay experiment to identify an approximate half-period time. By taking the corresponding frequency, a lower bounded estimate of the critical frequency $f_{critical}$ is found. From this critical frequency estimate, a chirp could be designed in multiple ways. With the motivation that lower frequencies were desired the angular frequency $\omega(t)$ in Equation 2.1 was formed as

$$\omega(t) = 2\pi \frac{\frac{f_{critical}}{2} \cdot t}{T_{period} \cdot ch}. \tag{2.2}$$

Here t the time from the start of the chirp and ch is the number of relative chirp lengths. Where one relative chirp length is the same time as the longest time between two subsequent relay switches. By the definition in equation 2.2 the angular frequency increases linearly from $\omega(0) = 0$ to $\omega(ch \cdot T_{period}) = 2\pi f_{critical}/2$, which is the same as the frequency ranging from zero to $f_{critical}/2$. One property of designing the chirp in this way is that the relative chirp length ch gets a nice relation to the number of relay switches. As for each length of extra chirp, the experiment time is increased proportional to the longest time between two subsequent relay switches, T_{period}.

Exponential start and asymmetrical relay

Looking at Figure 2.1 an exponential start can be seen in the relay. The exponential growth was included to catch processes that are sensitive to small deviations in the control signal. If the hysteresis level is reached before the exponential has finished, then the upper control signal amplitude is adjusted to the current control signal and the lower control signal is adjusted to keep the asymmetry. Having a lower amplitude on the second relay sometimes failed because the control signal being too small to push the signal past the hysteresis. Therefore, the first relay was set to be the smaller one. By using an asymmetric relay the excitation of the experiment was increased. Hence, the system identification became more accurate. This followed

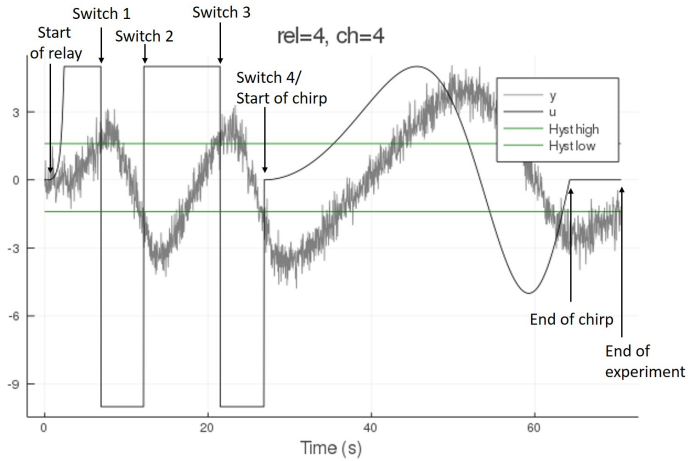


Figure 2.1: An example experiment with four relay switches and a chirp length of four. Here, y is the measured signal, u is the control signal, $Hystlow$, and $Hysthigh$ are the lower and upper hysteresis levels used for the relay experiment.

the work in [Berner, 2017] and used the asymmetry factor $\gamma = 2$. This can be seen in Figure 2.1 where the control signal u varies between $[-5 \cdot \gamma, 5] = [-10, 5]$.

2.2 Variance estimation and filter design

As the Experiment design was very central for this thesis, below is the derivation of how the variance is estimated and how the filter is designed to give a desired variance. This is used to set the hysteresis and achieve accurate relay switches. In most of the cases where autotuning and model identification are wanted it is reasonable to assume that one desires to find a model around a setpoint y_s with the corresponding control signal u_s which generates this setpoint. As it can be difficult to identify when the system is stationary, it is not desirable to demand complete stationarity before the experiment is started. For our test, the measured signal is assumed to locally follow a linear trend with noise. Hence, the measurement can be written on the form

$$y(t) = a \cdot t + b + v(t) \tag{2.3}$$

where a and b are two arbitrary constants and $v(t)$ is a zero mean white noise process with variance $V[v(t)] = \sigma_v^2$. With these assumptions one can form the one step difference

$$e(t) = y(t) - y(t-h). \tag{2.4}$$

From this expression it can be derived that the variance of the process $e(t)$ will be

$$\begin{aligned} V[e(t)] &= V[y(t) - y(t-h)] \\ &= V[a \cdot t + b + \nu(t) - (a \cdot (t-h) + b + \nu(t-h))] \\ &= V[a \cdot h + \nu(t) - \nu(t-h))] \\ &= 0 + \sigma_\nu^2 + \sigma_\nu^2 \\ &= 2\sigma_\nu^2. \end{aligned} \tag{2.5}$$

Hence, the underlying variance in the measurement signal y can be estimated by estimating the variance of the one-step difference. In this project the variance was chosen to be estimated recursively using Welford's online algorithm see [Welford, 1962]. The reason for using this method is that very little data is needed to be stored. This is summarised in Algorithm 1, where the the parameters M and S are internal parameters that are updated during the iterations.

Algorithm 1 Welford's online algorithm for iterative variance (s^2) estimation [Welford, 1962]. The "$\leftarrow$" indicate an assignment and the variance is estimated for the process x where x_k is the kth sample of a total of n samples.

for k in 1:n **do**
  $oldM \leftarrow M$
  $M \leftarrow M + (x_k - M)/k$
  $S \leftarrow S + (x_k - M) \cdot (x_k - oldM)$
end for
$s^2 \leftarrow S/(n-1)$

With the derivation of Equation 2.5 in mind, it can be useful to consider the legitimacy of the calculations. Through our experience the assumption of local linearity works well and catches the variance quite well.

With the variance estimated it is time to construct a filter, which sets our filtered measurement signal at a desired variance. The motivation for doing this was to achieve more precise switches in the relay. The chosen filter is of the form

$$\hat{y}(t) = a\hat{y}(t-h) + (1-a)y(t) \tag{2.6}$$

where $a \in [0,1)$ is a constant. This filter was chosen as it can set the filtered measurement signal at a desired variance. The variance of $\hat{y}$ can be found by solving the following equation

$$V[\hat{y}(t)] = V[a\hat{y}(t-h) + (1-a)y(t)]. \tag{2.7}$$

By assuming that the linear tendencies of $y(t)$ are small compared to the variance of $\nu(t)$ in Equation 2.3, that the variance is time invariant for $\hat{y}(t)$, and that $\mathrm{Cov}[\hat{y}(t-h), y(t)] = 0$ the previous equation simplifies to

$$V[\hat{y}(t)] = a^2 V[\hat{y}(t)] + (1-a)^2 V[y(t)]. \tag{2.8}$$

which can be simplified further to

$$V[\hat{y}(t)] = \frac{(1-a)^2}{1-a^2} V[y(t)]. \tag{2.9}$$

From this expression, it is easy to identify an expression which grants a filtered signal with arbitrary small variance. The desired variance σ^2_{des} of the filtered signal can be obtained by setting a to the solution of the following equation.

$$(1-a^2)\sigma^2_{des} = (1-a)^2 \sigma^2_v \tag{2.10}$$

The equation has two solutions the trivial $a = 1$, which is unuseful, and

$$a = \frac{1-v}{1+v}, \tag{2.11}$$

where $v = \sigma^2_{des}/\sigma^2_v \leq 1$.

2.3 Optimisation

Given that the data has been sampled the next big question is: how should a model be estimated using the data? We have chosen to follow [Berner, 2017] in the sense that the aim is to fit low-order models to match the sampled data. The chosen models were

$$\begin{aligned} P_{FOTD} &= \frac{b_0}{s+a_0} e^{-Ls} \\ P_{SOTD} &= \frac{b_0}{s^2+a_1 s+a_0} e^{-Ls} \\ P_{SOZTD} &= \frac{b_1 s+b_0}{s^2+a_1 s+a_0} e^{-Ls}. \end{aligned} \tag{2.12}$$

These models were chosen as they are of low orders which made them suitable for PID design.

To fit models optimisation algorithms implemented by [Steven G. Johnson, 2020] were used. They are based on [Svanberg, 2002] and [Kraft, 1988] and were available in NLopt for Julia. These methods are used because they are already implemented, tested, and available in Julia.

One problem which appeared when using optimisation was that most algorithms could only guarantee a local minimum given its existence whilst it is desired to find something as close to the global minimum as possible. Thus, this method requires a good initial guess to find something close to the global minimum. To find a good

initial guess, a few options were considered. Firstly, one could use a global optimisation method. The disadvantage with this is that it is hard to know when it would work. A second approach, which was used in [Berner, 2017], is to use multiple random guesses and then use the best one for initialisation. This was evaluated and we experienced problems with both computation time and with local minima. A third was to discretise the continuous-time models using a simple method such as Euler forward differentiation or Tustin's method and then identify a discrete-time model using least squares (LS). The identified discrete model can then be converted back to a continuous one which can then be used as an initial value for the optimisation. This was tried by identifying the discrete-time model using LS and a linear search for the best time delay. The method worked sometimes but could also give very bad models from time to time, especially when there was noise present. With the experience from these experiments, we came up with the idea to use smoothed interpolating cubic splines which have the attractive property of being continuously differentiable up to their second derivative [Michiel, 2001]. To make the splines' smooth a minimisation problem which penalises model complexity was introduced. The function to minimise was

$$J_\alpha(g) = \alpha \sum_{i=1}^{N} (y_i - g_i)^2 + (1-\alpha) \int_{t_0}^{t_n} g''(t)^2 dt \tag{2.13}$$

where g is a cubic spline, $\alpha \in (0,1]$ is a constant, y_i is the measured value at the time instance t_i. The first term in Equation 2.13 rewards a highly accurate model and the second term penalises a complex model. Given a fix α it is possible to find the cubic spline $g^{opt}(\alpha)$ which minimises $J_\alpha(g)$. To get the best cubic spline with respect to α, cross validation was used. The cross validation was performed using folds of size 150 and where every 10th sample was left out. An overlap of 60 % was also used to not miss any important dynamics. The performance was assessed based on which cubic spline made the best prediction on the left out samples. Using the results of the cubic splines' exact derivatives and least squares it is possible to estimate models on the form

$$y''(t) + a_1 y'(t) + a_0 y(t) = b_1 u'(t-L) + b_0 u(t-L) \tag{2.14}$$

and all forms which only use derivatives of lower order. Note that to estimate higher-order models using the same method one would need to use polynomials of a higher order. To further motivate why the smoothed cubic splines are used; in Figure 2.2 the estimated value, first, and second derivatives are shown compared to the measured data for a typical experiment. As can be seen in the figure, the higher the order of estimated derivative the noisier the estimate is. But the important part is that the estimated derivatives do not seem to be nonsensical and by using these estimates in a least-squares estimation one could get usable results for a rough initial guess. This can then be compared to computing the derivatives using Euler forward difference $\hat{y}'_n = (y_n - y_{n-1})/h$ and $\hat{y}''_n = (y_n - 2y_{n-1} + y_{n-2})/h^2$. The result

is shown in Figure 2.3. In the figure it is easy to see that the Euler forward difference gives very noisy estimates of the derivatives which is unsuitable for LS. However, the cubic spline permits this kind of initialisation as this method handles the noise nicely.

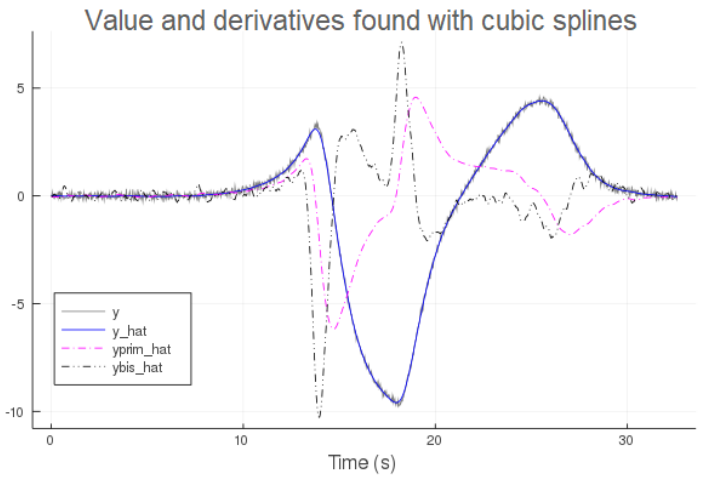


Figure 2.2: A smoothed cubic spline fit to match experiment data is shown. "Prim" denotes the first derivative, "bis" the second derivative and all values with "_hat" relate to the cubic spline.

One problem, which could occur when using LS on the cubic splines, was that the estimated system could be unstable and/or have negative stationary gain even when this was not making sense. To handle these problems, all unstable poles are mirrored in the imaginary axis and positive stationary gain is enforced. This is motivated as the experiment was not designed to work on unstable systems and to force the stationary gain is necessary to identify non-minimum phase systems with a right-half-plane zero. By mirroring the poles and by forcing positive stationary gain, the amplitudes of the estimated transfer functions were preserved while the phases were changed.

2.4 Comparison

When the optimisation has been run on data from an experiment one problem remains: how should the resulting models be evaluated? Throughout this project the following methods have been used to compare results:

- The fit itself; does it look visually good and is the MSE low?

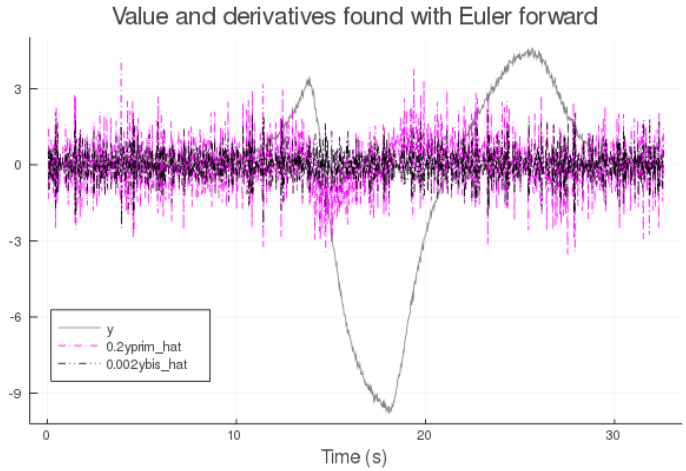


Figure 2.3: Process value, first, and second derivative found using Euler forward difference on experiment data. "Prim" denotes the first derivative, "bis" the second derivative and all values with "_hat" relate to values found with Euler forward difference. Note that the derivative values are scaled to fit in the figure

- Bode and Nyquist diagram; where is the fit good in the frequency domain?
- The ν-gap metric; how close is the estimated model to the real?

As the MSE indicates if the optimisation has been successful of fitting a curve to the data, this is used to assess whether the curve match is good or not. Furthermore, as it is easy to calculate the MSE, it was used as a measure for performance between the different models that were fitted. However, it is better to have models of lower order as it reduced the risk of overfitting. Therefore, a small proportional penalty to the MSE was added to promote the choice of lower-order models when the results are similar. From now on, when referring to MSE, what is meant is

$$MSE \triangleq (1+0.01(p-3))\frac{(\hat{y}-y)^T(\hat{y}-y)}{n} \tag{2.15}$$

where p is the number of parameters used in the model, and n is the number of elements in the measured data vector y. Hence p for FOTD is 3 as it has one parameter in the denominator, one in the numerator, and the time delay. By the same argument SOTD had $p = 4$ and SOZTD $p = 5$.

By only looking at MSE it is hard to get the full picture, which is why the frequency domain is used as well. Especially for situations when the real process is known which is the case when the data is simulated. Here Bode and Nyquist diagrams are used to analyse the frequency domain. The reason for using these are that they are commonly used within the control community and they are often easy

to use to get a visual intuition of how good identified models are. Another method to analyse the frequency domain is the Vinnicombe metric, also referred to as the ν-gap metric [Vinnicombe, 2001]. This is used to get an easily comparable measure of how well an estimated model matches the frequency properties of a known system without visually analysing plots. Here the geometrical interpretation of the metric is outlined. Figure 2.4 shows the projection of two complex points on a Riemann sphere. The Riemann sphere is a sphere with 1 unit in diameter located atop the complex plane with its centre located 0.5 units above the origin as depicted in the figure. Given two transfer functions $P_1(s)$ and $P_2(s)$ the Nyquist diagram maps their frequency response $P_x(i\omega)$ to the complex plane. The ν-gap metric can be visualised as the projection of the Nyquist diagram onto a Riemann sphere as visualised in Figure 2.4. If the projection of $P_1(i\omega)$ onto the Riemann sphere is parameterized as $PR_x(\omega)$ then the metric can be defined as

$$\nu_{gap}(P_1,P_2) = \sup_{\omega\in\mathbb{R}} |PR_1(\omega) - PR_2(\omega)| \tag{2.16}$$

where $|x_1 - x_2|$ is the Euclidean distance between the two points x_1 and x_2.

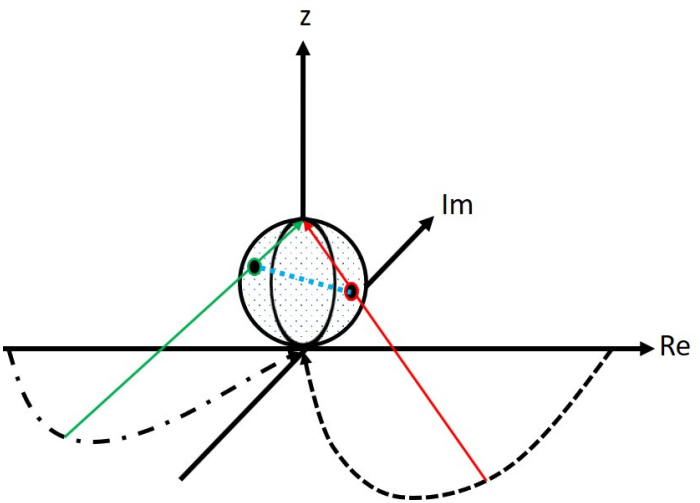


Figure 2.4: An illustration of the ν-gap metric. The Nyquist diagram of two transfer functions are shown (dashed and dash-dotted) and for a given frequency two straight lines (red and green) are drawn from the Nyquist curves towards the top of the Riemann sphere. The distance between the intersections is shown in light blue. The largest distance among all frequencies gives the ν-gap metric.

3

Algorithm

In this chapter, the algorithm is described in the order the experiment is performed, starting with the signal used to excite the system, followed by how the initialisation of the optimisation was done which then leads to the optimisation. Finally, the method of how the program chooses the model that should be used for PI and PID tuning is described.

3.1 Input signal

The input signal we used could be divided into four different parts. The first part was very short and the purpose was to estimate the variance of the noise in the signal so that a filter could be designed accordingly. This was done while keeping the control signal constant. The constant signal was the same as the last used input signal to the process before the experiment started, as this is the working point chosen by the operator. The variance of the noise was estimated iteratively according to Algorithm 1. From the estimated variance a first-order filter was designed using that the desired standard deviation was set to a tenth of the estimated standard deviation and then the filter was formed using Equations 2.5, 2.6 and 2.11. With the filtered signal, accurate and consistent relay switches were experienced. Also based on the noise estimation, the hysteresis was set to be three times the standard deviation of the estimated noise. The benefit of this was that the hysteresis then become large enough to give the signal a good signal to noise ratio. During this short start of the experiment, there is a side benefit of the process output getting closer to stationarity if it was not already there before the start.

The second part consisted of the asymmetric relay from [Berner, 2017] which gave good excitation in the area of the Nyquist curve close to the critical point, -180° phase lag. The asymmetric relay was set around the constant signal used in the noise variance estimation. Then, from this signal the first relay amplitude was set to (constant signal) +5% of the possible interval for the input signal, this was the positive relay signal. The second amplitude was set to (constant signal) -10% of the possible interval for the input signal, this was the negative relay signal. An

additional feature was the exponential start of the relay, as explained in the *Theory* Chapter. If the exponential start reached the hysteresis before it had reached its maximum amplitude, the amplitude of the relay was reduced. The positive relay amplitude was reduced to the same amplitude as the control signal had reached in the exponential start, and the negative relay amplitude to twice that but negative. The reason for the exponential start and amplitude reduction was that some processes could take damage from high amplitude relays. However the amplitudes could not be zero, so a lower bound was set to -2% for the negative and +1% for the positive. Two important notes here are that the relay shifted when the filtered signal reached the hysteresis, but the signal that was sent out from the experiment was unfiltered. Hence, the filter was only used internally to make the relay switches more accurate and consistent. The second note is that in the final implementation two relay switches are used but more were tested before deciding for this.

The third part was a chirp, which is a sinusoidal signal with time dependant frequencies. This part is an extra feature to the signal, with the idea that if the operator would like an extra good model and a long experiment is not a problem, this could be added. The purpose of the chirp was to extend the range of frequencies which were excited. The frequency range was chosen to go from zero up to half of the lowest frequency found from the time lengths between the relay switches and the amplitude was set to the same as the smallest of the relay. By doing this the excitation should be better in the third and fourth quadrant of the Nyquist plot, namely the frequency area where the real model has a phase lag of 0° to -180°. This frequency area can be of great interest when designing a PID, especially when margin to the -1 point in the Nyquist diagram is of interest. The length of the chirp was decided from the length of the longest time between switches in the relay, so one unit of chirp length is the same length as the longest time between two subsequent relay switches. In the final implementation of the experiment, it was decided to use a chirp length of 2 units.

The last part consisted of the same constant signal as the one used in the noise variance estimation. This part was active for the same period of time as the time from the start to the first relay switch. The purpose of this was to consider time delays to ensure that the experiment would not end before the effects of the relay or chirp had finished. An example control signal of four relay switches and four units of chirp can be seen in Figure 2.1.

3.2 Initialisation

Before explaining the initialisation and the optimisation it is important to know what the parameter vector looks like. For this project the following was used

$$\left[A, B, L, x_0, u_{off}\right] \tag{3.1}$$

where A was the transfer function denominator coefficients, B the numerator coefficients, L the time delay, x_0 the initial values of the estimated process internal states, and u_{off} the offset between input and output.

It was important for the optimisation to have good initial values, otherwise, there was a risk of it getting stuck in a local minimum away from the global minimum. To start the initialisation the data were downsampled to 300·((relay switches)+(chirp length)) samples and the data were set around zero, to make the data easier to process. The data were set around zero by taking the control signal data subtracted to the first value of the data and the measurement data subtracted to the value it had after the noise variance estimation. The reason for not subtracting the first value of the output was that if the process was not stationary when the experiment started, then it had the period of the noise estimation to get closer. The closer it was to stationarity the better the optimisation was. Because of noise and not being stationary, there could be an offset between input and output after being set around zero. u_{off} is used to compensate for this.

To get good initial values we used smoothed cubic splines on the input and output of the process to get continuous data. From the continuous data, it was possible to get the derivative and second derivative which was used in an LS estimate. The LS was tested for several different time delays starting from zero up to the time between the start and the first relay switch in steps of one sample time. The time between the start and the first relay switch was set as an upper bound since the time delay can not be longer than this, otherwise the switch would not happen. The LS-model with the best MSE compared to the real output was then used as the initial parameter values. This gave an estimation of the initial values of the model denominator A, numerator B and the time delay L. The other parameters' initial values in the optimisation vector 3.1 were set to zero.

In this master thesis, stability and the sign of the static gain were assumed to be input parameters from the user. If the user did not change these parameters the default settings were stable and a positive static gain. In the initialisation, this meant that if the estimated transfer function was unstable, the poles were mirrored and the numerator was set to be positive. In the case of stable estimated models with negative static gain, the numerator was changed to be positive for FOTD and SOTD. For SOZTD, two cases were tested against each other. First, the numerator was multiplied with -1 to handle cases of zero in the right-half-plane. Second, the entire numerator was changed to be positive. Both MSEs were tested against each other and the lowest remained to be used. In the case where the entire numerator was negative from the start, there were no differences between the methods.

A problem could occur when the model to be estimated was a first-order model or second-order model with a big difference in the poles making it a first-order model in practice. The estimated SOTD and SOZTD had a hard time since they tried to estimate a model of lower order than they were themselves. Therefore, when the weighted MSE was higher for SOTD and/or SOZTD than for the FOTD an extra initialisation was performed using the FOTD estimation according to

$$
\begin{aligned}
P_{FOTD} &= \frac{b}{s+a}e^{-Ls} \\
P_{SOTD} &= \frac{b \cdot a \cdot q}{(s+a)(s+a \cdot q)}e^{-Ls} \\
P_{SOZTD} &= \frac{b \cdot (s+a \cdot q)}{(s+a)(s+a \cdot q)}e^{-Ls}.
\end{aligned}
\tag{3.2}
$$

where q is 1000, to make the pole that was added much faster than the original. a, b and L are the FOTD parameters. The result from this was that SOTD and SOZTD were FOTD with an extra very fast pole (and zero). The fast pole was good because the dynamics from it hardly affected the model and the estimated SOTDs and SOZTDs could estimate FOTDs well. The initial parameters from this were compared to the ones from LS using MSE and the one with the best MSE was then used in the optimisation.

3.3 Optimisation

The optimisation consisted of three optimisations using two different local gradient-based optimisations from [Steven G. Johnson, 2020]. The first optimisation used the MMA method in [Svanberg, 2002], the second used the SLSQP method in [Kraft, 1988] and [Kraft, 1994], and the last one was the same as the first. The reason for doing it in three parts was because sometimes the optimisation stopped before it reached the minimum. The reason was thought to be the difference in the size of the different parameters; for example, x_0 can have values of order 10^{-4} while A is six orders of magnitudes larger. This makes the gradient in the optimisation too small to handle x_0. This also meant that the steps that the optimiser took were too small for A to converge. To handle this problem all parameters were used in the first and the last optimisation, but in the second x_0 and u_{off} were not used to let the possibly bigger values in A, B and L converge.

3.4 Comparison

When the optimisation for the three different models was finished the models were compared using a weighted MSE according to Equation 2.15. Hence, the cost for FOTD was $1.00 \cdot \text{MSE}_{FOTD}$, for SOTD $1.01 \cdot \text{MSE}_{SOTD}$ and for SOZTD $1.02 \cdot \text{MSE}_{SOZTD}$. This was to compensate for overfitting which happened sometimes before this was included. The one with the lowest weighted MSE is then chosen as the best model and to be used for controller design.

3.5 Implementation in ABB AC 800M controller

A part of this master thesis was to implement the new algorithm on ABB's AC 800M controller, so it could be used in the next autotuner by ABB. The program was written in a development environment called Compact Control Builder which supports the IEC standard 61131-3, containing five programming languages [IEC, 2003]. Of these five, Structured Text was used in combination with an ABB specific diagram language called Function Diagrams.

The Function Diagram is shown in Figure 3.1. The diagram contained the connection to the real process by input and output blocks with a PID-control block in between. The PID was used to control the process during normal operations. In the Function Diagram were also the blocks for the communication out of the controller to the Python-script.

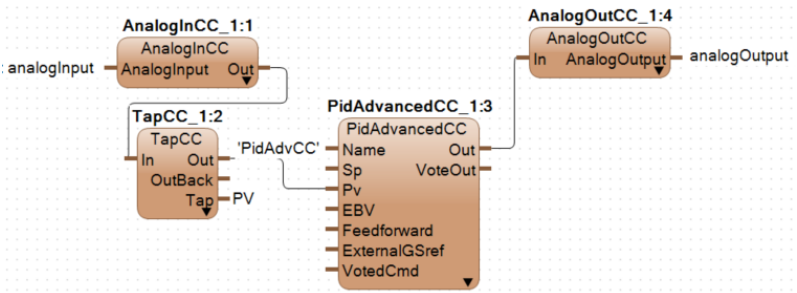


Figure 3.1: The Function Diagram implemented in Compact Control Builder which was used to perform experiments through the AC 800M Controller. The figure shows the conncetion from the measurement, which enters as an analog input, through a PID controller, which creates a control signal that is sent as an analog output to a process.

In Structured Text, the experiment was implemented. By setting a parameter equal to true in the program, the PID is put in manual mode and its manual control signal value is controlled from the experiment which was generated in Structured Text. Hence, the experiment signal is generated in the AC 800M controller, while the optimisation was implemented in Julia and performed on an external computer.

The way this works when running with a real process is that the Compact Control Builder code is downloaded from an external computer to a control unit. Then, when this control unit is operating, the control signal is generated in the AC 800M program and sent through an IO unit which is connected to the real process. The output of the process is sent back to the controller through another IO unit so it can be used to calculate the next control signal. While our experiment is running the control signal, from the PID, is controlled by the experiment algorithm in Structured Text and the control signal, time and measured output are sent out using User

Datagram Protocol (UDP). The sent out data is received by a UDP-server implemented in a Python-script and written to a text file. The modelling and PID-tuning is then performed with Julia code. To change the settings in the controller, based on the results in Julia, an operator must do it manually. An illustration of how the communication works can be seen in Figure 3.2.

Having the experiment implemented as stand-alone from the modelling/PID-tuning is new, compared to the current method at ABB, and has its pros and cons. The pros are that external computer power can be used during the optimisation which means that the execution time can be shorter, it does not disturb the other applications in the controller, it can be updated without having to upgrade/reconfigure the controller and there are more (free) available optimisation tools in the PC world. The cons are that the control unit must be able to communicate with external computers. And with external communication follows a security risk. Furthermore, in case of network issues samples can be lost which can impact the optimisation.

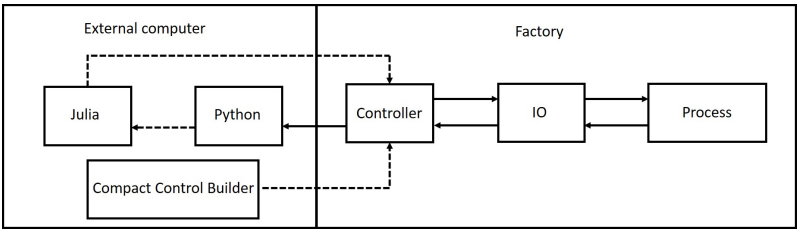


Figure 3.2: An illustration of the communication when using AC 800M controller on a real process. The solid lines represent communications performed automatically during operations, while the dashed lines represent communication made manually by an operator. The IO communicates with the process through voltages and currents. a voltage in the range 0-10 V.

4

Method

During this master thesis, several tests have been carried out to analyse and improve the implementation. Some of the biggest decisions regarded how many relay switches should be used, if the chirp should be used or not and how long the chirp should be if used? To do this, a batch consisting of 134 models was used, see equation 4.1, containing processes suitable for PID control encountered in process control. This batch was presented in [Åström and Hägglund, 2005] and it includes lag-dominated, delay-dominated, balanced and integrating processes. Using the model batch, three different combinations of relay switches and chirp lengths were tested and compared. The three different configurations were: two relay switches and no chirp, two relay switches and a relative chirp length of two, and the last one was three relay switches and no chirp. These configurations were chosen as two relay switches is the shortest possible asymmetric relay. Three different methods of comparison were used to see how good the estimations were and if the system selected the best model. The first was MSE and a curve fit plot. This indicated how well the optimisation had succeeded with fitting to the output. However, they did not guarantee a good model. The second one was the Vinnicombe metric between the correct model and the estimated one. This was done for all the different estimated models of the types FOTD, SOTD and SOZTD. The lower the Vinnicombe metric was, the closer to the real model the estimated model was. However, a large value of the Vinnicombe metric did not necessarily mean that the model was bad but a low value indicated a good model. Using our experience a Vinnicombe metric of 0.1 was set as the benchmark to be lower than. If it did not succeed with this the third method was used. This method consisted of plotting the estimated and the correct transfer functions in the same Bode and Nyquist plot for manual visual comparison. The simulations on the test batch were done using noise with a standard deviation of 0.5, input signal range was set to be from 0 to 100, 5 ms sampling time and three different random seeds for each test configuration of relay switches and chirp length.

$$
\begin{aligned}
\text{Models 1-21:} \quad & P_1(s) = \frac{e^{-s}}{1+sT} \\
& T = 0.02, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 1 \\
& \quad 1.3, 1.5, 2, 4, 6, 8, 10, 20, 50, 100, 200, 500, 1000 \\
\text{Models 22-42:} \quad & P_2(s) = \frac{e^{-s}}{(1+sT)^2} \\
& T = 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 1 \\
& \quad 1.3, 1.5, 2, 4, 6, 8, 10, 20, 50, 100, 200, 500 \\
\text{Models 43-52:} \quad & P_3(s) = \frac{1}{(s+1)(1+sT)^2} \\
& T = 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 2, 5, 10 \\
\text{Models 53-58:} \quad & P_4(s) = \frac{1}{(s+1)^n} \\
& n = 3, 4, 5, 6, 7, 8 \\
\text{Models 59-67:} \quad & P_5(s) = \frac{1}{(1+s)(1+\alpha s)(1+\alpha^2 s)(1+\alpha^3 s)} \\
& \alpha = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 \\
\text{Models 68-77:} \quad & P_6(s) = \frac{1}{s(1+sT_1)} e^{-sL_1} \\
& L_1 = 0.01, 0.02, 0.05, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0, \quad T_1 + L_1 = 1 \\
\text{Models 78-113:} \quad & P_7(s) = \frac{T}{(1+sT)(1+sT_1)} e^{-sL_1}, \quad T_1 + L_1 = 1 \\
& T = 1, 2, 5, 10 \quad L_1 = 0.01, 0.02, 0.05, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0 \\
\text{Models 114-124:} \quad & P_8(s) = \frac{1-\alpha s}{(s+1)^3} \\
& \alpha = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1 \\
\text{Models 125-134:} \quad & P_9(s) = \frac{1}{(s+1)((sT)^2 + 1.4sT + 1)} \\
& T = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0
\end{aligned}
\tag{4.1}
$$

Another test was to use our method on a water tank process. The water tank process was a real process borrowed from the Department of Automatic Control at LTH. The water tank process consists of two tanks placed on top of each other. A pump moves water into the upper tank, which has a hole at the bottom making the water flow down into the second tank below. The lower tank also has a hole at the bottom. In this setup the water level could be measured in both tanks and the voltage to the pump could be controlled. A photo of the setup can be seen in Figure 4.1. On this process the combination of two relay switches and no chirp was tested on the upper tank process and the combination of two relay switches and a relative chirp length of two on the lower tank.

As there was no available model of the process, a longer experiment was conducted using five relay switches and a chirp length of four on the upper tank. On the lower tank an experiment with four relay switches and a chirp length of two was performed. Doing these experiments was thought to ensure accurate models of the process to compare with.

Figure 4.1: A picture of the tank process used.

To compare our method, the current autotuner was run on the water tank system as well. The current autotuner allows the user to choose between two different tuning methods. The first method only uses a relay experiment. The second method uses a relay experiment followed by a closed loop reference step using PID-parameters from an earlier relay experiment. Since the current method does not

estimate a full model the methods were compared using the following: experiment time, a reference step test, and a load disturbance test. These tests compared PI-parameters obtained from the current autotuner with the AMIGO method [Åström and Hägglund, 2005] on a model from our model estimation.

5

Result

In this chapter results from a simulation study and experiments done on a real water tank process will be presented.

5.1 Simulation study

The Vinnicombe metric between the estimated and the real model, for each of the 134 models in the test batch can be seen in Figure 5.1. The figure shows the results from three runs with different random seeds. The test was performed for three different combinations of relay switches and chirp lengths. The average Vinnicombe metric from the three different random seeds is plotted in Figure 5.2 showing the difference between the different combinations of relay switches and chirp lengths that were evaluated.

In Figures 5.3, 5.4, and 5.5 the output, Bode, and Nyquist plots of model 41, in the batch, are shown for the different combinations of relay switches and chirp lengths. Model 41 has the transfer function

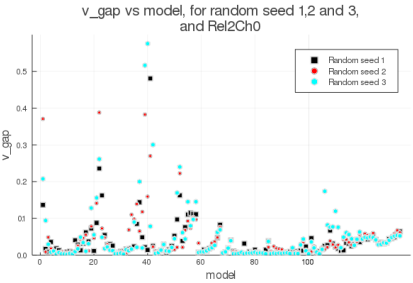
$$P_{41} = \frac{e^{-s}}{(1+200s)^2}. \tag{5.1}$$

The reason for showing this model is that it was one of the models with the largest difference in the Vinnicombe metric between the different relay and chirp combinations. By looking at the Bode and Nyquist diagrams in the Figures 5.3, 5.4, and 5.5 one can see that the estimated models are all close to the real process around the -180° phase lag which is thought to be the most important part for control design. The reason why model 41 had a relatively high Vinnicombe metric was the low frequency fit.

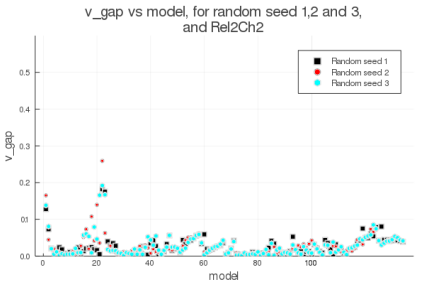
Another model which also had a high Vinnicombe metric was model 1, which has the transfer function

$$P_1 = \frac{e^{-s}}{1+0.02s}. \tag{5.2}$$

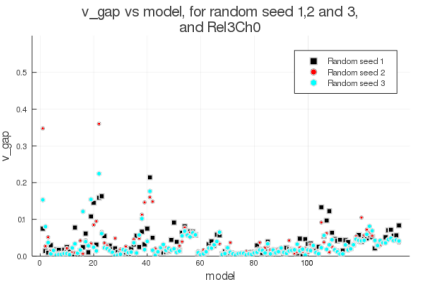
The high Vinnicombe norm occurred for the combination of two relay switches and no chirp, the result can be seen in Figure 5.6. The reason for showing these figures is that they were good examples from the batch of how the models with a large Vinnicombe metric behaved. By looking at the Bode and Nyquist diagrams of Figure 5.6 one can see that the estimated models are all close to the real process for the frequencies shown. In this case the Vinnicombe metric is relative high because the time delay estimate is slightly off which has a big impact on a fast model such as model 1.



(a) Two relay switches and no chirp.



(b) Two relay switches and chirp length two.



(c) Three relay switches and no chirp.

Figure 5.1: Scatter plots of the Vinnicombe metric for all models in the test batch using three different random seeds for the simulation of each model. The following combinations of relay and chirp were tested: two relay switches with no chirp, two relay switches with a relative chirp length of two, and three relay switches with no chirp.

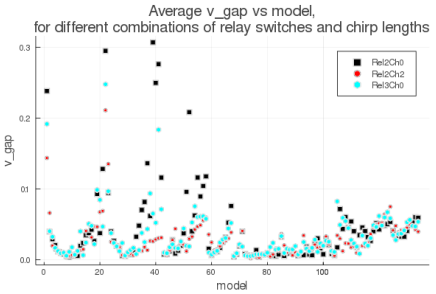
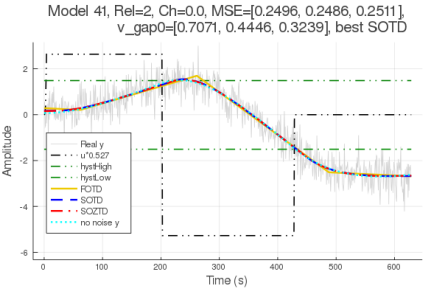
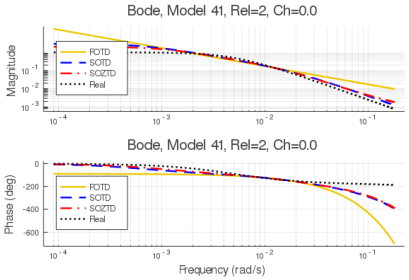


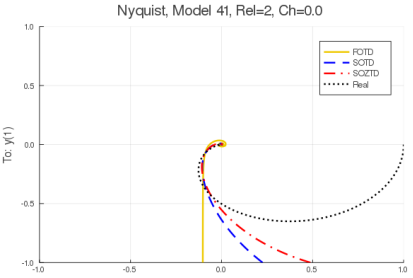
Figure 5.2: A scatter plot of the average Vinnicombe metric for all models in the test batch using three different random seeds for the simulation of each model. The following combinations of relay and chirp were tested: two relay switches with no chirp, two relay switches with chirp of length two, and three relay switches with no chirp.



(a) Experiment.

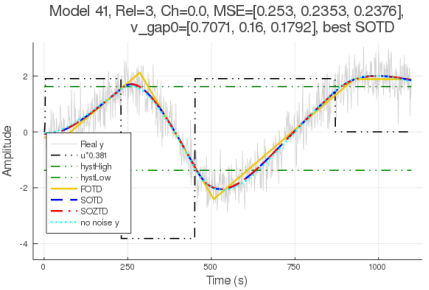


(b) Bode plot.

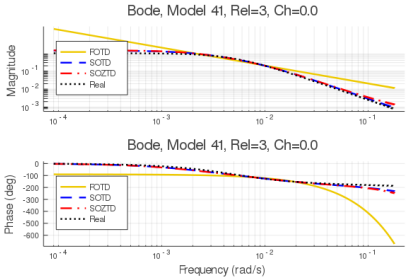


(c) Nyquist diagram.

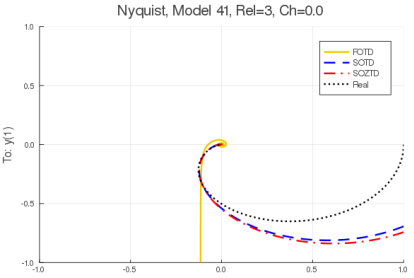
Figure 5.3: Experiment, Nyquist and Bode plots for the estimations of model 41, using two relay switches and no chirp. In all plots, "$FOTD$", "$SOTD$" and "$SOZTD$" were the estimated models. In the curve plot "*Real y*" was the real output that was used in the optimisation, "*no noise y*" was the real output without noise, "$u \cdot n$" was the input signal scaled by the factor n, "$hystHigh$" was the higher hysteresis, and "$hystLow$" was the lower hysteresis.



(a) Experiment.

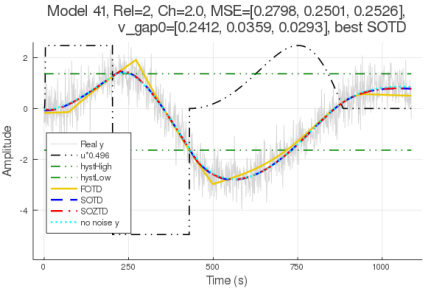


(b) Bode plot.

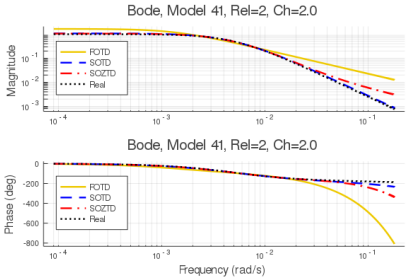


(c) Nyquist diagram.

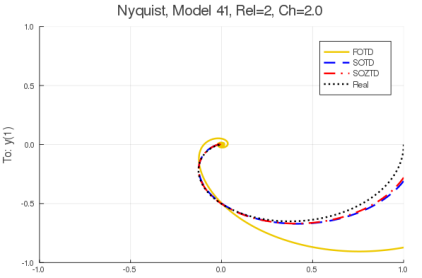
Figure 5.4: Experiment, Nyquist and Bode plots for the estimations of model 41, using three relay switches and no chirp. In all plots, "$FOTD$", "$SOTD$" and "$SOZTD$" were the estimated models. In the curve plot, "*Real y*" was the real output that was used in the optimisation, "*no noise y*" was the real output without noise, "$u \cdot n$" was the input signal scaled by the factor n, "*hystHigh*" was the higher hysteresis, and "*hystLow*" was the lower hysteresis.



(a) Experiment.

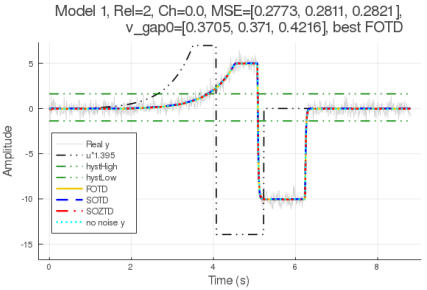


(b) Bode plot.

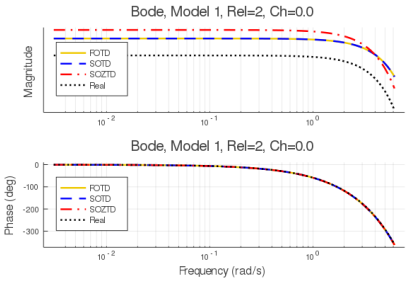


(c) Nyquist diagram.

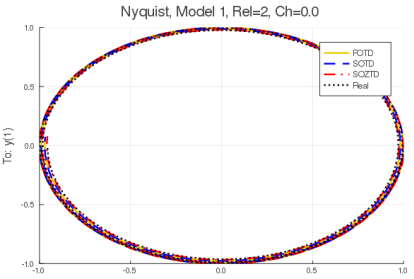
Figure 5.5: Experiment, Nyquist and Bode plots for the estimations of model 41, using two relay switches and a relative chirp length of two. In all plots, "$FOTD$", "$SOTD$" and "$SOZTD$" were the estimated models. In the curve plot, "*Real y*" was the real output that was used in the optimisation, "*no noise y*" was the real output without noise, "$u \cdot n$" was the input signal scaled by the factor n, "$hystHigh$" was the higher hysteresis, and "*hystLow*" was the lower hysteresis.



(a) Experiment.



(b) Bode plot.



(c) Nyquist diagram.

Figure 5.6: Experiment, Nyquist and Bode plots for the estimations of model 1, using two relay switches and no chirp. In all plots, "$FOTD$", "$SOTD$" and "$SOZTD$" were the estimated models. In the curve plot, "*Real y*" was the real output that was used in the optimisation, "*no noise y*" was the real output without noise, "$u \cdot n$" was the input signal scaled by the factor n, "$hystHigh$" was the higher hysteresis, and "$hystLow$" was the lower hysteresis.

5.2 Upper water tank experiment

The experiment was performed on the upper tank also using two relay switches and no chirp, the experiment data yielded the following models

$$\begin{aligned} P_{FOTD} &= \frac{0.049}{s+0.0017} e^{-0.804s}, \\ P_{SOTD} &= \frac{0.114}{s^2+2.116s+0.061} e^{-0.409s}, \\ P_{SOZTD} &= \frac{-0.002s+0.096}{s^2+1.766s+0.068} e^{-0.334s} \end{aligned} \tag{5.3}$$

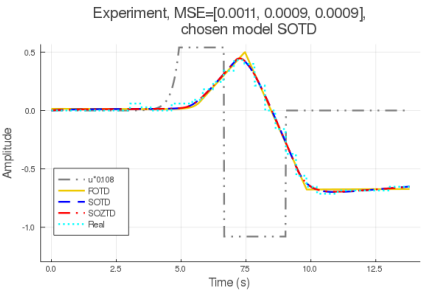
where SOTD was chosen as it had the lowest weighted MSE. The associated curve, Bode and Nyquist plots can be seen in Figure 5.7. The other test, performed using

five relay switches and a relative chirp length of four, had the results

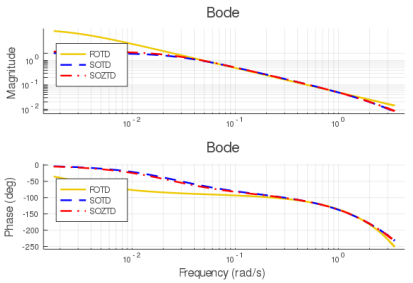
$$
\begin{aligned}
P_{FOTD} &= \frac{0.049}{s+0.016}e^{-1.682s}, \\
P_{SOTD} &= \frac{0.055}{s^2+1.109s+0.018}e^{-0.901s}, \\
P_{SOZTD} &= \frac{0.152s+0.284}{s^2+5.813s+0.090}e^{-1.987s}
\end{aligned}
\tag{5.4}
$$

where SOZTD was chosen as it had the lowest weighted MSE. The output, Bode and Nyquist plots can be seen in Figure 5.8. The biggest difference between the different combination of relay and chirp was the stationary gain. However, the most important frequency properties were close to each other.

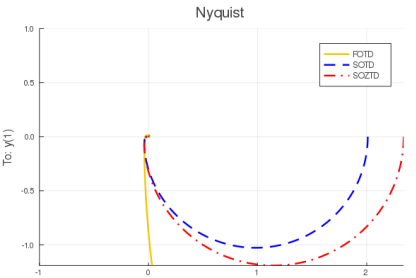
The plots of the experiments using the current autotuner on the upper tank, are shown in Figure 5.9.



(a) Experiment.

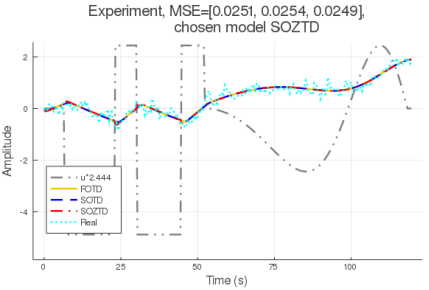


(b) Bode plot.

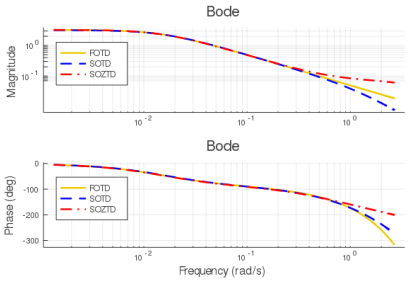


(c) Nyquist diagram.

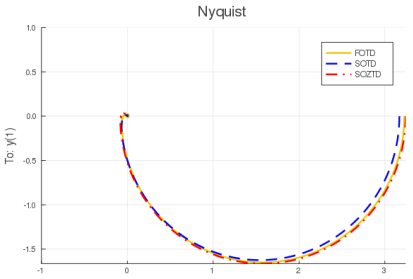
Figure 5.7: Experiment, Nyquist and Bode plots for the estimations of the upper tank process, using two relay switches and no chirp. In all plots, "$FOTD$", "$SOTD$" and "$SOZTD$" were the estimated models. In the curve plot, "*Real*" was the real output and "$u \cdot n$" was the input signal scaled by the factor n.



(a) Experiment.

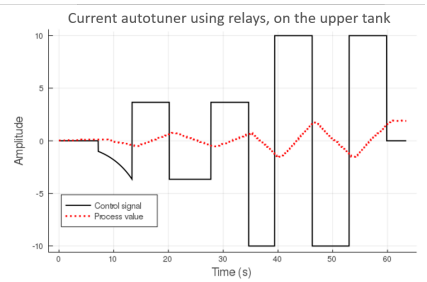


(b) Bode plot.

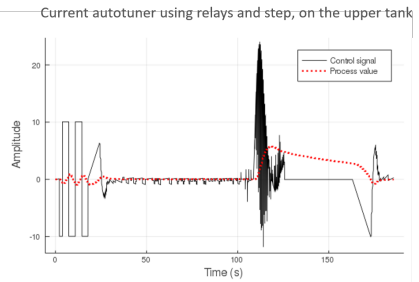


(c) Nyquist diagram.

Figure 5.8: Experiment, Nyquist and Bode plots for the estimations of the upper tank process, using five relay switches and a relative chirp length of four. In all plots, "$FOTD$", "$SOTD$" and "$SOZTD$" were the estimated models. In the curve plot, "*Real*" was the real output and "$u \cdot n$" was the input signal scaled by the factor n.



(a) Using relay only.



(b) Using relay and a reference step.

Figure 5.9: The current autotuner experiments on the upper tank using only relay in the left plot and adding a closed loop reference step in the right plot.

5.3 Lower water tank experiment

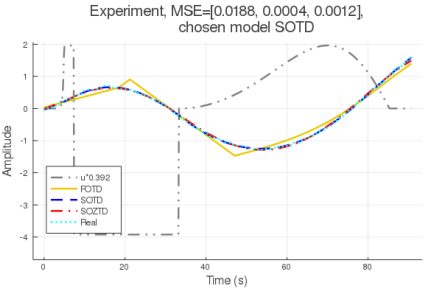
The experiment was also performed on the lower tank using two relay switches and a relative chirp length of two. The result can be seen in Figure 5.10 and the estimated models were

$$\begin{aligned} P_{FOTD} &= \frac{0.013}{s+2\cdot 10^{-19}} e^{-13.83s}, \\ P_{SOTD} &= \frac{0.00084}{s^2+0.028s+0.0012} e^{-1.01s}, \\ P_{SOZTD} &= \frac{-0.0004s+0.0005}{s^2+0.0094s+0.0033} e^{-0.33s}. \end{aligned} \tag{5.5}$$

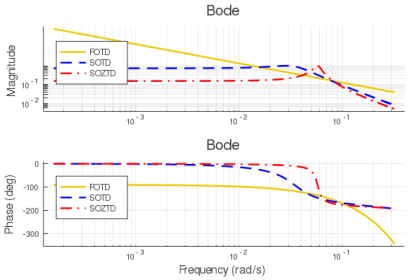
where SOTD was chosen as it had the lowest weighted MSE. To compare how good this result was, another test was performed using four relay switches and a chirp length of two which should give more accurate models. The resulting models were

$$\begin{aligned} P_{FOTD} &= \frac{0.012}{s} e^{-15.221s}, \\ P_{SOTD} &= \frac{0.0006}{s^2+0.034s+0.0003} e^{-0.929s}, \\ P_{SOZTD} &= \frac{-0.0001s+0.0006}{s^2+0.034s+0.0003} e^{-0.742s}. \end{aligned} \tag{5.6}$$

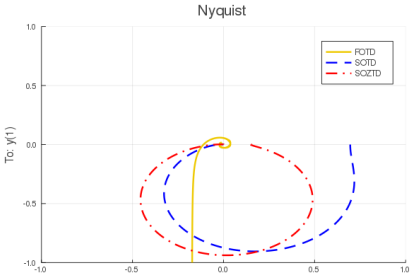
with the best model being SOTD and the resulting curve, Bode and Nyquist plots can be seen in Figure 5.11. Using this result it looked like the SOTD, from the two relay switches and a chirp length of two experiment, was a good model of the system. The current autotuner was also used on the lower tank and the result is shown in Figure 5.12. The experiment displayed a similar result to the one for the upper tank. The results in Figure 5.10 and 5.12 show that our experiment was faster and more robust since the number of relay switches and the relative chirp length were defined from the start, while the current autotuner would use four to nine relay switches and if a step was included in the experiment the process had to get stationary before starting the step which could take a long time.



(a) Experiment.



(b) Bode plot.



(c) Nyquist diagram.

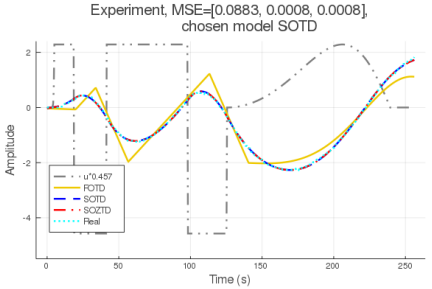
Figure 5.10: Experiment, Nyquist and Bode plots for the estimations of the lower tank process, using two relay switches and a relative chirp length of two. In all plots, "$FOTD$", "$SOTD$" and "$SOZTD$" were the estimated models. In the curve plot, "$Real$" was the real output and "$u \cdot n$" was the input signal scaled by the factor n.

5.4 Comparison between current and new method

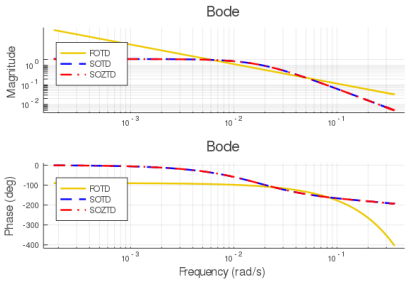
Another comparison was to test our method against the current autotuner at ABB. Comparing PI controllers suggested from the current autotuner with the AMIGO method [Åström and Hägglund, 2005] on the SOTD model from Equation 5.3, the PI settings in Table 5.1 were obtained. K and T_i are parameters of PI transfer function

$$C_{PI} = K(1 + \frac{1}{T_i s}). \tag{5.7}$$

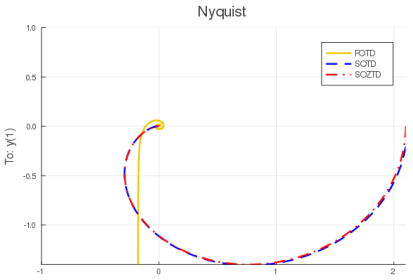
The reference step result can be seen in Figure 5.13 and the load disturbance result in 5.14.



(a) Experiment.



(b) Bode plot.

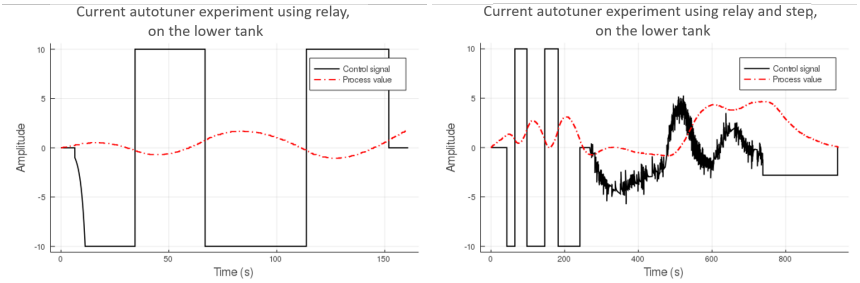


(c) Nyquist diagram.

Figure 5.11: Experiment, Nyquist and Bode plots for the estimations of the lower tank process, using four relay switches and a relative chirp length of two. In all plots, "$FOTD$", "$SOTD$" and "$SOZTD$" were the estimated models. In the curve plot, "$Real$" was the real output and "$u \cdot n$" was the input signal scaled by the factor n.

Table 5.1: The PI settings for the different autotuners.

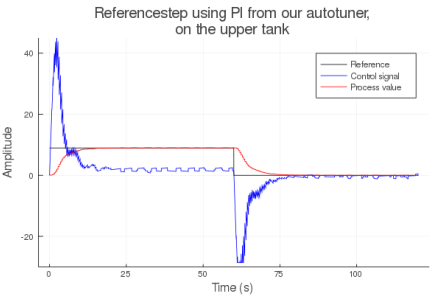
PI tuning method	K	T_i
Our model estimation with AMIGO-PI	15.42	4.8
Current autotuner without step	5.22	8.66
Current autotuner with step	1.8	8.61



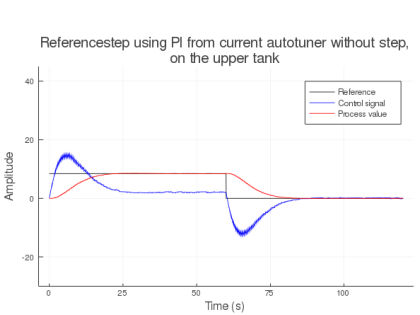
(a) Using relay only.

(b) Using relay and a reference step.

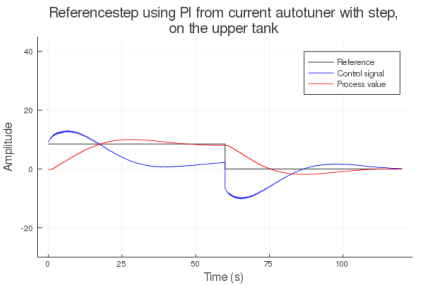
Figure 5.12: The current autotuner experiments on the upper tank using only relay in the left plot and adding a closed loop reference step in the right plot.



(a) AMIGO method with our model.

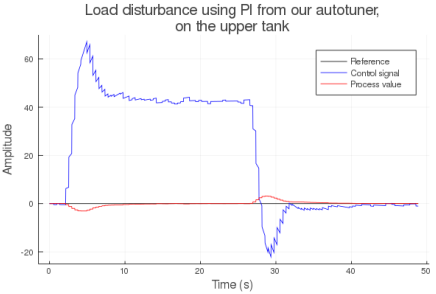


(b) Current autotuner, using only a relay.

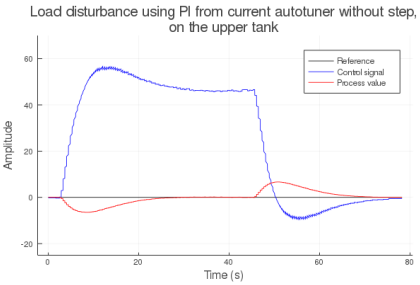


(c) Current autotuner, using a relay with a reference step.

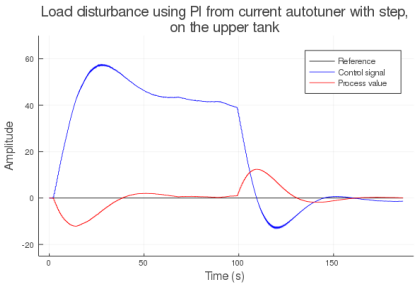
Figure 5.13: A positive and a negative reference step on the upper tank process using three different PI controllers from three different autotuners.



(a) AMIGO method with our model.



(b) Current autotuner, using only a relay.



(c) Current autotuner, using a relay with a reference step.

Figure 5.14: A positive and a negative load disturbance on the upper tank process using three different PI controllers from three different autotuners.

6

Discussion

6.1 Experiment choice

One of the goals of this project was to create a robust and short experiment to identify sufficiently good models to design simple controllers for unknown processes. As can be seen in Figure 5.1 the results of all experiment candidates provided good Vinnicombe metrics for the majority of the processes found in the test batch with only a few exceptions. Two of these exceptions were model 1 and 41 from the batch with the combination of two relay switches and no chirp. These can be seen in Figure 5.6 and 5.3. The reason for these models to have high Vinnicombe metric was that the stationary gains were a bit off. This was the common reason for high Vinnicombe. This happened especially when there was no chirp, so if the stationary gain is crucial a chirp should be used. Using a chirp, however, will approximately double the experiment time.

With this in mind, it was time to decide which combination of relay and chirp that was the best. From the perspective of finding the best model under ideal circumstances, the best combination would be using two relay switches and a chirp of relative length two as it had the best model estimations, as can be seen in Figure 5.2. But when considering the extra time the chirp takes and that for most cases all combinations tested performed roughly the same, the default choice ought to be using two relay switches only. Instead of being a default, the chirp should be an advanced feature used when accuracy is crucial and when experiment time is not.

6.2 Comparison to current autotuner

By comparing the current autotuner to our method it was apparent that our was much shorter. Even adding a chirp of length two, which would approximately double the experiment time, would result in a shorter experiment time. This was a very good result since one of the objectives was to have a short experiment time. Our method was also more robust as the results using the current autotuner might require a couple of attempts before it would succeed with the entire test without getting stuck.

For example, it could get stuck when waiting for stationarity between the relay and step, making the experiment very long as in Figure 5.9. It is hard to draw any conclusion from the results from the reference step and load disturbance, because the PI-tuning was not a part of this thesis. Through a more suitable tuning method the results based on our models could probably be improved.

6.3 Problematic scenarios

Throughout our experimentation and development of this method, we have encountered cases where the algorithm has failed to generate a sufficient model for our purposes. In this section, we discuss why and when this has happened.

Non-stationary start

In the current implementation of our new method, no check ensures the system to be stationary at the start of the experiment. The operator would need to ensure that before the experiment is started, which for some systems can be difficult. The problem from not being close enough to stationary is that the hysteresis will be off, compared to the stationary setpoint, and the experiment setpoint may not correspond to the control signals used to generate the deviations in the measurement. Given an extreme case the control signals might not be large enough to move the measurements past the hysteresis and the experiment would need a restart. However, the current autotuner has the same limitation, hence this is not a new problem that has been introduced through the new method. This is rather a problem that has not been fully solved yet.

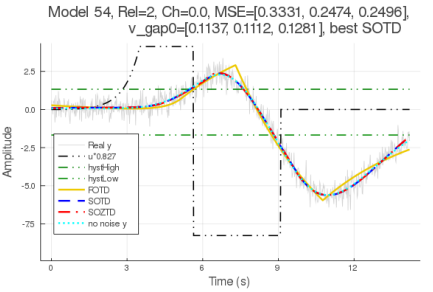
High-order systems

A natural limitation of our method is that it was not always possible to fit a low-order model to match the dynamics of a high-order system. This problem has shown up on models such as models 54-58 where the transfer functions are

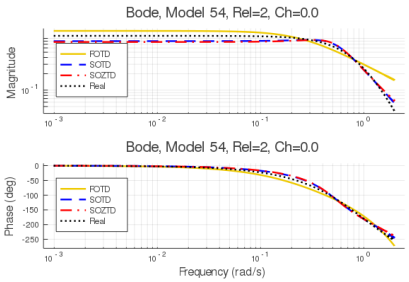
$$P_4(s) = \frac{1}{(s+1)^n} \tag{6.1}$$

with $n = 4,5,6,7,8$. For these models, the results have been slightly worse in comparison to the rest. What can be seen in these cases is that the estimated models tend to be well estimated around the critical frequency but not as good around the low frequencies which yielded the higher Vinnicombe metric. An example of this can be seen in Figure 6.1 where model 54 in the batch was estimated. Model 54 is a fourth order system. When designing a PID for such fit, the impact of the stationary gain estimation error can range from severe to minor, depending on the method that is used. For instance, if a PI-tuning method which depend on the static gain is used, the models found for the high-order systems may become bad. The AMIGO method is based on the static gain. Therefore, another tuning method should be used

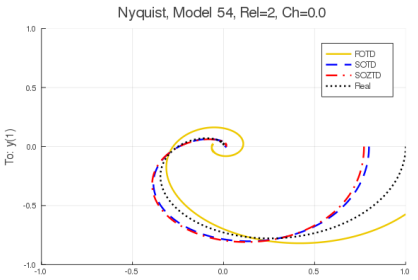
in combination with our model identification. Ideally, a method that focuses mostly around the critical frequency should be used as this is where the model identification is best. In this way the impact of the low-frequency response will not impact the control design as much.



(a) Experiment.



(b) Bode plot.



(c) Nyquist diagram.

Figure 6.1: Curve, Nyquist and Bode plots for the estimations of model 54, using two relay switches and no chirp. In all plots, "$FOTD$", "$SOTD$" and "$SOZTD$" were the estimated models. In the curve plot, "*Real y*" was the real output that was used in the optimisation, "*no noise y*" the real output without noise, "$u \cdot n$" the input signal scaled by the factor n, "*hystHigh*" the higher hysteresis and "*hystLow*" the lower hysteresis.

Lag-dominated processes

As can be seen in Figure 5.1a models such as 21 and 38-42, which have a lot of process lag compared to time delay, have a higher Vinnicombe metric. The reason for this was that they are more sensitive to noise than other processes. An example can be seen in Figure 5.3. One way we solved this problem was by oversampling the processes. From the oversampled data we formed local averages instead of exact measurements. This solved the noise problem for most of the cases. On the other hand, this method requires oversampling which might not always be possible. An-

other way to solve the noise problem is to increase the control signal amplitude. This could, however, give problems with the process itself or with nonlinearities.

Delay-dominated processes

From our experience, it was difficult to get a low Vinnicombe metric for this type of processes. We think the primary problem with a delay-dominant processes comes from the proportionally fast dynamics of the system in comparison to the experiment time, for example, model 1 from the batch in Figure 5.6. In this experiment, it is quite clear that for most of the time nothing happens. To capture the dynamics one would need very high sampling around the edges where much is happening, otherwise, the underlying dynamics become unobservable. To capture the dynamics there is a need for adaptive sampling or a general high sampling frequency.

High noise presence

Throughout this project, multiple ways of handling processes with very noisy signal have been tried. One problem which might occur is a low signal to noise ratio which makes the estimation go wrong often. To ensure a certain signal to noise ratio we chose the hysteresis level based on a noise estimation. By having the hysteresis based on the noise a problem arose. Will the control signal be large enough to move the signal past the hysteresis? As we are in general unaware of the static gain it is not possible to know this before the experiment has started. Thus, either one will have to lower the hysteresis level or increase the allowed control signal. We have chosen to do the latter to ensure a certain signal to noise ratio but we have also tried the first method which worked well in most cases. Although, when the signal to noise ratio became too low, it was no longer possible to identify meaningful models. This could in some cases be solved by taking local averages to lower the noise but overall it was more robust to force a high signal to noise ratio to begin with.

Unstable processes

In its current state, our method is not suitable for unstable processes as it is not certain that our experiment would stabilise such a system. This motivates why our method assumes stable systems in the optimisation. This is not a new limitation introduced by the proposed method and is also a limitation of the current autotuner. To model unstable processes more advanced methods should be used instead.

Unknown stationary gain

For a general system, the static gain is unknown and hence not possible to use in our method, this can impact the experiment. For example, if the static gain is very small and there is noise with high amplitude, the hysteresis levels might not be reached, as they are based on the noise. This problem can be fixed by increasing the allowed control signals to be used in the experiment. If this is not possible the user would need to change the settings regarding the hysteresis levels.

Processes with integrator

When working with integrating systems, the two control signals used in the relay must be on opposite sides of the integrating point. If they are not, the experiment will not be able to push the signal past one of the hysteresis levels and the experiment will not finish. Instead, the process will drift until it reaches its maximum or minimum value. To prevent such a scenario a limit for permitted process values should be used.

Highly nonlinear systems

In the case of nonlinear systems, the experiment could be a problem. Since the experiment executes around a starting point that was of interest for the user, there is a risk that the output leaves the local linear region of interest. If this is the case, the estimated model could be compromised from the nonlinearity. The operator can adjust the amplitudes of the input signal to the process to counter this but it might be hard for the user to know whether or not this was a problem.

6.4 Use in other contexts

In this master thesis, the objective was to design an experiment which yields sufficient models to be used for tuning PID controllers. However, the method could be applied to several other subjects where a good model is necessary. For example, it could be used for Kalman filters and Model predictive control.

6.5 Also tested

Throughout the testing and development of this project, several ideas have been tested. Below are the most important ideas addressed that due to various reasons did not make it to the final algorithm.

Unstable poles

In the beginning of the project, we allowed the optimiser to estimate models with unstable poles. However, sometimes when the real system had stable poles the estimation gave unstable poles instead. It was later removed to only allow stable poles. The reason for this was not only the aforementioned problem but also that the experiment might not be possible to perform on an unstable process.

Allowed arbitrary static gain

Negative static gain was another thing we allowed the optimiser to do initially in the project. Though, this gave the problem where the SOZTD could estimate a negative static gain in a case where the gain was positive. The models were good otherwise, except for the static gain. The ability to estimate a negative static gain was then

removed not only because of the optimisation getting it wrong but also because there could be an experiment problem. If any static gain was allowed then the experiment must be able to detect this by allowing the first step in the relay to hit both the high and low hysteresis. This was not a problem implementation-wise However, if the process to be estimated had a zero in the right half plane there was a risk of the experiment hitting the wrong hysteresis boundary. Then the system would think that the process has a static gain that is the opposite to the truth. For this reason, our method was restricted to not use arbitrary static gain when doing the experiments. Instead, the operator is expected to tell the system which sign the static gain has before the start.

Fixed hysteresis with adaptive filter

As mentioned in the algorithm chapter, in the final method the hysteresis chosen was three times greater than the standard deviation of the estimated noise and a filter was created with a tenth standard deviation compared to the noise. This was not always the case. Before, the hysteresis was set to be five times bigger than the standard deviation of the filter, hence, half of the estimated noise. The reason for this was to have a lower hysteresis which would make the experiment time shorter. The reason for not using this anymore and having a much larger hysteresis was that the signal to noise ratio was poor and the experiment did not excite enough to yield a sufficient model.

The chirp frequencies

The chirp frequencies were chosen to be between zero and half of the frequency corresponding to the longest time between two subsequent relay switches, as mentioned in the algorithm chapter. This was done to let the chirp excite lower frequencies in comparison to the relay. During this master thesis, a couple of different alternatives have been tested. The first was to have the frequencies around the frequencies from the relay, to enhance the excitation there. This did, however, not improve much since the relay had already excited these frequencies enough. Another alternative tested, was to set the frequencies between zero and the frequency corresponding to the longest time between two subsequent relay switches to excite all frequencies below those excited by the relay. When doing this the experiment plots showed that the chirp oscillated faster than the relay. We think that the reason for this was that the chirp was made with this formula

$$u_{chirp}(t) = \sin(t\omega(t)), \tag{6.2}$$

where ω was the angular velocity depending on time t. Because the sin function has two time depending functions being multiplied it takes bigger steps than it should. This is the same as having a higher angular velocity. This was solved by taking half of the frequency of the longest time between two consecutive relay switches as the highest frequency which reduced this effect.

6.6 Future work

This section discusses subjects that are not in the section problematic scenarios that could be done to improve or extend our proposed method.

PID controller

The focus of this master thesis was to obtain a good model of an unknown process so it could be used to tune a PID controller for that process. Our suggestion is to use a PID tuning rule that is not sensitive to errors in the static gain. The reason for this is that our estimated models are good in the frequency range around the -180° phase lag, which makes the static gain of the model uncertain. One alternative for PID tuning would be optimisation on the integrated absolute error of a reference step and/or a load disturbance response with demands on the sensitivity and complementary sensitivity function.

Communication

With our current implementation in AC 800M the data is saved in a text document that must be manually linked into the Julia optimisation code. This is something that should be automatic in the future. It could, for example, be done using some cloud based communication or, if it is possible, the Julia optimisation code could be implemented on an ABB computer unit sold together with the control unit.

Averaging when downsampling

One thing that we have performed tests on, and that would be interesting to extend is averaging when downsampling. This means that instead of taking some samples at specific time points more data would be used to take an average around each point. The benefit would be less noise which gives a smoother curve for the optimiser to adjust to. The disadvantage with this is that processes which are delay-dominant could get too suppressed around the edges and this would alter the real curve. If the real curve is altered there is a higher risk of the model estimation being wrong.

Adaptive downsampling

To solve the problem with averaging when downsampling, some form of adaptive downsampling that adjusts to the gradient of the curve could be used. This would also help without averaging because the downsampling always has a risk of missing some important points, especially points where there is much change in the curve.

User interface

To make the autotuner easy to use for the operator a nice user interface would be valuable. Making it easy to start the experiment and to change operator parameters like the static gain sign, limits of the control signal and if the chirp is wanted or not.

7

Conclusion

Through a simulation study and experiments on a real process, we have shown what a system identification method based on the work in [Berner, 2017] can accomplish and that there is still more work to be done. Through a short experiment, it is now possible to fit low-order models with time delays which match the frequency properties of the real processes with high accuracy in most cases. Furthermore, we have developed a novel method for initialising a model by combining the smooth properties of cubic splines and the fast algorithm LS to find good initial guesses for relay and chirp experiments. By implementing the algorithm onto ABB's platform it has been shown that this method can be used at large companies, within the field of automatic control. Furthermore, the algorithm has been compared to the currently used autotuner at ABB and our method was shown to generate good results with a shorter experiment time.

Bibliography

Åström, K. J. and T. Hägglund (1984). *Automatic Tuning of Simple Regulators with Specifications on Phase and Amplitude Margins*. Automatica, 20(5), 645-651. https://portal.research.lu.se/portal/files/6340936/8509157.pdf.

Åström, K. J. and T. Hägglund (2005). *Advanced PID Control*. ISA.

Berner, J. (2017). *Automatic Controller Tuning using Relay-based Model Identification*. Lund: Department of Automatic Control, Lund Institute of Technology, Lund University.

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

Kraft, D. (1988). *A software package for sequential quadratic programming*. Technical Report DFVLR-FB 88-28, Institut für Dynamik der Flugsysteme, Oberpfaffenhofen.

Kraft, D. (1994). *Algorithm 733: TOMP–Fortran modules for optimal control calculations*. ACM Transactions on Mathematical Software, vol. 20, no. 3, pp. 262-281.

Michiel, H. (2001). *Spline interpolation*. Encyclopedia of Mathematics. URL: `https://www.encyclopediaofmath.org/index.php?title=p/s086820`.

Steven G. Johnson (2020). "Nlopt algorithms". URL: `https://nlopt.readthedocs.io/en/latest/NLopt_Algorithms`.

Svanberg, K. (2002). *A class of globally convergent optimization methods based on conservative convex separable approximations*. SIAM J. Optim. 12 (2), p. 555-573.

Vinnicombe, G. (2001). *Uncertainty and Feedback, H Loop-Shaping and the V-Gap Metric*. Imperial College Press, London.

Welford, B. P. (1962). *Note on a Method for Calculating Corrected Sums of Squares and Products*. Technometrics, 4:3, 419-420, DOI: 10.1080/00401706.1962.10490022.

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

Document name
MASTER'S THESIS

Date of issue
June 2020

Document Number
TFRT-6107

Author(s)
Jonas Hansson
Magnus Svensson

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

Title and subtitle
Next Generation Relay Autotuners – Analysis and Implementation at ABB

Abstract

For a long time, a method which tunes a robust controller automatically has been researched. In this thesis, a method to estimate processes using a short experiment and low order time delayed models is described. The goal was to achieve a model that could be used for PI- and PID-tuning in an autotuner. Through a simulation study, various experiment designs were tested to find an optimal experiment for model identification. The method was implemented in the ABB AC 800M controller. Using a simulation study and ABB's AC 800M controller the proposed method was shown to be successful in estimating models for a variety of processes. Furthermore, the proposed method was shown to be more robust and have a shorter experiment time than the autotuner that is currently used.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-55

Recipient's notes

Security classification