ISSN 0280-5316
ISRN LUTFD2/TFRT--5728--SE

Autotuning of a PID-controller

Camilla Andersson
Mirjam Lindberg

Department of Automatic Control
Lund Institute of Technology
October 2004

Department of Automatic Control **Lund Institute of Technology** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER THESIS *Date of issue* October 2004 *Document Number* ISRNLUTFD2/TFRT--5728--SE
Author(s) Camilla Andersson and Mirjam Lindberg	*Supervisor* Mattias Grundelius TAC, Malmö Tore Hägglund LTH, Lund *Sponsoring organization*

Title and subtitle
Autotuning of a PID-controller. (Automatisk inställning av PID-regulatorer)

Abstract
This master´s thesis has been performed in cooperation with TAC in Malmö. The TAC group makes commercial buildings smarter by integrating and automating the technical systems required to run them. TAC:s control systems use PID-controllers to control processes such as heating and ventilation. The PID-controllers are often badly tuned, since it is too timeconsuming to calculate good PID-parameters at the time of deployment. A simple way of finding PID-parameters that give faster control loops is needed. To solve this problem the thesis proposes an autotuner based on the areamethod Method of Moments and the AMIGO tuning rules. The implementation of the autotuner using IEC 61131 is described. The resulting autotuner is tested on simulated processes and gives satisfactory results. The thesis also includes practical insights on the use of the autotuner.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 38	*Recipient's notes*
Security classification		

Contents

1 Introduction **3**
1.1 About TAC 3
1.2 Background 3
1.2.1 The Concept of Autotuning 3
1.2.2 Autotuning at TAC 4

2 Methods **6**
2.1 A Step Response Method 6
2.1.1 Process Identification: the Three-Parameter Model . . 6
2.1.2 Design Method 9
2.1.3 The Method of Moments in Practice 9
2.2 The Relay Feedback Method 11
2.2.1 Process Identification 11
2.2.2 Design Method 12
2.2.3 The Relay Feedback Method in Practice 13
2.3 Discussion 13

3 Implementation **15**
3.1 IEC 61131 and PLC programming in CoDeSys 15
3.2 Overview of the Implementation 16
3.3 Detailed Implementation 17
3.3.1 Initial Phase 17
3.3.2 Closed Loop Phase 18
3.3.3 Open Loop Phase 20
3.3.4 PID-design Phase 20
3.3.5 Disturbance Detection 20
3.3.6 Filtering 23

4 Test Results **24**
4.1 Lag Dominated Process 24
4.2 Delay Dominated Process 26
4.3 Balanced Process 27
4.4 Process with Noise and Dead Zone 28

4.5 Testing on an Air Handling Unit 30

5 Conclusions 31

5.1 Summary 31

5.2 User Notes 31

5.3 Future Work 32

A Figures 34

A.1 FBD — Overview of the Communication Parameters 34

A.2 SFC — Overview of the Autotuner Algorithm 35

Chapter 1

Introduction

1.1 About TAC

TAC designs and manufactures systems for Building IT. This covers functions such as access control, heating and cooling, ventilation and lighting. With TACs integrated systems, functions such as these can be controlled and monitored from a computer terminal.

Heating, cooling and ventilation functions include controlling entities such as temperature and airflow. This means using feedback control loops.

A typical TAC product is the *Xenta*® 300. It is a programmable controller with functionality including control loops, alarm handling and time control. The minimum sample time is 1 second. When control loops are implemented, different PID-controllers may be used. The autotuner is tested with the PIDP-block, which uses a discrete time positional PID algorithm. The PIDP-block has dead zone capability which means that when the control error is smaller than the dead zone the controller output remains unchanged. The size of the dead zone is related to the accuracy of the measuring instrument.

1.2 Background

1.2.1 The Concept of Autotuning

An autotuner is a device that automatically computes the parameters of a controller. The goal is to achieve the best control possible given the tuning objectives. The goal is not to replace a human control engineer. The autotuner should rather be seen as an aid to improvement. Many PID-controllers in the field use the default parameter settings, since it is too time-consuming to tune them manually and the systems seem to work anyway. However, badly tuned controllers cannot only lead to less efficient control loops, they can also lead to unpredictable system behavior when control loops are hierarchically dependent. An autotuner can be of great help in spotting these kinds of problems as well as remedying them [1].

It is important to distinguish between autotuning and adaptive control. An autotuning operation is run on the user's initiative, and takes place during a limited time. When autotuning has been performed the control loop resumes normal operation with the new parameters. An adaptive controller continually changes its parameters during operation.

An autotuner follows the same approach that a control engineer would when tuning a controller:

1. Process Model — Choose a model for the process
2. Process Identification — Collect process data and fit it to the model
3. Design Method — Compute the new parameters

The tuning objectives of the autotuner are process specific and must be determined before implementation. Some objectives could be low overshoot, fast setpoint tracking and good disturbance rejection. Sometimes trade-offs must be made. Another important feature of an autotuner is usability. If the autotuner is difficult to use, it will probably not be used. The autotuner should not require a lot of user input to achieve good tuning. In the ideal case a start button should be enough.

1.2.2 Autotuning at TAC

Most of TACs control systems use PID-controllers. It is important that the PID-parameters are correctly tuned to achieve the high energy conservation made possible by TACs integrated systems. Today not much time is spent finetuning the controllers when a system is deployed. Usually "safe" parameters will be set which ensure that the system is stable. These parameters mostly result in slow and inefficient control. Calculating efficient parameters manually is too timeconsuming. An autotuner will make the deployment process run smoother and give TACs customers better results.

Autotuning has been used at TAC before. It was implemented in the early 90's and based on a patented method developed at the Dept. of Automatic Control, Lund Institute of Technology. Together with the autotuner a controller was implemented with an algorithm different from the standard PID-algorithm. This autotuner does not only give the controller parameters but also a recommended sample interval based on the process dynamic.

The new autotuner differs from the old autotuner since it is detached and communicates with the controller through a network. The basic idea is that the new autotuner will be able to communicate with any of the today existing PID-controllers at TAC. It is also different in the way it identifies the unknown process.

Chapter 2

Methods

In this chapter two different process identification methods with corresponding design methods are described. Based on MATLAB testing, one of the methods is chosen for the autotuner.

2.1 A Step Response Method

2.1.1 Process Identification: the Three-Parameter Model

Using a step response, a process can be approximated by a three-parameter model:

$$G_p = \frac{K_p}{1 + Ts} e^{-Ls} \tag{2.1}$$

where K_p is the static gain, T the time constant and L the time delay. The response time $T_{63} = L + T$ for stable systems can be measured as the time when the step response has reached 63% of its steady state value, see Figure 2.1.

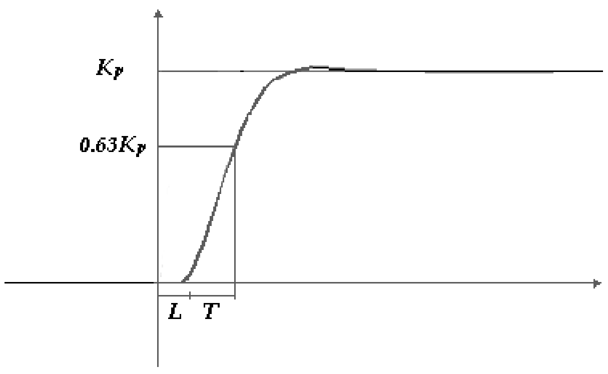

Figure 2.1: The response time $T_{63} = T + L$.

There are several different ways to determine K_p, T and L; three of them are described below.

Estimating the delay

One way is to estimate the delay L by measuring when the measurement signal is starting to increase. From the response time T_{63} and the estimated delay L, the time constant T can be calculated as $T = T_{63} - L$. The static gain K_p is calculated as:

$$K_p = \frac{y_{final} - y_{begin}}{u_{final} - u_{begin}} \tag{2.2}$$

where u is the control signal and y is the measurement signal

Ziegler-Nichols step response method

Another method is the *Ziegler-Nichols step response method* where the delay L can be estimated by using the maximum slope tangent of the step response. Using this tangent it is possible to find two points, one on the horizontal axis and one on the vertical axis. The delay is the time from the start of the step to the point where the tangent is crossing the horizontal axis. A new parameter, a, is introduced as the difference between the initial value and the tangent crossing the vertical axis. From these two parameters together with the static gain the relationship

$$a = \frac{K_p L}{T} \tag{2.3}$$

can be used to find T. K_p is calculated as in Equation 2.2.

Method of Moments

A third way to determine K_p , T and L is the *Method of Moments* described in [2]. The area A_0 is the integrated distance between the stationary end level of y and the measurement signal y in an open loop step. The area A_0/k, also called T_{ar}, can be used as a good estimate of T_{63} for the three-parameter model, see article [3]. The area A_1 beneath the step response is also integrated. The areas are shown in Figure 2.2.

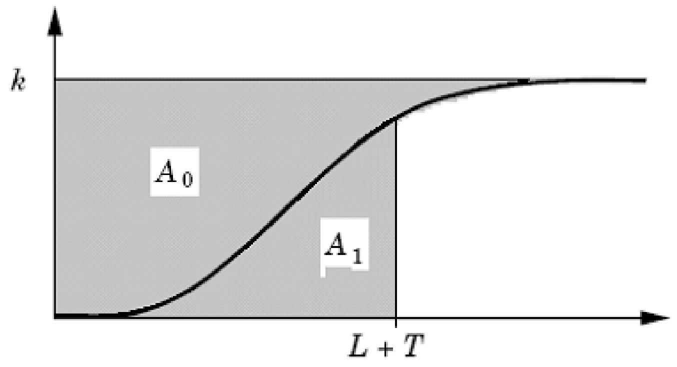

Figure 2.2: A step response in open loop with the areas used in Method of Moments.

The area A_1 is used to calculate the time constant:

$$T = \frac{eA_1}{hK_p} \tag{2.4}$$

and the area A_0 is used to calculate the delay:

$$L = T_{ar} - T = \frac{A_0}{hK_p} - \frac{eA_1}{hK_p} \tag{2.5}$$

h is the step amplitude of the control signal and K_p is the static gain from Equation 2.2.

The approximated processes yielded by these three different methods were simulated in MATLAB and were compared with the original process. The conclusion was that the Method of Moments gives the best approximation.

2.1.2 Design Method

When the parameters K_p, T and L are known a design method can be chosen to transform them into PI-parameters or PID-parameters. One such method is the well-known Ziegler-Nichols' set of tuning rules. A modern method is the AMIGO tuning rules proposed in [3]. It uses the following equations:

$$\begin{aligned} K &= \frac{1}{K_p}(0.2 + 0.45T/L) \\ T_i &= \frac{0.4L + 0.8T}{L + 0.1T}L \\ T_d &= \frac{0.5LT}{0.3L + T} \end{aligned} \tag{2.6}$$

Instead of only using two parameters, a and L like the Ziegler-Nichols tuning rules, the AMIGO tuning rules are based on three parameters, K_p, T and L. The AMIGO method has been developed with a robustness constraint and it uses a dependency on the normalized dead time $\tau = L/(L+T)$ unlike the Ziegler-Nichols tuning rules. The AMIGO method suits a large variation of processes and seems to be more universal than the Ziegler-Nichols tuning rules.

The AMIGO design method recommends a setpoint weighting which depends on τ, this is not used since the setpoint weighting is not changeable in the PIDP-block. The PIDP-block has its own setpoint weighting definition: $b = 0$ for PI and PID-control and $b = 1$ for P and PD-control.

2.1.3 The Method of Moments in Practice

A variation of the Method of Moments described in [2], which is further developed in [4] is used.

First a step change is applied in open or closed loop and T_{ar} is determined by integrating the area between the normalized control and measurement signals, see Figure 2.3.

$$A_0 = T_{ar} = \int_{t_b}^{t_f} u(t)dt - \int_{t_b}^{t_f} y(t)dt = L + T \tag{2.7}$$

where t_b is the time at the beginning of the step change and t_f is the time at the end of the step change. The stationary gain is used as an estimate of the process gain K_p.

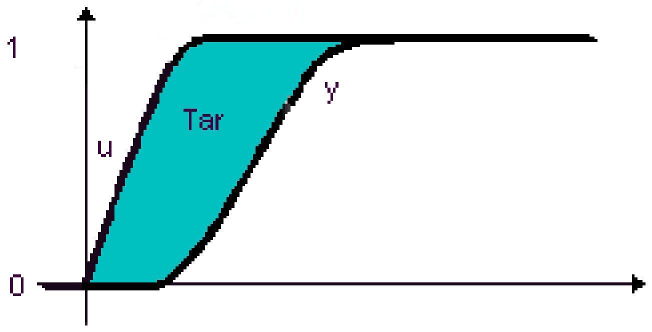

Figure 2.3: A step response in closed loop with the normalized control and measurement signal.

A second step change in opposite direction is then applied in open loop and the area A_1 of the step response is calculated during the time T_{ar}. The time constant T is then calculated as in Equation 2.4 and the delay L is equal to $T_{ar} - T$.

The Method of Moments can be executed in both open loop and closed loop. A closed loop is known to take care of process disturbances but requires some safe default values for the PID-parameters of the controller. When using closed loop, one must be aware that a high gain can cause instability problems when the process has long delays. With an open loop, it is difficult to be sure that the step amplitude is appropriate. An unknown high gain can lead to unforeseen consequences in the process output.

This area-method is easy to automate since calculation of the areas through integration can be done online. It is also robust to high-frequency disturbances since erroneous measurements during the integration process should cancel each other out due to the low pass nature of integration.

A lot of tests have been performed on different types of processes. The major test objective was to see if the new controller parameters improved the performance of the control loop. The range of tested processes includes first-order and higher-order processes as well as processes with small and large time constants and delays.

The tests were performed in SIMULINK. The Method of Moments was automated using an S-function. The AMIGO design method mentioned above in Equation 2.6 was used with the approximated parameters K_p, T and L. A step response of each process controlled with safe default PI-parameters and a step response of the same process controlled with the new PID-parameters were compared. The new controller parameters give a fast step response with an acceptably small overshoot.

2.2 The Relay Feedback Method

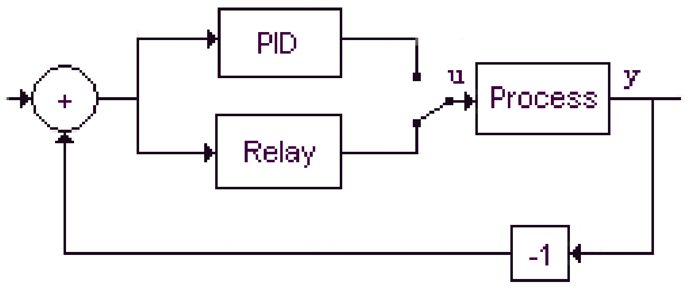

Figure 2.4: Relay feedback loop.

2.2.1 Process Identification

The relay feedback method focuses on identifying one point on the Nyquist curve of the process. The relay feedback causes the process to enter a limit cycle, i.e. a stable oscillation, see Figure 2.5. The frequency of the limit cycle approximates the ultimate frequency, ω_u, which is where the process has a phase lag of 180°. The process gain at the frequency ω_u is called the ultimate gain, K_u, and can be approximated by

$$K_u = |G(i\omega_u)| = \frac{\pi a}{4d} \tag{2.8}$$

where d is the relay amplitude and a is the limit cycle amplitude.

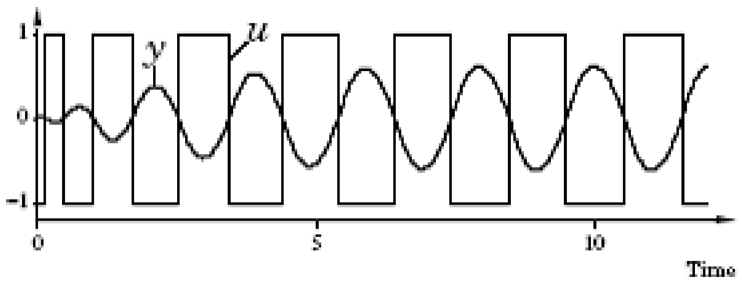

Figure 2.5: Relay feedback limit cycle.

An explanation for Equation 2.8 lies in describing function theory. A relay is a static nonlinearity and can be described by the gain $N(a)$ where a is the input amplitude. $N(a)$ is called the describing function, see [2]. The condition for oscillation is given by $N(a)G(i\omega) = -1$ which means that $G(i\omega) = -1/N(a)$, see Figure 2.6.

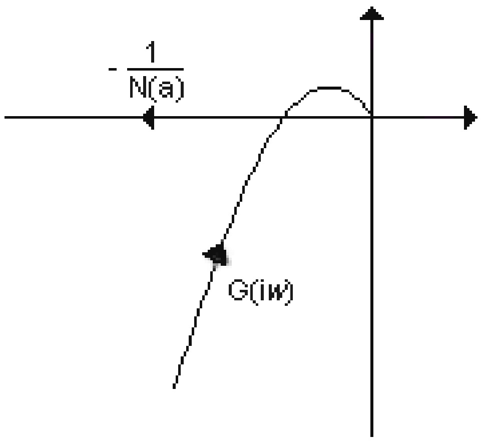

Figure 2.6: $N(a)$ is the describing function of the relay.

The describing function of the relay nonlinearity is $N(a) = \frac{4d}{\pi a}$.

To avoid random switching during noisy conditions a relay with hysteresis can be used. The signal must then be larger than the hysteresis width ϵ to cause a switch. The negative inverse of the describing function for this kind of relay is somewhat more complicated:

$$-\frac{1}{N(a)} = -\frac{\pi}{4d}\sqrt{a^2 - \epsilon^2} - i\frac{\pi\epsilon}{4d} \tag{2.9}$$

2.2.2 Design Method

In the article [6] this design method is recommended for most kinds of processes. The controlled system is specified as:

$$G(i\omega)G_{PID}(i\omega) = 0.5e^{-i\frac{135\pi}{180}} \tag{2.10}$$

The PID-parameters must be chosen so that the original point on the Nyquist curve is moved to this new point. The controller has to advance the process phase by $45\,^\circ$ since the original phase is $-180\,^\circ$. The PID-parameters can be calculated as:

$$
\begin{aligned}
K &= \frac{0.5\cos 45\,^\circ}{K_u} \\
T_d &= \frac{\tan 45\,^\circ + \sqrt{\frac{4}{\alpha} + \tan^2 45\,^\circ}}{2\omega_u} \\
T_i &= \alpha T_d
\end{aligned}
\tag{2.11}
$$

where $\alpha = 6.25$ according to [6].

2.2.3 The Relay Feedback Method in Practice

The relay method as described above is easy to automate. The output from the process is measured and when a stable limit cycle is detected its period time and amplitude are measured and averaged over several cycles. The period time gives the ultimate frequency ω_u and the amplitude a is used to calculate K_u as in Equation 2.8.

The method has been tested on a range of processes similar to the ones tested in part 2.1.3. The design method described in Equation 2.11 was used. Tests have been performed using SIMULINK and the process identification and design are both implemented in an S-function. A step response of each process controlled with safe default PI-parameters and a step response of the same process controlled with the new PID-parameters were compared. The immediate conclusion is that the design method does not give very good results for most of the tested processes. For processes including a substantial delay the control is very slow and inefficient and processes with large time constants receive a very large K.

The process delay could be approximated during the experiment and an appropriate design method chosen according to the result. In article [6] a PI-design is recommended for processes with delays.

2.3 Discussion

The step response method, represented by the Method of Moments, and the relay feedback method are both easy to automate. Positive aspects of the Method of Moments are that it is robust against high frequency disturbances and safe if the first step change is made in a closed loop. If the first step change is made in open loop it means taking a risk with the system. The relay feedback method is executed in a closed loop and is also safe for the system.

The decision of which method to choose is based on the test results with the new PID-parameters. The parameters generated with Method of Moments give fast step responses with a small overshoot. The parameters generated with the relay feedback method do not give so good step responses. Depending on the process characteristics the step responses are either very slow or have large overshoots. To get better results, a step change could be added to the relay feedback method to find the static gain of the unknown process. Then the PID-design could be based on three parameters instead of only two, K_u and ω_u. The process delay L could also be approximated. Based on L a decision could be made in the design phase on whether to use PID or PI-control. Since the Method of Moments together with the AMIGO design already gives good results this will be the method used in the autotuner.

Chapter 3

Implementation

3.1 IEC 61131 and PLC programming in CoDeSys

The autotuner is implemented in the IEC 61131 standard. IEC 61131 has been developed for PLC application programming in the automation industry. It contains several languages as follows:

- Instruction List (IL)
- Structured Text (ST)
- Sequential Function Charts (SFC)
- Function Block Diagram (FBD)
- Ladder Diagram (LD)

The first two languages, IL and ST, are text based. IL is an assembly-like language while ST resembles Pascal. The other languages are graphical languages, which means that applications are described using different blocks connected to each other. An IEC 61131 application can be created using any combination of these languages. The concept of hierarchy is well developed which means that an application can be highly structured. Blocks can contain applications written in any of the languages mentioned above.

The most specialised language in IEC 61131 is SFC, which is used to describe sequential behaviour. This is very useful for control applications since they are often time- and/or event-driven. A SFC application is built using steps and transitions. The steps represent the states in the control flow. The transitions are conditions that allow state change. A transition can only occur if the step immediately before the transition is active. Concurrent as well as alternative sequences of steps are allowed.

The IEC 61131 programming tool used for implementation of the autotuner is called CoDeSys, Controller Development System, by Smart Software Solutions.

3.2 Overview of the Implementation

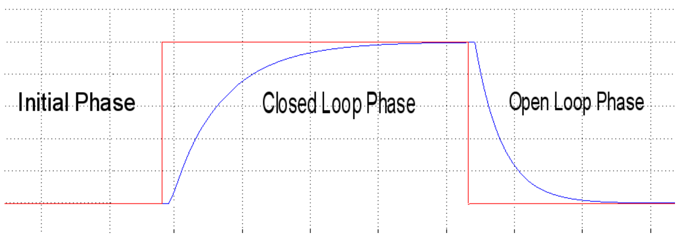

Figure 3.1: The phases in the autotuner experiment.

The autotuner algorithm is an implementation of the Method of Moments described in section 2.1.3. The implementation is divided into four phases, Initial phase, Closed Loop phase, Open Loop phase and PID-design phase:

1. Initial Phase — check noise levels, ensure stationarity
2. Closed Loop Phase — Calculate T_{ar}
3. Open Loop Phase — Calculate A_1
4. PID-design Phase — Calculate new PID-parameters

The PID-design phase occurs after the experiment.

The autotuner communicates with the user by receiving the size of the setpoint change and the size of the deadzone, if one exists. When the tuning is finished the new PID-parameters are displayed.

There is also communication with the process that is tuned. The measurement and control signals from the process are received by the autotuner. The autotuner sends the new setpoint to the process during the Closed Loop phase and the slavecontrol signal during the Open Loop phase.

The phases of the algorithm are implemented in a Sequential Function Chart. Communication is handled in a Function Block Diagram. These can be found in the appendix.

3.3 Detailed Implementation

3.3.1 Initial Phase

The Initial phase contains four steps, Go, NoiseDetect, SteadyStateDetect and Initialization, see Figure 3.2. During this phase the controller is active using the existing parameters. The step Go is used to set some constants and to have somewhere to jump back to when the algorithm is restarted.

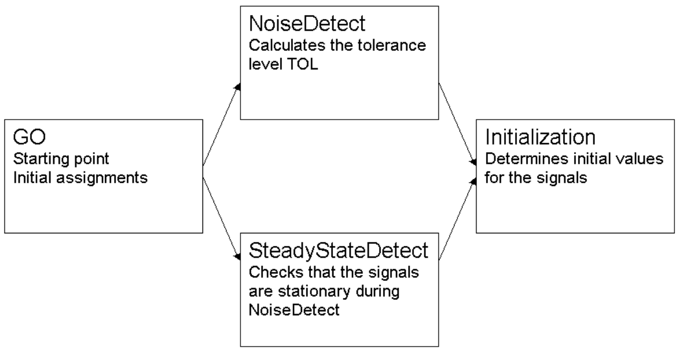
NoiseDetect
Calculates the tolerance level TOL
GO
Starting point
Initial assignments
Initialization
Determines initial values for the signals
SteadyStateDetect
Checks that the signals are stationary during NoiseDetect

Figure 3.2: The steps in the Initial phase.

The two following steps run in parallel. In NoiseDetect the noise level of the measurement and the control signal is measured. The noise level is the basis of the tolerance level that will be used throughout the experiment to determine if the signal is stationary, has reached a certain level or has experienced a load disturbance. The maximum and minimum values of the signal are picked during several 15-second intervals and the average max value, $\overline{max}$, and min value, $\overline{min}$, is computed. The length of the intervals poses a problem if the noise has periodic components. The noise level is calculated as $|\overline{max} - \overline{min}|$. After each interval a rough estimate of the tolerance is calculated for use in the parallel step SteadyStateDetect. The final tolerance will be the noise level multiplied by 1.2 to ensure that it is not too sensitive. If a dead zone is present the noise level will be multiplied by 1.5 instead. Since the noise level is measured while the system is stationary the control signal's influence will be lesser than when a setpoint change occurs and thus the noise will appear lesser. When the tolerance has been calculated the step height set by the user, i.e. the size of the setpoint change, is checked. It should be at least 10 times larger than the tolerance. Otherwise a warning is issued since the results of the tuning cannot be guaranteed.

The SteadyStateDetect step is necessary because the signal needs to be stationary while the noise level is detected to get accurate results. It tests that the measurement signal is stationary for 60 seconds by checking if it stays at the same level, which means that it does not change more than the tolerance level allows. 60 seconds is an arbitrarily chosen time period which seems to work. It means that the NoiseDetect step has four intervals to compute the noise level average from. The measurement signal is averaged online and the latest sample is compared with the average. If the difference is larger than the tolerance level for more than two samples in a row both the NoiseDetect and SteadyStateDetect steps are restarted. Allowing divergence for two samples takes care of sporadic outliers so that unnecessary restarts will not occur too often.

The last step in the Initial phase is called Initialization. Its main purpose is to determine good values for the initial levels of the control and measurement signals. The initial levels will be used later in the experiment. The same test as in SteadyStateDetect is performed to see if the signals are stationary, and the tolerance now has the value determined in NoiseDetect. When stationarity has lasted for half the time compared to SteadyStateDetect, i.e. 30 seconds, the averages of the control and measurement signals are deemed to be stable enough. If stationarity is interrupted, the procedure will be repeated including the two previous steps.

3.3.2 Closed Loop Phase

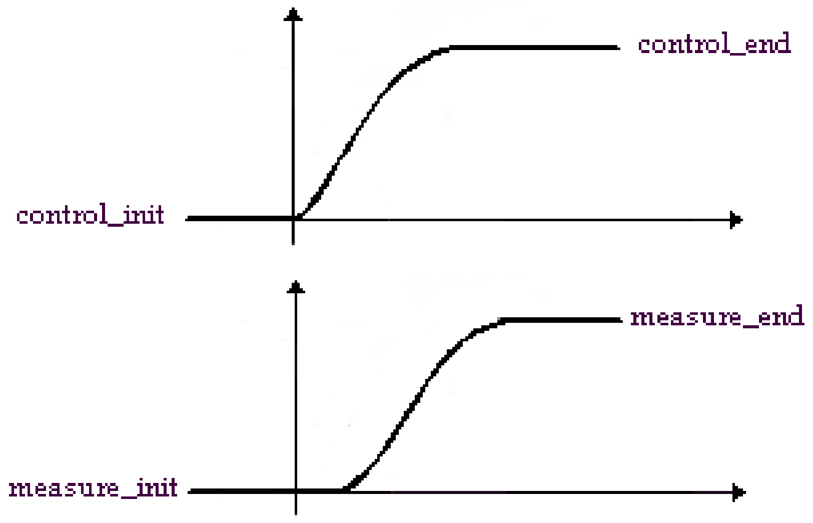

Figure 3.3: The initial and end levels for the control and measurement signals.

A step change is made with the controller active using existing parameters. The dead zone parameter, described in section 1.1, will contribute to a stationary level that differs slightly from the setpoint. The stationary level depends on the controller PI-parameters, slow parameters get a lower level and fast parameters get a higher level than the setpoint. For reaching the setpoint with slow PI-parameters the dead zone is added to the step size when computing the setpoint.

The step direction is set by the user in the beginning of the experiment with the step size. A negative step size gives a closed step in downward direction and a positive step size gives a closed step in upward direction.

The areas of the measurement and control signals are computed as Riemann-sums, using the sample time.

The delay is measured as the time until the signal passes 1% of the step change and is later used in the PID-design phase.

T_{63}, the time it takes to reach 63% of the step change, is measured and twice this time gives a time-limit for how long the system should remain stationary after reaching its new level. This time-limit should not be too short since slow step responses risk being interrupted too early, before the level is really reached. The time-limit should not be too long either, considering that the disturbance probability will increase. When a dead zone exists the parameter T_{63} is badly approximated, but still gives a long enough time-limit for reaching stationarity. When the measurement signal reaches T_{63} a boolean variable is set true and when 1/4 of the time-limit has passed the algorithm allows two samples in sequence to be out of range for the tolerance level. These samples are excluded when computing the average for the end level of the step response.

When the condition for stationarity fails the algorithm will check for load disturbances, see section 3.3.5. If a load disturbance has occurred the algorithm will be terminated. The algorithm will also be terminated if the worst-case time for the step change has passed. A normal Closed Loop phase is estimated to take less than $2 \cdot T_{63}$ + time-limit seconds, the worst-case time for the closed loop is set to twice the normal phase time. To check the control signal the only thing that is known in this phase is the initial level. If the control signal has opposite direction to the step change and has passed its initial level the algorithm will terminate.

Finally, when the stationarity condition is fulfilled, T_{ar} is computed by taking the difference of the normalized signal areas. T_{ar} is used in the next phase as the expected response time for an open loop step change, which corresponds to T_{63} for the closed loop step change. Checks to see that the control signal does not saturate are also performed. The level of the actual step change is compared with the one set by the user in the beginning of the experiment and checked against the noise amplitude.

3.3.3 Open Loop Phase

The controller is deactivated and the algorithm sets the control signal to its initial level. This will result in an open loop step in the opposite direction to the closed loop step.

The area of the measurement signal is computed until the response time is reached, see Figure 3.6. The total area of the signal is also computed, and will be compared with T_{ar} from the previous phase.

The delay in this phase is estimated as the time until the measurement signal changes direction from the stationary level. This delay will later be compared with the approximated delay in the PID-design phase.

A new time-limit for the measurement signal being stationary when the initial level has been reached is calculated as twice the response time. An average of the measurement signal is computed when the signal is stationary, in the same way as in the other phases. This average will be compared to the initial level computed in the Initial phase, and will be used as an accuracy parameter for the experiment in the PID-design phase.

If the stationary condition depending on the tolerance level fails, the algorithm will check if a load disturbance has occurred, see section 3.3.5.

3.3.4 PID-design Phase

The approximations of K_p, T and L are calculated using the areas computed in the Closed Loop and Open Loop phases. Using the approximations the new PID-parameters are calculated by using the AMIGO tuning rules described in the Equation 2.6. Then the credibility of the parameters is judged. For now the only check performed is that they should be non-negative.

3.3.5 Disturbance Detection

Initial Phase

A load disturbance in the Initial phase makes the measurement signal loose contact with the setpoint during the time it takes for the control signal to compensate for the error. When the measurement signal is stabilized the control signal gets a new initial level to keep the measurement signal at the setpoint. If the control signal does not saturate the experiment will go on without influencing the result.

A pulse disturbance will make the measurement signal loose its setpoint for a while and then the signals will return to their initial levels.

When a load or pulse disturbance occurs in the Initial phase, no specific disturbance detection algorithm is used. Instead the steady state detection is restarted and a limitation of the remaining experiment time is used to avoid infinite loop.

Closed Loop Phase

Disturbances in this phase will destroy the integration of the signal areas. The difference between the normalized signal areas, T_{ar}, is later used in the Open Loop phase as an estimate for the response time. The approximation of the three-parameter model, described in Equation 2.1, depends on a correctly estimated response time for computing the area A_1 in the Open Loop phase.

To detect a disturbance the algorithm checks for a change in the measurement signal's direction against the step direction. To determine what direction the measurement signal has the difference between two samples and the tolerance level is used.

```
dir_value :=(current_signal - old_signal);
IF dir_value <= -TOL  THEN
    dir := -1;
ELSIF dir_value >= TOL THEN
    dir := 1;
ELSE
    dir := step_dir;
END_IF
```

Figure 3.4: Code for deciding the direction of the signal.

If `dir` has the opposite direction as `step_dir` the level of the `current_signal` is checked against its initial level and the setpoint, see Figure 3.5. And if the `current_signal` is in the gray area number one or two in Figure 3.5 the program will be terminated.

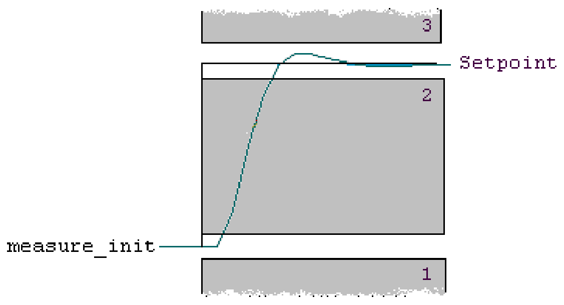

Figure 3.5: Closed loop step with `step_dir` = 1

If a change of direction has occurred in the gray area number three in Figure 3.5 the algorithm ignores it as an overshoot and then it will expect the measurement signal to reach stationarity. If the measurement signal changes direction again it will be considered as a disturbance though it might depend on too fast PI-parameters.

If the measurement signal becomes stationary in the gray area number two in Figure 3.5 the only possibility would be that the control signal has been saturated. This will not be detected until the Closed Loop phase is finished and the real step size is checked against the tolerance level and the user's desired step size.

When `dir` has the same direction as `step_dir` the only check carried out is if `current_signal` has reached a level larger than $2 \cdot (\text{step size} + \text{dead zone})$.

Open Loop Phase

A disturbance in this phase will be considered in a different way than in the previous phase. The phase is divided into two parts; before passing the response time, called the *critical part*, and after the response time, see Figure 3.6. The area which the approximated three-parameter model is based on is computed in the critical part. If a disturbance occurs in this part the algorithm will be terminated. After the critical part a special timer will end the program and allow the experiment to conclude. If a small disturbance has happened and not been detected the computed area between the measurement signal and the setpoint differs from the computed area in the Closed Loop phase. In this case the algorithm will terminate because it can not guarantee that the disturbance has occurred outside the critical part.

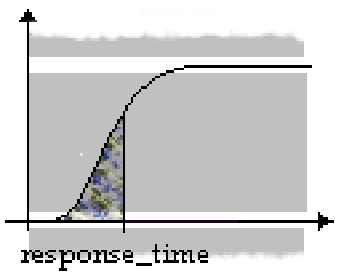

Figure 3.6: Open Loop Step with `step_dir` = 1, the *critical part* is defined as the time until `response_time`.

The detection is based on what direction the measurement signal has in relation to the `step_dir` of the phase. The same code as in the disturbance detection for the Closed Loop phase, see Figure 3.4, is used to get the signal's direction.

Disturbances in the same direction as the `step_dir` are complicated to detect in the critical part because of the different characteristics of the unknown processes. This makes it impossible to detect a too fast falling/raising step response. Instead a check is performed to see if the signal has passed 63% of the step change before leaving the critical part.

A disturbance in the opposite direction of the `step_dir` will be detected in the gray areas in Figure 3.6 but will be handled in different ways depending on what part of the Open Loop phase the disturbance has happened in.

3.3.6 Filtering

When the process is supposed to be stationary, as in the Initial phase and at the end of the Closed Loop and Open Loop phases, the measurement and control signals are averaged using a method that is unbiased for converging averages. The recursive average $\overline{m_N}$ is formulated like this, where N is the number of samples and a is the latest signal value:

$$\overline{m_{N+1}} = \frac{N}{N+1}\overline{m_N} + \frac{1}{N+1}a_{N+1}. \tag{3.1}$$

Chapter 4

Test Results

The autotuner has been tested on three first-order processes with different characteristics, one lag dominated, one delay dominated and one balanced process. Lag dominated means that the process time constant T is much larger than the process delay L and τ is small. Delay dominated means that $L > T$ and τ is large. τ is calculated as $\frac{L}{L+T}$ and used as a basis for the AMIGO tuning rules, see section 2.1.2.

To show the further capabilities of the autotuner, it has also been tested on a first-order balanced process with measure noise and dead zone.

The tests have been performed using processes simulated with TAC Menta, a FBD programming tool used by TAC to implement control applications.

"Original" PI-parameters that give a slow step response without overshoots are chosen for each process. New PID-parameters are generated with the autotuner and tested with a step response. Then T_{63}, the time it takes to reach 63% of the step height, for the new parameters is compared with T_{63} for the "original" parameters.

4.1 Lag Dominated Process

The process characteristics are as follows:

Static gain K_p	$=$	1
Time constant T	$=$	20
Delay L	$=$	1

The real L is larger than 1, probably due to network delays. The τ measured by the autotuner will therefore be larger than the theoretical τ.

Theoretical $\tau = 0.05$ Measured $\tau = 0.16$

The original and new PID-parameters are compared as well as T_{63}.

Original PI-parameters:			New PID-parameters:		
K	$=$	0.5	K	$=$	2.58
T_i	$=$	15.0	T_i	$=$	10.40
T_d	$=$	0	T_d	$=$	1.63
T_{63}	$=$	$49s$	T_{63}	$=$	$18s$

The new parameters give a fast step response with a small overshoot, approximately 5%.

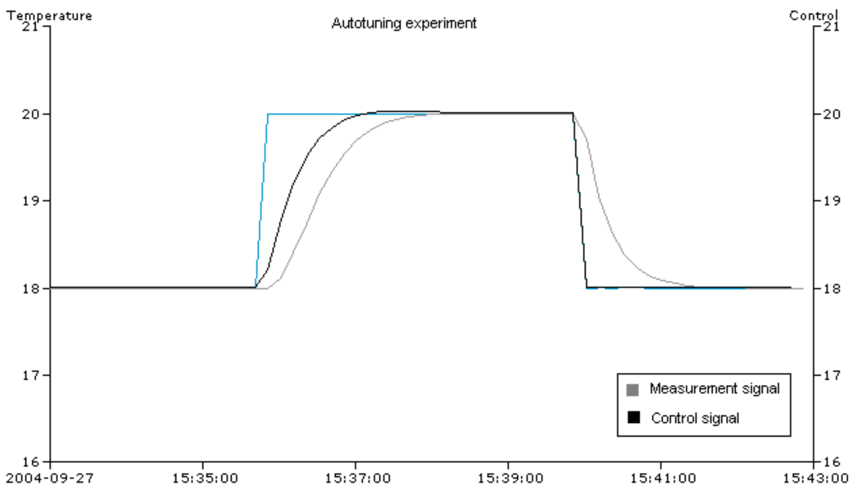

Figure 4.1: Lag dominated process. The autotuning experiment is shown.

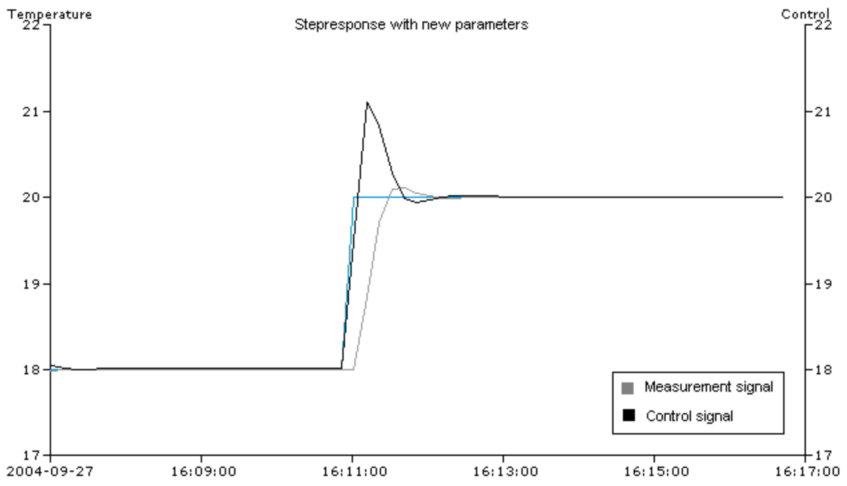

Figure 4.2: Lag dominated process. A step response with the new PID-parameters is shown.

4.2 Delay Dominated Process

The process characteristics are as follows:

Static gain K_p	=	1
Time constant T	=	10
Delay L	=	10

This is theoretically a balanced process but due to delays in the experiment setup the τ measured by the autotuner indicates a delay dominated process. Since this is what the controller will see, the process is classified as delay dominated.

Theoretical $\tau = 0.5$ Measured $\tau = 0.7$

The original and new PID-parameters are compared as well as T_{63}.

Original PI-parameters:			New PID-parameters:		
K	=	0.5	K	=	0.39
T_i	=	15.0	T_i	=	10.21
T_d	=	0	T_d	=	4.21
T_{63}	=	$46s$	T_{63}	=	$42s$

The new parameters give a step response that is only slightly faster than before.

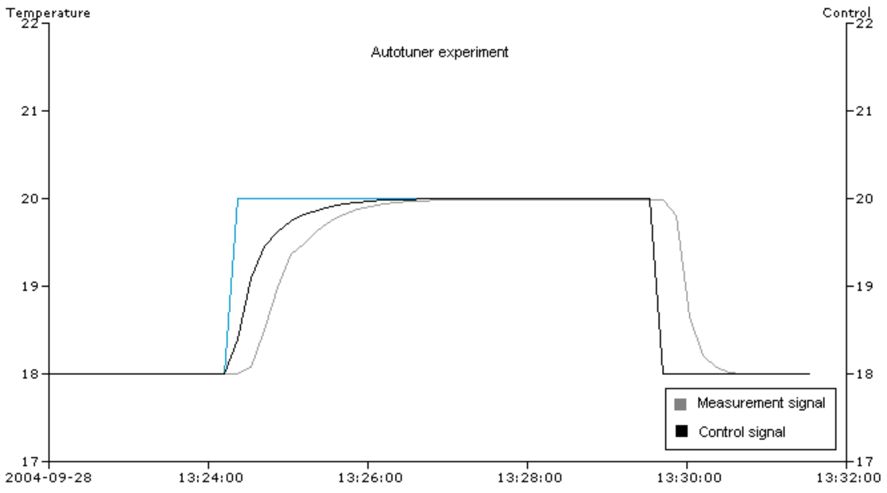

Figure 4.3: Delay dominated process. The autotuning experiment is shown.

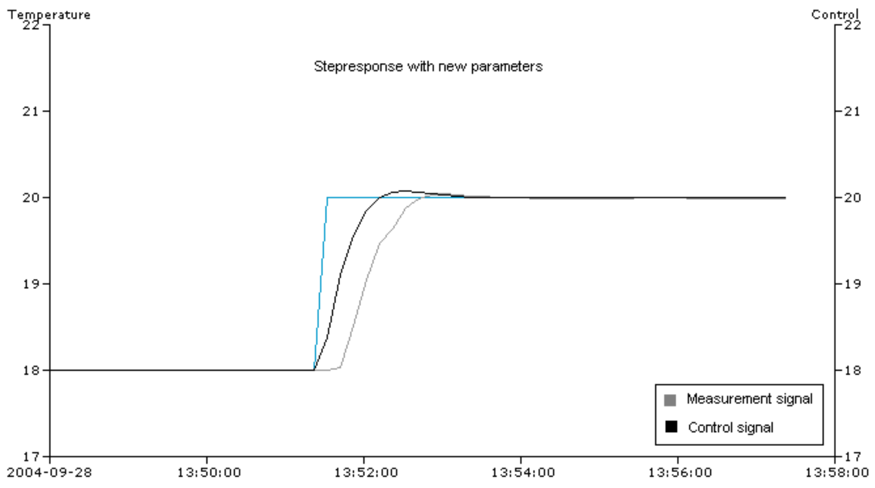

Figure 4.4: Delay dominated process. A step response with the new PID-parameters is shown.

4.3 Balanced Process

The process characteristics are as follows:

Static gain K_p	=	1
Time constant T	=	10
Delay L	=	3

This is theoretically a lag dominated process but due to delays in the experiment setup the τ measured by the autotuner indicates a balanced process. Since this is what the controller will see, the process is classified as balanced.

Theoretical $\tau = 0.23$ Measured $\tau = 0.43$

The original and new PID-parameters are compared as well as T_{63}.

Original PI-parameters:			New PID-parameters:		
K	=	0.5	K	=	0.8
T_i	=	15.0	T_i	=	7.5
T_d	=	0	T_d	=	2.4
T_{63}	=	$46s$	T_{63}	=	$20s$

The new parameters give a fast step response with a small overshoot, less than 5%.

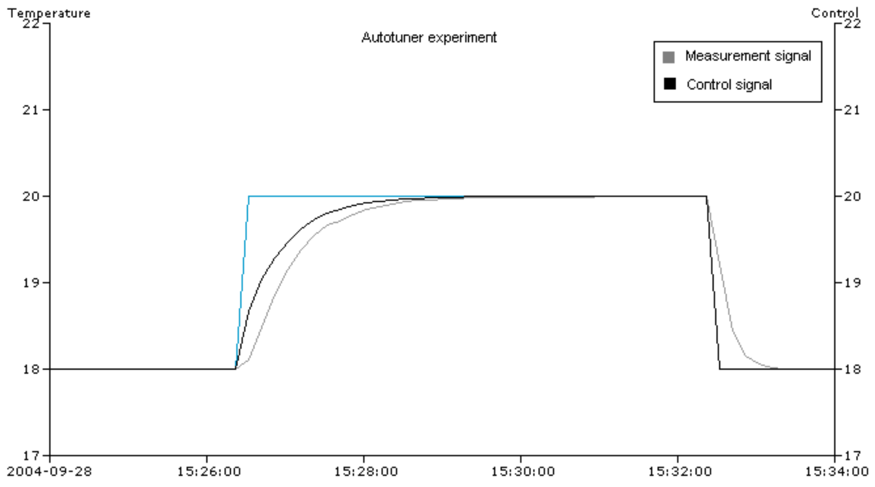

Figure 4.5: Balanced process. The autotuning experiment is shown.

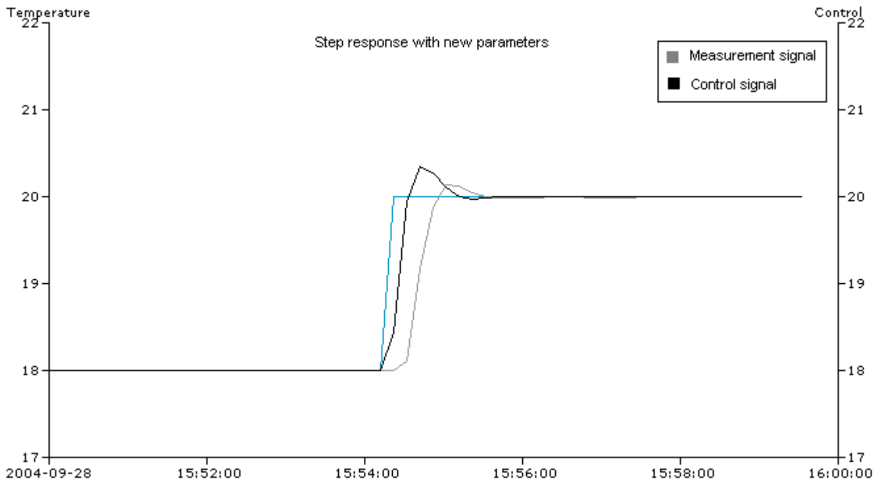

Figure 4.6: Balanced process. A step response with the new PID-parameters is shown.

4.4 Process with Noise and Dead Zone

This process is the same as the balanced process in the previous section, 4.3. The process is affected by measurement noise and the controller has a dead zone = 0.5.

The original and new PID-parameters are compared as well as T_{63}.

Original PI-parameters:			New PID-parameters:		
K	$=$	0.5	K	$=$	0.51
T_i	$=$	15.0	T_i	$=$	6.41
T_d	$=$	0	T_d	$=$	2.51
T_{63}	$=$	$45s$	T_{63}	$=$	$23s$

The new parameters give a fast step response with an overshoot, approximately 5%. The signal levels differ from the setpoint because of the dead zone.

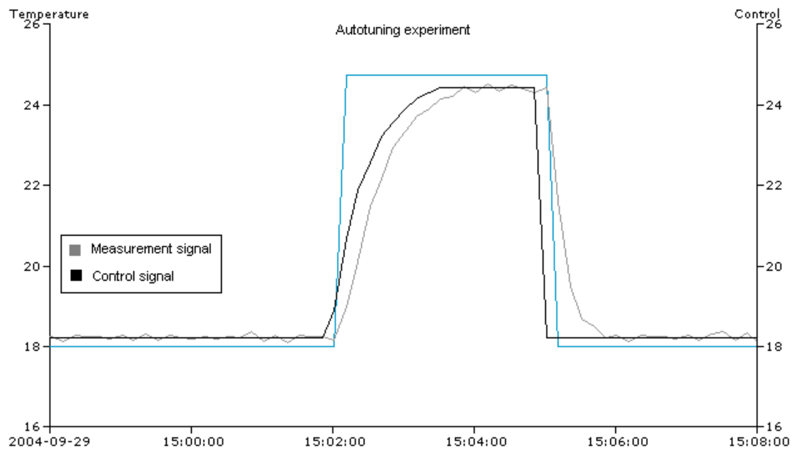

Figure 4.7: Balanced process with noise and dead zone. The autotuning experiment is shown.

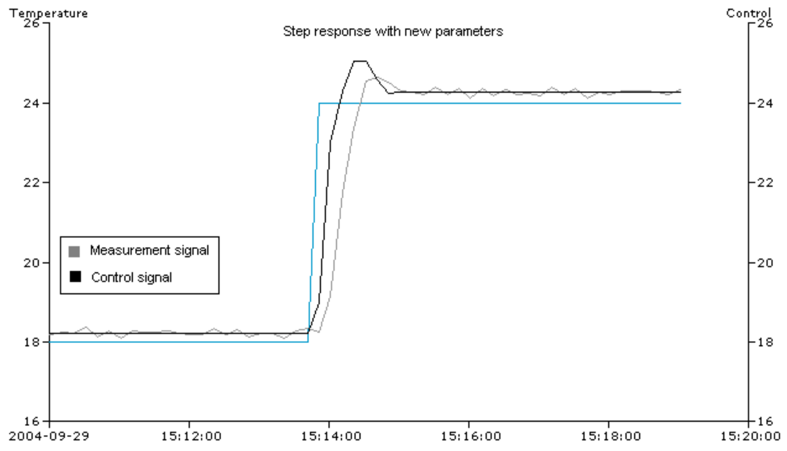

Figure 4.8: Balanced process with noise and dead zone. A step response with the new PID-parameters is shown.

4.5 Testing on an Air Handling Unit

The autotuner has also been tried with the heating process of an air handling unit. It was found that it is hard to bring the process to stationarity, which is a prerequisite for the tuning experiment. Another problem is that the outdoor temperature can change during the experiment and disturb the results.

The autotuner makes one upwards and one downwards step change. It is possible that the process has different dynamics going up or down. This problem could be avoided by making both step changes in the same direction.

Real processes are not linear and time invariant as the simulated processes are. This leads to a lot of interesting problems when trying to apply the autotuner. These problems will have to be taken care of in future versions.

Chapter 5

Conclusions

5.1 Summary

An autotuner based on the Method of Moments combined with the AMIGO tuning rules has been built. It has been tested on various processes and the generated PID-parameters give fast step responses with acceptable overshoots. It is hard to tell if the parameters are optimal but at least they give good results.

During testing we have noticed that the approximations of the process parameters T and L are often far from the original parameters. However, $T+L$ is close to the mark. Even though the approximations are bad, the resulting PID-parameters give good step responses.

The total time of an experiment depends on the controller parameters used and the nature of the process that is tuned. The autotuner uses two steps compared to manual tuning which might need several steps to achieve good controller parameters. This makes the autotuner time-economical when new parameters are needed.

In some experiments the controller parameters given by the autotuner result in an oscillating step response. This could be because some undetected disturbance has occured or it could depend on communication problems. The user should therefore assess the parameters before deploying them.

5.2 User Notes

Some basic knowledge about the process that will be tuned and a bit of control theory will still be needed for the user of the autotuner.

The autotuner should be connected to one PID-controller at the time, not to entire cascaded control loops.

Before running the autotuner the user should make sure that the system is controlled with slow and safe PID-parameters, giving a step response without overshoot. The system should also be in steady state.

The step size of the experiment should be sufficiently large, at least 20 times the approximated noise amplitude. If a dead zone is present in the process, the user must state its size.

The experiment will take at least as much time as two setpoint changes. If the experiment is interrupted an error message will be displayed with information about the possible cause. Load disturbances that occur during the experiment will invalidate the results and the autotuner therefore tries to detect them and interrupt the experiment.

When the experiment concludes, the new PID-parameters are displayed. They can then be manually transferred to the PID-controller.

Most of TACs control systems use only the proportional and integral parts of the PID-controllers. The autotuner will in most cases propose that the controller uses the derivative part as well. When the derivative part is used the system will become more sensitive to high frequency disturbances and therefore a lowpass filter should be used.

The autotuner will give optimal parameters, but the parameters will only remain optimal as long as system conditions remain unchanged. A tuning of a temperature control system that takes place during winter will probably not give parameters suitable for summer control.

5.3 Future Work

- Better identification and filtering of the measurement noise.
- Check the generated PID-parameters to see if they seem probable. Checks can be added, such as comparing the different T_{ar}:s calculated in the Closed Loop and Open Loop phases, comparing the measured L with the calculated L and checking what non-fatal errors have occurred.
- Use τ to make further adjustments to the PID-parameters as described in article [3].
- Use τ to adjust the setpoint weighting of the controller, according to [3].

- Save the information from previous tuning experiments and use it to get better process identification.
- Use the autotuner to generate several parameter sets for use in gain scheduling.
- Further development and testing to make the autotuner fit for real systems.

Appendix A

Figures

A.1 FBD — Overview of the Communication Parameters

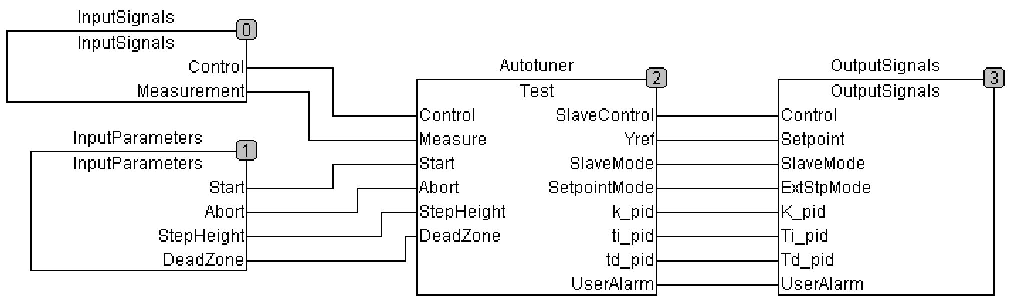

Figure A.1: The Function Block Diagram handles input and output.

A.2 SFC — Overview of the Autotuner Algorithm

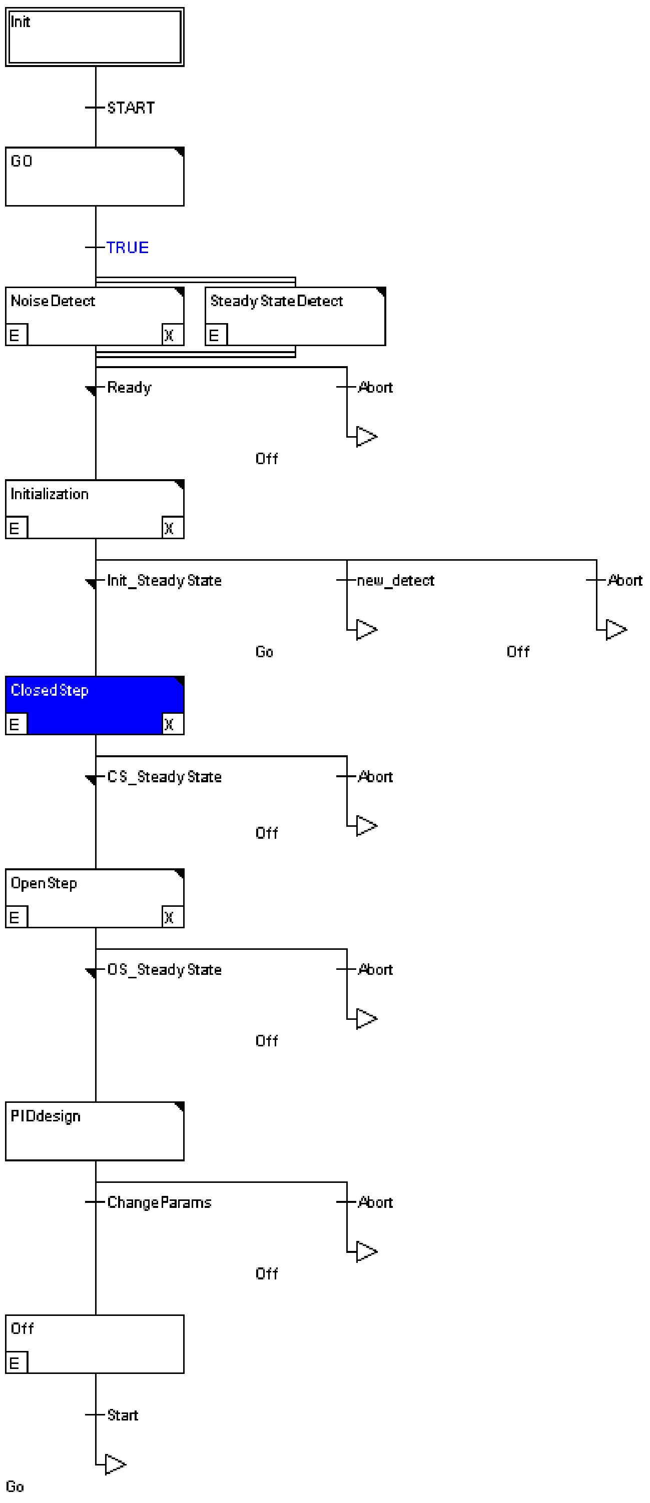

Figure A.2: The Sequential Function Chart of the autotuner algorithm.

Bibliography

[1] A. Leva, C. Cox, A. Ruano. *Hands-on PID autotuning: a guide to better utilisation.* IFAC Professional Brief 2002.

[2] K.J. Åström, T. Hägglund. *PID controllers: Theory, Design and Tuning.* Instrument Society of America, Research Triangle Park, NC 1995.

[3] K.J. Åström, T. Hägglund. *Revisiting the Ziegler-Nichols step response method for PID control.* Journal of Process Control 14 (2004) 635-650.

[4] A. Ingimundarson. *Dead-Time Compensation and Performance Monitoring in Process Control.* Department of Automatic Control, Lund Institute of Technology, Lund 2003.

[5] A. Robertsson. *lecture slides for Nonlinear Control and Servosystems.* Department of Automatic Control, Lund Institute of Technology, Lund 2004.

[6] T. Hägglund, K.J.Åström. *Industrial Adaptive Controllers Based on Frequency Response Techniques.* Automatica vol.27 no.4 (1991) 599-609.