Design of Robust PID Controllers with Constrained Control Signal Activity

Garpinger, Olof

2009

Document Version:
Publisher's PDF, also known as Version of record

Design of Robust PID Controllers with Constrained Control Signal Activity

Olof Garpinger

Department of Automatic Control
Lund University
Lund, March 2009

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

ISSN 0280–5316
ISRN LUTFD2/TFRT--3245--SE

Printed in Sweden,
Lund University, Lund 2009

Abstract

This thesis presents a new method for design of PI and PID controllers with the level of control signal activity taken into consideration. The main reason why the D-part is often disabled in industrial control loops is because it leads to control signal sensitivity of measurement noise. A frequently varying control signal with too high amplitude will very likely lead to actuator wear and tear. For this reason it is extremely important for any PID design method to take this into account.

The proposed controllers are derived using a newly developed design software that solves an IAE minimization problem with respect to H_∞ robustness constraints on the sensitivity- and complementary sensitivity function. The software is shown to be fast, easy to use and robust in giving well-performing controllers.

By extracting measurement noise from the process value of a real plant, one can estimate its effect on the control signal variance. The time constant of the low-pass filter, through which measurements are fed, is varied to design controllers with constrained control signal activity. By comparing control signal variance and IAE, the user is also able to weigh actuator wear to estimated performance.

The proposed PID design method has shown to give very promising results both on simulated examples and real plants such as a recirculation flow process.

Optimal Youla parametrized controllers are used both as a quality check of the designed PI and PID controllers and as a tool for determining when these are valid choices compared to more advanced controllers.

Acknowledgments

After three and a half years at the Department of Automatic Control in Lund there are several people that I would like to thank. The person that, without doubt, has my greatest appreciation is my supervisor, Tore Hägglund. You have always been available, no matter what I have wanted to discuss. I do not think there has been one single meeting with you, after which I have not walked out more motivated than when I walked in. You will always be a role model to me Tore, both as a researcher and as a person.

I am lucky to have many gifted colleagues and I owe some of them a lot for helping me come up with my ideas. Karl-Johan Åström for giving me the idea to rewrite Pontus Nordfeldt's original PID design software and for making me take a closer look at control signal sensitivity due to measurement noise. Andreas Wernrud for introducing me to the ideas of optimal Youla parametrized controllers and for providing me with the software that carries it out. Per-Ola Larsson has also had a lot of influence on my ideas after numerous discussions in "Fabriksrummet". I'd also like to thank Per-Ola and Tore for all time they have spent reading this thesis and coming with good comments.

I want to give my warmest thanks to Rasmus Olsson (pidab ab) and Erik Falkeman (Akzo Nobel Functional Chemicals AB) for making it possible for me to try out my ideas on an industrial plant. Without this help, my thesis would not have felt as complete.

The Department of Automatic Control is also very lucky to have plenty of people that makes work easier for the rest of us. Many thanks to our secretaries Eva Schildt, Britt-Marie Mårtensson, Ag-

neta Tuszynski and Eva Westin for helping and lifting the mood on all of us. Thanks to Leif Andersson and Anders Blomdell for all help regarding computers and for keeping them running smoothly. Thanks to Rolf Braun for making it possible to run my lab tests smoothly.

Besides this, I would also like to thank those brave persons that eat lunch in the city close to every week. Especially Anton and Toivo who have always appreciated the combination of fresh air, good food and exercise. It is also in place to thank my parents for always believing in and supporting me over the years, no matter what I have wanted to do. You should know that I am very grateful for this.

Contents

1

Introduction

This chapter will present some of the main goals of the PID design method proposed in this thesis. It will be followed by a motivation for developing new methods for PID design when there are already several in use giving satisfying results. The chapter will be concluded with an outline of the thesis, summarizing the content of the respective chapters.

1.1 Background

Many industrial processes have rather simple dynamics and they are, therefore, often modeled using straightforward methods like step response tests. This will typically lead to models of the form

$$P(s) = \frac{K_p}{1 + sT} e^{-sL}, \tag{1.1}$$

called First Order systems with Time Delay or just FOTD systems. One way of characterizing these processes is by the normalized time delay

$$\tau = \frac{L}{L + T}. \tag{1.2}$$

When $\tau \gtrsim 0$, the process is called lag dominant, while a process with $\tau \lesssim 1$ is referred to as delay dominant. τ is used extensively in, for

example, [Åström and Hägglund, 2005]. While τ is introduced for FOTD systems, it is a measure used for a much broader variety of systems (through FOTD approximation). For this reason, τ will be frequently used in this thesis as well.

The PI controller is, by far, the most common controller in industry today. Although a PI controller may often be sufficient, the main reason for the rare use of the D-part is due to measurement noise throughput to the control signal and the complexity of choosing yet another parameter. While PI and PID controllers are considered to give rather simple control laws, there are still a lot of poorly tuned controllers in industry. Two of the main reasons being lack of knowledge and time among the operators. As a consequence, many PID controllers have their parameters set to default values. So, it is important for any controller design method to be fast, simple and robust. If not, there is little chance it will be used.

1.2 What are reasonable control goals for industry?

In order to derive a successful controller design tool, no matter the controller structure, it is important to set reasonable demands on the closed loop system. Here follows a brief list of closed loop characteristics that the proposed design tool was built upon:

- Fast suppression of load disturbances

According to [Åström and Hägglund, 2005], the most important duty of the industrial controller is to suppress low frequency changes on the process value. Too high variations in the process value could typically lead to lower product quality.

- Modeling errors and process alteration not leading to instability

As stated already, the dominating method for derivation of an industrial process model is by running a step response test in open loop. This will likely result in an FOTD model like (1.1), no matter if the true process dynamics are much more complex or not. Even if the model is good to start with, it could very well be that the plant changes slowly

over time. In other words, it is important that the closed loop system is robust to such errors and variations.

- Actuator wear and tear should be low

For anyone that wants to design a PID controller where the D-part is active, it is extremely important that the control signal activity does not become too high. In this thesis, it will be assumed that high amplitude and frequency of the control signal are the major villains when it comes to cause actuator wear and tear. In [Buckbee, 2002], the relation between high expenses of valve maintenance and controller tuning are in focus. Buckbee emphasizes the commercial value of proper tuning in order to keep the control signal activity low.

1.3 Is there a need for a new PID design method?

It has already been mentioned that many industrial control loops are poorly tuned, often without use of any systematic design method. There are, however, several design methods that are used rather frequently in industry. The possibly most common of these is the lambda tuning method, which will be described in detail in the next chapter. Another common method is auto-tuning which determines a controller through, for example, relay and/or step response experiments on the process. These kinds of design methods are often incorporated in commercial software programs.

While many control loops can be controlled sufficiently well with these methods, they often lack guarantees that all three criterias in Section 1.2 are addressed. It could, for instance, be that a PI design gives a robust closed loop system with low control signal activity, but rather bad performance. A PID controller may then be a better option for fulfilling all demands on the system. The PID design method proposed in this thesis will take all three criterias into account simultaneously and should thus more likely be giving good control. Another fundamental aspect of the new design method is to choose either PI or PID control structure depending on which is the most suitable for the given process. For example, there is no reason to design a more advanced controller if a PI controller is just as good.

Furthermore, the proposed PID design method is software based, solving an optimization problem off-line (see [Garpinger and Hägglund, 2008]). Many other design methods are formula based, with their origin from some advanced optimization, like [Kristiansson and Lennartson, 2006] and [Hägglund and Åström, 2004]. But, if it is possible to solve the optimization directly on a computer, should it then not be better than using a generalized formula? It was shown in [Garpinger and Hägglund, 2008] that the proposed software program has good potential of being both a source for PID knowledge and a tool for further research. Such a tool can, in other words, be beneficial to people in industry in need of more PID education, at the same time as it is used by advanced researchers.

In order to get a quality check of the PI and PID controllers, derived by the proposed design method, Youla parametrized controllers have been used. A Youla optimization tool was used to derive nearly optimal linear controllers with the same criterias as the PID design. This way, one can see how close the controller designs are to the limit of performance. There will also be an attempt to use Youla controllers for giving a more general picture of when PI and PID controllers are useful compared to more advanced controllers.

1.4 Outline

Here follows a brief summary of the different thesis chapters.

Chapter 2

In Chapter 2 follows a presentation of the closed loop system in focus. In conjunction to this, an optimization problem is stated such that the criterias in Section 1.2 are taken into consideration. The chapter is also concerned with giving an introduction to the way Youla parametrized controllers can be optimized to give good estimates of the best possible linear controllers. The optimization problem is finally translated into a Youla parametrization framework for convex optimization.

Chapter 3

Chapter 3 introduces four other PID design methods, some sources of inspiration and a few other related methods.

Chapter 4

Chapter 4 contains a detailed explanation of how the proposed software tool for optimization of robust PID controllers works. The Nelder Mead algorithm is presented together with a motivation of its worth. The chapter is concluded with several examples, comparing the PI and PID controllers to those derived with the MIGO method (for a description of the MIGO method, see Section 3.1).

Chapter 5

The purpose of Chapter 5 is to describe how the PID design software can be used to constrain the control signal variance due to measurement noise. Challenges in connection to e.g. measurement data logging and variance estimation are also discussed. A suggestion on how to proceed with the method on a real plant is also presented. Several examples will conclude the chapter and show the potential of the method. Optimized Youla controllers will be used to validate the quality of the PI and PID controllers.

Chapter 6

The proposed PID design method has been tried out on an industrial plant. The chapter describes initial tests, modeling of the plant, controller design and results when the proposed controllers were run on a recirculation flow process.

Chapter 7

When is it justified to use PI and PID controllers rather than more advanced controllers? Chapter 7 will aim at giving at least a hint of the answer to this question. The research on this topic is, however, not yet finished, so the discussion will mainly serve as a source of inspiration. It should, however, still be possible to draw quite a few interesting conclusions from the results.

Chapter 8

The very last chapter will summarize the most important conclusions and present some ideas for future work in the research area.

2

Design Specifications

The following chapter will define the closed loop system of interest in this thesis. Given this, an optimization problem for design of robust PID controllers with adjustable control signal noise reduction will be stated. In addition to this, there will be two sections describing Youla parametrized controllers and how to rewrite the closed loop system to fit it into this framework.

2.1 Design criterias

The main purpose of proposed PID controller design tool is to work well for systems common in process industry. The kind of plants encountered there are often stable, monotone and primarily affected by low frequency load disturbances. This is also why the regulator problem is considered in this thesis rather than the tracking problem. A good tracking performance can be achieved using feed-forward design after the controller have been designed for disturbance rejection, see e.g. [Åström and Hägglund, 2005] for details.

In order for the controller design to work well on a real process, with model $P(s)$, it is important to take all system signals into consideration, especially if optimization is used. If not, some signals may easily blow out of proportions or the closed loop could become sensitive to changes in the process. Figure 2.1 shows a block diagram of the system that the PID (or PI) controller, $C(s)$, is designed for. There are two external signals entering the system, the load disturbance d (mainly

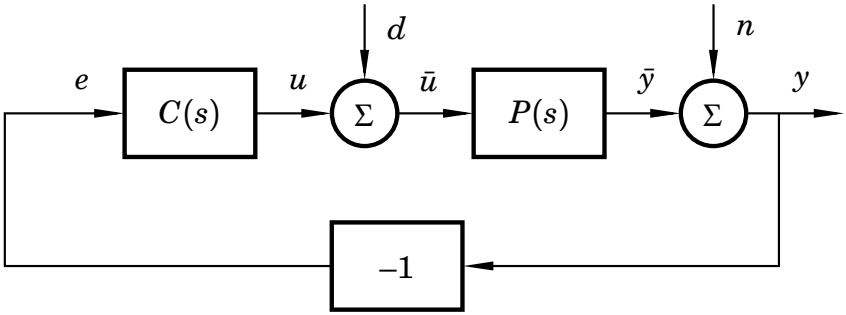

Figure 2.1 A load disturbance, d, and measurement noise, n, act on the closed loop system with process $P(s)$ and PID controller $C(s)$.

low frequency) and measurement noise n (assumed high frequency). A popular name for the four transfer functions from disturbances to control signal u and measurement signal y is the gang of four. These are

$$\begin{pmatrix} T(s) & P(s)S(s) \\ C(s)S(s) & S(s) \end{pmatrix} = \begin{pmatrix} \dfrac{P(s)C(s)}{1+P(s)C(s)} & \dfrac{P(s)}{1+P(s)C(s)} \\ \dfrac{C(s)}{1+P(s)C(s)} & \dfrac{1}{1+P(s)C(s)} \end{pmatrix},$$

where the top left corner function, $T(s)$, is called the complementary sensitivity function and the function in the bottom right corner, $S(s)$, is named the sensitivity function.

The PID controller is assumed to be on parallel form

$$C(s) = K\left(1 + \frac{1}{sT_i} + sT_d\right) \cdot \frac{1}{1 + sT_f + (sT_f)^2/2},$$

with a second order low pass filter on the measurement signal. In case a PI controller is designed instead, it will have the form

$$C(s) = K\left(1 + \frac{1}{sT_i}\right) \cdot \frac{1}{1 + sT_f}.$$

T_f is chosen, in both cases, to weigh the degree of measurement noise rejection against closed loop performance. A low value on T_f will generally result in better load disturbance rejection, but may lead to strong noise amplification in the control signal. Therefore, choosing T_f is a balance between getting good performance and keeping the actuator wear low. A large portion of this thesis will be focused on how to choose T_f in a sensible way. The low-pass filter has been chosen to be of second order, even though it is common in industry to only have low-pass filter of order one and then often on the derivative part of the controller only. The reason for choosing a second order filter is because it guarantees roll-off on all sensitivity functions (The Gang of Four). It does also make sense to filter the P-part of the controller as it could otherwise lead to considerable noise throughput to the control signal. It should, however, be possible to use a different low-pass filter setup for the method described in this thesis if desired, but it would require some modifications to the software.

The objective of the proposed PID design method is to find the PID controller giving the least Integrated Absolute Error (IAE) value,

$$IAE = \int_0^\infty |e(t)|dt,$$

when a load disturbance d, modelled as a step, is acting on the closed loop system. The optimization is done under the constraints that the open loop Nyquist curve is tangent to one or two prespecified circles in the complex plane, without entering either of them (see Figure 2.2). These two circles will be called the M_s- and M_p-circle (although they will sometimes be referred to as the M-circles), which sizes and positions are given by

$$M_s = \max_\omega |S(i\omega)|, \quad M_p = \max_\omega |T(i\omega)|,$$

hence the names. According to, for example, [Åström and Hägglund,

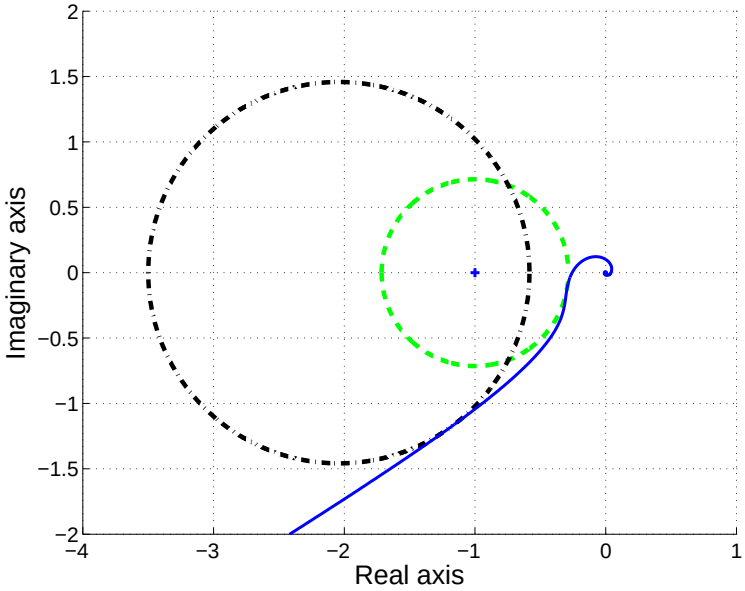

Figure 2.2 The M_s-circle (dashed), M_p-circle (dash-dotted) and the open loop Nyquist curve (solid) when the optimization criterias are fulfilled.

2005] the center points and radius of the two circles are given by:

$$C_{M_s} = -1, \quad R_{M_s} = \frac{1}{M_s},$$
$$C_{M_p} = -\frac{M_p^2}{M_p^2 - 1}, \quad R_{M_p} = \frac{M_p}{M_p^2 - 1},$$

where the center points are denoted with C, the radius with R and the indices refer to the circles.

The resulting, non-convex, optimization problem can be written as

$$\min_{K,T_i,T_d \in \mathcal{R}^+} \int_0^\infty |e(t)|dt = IAE_{load} \tag{2.1}$$

$$\text{subject to } |S(i\omega)| \leq M_s, \quad |T(i\omega)| \leq M_p, \quad \forall \omega \in \mathcal{R}^+,$$
$$|S(i\omega^s)| = M_s \text{ and/or } |T(i\omega^p)| = M_p$$

where $e(t)$ is the control error, ω^s are frequencies for which the open loop frequency response is tangent to the M_s-circle and vice versa for ω^p on the M_p-circle. Either ω^s or ω^p could be an empty vector, but not at the same time. Small M_s- and M_p-values result in large circles. In the software, the maximum allowed M_s- and M_p-values can be prespecified by the user. The M_s- and M_p- criterias are known to set the closed loop robustness towards process variations, disturbances and nonlinearities as described in [Åström and Hägglund, 2005]. $M_s = M_p = 1.4$ has been chosen as default values in the optimization software, resulting in 41.8° phase margin and a gain margin of 3.5. Some sources list values spanning from 1.2 to 2 giving reasonable robustness. MIGO on the other hand uses a simplified robustness criterion called the M-circle, defined as the smallest circle that encloses both the M_s- and M_p-circle.

In Chapter 5, a constraint on the control signal variance, due to measurement noise, will be added. This constraint is, therefore, set on $C(s)S(s)$ which will be called $S_k(s)$ in the future. When the spectrum of the measurement noise is taken into consideration, $S_k(s)N(s)$ will be used instead. $N(s)$ is assumed to be the transfer function filtering white noise into the current measurement noise. In this thesis a variance constraint on the control signal was selected, such that

$$\|S_k\|_2^2 = \frac{\sigma_u^2}{\sigma_n^2} \leq V_k,$$

where σ_n^2 is the variance of the measurement noise and σ_u^2 is the variance of the control signal that the noise results in. V_k is the design parameter and $V_k = 1$ will correspond to u and n having the same variance.

One could argue that the optimization problem lack a warranty for time delay robustness. Such a constraint will, however, not be used in this thesis. Main reason being that PI and PID controllers seldom lead to closed loop systems with poor robustness towards time delay variations. The issue will, however, come up when comparing the designs to more advanced Youla controllers in Chapter 5.

2.2 Youla parametrized controllers

Assume that the process, $P(s)$, is both stabilizable and detectable. A more general way of representing a closed loop system around $P(s)$ is shown in Figure 2.3. The feedback scheme presented in Figure 2.1 is just one possible loop among all that can be represented this way. The signals in w are exogenous disturbances acting on the closed loop, such as: measurement noise, load disturbances and reference signals. The exogenous outputs, z, on the other hand represent the closed loop signals one wants to control. u is simply the control signal, while e are generally measurements entering the controller. $\bar{P}(s)$ is a more generalized representation of the process and will be used here when deriving, so called, Youla parametrized controllers.

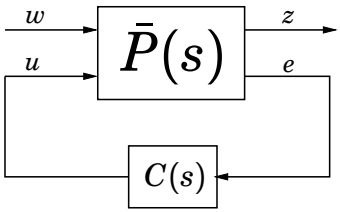

Figure 2.3 A general representation of a closed loop system. $\bar{P}$ is the generalized process that will be used to find Youla parametrized controllers.

It is known (see for instance [Boyd *et al.*, 1990] and [Wernrud, 2008]) to be possible to parametrize all stable control loops to depend affinely on the transfer function Q, defined as

$$Q(s) = \frac{C(s)}{1 + P(s)C(s)}. \tag{2.2}$$

Many known control problems can then be written such that they are closed loop convex in Q. Some examples of closed loop convex cost functions and constraints are:

- l_1- and l_2-norm costs.
- Time domain envelop constraints on the closed loop.
- Frequency domain upper bound constraints.
- Upper bound on the H_2-norm of a closed loop transfer function.

The stability of the controller $C(s)$ can, however, not be guaranteed, nor can its order be constrained. This means that the controller could end up having very high order and potentially be unstable. The closed loop must still be stable of course.

When the closed loop convex optimization problems are solved, it is commonly done in discrete time. One way of optimizing the controller is by defining Q as an FIR filter,

$$Q(z) = \sum_{l=0}^{N_Q-1} q_l z^{-l},$$

and then have an optimization solver determine the coefficients q_l. The controller can then readily be derived from (2.2). $C(z)$ will be an estimate of the best possible linear controller for the optimization problem. The order of the FIR filter will determine how close to the limit of performance one gets.

The optimization tool used here for deriving Youla parametrized controllers is described in [Wernrud, 2008]. The main use of these controllers in this thesis is to show whether or not the PI and PID controller designs can come close to the limit of performance. As stated in [Boyd *et al.*, 1990], this would be a very strong point in favour of a simple controller.

For more information on Youla parametrized controllers one can read any of the sources; [Norman and Boyd, 1989], [Boyd *et al.*, 1990], [Boyd and Barratt, 1991], [Boyd and Barratt, 1992], [Wernrud, 2008].

2.3 System description for Youla optimization

In order to derive Youla parametrized controllers for the given problem, the process has to be written on the general form shown in Figure 2.3. For this reason, the process $P(s)$ is transformed to state space form

$$\begin{aligned}\dot{x}(t) &= Ax(t) + B\bar{u}(t) = Ax(t) + B(d(t) + u(t))\\ \bar{y}(t) &= Cx(t).\end{aligned}$$

The exogenous inputs, w, and outputs, z, are defined as

$$w(t) = \begin{pmatrix} d(t) \\ n(t) \end{pmatrix}, \quad z(t) = \begin{pmatrix} \bar{y}(t) \\ \bar{u}(t) \end{pmatrix},$$

such that the outputs from the general system, $\bar{P}$, are

$$\begin{pmatrix} z(t) \\ e(t) \end{pmatrix} = \begin{pmatrix} \bar{y}(t) \\ \bar{u}(t) \\ e(t) \end{pmatrix} = \begin{pmatrix} Cx(t) \\ d(t) + u(t) \\ -Cx(t) - n(t) \end{pmatrix}.$$

The complete state space form of $\bar{P}$, is thus

$$\dot{x}(t) = Ax(t) + \begin{pmatrix} B_w & B_u \end{pmatrix} \begin{pmatrix} w(t) \\ u(t) \end{pmatrix} = Ax(t) + \begin{pmatrix} B & 0 & B \end{pmatrix} \begin{pmatrix} d(t) \\ n(t) \\ u(t) \end{pmatrix}$$

$$\begin{aligned}\begin{pmatrix} z(t) \\ e(t) \end{pmatrix} &= \begin{pmatrix} C_z \\ C_y \end{pmatrix} x(t) + \begin{pmatrix} D_{zw} & D_{zu} \\ D_{yw} & D_{yu} \end{pmatrix} \begin{pmatrix} w(t) \\ u(t) \end{pmatrix}\\ &= \begin{pmatrix} C \\ 0 \\ -C \end{pmatrix} x(t) + \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} d(t) \\ n(t) \\ u(t) \end{pmatrix}.\end{aligned}$$

Since P often contains time delays and the Youla optimization software needs a discrete time system, the process model is discretized with a sampling time h, before $\bar{P}$ is derived. The form of the state space realization is, however, not changed by the discretization.

By block diagram calculations, one can easily derive the closed loop system transfer matrix

$$H = \begin{pmatrix} \dfrac{P}{1+PC} & -\dfrac{PC}{1+PC} \\ \dfrac{1}{1+PC} & -\dfrac{C}{1+PC} \end{pmatrix},$$

which holds all sensitivity functions of interest for the optimization problem.

When the Youla optimization is run, the order of the Q-filter needs to be above a certain number (changes depending on the process) in order for the final controller to hold an integrator.

The sampling time selection is especially vital for delay dominant systems. The sampling interval should be chosen short enough to cover all vital parts of the system dynamics. At the same time it should not be too small. A sampling interval shorter than the time delay will be represented in extra states as described in for instance [Åström and Wittenmark, 1997]. If the sampling time, h, is a lot smaller than the time delay, L, the discrete time system will be of very high order, typically leading to very slow optimization and potentially even numerical problems.

3

Sources of Inspiration and Related Design Methods

In this chapter, four methods for design of PID controllers will be presented. They all have in common that they will be frequently referred to later within this thesis. In addition to these, the chapter will be concluded by a brief overview of other related methods and some sources of inspiration.

3.1 Closely related PID design methods

There are many PID design methods available today and some of the most famous are collected and analysed in [Åström and Hägglund, 2005]. A few of these will briefly be presented in this section as well, together with a newer method developed in [Nordfeldt, 2005].

Lambda tuning

The lambda tuning method which was first introduced in [Dahlin, 1968] has grown to be very commonly used in industry (see e.g. [Olsen and Bialkowski, 2002]). The method is built around the idea of pole placement in relation to the process time constant.

An FOTD model, like (1.1), is first derived through for instance a step response experiment on the process. The integral time is then

chosen to be $T_i = T$ such that the open loop transfer function becomes

$$P(s)C(s) = \frac{K_p K}{sT} e^{-sL},$$

in case of a PI controller. Note that the process pole has been canceled by the controller zero. Using a Taylor series approximation of the time delay will now make it possible set an approximate closed loop system time constant, which will be called T_{cl}. The lambda tuning formulas for PI parameters thus becomes

$$\begin{aligned} K &= \frac{1}{K_p} \frac{T}{L + T_{cl}}, \\ T_i &= T. \end{aligned}$$

There are PID controller formulas as well, but these are not as commonly used.

As stated in [Åström and Hägglund, 2005], the lambda tuning rules will give good control under certain circumstances. The cancellation of the process pole will, however, give sluggish load disturbance responses for lag-dominant processes ($\tau \gtrsim 0$). The method do have several advantages as well, since it is both a simple and intuitive method to use. It should, however, be pointed out that there are no guarantees on good performance, robustness or low control signal activity. Take for instance the PI design for

$$P(s) = \frac{1}{0.1s + 1} e^{-s}$$

as an example. According to [Åström and Hägglund, 2005], choosing $T_{cl} = 3T$ is normally considered to give good robustness. With this process, however, one would have to choose $T_{cl} \approx 16.8T$ in order to get the same robustness as the one, by default, given by the proposed PID design software. Also, setting this T_{cl} will lead to a controller giving very poor performance. Cases like this can of course be taken care of by people with good control knowledge, but it corresponds to unnecessary work if there are methods giving good controllers right away.

The MIGO and AMIGO methods

The method used to receive initial controllers in the proposed software algorithm is called AMIGO (see [Hägglund and Åström, 2004]), which is a tool for robust PID (and PI) synthesis. To understand AMIGO, it is also important to understand the MIGO method (see [Panagopoulos *et al.*, 2002]) for PI and PID design.

The optimization problem that the MIGO design deals with is very similar to (2.1). But instead of minimizing over the IAE-value, it uses the Integrated Error,

$$IE_{load} = \int_0^\infty e(t)dt,$$

as cost function and the M-circle as robustness constraint, to determine the PID parameters.

The IE cost is proportional to $1/k_i = T_i/K$, which reduces the problem to maximizing the k_i-gain over the robustness area. Such an optimization problem is advantageous as it can be solved by known optimization routines. There are, however, a few drawbacks related to the use of the IE-value and it can sometimes be an insufficient cost function. The IE-cost will decrease if the load disturbance response assumes negative values, which means that oscillating responses may be preferred. The M-circle criterion will prevent the worst of the oscillatory responses from occurring, but there are examples when even this is not enough. The problem has been avoided in the final MIGO-algorithm by a fix that searches for the the best controller for which the cost function has a defined gradient.

A drawback with the MIGO method is that it requires quite a bit of preparational work before it can run.

The AMIGO design is basically a set of formulas yielding K, T_i and T_d. These were derived using MIGO on a test batch, which includes 134 systems commonly encountered in process industry

$$
\begin{aligned}
P_1(s) &= \frac{e^{-s}}{1+sT}, \\
T &= 0.02, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 1, \\
&\quad 1.3, 1.5, 2, 4, 6, 8, 10, 20, 50, 100, 200, 500, 1000 \\
P_2(s) &= \frac{e^{-s}}{(1+sT)^2}, \\
T &= 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 1, \\
&\quad 1.3, 1.5, 2, 4, 6, 8, 10, 20, 50, 100, 200, 500 \\
P_3(s) &= \frac{1}{(s+1)(1+sT)^2}, \\
T &= 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 2, 5, 10 \\
P_4(s) &= \frac{1}{(s+1)^n}, \\
n &= 3,\ 4,\ 5,\ 6,\ 7,\ 8 \\
P_5(s) &= \frac{1}{(1+s)(1+\alpha s)(1+\alpha^2 s)(1+\alpha^3 s)}, \\
\alpha &= 0.1, 0.2, 0.3, 0, 4, 0.5, 0.6, 0.7, 0.8, 0.9 \\
P_6(s) &= \frac{1}{s(1+sT_1)} e^{-sL_1}, \\
L_1 &= 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 0.9, 1.0, \quad T_1 + L_1 = 1 \\
P_7(s) &= \frac{1}{(1+sT)(1+sT_1)} e^{-sL_1}, \qquad T_1 + L_1 = 1, \\
T &= 1, 2, 5, 10 \qquad L_1 = 0.01, 0.02, 0.05, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0 \\
P_8(s) &= \frac{1-\alpha s}{(s+1)^3}, \\
\alpha &= 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1 \\
P_9(s) &= \frac{1}{(s+1)((sT)^2 + 1.4sT + 1)}, \\
T &= 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0.
\end{aligned}
\tag{3.1}
$$

Secondly, each and every process in the batch was approximated as an FOTD, (1.1). PID-parameter formulas were then derived through

parameter fittings with respect to normalized time delay, τ, resulting in

$$K^A = \frac{1}{K_p}\left(0.2 + 0.45\frac{T}{L}\right)$$
$$T_i^A = \frac{0.4L + 0.8T}{L + 0.1T}L$$
$$T_d^A = \frac{0.5LT}{0.3L + T}.$$

The index A denotes the AMIGO parameters. There are also AMIGO formulas for PI control, namely

$$K^A = \frac{0.15}{K_p} + \left(0.35 - \frac{LT}{(L+T)^2}\right)\frac{T}{K_p L},$$
$$T_i^A = 0.35L + \frac{13LT^2}{T^2 + 12LT + 7L^2},$$

which will be used as well.

In contradiction to the lambda tuning method, AMIGO has some guarantees on both robustness and performance. The PID parameters, however, tend to blow up in proportions for lag dominant systems. Take for instance the case when $K_p = 1$, $L = 1$ and $T = 500$. This gives $\tau = 0.002$, $K^A = 225.2$, $T_i = 7.85$ and $T_d = 0.50$. The high proportional gain makes this a controller that people in industry would be very reluctant to implement due to measurement noise throughput. In [Åström and Hägglund, 2005] there is a suggested way of detuning the AMIGO rules, but it does not take the measurement low-pass filter into consideration.

Pontus Nordfeldt's Design Method

A further development of the MIGO method, and largely based on the same optimization problem used in this thesis, was presented in [Nordfeldt, 2005]. Nordfeldt wrote a Matlab script designing PID controllers that minimize IAE with respect to an M-circle robustness constraint. The algorithm used is based on extensive gridding of the cost function, while choosing K in the same fashion done in this work. The main goal

of Nordfeldt's method was to find controllers for processes like

$$G(s) = G_1(s)e^{-sL_1} + G_2(s)e^{-sL_2},$$

where G_1 and G_2 are stable, rational, linear transfer functions. The case $L_1 \neq L_2$ is, for instance, of interest when a system with two inputs and two outputs is dynamically decoupled. While the controller design for advanced process structures was the main aim of Nordfeldt, this thesis focuses more on the software tool solving the optimization problem and the usefulness of the same.

3.2 More distantly related methods and sources of inspiration

There are many good reasons to have a software based tool for control design and analysis, like the one presented here. In [Åström and Hägglund, 2001] it is pointed out that it would be of great value to have software that can give persons with moderate knowledge of PID controllers a possibility to experiment on those and at the same time be able to use the program to build controllers for a real plant, by incorporating it into an auto-tuning procedure. Besides the proposed design tool, which is freeware, there are already several commercial software packages able to provide PID designs using a variety of methods. Many of these are collected in [Li *et al.*, 2006]. Another method with very similar features to the proposed one is presented in [Oviedo *et al.*, 2006].

There are several papers that acknowledge the importance of four parameter design for PID controllers. One of these is [Isaksson and Graebe, 2002] that also points out the importance of knowing the structure of the controller implementation. The authors believe that there is plenty of use for the D-part in industrial applications. They think that the main reasons for PI controllers still being dominant in industry are lack of four parameter design methods and the ease of tuning PI controllers. In addition to this, the authors ask for model-based methods that can be implemented in software. Another source that highlights tuning of low-pass measurement filters is [Luyben, 2001].

Two other sources that use similar methods to the one proposed here are [Marlin, 1995] and [Kristiansson and Lennartson, 2006]. They

both have similar optimization problems and take noise into consideration. It should be mentioned that Marlin design controllers with respect to all three criterias given in Section 1.2, just like the proposed design method.

4

A Software Tool for Robust PID Design

This chapter will focus on the Matlab software tool that was written in order to solve the optimization problem described in Section 2.1. Note that this tool does not take the control signal variance constraint into consideration. Deriving controllers with that constraint added will be covered in the next chapter.

The aim of the new tool is to have a robust and fast way of designing PID controllers, solving the optimization problem (2.1). The program works on any stable, linear, process model with a phase shift of at least –90° for high frequencies. The software should also be easy to modify, educational and a tool for further research. As a part of accomplishing these goals, the software can be downloaded from `http://www.control.lth.se/user/olof.garpinger/`.

The chapter will start with a motivation of the algorithm used in the software. All methods used will thereafter be explained in detail. On top of this, there is a short section about design of PI controllers. Last, there will be several examples showing that the software tool works well.

4.1 Algorithm overview

A non-convex optimization problem like (2.1) may have many local minima. It is therefore hard to guarantee that the solution obtained always

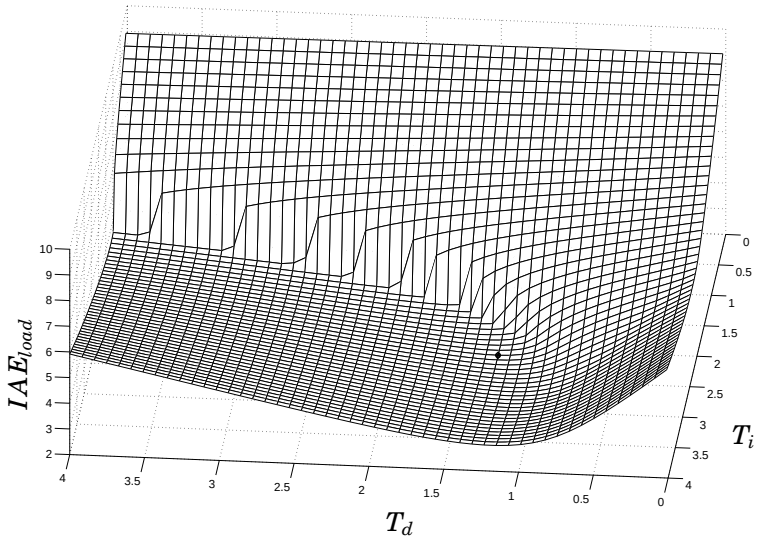

Figure 4.1 In the new design software, the cost (IAE-value), can be drawn as a function of the PID parameters T_i and T_d. The system in this particular case is the fourth order lag filter $G(s) = (s+1)^{-4}$. The minimum is marked in the picture.

is the global solution. It is also difficult to draw any general, analytical, conclusions as the problem is far from trivial. The method of gridding does, however, give a possibility of drawing surface plots of the cost function. These can be used to determine whether or not it is likely that a given solution is in fact the global minimum. This is also the major reason why gridding is an optional optimization method in the proposed design program. Figure 4.1, for instance, shows such a surface plot for the fourth order system $G(s) = (s+1)^{-4}$. Analysis of many cost function surfaces has shown that if not all, then at least a majority of them only have one minimum. This finding gave the idea to use a faster and more advanced optimization tool than gridding, namely the Nelder Mead (NM) method, [Nelder and Mead, 1965], in order to find the minimum in the T_i-T_d plane.

The proportional gain, K, is chosen such that the open loop Nyquist curve is tangent to one or both robustness circles. It is reasonable to choose such a gain since it is likely to maximize the speed of the closed loop system. Optimizations with more general tools, such as Optimica (see [Åkesson, 2008]), have also shown that solutions with active constraints are generally preferred. There are, however, some cases when the solution is not expected to give active robustness constraints, which will be dealt with in the end of Section 4.2.

The algorithm can be summarized by

1. Given a linear process model, initial PID parameters are chosen using the AMIGO method.
2. Nelder Mead optimization finds the PID controller giving the minimum cost in the T_i-T_d plane.
    1. For each (T_i,T_d), a proportional gain, K, is found such that the constraints are fulfilled.
    2. Simulations are used to calculate IAE-values in the points through which the Nelder Mead optimization proceeds.

An interactive program menu has been added to make it possible for the user to change a number of settings in the algorithm as well as for the presentation of the results. When the program is run in Matlab, the menu will come up unless the opposite is stated by the user. New default values for the optimization can also be set as input parameters. This is especially useful for batch runs, when one may want to choose the settings before a number of program runs are started. An advanced user should easily be able to modify the program to, for instance, change the optimization method or at least change the cost function. For those that download the software tool, there is a small tutorial included which explains how to get started.

4.2 Algorithm details

In this section, the optimization algorithm will be explained in further detail.

The Nelder Mead method

Nelder Mead (NM) optimization belongs to the subclass of optimization methods called direct search methods. The main theme among these is that they only use function values without creating approximations of the function gradients explicitly. These methods are especially useful if, for instance, the cost to evaluate the function is high and if it is impossible to derive the exact gradient. These statements apply to the optimization problem (2.1). Whenever the cost function is evaluated, the feasible proportional gains must be calculated and Simulink simulations run. The simulations are particularly costly if the given PID parameters, at a certain grid point, give a very sluggish closed loop.

The Nelder Mead method is a simplex-based method. There are many papers and books which describe in detail how the NM algorithm works (see for instance [Walters *et al.*, 1991] and [Lagarias *et al.*, 1998]), but quite few of them deals with the theoretic aspects. [Lagarias *et al.*, 1998] is an exception, in which a convergence proof for strictly convex functions, in $\mathcal{R}^2$, with bounded level sets is given. The NM method is commonly used in fields of chemistry and medicine. Two of the reasons why the method is popular are that it is easy to both understand and implement. The method is also fast compared to other algorithms and often has a great improvement of performance in the first few iterations. The Nelder Mead method can be applied to minimization problems in many dimensions. It is, however, only necessary to consider two dimensional NM optimization when designing PID controllers and one dimension for PI control. The reason being that when K is chosen separately to take care of the constraints, it becomes an unconstrained minimization problem in $\mathcal{R}^2$. Two dimensional NM optimization can be interpreted as triangle search progression with variable area.

In order to begin the NM optimization, an initial triangle has to be specified. The function to be minimized, f, is evaluated at all three edges and the points are sorted in the order:

1. B: Best, lowest function value $f(B)$
2. G: Good, function value, $f(G)$, in between the other two
3. W: Worst, highest function value $f(W)$

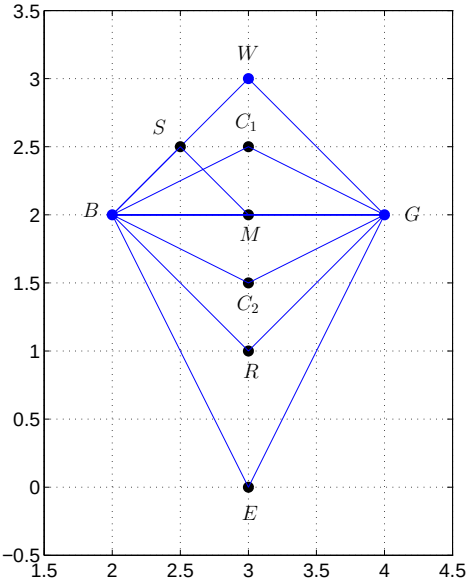

Figure 4.2 The Nelder Mead progression in one iteration. The initial simplex is the one with corners in B, G and W. The simplex will change its shape depending on function evaluations in closely situated points.

From this point, the algorithm will proceed along the following steps (see Figure 4.2):

1. Determine the midpoint M between B and G,

$$M = \frac{B + G}{2},$$

and reflect the point W through M to achieve R. R is determined by the formula

$$R = 2M - W.$$

If $f(B) < f(R) < f(G)$, the new triangle will be the one with edges in B, R and G and the algorithm starts over. If this condition is not fulfilled, continue to step 2.

2. If $f(R) < f(B)$ (if not, proceed to step 3), the chances are good that the optimization is proceeding in a beneficial direction. The algorithm will therefore investigate if it is worth expanding the simplex to the point
$$E = 2R - M.$$
If $f(E) < f(R)$, choose the new simplex with corners in B, G and E. Otherwise, replace W with R. Go back to step one with the new simplex.

3. If $f(R) > f(W)$ the simplex will be forced to shrink. Evaluate the function in the two points
$$C_1 = \frac{W + M}{2},$$
$$C_2 = \frac{R + M}{2}.$$
If any of these evaluations give a value less than $f(W)$, replace W with C_1 or C_2 depending on which gives the lowest value and go back to step 1 with the new simplex. If none of them is lower than $f(W)$, continue to step 4.

4. If all points given so far (R, E, C_1 and C_2) result in function values greater than $f(W)$, shrink the simplex towards B. The new simplex will have it's edges in B, M and S, where S is determined from the relation
$$S = \frac{B + W}{2}.$$

The proposed algorithm will iterate until the termination criterion

$$\begin{aligned}\frac{1}{3}(&\|(K^{(B)}, T_i^{(B)}, T_d^{(B)}) - (K^{(W)}, T_i^{(W)}, T_d^{(W)})\|_2 + \\ &\|(K^{(B)}, T_i^{(B)}, T_d^{(B)}) - (K^{(G)}, T_i^{(G)}, T_d^{(G)})\|_2 + \\ &\|(K^{(G)}, T_i^{(G)}, T_d^{(G)}) - (K^{(W)}, T_i^{(W)}, T_d^{(W)})\|_2) \leq \epsilon,\end{aligned}$$

has been fulfilled. The indices refer to the three simplex corners B, G and W. In this sense, it is possible to achieve a solution with a

somewhat prespecified accuracy in contrast to the gridding method which is based on luck and brute force rather than precision.

The regular Nelder Mead method is not suited for problems like (2.1) that only allow the simplexes to progress in $\mathcal{R}^+$. This has been solved in the proposed design method by granting the NM method access to negative T_i- and T_d-values. But the simulations will only run with the positive PID-parameters, taking the absolute values of the simplex corner points. For example, if one starts with the simplex having corners in $B = (T_i^{(1)}, T_d^{(1)}) = (1,1)$, $G = (T_i^{(2)}, T_d^{(2)}) = (5,1)$ and $W = (T_i^{(3)}, T_d^{(3)}) = (3,4)$. Using the original procedure of the NM algorithm, $R = (3,-2)$ would be the next grid-point to investigate. However, the changes made to the algorithm will instead alter this point to $(|3|, |-2|) = (3,2)$. The program could fail if this point would end up on the same line as B and G in which case the new simplex becomes a line. The NM method is always depending on the matrix containing the simplex points to have rank equal to the dimension of the optimization problem. This is always true when the simplex has an area greater than zero. The proposed software will therefore monitor the rank of the simplex matrix and give a warning if the rank is too low. It could also be a point in giving a warning if the matrix is poorly conditioned. It may then be a good idea to spread the simplex corners. Same could be applied if there are series of small simplexes for a very long time, for example typical for poorly damped systems with a badly chosen initial simplex. See figure 4.3 for an example of the behaviour, which is fortunately quite uncommon. None of these fixes have been implemented in the software though.

Initial values

As seen in Figure 4.3, it would be preferable to have a good initial guess of where the minimum is located for fast convergence of the NM optimization.

In the proposed PID design method, the system of interest is approximated as an FOTD system, (1.1), through a step response test after which the AMIGO PID parameters are determined. Let the index A denote these parameters. The AMIGO parameters will then be used as one of the corners, (T_d^A, T_i^A), in the initial Nelder Mead simplex. In [Walters *et al.*, 1991] it is recommended to start with a big

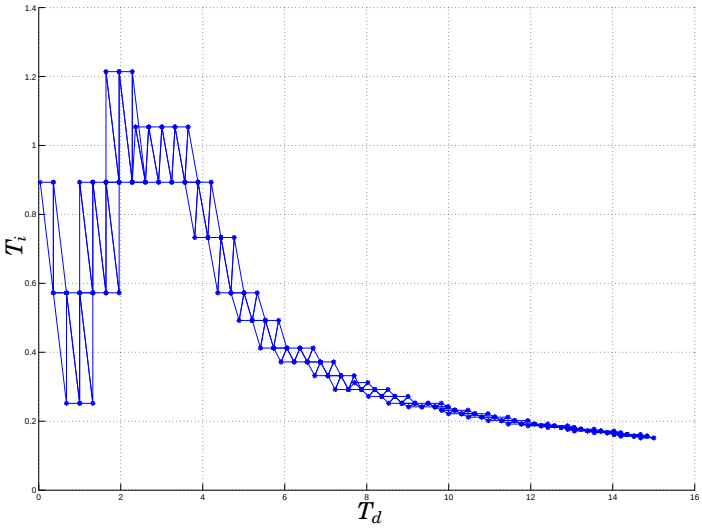

Figure 4.3 Nelder Mead progression for a highly oscillatory system. The shortcomings of the AMIGO method for this and some other types of processes could lead to a slowly converging Nelder Mead optimization.

initial simplex, which can then shrink to the area around the minimum, rather than starting with a small triangle which will have to expand before it can shrink again. Taking this into account as well as that the evaluation time is usually greater far out in the T_i-T_d plane, the other two corners have been set to $(0.4T_d^A, T_i^A)$ and $(T_d^A, 0.4T_i^A)$. In this way it is also avoided that the second simplex end up in any other quadrant than the first. Even though the modification of the Nelder Mead method can handle negative values on the PID parameters, it still seems wise to avoid these if possible since it alters the purpose of the original Nelder Mead method.

It is known that the AMIGO PID parameters are less suited for lag dominant systems ($\tau \approx 0$) and oscillatory systems (as Figure 4.3 shows). They have also been developed for the case when M_s and M_p are

both equal to 1.4. This means that the initial NM simplex can be quite a distance from the minimum in some cases and the method therefore needs to be quite robust to work properly. This said, a vast majority of the program runs end up in the global minimum within reasonable time. If not, then the optimization settings can just be modified until a satisfying result is given.

Determining the proportional gain K

As stated before, the proportional gain K - given fix values on T_i and T_d - is derived such that the open loop Nyquist curve is tangent to one or both robustness circles. Finding this gain can, however, be tricky. This is illustrated with a short example.

EXAMPLE 4.1—SEVERAL SOLUTIONS TO K
Consider the first order system

$$G(s) = \frac{1}{50s+1}e^{-s}.$$

The goal is to find a controller gain K, such that the PID controller with $T_i = 6.8$ and $T_d = 0.4$ puts the open loop Nyquist curve on the M_s-circle and/or the M_p-circle using $M_s = 1.4$, $M_p = 1.4$. Drawing M_s and M_p versus the proportional gain K, one receives the plot in Figure 4.4. It is apparent that there are several K-values fulfilling the criteria. In this particular case, M_s and M_p will both be less than 1.4 when $K = 0 \rightarrow 0.19$ as well as when $K = 8.09 \rightarrow 23.71$. Searching for the K that fulfills the constraints and gives the least IAE may therefore not be trivial. It could very well be that one ends up with $K = 0.19$ instead of $K = 23.71$ which gives the least IAE in this example. The algorithm in [Nordfeldt, 2005] for finding K has exactly this problem, while the new algorithm has been designed to take care of these cases as well. □

It was shown in Example 4.1 that finding the optimal proportional gain, K, is a non-convex problem. Therefore, a new algorithm has been developed in order to find all possible K-solutions for fix values on T_i and T_d. The key idea is to determine all K-values putting the open loop Nyquist curve on a circle in the complex plane, at every frequency point ω, resulting in a function $K(\omega)$. Since the method is numerical,

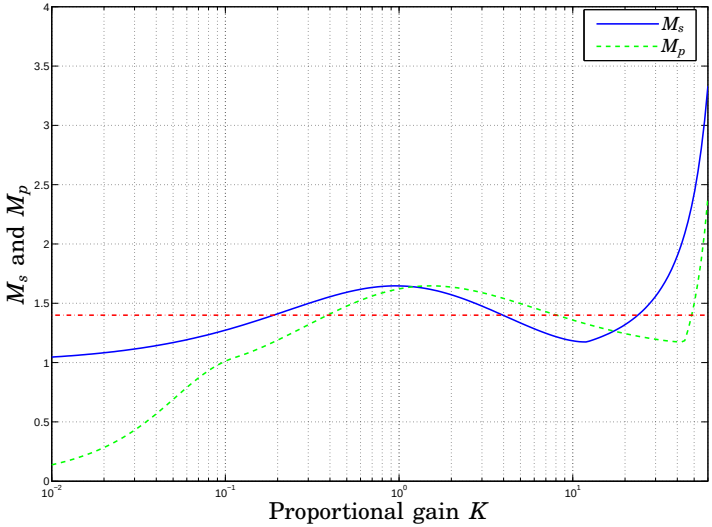

Figure 4.4 M_s and M_p as functions of the proportional gain K. The optimization constraints says that both should be below or equal to 1.4, but given that there are two separate intervals when this is feasible can make it challenging to find the K giving the least IAE.

the frequency span is divided into a finite number of points ω_k, $k = 1, 2, \ldots, N$. In order to determine $K(\omega)$, it will first be assumed that the open loop frequency response, $G_o(i\omega)$, can be written as

$$G_o(i\omega) = KG_o'(i\omega) = K(X(\omega) + iY(\omega)). \tag{4.1}$$

Where $X(\omega)$ and $Y(\omega)$ are the real and imaginary parts of $G_o'(i\omega)$ respectively. Circle constraints, like those in (2.1), can be written as

$$|G_o(i\omega) - C|^2 = R^2, \tag{4.2}$$

where C is the center of the circle with radius R. Using (4.1) and (4.2),

but changing K to $K(\omega)$, will lead to

$$\begin{aligned}&(K(\omega)X(\omega)-C)^2+(K(\omega)Y(\omega))^2=R^2\Rightarrow \\ &K(\omega)^2-\frac{2CX(\omega)}{X(\omega)^2+Y(\omega)^2}K(\omega)+\frac{C^2-R^2}{X(\omega)^2+Y(\omega)^2}=0.\end{aligned} \tag{4.3}$$

The two solutions correspond to the gains for which the open loop Nyquist curve crosses the front and back side of the circle respectively (see Figure 4.5)

$$K_{1,2}(\omega)=\frac{CX(\omega)\pm\sqrt{R^2(X(\omega)^2+Y(\omega)^2)-C^2Y(\omega)^2)}}{X(\omega)^2+Y(\omega)^2}. \tag{4.4}$$

$K_{1,2}(\omega)$ could for instance look like the function plots in Figure 4.6. For some frequency points, (4.4) will provide imaginary or negative numbers, which are discarded. In the intervals for which K assumes positive real values, there can be multiple minima and maxima.

There are a few observations needed in order to conclude which K-values will fulfill the constraints.

THEOREM 4.1
The open loop Nyquist curve (4.1) of an arbitrary controlled process will be tangent to a circle in the complex plane, defined by the center point C and radius R, if and only if

$$\frac{dK_1(\omega)}{d\omega}=0 \quad \text{or} \quad \frac{dK_2(\omega)}{d\omega}=0 \tag{4.5}$$

PROOF. In vector form, the open loop frequency response is

$$G_o(i\omega)=K\begin{pmatrix}X(\omega)\\ Y(\omega)\end{pmatrix}. \tag{4.6}$$

There are two conditions that has to be fulfilled in order for the open loop Nyquist curve to be tangent to the circle at a given frequency point ω^*. The open loop Nyquist curve should lie on the circle determined by

$$(KX(\omega^*)-C)^2+(KY(\omega^*))^2=R^2, \tag{4.7}$$

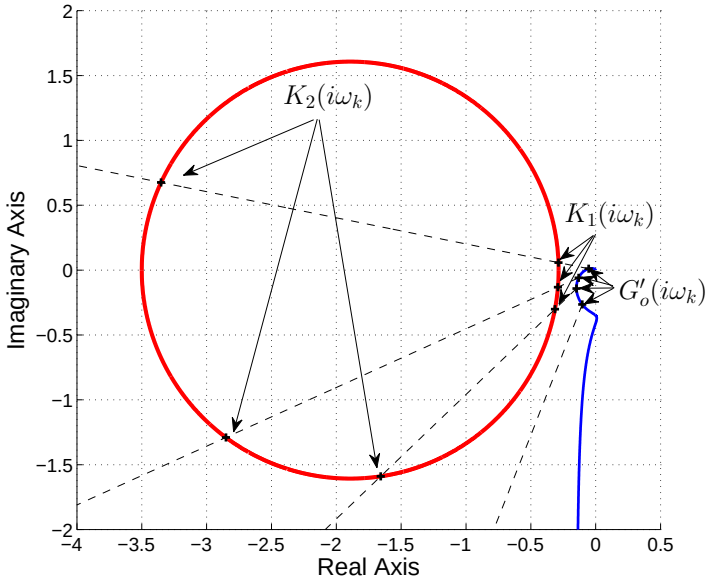

Figure 4.5 Proportional gain values $K_1(\omega_k)$, $K_2(\omega_k)$, for which $K_j G'_o(i\omega_k)$, $j \in [1,2]$, lies on a circle in the complex plane. $G'_o(i\omega_k)$ is the open loop frequency response with $K = 1$ and ω_k denotes the discrete frequency points.

while the tangent of the open loop Nyquist curve and the vector between the center point and Nyquist curve should be orthogonal

$$\left(\frac{dG_o(i\omega^*)}{d\omega}\right)^T \begin{pmatrix} KX(\omega^*) - C \\ KY(\omega^*) \end{pmatrix} = 0. \tag{4.8}$$

Denoting $X'(\omega) = dX(\omega)/d\omega$, $Y'(\omega) = dY(\omega)/d\omega$, (4.8) becomes

$$\begin{pmatrix} KX'(\omega^*) \\ KY'(\omega^*) \end{pmatrix}^T \begin{pmatrix} KX(\omega^*) - C \\ \\ KY(\omega^*) \end{pmatrix} =$$

$$K^2 X(\omega^*)X'(\omega^*) - KCX'(\omega^*) + K^2 Y(\omega^*)Y'(\omega^*) = 0,$$

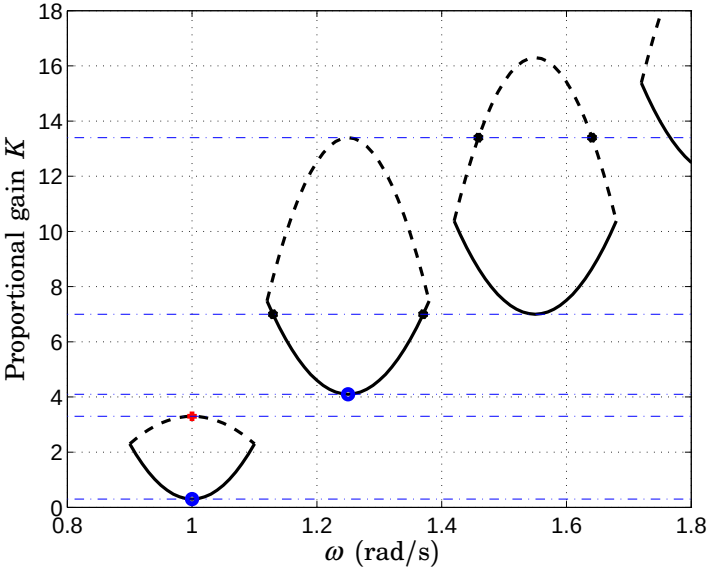

Figure 4.6 A sketch of how the functions $K_1(\omega)$ (solid) and $K_2(\omega)$ (dashed) could look for a time delayed system. Only K-values unique in ω - i.e. the first two minima and first maximum in this case - will fulfill the circle constraints.

and in turn, by solving for K, one ends up with

$$K = \frac{CX'(\omega^*)}{X(\omega^*)X'(\omega^*) + Y(\omega^*)Y'(\omega^*)}. \tag{4.9}$$

Consider equation (4.3) once again. Taking the derivative with respect to ω, given that $K'(\omega) = dK(\omega)/d\omega$, leads to

$$\begin{aligned} 2KK'X^2 + 2K^2XX' - 2CK'X - 2CKX' + \\ 2KK'Y^2 + 2K^2YY' = 0, \end{aligned} \tag{4.10}$$

where ω has been omitted. Using $K'(\omega) = 0$, results in

$$K(\omega)X(\omega)X'(\omega) - CX'(\omega) + K(\omega)Y(\omega)Y'(\omega) = 0 \Rightarrow$$
$$K(\omega) = \frac{CX'(\omega)}{X(\omega)X'(\omega) + Y(\omega)Y'(\omega)}, \tag{4.11}$$

which is identical to (4.9) in ω^*. Since (4.7) is fulfilled for all frequencies in $K(\omega)$, the proof is concluded.

□

$G_o(i\omega)$ can, however, be tangent to the circle but still have points within (thus giving an infeasible solution). To explain why, it is a good idea to once again view Figure 4.5. At a given frequency point ω_k, it is obvious that all proportional gains between $K_1(\omega_k)$ and $K_2(\omega_k)$ will place $G_o(i\omega)$ inside the circle, thus resulting in infeasible K. Looking at Figure 4.6, this means that only the minima and maxima, for which K is unique in ω, are feasible. For this particular case, it corresponds to the first two minima and first maximum.

Once all possible K-values have been determined (the two minima and the maximum from Figure 4.6 for example), closed loop stability is evaluated. If there is stability, the optimal proportional gain at a given point in the T_i-T_d plane is then given by the K resulting in the lowest IAE-value (determined through Simulink simulations).

Up to now it has been assumed that it is just one circle in the complex plane that the open loop Nyquist curve may be tangent to. Since the constraints of (2.1) demands that the Nyquist curve is located outside both the M_s- and M_p-circles, the algorithm has to be run twice.

It is important that the frequency span is sufficiently wide and dense. If not, it could lead to erroneous results. It may therefore be needed to change the frequency settings in the software menu. It is also wise to have a brief check of the open loop Nyquist plot from the program to see that the curve looks alright.

Finally, a brief summary of the algorithm finding the optimal proportional gain:

1. Determine the functions $K_1(\omega)$ and $K_2(\omega)$, giving the proportional gains that will put each frequency point ω_k on both the M_s- and the M_p-circle.

2. Remove all imaginary and negative numbers from $K_1(\omega)$ and $K_2(\omega)$.
3. Determine the minimum and maximum of each frequency interval.
4. Pick out the K-values that are not represented in any other interval.
5. Examine closed loop stability for all K-values.
6. Make sure Nyquist curves touching the M_s-circle do not lie within the M_p-circle and vice versa.
7. Simulate the closed loop system for all feasible K-values and pick the one resulting in the lowest IAE-value.

A note on choosing K The optimization method is based on choosing the proportional gain such that the open loop Nyquist curve is tangent to one or both robustness circles. This has the major advantage that one can split the optimization problem into two parts; choosing K and minimizing the cost function in the T_i-T_d plane (which generally only gives one minimum in total). While this proportional gain selection is generally a clever solution, when minimizing the IAE, it may not always be optimal. One way to see this is to set $M_s = M_p = M$ and vary M over a span of different values. Figure 4.7 shows how the IAE depends on M for the process

$$P(s) = \frac{e^{-s}}{(s+1)^2}.$$

At M-values around 3.4, the IAE starts to increase again, suggesting that a smaller value on K is in fact better. The reason for this behaviour is likely the decrease in robustness giving a less damped step response which ultimately leads to worse performance too. While this might be disturbing results, it is mainly a problem for higher values on M. Such values are not likely to be used in the end, especially not in industry. Also, the increase in IAE often seem to be quite minor, like in the plot, suggesting that this is hardly an issue when designing controllers.

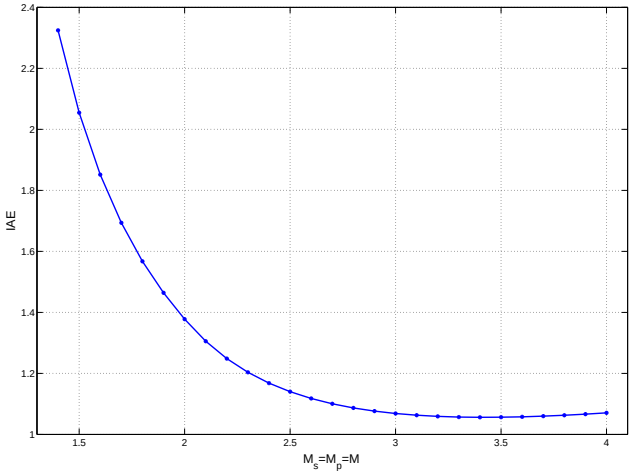

Figure 4.7 Plotting IAE as a function of $M_s = M_p = M$, for $P(s) = e^{-s}(s+1)^{-2}$, shows that it may not always be optimal to choose K such that one or both constraints are active.

4.3 PI control

There are cases for which the D-part in a PID controller may not be necessary to use. In order to gain deeper insight into when this occurs, a one-dimensional Nelder Mead algorithm was implemented in the software program to derive optimal PI controllers. This controller has the following transfer function

$$C(s) = K\left(1 + \frac{1}{sT_i}\right) \cdot \frac{1}{sT_f + 1}.$$

Note that the second order low-pass filter has been replaced by one of first order. This will keep the same relative degree in the open loop transfer function as for the PID design and guarantee roll-off without introducing unnecessary phase-lag.

The initial controller is given with AMIGO, just as in the PID case.

The difference being that the PI formula

$$T_i^{(A)} = 0.35L + \frac{13LT^2}{7L^2 + 12LT + T^2}$$

is used instead of the PID counterpart. The other initial T_i is set to $0.7T_i^{(A)}$ to not risk having a negative T_i within the next iteration. If one, after all, ends up with negative T_i-values, these will be mirrored (using $|T_i|$) into the positive value as in the PID case. From there, the Nelder Mead method progresses as follows:

1. Calculate the IAE of the two initial controllers.

2. Set B and W.

3. Set the midpoint to the same point as B and calculate the IAE for $R = 2M–W$, $f(R)$.

4. If $f(R) < f(B)$, determine $f(E)$ when $E = 2R–M$. Replace W with R if $f(R) < f(E)$, else with E and go back to step 1.

5. If $f(R) \geq f(B)$, compute the IAE for $C_1 = (W + M)/2$ and $C_2 = (M + R)/2$. Replace W with whichever of these points gives the lowest IAE and go back to step 2.

This algorithm is quite similar to the PID counterpart.

As an example, consider PI control of a third order lag process

$$G(s) = \frac{1}{(s+1)^3}.$$

Selecting default optimization ($M_s = M_p = 1.4$, $T_f = 0.001$), the Nelder Mead algorithm found the PI controller with $K = 0.63$, $T_i = 1.95$ and IAE $= 3.08$. This can be compared to the PID controller for the same problem with $K = 3.81$, $T_i = 1.14$, $T_d = 1.12$ and IAE $= 0.53$. The NM progression for the PI optimization can be seen in Figure 4.8 together with the result from gridding the cost function.

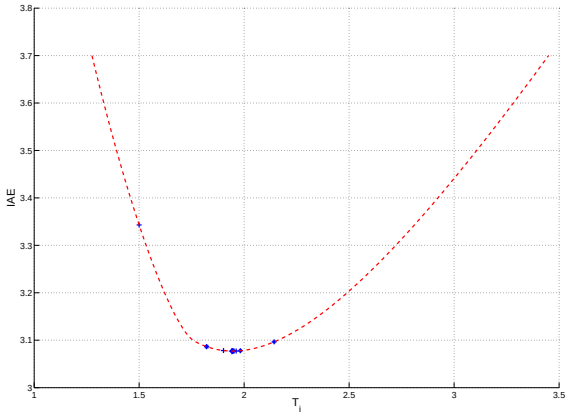

Figure 4.8 The Nelder Mead optimization progression for PI control of $G(s) = (s+1)^{-3}$ is shown as stars. The dashed line shows the cost function derived through gridding.

4.4 Examples

In this section there will be a couple of examples highlighting the benefits of the proposed design software and algorithm compared to other methods. It will also show that the new method is reliable in many design cases.

EXAMPLE 4.2—THE AMIGO TEST BATCH

In order to compare the new PID design method with the MIGO PID designs on the AMIGO test batch, (3.1), the proposed software was modified to use the M-circle as constraint. It took just more than one hour to run through all sub-batches except the integrating processes in sub-batch 6. This gives an average time of 30 seconds per process. The batch was, however, run to get a high accuracy on the optimal solution rather than optimized for fast designs. If speed is of essence, the average design time per process could be cut considerably. The designs were run on an Intel® Dual-Core™, 2.13 GHz with 1 GB RAM

and Fedora 7 using Matlab® 7 R2007a. The only two parameters that had to be modified from the default values depending on the process, in order for the batch to run through properly, were T_f and the frequency grid.

The PID parameters derived by the proposed algorithm were compared to those given by the MIGO method. Since the MIGO method do not minimize IAE, the proposed method should always give lower values. A comparison with the MIGO method does, however, still give a good indication on whether or not the new algorithm works properly. The batch run showed that the two methods are very similar. In average, the new controllers resulted in IAE-values at 95% of what the MIGO controllers gave over the whole batch. This gives both a strong indicator that the new program works properly and that the MIGO method gives essentially IAE minimized controllers for the batch.

To see the benefits of using the M_s- and M_p-circles instead of the M-circle, the whole batch was compared when the two different constraints were used respectively. Figure 4.9 shows that the biggest percentual gain is given for low values on the normalized time delay, τ, while more delay dominated systems depend less on the choice of the constraints. This indicates that the performance for lag dominant processes is very sensitive to changes in the constraints.

The sub-batch where the proposed design software gave IAE values with the biggest difference from the MIGO ones was

$$P(s) = \frac{1}{(s+1)((sT)^2 + 1.4sT + 1)}, \qquad (4.12)$$
$$\text{with} \quad T = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0,$$

i.e. processes with complex poles. In particular when $T = 0.1$ the new IAE is as small as 62.5% of the MIGO IAE, corresponding to a significant improvement. Figure 4.10 shows the output signal, y, and control signal, u, when a load disturbance, d, is acting on the process input. The dashed curves correspond to the MIGO controller ($K = 3.96$, $T_i = 0.46$, $T_d = 0.08$), the solid lines to the proposed controller with the M-circle constraint ($K = 5.42$, $T_i = 0.29$, $T_d = 0.16$) and the dash-dotted line to the new design method with the M_s- and M_p-constraints ($K = 6.53$, $T_i = 0.22$, $T_d = 0.16$). The open loop Nyquist curves for the three cases are shown in Figure 4.11. It is known that the MIGO

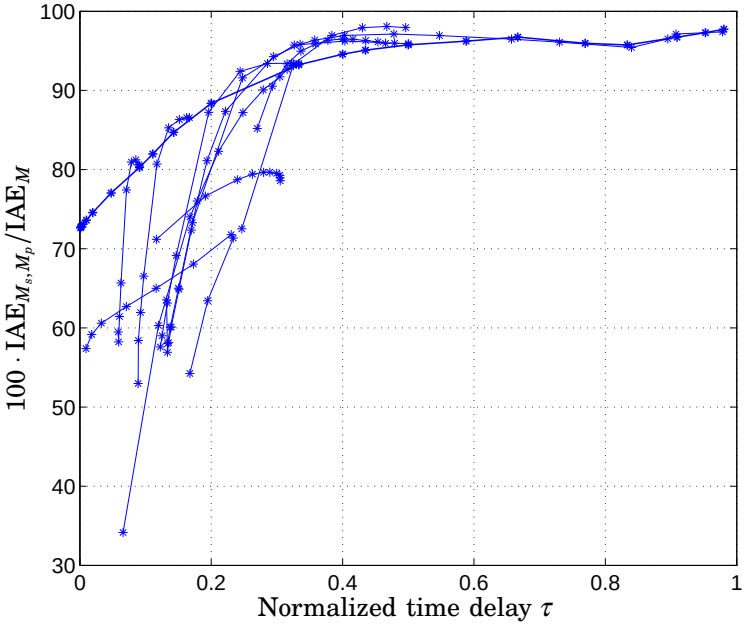

Figure 4.9 The IAE-values for the test batch was compared using either the M-circle constraint alone or both the M_s- and M_p-circle. The plot displays $100 \cdot \mathrm{IAE}_{M_s,M_p}/\mathrm{IAE}_M$ as a function of the normalized time delay τ.

method discards solutions that touches the M-circle twice. This example shows that this choice may be overly conservative. It is also evident that the substitution of the M-circle to the M_s- and M_p-circles gives a much lower IAE-value in this case. □

The previous example showed that the program works well and that it has the potential to give birth to new findings on PID control. It is also a lot easier to start using than the MIGO method, which relies more on the user.

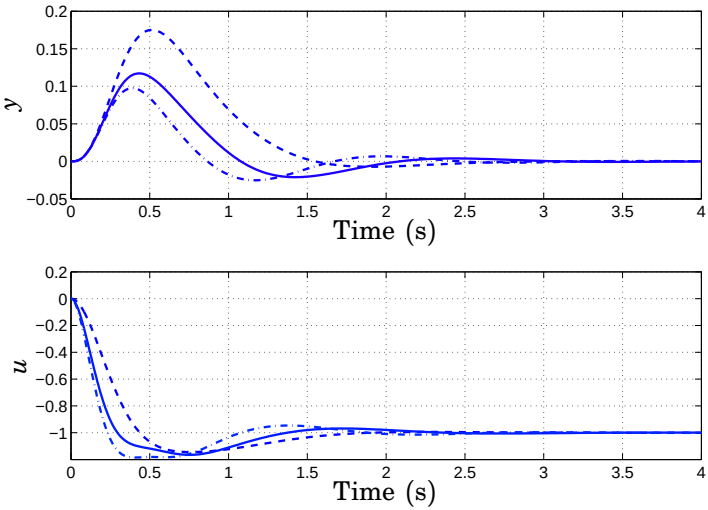

Figure 4.10 Output (y) and control signal (u), during a load disturbance, for three different designs on (4.12), $T = 0.1$. Dashed line: MIGO PID; Solid line: Proposed PID with M-circle constraint; Dash-dotted line: Proposed PID with the M_s- and M_p-circle constraints.

EXAMPLE 4.3—AN OSCILLATORY PROCESS

Consider an oscillatory system with the linear transfer function

$$P(s) = \frac{9}{(s^2 + s + 9)(s + 1)}, \tag{4.13}$$

which has two complex poles with relative damping $\zeta = 1/6$. An IE-cost function is not suitable for PID design when the system is oscillatory, ruling out use of the MIGO method. The proposed design algorithm, however, can derive a PID design without problems. For $M_s = M_p = 1.4$ the program gave the parameters: $K = 0.37$, $T_i = 0.23$, $T_d = 0.80$. Figure 4.12 shows the control- and output signals. The IAE-value for the design is 0.94.

□

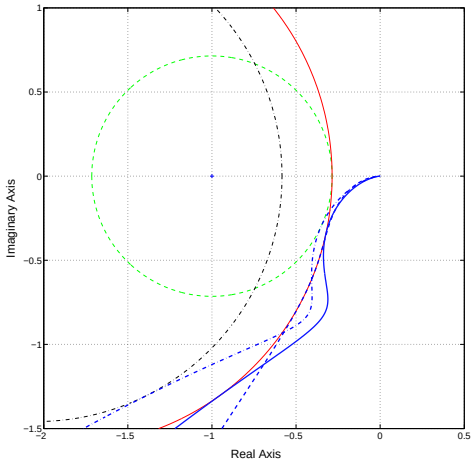

Figure 4.11 The open loop Nyquist curves when three different design methods were used on (4.12), $T = 0.1$. Dashed line: MIGO control; Solid line: Proposed design with M-circle constraint; Dash-dotted line: Proposed design with the M_s- and M_p-circle constraints.

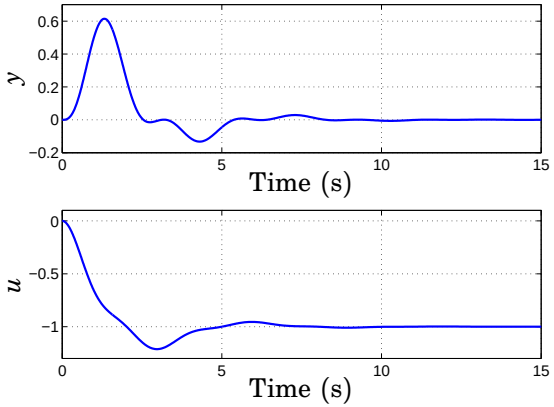

Figure 4.12 Output signal, y, and control signal, u, when the proposed design method was used to find a controller for the oscillatory process (4.13). $M_s = M_p = 1.4$.

5

Adjustable Control Signal Noise Reduction

Methods like MIGO/AMIGO and lambda tuning have the weakness that they have no systematic way of designing the low-pass filter acting on the measurement signal or the D-part. The same goes for the approach described in Chapter 4 where the effect of the low-pass filter was completely ignored. Designing PID controllers without roll-off can give severe throughput of measurement noise in the control signal for a real plant. This chapter will, therefore, introduce a new way of tuning the low-pass filter such that the control signal variance, due to measurement noise, is constrained. Or in other words, choosing T_f such that

$$\|S_k\|_2^2 = \left\| \frac{C}{1+PC} \right\|_2^2 = \frac{\sigma_u^2}{\sigma_n^2} \leq V_k. \tag{5.1}$$

S_k is the transfer function from measurement noise n to control signal u, σ_u^2 is the variance of the control signal and σ_n^2 is the variance of the measurement noise. V_k is a design variable that is user specified.

This chapter will begin with a description of the principles of the new design method. There are also several new challenges associated with the proposed method, which will be presented in a section of its own. Finally, the chapter will be concluded by a suggestion of design algorithm together with several examples on the use of the method. But, first of all, a short note on the software tool.

Designing discrete time controllers for real plants

Even though controllers have to be implemented digitally, most PID design methods are based on continuous time analysis only. The PID parameters are then used directly in a discrete time approximation of the controller when implemented. Such an approximation will, however, introduce phase lag (see e.g. [Åström and Wittenmark, 1997]) and is thus likely to make the closed loop system less robust than intended. This problem will be especially severe if the sampling time assigned to the process has been chosen too long by mistake. This could very well be the case in industry, which will be shown in Chapter 6. For these reasons, a discrete time version of the design software has been developed in parallel with the continuous time version. While the previous chapter was all about the continuous time version, the discrete time software will mainly be used in this chapter.

The discrete time controllers have been approximated with forward Euler on the I-part, backward Euler on the D-part - according to [Årzén, 1996] - and Tustin's approximation on the low-pass filter such that the final PID controller becomes

$$C(z) = K\left(1 + \frac{1}{T_i\frac{z-1}{h}} + T_d\frac{z-1}{zh}\right)\cdot\frac{1}{\left(1+\frac{2(z-1)}{h(z+1)}T_f + \left(\frac{2(z-1)}{h(z+1)}T_f\right)^2/2\right)}.$$

The PI controller is very similar.

The discrete time version of the software is not yet released. Anyone can, however, take the program and redo it in the same fashion. Few changes are needed from the original version.

5.1 Principle

The idea for finding a controller that fulfills constraint (5.1) will be illustrated through an example.

EXAMPLE 5.1—THE RELATION BETWEEN T_f AND V_k
Determining the variance $||S_k||_2^2$ for a lot of different T_f-values (using

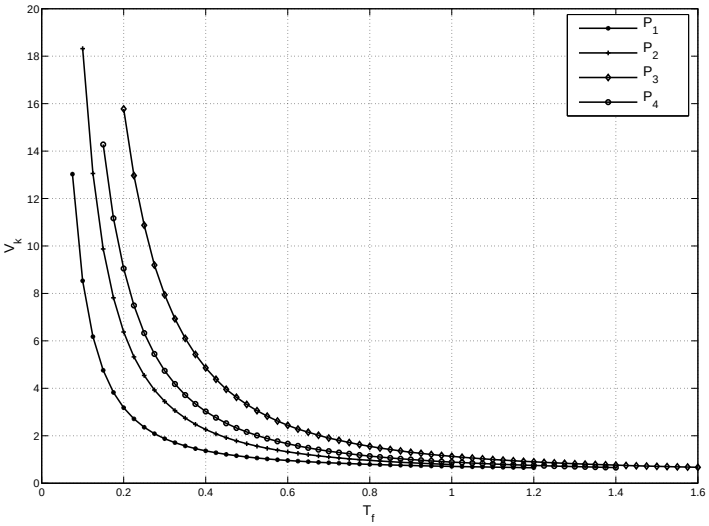

Figure 5.1 Different processes give similar dependencies between T_f and the control signal variance.

the proposed Matlab software) on the systems

$$P_1(s) = \frac{1}{s+1}\,e^{-s}, \quad P_2(s) = \frac{1}{(s+1)^2}\,e^{-s},$$
$$P_3(s) = \frac{1}{(s+1)^4}, \quad P_4(s) = \frac{(-0.5s+1)}{(s+1)^3},$$

have shown that there are very similar dependencies between T_f and V_k in all four cases, see Figure 5.1. Having relations like these makes it possible to use T_f as a design variable to fulfill (5.1). □

Finding the T_f that gives a certain V_k can be done using different search methods. Another approach, and the one that will ultimately be used in this thesis, is to determine the relation between performance (IAE) and noise amplification (V_k). That way, the control designer can

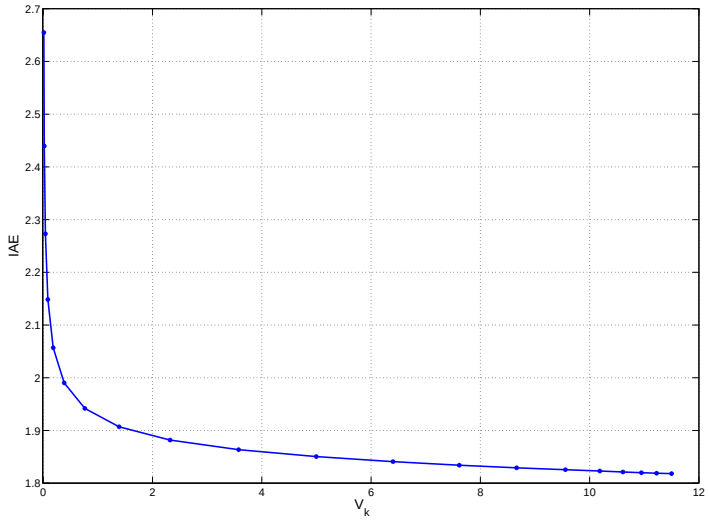

Figure 5.2 Trade-off curve between noise amplification and performance for the process P_1, in Example 5.1. Note that the noise characteristics used here was not the same as in Figure 5.1, which explains the difference in V_k-values.

take the trade-off between noise throughput and performance into account when choosing a suitable controller. Figure 5.2 shows an example of this relation for process P_1.

There are also some other interesting effects one can analyse after adding the new constraint into the picture. Like for instance:

- For which types of systems will the D-part be most important?
- When will the low-pass filter make the least difference?

Here is one example showing some interesting effects.

Table 5.1 The effect of the low-pass filter time constant on the control of first order systems with time constant T and delay $L = 1$. V_k was set to 1.

T	h (s)	K	T_i	T_d	T_f	IAE_{inc} (%)
0.1	0.02	0.21	0.42	0.14	0.014	0.7
1	0.05	0.59	1.04	0.44	0.13	12.6
50	1.5	2.44	21.8	11.4	12.4	1647

EXAMPLE 5.2—FOTD SYSTEMS WITH VARIANCE CONSTRAINT
Take three FOTD systems

$$P(s) = \frac{1}{Ts+1}e^{-s}, \quad T = 0.1, 1, 50,$$

which have $\tau = 0.91$, 0.5 and 0.02 respectively. Assume furthermore that the measurement noise in all three cases is white, unit variance, noise (each sampled with different sampling time h). As will be explained in Section 5.2, it is not trivial to compare different systems, so these three cases will be treated without comparison.

V_k will be set to 1 when designing controllers for all three processes, such that the control signal get the same variance as the white measurement noise. Table 5.1 shows the results when designing PID controllers as described in the previous section. The table displays sampling time, PID parameters, low-pass filter time constant as well as IAE increase compared to an unfiltered controller. Figure 5.3 shows the Bode plots of all three PID controllers.

Analysing the results, one can see that the most lag dominant system ($T = 50$) has the biggest increase in IAE compared to when the controller does not include a measurement filter. This relates to what examples from the previous chapter said about lag dominant systems being the most sensitive to changes in the optimization problem. It seems like the same holds for when the variance is limited. Several similar runs have shown that this example is in no way particular in this aspect. The performance of lag dominant systems are generally sensitive to changes, which also makes them quite hard to analyse. Delay dominant systems, on the other hand, are barely affected at all

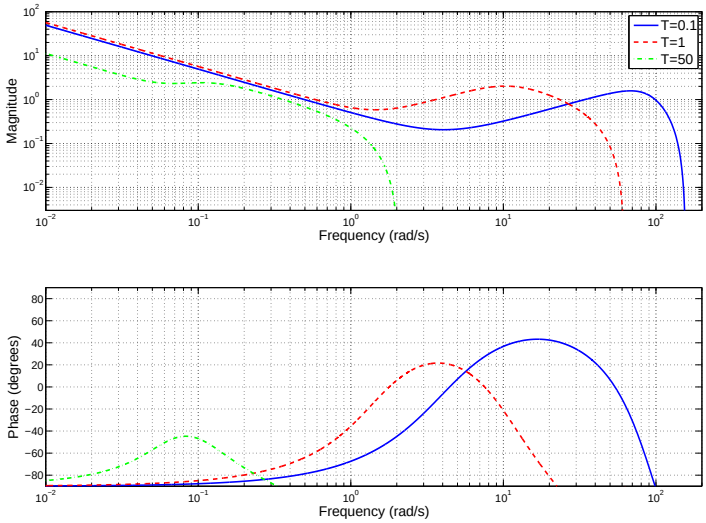

Figure 5.3 Bode diagrams of the PID controllers for the three FOTD systems in Example 5.2.

by the low-pass filter, which would pretty much mean that one can add the filter after the controller design if one would want to.

Another very interesting feature, is that the PID controllers for lag dominant systems are more similar to PI- or even I-controllers. In this example, it can be seen both in the Bode plots and by comparing T_d and T_f which are very close. In other words, the low-pass filter rolls off close to the frequency where the D-part starts to increase the gain. □

Motivation of the constraint and options

It has already been mentioned that one of the main reasons for not including the D-part in industry is because it makes the control signal sensitive to measurement noise. Highly fluctuating control signals are likely leading to wear and tear on actuators, which are typically very expensive to exchange. This is the main reason for including the

variance constraint in this chapter. Such a constraint will capture the variation of the control signal and make the designer able to use the D-part without fear of getting too much noise throughput.

One disadvantage with using the variance constraint as a measure is that it does not weigh in for which frequencies the sensitivity is high. A low frequency sine wave can for instance have the same variance as a swiftly shifting noise signal. It does, however, seem more likely that high frequency components in the noise will wear on actuators rather than the low frequency part. For that reason it is a good idea to weigh S_k with respect to frequency. In this thesis this will be done by high-pass filtering measurement data to try to single out the measurement noise (see Section 5.2). Whether or not this is enough to get the desired characteristics for the closed loop system will be left for future work to show.

Another option for a constraint is what was used by for instance [Kristiansson and Lennartson, 2006], namely $\|S_k\|_\infty$. This gives a limit on how much any frequency can be amplified. The measure is quite straight forward, but it would ideally need weighting depending on frequency, just like the variance constraint. There is little point in lowering the peak of S_k for low frequencies.

Instead of the variance, one could just as well have constrained the standard deviation or the quantiles of the signal. Both measures are related to the variance. The standard deviation has a more direct relation to the magnitude of the control signal than the variance. Using quantiles would mean that one can specify how large quantity of the control signal would lie outside a certain limit. This would possibly be a more easily explainable measure to people in industry. Nevertheless, the choice of constraint fell on variance for this thesis. It can, however, easily be translated into these other two quantities.

Why is the method not optimal?

It is important to realize that fulfilling constraint (5.1), choosing a T_f giving a certain V_k, will not give controllers that are optimal with respect to the constraint. The software calculates controllers that are optimal with respect to the two robustness constraints, given a certain T_f. It could, however, very well be that a different T_f could give even lower IAE, using other controller parameters that alters $\|S_k\|_2^2$. One example could be a controller derived with a certain T_f that results in

too high variance. A little less aggressive controller, but with the same T_f, may still give less IAE than the controller which the proposed design method chooses.

In Section 4.3 it was shown that unless the control signal variance constraint is taken into account, the PID controller can be quite superior to the PI controller regarding IAE. The D-part in a PID controller will, however, also give a significant raise to the control signal variance due to measurement noise. Therefore, it is not so straightforward to assume that a controller always benefits from having a D-part. It is rather a combination of process characteristics and the amount of control signal variance allowed that will determine the usefulness of the D-part.

Depending on the sampling time, it may not be possible for the control signal variance to reach the V_k that has been set (see Figure 5.4). The reason being that the low-pass filter cut-off frequency would have to be so far beyond the Nyquist frequency to give V_k that it just does not affect the variance enough. This is not necessarily negative, but shows that one could perhaps have used a different controller for which V_k is higher and the IAE is lower.

The fact that the PI controller may give a control signal variance lower than V_k makes it a bit difficult to compare with the PID controller. But, it is logical to let the PID controller take whatever T_f gives V_k, even if the PI controller gives $\|S_k\|_2^2 < V_k$, when comparing. After all, it is the user that sets the rules and even if both controllers can not reach the limit, they are both playing by the rules.

Assume that one wants to examine the behaviour of the following two systems

$$P_1(s) = \frac{e^{-s}}{(10s+1)^2}, \quad P_2(s) = \frac{e^{-s}}{(50s+1)},$$

with respect to control signal variance, using both PI and PID control. The sampling time has been chosen to $h = 0.01$ seconds with $M_s = M_p = 1.4$ in both cases. In Figure 5.5, the IAE-values are plotted versus $\|S_k\|_2^2$ for a number of different T_f-values. The curves reveal that the PI controller can actually have lower IAE value than the PID controller when the control signal variance, $\|S_k\|_2^2$, is the same. For $P_1(s)$, this occurs when $V_k \approx 0.002$ and for $P_2(s)$ when $V_k \approx 1.75$. These results

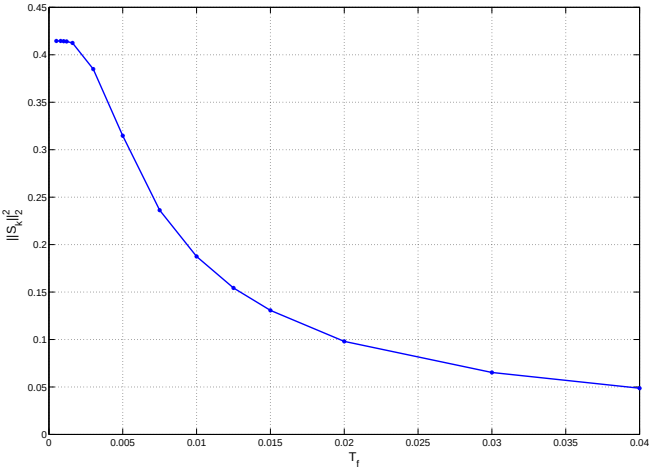

Figure 5.4 PI control of $G(s) = (s+1)^{-3}$ with different choices of T_f is not necessarily able reach $\|S_k\|_2^2 = V_k$. In this case for instance, $\|S_k\|_2^2$ does not take larger values than 0.42. The measurement noise is here assumed to be white, unit variance, sampled with $h = 0.01$s.

lead to two conclusions: The PID controller is not optimal for the new optimization problem and the closed loop system around P_2 seems to benefit quite a lot less from the D-part than the P_1 loop does.

Now that it is known that the selected controllers are not optimal, one could ask a few justified questions: When would it be smart to switch to PI control? Are the controllers one gets through the proposed method still good enough to use? How far from optimality will the PID (or PI) controllers be? These are questions that to some extent will have answers at the end of this thesis. In this section, there will be an attempt to answer the first question.

Analysis of the controllers for P_1 and P_2 shows that there seems to be at least one measure that gives an indication of when it is a good idea to check how well a PI controller would work. That measure is the relation between T_f and T_d of the final PID controller. Figure 5.6 shows T_f/T_d plotted versus T_f for $P_2(s)$. At $T_f \approx 0.8$, this quotient comes very close to 1, meaning that the final PID controller is pretty much a PI controller. When comparing this T_f value to the ones in Figure 5.5 for

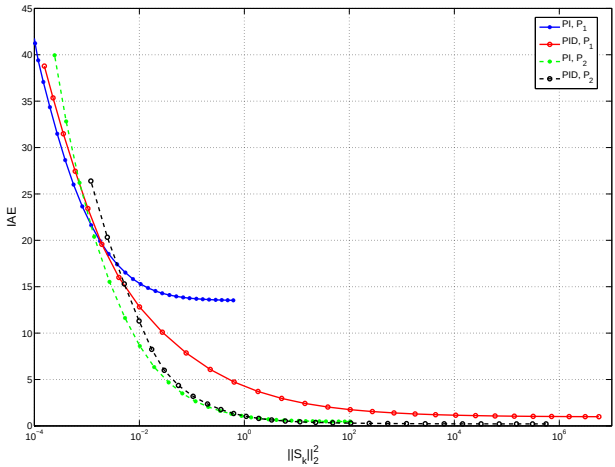

Figure 5.5 Processes P_1 and P_2 were run with several different T_f-values using $h = 0.01$s. This gave the following relation between IAE and control signal variance, showing that the PID controllers are not necessarily optimal.

when the PI is better than PID, it shows that for $T_f \approx 1.15$ and above, the PI controllers are better. This is rather close to 0.8. Experiments on several other processes in the AMIGO batch have shown that this is a repeating behaviour for the cases when PI is preferable. Therefore, it seems like a good idea to first run the PID design and then, if $T_f/T_d \approx 1$, also run the PI algorithm to see if that gives an even better result. When thought through, it seems quite logical that as V_k goes to zero, the controller will be given less freedom to increase the gain and phase, thus becoming more and more like a PI controller or even a pure I controller for that matter.

5.2 Challenges with the use of a variance constraint

Using variance as a measure of when the control signal is sensitive to measurement noise is in many ways a simple and intuitive. But there also comes quite a few challenges together with using the constraint. Some of these will be presented in this section.

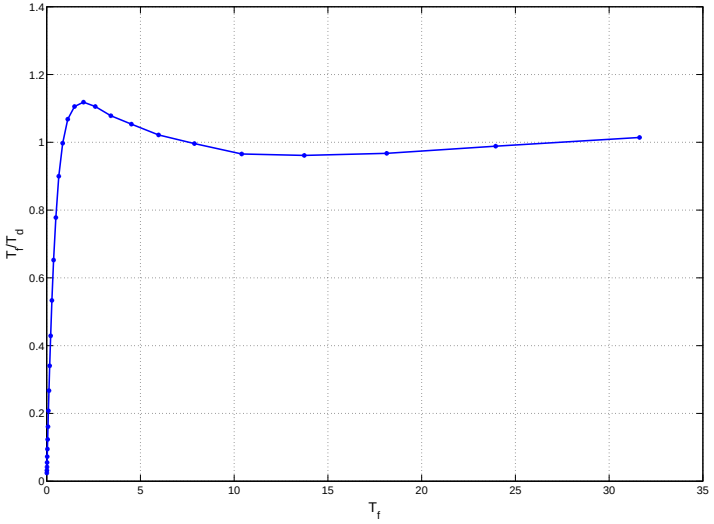

Figure 5.6 Designing several PID controllers with different T_f for process $P_2(s)$ shows that these are almost PI controllers when T_f reaches a certain value (and $\|S_k\|_2^2$ decreases). Checking if $T_f/T_d \approx 1$ can then be a measure to see when it is a point in switching to PI control.

How to calculate the variance?

In order to get a good estimate of the real variance relation σ_u^2/σ_n^2 one will first need a representative selection of noise data. One possibility is to run the process in open loop, log the measurement data and then filter out what is assumed to be the noise. Logging the data could be done either by having the system run in manual with a constant control signal or in connection to a step response test on the process. The latter option seems more attractive since these kinds of tests are often run on a process to determine an FOTD model just before the controller design. It would thus be a very natural way to incorporate noise logging in the normal procedures of controller design. Another option would be to get the data either through relay tests, which are common in auto-tuning, or through closed loop runs. If it is possible to derive good noise data like this it would be advantageous, since the

process should ideally be in closed loop all the time. In this thesis, however, it will be assumed that the measurement data is collected in open loop.

Now assume that one can collect a sequence of representative noise data. How long does this log need to be in terms of time in order to provide a good sample for estimation of the variances? And what method should be used to calculate σ_n^2 and σ_u^2? There will not be any real analysis in this thesis trying to answer the first question. But it seems reasonable to believe that, if the step response is used to collect noise data, it will depend on how fast the process is. If the process is fast, one may need to hold the process a little longer in open loop to gather some more noise data. Since one may get very long measurement logs for a process with slower dynamics, it is more likely that the length of the noise data will be sufficient in this case. If the noise data is extensive enough, it should be fine to calculate the noise variance simply by using the formula

$$\sigma_n^2 = \frac{1}{N}\sum_{i=1}^{N}(n_i - \bar{n})^2,$$

where N is the number of noise data, n, and $\bar{n}$ is the mean value of this data. Another option is to determine a model, N, of the noise such that

$$n(k) = N(q)e(k)$$

where e is white noise. This model can then be used to estimate σ_u^2 as well. This approach will be used here when Youla parametrized controllers are compared to PI/PID controllers. When running the PI/PID design on a real process, however, the actual noise data will be used as part of a Simulink simulation to estimate σ_u^2. The advantage with the model approach is that it allows for better analysis, but it is bad in practise since one needs to automatize the model making which is not trivial. Using the real data is a much easier method. On a whole, it should be part of future research to look more at how to get representative information of the measurement noise.

How to disregard other sources when gathering noise data?

Assume that one wants to extract the measurement noise out of an open loop run on the system. What part of the signal should be kept

and which can be thrown away? [Marlin, 1995] points out that the variations showing in the measurement data are a result of several effects. Some of them come from the process itself, some from the sensors, while others are due to transmission. All these effects contribute to the signal over a wide range of frequencies. The idea of the proposed method is to be able to set the control signal variance such that the wear on actuators, like valves, can be kept low. Generally speaking, it is, large, high frequency movements in the control signal that leads to this wear. Therefore, the idea is to weigh the measurement data by sending it through a high-pass filter, $W(s)$. The low-frequency part of the data is thus assumed to have small contribution to actuator wear. The high-pass filter

$$W(s) = \frac{s^2}{s^2 + 2\zeta\omega_o s + \omega_o^2}, \tag{5.2}$$

was chosen for this reason. It is discretized, with the current sampling time of the process, before the measurement data is run through it. The damping coefficient ζ was set to 0.7 to receive a filter that is steep, but still does not give much overshoot in a Bode magnitude diagram. Since the filtering is handled offline, one can use non-causal filters that are almost ideal high-pass filters. But, since the hardest question is how to choose ω_o properly, it seems like a good idea to keep some of the lower frequencies in the filtered data as well.

The noise data can roughly be divided into controllable disturbances, uncontrollable disturbances and measurement noise. These three sources will to some degree overlap each other, but it seems reasonable to assume that the measurement noise throughput will mainly wear on actuators for frequencies above the cut-off frequency of the closed loop. In many industrial systems, the closed loop is rarely faster than the process itself. For that reason it is a possibility to choose ω_o to depend on the process model. If the process model is an FOTD system, ω_o could for instance have a default value of $3/T$, where T is the time constant of the process. The user should, however, be ready to change this value from case to case.

Sampling time issues

When making discrete time controllers, the sampling rate will be different depending on the process. Too slow sampling could introduce

unnecessary phase lag, thus decreasing the best possible performance significantly. Really fast sampling may on the other hand not be possible, due to for instance resource limitations. So, generally speaking, one should strive to choose sampling time from case to case. This does, however, also give some additional effects in connection to the variance constraint that are important to be aware of if one wants to compare noise effects on different processes.

Assume that one has a certain sensitivity function $S_k(s)$ for some closed loop system. This system is sampled with the two different sampling times h_1 and h_2. The variance relation between the continuous time system and the sampled time systems is

$$h_i \|S_{k_i}(s)\|_2^2 = \|S_{k_i}(z)\|_2^2,$$

when zero order hold sampling is used. The index i refers to the different sampling rates. The control signal variance should, however, be almost the same for both sampling cases, as long as the sampling rate is chosen properly to avoid aliasing. In other words, energy should be conserved. It was also shown that, on a water tank process (see Section 5.4), the relation σ_u^2/σ_n^2 was almost constant for two different sampling rates. The controller was the same in both cases. This observation gives

$$\left\| \frac{S_{k_1}(z)}{\sqrt{h_1}} \right\|_2^2 = \left\| \frac{S_{k_2}(z)}{\sqrt{h_2}} \right\|_2^2,$$

which will force the relation

$$\|S_{k_1}(z)\|_2^2 = \left\| \sqrt{\frac{h_1}{h_2}} S_{k_2}(z) \right\|_2^2 = \left\| \sqrt{\frac{1}{J}} S_{k_2}(z) \right\|_2^2,$$

between the two cases. $J = h_2/h_1$ and corresponds to an up- or down-sampling ratio. It is very important to take the J-ratio into account when designing controllers. If not, variance will depend heavily on sampling rate, such that $\|S_k(z)\|_2^2 \to 0$ as $h \to 0$. Figure 5.7 shows PID designs for

$$P(s) = \frac{1}{(50s+1)} e^{-s} \tag{5.3}$$

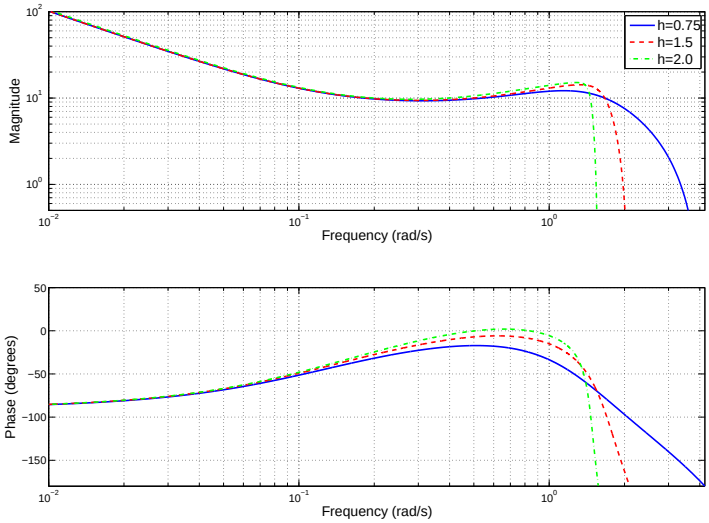

Figure 5.7 Three PID controllers determined for process (5.3) with $V_k \approx 1$, but sampled with different sampling time h.

using three different sampling times $h = 0.75$, 1.5 and 2 seconds. The measurement noise was assumed to be white, unit variance, noise sampled with $h = 0.01$ seconds. Using the noise sampling time as h_1 reference, this resulted in down-sampling ratios of $J = 0.75/0.01 = 75$, 150 and 200 respectively. As one can see, taking $V_k \approx 1$ gave very similar controllers. Performance was also almost the same in all three cases. If J had not been taken into account, however, the controllers would have been very different, depending on sampling time. In Example 5.2, this ratio was not part of the design and one could thus not compare the different designs directly. From now on, when comparing designs on different systems, J will be part of the procedure. In reality, however, one will not need to think about this when using the proposed method. There will only be a series of measurement data which can then be used to form a noise model or be directly used in simulation estimations.

5.3 Suggested procedure for design of PID controllers on real processes

Below follows one suggestion on how to design PI or PID controllers for real processes.

1. Collect noise data from the process, detrend it and estimate the variance σ_n^2.
2. Choose a number of different T_f values. For each T_f:
   - design a PI or PID controller using the proposed software.
   - simulate the closed loop system using the gathered noise data and estimate the variance, σ_u^2, of the control signal.
3. Plot IAE versus V_k (like in Figure 5.2) to get a picture of how much performance costs in terms of measurement noise throughput to the control signal.
4. Choose a suitable controller from the given set, taking the trade-off curve into consideration.

This new method ought to give the user a systematic way of bringing the D-part of the PID controller into industrial controller design. The structure of the controller should, however, be determined with respect to what specifications are set on the process and not be chosen in advance. A PI controller should thus be chosen if it is better suited than a PID and vice versa.

It would also be a good idea to have some software that automatizes the procedure. The user should, however, be able to modify parameters like ω_o and V_k. A set of different controllers should be kept in case one would later want to change how sensitive the control signal is to measurement noise.

5.4 Examples

This section will contain three simulated and one real example of how the proposed design method can be used. Youla parametrized controllers will be derived for comparison with the PI and PID controllers.

The three examples, in which simulations are used, will be assumed to have white measurement noise, with unit variance, acting on the process value. In the first example, three processes will be compared as described in the sampling issues part of Section 5.2. In other words, they will be assumed to have the same, but differently sampled, white measurement noise. The other two simulated examples will have measurement noise with the same sampling time as the process is sampled with. The three examples can therefore not be related to one another. It will be left to Chapter 7 to deal more with comparisons like that.

Before the examples are presented, the IAE measure will be discussed a bit.

Performance measure for time delayed systems

Assume that a process has a time delay of $L = 1$ second. This means that it will take one second until a load disturbance on the control signal shows up in the measurements. It will then take one more second before the control signal can affect the process value. So, no matter which controller is used, it will always take two seconds until the control signal can alter the process value. This will correspond to a fix amount of IAE that can not be reduced. Depending on how fast and time-delayed the system is, this part of the IAE may very well be the majority of the full IAE. This is especially true for systems with $\tau \approx 1$. Comparing two different controllers with the standard IAE measure could therefore give a false picture of how close they are in performance. For this reason, only the remaining part of the IAE, IAE_u (u denotes IAE affected by the control signal) will be used in the first of the simulated examples in this section. The new measure will, however, not be used for the real process example.

The examples

Example 5.3—First order time delayed processes

PID- and Youla parametrized controllers were derived for the three different FOTD processes,

$$P_1(s) = \frac{e^{-s}}{0.1s+1}, \quad P_2(s) = \frac{e^{-s}}{s+1}, \quad P_3(s) = \frac{e^{-s}}{500s+1},$$

Table 5.2 Results when PID and Youla controllers were derived for three different FOTD systems.

	P_1	P_2	P_3
K	0.21	0.61	28.3
T_i	0.42	1.01	44.6
T_d	0.13	0.41	0.00
T_f	0.0074	0.06	2.58
IAE_u^{PID}	1.09	1.51	1.62
IAE_u^{Youla}	0.61	1.20	1.38
IAE_u^{inc}	78%	26%	17%

where P_1 is delay dominant ($\tau = 0.91$), P_3 lag dominant ($\tau = 0.002$) and P_2 is somewhere in between, with $\tau = 0.5$. In all three cases, V_k was set to 1 although it was weighted with the down-sampling factor J as described before. The three systems were given different sampling times $h_{P_1} = 0.02$ s, $h_{P_2} = 0.05$ s and $h_{P_3} = 5$ s. The white measurement noise was assumed sampled with $h = 0.01$ seconds. In other words, $J = 2$, 5 and 500 respectively. Table 5.2 shows data from each of the three cases. The order of the Q-filter was 140 in all three Youla optimizations. Note that for P_3, a PI controller is given instead of a PID for the reasons mentioned in Section 5.1. The PID was, in this case, 27% worse than the PI controller regarding IAE_u. Let us now go into more detail on the different control loops.

Figure 5.8-5.13 shows the Bode diagrams and load disturbance responses of the PI/PID- and Youla controllers respectively for each of the three processes. The PID controller for P_1 gives an IAE_u that is 78% higher than that of the Youla controller. This is by far the biggest difference of the three. Looking at the load disturbance response (Figure 5.9), it is obvious that the great potential of the Youla controller makes it quite a lot better performance wise than the PID. The Youla controller does, however, have several peaks in the Bode diagram resulting in a major phase advance. While this is the secret to the grand control, it is also likely to give very bad robustness towards time de-

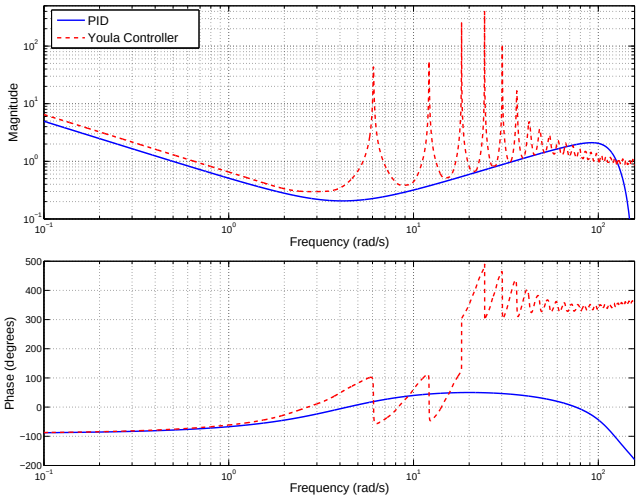

Figure 5.8 Bode diagram of the PID and Youla controllers for process P_1 in Example 5.3.

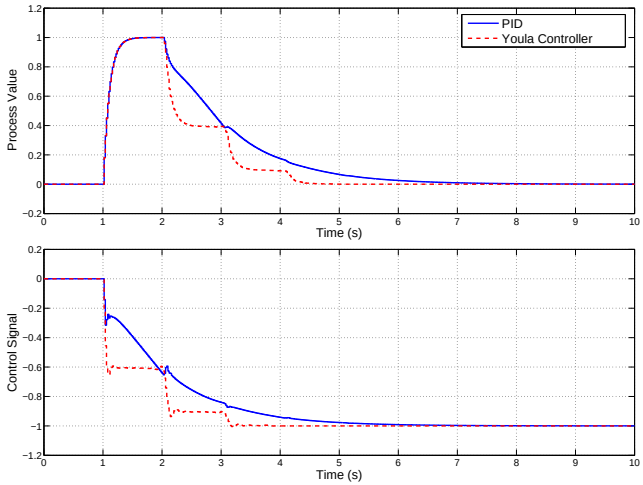

Figure 5.9 Load disturbance responses when the PID and Youla controllers are used respectively on process P_1 in Example 5.3.

lay variations. A small error in the modeled delay could potentially make the closed loop system unstable. Therefore, it would be advisable to have additional constraints put on the Youla optimization in order to compare systems really thoroughly. Adding such a constraint is, however, not trivial since it has to be closed loop convex. Given this, the PID controller may not give such a bad result after all and one could argue that a PI controller would be even better to use. The PI controller is only able to reach $\|S_k\|_2^2 = 0.02$. Its IAE_u-value of 1.39 is, however, only 28% higher than that of the PID controller. So, one gets an even less advanced controller with this rather small decrease in performance. It is also a known trick to switch off the D-part when controlling delay dominant processes (see for instance [Hägglund and Åström, 1991]). For P_2, there are still some peaks left in the Bode plot of the Youla controller (see Figure 5.10) although not quite as pronounced as for P_1. The performance gain in using the Youla controller is quite a lot smaller than for P_1, where IAE_u of the PID is just 26% off. The two load disturbance responses in Figure 5.11 are also quite similar with the biggest difference being the small undershoot of the Youla controller. This indicates that the PID controller is a valid choice in this specific case. With process P_3 (see Figures 5.12 and 5.13), a PI controller actually ends up giving better IAE_u than the corresponding PID controller (27% higher in this specific case), when using the proposed design method. Taking a look at the Youla controller should give a good hint of why this has happened. The best linear controller is basically a PI controller. This gives a hint that an alternative way of designing the controller could be to:

1. Derive a balanced model of the Youla controller
2. Reduce the new model, looking at the Hankel singular values

This way, one would end up with a good controller of low order, which does not depend on any beforehand given structure. One example of this type of reduction is given in Example 5.5 and another can be found in [Norman and Boyd, 1989].

It could be that the careful reader reacts to the PI control parameters being a bit too aggressive for P_3. This is, however, merely an effect of the measurement noise being down-sampled by quite a lot compared to the other two processes. On a real process it should never be a prob-

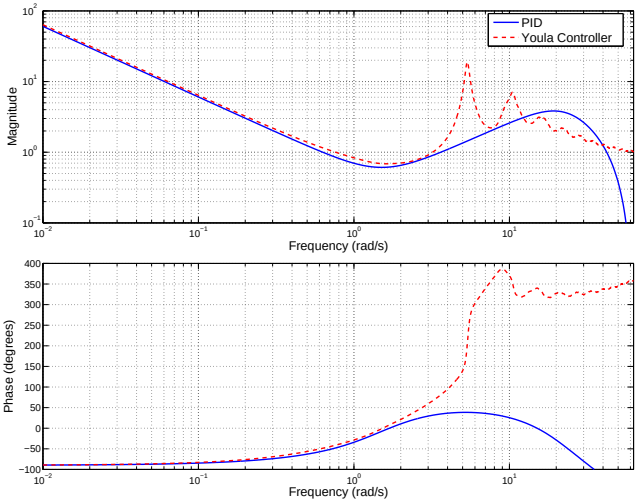

Figure 5.10 Bode diagram of the PID and Youla controllers for process P_2 in Example 5.3.

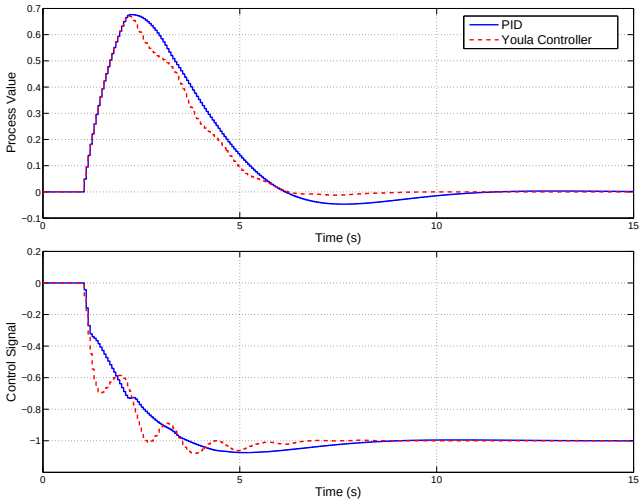

Figure 5.11 Load disturbance responses when the PID and Youla controllers are used respectively on process P_2 in Example 5.3.

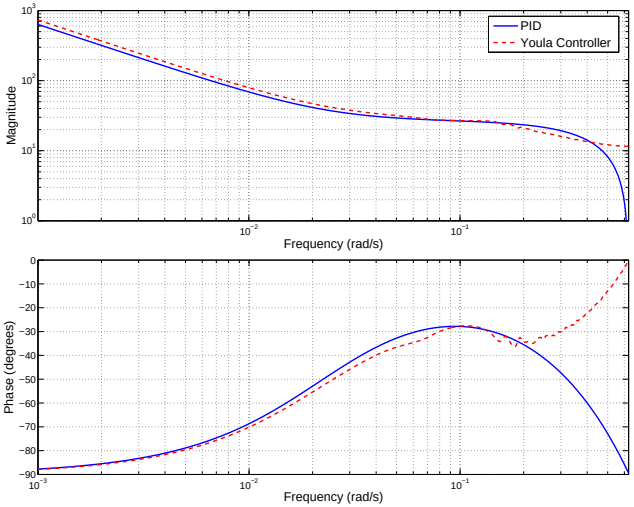

Figure 5.12 Bode diagram of the PID and Youla controllers for process P_3 in Example 5.3.

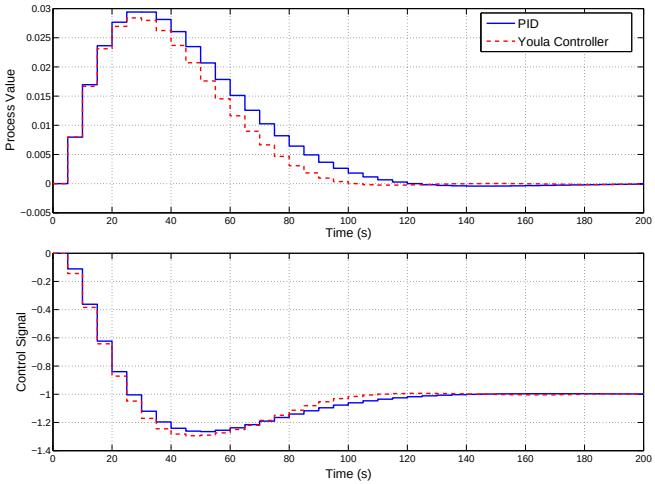

Figure 5.13 Load disturbance responses when the PID and Youla controllers are used respectively on process P_3 in Example 5.3.

lem. The controller will only be aggressive if the measurement noise and user allow for it.

Once again, the measure T_f/T_d can be used to get an indication of when a PI controller is preferred. For the PID controller on P_3, $T_f/T_d = 4.72/2.83 = 1.67$, which is quite high above 1.

The Youla controller for process P_3 confirms that, for a lag dominant FOTD, it is less likely that the D-part is valid than for a more delay dominant system. It also shows that it is reasonable to only have a first order low-pass filter on the PI controllers derived from the PID software. Another interesting fact is that all three Youla controllers have some sort of roll-off. This gives a hint that a PID controller with second order low-pass filter is the preferred choice of structure. In industry, however, it is most common to use filters of order one.

Youla parametrization with a finite FIR filter will provide a controller that is approximately the best linear controller. Or rather, it is the optimal controller for the set of controllers that can be created with the given Q-filter. The higher the order of the Q-filter is, the better the approximation of the optimal linear controller will be, such that it is close to the limit of performance. In the cases considered here, one can get an estimate of how close the Youla controllers are to this limit by plotting IAE_u versus the filter order, N_Q. Figure 5.14 shows this relation for process P_2. $N_Q = 73$ was the smallest value giving an integrator in the controller. As can be seen, the IAE_u changes very little for N_Q greater than 120. This gives an indicator that the Youla controller, for which $N_Q = 140$, gives an optimum close to the limit of performance and it will therefore be a good approximation of the best linear controller. Besides, the difference between the highest and lowest IAE_u is, in fact, only 9%. P_1 and P_3 have similar looking plots to Figure 5.14. Making these plots is also a fairly fast way of making sure that the Youla controller quality is good. □

EXAMPLE 5.4—FOURTH ORDER LAG PROCESS

The fourth order lag process

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

was investigated in the same fashion as the FOTD systems in the previous example. V_k was initially set to 1, when using a sampling time

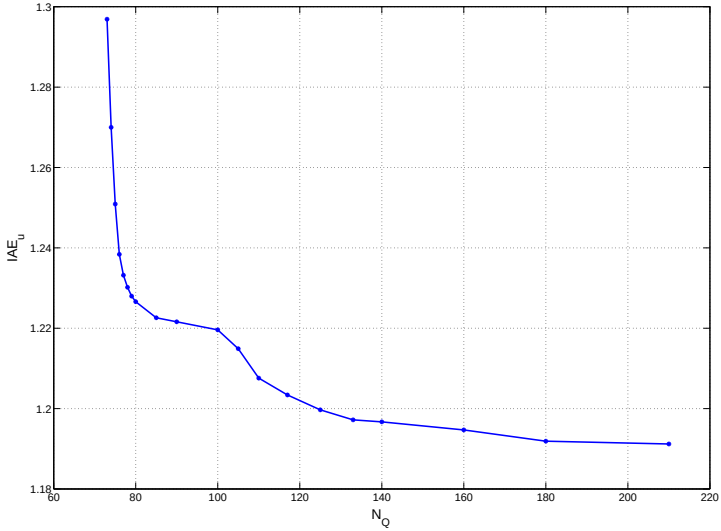

Figure 5.14 The relation between IAE_u and the order of the Q-filter for process P_2 in Example 5.3 indicates that the solution of the Youla optimization comes close to the limit of performance.

of $h = 0.25$ seconds and a Q-filter of order 150. Bode plots and load disturbance responses for the PID and Youla controllers are shown in Figure 5.15 and 5.16. The designed PID controller has the parameters $K = 0.73$, $T_i = 2.38$, $T_d = 1.29$ and $T_f = 0.52$, with IAE = 4.06. This can be compared to the IAE of the Youla controller which is 3.43. The PID controller thus gives about 18% higher IAE compared to the best possible linear controller. Considering this and how similar the load disturbance responses look, one gains fairly little from having a more advanced controller than PID in this case.

Now consider how the Youla controller alters with varying V_k. Figure 5.17 shows the Youla controllers when $V_k = 5$, 1, 0.2 and 0.04. The difference in IAE from the PID counterpart goes from 28 % when $V_k = 5$ to basically 0 % when $V_k = 0.04$. Looking at the Bode diagram, it is apparent that the controllers get less freedom for lower V_k-values. The resemblance also goes first towards PID controller and then PI,

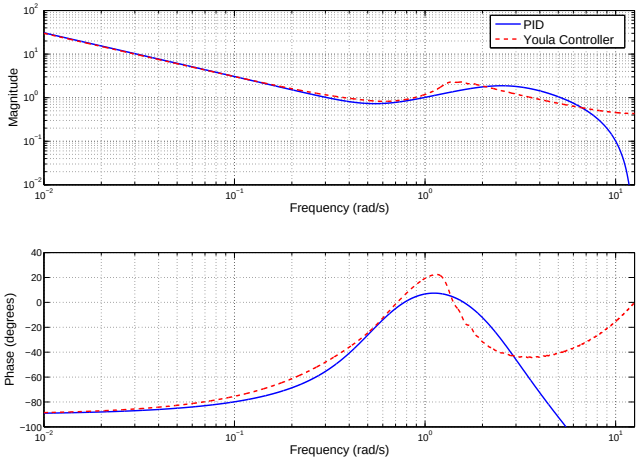

Figure 5.15 Bode diagrams of the PID and Youla controllers for the fourth order lag process in Example 5.4.

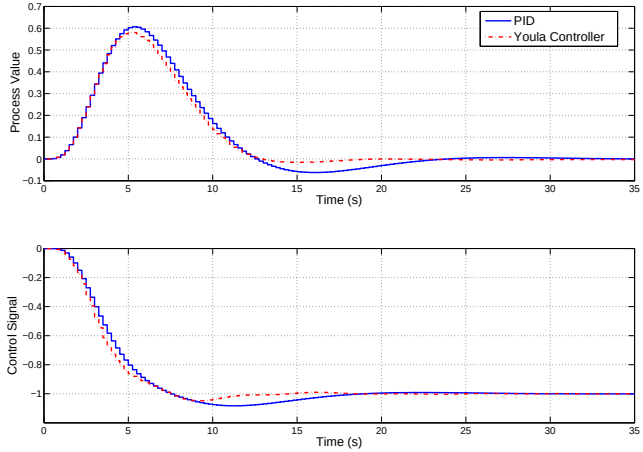

Figure 5.16 Load disturbance responses when the PID and Youla controller are used respectively on the fourth order lag process in Example 5.4.

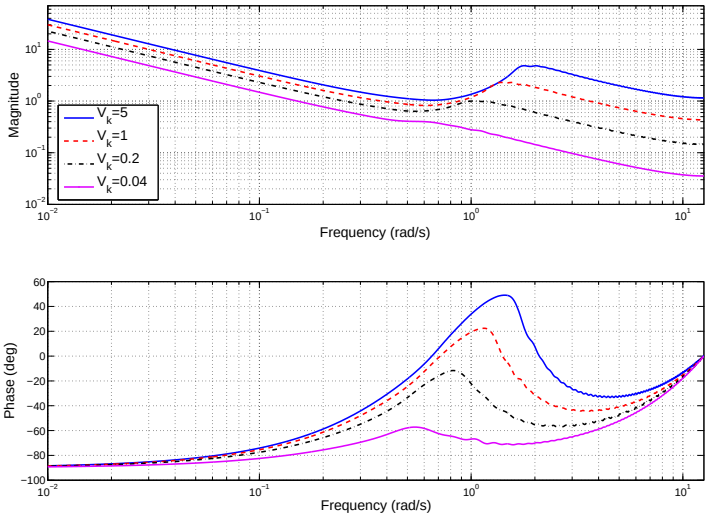

Figure 5.17 Fourth order lag process: Youla Controllers for four different values on V_k. As V_k decreases, the controller gradually becomes more and more similar to a PI controller.

when V_k is really small. For even lower values on V_k, it could even be that an I controller would be preferred. This means that the preferred controller structure depends on what expectations one has on the control signal variance. The level of variance at which a more advanced controller is preferred does of course vary from process to process. □

EXAMPLE 5.5—AN OSCILLATORY PROCESS
The next process to be examined is

$$P(s) = \frac{9}{(s^2 + s + 9)(s + 1)},$$

which was previously considered in Section 4.4. It is a highly oscillatory process, having two poles with relative damping $\zeta = 1/6$, known to give PID controllers trouble to quickly damp out load disturbances.

The process was run with $V_k = 1$, $h = 0.1$ seconds and a Q-filter of order 150. This gave the two controllers shown in Figure 5.18, with the load disturbance responses in Figure 5.19. The PID controller has $K = 0.30$, $T_i = 0.30$, $T_d = 0.43$ and $T_f = 0.06$ giving IAE = 1.44. The IAE of the PID is 48% higher than that of the Youla Controller (IAE = 0.97). This is a more distinct difference than most of the other cases investigated so far. The Youla controller is also free from the high peaks that characterized the controller for process P_1 in Example 5.3. If V_k is set to a higher value, the Youla controller will dominate even more since it has much higher design freedom.

In Figure 5.18, there is also a Bode plot of a sixth order controller that has been derived by reduction of the Youla controller in the same fashion described in Example 5.3. This can be compared to the Youla controller that was of order 151. The new, reduced, controller has the transfer function

$$C(z) = 0.80\frac{(z-0.87)(z-0.04)(z^2-1.85z+0.93)(z^2-1.72z+0.92)}{(z-1)(z-0.68)(z^2-1.53z+0.73)(z^2-1.74z+0.94)},$$

giving $M_s = 1.44$, $M_p < 1.4$, $V_k = 1.0002$ and an IAE-value only 0.03% higher than the Youla controller. These are, in other words, basically the same controller, which is quite obvious from looking at the Bode plot. Seeing that there is an almost identical pole/zero couple in $C(z)$, a controller of order 4 was also derived, in the same fashion. This controller gives the properties $M_s = 1.465$, $M_p < 1.4$, $V_k = 1.003$ and an IAE-value that is 3.4% higher than that of the original Youla controller. This shows that the Youla controller can be reduced considerably, giving new controllers that are very close to optimal. This controller design method could be carried out on its own by a more advanced user or just give a good initial guess for further optimization of a controller of lower order (4 or 6 in this case).

All in all, the Youla optimization seems to be a good tool to have in combination with the PID design software. The PID software can quickly derive a large quantity of good PID controllers for simple processes. The Youla parametrization software can then be used to justify the choice of PID controllers. If the PID controllers are not performing well enough for a given process, the Youla controller could give a hint of a better design and perhaps even give such a controller through reduction. The reduction is, however, far from trivial and should therefore

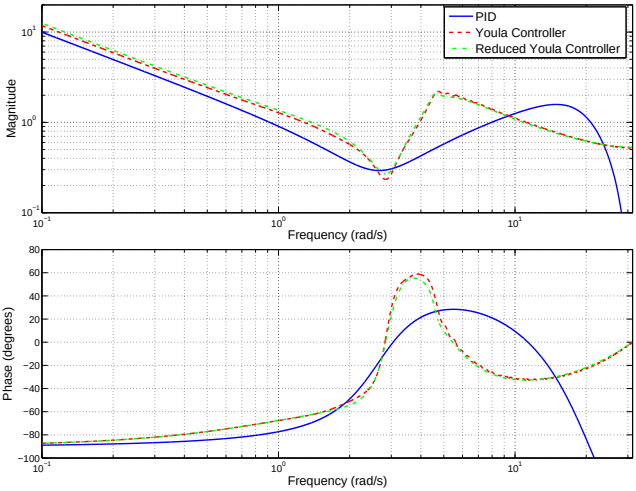

Figure 5.18 Bode diagrams of the PID controller, Youla parametrized controller and a reduced Youla controller of order 6 for the highly oscillatory process in Example 5.5. The Youla controller and reduced order controller can just barely be distinguished by the eye.

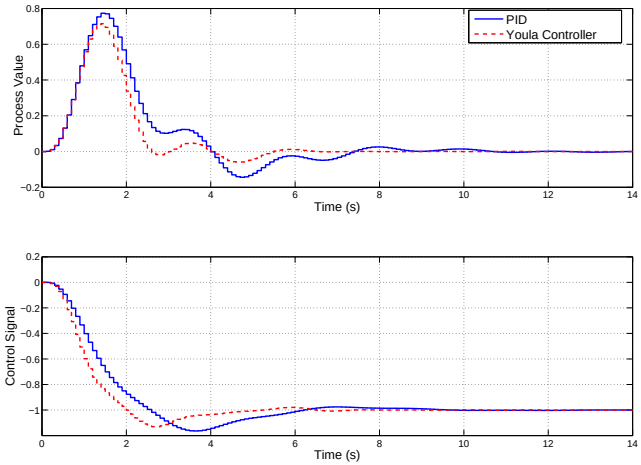

Figure 5.19 Load disturbance responses when the PID and Youla controllers are used on the oscillatory process in Example 5.5.

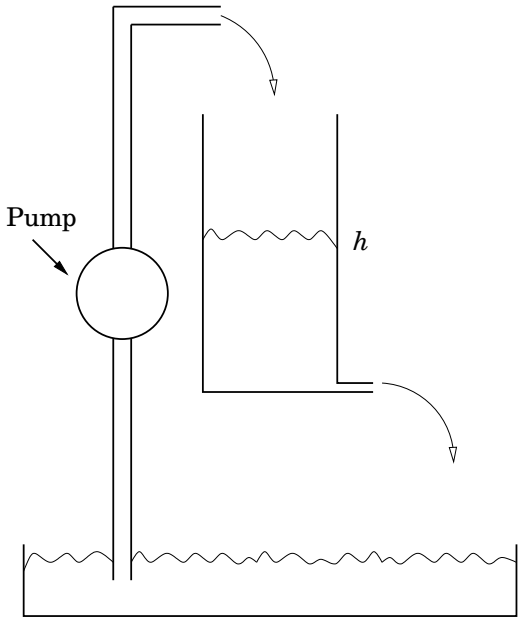

Figure 5.20 Sketch of a water tank process.

be done with care. It could be that the optimal controller enhances some aspects that the PID controller does not, like for instance giving bad delay margins or even give an unstable controller. This should be carefully investigated before such a controller is implemented. It should also be made sure that the extra time spent on deriving a more advanced controller is really worth it. □

EXAMPLE 5.6—A WATER TANK PROCESS

The following example will illustrate that the control signal variance for a real process can be sufficiently estimated. The process of concern is the water tank system shown in Figure 5.20. The equipment is frequently used in education and therefore well documented. The water tank itself is cylindrical in its shape, with a cross-section area, A. In the bottom of the tank is an outlet hole with area a for the water to

flow out. The tank is provided with a water flow assumed proportional to the voltage, u, sent to a pump. The objective of the control is to keep the water tank level, h, close to a certain operating point, which in this case was chosen to be 10 cm (mid-level). The water level is given through an instrument, measuring the pressure in the tank and the output is a voltage, y, proportional to h.

A physical model of the process can be derived using mass balance and Bernoulli's law:

$$A\frac{dy(t)}{dt} = -a\sqrt{2gy(t)} + ku(t), \quad u(t) > 0$$

Where g is the acceleration of gravity and k is some proportionality constant. If the model is linearized around a certain water level y^0 and input voltage u^0, it becomes

$$\frac{d\Delta y(t)}{dt} = -\frac{1}{T}\Delta y(t) + \frac{k}{A}\Delta u(t),$$

with $\Delta y(t) = y - y^0$, $\Delta u(t) = u - u^0$ and

$$T = \frac{A}{a}\sqrt{\frac{2h}{g}}.$$

The water tank process is therefore essentially a first order process and easily controlled with a PI controller. To make the system complexity a bit higher, a time delay of $L = 10$ seconds was added.

The process was modeled using a step response test, first setting the voltage to the pump to give a water level just below 10 cm. The control signal was then increased by 0.1 V, sending the water level just above the reference. Such a test ought to give quite a good linearized model of this first order process. Fitting an FOTD model to the step response data gave

$$P(s) = \frac{4.1}{80s + 1}e^{-10s},$$

which has a normalized time delay of

$$\tau = \frac{L}{L + T} = 0.11.$$

Measurement data was then collected while running the process with a constant control signal. Figure 5.21 shows the raw data from the tank level measurements. Choosing ω_o in the high-pass filter (5.2) to 0.16 rad/s (well above possible closed loop cut-off frequencies) resulted in the detrended signal shown in Figure 5.22. This frequency was chosen more on feel than using a systematic method. Trying a few different choices of ω_o, however, gave very similar results in variance, so it should not give any drastical errors if ω_o is not set optimally. It is, however, wise to let the user be able to specify ω_o. Figure 5.23 shows the low frequency part of the original signal that was removed.

The water tank was sampled every 0.1 seconds. In contradiction to the other examples in this chapter, the controller was not derived using the discrete time version of the design software. A continuous time controller was derived for the sole reason that the discrete version had not yet been written at the time the tests were carried out. Instead, a zero order hold sampled version of the continuous controller was implemented in a Simulink environment. Since the sampling rate is quite fast for these experiments, this barely affected the robustness constraints. There could, however, be cases when the sampling time is chosen differently and it is more critical to use a discrete time designed controller. This will become obvious in the next chapter.

Two different PID controllers

$$\mathrm{PID}_1(s) = 0.92\left(1 + \frac{1}{26.52s} + 4.56s\right) \cdot \frac{1}{0.5^2 s^2/2 + 0.5s + 1},$$
$$\mathrm{PID}_2(s) = 0.78\left(1 + \frac{1}{29.65s} + 5.42s\right) \cdot \frac{1}{2.44^2 s^2/2 + 2.44s + 1},$$

were designed using the PID software. The first PID controller has T_f fixed to 0.5 while the second one has $T_f = 2.44$. Both controllers were run in closed loop for several hundred seconds and Figure 5.24 shows the detrended control signals in both cases. The first PID controller gave a control signal variance of $9.17 \cdot 10^{-5}$ ($V_k = 37.1$) while the second one gave the variance $3.46 \cdot 10^{-6}$ ($V_k = 0.79$). Simulink simulations, using the detrended measurement noise, gave $V_k = 35.5$ and $V_k = 0.76$ respectively. Thus, for PID_1 there was a difference of 4.5% between the real variance and the predicted one. For PID_2 this difference was 4%. Figure 5.25 shows a short sequence of the real and simulated

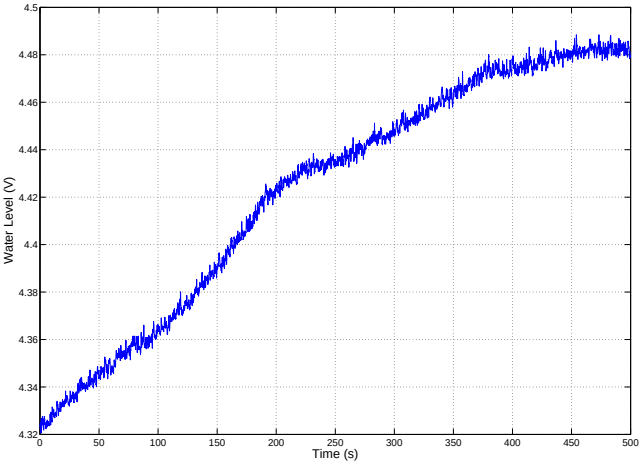

Figure 5.21 Raw measurement data from the water tank process while the control signal was held constant. The low frequency content of this data is to be removed, such that only the part of the signal that is assumed affecting actuator wear is left.

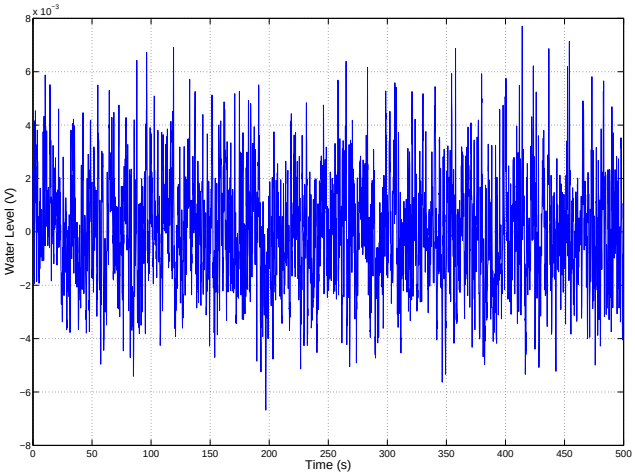

Figure 5.22 Noise data left when the signal in Figure 5.23 is removed from the one in Figure 5.21. This signal can be used for estimation of the control signal variance.

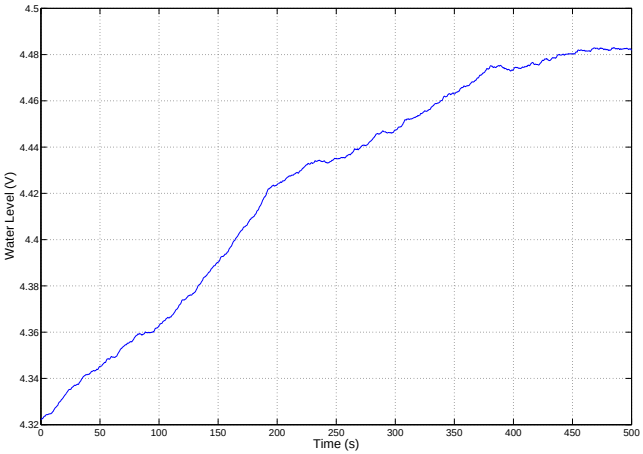

Figure 5.23 Low frequency data removed from the water tank measurement in Figure 5.21.

control signals (for PID_1) verifying that the two signals are indeed very similar. Figure 5.26, shows the frequency content, using Bode diagrams, of the real and simulated control signals (still with PID_1). The curves show once again the similarities between the signals and that the biggest difference is for low frequencies. The low frequency content, however, matters quite little as it only indicates towards the real control signal still having some low frequency trends left.

In all, this example shows that the control signal variance can be quite well predicted on a real process, in the manner that the proposed design method demands. □

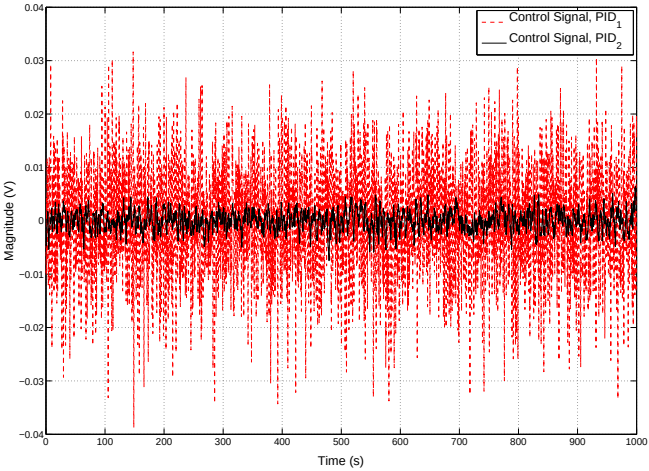

Figure 5.24 Real, detrended, control signals for PID_1 and PID_2 on the water tank process.

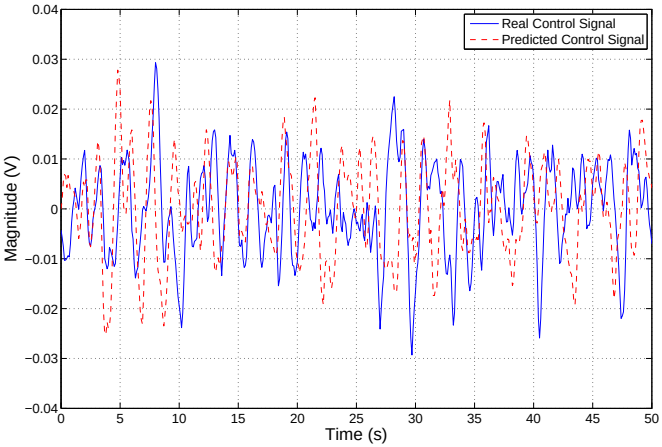

Figure 5.25 Real, detrended, control signal and predicted one when PID_1 was used on the water tank process.

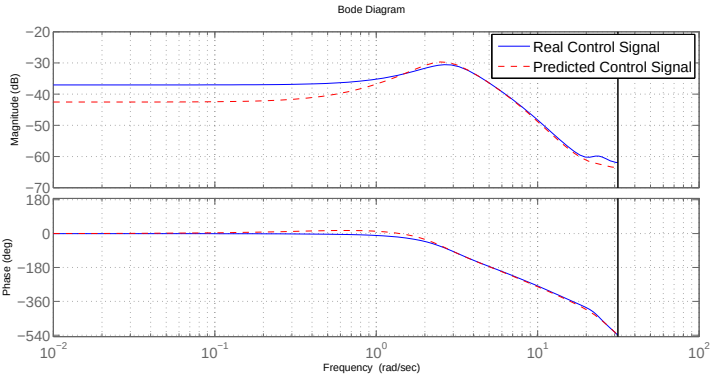

Figure 5.26 Bode diagram showing the frequency content of the real and predicted control signals when PID_1 was used.

6

Industrial Example

While the proposed PID design method has proved to work properly in both simulations and on lab processes, it should also go through exhaustive tests on industrial processes to show its worth. Through Akzo Nobel Functional Chemicals AB, Stenungsund, it has been possible to do such initial experiments on an industrial distillation column. The task was to control a recirculation flow in this column.

6.1 Background

Previous modeling of the recirculation flow process (done by people in the industry) has given the following first order system with time delay

$$P_0(s) = \frac{3.8}{5.5s + 1} e^{-1.4s}.$$

The PI-controller used to control this process was determined through lambda tuning with $T_{cl} = 1.5T$. In other words, the design goal was to make the closed loop system about 1.5 times slower than the process. This PI controller, PI_λ, had the parameters $K = 0.15$ and $T_i = 5.5$. It is important to know that the measurement data used for deriving the model $P_0(s)$ was filtered through a first order low-pass filter

$$G_{f_0}(s) = \frac{1}{s + 1},$$

which is later assumed to be part of the controller in the same fashion as the controllers designed using the proposed method. Setting the

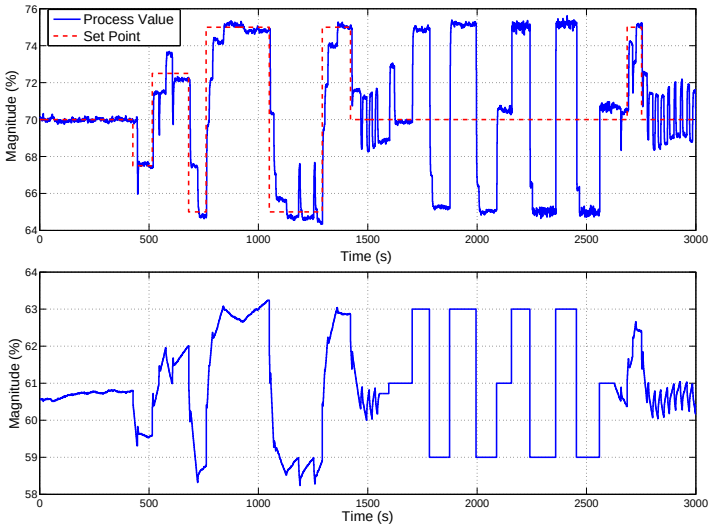

Figure 6.1 The upper plot shows the filtered process value (solid) and set point (dashed) from the recirculation flow system, while the lower displays the control signal. The process has been run in both open and closed loop.

filter time constant, however, was never part of the PI design. It was set in advance. The sampling time of the process was also preset to $h = 1$ second.

6.2 Initial tests

Data from the process is presented in Figure 6.1. The upper plot shows the process value and set point while the lower one shows the control signal, all during experiments on the process. The system was first run in closed loop at a constant set point and then with step changes to the same. Looking at these changes it is evident that there is stiction in the actuation. The integral part of the controller grows until the control signal is high enough to overcome the friction. This is obviously giving some unwanted behaviour. Stiction effects of this sort is a problem of

its own which will not be treated in this thesis. The part of the data that is heavily affected by stiction will therefore be ignored. After the closed loop tests, the process was run in open loop, making several step changes in the control signal. The filter on the measurements is activated at first, but then switched off after a while. In the end, the process is put back in closed loop, giving quite bad control with a limit cycle as a result, most likely due to the stiction.

6.3 Modeling the process

Two new models of the process were derived using the Matlab based modeling tool presented in [Wallén, 2000] - one with the measurement filter active and one without. Several different models were first derived based on step response data. The final two models were given using the mean values on K_p, T and L of the set of models. The first model, derived without the measurement filter active, is

$$P_1(s) = \frac{2.60}{0.50s + 1} e^{-0.84s},$$

while the second model, with filter, is

$$P_2(s) = \frac{2.60}{1.30s + 1} e^{-0.84s}.$$

The time constants and the time delay varied quite a bit between the different model derivations, the reason being that they are so close to the sampling time. $P_2(s)$ is quite different from the original one, $P_0(s)$. The reason for this could, for instance, be that the process has changed over time. Whether or not this is the case, it is still quite apparent that the sampling time has been chosen too long for this particular system. The rise time of the process, for instance, is just one or two samples. The slow sampling rate will actually make the system harder to design for, introducing even more time delay. This is something that has to be taken into account when designing the controllers later.

The discrete time version of $P_1(s)$ was used to derive PI and PID controllers with the proposed method.

6.4 Noise data collection

A high-pass filter with a cut-off frequency three times as high as for the original closed loop ($\omega_o = 0.23$ rad/s) was used to filter out the noise from the measurements. The remaining signal had a variance of $\sigma_n^2 = 0.0255$, giving $V_k = 0.006$ for the lambda controller which corresponds to very little noise throughput. Estimations of this relative variance, using the detrended noise data as input in system simulations gave 40 percent higher variance, but this could very well be due to, for instance, differences in the discretization of the controller. The simulated controller used a forward Euler approximation of the I-part and backward Euler on the low-pass filter. It will be assumed that the noise data is at least good enough to use for control signal variance estimations.

6.5 Controller designs

First a brief look at the existing controller for the system. Given that it is discretized as described, it gives $M_s = 1.16$ and $M_p = 1$. The IAE_u value becomes 36.7 and a load disturbance response is quite sluggish. The reason being that the controller is tuned for quite a different process $P_0(s)$. If one instead makes a new lambda tuning on the process, with $T_{cl} = 1.5T$ once again, the controller becomes

$$G_r^{\lambda}(s) = 0.18\left(1 + \frac{1}{1.3s}\right) \cdot \frac{1}{(s+1)},$$

i.e. with $K = 0.18$, $T_i = 1.3$ and $T_f = 1$. The controller was tuned with $P_2(s)$ as process model, in the same fashion as the original lambda controller. If all systems within the closed loop were continuous, M_s would be 1.49 (M_p is small enough to be neglected). If the new controller and plant now instead are discretized, like they would be in reality, the slow sampling rate will distort the robustness such that M_s becomes 1.60. The relative variance, V_k, with G_r^{λ} is 0.035 with an IAE_u of 10.33. The change in robustness after discretizing the system shows that it is a good idea to use the discrete time version of the Matlab software.

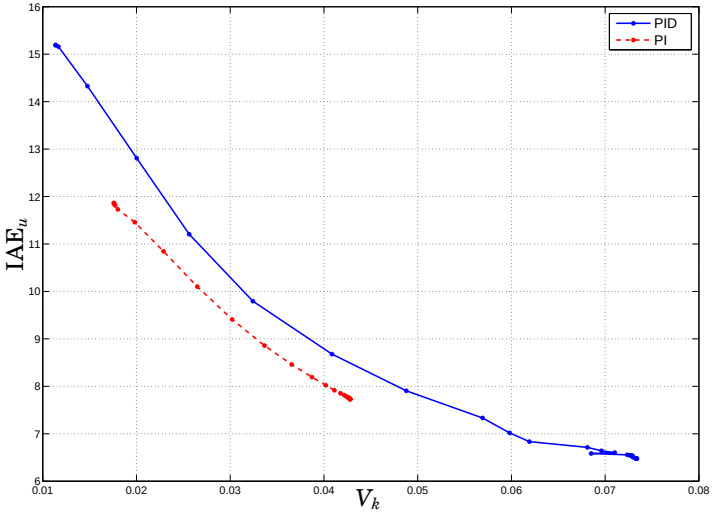

Figure 6.2 For thirty different values on the low-pass filter time constant, T_f, PI and PID controllers (with $M_s = M_p = 1.4$) were designed for the flow control, using the real noise as input to determine V_k. The plot shows the relation between IAE$_u$ and V_k. That is, how much performance costs in terms of noise in the control signal.

Thirty PID controllers and thirty PI controllers were designed using a logarithmic span of T_f values from 10^{-5} to 15. The controllers were designed on the zero order hold discretization of process $P_1(s)$. M_s and M_p were both set to 1.4 in all runs. Figure 6.2 shows how much performance, i.e. IAE$_u$, costs in terms of control signal variance, V_k. Having access to this plot should be of great help to a PID designer when choosing a suitable controller for the process at hand. Analysing the plot further shows that it could, as before, be that the PI controllers are actually better suited than the PID counterparts. Once again, T_f/T_d can be used as a measure of when the performance of PI controllers should be checked too. If this relation is above or close to 1, design PI controllers as well. When $T_f \to 0$, V_k will approach some fix value. The reason being that the Nyquist frequency will cut off the influence of the low-pass filter, such that there is a lower limit on IAE$_u$, thus also

limiting how high V_k can become. For the PID controllers, this IAE_u limit is lower than for the PI controllers. As will be seen later, however, it can be adjusted by varying the robustness measures.

Figure 6.3 shows the Bode diagrams of all PID controllers and Figure 6.4 pictures how the proportional gain varies with V_k. The other two controller parameters vary along the same pattern. There are a few conclusions one can draw from these plots. Looking at the Bode diagrams shows that the PID controller, due to the long sampling time, will not be able to gain full use of the D-part as it is cut off at the Nyquist frequency. As T_f increases and V_k decreases, the controller will assume both PI control structure and, in the end, even the form of an I controller. For this reason it may look a bit odd that all the controller parameters increase as V_k decreases. The explanation to this is that the zeros of the controller is used to cancel out the low-pass filter. So when T_f grows, so will K, T_i and T_d. This is not a welcome behaviour since it would be better to just use an I controller right away. The question is of course if one would really want an I control structure anyways. The V_k values for these controllers are quite low.

Now select the best PI or PID controller that gives a V_k similar to what G_r^λ resulted in (and with roughly the same robustness). That controller is, in this particular case, the PI controller giving $V_k = 0.039$ with an IAE_u of 8.77 (and $M_s = M_p = 1.6$). The controller have $K = 0.27$, $T_i = 2.34$ and $T_f = 1.29$. The power of the new method is in other words not necessarily that it gives massively better performance compared to the lambda tuning, but rather that the designer is able to see the cost of performance in terms of V_k and then choose the controller knowing this trade-off relation. If one would instead prefer the best PI controller, it would give a $V_k = 0.043$ and an IAE_u of 7.73. The best PID on the other hand, would result in $V_k = 0.073$ and an IAE_u that is 6.48.

The design software was also run with several other robustness measures, namely $M_s = M_p = 1.2$, 1.3 and 1.6. Figure 6.5 shows all eight (PI and PID) trade-off curves for the different robustness measures. First of all, it is apparent from these curves that the method does not give optimal PI/PID controllers. The PI controllers are in general giving lower IAE_u-values than the PID counterparts at the same V_k. A true PID optimization would not give worse result than a PI optimization. It should ideally be able to assume the same form as the PI

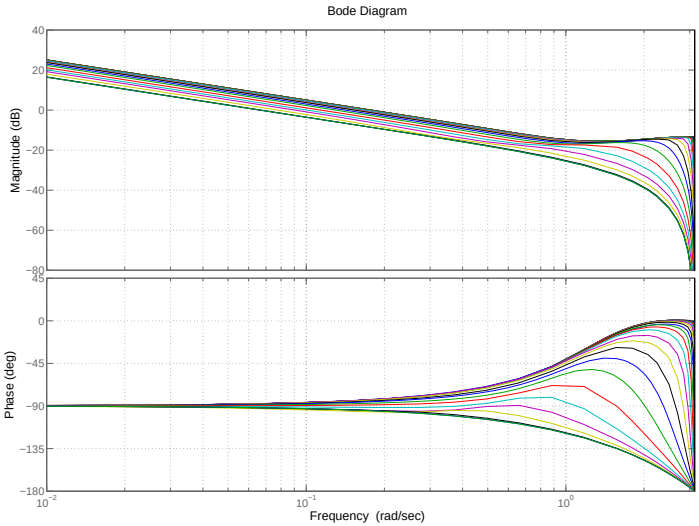

Figure 6.3 Bode plots for all thirty PID controllers, using $M_s = M_p = 1.4$. As T_f increases, the controllers will range from PID to PI and even I control structure.

if that is the best structure for the process. That will, however, not be possible due to the structure of the low-pass filter. The difference in performance, however, is not massive and it should be enough to always check the T_f/T_d relation to see if it is worth using the D-part or not. It is also clear from the plot that the robustness measure can be used to drive the limit of V_k up or down. It should, however, not be compromised too much such that the closed loop is given poor robustness.

6.6 Comparison with Youla controllers

An interesting question is: How do the designed PI and PID controllers stand up against the best linear controllers? To answer this, Youla parametrized controllers were derived both with and without any V_k.

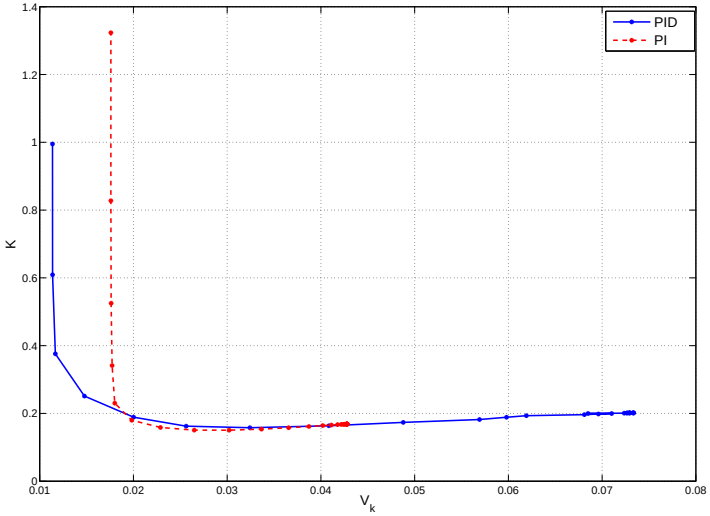

Figure 6.4 Proportional gain, K, as a function of V_k for the PI and PID controllers when $M_s = M_p = 1.4$. T_i and T_d varies along the same pattern.

The controllers were designed with M_s and M_p equal to 1.4, using a Q-filter of order 100. Without any constraint on the variance, the robustness measures and sampling time still limited V_k to 0.24, resulting in an IAE_u of 5.34. This can be compared to the PID controller giving the lowest IAE_u of 6.48 with $V_k = 0.0734$. Figure 6.6 shows the Bode diagram of this Youla controller. It is obvious that the increase in gain and phase close to the Nyquist frequency is what differs most from a regular PI/PID controller. Such a peak could, however, give rise to poor robustness to time delay changes. If the Youla controller instead is constrained to a V_k close to the best performing PI and PID controllers, one can see that the relative differences in IAE_u between these controllers are 4% (PI relative to Youla) and 5.5% (PID relative to Youla). These differences in performance are very small. If the best performing PID is instead compared to the best performing Youla controller, the difference is 25%, which is not too considerable an amount either. Fig-

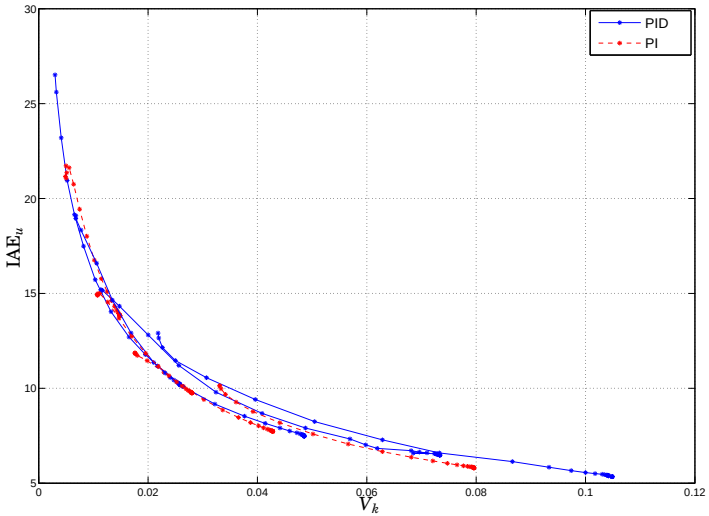

Figure 6.5 PI and PID controllers for $M_s = M_p = 1.2$, 1.3 and 1.6

ure 6.7 shows the IAE_u plotted against V_k for all Youla parametrized controllers (when $M_s = M_p = 1.4$) together with the PI/PID curves that were already given in Figure 6.2 and 6.5. The conclusion here is that the process does not benefit much from having a more advanced controller.

It has been concluded that the high sampling time gives a rather conservative limit on how well the controllers can perform, given reasonable robustness measures.

6.7 Results and conclusions

After the first industrial data had been analyzed, a request was sent to Akzo Nobel to try out the three controllers listed in Table 6.1. PID_1 is the best performing PID controller for $M_s = M_p = 1.4$ and PI_2 is the equivalent of PI controllers. The reason for selecting the best possible controllers was because the noise is quite insignificant for this process.

Table 6.1 PI and PID controllers to be evaluated on the real process.

Controller	K	T_i	T_d	T_f	IAE_u	Est. σ_u^2/σ_n^2
PID_1	0.20	1.13	0.26	0.00	6.48	0.073
PI_1	0.27	2.34	0.00	1.29	8.77	0.039
PI_2	0.17	1.30	0.00	0.00	7.73	0.043

It should thus be alright to push the performance while still keeping the system robust. PI_1 was chosen because of (as described earlier) its close relations to a lambda control strategy.

About four and a half months after initial data had been collected, result data arrived. Since the new experiment was conducted so long after the initial ones, the old model had to be validated with the new

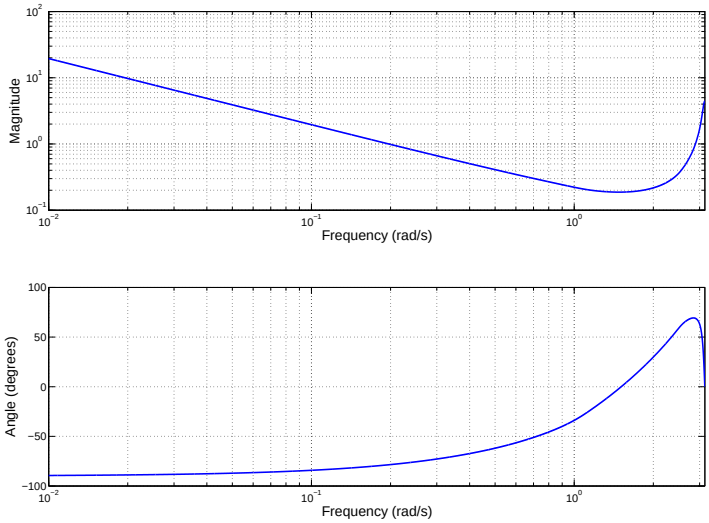

Figure 6.6 Youla parametrized controller when $M_s = M_p = 1.4$ and there is no constraint on the relative variance V_k. Note the phase and magnitude increase close to the Nyquist frequency.

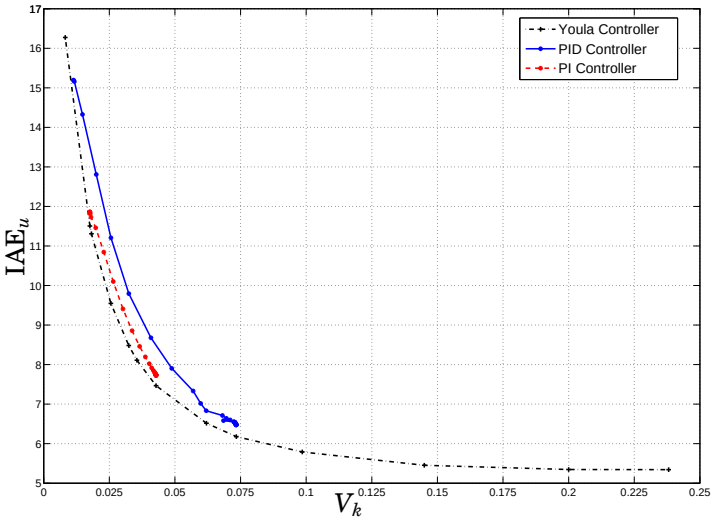

Figure 6.7 Trade-off curves for PI, PID and Youla parametrized controllers when $M_s = M_p = 1.4$.

data to see how significant the process alteration had been. The same had to be conducted for the noise selection. The filtered process values (solid) and set point (dashed) from the evaluation experiments can be found in the upper part of Figure 6.8, while the control signal is shown below. The red signals show the time line when the process is controlled by the old lambda tuned controller PI_λ, however, with the big difference that $T_f = 2$ instead of 1 (which used to be the case). The reason for this change is not known. The blue curves show step response data, used to verify if the process and measurement noise characteristics had changed or not. The black data shows discarded information clearly affected by actuator stiction. As stated before, this problem is not related to the choice of controller and can thus be ignored. The purple data shows the closed loop signals when controlled by PID_1. The sharp twist in the middle of the data will, however, not be part of the analysis. The first sequence of green data corresponds to control using PI_1 and the second to PI_2 control. When analysing

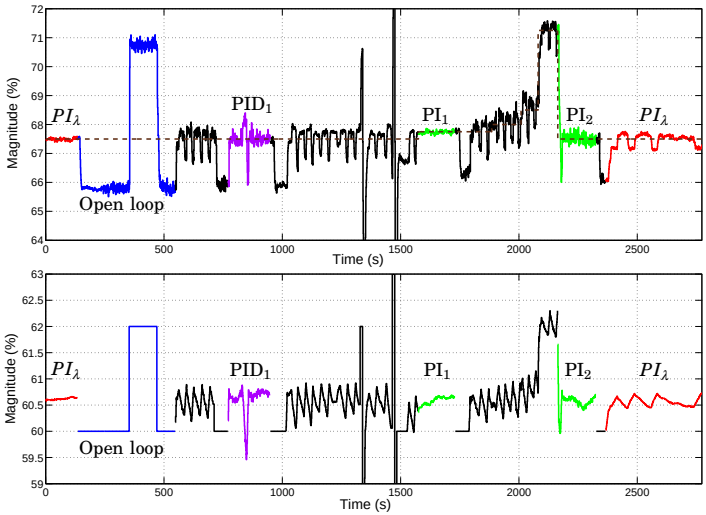

Figure 6.8 The upper plot shows the filtered process value (solid) and set point (dashed) from the industrial verification data, while the lower displays the control signal. The three controllers from Table 6.1 was used to evaluate the estimated variances.

the process values, it is important to realize that these are the signals coming in after the low-pass filter. Since PI_λ and PI_1 both have active filters, these signals may look much nicer to the eye than the sequences of the other two controllers. For that reason it would have been preferable to have access to raw measurement data instead.

First of all, the step response data was used to derive a new model. Since the old model seemed to be valid for the new data as well, it was assumed that the process had not changed significantly enough to alter the model. The process noise had changed a little over time, so that $\sigma_n^2 = 0.022$ instead of 0.026, but that is not a significant change either.

Earlier, quantization effects have not been considered. Since the control signal is sometimes quite close to the quantization level in magnitude (most significantly for PI_λ and PI_1), however, the effect was introduced in the evaluation analysis. The quantization level for

the control signal is 0.01%.

Starting out the analysis with PID_1, it is obvious that the control signal variance is highest for this particular case, just as expected. The difference from the other two cases would, however, be rather insignificant if the plot had been zoomed out a bit more. One has to realize that the levels of the noise are in magnitude $0.1 - 0.3\%$ of the maximum possible control signal, which are quite small values. Taking the noisy signal after the sharp switch as data, $\sigma_u^2 = 0.074$ which is 1% off the expected variance (all estimates were recalculated for the new noise). This is, in other words, a very good match.

The two PI controllers have control signal variances $\sigma_u^2 = 0.0137$ for PI_1 and 0.0348 for PI_2. In the case of PI_2, the estimate (0.0432) is 24% higher than the real value which seems reasonably close. For PI_1, on the other hand, the variance is quite far from the expected value of 0.0303 (121% difference). Taking a closer look at the control signal for PI_λ shows that its variance is also quite a lot lower than the expected value (about half of it). One possible reason for these differences is that the low-pass filter might have the wrong model, which would explain why the other two cases, where the filter is inactive, are much closer to the truth. An interesting observation is that setting the filter model to second order (double pole) actually gives an estimate exactly at the correct variance for PI_λ. For PI_2 the difference is still 85% after changing the filter order, but this do at least give one option of a possible error source. It also shows how important it is to have the correct setup for the actual control system. Even a slight alteration could mean that the values end up quite far from the real. For this reason it is highly desirable that control loops are well documented. This includes the way in which the controller has been implemented. Just assuming that the low-pass filter is approximated by forward Euler instead of backward would give quite a big variance increase. One way to get around this issue would be to:

1. Determine the controller that is likely to give the correct variance.
2. If the variance is not the expected, choose either a more or less aggressive controller on the trade-off curve.
3. Iterate until the control looks satisfactory.

Since the experiments were so short, it is hard to compare the perfor-

mances of the different controllers. In order to make such a comparison, one would need to have the plant running a long time with the different controllers active and then compare how the process value varies over time.

The experiments on the real plant have shown promising results. It was not an ideal plant to try the new method on considering sampling rate and stiction effects, but it has given a good indicator that the proposed design method is reasonable. Not all three cases gave variance close to the estimated, but then again, all signals were held within reasonable bounds and none of them blew up in magnitude.

The possibly most interesting conclusion one can draw in conjunction to these industrial experiments is that one can now tune a real PID controller without the fear of having the control signal vary too much. One would, however, still need a lot more tests on industrial plants before the method can be fully trusted and utilized.

7

When are PI and PID Controllers Valid Choices?

Previous chapters have shown that PI and PID controllers designed, using the proposed method, may very well be close in performance to the best possible linear controllers. This was shown by optimization of Youla parametrized controllers. In this chapter, it will be attempted to give a bit more general guidelines on when PI and PID controllers are valid choices compared to more advanced control structures. The work presented here has, however, not yet been in development for very long and there is thus more to do before any general conclusions can be drawn. The main goal of this chapter is therefore to inspire future work rather than presenting the definite answer to when PI and PID controllers should be used.

Three sub-batches from the AMIGO test batch (3.1) have been examined. PI and PID controllers have been compared with Youla controllers for several batch processes. In all cases, it has been assumed that $M_s = M_p = 1.4$ and that the measurement noise is white, with unit variance, sampled every $h = 0.01$ seconds. This sampling time will be treated as the reference sampling time when the up- or down-sampling ratio, J, is derived. That way, one can compare all processes no matter the respective sampling times.

It has already been pointed out that the PI and PID controllers derived, using the proposed design method, are not optimal with respect to the variance constraint. To get the full picture of when PI and PID controllers are valid choices, the optimal controllers should ideally be

used. One way to determine these would be through gridding. This will not be done here though and one should thus see this chapter mainly as a way of justifying the proposed design method. It should, however, still be possible to see some trends that are likely to occur even when the optimal PI and PID controllers are employed.

For the three sub-batches, a limited proportion of the total amount of processes were analysed. Each process had seven PI controller designs and seven PID controller designs - spanning over T_f. The PI controllers have, however, not been used much in the analysis here. The V_k-values resulting from the runs were used to design Youla controllers giving the same variance to render a comparison possible.

7.1 Sub-batch 1 – First order systems with time delay

The first sub-batch to be investigated is the one with time delayed first order systems, that is,

$$P_1(s) = \frac{e^{-s}}{1+sT},$$
$$T = 0.1,\ 0.5,\ 1,\ 2,\ 5,\ 10,\ 50,\ 500,$$

such that 8 of 21 processes were analysed.

Figure 7.1 shows a plot of the relation between $\text{IAE}_u^{PID}/\text{IAE}_u^{Youla}$ and V_k. In other words, this shows how well the different PID controllers perform compared to the Youla controller fulfilling the same variance constraint. Most of the curves in Figure 7.1 have the same appearance with a fast growing relative IAE_u for low V_k-values, reaching stationarity for high V_k-values (see for instance the $T = 0.1$ case). This seems rather reasonable since the robustness measures will likely destroy the chances of the Youla controller taking full advantage of its superior freedom when V_k is high. For values on T from 5 and above, the curves are shaped such that the relative IAE_u decreases for high V_k. It does, however, seem logical that the shape of the curve is more accurate for low values on T. The reason being that it is reasonable if the PI or PID structure has the highest benefits for low V_k-values, leaving the Youla controller with little freedom. This is also confirmed by looking at the relative IAE_u for PI controllers. These have a lot lower

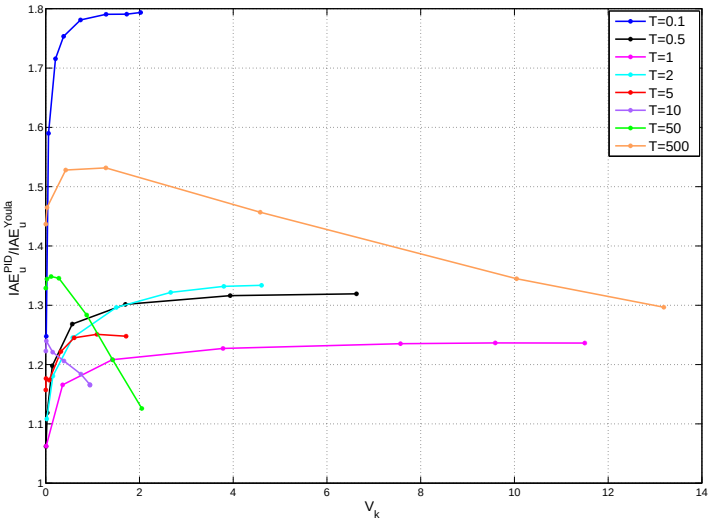

Figure 7.1 Sub-batch 1: The plots show how well PID controllers perform on FOTD processes in terms of IAE_u compared with Youla controllers, for a number of different V_k-values. The curves correspond to different time constants of the FOTD processes.

values than the PID controllers for small V_k. This should mean that the curves for which the relative IAE_u-values decrease give a hint of when the PID design method performs at its worst. These are, however, also cases for which one would presume the PI controller to be the preferred structure. So, simply switching to PI controllers seems like the way to go for lag dominant FOTD processes.

Note that the curves in Figure 7.1 have different max values on V_k. This has to do with the variation in sampling time being used for the 8 processes. The sampling rate will have a slight effect on how these curves look and how high V_k can become (due to the Nyquist frequency). This effect is, however, neglected here.

The curves in Figure 7.1 seems to cross each other several times for different V_k-values. Overall, it seems to be a difficult problem to analyse these behaviours, especially since the PID controllers are not op-

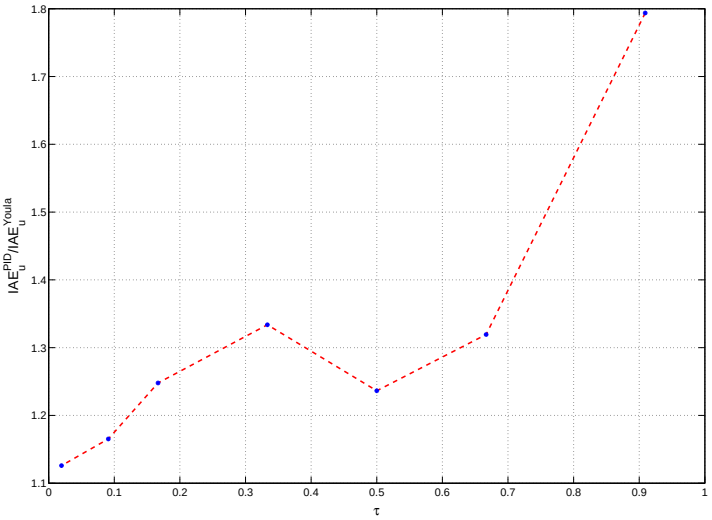

Figure 7.2 Sub-batch 1: Relative IAE_u-values for maximal V_k from the curves in Figure 7.1. These shows a trend that more complex controllers have a bigger advantage for delay dominant FOTD processes.

timal, in contradiction to the Youla controllers. Small variations could therefore be due to this imperfection rather than real differences between the processes. The stationary values (high V_k) in all cases, except $T = 500$, were compared. These values, that are plotted in Figure 7.2, gives a measure of when a PID controller is close in performance to a Youla controller. The benefits from using the advanced controller structure are very small for low values on the normalized time delay, τ. This is quite reasonable since these systems are basically first order systems with negligible time delay. When the time delay is the dominant part of the system, however, the Youla controller shines the most. These controllers will typically include a lot of peaks though, thus ruining robustness towards time delay changes. Also note that there is a dip in the curve, shown in Figure 7.2, for $\tau \approx 0.5$ and a peak when $\tau \approx 0.3$. It would have been logical to assume that this curve is monotonically increasing with τ, with the motivation that the

time delay makes the system harder to control. The time delay robustness will, however, sabotage this relation and one would really need to derive a good time delay compensating controller with sufficient robustness to really get a good picture of whether the advanced structure is important for $\tau \approx 1$ or not. A guess is that the curve, taking delay robustness into consideration would have rather low values for both $\tau \gtrsim 0$ and $\tau \lesssim 1$. Such that there would be some maximum in between when the advanced controller is preferred the most compared to a PI or PID controller. Whether or not this is actually the case, it still seems like the FOTD systems are gaining relatively little from using the advanced structure of the Youla controller. The PI and PID controllers seem to be valid choices to use for these systems. PI for low V_k and τ close to 0 and possibly also close to 1. PID for higher values on V_k and when the system is somewhere in between lag- and delay dominant.

As a complement to the analysis above, consider the case when the variance of the control signal can not be higher than that of the measurement noise, that is when $V_k = 1$. Figure 7.3 and 7.4 shows the Youla and PID controllers respectively for 5 of the 8 different sub-batch processes. Looking at the Youla controllers show how they gradually become less and less complex as T increases. The same is true for the PID controllers that start out with a distinct D-part, but ends with a roughly filtered controller when $T = 500$. Figure 7.5 confirms this observation, showing how the relation T_f/T_d increases with decreasing τ for the PID controllers, thus basically giving PI controllers in the end.

7.2 Sub-batch 2 – Second order systems with time delay

Next in line of analysis is the second sub-batch, containing second order systems with time delay

$$P_2(s) = \frac{e^{-s}}{(1+sT)^2},$$
$$T = 0.1,\ 0.5,\ 1.5,\ 6,\ 20,\ 200,$$

such that 6 of 21 processes were analysed. These processes span from being delay dominant when $T = 0.1$ to become lag dominant, essentially of order two with negligible dead time, when T is high.

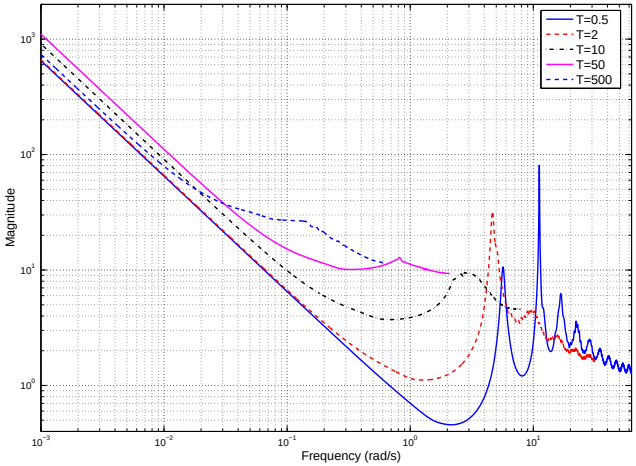

Figure 7.3 Sub-batch 1: Youla parametrized controllers for 5 different FOTD processes, given that $V_k = 1$. Advanced controllers are preferred when the system is delay dominant, while lower order controllers are preferred for lag dominant processes.

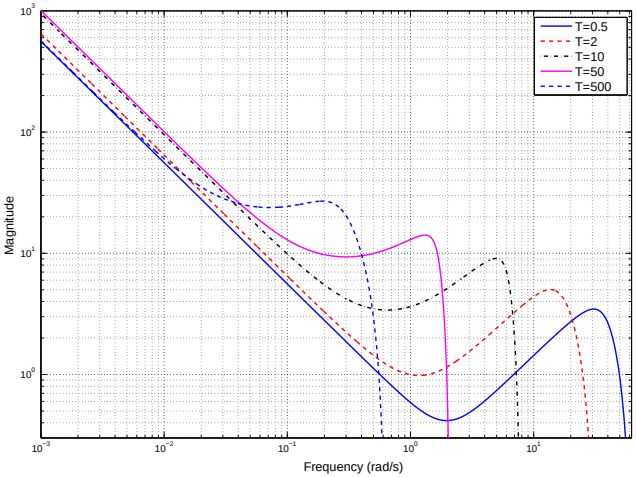

Figure 7.4 Sub-batch 1: PID controllers for 5 different FOTD processes, given that $V_k = 1$.

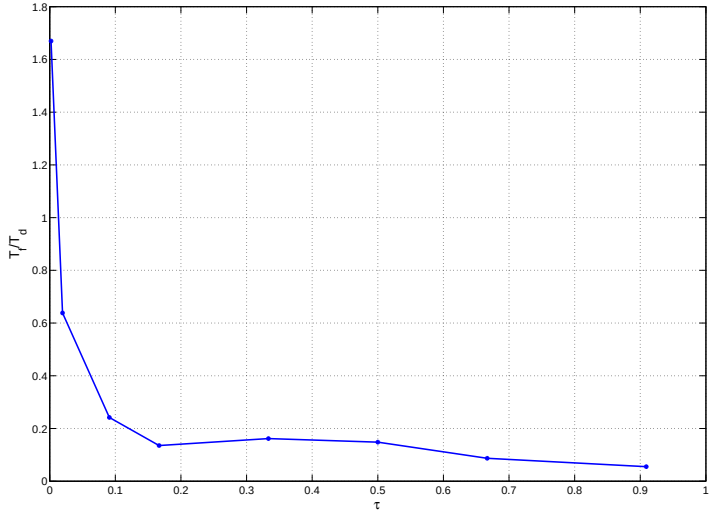

Figure 7.5 Sub-batch 1: The relation between T_f/T_d and τ, when $V_k = 1$ shows that the D-part of the PID controller is the most useful for processes that are not lag dominant.

Figure 7.6 shows once again how $\text{IAE}_u^{PID}/\text{IAE}_u^{Youla}$ changes depending on process dynamics and the limit of control signal variance, i.e. V_k. The patterns are quite similar to those plotted in Figure 7.1, suggesting that the proposed design method is furthest from optimality for lag dominant systems with small V_k-values.

Plotting the values for which V_k is the highest, gives the curve in Figure 7.7. This plot is also very similar to the sub-batch 1 equivalent in Figure 7.2, but sub-batch 2 seems a little bit harder to control for equivalent numbers on τ. The possibly most interesting bit is that the curve dips at $\tau \approx 0.5$ just like before. Whether or not this is an effect caused by the proposed method alone remains an open question for future work.

Fixating V_k to 1 gives the Youla controllers shown in Figure 7.8. The controllers for the delay dominant systems remind a lot of those from sub-batch 1. As τ decreases, the controllers do not seem to become sim-

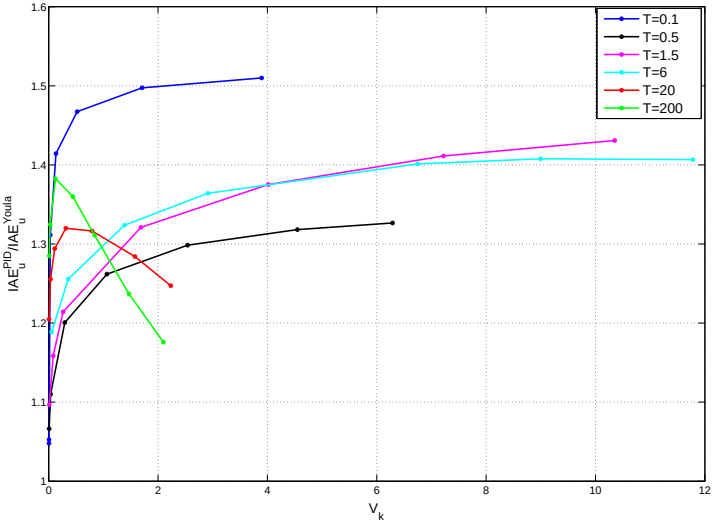

Figure 7.6 Sub-Batch 2: These plots show how well PID controllers perform on second order processes with time delay in terms of IAE_u compared with Youla controllers, for a number of different V_k-values. The curves correspond to different time constants of the processes.

ilar to PI controllers which was the case for the FOTD systems. That is also quite logical since a second order process without time delay is known to be controlled sufficiently well with a PID controller. A PI controller will, however, lack freedom to set the closed loop dynamics. Unless of course one puts really high demands on the control signal variance, thus gimping the potential of the more advanced controllers.

Plotting T_f/T_d versus τ, when $V_k = 1$, confirms that the D-part is more important for the second order processes with dead-time compared to the FOTD systems. The T_f/T_d values are much smaller than before. An interesting detail is that the relation has a maximum around $\tau = 0.2$ and decreases for lower values on τ. If this is an indicator of bad controllers or a real feature of these systems remains to be shown.

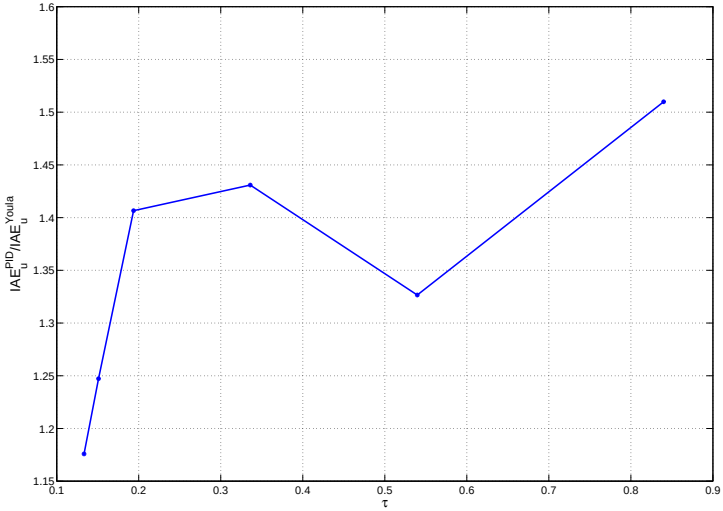

Figure 7.7 Sub-batch 2: Plot that shows the relative IAE_u-values for maximal V_k from the curves in Figure 7.6. Just as in the case of sub-batch 1, the trend is that more complex controllers have a bigger advantage for delay dominant FOTD processes.

7.3 Sub-batch 9 – Systems with complex poles

The third and last sub-batch to be analysed is

$$P_1(s) = \frac{1}{(s+1)((sT)^2 + 1.4sT + 1)},$$
$$T = 0.1,\ 0.4,\ 1,$$

which consists of processes with one real pole and two complex conjugated poles. When $T = 1$, these poles are at the same distance from origin, while the two complex poles are quite insignificant when $T = 0.1$. The system will thus span from basically being a first order system to become a third order process. Looking at the relative IAE differences between the PID and Youla controllers show (see Figure 7.10) that there are more significant differences in performance compared

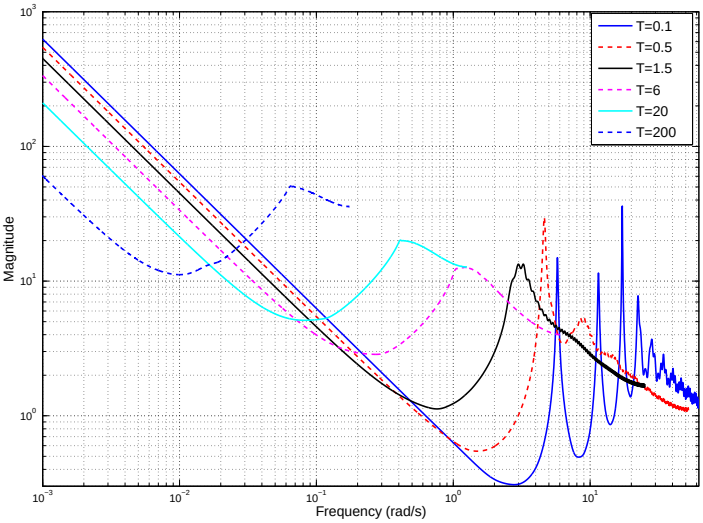

Figure 7.8 Sub-batch 2: Bode magnitude plots of the Youla controllers when $V_k = 1$.

to the previous sub-batches. The τ-values will span a much smaller region than for earlier batches, with: $\tau = 0.12$ when $T = 0.1$, $\tau = 0.26$ for $T = 0.4$ and $\tau = 0.31$ with $T = 1$. It is obvious that the Youla controller is superior when there are three significant poles and V_k is set high. The process with $T = 0.1$ can, however, be controlled quite well with the PID controller although the shape of the curve suggests that the given controllers are not quite optimal.

Taking the case when $V_k = 1$ shows once again that the more delay dominant system (here $T = 1$) have more use for an advanced controller than the lag dominant systems. This is shown in Figure 7.11. Also note that these controllers will give good robustness to time delay variations (no high peaks), so the Youla controllers will, without doubt, be a lot better than PID controllers in this case. It should also be fine to reduce these controllers to receive ones that are really close to optimal.

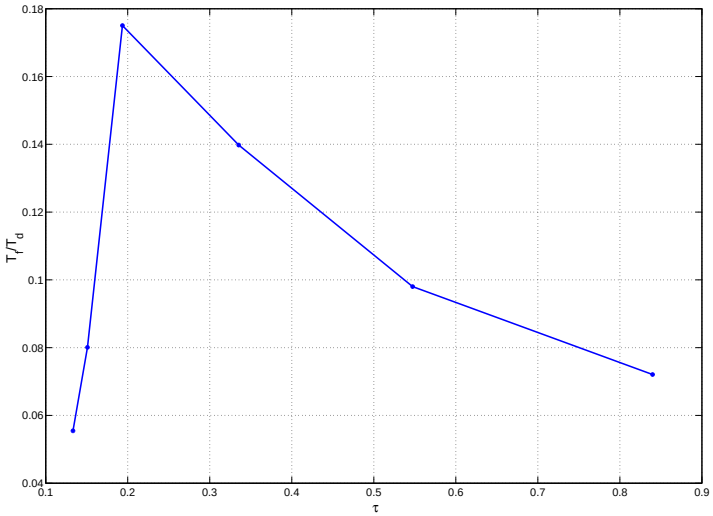

Figure 7.9 Sub-batch 2: The relation between T_f/T_d and the normalized time delay when $V_k = 1$. The D-part is more significant than for the processes in sub-batch 1.

The T_f/T_d versus τ relation, for the PID controllers when $V_k = 1$, is strictly decreasing in this case. For $T = 1$ one gets $T_f/T_d = 0.17$, while $T = 0.4$ lead to $T_f/T_d = 0.19$ and $T = 0.1$ to $T_f/T_d = 0.37$. This is, however, just as expected. The process that is the closest to a first order system being the process that needs the D-part the least.

7.4 Summary

Looking through the different plots presented in this chapter, it is obvious that whether or not a PI or PID controller is sufficient will depend on both the process dynamics and the allowed control signal activity. If the control signal variance due to measurement noise is allowed to be large, then the freedom of an advanced controller will come to its right to a higher degree. If the system is simple, however, it will generally

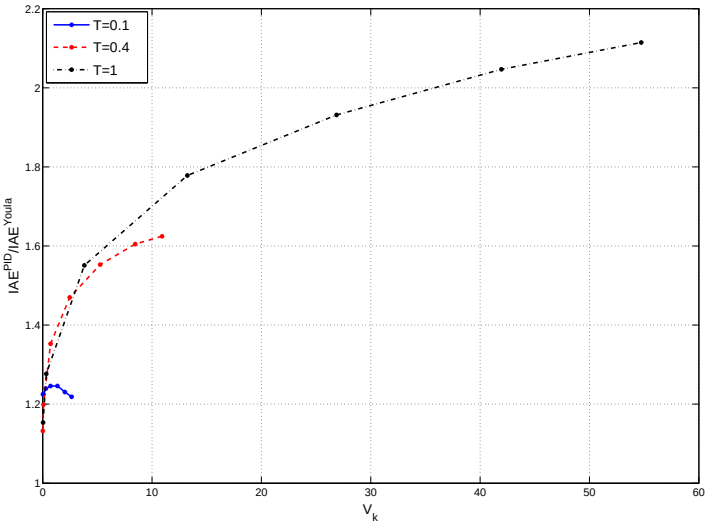

Figure 7.10 Sub-Batch 9: The plots show how well PID controllers perform on third order processes, with complex poles, in terms of IAE_u compared with Youla controllers, for a number of different V_k-values.

not be any point in using the advanced controller and a PID or even PI is close to optimum. If the process is delay dominant, the advanced controllers may seem to give really good results too, but one will then have to take into account that the given optimization problem does not handle robustness towards time delay changes. Such robustness measures could potentially mean that the most delay dominant processes should really be controlled by simple controllers too.

As stated in the beginning of this chapter, these results were derived in the end of the making of this thesis. The results should be seen as a source of inspiration for future work rather than a presentation of the absolute truth. In order to get the full picture, one would really need to derive the PI and PID controllers that are optimal with respect to the variance constraint. The results derived here should, however, at least be representative for the proposed design method.

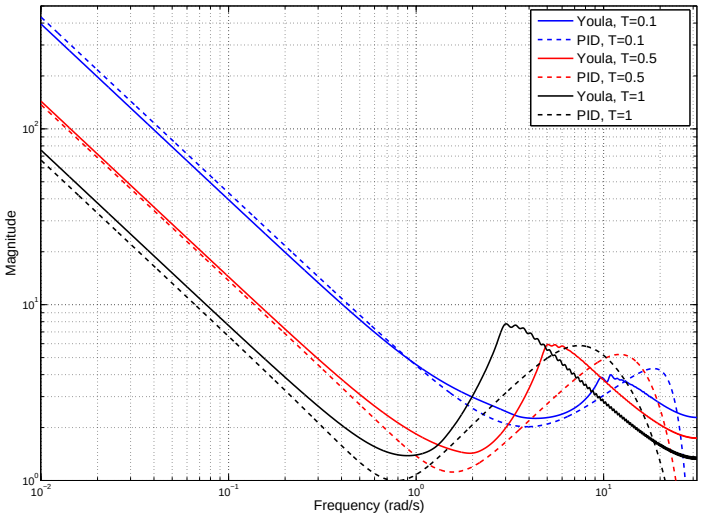

Figure 7.11 Sub-Batch 9: Bode magnitude plots for the Youla and PID controllers when V_k is fixated to 1. The PID controller is the most similar to the Youla controller when $T = 0.1$, i.e. when the process is essentially a first order system.

8

Conclusions and Future Work

The following chapter will give a brief summary of the most important contributions of the thesis and present some suggestions for future work within the research area.

8.1 Conclusions

Many industrial processes have relatively simple dynamics, that are generally easy to control with PI or PID controllers. Even so, the sheer number of control loops makes it a very time-consuming duty to tune all of them. Lack of time and knowledge among operators will thus often leave controllers with default settings, far from optimal. For this reason it is important to have a controller design tool that is both fast and easy to use. While there are many design methods already satisfying this, few guarantee to fulfill all of the following three closed loop criterias:

- Fast load disturbance suppression
- Closed loop robustness to model errors and process variations
- Low actuator wear and tear

While the third criteria is seldom an issue with PI control, it is extremely important when designing PID controllers. Control signal sensitivity to measurement noise is one of the most important reasons why

the D-part is seldomly used in industry today. It is simply too costly to exchange actuators in relation to the performance gain one gets through using the D-part. Add to this that tuning three, or even four, controller parameters is far more complicated than just two and it is not difficult to understand the decision.

The PID design method proposed within this thesis takes all three of the above stated criterias into consideration. The controllers are optimized through a fast and robust Matlab software tool such that the IAE-value during a load disturbance is minimized, with respect to robustness constraints on the sensitivity- and complementary sensitivity function. The tool can then be run several times for different settings on the low-pass filter, through which the measurements have to pass. Estimating the control signal variance due to measurement noise will provide the user with a trade-off curve between performance and control signal activity. This way, the wear on actuators can be taken into consideration when designing the controllers. Furthermore, the design tool can be used to choose a PI or PID controller depending on which is the best suited for the given process. This will be a decision that is taken both with respect to the process and the allowed control signal activity.

It is a good idea to check the quality of the controllers since the proposed software design tool give optimal controllers with respect to robustness, but not to the control activity constraint. In order to do so, optimized Youla parametrized controllers were used. These give a good estimate of how close the designed PI and PID controllers are to the limit of performance. A small difference in performance is a very good sign that the simple controllers are well-tuned. Several examples provided within this thesis have also shown that this is very often true. In the cases that PID controllers are less suitable, it is often advisable to switch to PI control. This is also supported by the observation that the Youla controllers are basically PI controllers in these cases. There are of course other types of processes, like oscillatory systems, when PID control is known to give rather poor results. In such cases, it seems advisable to use more advanced controllers. One way of doing so could be to start out from the Youla controllers and proceed through some controller reduction method. It is, however, important that extra time spent on deriving more advanced controllers is really worth it, or else a simpler controller may very well be better to use.

Several experiments on real processes have shown that the proposed design method has good potential to work well in industry. It is possible to estimate the control signal variance due to measurement noise and keep it as low as specified, which opens up for use of PID controllers in industry.

8.2 Future work

During the work of this thesis, several questions have come up that will need further attention in the future. Here follows a few of them:

- How can one extend the controller design to include fully automatized estimation of the control signal variance and plotting of the trade-off curve?
- Is it possible to use Youla controllers to systematically tune even better controllers whose level of complexity depends on the process dynamics and allowed control signal activity?
- The proposed controller design tool should ideally be able to choose optimal controller structure depending on the process. Is this possible to achieve?

In addition to these questions, it should be reminded that the issues presented in Chapter 7 are not yet fully investigated. More work is needed before it is possible to give any general guidelines on when PI and PID controllers are valid choices compared to more advanced options. Furthermore, the PI and PID controllers used in the comparisons should ideally be optimal with respect to the variance constraint. This could for instance be achieved through gridding.

Experiments on an industrial process has already been conducted, but several more tests are needed before the proposed design method can be utilized with full confidence.

There are also several nearby areas like event-based PID control and control of two-inputs-two-outputs systems that could possibly be treated similarly to the methods described in this thesis. Whether or not these ideas have potential will also be left for future work to show.

9

Bibliography

Åkesson, J. (2008): "Optimica—An Extension of Modelica Supporting Dynamic Optimization." In *6th International Modelica Conference 2008*. Modelica Association.

Årzén, K.-E. (1996): *Real-Time Control Systems*. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Åström, K. J. and T. Hägglund (2001): "The future of PID control." *Control Engineering Practice*, **9**, pp. 1163–1175.

Åström, K. J. and T. Hägglund (2005): *Advanced PID Control*. ISA - The Instrumentation, Systems, and Automation Society, Research Triangle Park, NC 27709.

Åström, K. J. and B. Wittenmark (1997): *Computer Controlled Systems—Theory and Design*. Prentice Hall, Upper Saddle River, New Jersey.

Boyd, S., C. Barratt, and S. Norman (1990): "Linear Controller Design: Limits of Performance Via Convex Optimization." *Proceedings of the IEEE*, **78:6**, pp. 529–574.

Boyd, S. P. and C. Barratt (1992): "Closed-loop Convex Analysis of Performance Limits for Linear Control Systems." In *IEEE Symposium on Computer Aided Control System Design*, pp. 301–304. Napa, California, USA.

Boyd, S. P. and C. H. Barratt (1991): *Linear Controller Design—Limits of Performance*. Prentice Hall, Englewood Cliffs, New Jersey.

Buckbee, G. (2002): "Poor Controller Tuning Drives Up Valve Costs." *Control Magazine*, April.

Dahlin, E. B. (1968): "Designing and tuning digital controllers." *Instruments and Control Systems*, **42**, June, pp. 77–83.

Garpinger, O. and T. Hägglund (2008): "A Software Tool for Robust PID Design." In *IFAC World Conference*. Seoul, South Korea.

Hägglund, T. and K. J. Åström (1991): "Industrial adaptive controllers based on frequency response techniques." *Automatica*, **27**, pp. 599–609.

Hägglund, T. and K. J. Åström (2004): "Revisiting the Ziegler-Nichols step response method for PID control." *Journal of Process Control*, **14:6**, pp. 635–650.

Isaksson, A. J. and S. F. Graebe (2002): "Derivative filter is an integral part of PID design." *Control Theory and Applications, IEE Proceedings*, **149**, January, pp. 41–45.

Kristiansson, B. and B. Lennartson (2006): "Evaluation and simple tuning of PID controllers with high-frequency robustness." *Journal of Process Control*, **16**, pp. 91–102.

Lagarias, J. C., J. A. Reeds, M. H. Wright, and P. E. Wright (1998): "Convergence properties of the Nelder-Mead simplex algorithm in low dimensions." *SIAM Journal on Optimization*, **9**, pp. 112–147.

Li, Y., K. H. Ang, and G. C. Y. Chong (2006): "PID control system analysis and design - Problems, remedies, and future directions." *IEEE Control Systems Magazine*, **26**, pp. 32–41.

Luyben, W. L. (2001): "Effect of derivative algorithm and tuning selection on the PID control of dead-time processes." *Ind. Eng. Chem. Res.*, **40**, pp. 3605–3611.

Marlin, T. E. (1995): *Process Control. Designing Processes and Control Systems for Dynamic Processes*. McGraw-Hill.

Nelder, J. A. and R. Mead (1965): "A simplex method for function minimization." *Computer Journal*, **7**, pp. 308–313.

Nordfeldt, P. (2005): “PID Control of TITO Systems.” Licentiate Thesis ISRN LUTFD2/TFRT--3238--SE. Department of Automatic Control, Lund University, Sweden.

Norman, S. A. and S. P. Boyd (1989): “Numerical solution of a two-disk problem.” In *American Control Conference*, pp. 1745–1747. Pittsburgh.

Olsen, T. and B. Bialkowski (2002): “Lambda Tuning as a Promising Controller Tuning Method for the Refinery.” In *2002 AIChE Spring National Meeting*. New Orleans, USA.

Oviedo, J. J. E., T. Boelen, and P. van Overschee (2006): “Robust advanced PID control (RaPID) - PID tuning based on engineering specifications.” *IEEE Control Systems Magazine*, **26**, pp. 15–19.

Panagopoulos, H., K. J. Åström, and T. Hägglund (2002): “Design of PID controllers based on constrained optimisation.” *IEE Proceedings - Control Theory & Applications*, **149:1**, pp. 32–40.

Wallén, A. (2000): *Tools for Autonomous Process Control*. PhD thesis ISRN LUTFD2/TFRT--1058--SE, Department of Automatic Control, Lund Institute of Technology, Sweden.

Walters, F. H., L. R. Parker Jr, S. L. Morgan, and S. N. Deming (1991): *Sequential Simplex Optimization*. CRC Press LLC.

Wernrud, A. (2008): *QTool 0.1—Reference Manual*. Department of Automatic Control, Lund University, Lund, Sweden.

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

Document name
LICENTIATE THESIS

Date of issue
March 2009

Document Number
ISRN LUTFD2/TFRT--3245--SE

Author(s)
Olof Garpinger

Supervisor
Tore Hägglund

Sponsoring organisation
Swedish Research Council (VR)

Title and subtitle
Design of Robust PID Controllers with Constrained Control Signal Activity

Abstract

This thesis presents a new method for design of PI and PID controllers with the level of control signal activity taken into consideration. The main reason why the D-part is often disabled in industrial control loops is because it leads to control signal sensitivity of measurement noise. A frequently varying control signal with too high amplitude will very likely lead to actuator wear and tear. For this reason it is extremely important for any PID design method to take this into account.

The proposed controllers are derived using a newly developed design software that solves an IAE minimization problem with respect to H_∞ robustness constraints on the sensitivity- and complementary sensitivity function. The software is shown to be fast, easy to use and robust in giving well-performing controllers.

By extracting measurement noise from the process value of a real plant, one can estimate its effect on the control signal variance. The time constant of the low-pass filter, through which measurements are fed, is varied to design controllers with constrained control signal activity. By comparing control signal variance and IAE, the user is also able to weigh actuator wear to estimated performance.

The proposed PID design method has shown to give very promising results both on simulated examples and real plants such as a recirculation flow process.

Optimal Youla parametrized controllers are used both as a quality check of the designed PI and PID controllers and as a tool for determining when these are valid choices compared to more advanced controllers.

Key words
PID control, process control, disturbance rejection, linear systems, constrained control signal activity, measurement noise, algorithms and software, Youla parameterized controllers

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
122

Recipient's notes

Security classification