ISSN 0280–5316
ISRN LUTFD2/TFRT--5621--SE

Kappa Tuning– Improved relay auto-tuning for PID controllers

Albert Norberg

Department of Automatic Control
Lund Institute of Technology
June 1999

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

Document name
MASTER THESIS

Date of issue
June 1999

Document Number
ISRN LUTFD2/TFRT--5621--SE

Author(s)
Albert Norberg

Supervisor
Lars Pernebo, ABB Satt AB
Tore Hägglund

Sponsoring organisation

Title and subtitle
Kappa Tuning – Improved relay auto-tuning for PID controllers

Abstract

In this report a method for tuning PID controllers is developed. It is based on old ideas of extending a Ziegler-Nichols type of tuning. It is made for the frequency method, but it is shown how the step response method could be extended as well. The extension is to use a third process parameter kappa, in addition to the ultimate point parameters Ku and Tu. It is shown that good tuning rules can be empirically found through simple parameterisations in Ku, Tu, kappa and the PID parameters. The PID parameters tuned are proportional gain K, integral time Ti, derivative time Td, and the setpoint weighting factor b. For integrating processes a new definition of the kappa parameter is suggested. It is shown that this kappa definition can be used to formulate similar tuning rules for integrating plants. The adequacy of the proposed kappa definition is also discussed in a minor theoretical contemplation. A part in the report also treats suitable identification to gain the kappa parameter. A relay experiment is used to identify the ultimate point and only a simple closed loop step is done in addition. However, in this report we also use the closed loop step to estimate two extra parameters, apparent lag T, and apparent dead time L. These parameters are during tuning used to evaluate the need for a dead time compensating controller, and also for the tuning of such.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
67

Recipient's notes

Security classification

Contents

1. Introduction

Tuning of PID controllers is an extensively explored area within the control community. Many tuning methods have been suggested throughout the years and new methods arrive frequently. The purpose with a specific tuning method differs. Some are simple heuristic rules intended for practical use on real plants, while others are highly theoretical methods applying to mathematical models. A practical tuning method for PID controllers is a manual adjustment by observations and simple engineering rules, and this possibility is an often heard argument for the use of PID control. For the practical use however, automatic tuning procedures are used to a large extent. With an automatic tuning we mean an adaptive tuning mechanism, ordered by an operator, that is made under a limited period of time and gives the controller parameters as result. Automatic tuning facilities exist in almost all commercial PID controllers today. The arguments for a manual tuning possibility still hold though, since it can be used to refine the results from the auto-tuner.

All automatic tuners have to be based on some fundamental controller design method. For an auto-tuner to be successful it is not enough with a good design method. Other aspects have to be considered as well. The automatic tuning also involves an identification part of equal importance. Many aspects of more practical nature also exist. One example is the amount of prior, or user supplied, information that an auto-tuner requires. This should be kept at a minimum to suit operators without advanced knowledge in control. Another practical issue is how long time an auto-tuning takes. During the tuning experiment the process must be kept undisturbed and this can be hard to achieve in many applications. Therefore it is important that a tuning experiment can be done under a short period of time.

In this report the efforts of developing a new auto-tuning procedure is presented. It will be based on a relay based automatic tuning, and it can be seen as an extension of this method. In our new auto-tuner the controller design part will be different, and the development of the design method is also presented in the report. This design, or equally, tuning method is an old idea of extending a Ziegler-Nichols type of tuning with an additional process parameter. This parameter kappa, has properties that makes it appealing for tuning, and we will investigate it closely in the report. The process information needed will be gained from a relay experiment, and in addition a simple step experiment to gain kappa. We will refer to the total procedure; tuning a controller from the relay experiment data and the kappa parameter, as kappa-tuning. The notation of auto-tuning will not be used very much in the sequel, although the kappa-tuning indeed can be seen as an auto-tuning procedure. The report will mainly be focused on the controller design part of the tuning. As discussed above there are other aspects as well, but they will only be treated shortly in the last subsection of chapter five.

The identification experiment that has to be done except for the relay experiment will be treated in a separate chapter. We will use a closed loop step experiment to estimate the kappa parameter. We will also show how this experiment can be used to extract some extra process information. This information will be used to give guidance of the dead time properties in the process, and if large dead times occur a special dead time compensating controller will be suggested. The estimated parameters can in these cases be used to tune this controller, called a PPI controller.

Integrating processes are often treated separately in tuning methods, and this will be done here as well. The difficulty with the integrating processes is that they all have kappa equal zero, and thereby we lose one tuning parameter. However, it is possible to use a different definition of kappa for integrating processes, to still be able to tune them with three parameters. We will in this report suggest a new definition of kappa that has good properties for tuning controller for integrating processes.

The thesis outline will be as follows: In chapter one we will recapitulate some general control theory. Most importantly, we will here take a closer look at the relay auto-tuner. In chapter three we treat the main subject in the report, the kappa-tuning. In this chapter we define kappa, derive the design rules and describe the tuning procedure. We also treat integrating processes here. In chapter four we compare the kappa-tuning to three other tuning methods. One of them is the relay auto-tuner. In chapter five we treat the additional identification experiment in detail. After this chapter we are also ready to describe a total auto-tuning procedure using the kappa-tuning, and the last subsection in chapter five is devoted to this. The last chapter contains some conclusions and further improvements.

2. General principles

This chapter will mainly be concerned with control theory adequate for this thesis, and it is all somewhat related to control within the processing industry. Nothing new is presented and with good control knowledge a skim through might be enough. The first topic is the process dynamics found in the area of processing and the question of how to model it. Next the PID controller's basic structure is presented along with some extensions of the schoolbook algorithm. The PPI controller, a controller suitable when long dead times are present, is also described. The chapter also includes a look at automatic tuning using a relay experiment, a method used today in many commercial products. This method is also a main component of the kappa-tuning, and is therefore of interest for this work.

2.1 Process models and plant dynamics

From the control engineering's point of view a mathematical model of the system to be controlled is very interesting. Many design methods for controllers are model based, and a good model is often required for good control. The work presented here is not concerned with good modelling of single plants, but too some extent a more general process model is used.

Before moving on we must specify the class of systems that will be our objective. In the processing industry many types of control problems can arise but typical examples are temperature-, concentration-, pressure-, level- or flow-control. This type of dynamic can mostly be described as having low pass character and being well damped. Oscillatory systems are uncommon, but can occur for example in mechanical parts. Those will not be dealt with in this work. Another property sometimes heard is "S-shaped" step response. For the present this vague description will define the processes encountered.

With the type of processes mentioned above a common model used is

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

The three parameters are static gain K_p, time constant T, and dead time L. Another useful parameter is the average residence time T_{ar}, and for the model above $T_{ar} = L + T$. From a step response the three parameters can be identified with various methods. Despite its simplicity this model can capture much of process behaviour quite well. A step response can be seen in Figure 1.

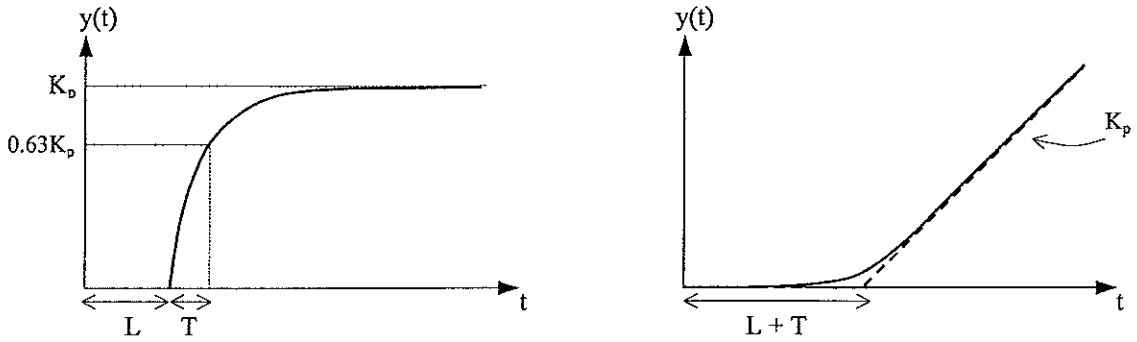

Figure 1: Step responses of model (1) to the left, and the integrating process models (2) (dashed) and (3) to the right.

The model above is not well suited for integrating processes and for these, other models should be used. Two often suggested models are

$$G_{INT}(s) = \frac{a}{sL'} e^{-sL'} \tag{2}$$

$$G_{INT}(s) = \frac{1}{s} \frac{K_p}{1+sT} e^{-sL} \tag{3}$$

The step response of model (2) is simply a delay L', and a velocity gain a/L'. The model (3) has a final velocity gain K_p, and the tangent with slope K_p intersects the time axis at t = L + T. See Figure 1. Approximating model (3) with (2) would then give

$$\begin{aligned} a &= K_p L' = K_p (L+T) \\ L' &= L+T \end{aligned} \tag{4}$$

In section 3.5 we will use these simple relations.

2.2 The PID controller

The PID controller has shown to be a very useful structure of a controller for many applications, not due to a superior control action, but rather to its simplicity and user-friendliness. This is of course the reason why many manufactures of control systems mainly supply PID controllers, and also why researchers world over still are concerned with this old idea. The basic structure is often formulated as

$$u = K(e + \frac{1}{T_i} \int e \cdot dt + T_d \cdot \frac{de}{dt}) \tag{5}$$

or Laplace transformed

$$U = K(1 + \frac{1}{sT_i} + sT_d) \cdot E$$

where u is the control signal and e is the feedback control error, i.e. the difference between the setpoint and the measured output. This form is known as noninteracting form, where the first term is referred to as proportional part (P-part), the second term as the integral part (I-part) and the last term as the derivative part (D-part). The three parameters are called proportional gain K, integral time T_i, and derivative time T_d. Care must be taken though since some other forms are common in literature on PID controllers. One of them is the interacting form, which looks like

$$U = K'(1 + \frac{1}{sT_i'})(1 + sT_d') \cdot E$$

and here the three parameters have a slightly different interpretation. What they all have in common though is that the parameters have the same qualitative meaning, and it is known among users of PID controllers how changes in the three parameters affects the control behaviour. Herein lies much of the strength in the PID controller.

When implementing a PID controller some modifications of the above algorithms are often made. First, a pure derivative part is not realistic since it gives very high gain to rapid signal changes, i.e. high frequency signal. That would amplify high frequency noise significantly, and the derivative part is therefore filtered with a low pass filter. The filter time constant T_f, is often expressed as a fraction of the derivative time T_d, like $T_f = T_d/N$. Further it is desirable to act not only on the error, but the setpoint signal and the measured output separately. This two-degree of freedom is used to weight the response to the setpoint signal since changes here can be very drastic. The control law now looks like

$$U = K \cdot (bY_{sp} - Y + \frac{1}{sT_i} E - \frac{sT_d}{1 + sT_d / N} Y) \qquad (6)$$

Here the setpoint weighting on the proportional part is called b, and on the derivative part it is set to zero, which is a common choice. b is normally restricted between zero and one.

As almost all controllers today are implemented digitally the control law must be discrete, and continuous time controllers, like the PID controller, must be approximated. This is often done by replacing a derivation with a difference quotient. Taking the difference one sample ahead is called a forward difference while taking it between previous and present sampled data is called a backward difference. More sophisticated derivative approximations exist, but for the PID controller a simple but common choice is to approximate the integral part with a forward difference and the derivative part with a backward difference. The discrete version of the control law (6) then becomes

$$\begin{aligned} u(k) &= P(k) + I(k) + D(k) \\ P(k) &= K \cdot (by_{sp}(k) - y(k)) \\ I(k) &= I(k-1) + \frac{Kh}{T_i} e(k-1) \qquad (7) \\ D(k) &= \frac{T_d}{T_d + Nh} \cdot D(k-1) - \frac{KT_d N}{T_d + Nh} \big(y(k) - y(k-1)\big) \end{aligned}$$

where k is the time expressed in number of samples, and h the sampling period.

2.3 The PPI controller

When the process contains a long dead time the control performance obtained with a PID controller is limited. A predictive controller must be used for improved control action. They are often referred to as dead time compensating controllers, and a common type is the Smith predictor. Here the controller contains an internal model of the process, and to the feedback signal the output from the model is subtracted and the output from the model without dead time is added. The feedback signal can then be interpreted as a prediction of the process output one dead time ahead. Ideally, in the situation of perfect modelling, the controller will act on a simulated process that behaves as the real process without dead time. In Figure 2 a Smith predictor using the internal model (1) combined with a PI controller is shown. It is called a PPI controller (Predictive PI controller).

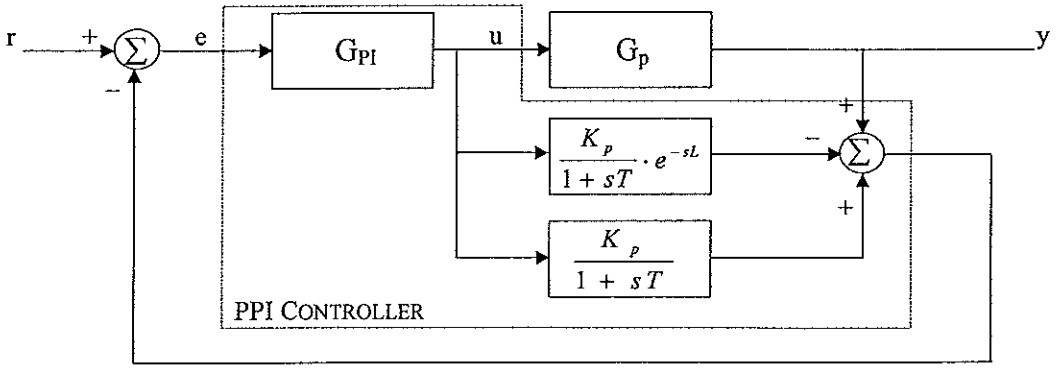

Figure 2: The PPI controller. A Smith predictor with PI control and internal model (1).

The PPI controller contains the five parameters K, T_i, K_p, T and L. This can be reduced to three by expressing the PI parameters K and T_i in the parameters of the model. This was done in [2], and it is suggested here that a reasonable choice is

$$\begin{aligned} K &= 1/K_p \\ T_i &= T \end{aligned} \tag{8}$$

This lead to a control law (in the time domain with derivative operator p)

$$u(t) = K\left(1 + \frac{1}{pT_i}\right) \cdot e(t) - \frac{1}{pT_i}\left(u(t) - u(t-L)\right) \tag{9}$$

where e is the control error $e(t) = y_{sp}(t) - y(t)$. With the parameter choice (8) this can be rewritten as

$$u(t) = K\left(1 + \frac{1}{pT_i}\right) \cdot \left(e(t) - \frac{K_p}{1+pT}\left(u(t) - u(t-L)\right)\right)$$

and the relation to Figure 2 is easier seen. The last parenthesis correspond to the error feedback in Figure 2 and it is a prediction of the control error at time t + L. The control law can thus be interpreted as a PI controller acting on the control error $\hat{e}(t + L)$.

Another type of dead time compensating controller is obtained with the method called λ-tuning, see [1]. It is interesting to compare the λ-tuning to the PPI controller above, and it can be evidenced from [1] pp. 157, that we obtain the same control law as (9) for $\lambda = 1$.

To compare different controllers the integrated absolute error, IAE, under a step load is a sometimes used quantity. A low value indicates fast rejection of load disturbances. The unit step load is applied to the input of the process, and if simple error feedback is used the transfer function and error is

$$G_{l\to e} = \frac{G_p}{1 + G_c G_p} \qquad \Rightarrow \qquad E(s) = \frac{G_p}{1 + G_c G_p} \cdot \frac{1}{s}$$

Assuming that the controller tuning is such that the closed loop system is critically damped, the integrated absolute error IAE, equals the integrated error IE. Using the final value theorem we find

$$IAE = \lim_{t\to\infty} \int e \cdot dt = \lim_{s\to 0} s \cdot \frac{1}{s} E(s) = \lim_{s\to 0} \frac{G_p}{1 + G_c G_p} \cdot \frac{1}{s} = \lim_{s\to 0} \frac{1}{sG_c}$$

where we assumed that $s/G_p(s)$ goes to zero as s goes to zero. With this relation we can calculate the IAE for a PID and a PPI controller. With the transfer functions obtained from (6) and (9) we get

$$IAE_{PID} = \lim_{s\to 0} \frac{1}{s} \cdot \frac{\frac{T_d}{N} s^2 + s}{(KT_d + \frac{KT_d}{N})s^2 + (K + \frac{KT_d}{T_i N})s + \frac{K}{T_i}} = \frac{T_i}{K} \tag{10}$$

$$IAE_{PPI} = \lim_{s\to 0} \frac{1}{s} \cdot \frac{1 - e^{-sL} + sT_i}{KT_i s^2 + Ks} = \frac{T_i + L}{K} \tag{11}$$

In the last equality we have used that $(1-e^{-sL})/s$ goes to L as s goes to zero. These relations will be used later in chapter 3.7 to determine if a PPI controller is expected to give better performance than a PID controller.

To be implemented digitally, the control law (9) must be approximated. This can be done in much the same way as the PID controller, by approximating derivatives with difference quotients. In the PPI algorithm though, the control signal u(t – L) will appear, and some kind of buffer must be used to store all value of u between t - L and t. The discrete form also lead to that the dead time L, used in the controller, only is allowed to be a multiple of the sampling period.

2.4 The relay auto-tuner

The key idea behind the relay auto-tuning is relay feedback. By connecting a process with a relay in a simple feedback loop, oscillations can occur. From these limit cycles, process information can be extracted and used for tuning. A relay experiment identifies the point on the Nyquist curve with phase lag 180°, i.e. the ultimate point. The frequency at this point, called the ultimate frequency, corresponds to the frequency of the limit cycles. Further, the process gain at this frequency is proportional to the ratio between the limit cycles amplitude and the relay amplitude. This only holds approximately but ideally the parameters ultimate gain, K_u and ultimate period time, T_u are gained. Ultimate gain is defined as the inverse of the process gain at the ultimate point. Not all processes will exhibit limit cycles, but most of the processes we are dealing with do. Based on the ultimate point, a reasonable design of a PID controller can be made. Below a design procedure found in for example [2], is presented. This special method will also be the one that is used in the kappa-tuning, and it is referred to as "the relay auto-tuner".

From a practical point of view, to avoid random relay switching on noisy signals, the relay must have hysteresis. This will move the point that is identified towards a lower frequency. From describing function analyses (see [10]) we can predict the limit cycle to a point on the Nyquist curve where it intersects $-1/N(a)$. N(a) is the describing function, and for the relay with hysteresis it is a complex valued function of the oscillation amplitude a, see Figure 3. The identified point can now be characterised by its length $1/K_u$, its frequency ω_u and its phase α (see Figure 3). This with some abuse of the notation of "ultimate", since it is no longer the ultimate point. From the expression for the describing function (relay amplitude d, and relay hysteresis ε)

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

it is found that

$$\left|G(i\omega_u)\right|=\frac{1}{K_u}=\frac{\pi a}{4d} \qquad \alpha=\arcsin\left(\frac{\varepsilon}{a}\right) \qquad (12)\ \text{a,b}$$

and the frequency ω_u is gained directly from the frequency of the limit cycles.

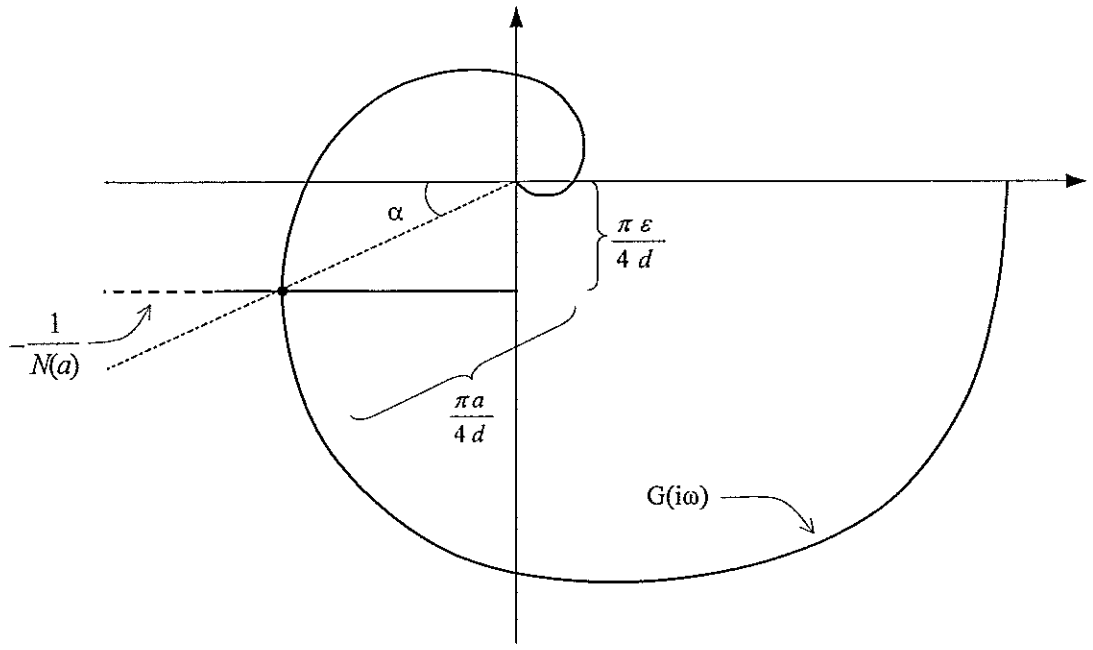

Figure 3: The Nyquist curve of the process intersecting the describing function for a relay with hysteresis.

The design procedure in the relay auto-tuner is to move the identified point to a point with length 0.5 and phase -135°, and this can be viewed as giving the closed loop system a certain amplitude- and phase-margin. It is accomplished by adjusting the PID parameters K, T_i and T_d, but since the criterion only specify two parameters, it is further required that $T_i = 6.25T_d$.

In equation (12) b it is seen that the identified point change between α = 0 - 90°, depending on the ratio relay hysteresis to oscillation amplitude. In the relay auto-tuner the hysteresis is fix but the amplitude of the relay is adjusted during the experiment to give suitable amplitude of the oscillation. The oscillation amplitude should be kept at a minimum but distinguished from noise. Therefore the noise level is detected, a hysteresis is chosen well above this level, and then the desired oscillation amplitude is set as twice the hysteresis. This will, according to (12) b, give a nominal angle $\alpha_{nom} = 30°$.

For further use in chapter 3 we shall derive how the controller parameters are calculated from the identified parameters, $K_{u\alpha}$ and $T_{u\alpha}$. The notation uα indicates that it is not the ultimate point, but a point at angle α. We assume that the identification is made at $\alpha_{nom} = 30°$. Using the basic form (5), of PID structure we find

$$G_{PID}(i\omega_{u\alpha}) = K(1 + \frac{1}{\omega_{u\alpha}T_i} + i\omega_{u\alpha}T_d) = K\left(1 + i\left(\frac{(\omega_{u\alpha}T_d)^2 - 0.16}{\omega_{u\alpha}T_d}\right)\right)$$

where the last equality follows from $T_i = 6.25T_d$. Moving the identified point to the specified location implies

$$G_p(i\omega_{u\alpha})G_{PID}(i\omega_{u\alpha}) = 0.5 \cdot e^{-i135°}$$

This means that the controller must phase advance 15° and amplify $0.5K_{u\alpha}$ at the identified point, giving the two equations

$$\frac{(\omega_{u\alpha}T_d)^2 - 0.16}{\omega_{u\alpha}T_d} = \tan 15^\circ$$

$$K\sqrt{1^2 + \tan^2 15^\circ} = 0.5K_{u\alpha}$$

By using $\omega_{u\alpha} = 2\pi/T_{u\alpha}$ and solving the equations we get

$$\frac{K}{K_{u\alpha}} = 0.48$$

$$\frac{T_i}{T_{u\alpha}} = 0.55 \qquad (13)$$

$$\frac{T_d}{T_{u\alpha}} = 0.088$$

If instead assuming that the real ultimate point is identified, i.e. $\alpha = 0°$, the controller must phase advance 45° and as above we get

$$\frac{K}{K_u} = 0.35$$

$$\frac{T_i}{T_u} = 1.13 \qquad (14)$$

$$\frac{T_d}{T_u} = 0.18$$

3. Kappa-tuning

In this chapter we treat the main objective of the report. After a definition and some properties of the kappa parameter, subsection two deals with how kappa can be used for tuning. In section three the derivation of the tuning rules is made, and section four describes the tuning procedure. Integrating processes has to be treated separately, and section five is about tuning them with a modified method. A similar tuning method using another process parameter tau, is presented next in section six. In the last section some extra tuning possibilities are discussed e.g. the choice of a PI controller instead of PID.

3.1 Kappa, a process parameter

For a process with transfer function $G_p(s)$ the parameter kappa, or the gain ratio, is defined as

$$\kappa = \left| \frac{G_p(i\omega_u)}{G_p(0)} \right| = \frac{1}{K_p K_u} \tag{15}$$

i.e. the ratio of the gain at phase $-180°$ and the static gain. For the specified type of processes, see 2.1, having low pass character etc, kappa lies in the interval 0 to 1. Kappa was introduced in [4] and has shown to be a good measurement of process-behaviour from the controller's point of view. In some sense kappa indicates the difficulty to control the process, and a kappa value closer to one means a more difficult process to control. One way to see this is by adding dead time to the process, thereby making it harder to control. This will widen the Nyquist curve towards a circle, and kappa will increase (see Figure 4). By increasing the system order we can also get a more difficult process, and for linear systems, without dead time, kappa is related to the system order. For a first- or second-order system kappa is zero, and in most cases kappa increases with increasing system order. For example, the systems with transfer functions $G(s) = 1/(s+1)^n$ have $\kappa = 0.125$ for $n = 3$, $\kappa = 0.25$ for $n = 4$ and $\kappa = 0.53$ for $n = 8$. The exact meaning above of a process being "difficult to control" is hard to give, but it refers to the control performance that can be achieved. For example, a difficult process might have to be controlled with a slower controller giving the closed loop system a larger time constant than a easy process although they have the same open loop time constant.

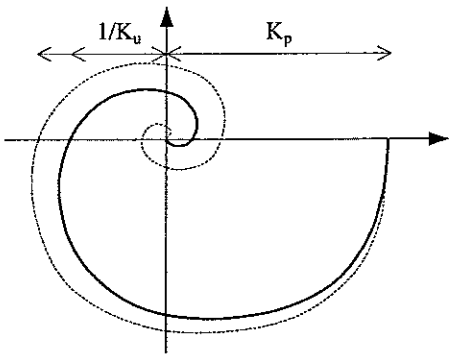

Figure 4: The Nyquist curve of a process and the corresponding kappa. If dead time is added (dashed) kappa grows.

3.2 Using kappa for tuning

Since the gain ratio reveals interesting properties of a process, it seems to be useful for tuning controllers. For PID controllers, one way would be to use a Ziegler-Nichols type of tuning and to use kappa as an additional tuning parameter. This is an idea presented in [5] and the material here is a continuation of that work.

Ziegler-Nichols frequency method (found in for example [1] pp. 136) is based on the ultimate point with parameters K_u and T_u. The PID parameters are then chosen as K=

0.6Ku, T_i= 0.5Tu and T_d= 0.125T_u, i.e. constant values of the quotients K/K_u, T_i/T_u and T_d/T_u. The relay auto-tuner, as shown in section 2.4, results in the same design procedure with constant values of the quotients $K/K_{u\alpha}$, $T_i/T_{u\alpha}$ and $T_d/T_{u\alpha}$ (equation (13)). This approach have shown to give quite good tuning for most processes, but in some cases these methods give poor tuning. A process with long dead time for example, can be significantly better tuned. The conclusion is that constant values of the quotients K/K_u, T_i/T_u and T_d/T_u should not be used for all processes. It seems likely from the discussion above that kappa captures this behaviour, and the idea behind the kappa-tuning is to replace those constant values with functions of kappa.

A natural way would be to find suitable kappa functions empirically, just like Ziegler and Nichols found their constant values from extensive simulations. We should then find a large number of process models that represent the systems typically encountered, and build up a test batch. With this test batch as basis we could now design a in some sense optimal PID controller for each process. Then, by looking at the quotients K/K_u, T_i/T_u and T_d/T_u for all designs, it would be possible to find relations with kappa, being just the kappa functions mentioned above. If we find that there is a correlation between the kappa value and how the values of the above quotients should be chosen, we can construct a tuning method for any process just by using its kappa value and a procedure similar to Ziegler-Nichols or the relay auto-tuner.

3.3 Deriving the tuning method

The procedure we use here is the same used in [5] and the starting point is a large test batch of simple process models. The batch is

$$
\begin{aligned}
G_1(s) &= \frac{e^{-s}}{(1+sT)^2} && T = 0.1,\ldots,10 \\
G_2(s) &= \frac{1}{(1+s)^n} && n = 3,4,8 \\
G_3(s) &= \frac{1}{(1+s)(1+\alpha s)(1+\alpha^2 s)(1+\alpha^3 s)} && \alpha = 0.2,0.5,0.7 \\
G_4(s) &= \frac{1-\alpha s}{(1+s)^3} && \alpha = 0.1,0.2,0.5,1.0,2.0
\end{aligned}
\qquad (16)
$$

making totally 23 process models. They are chosen to be representative for systems found in the processing industry, according to the discussion in 2.1. The four types represent different dynamic behaviour and each type consists of processes over the whole kappa range. The first type is a second order system with a double pole and dead time. The systems G_1, span from dead time dominant with large kappa (small T:s), to lag dominant with small kappa (large T:s). They are very alike the process model (1), and the reason for not using the type (1) in the test batch is that it do not represent typical dynamical behaviour. The step response of (1), seen in Figure 1, has a fast transient response seldom found, while systems like $G_1(s)$ has a more S-shaped step response. The second type, $G_2(s)$, has a multiple pole of different order, where a higher order gives larger kappa. Type three are systems with more or less spread out poles, and these represent processes that have dynamics with different time constants. The more the poles are spread, the smaller is kappa. From the aspect that kappa measures difficulty to control, this can be interpreted as that fast dynamics has no influence, making the system easier to control.

The last type has an unstable zero, resulting in a non minimum phase system. This will affect the transient behaviour in the way that it starts-off in the wrong direction, and as the zero gets closer to the origin (growing α) this effect gets stronger. This will make the system harder to control, and as expected this also implies an increasing kappa.

Next step is to design an optimal PID controller to each process. A special design method presented in [7] is used, and a short summary of this method is presented here. As seen from equation (10) the integrated error under a step load for a system controlled with a PI or PID controller equals the inverse of the integral gain $K_i = K/T_i$. Since attenuation of load disturbances is the main objective in most control problems within the processing industry, it is desirable to use as high value of K_i as possible in a PI or PID controller. The conflicting goal is the robustness of the controller. In [6] it is shown that an efficient design method for PI controllers is to maximise K_i with the requirement that the maximum sensitivity M_s, is less than a specified value. M_s is defined as

$$M_s = \max_{\omega} \left| \frac{1}{1 + G_p(i\omega)G_{PID}(i\omega)} \right| \tag{17}$$

The value $1/M_s$ can be interpreted as the shortest distance from the critical point -1, to the Nyquist curve of the loop transfer function $G_p(s)G_{PID}(s)$. The design method can thereby be viewed as moving the Nyquist curve for the loop transfer function by varying controller parameters without intersecting the M_s circle, and then choose the parameters giving highest K_i. A skeleton sketch is shown in Figure 5. The M_s value is a good design parameter, determining performance of the closed loop system, where higher values of M_s give faster and less damped systems. At the same time it is directly related to the robustness and stability. Reasonable values is in the range, $M_s = 1{,}2 - 2$. The method requires that a process model exists, preferably as a transfer function. Implementing this design procedure requires solving a quite complex optimisation problem, but it is shown in [6] that the problem can be reduced to solving a set of nonlinear algebraic equations. In most cases it is even enough to solve a single algebraic equation for the tangent frequency, and then calculate the controller parameters K and T_i from simple relations. This makes the PI design method reliable and deterministic and it can thereby be made autonomous. In [7] a similar approach is made for the design of PID controllers. Here it is shown that the PI design method can not be extended directly to a PID design. Although

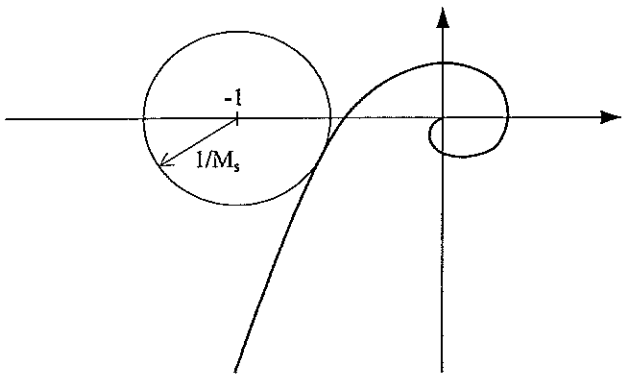

Figure 5: The Nyquist curve of the loop transfer function and the M_s circle.

the optimisation procedure, maximise K_i under the M_s constraint, is solvable, but now with an additional variable T_d, other problems arise. The solutions found are often very narrow, meaning that small changes in the parameters give large changes in the K_i value. This gives a sensitive controller, which is undesirable. These problems exist in the PI case as well, but are here very rare. It is indicated in [7] that the cases with narrow optimum imply that more than one point on the Nyquist curve touches the M_s circle. Despite the fact that these solutions give a higher K_i value, no significant improvement of the control performance is made. Therefore, two additional constrains are added to the optimisation to prevent this type of solution. The first is to require negative curvature of the Nyquist curve in an interval round the touching point, and the second is decreasing phase in this interval. With these extra constraints a in a new sense optimal PID controller can be designed. In [7] the design is made with the Optimisation Toolbox in Matlab 5.2. The Matlab routines written for this have also been used in our work.

As an extra feature in both the PI and PID design the setpoint weighting factor b, see equation (6), can be designed. The purpose is to prevent overshoot at setpoint changes that can occur since the procedure above only considers attenuation of load disturbances, disregarding setpoint response. The design is done by calculating the transfer function from setpoint to process output $G_{sp}(s)$, after the controller parameters K, T_i and T_d are found. The b factor is then chosen so that the maximum gain of G_{sp} is below a specified value. In [7] it is chosen so that $|G_{sp}(\omega)| < 1$ for all ω. Knowing that $G_{sp}(s)$ equals the complementary sensitivity function T(s) when no setpoint weighting is used (meaning derivative action on setpoint as well), a good approximation is that the maximum gain of G_{sp} occurs at the frequency that maximise $|T(\omega)|$. As this frequency can be found from the design, b can be found from setting $|G_{sp}(\omega)| = 1$ at that frequency. The b factor is in the design limited to positive values, and for those cases where b=0 is not enough to damp the overshoot, a setpoint filter is designed in [7]. This will not be used in this work.

The last part in developing the kappa-tuning rules consists of looking at the quotients K/K_u, T_i/T_u and T_d/T_u versus kappa to determine an eventual correlation, being the kappa functions. We start with the PI design. For all the 23 models in the batch (16) a PI design, giving parameters K, T_i and b, is made in Matlab. The ultimate point, with parameters K_u and T_u, is also obtained in Matlab, and since all processes in the batch has unit static gain kappa is simply $1/K_u$. Some of the Matlab routines used can be found in Appendix C. The results are presented in diagrams with logarithmic vertical axis with the quotients K/K_u, T_i/T_u plotted against kappa, and the b factor plotted against kappa. See Figure 6, left. For the PID design the same procedure is used, and it is presented in a similar way, now with the additional diagram T_d/T_u versus kappa. The PID design is seen in Figure 6, right. Both the PI and PID design has been made with two M_s values. M_s=1.4 marked with o and M_s=2.0 marked with x.

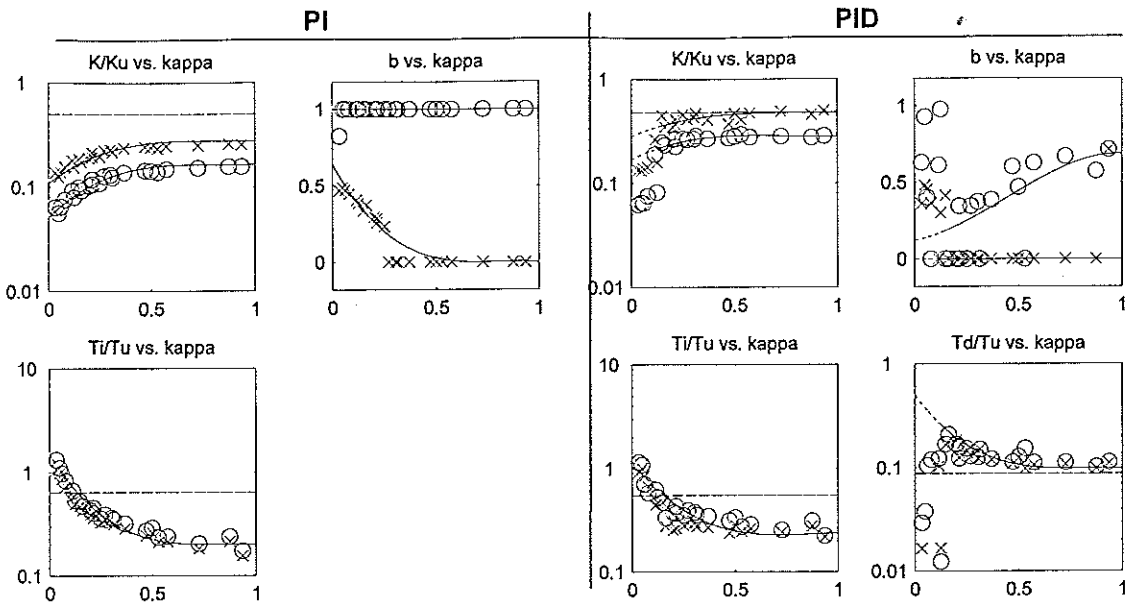

Figure 6: Tuning diagram for PI (left) and PID (right) controllers. The standardised controller parameters K/K_u, T_i/T_u, T_d/T_u and the setpoint weighting factor b are plotted versus kappa. Two M_s values is shown. M_s=1.4 marked with o and M_s=2.0 marked with x. Each cross and circle represent a process from the batch (16). The horizontal dash-dotted lines correspond to the values obtained with the relay auto-tuner.

Looking at the PI design to the left above, a correlation between kappa and the optimal parameter choice can clearly be evidenced. It is also evidenced that constant values of the quotients K/K_u and T_i/T_u is not optimal for the whole kappa range. In the upper leftmost figure above for example we see that the choice of K/K_u vary a factor 2.5 when going from low to high kappa values. Looking at T_i/T_u for the PI design we see an even larger range of the quotient from 0.2 – 1.4, i.e. a factor 7. It is also interesting to note that the optimal choice of Ti/T_u do not differ much between the two M_s values. For K/K_u, on the other hand, the two M_s values seems to differ only by a scale factor. This agrees well with the statement M_s being a god design parameter. Examining the PID design to right in Figure 6, the same nice behaviour is not seen. The three quotient seems to behave irregularly for small kappa values, but forgetting this for a moment, even the PID design show a clear correlation between optimal choice of K/K_u, T_i/T_u, T_d/T_u and the kappa value of the process. The resemblance with the PI design is notable, particularly for the quotient T_i/T_u. Just as T_i/T_u the quotient T_d/T_u does not seem to differ for the two M_s values, and further more, T_i/T_u and T_d/T_u behaves in the same way with decreasing values for increasing kappa. This is interesting since T_d often is chosen as a fraction of T_i, e.g. the relay auto-tuner where it is chosen $T_i = 6.25T_d$. Plotting T_i/T_d versus kappa (not shown) reviles that it is rather constant, approximately 2.5.

For the PID design it is obvious that something happens for kappa values below 0.2. The quotient T_d/T_u for example, do not show any correlation with kappa and K/K_u suddenly drops a factor 3 at $\kappa = 0.13$. If this behaviour really was the optimal choice the kappa-tuning would be of no use, but actually it is not. Investigating the results for small kappa values show that these correspond to the PI design. The explanation is that the optimisation routines for the PID design described above starts with the optimal PI solution, and from this the routine searches an optimal PID solution by adding derivative action. What happens for small kappa values is that no clear optimum can be found, and the design "get caught" in the initial PI design. It is not surprising that this happens for processes with low kappa values. As mentioned, a low kappa value can be interpreted as

processes that are easy to control, low or approximately low model order or non-dominating dead time. These are all factors that indicate that for small kappa values a PI controller is sufficient. This is well known, and from a pole placement point of view it can be thought of as the PI controller being able to place all the closed loop poles, which is true for a pure first or second order system. Trying to determine a PID controller for such processes gives an over determined system. Knowing this it is comprehensible that an optimisation routine has trouble designing PID controllers to processes being almost second or first order systems. The solutions found by the routine might indeed be optimal but only locally. Further, there probably exist solutions to some processes with very extreme PID parameters, indeed giving optimal response to load disturbances, but having other undesirable properties like e.g. high frequency gain. The true optimal solution is hard to find, but at least we can conclude that the results shown in Figure 6, right for small kappa values originates from an inherent problem when designing PID controllers. This matter will be discussed further in chapter 3.7.

Yet not mentioned is the kappa function for the b parameter shown in Figure 6 both for PI and PID design. In the PI design the choice would be constant b=1 for M_s=1.4 and a decreasing function to b=0 for M_s=2.0. The fact that the b factor is set to zero implies that the design of the b factor might not be enough to guarantee well damped setpoint response. For the PID case the b factor is less well-determined, but disregarding small kappa values a trend of increasing b values for increasing kappa can be seen for M_s=1.4 and for M_s=2.0 most the designs have given κ=0. A question that might be asked though, is whether there really exists a good correlation between the optimal choice of b factor and the kappa value for the process, as it does for K/K_u, T_i/T_u and T_d/T_u. From results presented it is not obvious that there do, neither proven that there do not. A contradictory fact, for example, is that for the PID design between κ = 0.2 – 0.3 there exists two groups of solutions, at b=0 and b=0.3 for the same M_s value. We will later, in chapter 4, see results from other design methods that however indicates that design of the b factor could be done from the kappa value. In [5] the design of the b factor is also made based on a kappa function.

As seen in Figure 6, curve fits have been made to generate the four kappa functions $f_K(\kappa)$, $f_{Ti}(\kappa)$, $f_{Td}(\kappa)$ and $f_b(\kappa)$. The functions $f_K(\kappa)$, $f_{Ti}(\kappa)$ *and* $f_{Td}(\kappa)$ have the form

$$f(\kappa) = a_0 \cdot e^{(a_1\kappa + a_2\kappa^2 + a_3\kappa^3)} \qquad (18)$$

In a logarithmic scale this will simply be a cubic polynomial. For the b factor a simple polynomial fit is made as

$$f_b(\kappa) = a_0 + a_1\kappa + a_2\kappa^2 + a_3\kappa^3 \qquad (19)$$

The development of kappa functions is made for totally four M_s values namely: 1.2, 1.3, 1.4 and 2.0. The diagrams for M_s=1.2 and 1.3 are shown in Appendix A, Figure A.1 and A.2. The coefficients in (18) and (19) for all M_s values, PI and PID design, are presented in Appendix B, Table B.1. The choice (18) might seem odd but it gives good fit to data with few parameters. A drawback with the choice (18) is that the approximation is difficult to perform. A least squares approximation is for example not possible. However, a least squares approximation would not be useful since so many data points for the PID design is scattered. Further, the method of least squares would be unfair since there are so few data points at larger kappa values. The curve fit has therefore been done in a more visual fashion, giving trends that seem reasonable. For some trends however, there are no strict motivation, like for example the PID design in Figure 6 for $\kappa < 0.13$ (dashed lines). By taking the logarithm of the data for K/K_u, T_i/T_u and T_d/T_u it can be fitted to a cubic polynomial in kappa with an ordinary least squares method. Observe that this will not give a least squares solution for the approximation (18) to data. End constrains have been added at kappa 0 and 1 to give reasonable extrapolation outside the data material.

3.4 The tuning procedure

The procedure for tuning a controller with the method above will now be quite simple. First a relay auto-tuning experiment will be run, giving the parameters K_u and T_u. An additional experiment will then be run, where the static gain K_p, is found. This additional experiment will be treated in detail in chapter 5. With K_u and K_p kappa is calculated as $\kappa=1/K_pK_u$, and the kappa value is inserted in the four kappa functions $f_K(\kappa)$, $f_{Ti}(\kappa)$, $f_{Td}(\kappa)$ and $f_b(\kappa)$. Since the kappa functions give values of the quotients K/K_u, T_i/T_u , T_d/T_u, the parameters K, T_i and T_d are simply found by multiplying the kappa function with K_u or T_u. The b factor is gained directly from its kappa function.

A problem with the described procedure is that it is not the real ultimate point that is identified in the relay auto-tuner, but rather a point at angle α, see Figure 3. Using the values at the real point, $K_{u\alpha}$ and $T_{u\alpha}$ can not be justified by neglecting the angle α as small. As seen in section 2.4 the nominal angle is $\alpha_{nom} = 30°$. If assuming small variations in α however, the design above can still be used. This will only mean that we do a systematic error that to some extent can be compensated for by trimming design parameters. The real identified point will always have a smaller value of K_u and a larger value of T_u. This will result in a controller with less gain and less integral action being more conservative. The kappa parameter will be estimated systematically to large.

The problems with identifying the wrong point on the Nyquist curve might not be severe, but there is a solution to the problem. The design procedure above could be developed in exactly the same manner but instead using a kappa defined as the gain ratio at the nominal angle, i.e. $\kappa = |\,\mathrm{Re}(-150°)/\mathrm{Re}(0)\,|$. Of course we must standardise the controller parameters with $K_{u\alpha}$ and $T_{u\alpha}$ when deriving the kappa-tuning rules now. This suggestion would eliminate the problem as long as we have a known, constant identification angle α. However, the angle α is not constant. In the relay auto-tuner, after the noise level has been estimated, a desired oscillation amplitude is calculated. The relay amplitude is then adjusted to roughly achieve this oscillation amplitude. To obtain stable limit cycles fast, only a few adjustments is allowed. This means that the oscillation amplitude can differ a lot from the desired and this will, according to equation (12)b, alter the angle α. No quantitative value of the variations in α has been looked at, but angles between 10 - 60° are not uncommon when running the relay auto-tuner. This fact will make the above suggestion questionable. A design procedure developed at nominal angle

30°, would give less conservative control for cases where identification is made at $\alpha <$ 30°. This is undesirable, and it might be preferred having a design procedure sometimes giving a too conservative control instead.

3.5 Integrating processes

Calculating the kappa value for an integrating process gives $\kappa=0$, since such process has infinite static gain. Tuning PI or PID controllers for integrating processes with the kappa-tuning above would thereby always result in a design with the same values of the quotients K/K_u, T_i/T_u, T_d/T_u and the same b factor. This would probably work fine and give reasonable controller parameter. However, this will mean that we lose one tuning parameter, and we have Ziegler-Nichols type of tuning procedure in these cases. Since the idea with the kappa-tuning was to extend the Ziegler-Nichols type of tuning from two to three dimensions, it would be natural to have three tuning parameters even when tuning controllers for integrating processes. We can not use kappa though, since we lost the resolution here, kappa (15) is that we divide with $G_p(0)$. Since we will have data for the ultimate point for integrating processes as well, it would be attractive to keep the definition (15) but instead standardise with a quantity other than $G_p(0)$. A suggestion proposed here is.

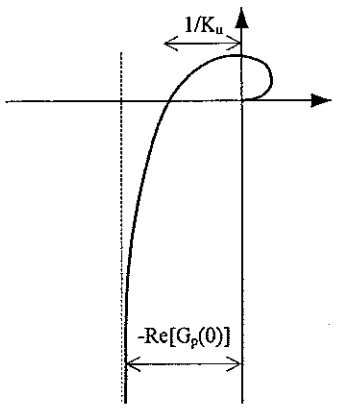

Figure 7: The Nyquist curve of an integrating process and a suggested kappa definition

$$\kappa = \left| \frac{\mathrm{Re}\left[G_p(i\omega_u)\right]}{\mathrm{Re}\left[G_p(0)\right]} \right| = \frac{1}{\left|\mathrm{Re}\left[G_p(0)\right]\right| \cdot K_u} \qquad (20)$$

This definition could actually be seen as a more general definition of kappa, since it gives the same result as (15) for non-integrating processes. The definition requires that a limit value of $\mathrm{Re}[G_p(i\omega)]$ exists as ω goes to zero, see Figure 7. If we assume a transfer function for the integrating process as

$$G_p(s) = \frac{1}{s} \cdot G_{NoInt.}(s) = \frac{1}{s} \cdot \frac{B(s)}{A(s)} = \frac{1}{s} \cdot \frac{b_n s^n + \ldots\ldots + b_2 s^2 + b_1 s + b_0}{a_n s^n + \ldots\ldots + a_2 s^2 + a_1 s + a_0} \qquad (21)$$

it follows that

$$\mathrm{Re}\left[G_p(i\omega)\right] = \frac{1}{\omega} \cdot \mathrm{Im}\left[G_{NoInt}(i\omega)\right] \qquad (22)$$

For this expression to have a limit value when ω goes to zero, $\mathrm{Im}[G_{NoInt}(i\omega)]$ must be of $O(\omega)$. With * meaning complex conjugation it holds that

$$\mathrm{Im}\left[G_{NoInt}\right] = \mathrm{Im}\left[\frac{BA^*}{AA^*}\right] = \frac{1}{AA^*}\mathrm{Im}\left[BA^*\right]$$

For small ω this will give, with ordo notation O

$$AA^* = \mathrm{O}(a_0^2)$$
$$\mathrm{Im}\left[BA^*\right] = \mathrm{O}(\omega(a_0 b_1 - a_1 b_0))$$

This leads to

$$\lim_{\omega \to 0} \mathrm{Re}\left[G_p(i\omega)\right] = -(a_1 b_0 - a_0 b_1)/a_0^2 \qquad (23)$$

From (23) it is clear that a limit value of $\mathrm{Re}[G_p(i\omega)]$ often exist, but it is still hard to motivate that (20) is a good definition of kappa when the process is integrating. Actually, a proper investigation of (20) is rather difficult to perform, and it will not be done in this thesis. We will only show that for the integrating processes we have encountered, the choice (20) seems to be good. The problem of showing that (20) is a sensible definition goes back to defining the class of integrating processes for which it shall be used. This problem exists for the definition (15) as well, but here we defined a subclass of systems for which the kappa definition (15) made sense. In this work we will only think of integrating processes as the ordinary processes, low pass character etc., where we added an integrator.

Now, with the believes that (20) is a good tuning parameter, the same procedure as above for developing kappa-tuning rules can be applied to integrating processes. We will look at the same quotients K/K_u, T_i/T_u, T_d/T_u and the b factor versus the kappa value obtained from (20). The same design procedures for PI and PID controllers will also be used, although the PID design is a slight modified variant to be able to handle integrating processes. The batch of integrating processes that we used is simply the batch (16) where we added an integrator to each process. The results are presented below in the same way as before with PI to the left and PID to the right, see Figure 8. Before discussing the results it is interesting to look at the kappa interval. In Figure 6 the kappa range was 0 – 1, while all kappa values lie between 0.2 – 0.7 now. What can not be seen from the figures is that the processes in the batch lie approximately in the same order, e.g. process number 16 is the one with smallest kappa both in Figure 6 and Figure 8, and the same holds for number 1 being the one with highest kappa value.

If we start by looking at the PI design, we find a clear correlation between kappa and quotients K/K_u, T_i/T_u, T_d/T_u and b. This can actually be taken as an indication of (20) being a reasonable choice of kappa for integrating processes. Unlike the design for non-integrating processes, the quotient T_i/T_u is chosen different for different M_s values (compare with Figure 6). As for the non-integrating processes the PI design is much more deterministic than the PID design. As seen below the quotient T_d/T_u in the PID design is well scattered. Surely, this is not the optimal choice of T_d/T_u, and it is probably due to problems in the design method. If we look at the quotient T_i/T_d (not shown) we find it larger than 30 for most PID designs below. For the non-integrating processes the same quotient was 2.5, and it is clear that vary little derivative action is proposed by the design method. Three different explanations can be thought of: One, it really is optimal with such small derivative action. Two, the design method "get caught" in its initial PI design or three, a PI controller is the optimal solution. It is tempting to use explanation two since the diagrams for K/K_u and T_i/T_u is so alike in the PI and PID design. However, looking closer at the values, it shows that in the PID design, both K/K_u and T_i/T_u is chosen a factor 1.3-1.4 higher as compared to the PI design, so this explanation is not all true. We leave this issue but later in chapter 4, we will compare this with another design method.

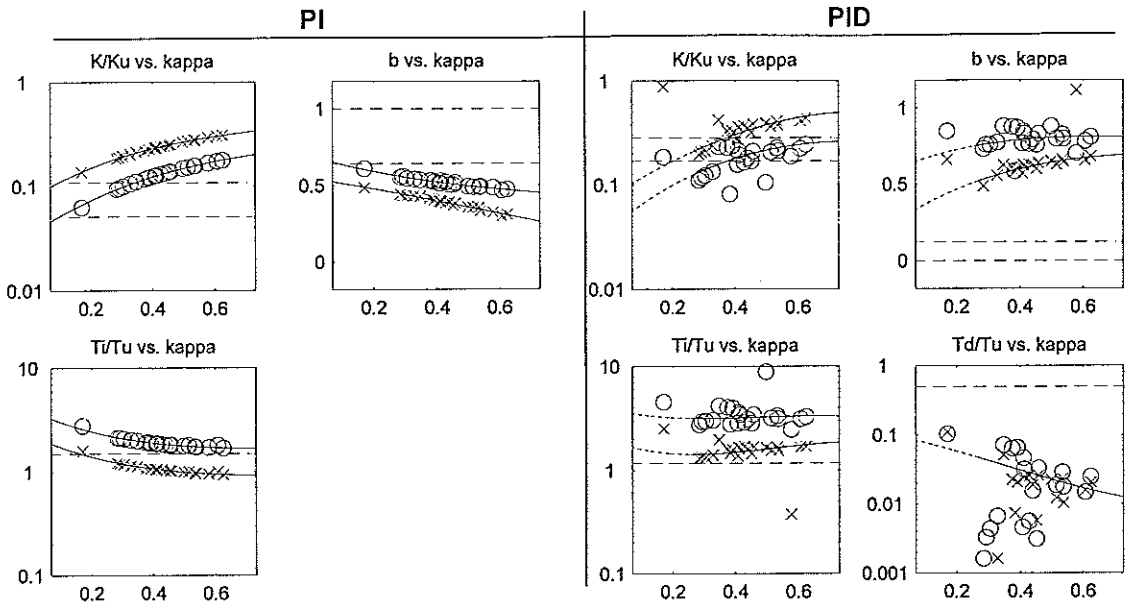

Figure 8: Tuning diagram for integrating processes.(PI left and PID right). The standardised controller parameters K/K_u, T_i/T_u, T_d/T_u and the setpoint weighting factor b are plotted versus kappa. Two M_s values is shown. M_s=1.4 marked with o and M_s=2.0 marked with x. Each cross and circle represent a process from the batch (16) where an integrator has been added. The horizontal dashed lines correspond to the values obtained if κ=0 is inserted in the kappa functions for non-integrating processes (see Figure 6).

It is interesting to compare the kappa functions gained below with the constant values that had been given K/K_u, T_i/T_u, T_d/T_u and b if κ=0 had been used in the kappa functions for non-integrating processes. This is marked with dash-dotted lines in Figure 8. For the PI design the κ=0 value would give a quite low value of K/K_u giving more conservative control. The T_i/T_u value would be the same for all M_s values since this is the case for non-integrating processes. The conclusion would be that we can expect much improved tuning of integrating processes by using the third tuning parameter defined by (20), at least for the PI design. For the PID design the κ=0 values is uncertain since the kappa functions for the non-integrating processes only are extrapolations for kappa values below 0.13 (see Figure 6). The same result as in Figure 8 for the M_s values 1.2 and 1.3 is found in Appendix A, Figure A.3 and Figure A.4. The coefficients for the kappa functions for all four M_s values are found in Table B.2.

To illustrate the consequence of choosing a kappa definition other than (20) for integrating processes, we have made diagrams identical to Figure 8 but with a different kappa definition. Two alternatives are shown, and the results can be found in Appendix A, Figure A.5, A.6 and A.7. Only the PI design is shown, and it should be compared to the PI diagram above (Figure 8), which also is the one shown in Figure A.7. The first alternative shown in Figure A.5 is a kappa definition similar to (15) where we instead divide with the static gain of the process without integrator. This gain could be interpreted as a velocity gain for the integrating process. As seen in Figure A.5 this kappa definition is not well suited as tuning parameter. No distinct kappa functions can be matched to data. The second alternative is simply the kappa value for the process without integrator gained from (15). This result is shown in Figure A.6, and as can be seen, this alternative would work fine. A problem with the last alternative, though, is that this kappa value is very hard to find from simple experiments on the process. As the idea with the kappa-tuning is to use data from a relay experiment, this alternative is not possible. The alternative one, using the static gain for the process without integrator, was actually the

original idea for integrating processes. The velocity gain could be found from quite simple experiments, but from Figure A.5 it is evidenced that this choice is useless. The proposed alternative (20) looks dubious since the quantity $\mathrm{Re}[G_p(0)]$ is hard to estimate from simple experiments. By using a model though, we are able to estimate this quantity, and from the experiment made to find the static gain, $\mathrm{Re}[G_p(0)]$ can be calculated. This will be discussed in chapter 5, but before moving on we will make a small contemplation. The model used to estimate $\mathrm{Re}[G_p(0)]$ is model (3) and we find

$$\begin{aligned} &\lim_{\omega\to 0}\mathrm{Re}\left[G(i\omega)\right] = \lim_{\omega\to 0}\mathrm{Re}\left[\frac{1}{i\omega}\frac{K_p}{1+i\omega T}e^{-i\omega L}\right] \\ &\lim_{\omega\to 0}\left(-\frac{K_p T}{1+T^2\omega^2}\cos\omega L - \frac{K_p}{\omega+T^2\omega^3}\sin\omega L\right) = -K_p(T+L) \end{aligned} \tag{24}$$

It is also interesting to derive $\mathrm{Re}[G_p(0)]$ for the alternative integrating model (2)

$$\lim_{\omega\to 0}\mathrm{Re}\left[G(i\omega)\right] = \lim_{\omega\to 0}\mathrm{Re}\left[\frac{a}{i\omega L'}e^{-i\omega L'}\right] = \lim_{\omega\to 0}\left(-\frac{a}{\omega L'}\sin\omega L'\right) = -a \tag{25}$$

From equation (4) it is seen that $a = K_p(T+L)$, so both models give the same result. The result (24) will be used later in chapter 5. With the model (3) an analytical expression for kappa according to (20) can be derived. The same can be done for the model (1). (Remember that the definition (20) gives the same result as (15) for non-integrating processes). It can be shown that

$$\begin{aligned} G_{INT}(s) = \frac{1}{s}\cdot\frac{K_p}{1+sT}e^{-sL} \quad &\Rightarrow \quad \kappa = \frac{\sin\omega_u L}{\omega_u(L+T)} \quad \text{where} \quad \omega_u T = \frac{1}{\tan\omega_u L} \\ G(s) = \frac{K_p}{1+sT}e^{-sL} \quad &\Rightarrow \quad \kappa = \frac{\sin\omega_u L}{\omega_u T} \quad \text{where} \quad \omega_u T = -\tan\omega_u L \end{aligned} \tag{26}$$

From (26) we see that the choice (20) gives a similar expression for the integrating process model (3) and the non-integrating process model (1). For the model (1) we see that kappa will always be in the range 0–1 (using the equation for the ultimate point we get $\kappa = -\cos\omega_u L$ and $\pi/2 < \omega_u L < \pi$). For the model (3) it can be shown that kappa always lies in the interval $0 - 2/\pi \approx 0.64$. This explains that the kappa range gets much smaller for integrating processes (see Figure 6 and Figure 8).

In this section we have proposed the kappa definition (20). Although looking strange at first we have shown appealing properties in a few aspects. A conclusion drawn is that this kappa definition is very useful for integrating processes and has much in common with the traditional kappa definition for non-integrating processes.

3.6 A similar approach using tau

This section will treat an alternative approach to the tuning procedure above. Just like the kappa-tuning was an extension of the Ziegler-Nichols frequency response method, the method presented now is an extension of the Ziegler-Nichols step response method. In the Ziegler-Nichols step response method the basis is a step response from which two parameters are obtained. By finding the tangent with maximum slope on the step response, the first parameter L is found from the intersection with the time axis. The

second parameter, a, is found from the intersection of the tangent with the y-axis. PID parameters are then found from the simple relations K=1.2/a, T_i=2*L and T_d=0.5*L, i.e. constant value of aK, T_i/L and T_d/L. The extension of this tuning method will be to replace those constant values with some function of an additional tuning parameter. The additional tuning parameter will be the relative dead time, tau. The derivation of this tuning method will only be stated shortly, but for further notice consult [1] and [5]. We will refer to the procedure as tau-tuning.

As the ultimate point and the static gain were used to characterise process dynamics in the kappa-tuning, we will now use the three parameters: the apparent time constant T, the apparent dead time L and the static gain K_p. They can be seen as estimates of the parameters in the model (1). Numerous ways to estimate them exist, but here we will use a simple step response consideration. For stable systems we take the tangent with maximum slope and find L and T according to Figure 9. The static gain is found from the final value. We now define parameter a as

$$a = K_p \frac{L}{T} \tag{27}$$

Note that this parameter is similar, but not identical to parameter a, as defined o riginally by Ziegler-Nichols. The a parameter will be used to standardise the controller gain as aK. The definition of tau is

$$\tau = \frac{L}{L+T} = \frac{L}{T_{ar}} \tag{28}$$

With this definition we find that processes with dominating dead times has tau values close to one. For processes with dominating time constant we have tau close to zero. The parameter tau has much in common with kappa. Both range between zero and one and heuristically we could say $\tau \approx \kappa$. It seems likely that tau would be a suitable tuning parameter to use for extending the Ziegler-Nichols step response method. In analogy with the kappa-tuning, we will derive functions of tau that describe aK, T_i/L, T_d/L and the b factor. These are the quantities considered in Ziegler-Nichols, but in addition to this we will now also look at the quotients T_i/T, T_d/T versus tau, as a complement to the quotient T_i/L, T_d/L.

The procedure for developing the tau-tuning rules is actually identical to the one used for the kappa-tuning. The basis is the batch (16) of processes and the design procedures giving a PI or PID design is the same. Matlab has been used to calculate the step response

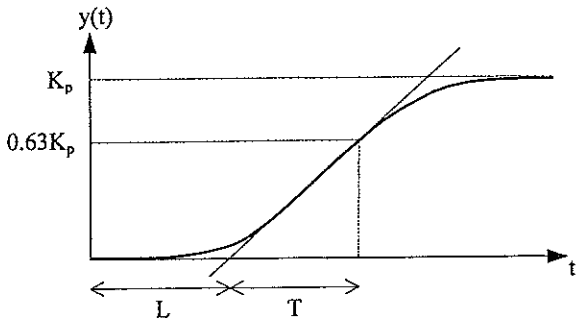

Figure 9: Graphical determination of the three parameters used in the tau-tuning

data to each process, see Appendix C.3. The result is presented in logarithmic vertical axis with aK, T_i/L, T_i/T, T_d/L, T_d/T plotted versus tau. The b factor is plotted in a linear vertical axis versus tau. See Figure 10. As for the kappa diagrams, two M_s values are shown, M_s=1.4 marked with o and M_s=2.0 marked with x.

By looking at the diagrams it can be evidenced that also for the step response method we can expect improved tuning with the use of an additional tuning parameter. The aK value varies approximately a factor 5 in the PI design and a factor 4 in the PID design. This is even more than the variation of K/K_u in the kappa-tuning. The variations of T_i/L is bigger than in T_i/T and this indicates that the Ziegler-Nichols step response method would do better if it instead had used T for determining T_i. This is most visible in the PI design. As mentioned, the PID design procedure has problems for small kappa and tau values, which means that the design get caught in its initial PI design. This can be seen from Figure 10 where, for tau values below 0.25, the data points are well scattered.

No curve fits representing "tau functions" has been made since the tau-tuning is not the focus of this thesis. This can of cause be done. Even though we do not use the tau-tuning, it is useful for comparing. Further, we will also use the results found here in next section when comparing the PID controller to a PPI controller.

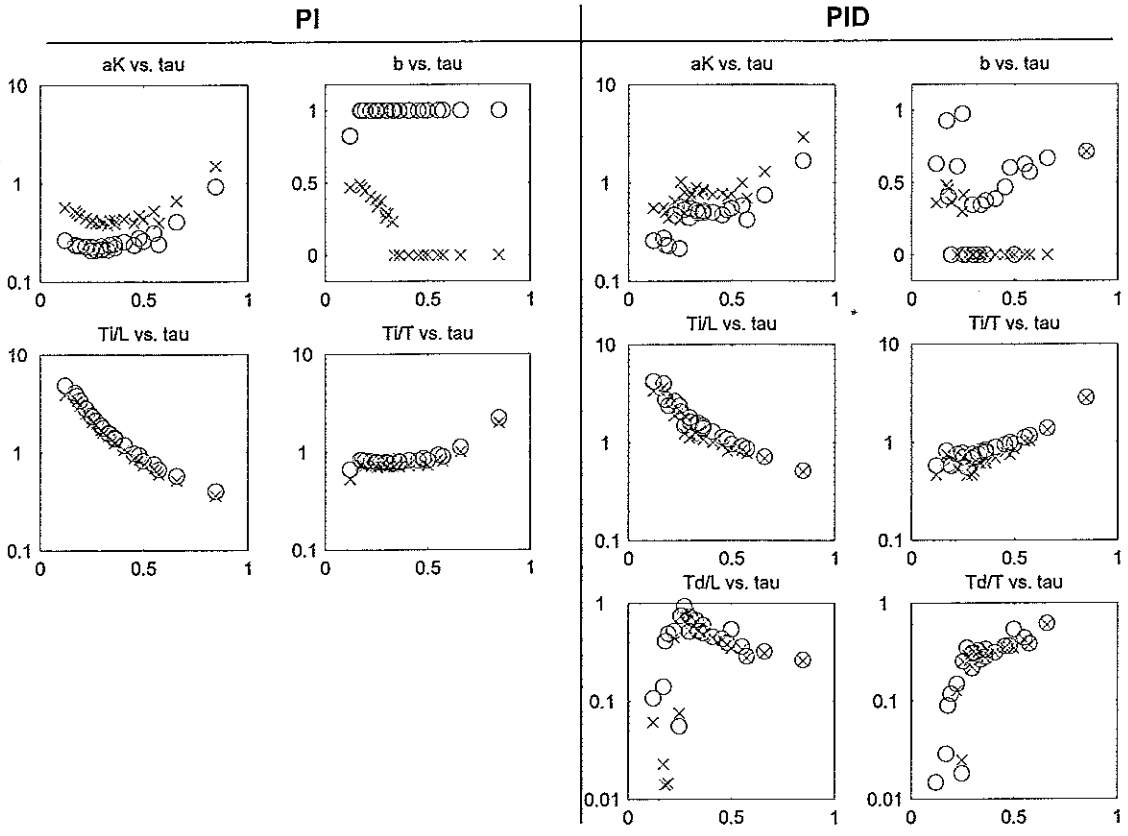

Figure 10: Tuning diagrams for PI and PID controllers using a step response method. The standardised controller parameters aK, T_i/L, T_i/T, T_d/L, T_d/T and the b factor are plotted versus the relative dead time tau. Two M_s values is shown for each process. M_s=1.4 marked with o and M_s=2.0 marked with x. Note that two alternatives exists for standardising T_i and T_d. Either with T or L.

The procedure for tuning a controller with the tau-tuning method would be as follows: Make a step experiment with the process in open loop and determine the parameters K_p, T and L as shown in Figure 9. Calculate the parameter tau from equation (28), an use it to determine values for aK, T_i/L, T_i/T, T_d/L, T_d/T and b, either by looking in diagrams like Figure 10 or by inserting in tau functions fitted to data. The controller gain K is then found from dividing the value for aK by a, where a is calculated from (27). The integral time T_i could be calculated either by multiplying T_i/L with L or T_i/T with T, of course giving the same result. If we have chosen to tune a PID controller, the derivative time T_d is calculated in the same way as T_i. The b factor is gained directly from the diagram (or tau function).

For integrating processes the tau value will be zero, which can be seen by letting the time constant T go to infinity in equation (28). The parameters a and L can still be defined for integrating processes, while the time constant T is infinite. To have three tuning parameters even in the case of integrating processes we have to use another tau definition here. We will not treat this matter here but in [1] further information can be found.

3.7 Additional tuning information

So far we have only been concerned with tuning the four parameters K, T_i, T_d and b. In this last section will discuss two more tuning features that can be extracted from the process information gained. The first one, already mentioned, is the desire to choose a PI controller for low order processes. The second is the choice of a PPI controller where the dead time is sufficiently large.

The kappa-tuning procedure has the possibility to tune both PI and PID controllers. It is intended as a possibility for the operator to choose either PI or PID design. It is important to have this option since in many applications derivative action is not wanted. It would then be a bad idea to design a PID controller and set T_d zero, since this in most cases would give a PI control with less stability margin. However, there exist cases when the operator has chosen PID control, but the process is such that PI control would actually be preferred. From earlier discussions we can conclude that those cases can be described as processes having small kappa values. We would now like to find a limit value of kappa under which a PI design is chosen. Since the kappa value is calculated before the tuning, this choice could be integrated in the kappa-tuning. Either it could be automatically done our alternatively just notifying the operator that a PI design is proposed. The problem of finding a suitable limit value is rather difficult. The PID design might indeed be better from the specific design objective point of view, but now other factors like high frequency gain, the magnitude or "smoothness" of the control signal, or the like, will be crucial for the design. This qualitative reasoning is however very hard to use for determining a quantitative limit between the PI and PID design. In this thesis we shall make quite a heuristic approach. By looking at the optimisation routine used for designing the PID controllers, we find that it has problems finding an optimal solution for processes in the batch (16) with kappa values below 0.13 (see Figure 6). Now, as discussed before, this problem really originates in the same problem as the choice whether PI control is sufficient. Therefore a suggested limit between PI and PID control is this kappa value, $\kappa = 0.13$.

The background to the other tuning feature, the choice of PPI controller, is that processes with long dead times are difficult to control with a PID controller (see chapter 2.3). It would be desirable that the tuning procedure gave guidance for this choice as well. It can be questioned if the choice of a PPI controller should be made automatically from a PID tuning procedure, so this feature will more be thought of as an alert to the operator. Knowing that kappa increase for increasing dead time, it would be possible to use an upper limit value on kappa above which a PPI controller is suggested. The parameter tau seems to be more suited for the decision though, since it involves the dead time more explicitly. As presented yet the kappa-tuning do not include determining tau, but as will be shown in chapter 5 we will estimate some extra process parameters in the experiment made to find the static gain, making it possible to calculate tau. We will now try to find a boundary for the PPI controller expressed as a limit value on tau. The basis will be the equations for the integrated absolute error under a step load found in (10) and (11). We will assume that the PID controller is tuned with the tau-tuning procedure described above, and that the PPI controller is tuned according to (8). Further, we will refer to the tau functions for $aK=f_K(\tau)$ and $T_i/T=f_{Ti/T}(\tau)$, although no explicit expressions for them has been derived.

$$\begin{cases} IAE_{PID} = \dfrac{T_i}{K} \\ K = \dfrac{1}{a} \cdot f_K(\tau) = \dfrac{1}{K_p}\dfrac{T}{L} \cdot f_K(\tau) \quad \Rightarrow \quad IAE_{PID} = K_p L \cdot \dfrac{f_{Ti/T}(\tau)}{f_K(\tau)} \\ T_i = T \cdot f_{Ti/T}(\tau) \end{cases} \tag{29}$$

$$\begin{cases} IAE_{PPI} = \dfrac{T_i + L}{K} \\ K = \dfrac{1}{K_p} \quad \Rightarrow \quad IAE_{PPI} = K_p(T + L) \\ T_i = T \end{cases} \tag{30}$$

By looking at the ratio IAE_{PID} to IAE_{PPI} we can compare the PID and PPI controller regarding the integrated absolute error. We expect the ratio to increase if the controllers are tuned for processes with larger tau, since the PPI controller is supposed to perform well for dead time dominant processes. Therefore we plot IAE_{PID}/IAE_{PPI} versus tau and investigate when the quotient gets larger than one. From (29) and (30) we find

$$\frac{IAE_{PID}}{IAE_{PPI}} = \frac{L}{L+T} \cdot \frac{f_{Ti/T}(\tau)}{f_K(\tau)} = \tau \cdot \frac{f_{Ti/T}(\tau)}{f_K(\tau)} \tag{31}$$

Here we have used the definition (28) of tau.
The expression (31) can now be plotted as a function of tau if we use the data points from Figure 10 for $aK=f_K(\tau)$ and $T_i/T=f_{Ti/T}(\tau)$. This has been done in Figure 11.

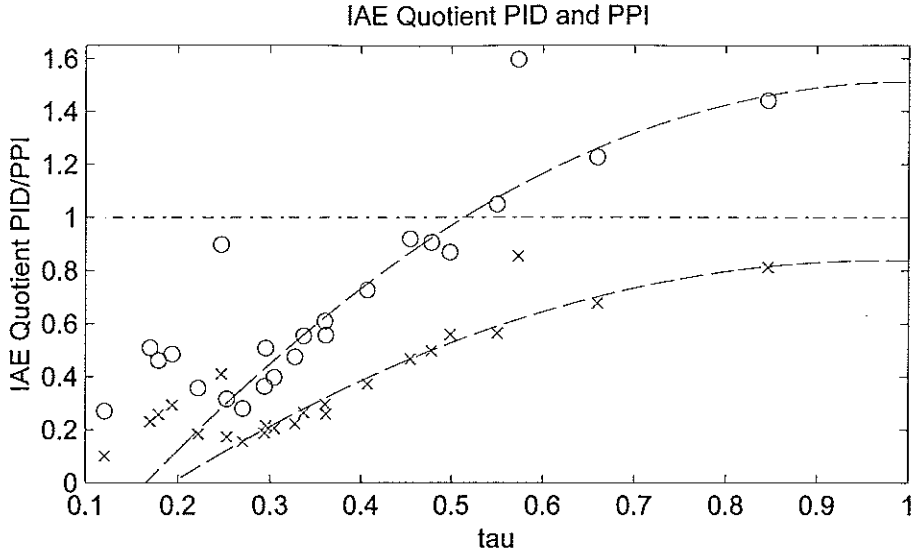

Figure 11: The quotient IAE_{PID} / IAE_{PPI} plotted versus tau. Two different PID designs are shown, while the PPI design is the same. The o marks the PID design with M_s=1.4 and the x marks M_s=2.0. The PID controller gives a larger integrated error when the quotient is above one (dash-dotted line).

Looking at the PID design with M_s=1.4 (circles) we find that the PPI controller will give smaller IEA for $\tau > 0.5$. Only considering this case, we would conclude that the PPI controller would give better rejection of load disturbances for processes where L > T. This is interesting since this is a conclusion drawn in [2] where the PPI controller is compared with a PI controller tuned with a completely different strategy. Now, instead looking at the PID design for M_s=2.0 in Figure 11, we get a different picture. As the M_s=2.0 gives a faster PID controller, we can expect it to give slightly smaller IEA than the M_s=1.4 design. As compared to the IAE for the PPI controller though, there is a very big difference between the two PID designs. From the M_s=2.0 design we would, according to Figure 11, actually conclude that the PID controller always gives smaller IEA than the PPI controller. Of course such conclusion can not be justified based on so little data. As seen in Figure 11 we only have two data point above $\tau = 0.6$, and the final trend of the curves is rather ambiguous. It should also be pointed out that it is actually the IE that we calculate (se section 2.3) with the motivation that for critically damped systems IAE equals IE. This is not all true in the PID case with M_s=2.0, since this design gives closed loop systems that are not completely damped. This design would therefore give slightly higher values on IAE, and the quotient marked with x in Figure 11 would be larger. Nevertheless another conclusion can be drawn. The fact that the ratio IAE_{PID} / IAE_{PPI} is so sensitive to modest variations in the design criteria, makes it a bad idea to use the IAE as basis for the choice, PID or PPI control. It is not reasonable that this choice is so strongly dependent on tuning adjustments. Now this argument is a bit dubious since we only consider one PPI design, but there exists no similar design parameter for the PPI controller. Even though the IAE do not seems to be a good quantity for comparing PID and PPI control, we still believe that tau could be used for the choice. For the time being no other suggestions determining a suitable limit tau value can be thought of.

However, the amount of dead time (tau) might not be the only factor determining weather a PPI controller is desired. It is also a question of e.g. robustness, desired control performance or setpoint following being important. We will not discuss the matter further, but it is yet another argument for not letting a tuning procedure for PID controllers automatically choose a PPI structure.

4. Kappa-tuning compared with other methods

In this chapter we will take a look at three other tuning methods for PID controllers and compare them to the kappa-tuning described in the previous chapter. We will begin with the relay auto-tuner, and add some comments to those already mentioned. In section two we describe a method proposed by people from Chalmers, Control Engineering Lab. This method is very similar to the kappa-tuning presented here, and we will concentrate on the results just making a short summary of the ideas presented. The last section contains an investigation of an industrial product that is a software package for tuning PI or PID controllers. The comparison with the kappa-tuning is here not really fair since we only consider the design part of the program, and not the complete tuning procedure. The program makes a very sophisticated process identification that we unfortunately not have been able to test. The identification part in the other three tuning methods, including the kappa-tuning, are all based on a relay identification, and are therefore more fair to compare.

4.1 The relay auto-tuner

The principles for the relay auto-tuner has already been discussed in chapter 2.4. As was shown here, the tuning procedure emerge in constant values of the quotients K/K_u, T_i/T_u and T_d/T_u. We have not mentioned how the relay auto-tuner makes a PI design, but it is found from [6] that here $K=0.5*K_u$ and $T_i=0.64*T_u$ is the design rules. This choice is independent of the identification angle α. The constant values of the quotients for PI and PID tuning is shown in Figure 6 as horizontal lines together with the kappa functions for $M_s=1.4$ and 2.0. For the PI design it is notable that the relay auto-tuner gives a larger value of K/K_u than the kappa function $f_K(\kappa)$. The value of T_i/T_u is for $\kappa > 0.15$ chosen smaller in the kappa-tuning. The conclusion is that the kappa-tuning gives a smaller proportional gain, but compensate that with a larger integral action. Remember that rejection of load disturbances is determined by K/T_i. The conclusion is not true for small kappa values. Here the kappa-tuning uses less gain and less integral action as compared to the relay auto-tuner, leading to a more conservative control. Now, we must keep in mind that the design criteria used in the generation of the kappa functions guaranties a certain robustness via the M_s constraint, while optimising K/T_i. Thereby the relay auto-tuning, giving higher gain and more integral action, must imply a less robust controller, at least for the processes in our batch. For larger kappa values we will have the opposite situation. If we investigate the results in Figure 6 closer we find that K/T_i gets bigger for the kappa-tuning when kappa is sufficiently large. This despite the fact that we always use less gain.

Comparing the PID designs we find much the same relations as when comparing the PI designs concerning K/K_u and T_i/T_u. For T_d/T_u it is interesting to see that the kappa-tuning use the derivative action to increase the gain (compare PI and PID plot in Figure 6) while approximately using the same integral action as for PI control (remember that derivative action has an "stabilising" effect). The relay auto-tuner rather increase the integral action and use the same gain. We mentioned before that the ratio T_i/T_d is set constant to 6.25 in the relay auto-tuner, while the kappa-tuning leads to a ratio approximately 2.5 (for $\kappa > 0.2$, see Figure 6). This is accomplished mostly by a smaller T_i but even a higher T_d, in the kappa-tuning compared to the relay auto-tuner.

We believe that the above comparison is fair since we use the same identification method to estimate K_u and T_u in both the kappa-tuning and the relay auto-tuner.

4.2 An idea from Chalmers

We will now look at a tuning procedure suggested in [9]. The objective here is really not the design of PID controllers but rather a method for fair evaluation of different controller structures and controller tunings. The evaluation procedure was presented earlier in [8]. The key idea here is to treat low, high and mid-frequency properties separately in the evaluation of the closed loop performance and robustness. In [9] the evaluation procedure is applied to PID controllers, and as an outcome of this some design strategies are presented. Even rules for automatic tuning are presented, and it is actually those rules that we will concentrate on.

We will start by giving a short summary of the evaluation procedure, and how the three frequency ranges are characterised. For the low frequency properties of the closed loop system the integral gain K/T_i can be chosen. As mentioned this quantity should be large for good load disturbance rejection. The high frequency properties are described by the high frequency gain of the controller. This comes natural from the fact that the transfer function from measurement noise to control signal can be approximated with the high frequency gain of the controller for large ω (assuming low pass character of the process). It should be said that in [8] and [9] the low- and high frequency criteria are actually chosen a bit more complex, but the simplifications above will be sufficient for our discussion. The mid-frequency range, characterising stability and robustness, is captured by a generalised maximum sensitivity GM_s, that is an extended version of the maximum sensitivity defined by (17). The GM_s is defined so that, except from specifying a maximum sensitivity, also specifying a maximum of the complementary sensitivity and as a third restriction even the amplitude margin. This extension is made to be able to improve phase- and amplitude margin without reducing M_s too much. Now with these three quantities specifying the different frequency ranges, evaluation is done by looking at one frequency range at a time, while keeping the properties in the other two constant and equal. Controller design can be made in the same way as the evaluation, by optimising the quantity in one frequency range, while keeping the other two fixed.

We will now look at the result from applying these ideas to PID design. To see the coupling to the high- and low frequency properties, in [9], the PID structure

$$G_{PID}(s) = K \cdot (1 + \frac{1}{sT_i} + \frac{sT_d}{1 + sT_d / N}) \tag{32}$$

is reparameterised as

$$G_c = k_0 \frac{1}{s} + k_\infty \frac{s + \theta}{s + (1 + N)\theta} \tag{33}$$

where we have

$$\begin{aligned} k_0 &= \frac{K}{T_i}, \quad k_\infty = K(N+1), \\ \theta &= \frac{1}{T_d + T_d / N} \end{aligned} \tag{34}$$

Here the low frequency properties is directly seen from $k_0 = K/T_i$ and the high frequency properties from k_∞.

The evaluation and design procedure is in [9] now made with an optimisation of the low frequency performance under the constraints of constant mid- and high frequency properties. The optimisation has been done both by varying all four controller parameters K, T_i, T_d and N, and by varying only three with a fixed value of T_i/T_d. Some general, very interesting, conclusions are drawn. For example that N, the filter time factor in the derivative part, should mostly be chosen much lower than often suggested N=8-10, to be optimal. It is however not optimal to choose a constant value of N for all types of processes. The choice is of course strongly related to how the limit of the high frequency gain is chosen. Another conclusion drawn is that T_i/T_d can be chosen constant for most processes, but a rather small value, $T_i/T_d = 2.5 - 3.0$, is proposed.

In [9], the kappa value is also mentioned in the discussion, and it is said that investigations have been made for processes with kappa values from 0.05 to 0.82. It is also said that during the attempts to optimise PID controllers, regularities were found making it possible to formulate simple tuning rules. After a short discussion the following rules are then presented as being based on experience:

$$\begin{aligned} k_\infty &= (30\kappa^2 - 35\kappa + 12) / |G(0)| \\ k_0 &= (1.4\kappa^2 - 2\kappa + 0.95)\omega_u / |G(0)| \\ \frac{T_i}{T_d} &= 3 \\ N &= 2.5 \cdot k_\infty |G(i\omega_u)| \end{aligned} \tag{35}$$

It is interesting to note that, just like the kappa-tuning, these rules are based on kappa. We will now take a closer look at the rules given by (35), and mostly how they relate to the rules found for the kappa-tuning. The rules (35) gives a simple way to decide the parameters to the controller (33), but from our point of view it would be better with rules giving parameters to (32). This can be done simply with the use of (34), but it would be even more interesting if we could rewrite (35) with the use of (34) so that K/K_u, T_i/T_u, T_d/T_u and N could be expressed only as functions of kappa. It would then be very easy to compare the suggested rules with the kappa-tuning. This can indeed be done, and it is not difficult show that the following holds

$$\begin{aligned} \frac{K}{K_u} &= \frac{\kappa \cdot (30\kappa^2 - 35\kappa + 12)}{1 + 2.5\kappa \cdot (30\kappa^2 - 35\kappa + 12)} \\ \frac{T_i}{T_u} &= \frac{(30\kappa^2 - 35\kappa + 12)}{2\pi \cdot (1.4\kappa^2 - 2\kappa + 0.95) \cdot (1 + 2.5\kappa \cdot (30\kappa^2 - 35\kappa + 12))} \end{aligned} \tag{36}$$

$$\begin{aligned} \frac{T_d}{T_u} &= \frac{1}{3} \cdot \frac{(30\kappa^2 - 35\kappa + 12)}{2\pi \cdot (1.4\kappa^2 - 2\kappa + 0.95) \cdot (1 + 2.5\kappa \cdot (30\kappa^2 - 35\kappa + 12))} \\ N &= 2.5\kappa \cdot (30\kappa^2 - 35\kappa + 12) \end{aligned} \tag{36}$$

These expressions could be interpreted as kappa functions in the same way as we have used the term kappa functions in the kappa-tuning method. It is straight-forward to plot them in the same way as we have plotted kappa functions before. The result can be seen in Figure 12 to the left. Note that the N factor is plotted where we have plotted b in previous diagrams. If we begin with K/K_u we see that this quotient goes to zero as kappa goes to zero. This will however not mean that the controller gain K goes to zero since we

divide with K_u that goes to infinity (remember $\kappa=1/K_pK_u$). From (36) it is found that a process having $\kappa=0$ and $K_p= 1$ will get a K=12. This strange looking behaviour is therefore only a consequence of the standardisation K/K_u and the logarithmic scale. Looking at the N factor we see that it is chosen as low as 3 for kappa in the range 0.1 - 0.6 and even lower for $\kappa<0.1$. The reason for using lower values of N for small kappa is that we increase the controller gain K here and in order to maintain a small high frequency gain K(N+1), we must reduce N. The curves for T_i/T_u and T_d/T_u only differ by a scaling factor since $T_i/T_d = 3$ (see (35)).

To the right in Figure 12 we have added the data points from the kappa-tuning diagram for PID design $M_s=1.4$ (found in Figure 6). It is surprising how well the two methods match. If we disregard kappa values below 0.15 we find much the same trends. However, there are a little divergence between the methods for higher kappa values. T_i/T_u and T_d/T_u decreases for $\kappa>0.7$ in the curves given by (36) while K/K_u increase. These trends is hard to see in the kappa-tuning data but in T_i/T_u it would indeed be an acceptable approximation. In our kappa functions presented in Figure 6 we do not have these "tails" but as pointed out before, the approximations in both the end points is quite ambiguous in the kappa functions. Considering the choice $T_i/T_d = 3$ in (35), and the fact that both T_i/T_u and T_d/T_u seems to agree between the methods, indicates that a constant choice of the ratio T_i/T_d might be close to optimal. In the kappa-tuning it is found to be slightly lower, $T_i/T_d = 2.5$. Both values can be considered as lower than normally suggested.

The above comparison can be considered quite fair. The parameters needed for tuning is in the presented method the same as for the kappa-tuning, and similar identification experiment can therefore be used. Comparing the different tunings more closely must of course also involve the tuning of the N and b factors, but this will not be done here.

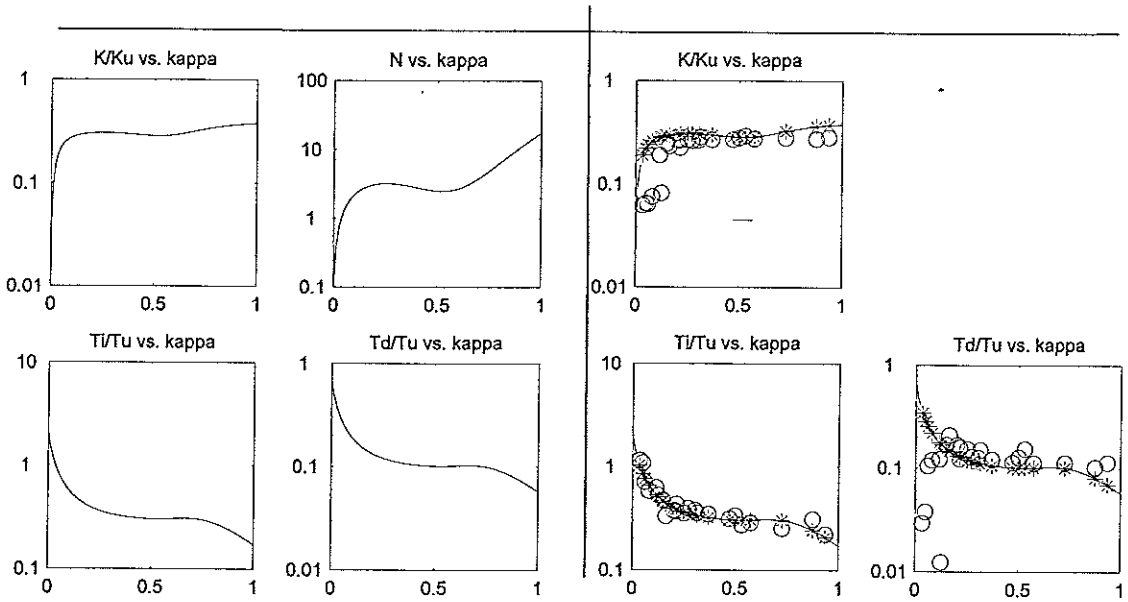

Figure 12: Left: The kappa functions in equation (36) plotted in logarithmic vertical axis. Observe that the top left diagram now show the N factor, and not the b factor as before. Right: Equation (36) together with data from the design procedure for the kappa-tuning development with $M_s=1.4$ (circles). The stars only mark kappa values for the processes used in previous chapter, to make it simpler to compare.

4.3 The AdvaControl® Loop Tuner from ABB[1]

In this section we will compare the kappa-tuning with a tuning procedure used in a commercial product. The background for this is that both methods eventually will be implemented in products from the same company, namely ABB Automation. The section will differ from the previous in the way that it contains less control-theoretical considerations. Partly since this is beyond the scope of the comparison and partly since the manuals for the product do not contain details of the theoretical origin.

ABB Automation is a large manufacture of control systems and supplies many products for automation and control. The one we will look at, the AdvaControl Loop Tuner, is a stand-alone software package for process identification and tuning of PID controllers. It also supports process analysis and both open- and closed loop simulations. It runs on PC under Windows. For data acquisition a serial communication with the ABB Master system is used, but there is also a possibility for DDE communication with other programs. This gives an ability to use the AdvaControl Loop Tuner without having a control system from ABB.

The process identification is a least-squares method based on a discrete process model. Process dead time is estimated and a suitable model order is chosen automatically. There are however possibilities to interfere with the procedure and specify desired properties of the estimated model. This can be done also in continuos time by specifying a transfer function and a dead time. The continuous process model must be of order five or lower. This model is then sampled, since the identification always is made on the discrete model. A sampling period must be chosen by the user, and it is restricted to a multiple of the dead time and larger than 50 ms. The identification can be based on either historical data or on-line data. After the identification a process model validation is also made.

The controller design part is a dominant pole placement giving parameters to a PID structure like the one in (6). The four parameters K, T_i, T_d and b are designed while no design of the N factor is made (the N factor can be chosen by the user). The pole placement is made for the discrete model. A design parameter is the damping of the closed loop poles. Four alternative are suggested: ζ=0.6 ("fast"), ζ=0.8 ("normal"), ζ=1.2 ("damped") and ζ=1.6 ("extra damped"), but the parameter ζ can be chosen continuously. We will not go into the other features available in the AdvaControl Loop Tuner, but it ought to be mentioned that quite an extensive data analysis like FFT, statistical data and correlation analysis can be made. There are also possibilities to make closed loop simulations to evaluate the achieved controller, look at a bode diagram and even a pole-zero map. Of this we have only used the simulation in the time domain, which is a closed loop step followed by a load disturbance.

[1] AdvaControl is a registered trademark of ABB Process Automatic Corp.

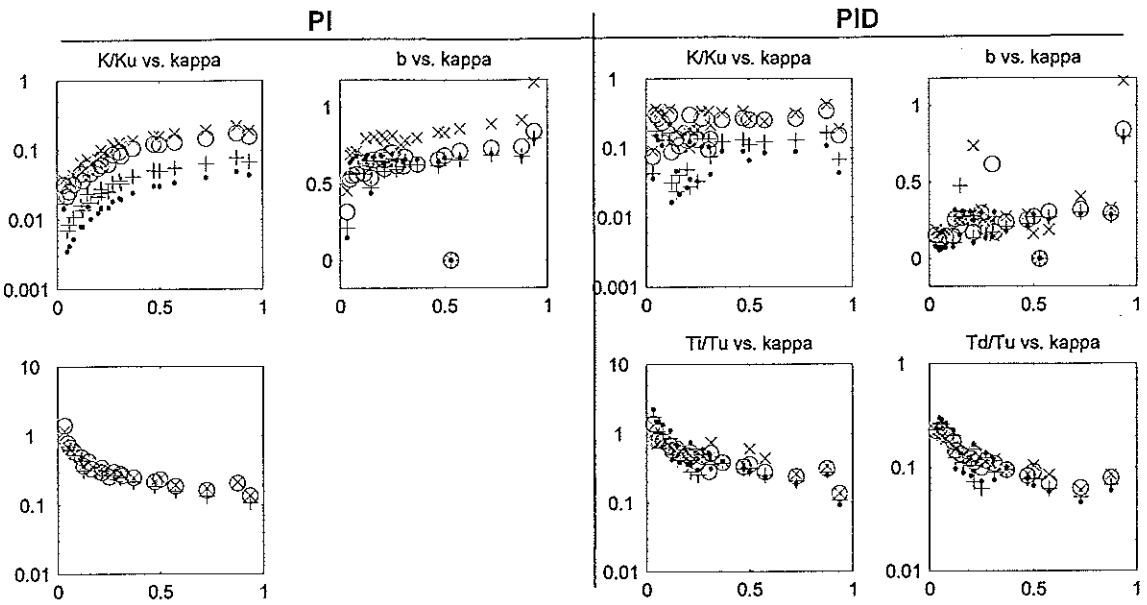

Figure 13: A PI and a PID design made with the AdvaControl Loop Tuner for the processes in the batch (16). Four different values of the design parameter ζ are shown: ζ=0.6 ("fast") [x], ζ=0.8 ("normal") [o], ζ=1.2 ("damped")[+] and ζ=1.6 ("extra damped") [•]. The diagrams are made in a similar way as the previous.

Now, to compare the AdvaControl Loop Tuner to the kappa-tuning it would be desirable to do full-scale experiment with both identification and controller design. It could be done on either simulated processes or real processes. This would have been possible but has not been done, most due to lack of time. Instead we have made another approach. We have taken all the 23 process models in the batch (16), put each one of them in the AdvaControl Loop Tuner, and then run the tuning part of the program. Then we have plotted the results, with the same standardisation i.e. K/K_u, T_i/T_u, T_d/T_u and b, versus the kappa value of the process, thereby making it comparable to the kappa functions as presented previously. This is of course nothing but a simple replacement of the earlier optimisation routine by this pole placement design, and it can thereby not be claimed to be a comparison between the kappa-tuning and the AdvaControl Loop Tuner. Therefore we will rather look on it as a way to gain further insight in the choice of suitable kappa functions.

Since we specify our models in continuous time, but the design is made for the sampled model, we have to be careful with the choice of sampling period. To not be influence by the choice of sampling period, we could always use a very high sampling rate as compared to the time constant of the closed loop system, and thereby neglecting sampling effects. Unfortunately this was not possible since the sampling period could not be chosen below 50 ms, and some of our processes had time constants in the region 200 ms. Therefore we had to use a more moderate sampling, adjusted to each process so that they all were affected equally by the sampling. For this we have tried to achieve approximately ten samples per rise time of the closed loop system.

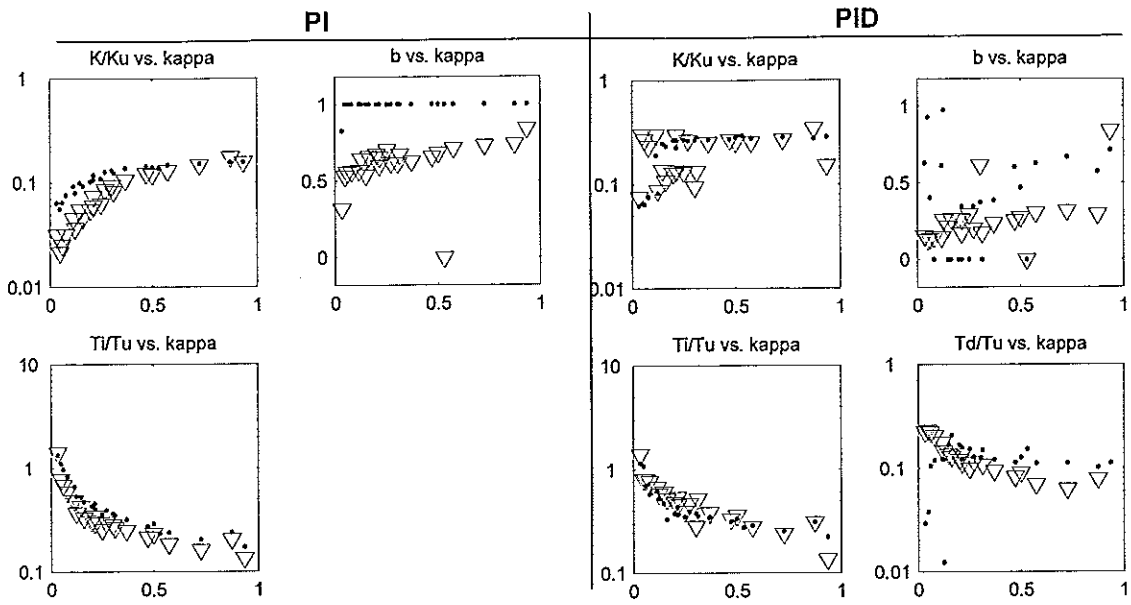

Figure 14: The AdvaControl Loop Tuner design [∇], compared to kappa-tuning [•], for PI and PID controllers. The AdvaControl Loop Tuner design is shown for ζ=0.8 ("normal") marked with ∇. The kappa-tuning data is shown for M_s=1.4 and it is marked with •.

The results are shown in Figure 13. Four value of the design parameter ζ are shown. ζ=0.6 (x), ζ=0.8 (o), ζ=1.2 (+) and ζ=1.6 (•). The first reflection that can be made is that even the pole placement design has a clear correlation to the kappa value of the process. We would also with this design method be able to find suitable kappa functions that could be used for tuning. Regarding the b factor, for which we hesitated in drawing conclusions about earlier, we now find data that would be easier approximated with a kappa function. However, the design procedure for the b factor in the AdvaControl Loop Tuner has not been revealed. For T_i/T_u in the PI design, and T_i/T_u and T_d/T_u in the PID design we do not have much divergence between the different design parameters. We can also note that the ratio T_i/T_d (not shown) seems to be rather constant even in this design method, but that the value is higher. We estimate it to T_i/T_d=4. Looking at the PID design in Figure 13 for kappa values below 0.3 we find that some data points for K/K_u lies much lower. These are not PID designs that has degenerated to PI designs, as happen for some designs in the kappa-tuning, and it is only the K/K_u that deviates. We have simulated these designs both as they are suggested by the design, but also with a higher gain as suggested by the trends of the others. We can not find any reason for using this much lower gain in these specific cases.

To compare the pole placement design by the AdvaControl Loop Tuner with our results from developing kappa functions we have plotted them together in Figure 14. We have chosen the M_s=1.4 design (marked •) from the kappa-tuning, and the ζ=0.8 (marked ∇) from AdvaControl Loop Tuner. Looking at the PI designs we find that both methods assign similar T_i/T_u. For K/K_u the pole placement has a faster decay as kappa decrease. In the PID design we do not find this discrepancy in K/K_u. We might also point out that the process with highest value of kappa gave a PID design with T_d=0 in the AdvaControl Loop Tuner.

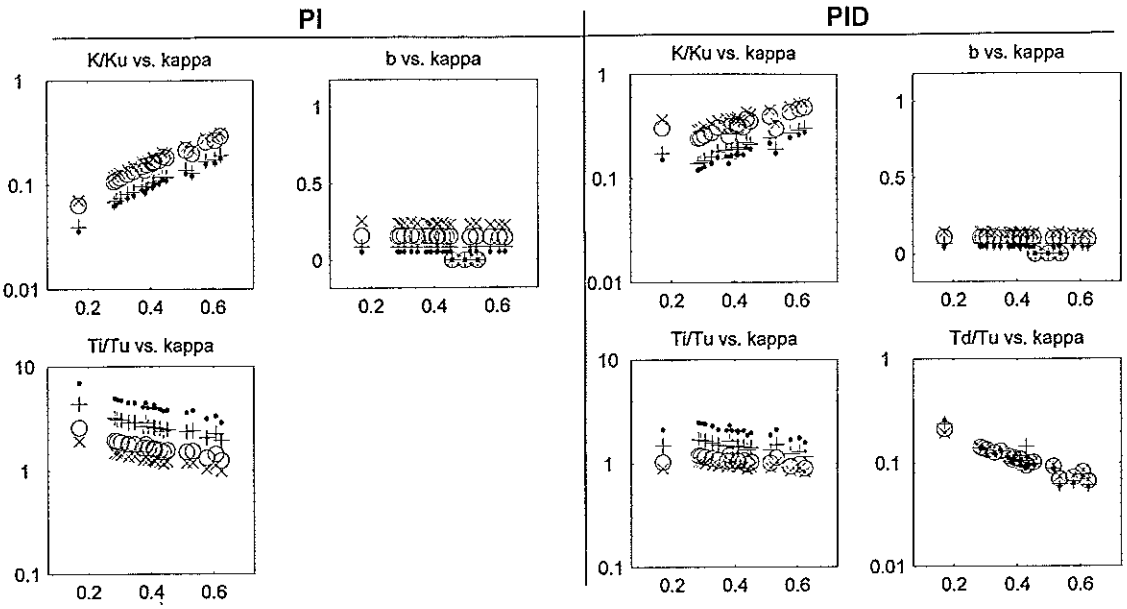

Figure 15: A PI and a PID design for integrating processes made with the AdvaControl Loop Tuner. Four different values of the design parameter ζ are shown: ζ=0.6 ("fast") [x], ζ=0.8 ("normal") [o], ζ=1.2 ("damped")[+] and ζ=1.6 ("extra damped") [•]. Note that the x-axis only range from 0.1 to 0.7.

The AdvaControl Loop Tuner also has the ability to tune PI and PID controllers for integrating processes. For this matter we have done the same procedure as above for all the processes in the batch (16) where we added an integrator to each process. This was done in the development of the kappa-tuning as well, and here we used another kappa definition (20). This definition has now been used when presenting the result from AdvaControl Loop Tuner, and it can be seen in Figure 15. We will not comment the results much. It must be noted though, that the interval of kappa is much smaller now and only range from 0.2 to 0.7.

For comparison with the kappa-tuning we have also plotted both design methods together in Figure 16. As before it is the M_s=1.4 design (marked •) from the kappa-tuning, and the ζ=0.8 (marked ∇) from AdvaControl Loop Tuner. The PI design needs no further comments, but in the PID design some notable facts must be pointed out. When we compare the tuning suggested by the pole placement design with the one suggested by our kappa-tuning we find rather large differences. Such large differences were not found between the methods for the non-integrating processes (see Figure 14). The kappa-tuning gives smaller gain, less integral action and less derivative action. As we have mentioned before the ration T_i/T_d is 30 and even larger in same cases. When this strange behaviour of the optimisation routine was discussed earlier some possible explanations were found. We found for example that the PID solutions is similar to the PI solutions, but that this is not all the truth. The suggestion that the result is truly an optimal solution might still be possible, considering the constraints, but it seems strange that the design gives so conservative control as compared to the pole placement. Yet another explanation would be that the solutions are optimal as the optimisation problem is formulated, but that this formulation gives undesirable results. We will have to leave these questions unanswered.

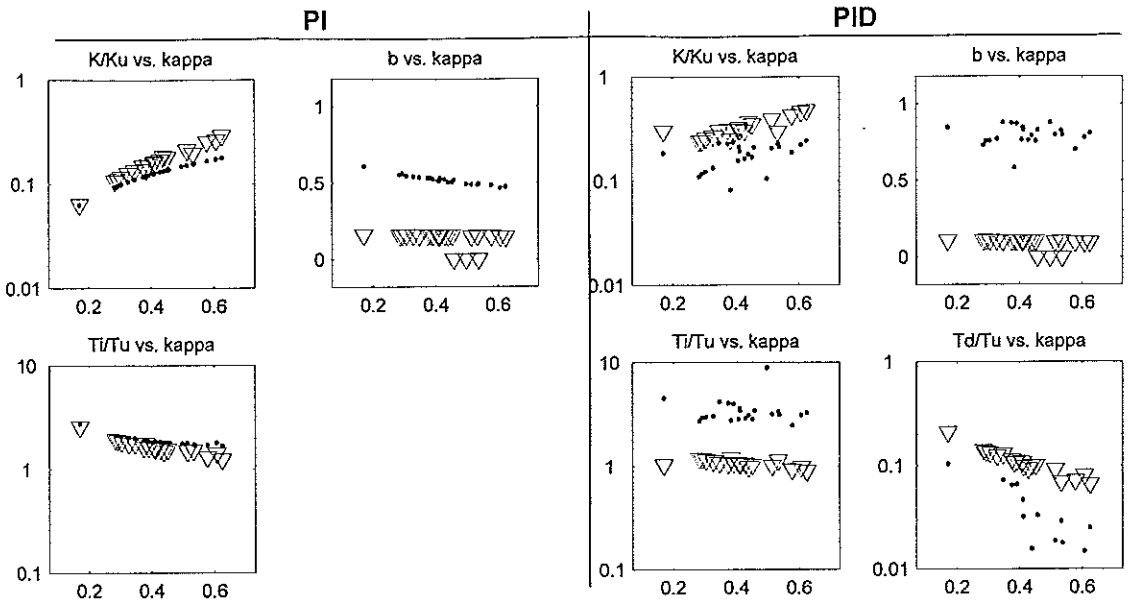

Figure 16: The AdvaControl Loop Tuner design [∇], compared to kappa-tuning [•], for integrating processes. The AdvaControl Loop Tuner design is shown for ζ=0.8 ("normal") marked with ∇. The kappa-tuning data is shown for M_s=1.4 and it is marked with •.

5. The additional identification experiment

As mentioned in the introduction we refer to the kappa-tuning as the procedure of tuning a controller from relay experiment data and the kappa parameter. We have yet only described the relay identification and assumed that we have the kappa parameter, but for the determining of kappa we must also know the static gain of the process. This chapter will deal with the experiment that we use to estimate the static gain. We will also show how the experiment can be used to estimate some extra parameters that can be used to tune a PPI controller. The identification will be based on a closed loop step. For this to be possible we must have an initial controller for the plant. The thought is that we shall use the relay auto-tuner for this, thereby using the relay experiments both for identifying the ultimate point and give initial controller parameters. In section one we will define the parameters that we would like to estimate. In section two we will describe the procedure for estimating parameters with methods of moments. Next in section three we describe how the experiment will be done. Since we now have all components in an auto-tuning procedure, the last section will be devoted to a discussion about the kappa-tuning as a complete automatic tuning.

5.1 Additional parameters needed

First of all we need the static gain K_p, for the process. With this parameter and the data from the relay experiment we have K_u, T_u and $\kappa=1/K_pK_u$, which is sufficient for the kappa-tuning. For determining K_p we only need two equilibrium points where the static gain could be found from simply dividing the difference in output signal with the difference in input signal between the two states (we assume linear systems). Such experiment is easy to perform and what first comes to mind is of course a simple step signal on the process input. This experiment has the drawback that we do not know how large the response in the process output will be, and it can be hard to automate without prior information of the process. It would be safer to make an experiment in closed loop with a reasonable controller. For these reasons we have chosen a simple setpoint step from one equilibrium point to another, and the static gain can be found from just recording the changes in control signal and process output. One advantage now is that we always know how large the process output will be, and we can fix it to say, five percent of the process span.

From chapter three we know that the dead time L, and the time constant T, of the process would be very useful parameters. They would give the ability to tune a PPI controller for cases when such controller would be needed, and that decision could be based on the parameter tau, also available now. Since we now make a setpoint step, it seems reasonable that these two parameters could be estimated as well. We could express this as that we from the closed loop step want to estimate the three parameters K_p, T and L in

$$G(s)=\frac{K_p}{1+sT}e^{-sL} \tag{37}$$

In section 3.6 we dealt with integrating processes and found a suitable kappa definition according to (20). We also looked at the integrating process model

$$G_{INT}(s)=\frac{1}{s}\frac{K_p}{1+sT}e^{-sL} \tag{38}$$

and it was shown that

$$\lim_{\omega\to 0}\mathrm{Re}\left[G_{INT}(i\omega)\right]=-K_p(T+L) \tag{39}$$

With the definition (20) we find that kappa for the process model (38) would be

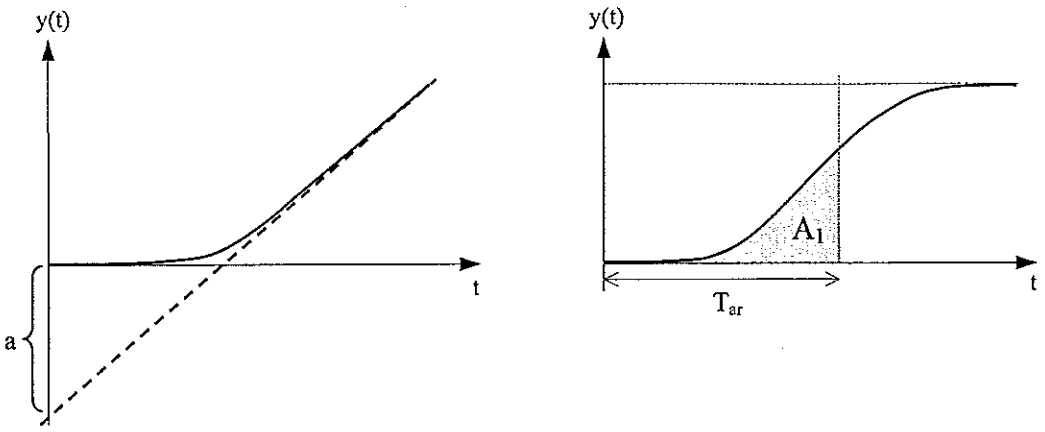

Figure 17: Left: The step response of the integrating process model (38). $a = K_p(T+L)$. Right: The step response of a non-integrating process and an area A_1 to estimate the time constant T. $T_{ar} = T+L$.

$$\kappa = \frac{1}{\left|\mathrm{Re}\left[G_p(0)\right]\right| \cdot K_u} = \frac{1}{(T+L)K_pK_u} \qquad (40)$$

(note that this quantity is dimensionless since K_pK_u has the dimension s^{-1})
Now, even for integrating processes we have the parameter K_u from the relay experiment, and to calculate a kappa value according to (40) we would need the quantity $K_p(T+L)$ as an estimate to model (38). In Figure 17 to the left the step response of model (38) and the quantity $a = K_p(T+L)$ is shown.

5.2 Area methods and methods of moments

In this section we will show that a technique called methods of moments (for further information confer [1]) can be used to estimate T and L in the process model (37) and the quantity $K_p(L+T)$ in (38). The technique is based on integrals, or "areas" under the signals in a step response. For the completeness of the presentation we will briefly describe the theoretical background.

We begin with a simple way of estimating T in the model (37). For this model we have earlier defined the average residence time $T_{ar} = T+L$. It can be shown that the integral from t=0 to $t=T_{ar}$ of the step response for the model (37) is

$$A_1 = \int_0^{T_{ar}} s(t)dt = K_pTe^{-1} \qquad \Rightarrow \qquad T = \frac{eA_1}{K_p} \qquad (41)$$

An estimation of the time constant T can thereby be found from the area A_1 if the average residence time T_{ar}, is known. See Figure 17 to the right. This can be seen as an alternative estimation of T to the 63 % level as we used earlier, see Figure 9.

The method with area determining above has the advantage that it is not so sensitive to noise as for example the determining of the 63 % level. We will therefore now in this spirit try to use area calculations in a closed loop step as well. We will be guided by the methods of moments. Here the key idea is to estimate the quantities G(0) and G'(0) where G(s) is the transfer function of the process, and the differentiation is made in s. By using that

$$G(s) = \frac{K_p}{1+sT}e^{-sL} \quad \Rightarrow \quad \begin{cases} G(0) = K_p \\ G'(0) = -K_p(T+L) = -K_pT_{ar} \end{cases}$$

we can for our purpose calculate T_{ar} as

$$T_{ar} = -\frac{G'(0)}{G(0)} \tag{42}$$

Generally the method of moments can be used to estimate other quantities that can be expressed in G(0), G'(0), G''(0) etc, but we will only use it for determining T_{ar}. To gain G(0) and G'(0) we use the following, where Y(s) and U(s) is the Laplace transformed process value and control signal

$$\begin{cases} Y(s) = G(s)U(s) \\ Y'(s) = G'(s)U(s) + G(s)U'(s) \end{cases} \Rightarrow \begin{cases} Y(0) = G(0)U(0) \\ Y'(0) = G'(0)U(0) + G(0)U'(0) \end{cases} \tag{43}$$

Further the following holds from the definition of the Laplace transform where small letters u(t) and y(t), are the corresponding time signals

$$\begin{cases} U^{(n)}(0) = (-1)^n \int_0^\infty t^n u(t)dt \\ Y^{(n)}(0) = (-1)^n \int_0^\infty t^n y(t)dt \end{cases} \Rightarrow \begin{cases} u_o = U(0) = \int_0^\infty u(t)dt \\ u_1 = U'(0) = \int_0^\infty t \cdot u(t)dt \\ y_o = Y(0) = \int_0^\infty y(t)dt \\ y_1 = Y'(0) = \int_0^\infty t \cdot y(t)dt \end{cases} \tag{44}$$

Now it is easily shown from (42), (43) and (44) that

$$T_{ar} = \frac{y_1 u_0 - y_0 u_1}{y_0 u_0} \tag{45}$$

With this expression and the four integrals in (44) we can estimate the parameter T_{ar} for arbitrary signals, but for this to be true the integrals in (44) must converge. If we make a closed loop step this will not really be the case, since we then will reach stationary values on both u and y, but a simple trick solves this problem. To derive the final expression for T_{ar} we will start with the derivatives of the signals u(t) and y(t), with the notation $u_d(t)$ and $y_d(t)$. Equation (43) and (44) still holds if we just replace y and u with y_d and u_d, and for the four integrals in (44) we use the notation u_{d0}, u_{d1}, y_{d0} and y_{d1}. As in equation (45) we now simply find

$$T_{ar} = \frac{y_{d1} u_{d0} - y_{d0} u_{d1}}{y_{d0} u_{d0}} \tag{46}$$

To calculate the four integrals in (44) we will use the notations introduced in Figure 18, which shows the closed loop step. The final values of the process output and the control signal are referred to as y_f and u_f.

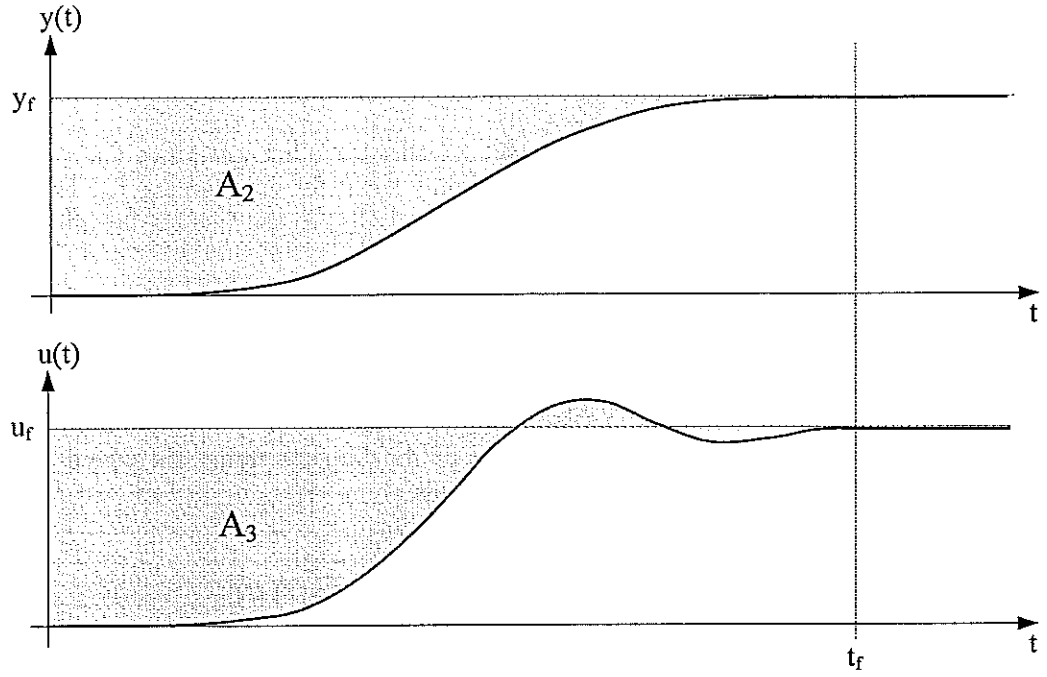

Figure 18: A closed loop step. The uppermost curve is the process output, and the lower is the control signal. At t_f we are back at steady state now with new stationary values y_f and u_f. We assume that we start from zero. The areas A_2 and A_3 are indicated in the figure.

We now find

$$\begin{cases} u_{do} = \int_0^\infty u_d(t)dt = \int_0^\infty u'(t)dt = u(\infty) - u(0) = u_f \\ u_{d1} = \int_0^\infty t \cdot u_d(t)dt = \int_0^{t_f} t \cdot u_d(t)dt = \left[t \cdot u(t)\right]_0^{t_f} - \int_0^{t_f} u(t)dt = \int_0^{t_f} (u_f - u(t))dt = A_3 \\ y_{do} = \int_0^\infty y_d(t)dt = \int_0^\infty y'(t)dt = y(\infty) - y(0) = y_f \\ y_{d1} = \int_0^\infty t \cdot y_d(t)dt = \int_0^{t_f} t \cdot y_d(t)dt = \left[t \cdot y(t)\right]_0^{t_f} - \int_0^{t_f} y(t)dt = \int_0^{t_f} (y_f - y(t))dt = A_2 \end{cases} \quad (47)$$

With (46) we can now calculate an estimate of T_{ar}, and this only from simple "areas" in the closed loop step and the final values y_f and u_f.

We will now look at the integrating processes for which we want to estimate $K_p(T+L)$. The basis will be a similar closed loop step. The behaviour of the process output will be the same if we have a reasonable controller, but the control signal will go back to its original value since the process is integrating. See Figure 19. We will now show that even $K_p(T+L)$ can be estimated from simple area calculations. The trick is to rewrite (43) with

$$Y(s) = G_{Int}(s)U(s) = \frac{1}{s} G_{NoInt}(s)U(s) = G_{NoInt}(s)U_p(s)$$

so that the model (38) can be expressed as model (37) and the integrated control signal u_p. For model (37) we know that

$$K_p(T+L) = -K_p \cdot \frac{G'_{NoInt}(0)}{G_{NoInt}(0)} = -G'_{NoInt}(0)$$

With the same trick as above we introduce the derivatives of y(t) and $u_p(t)$ as $y_d(t)$ and $u_{pd}(t)$. Further we use the notations u_{pd0}, u_{pd1}, y_{d0} and y_{d1} for the integrals in equation (44) and can then similar to (46) write

$$K_p(T+L) = \frac{y_{d1}u_{pd0} - y_{d0}u_{pd1}}{u_{pd0}^2} \tag{48}$$

To evaluate this expression we use the notations in Figure 19 below. The integrals that we must calculate will now be

$$\begin{cases} u_{pdo} = \int_0^\infty u_{pd}(t)dt = \int_0^\infty u_p'(t)dt = \int_0^\infty u(t)dt = A_4 \\ u_{pd1} = \int_0^\infty t \cdot u_{pd}(t)dt = \int_0^{t_f} t \cdot u_{pd}(t)dt = \left[t \cdot u_p(t)\right]_0^{t_f} - \int_0^{t_f} u_p(t)dt = \int_0^{t_f} (u_{pf} - u_p(t))dt = A_5 \\ y_{do} = \int_0^\infty y_d(t)dt = \int_0^\infty y'(t)dt = y(\infty) - y(0) = y_f \\ y_{d1} = \int_0^\infty t \cdot y_d(t)dt = \int_0^{t_f} t \cdot y_d(t)dt = \left[t \cdot y(t)\right]_0^{t_f} - \int_0^{t_f} y(t)dt = \int_0^{t_f} (y_f - y(t))dt = A_2 \end{cases} \tag{49}$$

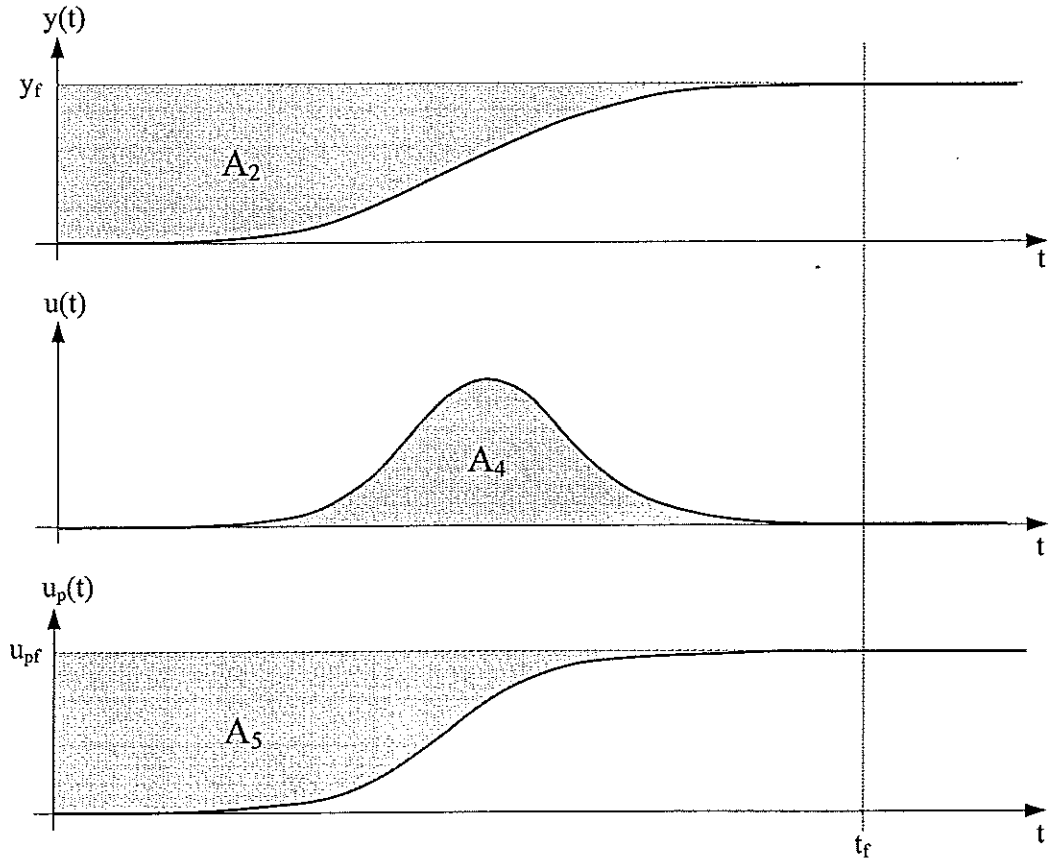

Figure 19: The closed loop step for an integrating process. The uppermost curve is the process output, and the middle is the control signal. The lowest curve is the integrated control signal. We assume that we start from zero. The areas A_2, A_4 and A_5 are indicated in the figure.

5.3 The experiment

We shall now describe the total experiment that we will make to estimate K_p, T and L for non-integrating processes and $K_p(T+L)$ for integrating processes. The experiment is shown in Figure 20 for a non-integrating process. The experiment starts with a setpoint step. This size of the step should be chosen small enough to keep the process close to the working point but large enough to be well distinguished from noise. During the closed loop step we integrate the process output y(t), and the control signal u(t) to gain the areas A_2 and A_3 shown in Figure 20. When we have reached steady state we note the final values of y(t) and u(t) (we assume y(0)=u(0)=0) and can according to (46) and (47) calculate T_{ar} as

$$T_{ar} = \frac{A_2 u_f - A_3 y_f}{y_f u_f} \tag{50}$$

The static gain is simply found from

$$K_p = \frac{y_f}{u_f} \tag{51}$$

Now we make an open loop step back by setting the control signal to the value it had before the setpoint step (u(t)=0 as we assumed u(0)=0). As earlier in Figure 17 we can now estimate the time constant T by integrating the process output until time $t=T_{ar}$. See Figure 20. By using equation (41) standardised with the size of the open loop step we find

$$T = \frac{eA_1}{K_p} \cdot \frac{1}{u_f} = eA_1 \cdot \frac{u_f}{y_f} \cdot \frac{1}{u_f} = \frac{eA_1}{y_f} \tag{52}$$

And at finally L is found from

$$L = T_{ar} - T \tag{53}$$

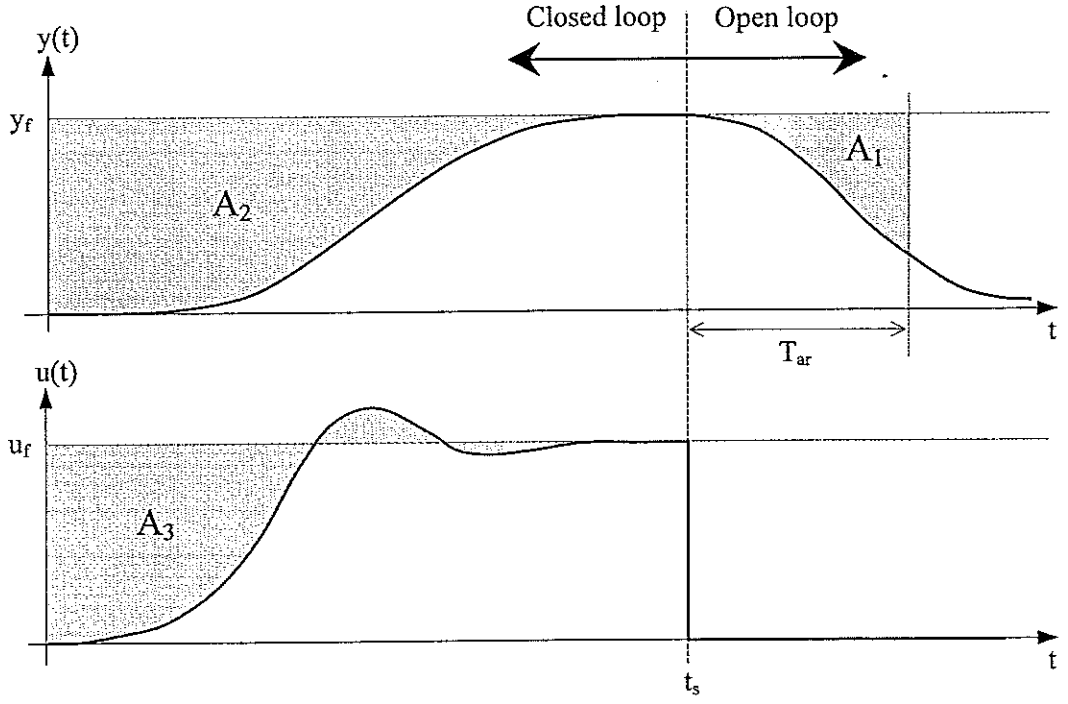

Figure 20: The total step experiment. The figure shows a non-integrating process. The experiment starts with a setpoint step. At time t_s steady state is reached and an open loop step is made. Areas A_1, A_2 and A_3 that is determined are indicated in the figure.

If the process is integrating we detect this by finding the same value of the control signal at the new steady state point. Then no open loop experiment will be necessary. The situation will be like the one in Figure 19. The area A_2 is the same as for non-integrating, but the areas A_4 and A_5 will only be used if the process is integrating. Since we do not know if the process is integrating before the experiment, we must always integrate A_4 and A_5. If the process is found to be integrating we only calculate the parameter $K_p(T+L)$ and from equation (48) and (49) it is found that

$$K_p(T+L) = \frac{A_2 A_4 - y_f A_5}{A_4^2} \tag{54}$$

5.4 The auto-tuning procedure

At this point we have all components of an automatic tuning procedure, and this last section will be a survey of the total procedure. We will start from the beginning and assume that we have no prior information of the process. First a relay auto-tuning will be made as described in chapter 2.4. This will give us the process parameters $K_{u\alpha}$ and $T_{u\alpha}$ but the design procedure in the relay auto-tuner will also give us an initial controller. With this controller we can make the closed loop step as described in this chapter and identify, most importantly, the static gain K_p. With $K_{u\alpha}$, $T_{u\alpha}$ and K_p we apply the design procedure as described in chapter three. This will also include the automatic choice of a PI controller for small values of kappa. The estimates of L and T will be used to calculate the relative dead time, tau. For large values of tau the operator can be notified and a PPI controller suggested. If the PPI controller is chosen, it can be automatically tuned according to (8) with the estimates of L and T.

As we discussed in the introduction there are some practical aspects of an auto-tuning that has to be considered. One aspect mentioned was the amount of prior information that has to be supplied by the user. In our tuning procedure no prior process information is needed. For the controller design a simple choice of performance like: fast, normal or slow, can be offered by the choice of M_s value in the tuning. Another property of the auto-tuner was the time it requires. If we want to estimate the time requirements of our procedure we must look at both the relay- and step experiment. We could roughly approximate the relay experiment to take a few periods of the limit cycles, and a period is approximately equal to a time constant of the process. For the step experiment there are two parts. The first is the closed loop step, and this will be the most time consuming since we must await steady state. After five time constants of the closed loop system we are 1% from steady state so this could be a rough estimate of the first part. The second part is the open loop step, and here we integrate under a time T_{ar} which we can be approximated to one time constant of the open loop. The total time for the step experiment would therefore be a few time constants of the process, say 6-8 time constants, if we assume approximately the same time constant for the open and closed loop. Now, this would give us a total time for the whole tuning in the region teen time constants of the process. This is actually quite long. However, an advantage of the above procedure is that it does not need to be done sequentially. After the relay experiment we might wait with next phase if we wish.

6. Conclusions

In the report we have presented a new auto-tuning procedure for PID controllers. Both the design- and identification part is based on old ideas. The identification is a well-known relay experiment plus a simple closed loop step. The controller design is a method presented in [5] that has been reconstructed with different conditions in this work. As an extra feature in the auto-tuning, large dead times are detected, and in occurring cases a dead time compensating controller can be automatically tuned.

6.1 The controller design part

The controller design is a set of empirical tuning rules based on the three process parameters K_u, T_u and κ. The rules have been developed by designing optimal controllers to a large set of representative process models. From the data material, general tuning strategies has been extracted. The optimal controller designs have been made with methods presented in [6] and [7] (PI and PID controllers respectively). These designs are based on constrained optimisation, where load disturbance rejection is optimised with constraints on robustness/stability margin.

Our conclusion of the design part would be that we can expect good tuning for the type of processes in our test batch (16), since these designs has been verified during the development of the rules. For process types other then the types found in the batch no tests have been made. However we do not claim the rules for general processes, but a rather restricted class of systems. Oscillatory systems for example are not included, and we do not expect good tuning for them. With the assumption that the batch has covered most of the process types in this class, our at least enough of various process behaviour here, we expect reasonable tuning for all those.

6.2 The identification part

Identification experiments for K_u, T_u and K_p have been suggested in the report. A specific relay experiment has been proposed for identifying the ultimate point. The specific implementation of this relay experiment leads to that a point other then the ultimate point is identified, and we show that it is rather a point 30° from the ultimate point, i.e. at phase -150°. We have concluded that this fact will affect or controller design, but in a way that gives more conservative control. It has also been pointed out that the calculations of ultimate point data from the relay experiment data only are approximations.

For identifying the static gain K_p, a simple closed loop step is proposed. The static gain is found simply from the final values of the control signal and process output. We also show how this step experiment can be used to estimate the apparent dead time L, and the apparent lag T. These parameters are calculated only from areas (zero order moments) and final values in the closed loop step response. Our conclusions from the step experiment are that we get a reliable estimate of K_p. For the estimates of T and L we can conclude that they are noise insensitive. However, as approximations to the model (37), they should be discussed further.

6.3 The auto-tuning procedure

The total result, automatic tuning, has been proposed by simply putting the identification and controller design together. In the procedure we use the relay auto-tuning as an intermediate result, making it possible to do closed loop identification. As presented the auto-tuning requires no prior process knowledge. A good tuning parameter, reflecting closed loop performance, is also provided the method via the M_s value in the design part. We have found that the time requirement for the auto-tuning we be in the region ten time constants of the process.

6.4 Future work

The most important thing that must be done is verifying the results with extensive tests and simulations. The design part should maybe verified in simulations for some other types of processes than those in the batch (16). It should also be interesting to examine the result for systems not claimed by the rules, e.g. oscillatory systems. Next step would be to test the total auto-tuning, including the both identification experiments. The tuning should be verified for as many different types of systems as possible. This can preferably be done in simulations first, and a good starting point would be some of the systems in our batch (16). These tests would revile effects of the approximations made in the describing function analysis, which is the basis for calculating ultimate point parameters. They would also show the effects of not identifying the true ultimate point, due to hysteresis in the relay. The last part in testing should of course also include some verification on real plants.

7. References

[1] ÅSTRÖM, K. J. and T. HÄGGLUND (1995): *PID Controllers: Theory, Design and Tuning.* Instrument Society of America, Research Triangle Park, North Carolina, second edition.

[2] HÄGGLUND, T. (1996): "An industrial dead-time compensating PI controller." *Control Engineering Practice*, **4**, pp. 749-756.

[3] HÄGGLUND, T. and K. J. ÅSTRÖM (1991): "Industrial adaptive controllers based on frequency response techniques." *Automatica*, **27**, pp. 599-609.

[4] HANG, C. C., K. J. ÅSTRÖM and W. K. HO (1991): "Refinements of the Ziegler-Nichols tuning formula." *IEE Proceedings, Part D*, **138**, pp. 111-118.

[5] ÅSTRÖM, K. J. and T. HÄGGLUND (1995): "New tuning methods for PID controllers." *European Control Conference*, Rome. pp. 2456-2462.

[6] ÅSTRÖM, K. J., H. PANAGOPOULOS and T. HÄGGLUND (1998): "Design of PI controllers based on non-convex optimization." *Automatica*, **34**, pp. 585-601.

[7] PANAGOPOULOS, H., K. J. ÅSTRÖM and T. HÄGGLUND (1998): "Design of PID controllers based on constrained optimization." *Internal report*, Department of Automatic Control, Lund Institute of Technology, S-221 00 Lund, Sweden.

[8] LENNARTSON, B. and B. KRISTIANSSON (1997): "Pass band and high frequency robustness for PID control." *Conference on Decision and Control*, San Diego. pp. 291-299.

[9] KRISTIANSSON, B. and B. LENNARTSON (1998): "Robust design of PID controllers including auto-tuning rules." *Internal report*, Control Engineering Lab, Chalmers University of Technology, S-412 96 Gothenburg, Sweden.

[10] SLOTINE, E. and W. LI (1991): *Applied Nonlinear Control.* Prentice-Hall, Englewood Cliffs ,New Jersey.

Appendix

Appendix A – More diagrams of kappa functions

A.1)

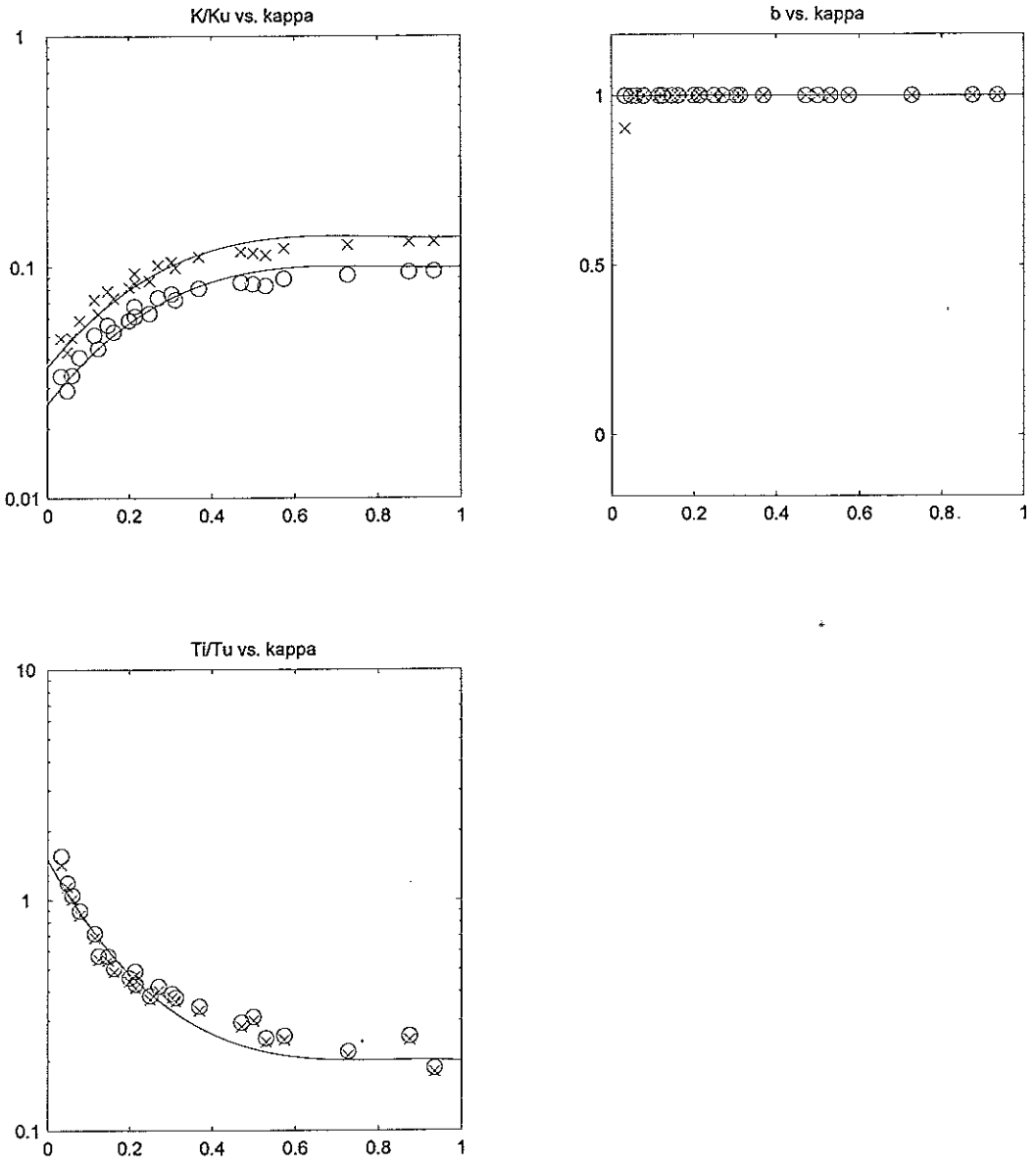

Figure A.1: Tuning diagram for a PI design with M_s=1.2 marked with o, and M_s=1.3 marked with x. Note the similarity with Figure 6 (left) on page 19, that shows M_s=1.4 and 2.0. The function $f_{Ti}(\kappa)$ is identical for all four M_s values. The function $f_K(\kappa)$ differ approximately only with a scaling factor. The b factor can be set to one for most designs except for M_s=2.0 where b decreases with increasing kappa (see Figure 6).

A.2)

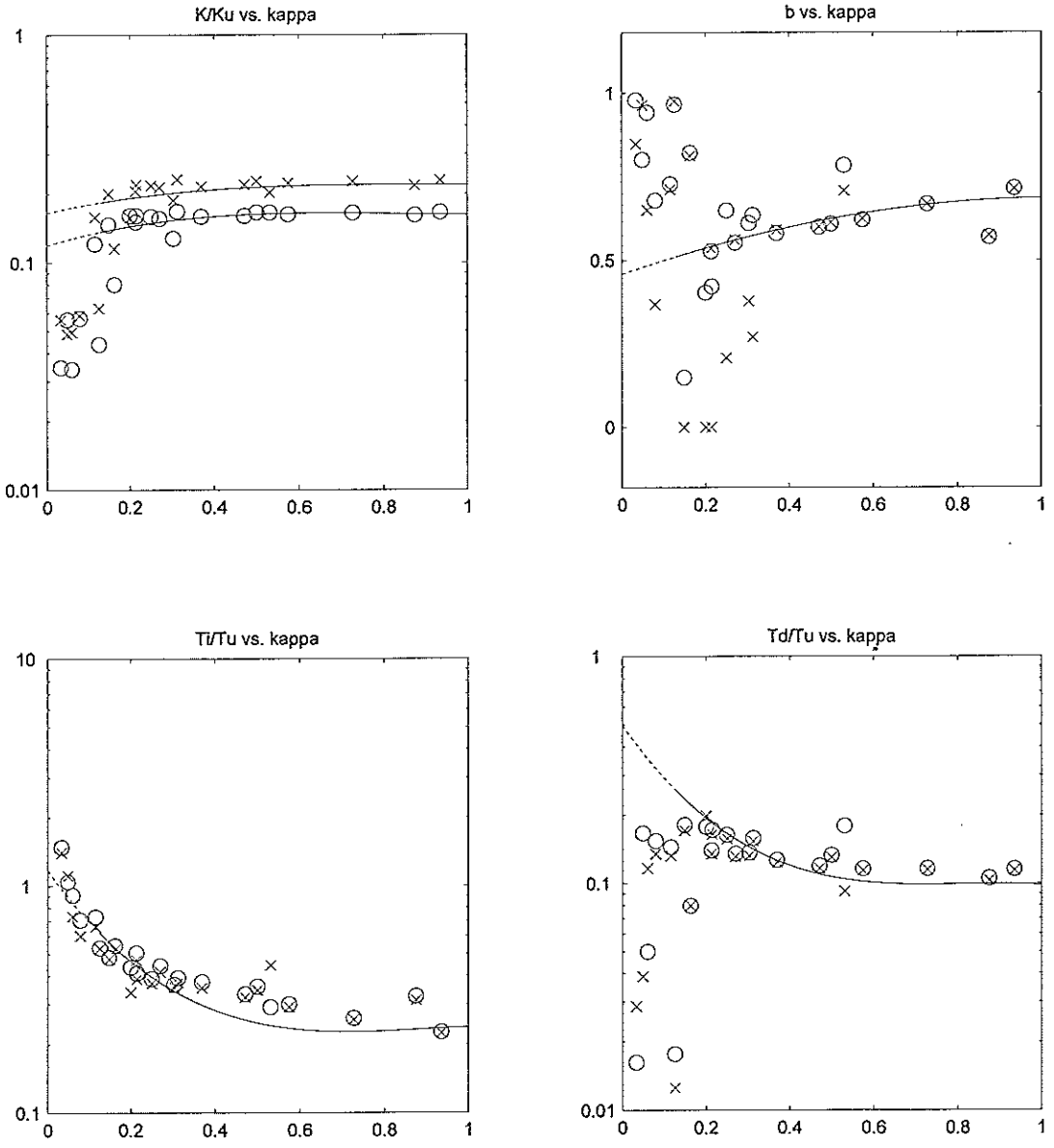

Figure A.2: The results from a PID design with M_s=1.2 marked with o, and M_s=1.3 marked with x. Please compare it with Figure 6 (right) on page 19, that shows M_s=1.4 and 2.0. The functions $f_{Ti}(\kappa)$ and $f_{Td}(\kappa)$ is identical for all four M_s values. The behaviour of the b factor above is hard to approximate, but some trends can be found for larger kappa. In appendix B, Table B.1, the kappa functions in this figure and Figure A.1 above can be found explicitly.

A.3)

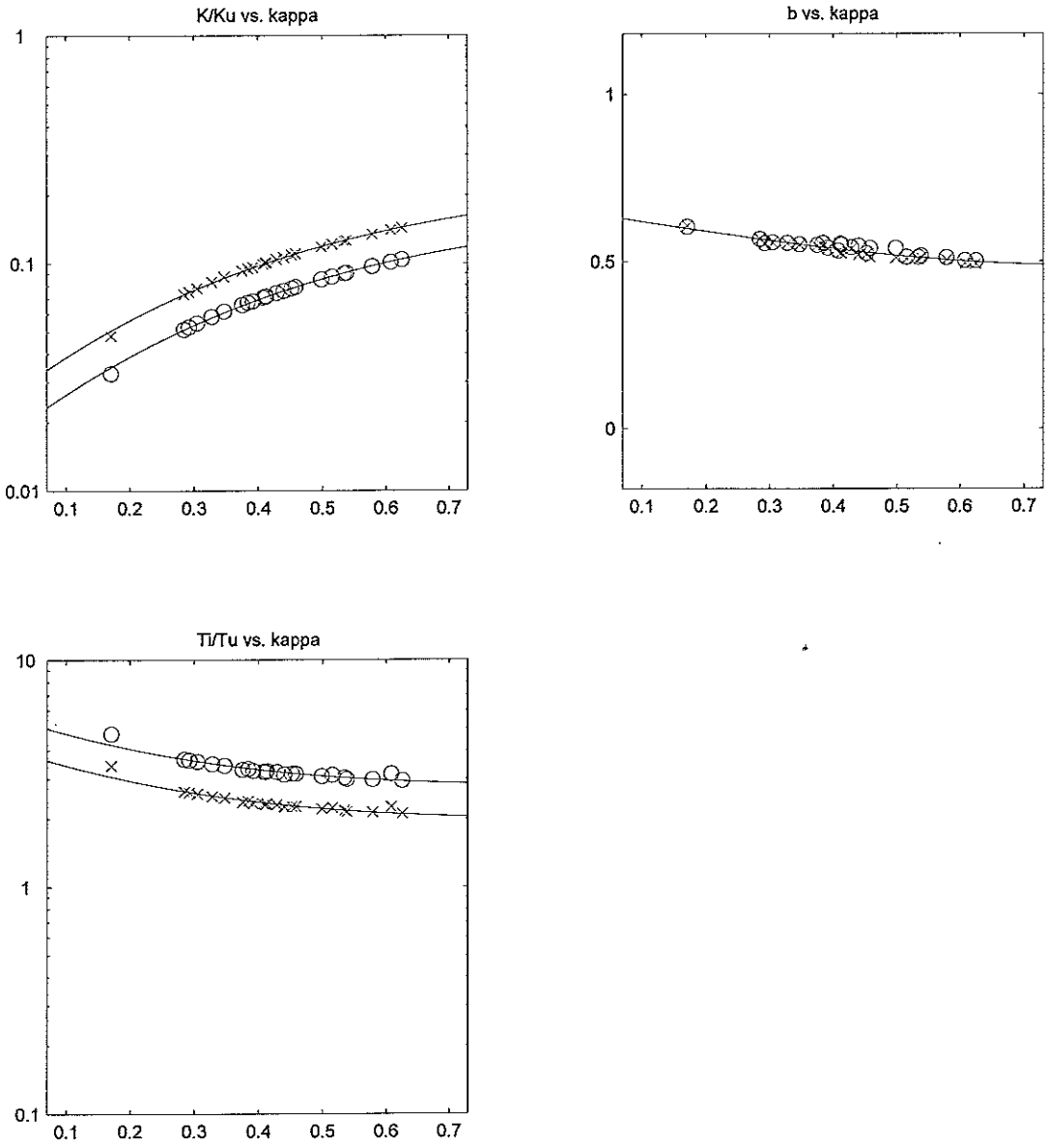

Figure A.3: Tuning diagram, PI design, for integrating processes with M_s=1.2 marked with o, and M_s=1.3 marked with x. The M_s values 1.4 and 2.0 is found on page 24 in Figure 8 (left). The function $f_{Ti}(\kappa)$ is not identical for all four M_s values as it is for non-integrating processes. Note that the kappa range is much smaller now as compared to non-integrating processes.

A.4)

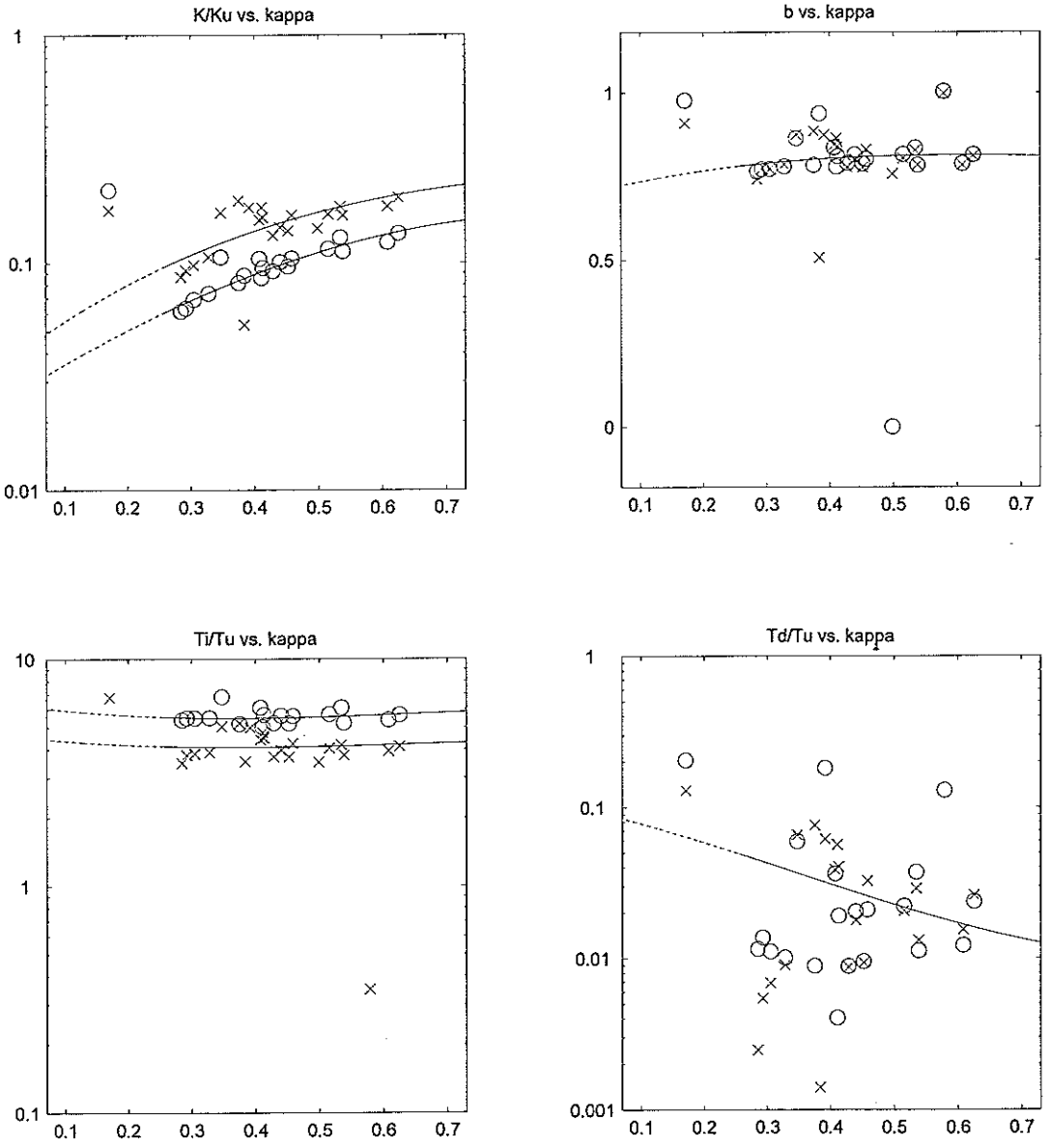

Figure A.4: Tuning diagram, PID design, for integrating processes with M_s=1.2 marked with o, and M_s=1.3 marked with x. The M_s values 1.4 and 2.0 is found on page 24 in Figure 8 (right). The function $f_{Ti}(\kappa)$ is not identical for all four M_s values as it is for non-integrating processes. Just like the design with M_s=1.4 and 2.0, the quotient T_d/T_u is very scattered. The kappa function $f_{Td}(\kappa)$ is the same as for M_s=1.4 and 2.0.

A.5)

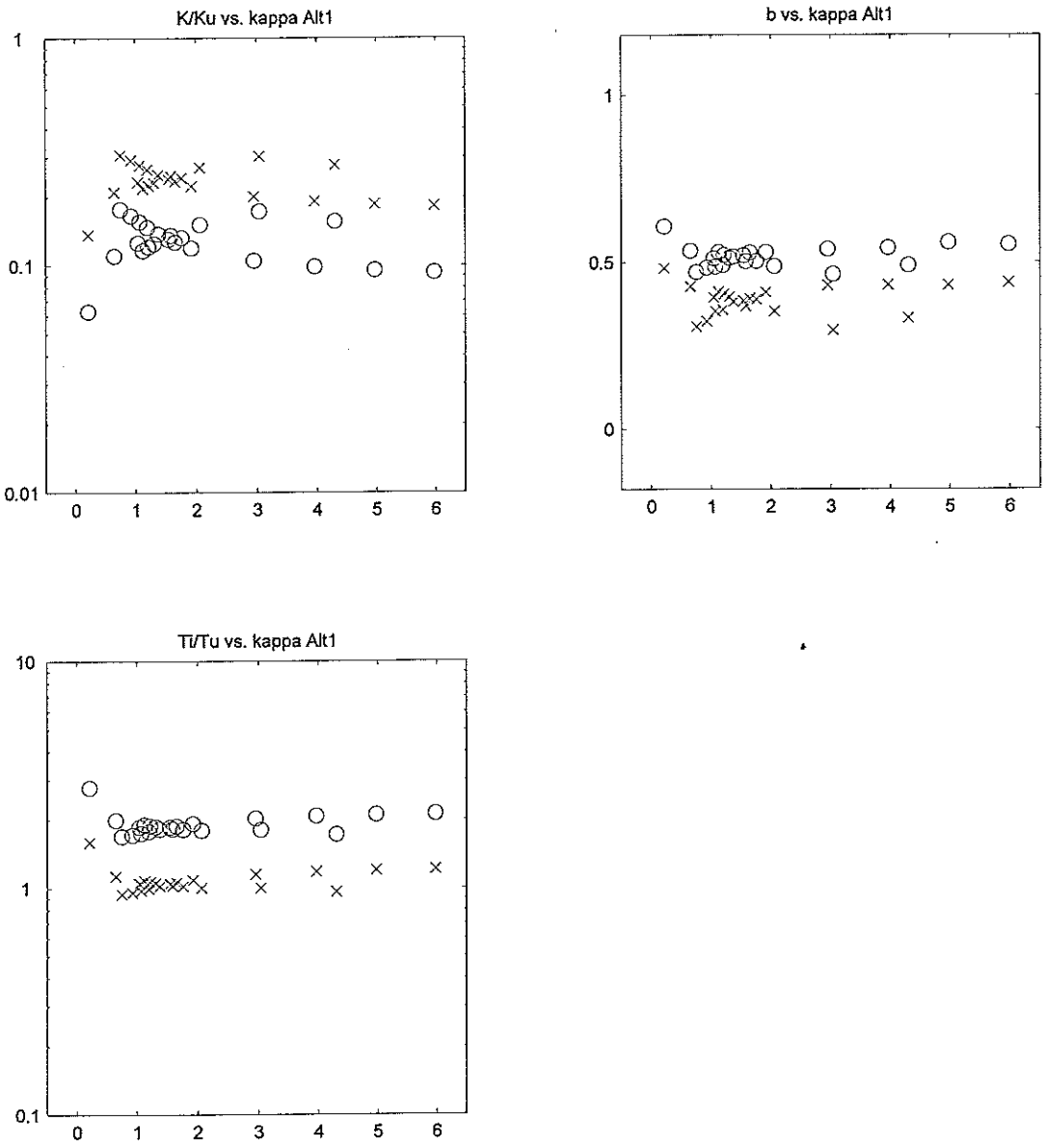

Figure A.5: The kappa functions for integrating processes, PI design. This diagram shows an alternative kappa definition where the gain at the ultimate point is divided with the static gain of the process without integrator. For comparison the real kappa definition we have used is shown in Figure A.7. Note that the range of kappa is 0 – 6.

A.6)

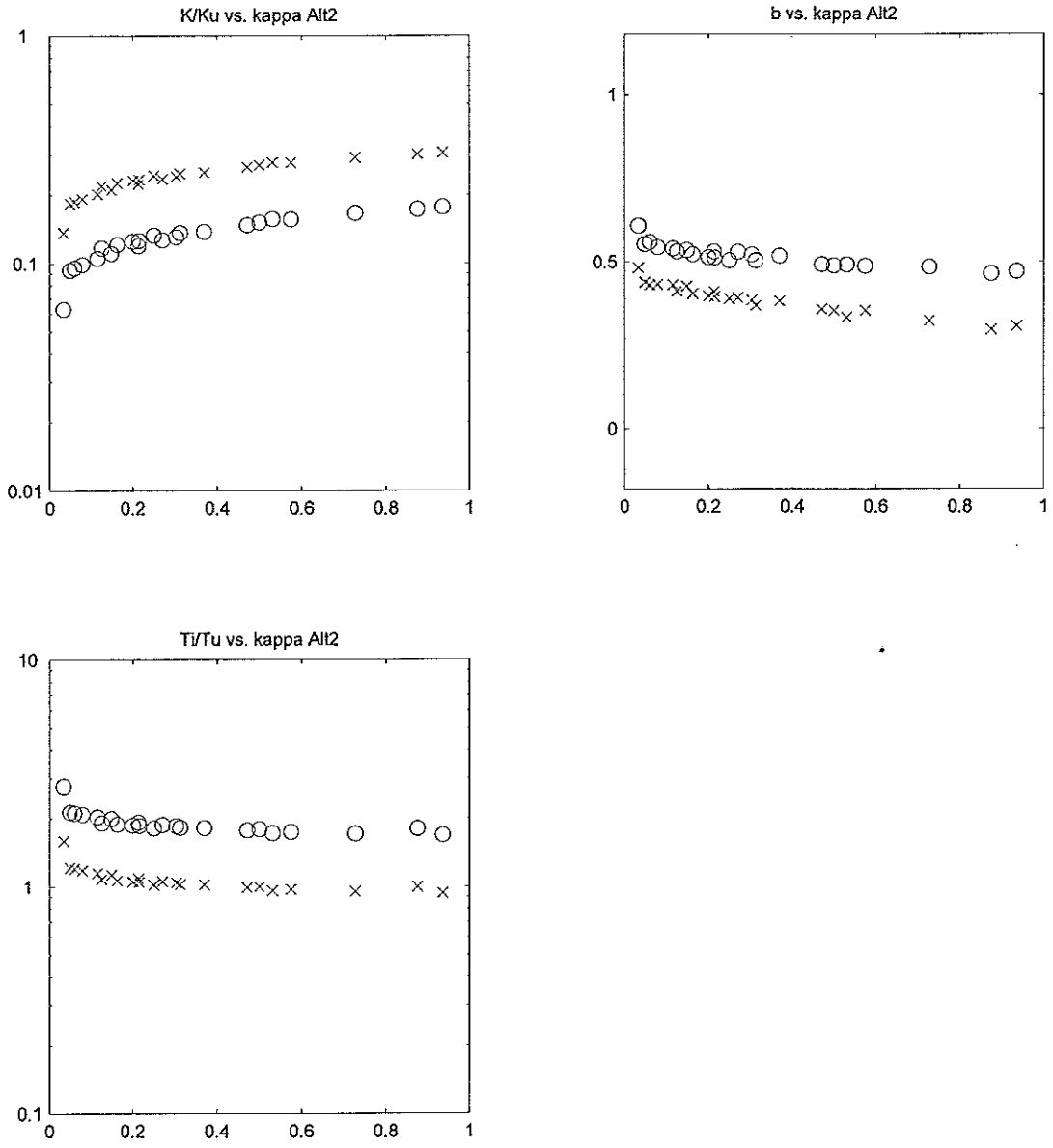

Figure A.6: The kappa functions for integrating processes, PI design. This diagram shows an alternative kappa definition where we used the kappa value for the process without integrator. For comparison the real kappa definition we have used is shown in Figure A.7. Note that the range of kappa is now 0 – 1.

A.7)

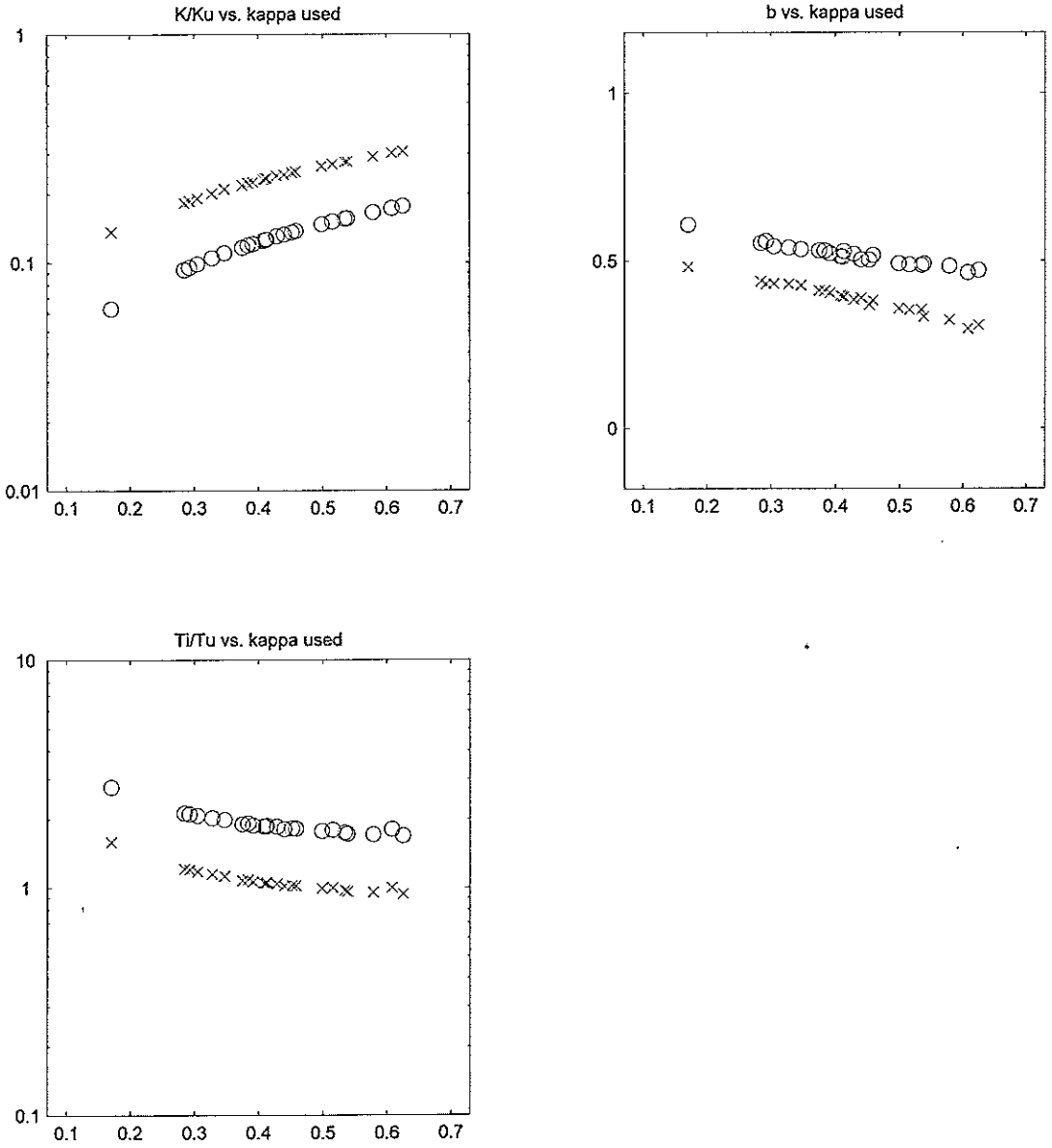

Figure A.7: The kappa functions for integrating processes, PI design. This diagram shows the real kappa definition used in this report. It is the same diagram as Figure 8 (left) on page 24. It should be compared to Figure A.5 and Figure A.6 above.

Appendix B – Kappa function tables

$f_{K,Ti,Td}(\kappa) = a_0 \cdot e^{(a_1\kappa + a_2\kappa^2 + a_3\kappa^3)}$ $\qquad$ $f_b(\kappa) = a_0 + a_1\kappa + a_2\kappa^2 + a_3\kappa^3$

		PI					PID			
		a_0	a_1	a_2	a_3		a_0	a_1	a_2	a_3
$f_K(\kappa)$ K / K_u	M_s=1.2	0.0255	5.24	-6.53	2.67		0.119	1.25	-1.52	0.580
	M_s=1.3	0.0368	5.02	-6.29	2.57		0.166	0.918	-0.961	0.324
	M_s=1.4	0.0510	4.43	-5.59	2.33		0.169	2.32	-3.30	1.48
	M_s=2.0	0.108	3.43	-4.16	1.66		0.283	1.91	-2.34	0.938
$f_b(\kappa)$ b	M_s=1.2	1.00	0.00	0.00	0.00		0.459	0.426	-0.182	-0.0164
	M_s=1.3	1.00	0.00	0.00	0.00		0.459	0.426	-0.182	-0.0164
	M_s=1.4	1.00	0.00	0.00	0.00		0.123	0.256	1.24	-0.932
	M_s=2.0	0.641	-2.47	3.14	-1.31		0.00	0.00	0.00	0.00
$f_{Ti}(\kappa)$ T_i / T_u	M_s=1.2	1.50	-7.36	8.95	-3.60		1.17	-5.89	6.87	-2.57
	M_s=1.3	1.50	-7.36	8.95	-3.60		1.17	-5.89	6.87	-2.57
	M_s=1.4	1.50	-7.36	8.95	-3.60		1.17	-5.89	6.87	-2.57
	M_s=2.0	1.50	-7.36	8.95	-3.60		1.17	-5.89	6.87	-2.57
$f_{Td}(\kappa)$ T_d / T_u	M_s=1.2	-	-	-	-		0.499	-6.12	7.64	-3.15
	M_s=1.3	-	-	-	-		0.499	-6.12	7.64	-3.15
	M_s=1.4	-	-	-	-		0.499	-6.12	7.64	-3.15
	M_s=2.0	-	-	-	-		0.499	-6.12	7.64	-3.15

Table B.1: Tuning formulas for PI and PID controllers. The parameters for the kappa functions are presented at four different M_s values. Note that in some cases only one kappa function is used for many M_s values. See corresponding kappa plots for further insight.

$f_{K,Ti,Td}(\kappa) = a_0 \cdot e^{(a_1\kappa + a_2\kappa^2 + a_3\kappa^3)}$ $\qquad$ $f_b(\kappa) = a_0 + a_1\kappa + a_2\kappa^2 + a_3\kappa^3$

		PI					PID			
		a_0	a_1	a_2	a_3		a_0	a_1	a_2	a_3
$f_K(\kappa)$ K / K_u	M_s=1.2	0.0168	4.94	-3.97	1.17		0.0246	3.87	-1.35	-0.717
	M_s=1.3	0.0247	4.88	-4.16	1.38		0.0355	4.80	-3.94	1.08
	M_s=1.4	0.0344	4.43	-3.58	1.14		0.0383	5.75	-5.33	1.44
	M_s=2.0	0.0759	4.12	-4.07	1.68		0.0773	4.29	-1.97	-0.572
$f_b(\kappa)$ b	M_s=1.2	0.651	-0.311	0.0238	0.137		0.693	0.484	-0.609	0.229
	M_s=1.3	0.651	-0.311	0.0238	0.137		0.693	0.484	-0.609	0.229
	M_s=1.4	0.691	-0.605	0.540	-0.225		0.595	0.835	-1.04	0.410
	M_s=2.0	0.547	-0.324	-0.152	0.0788		0.247	1.34	-1.32	0.430
$f_{Ti}(\kappa)$ T_i / T_u	M_s=1.2	5.74	-2.13	2.13	-0.713		6.46	-1.02	1.93	-0.985
	M_s=1.3	4.15	-2.12	2.05	-0.654		4.65	-0.790	1.48	-0.759
	M_s=1.4	3.99	-3.18	4.00	-1.74		3.90	-1.65	3.99	-2.79
	M_s=2.0	2.30	-3.17	3.72	-1.49		1.95	-2.65	6.64	-4.26
$f_{Td}(\kappa)$ T_d / T_u	M_s=1.2	-	-	-	-		0.0995	-2.17	-3.02	2.90
	M_s=1.3	-	-	-	-		0.0995	-2.17	-3.02	2.90
	M_s=1.4	-	-	-	-		0.0995	-2.17	-3.02	2.90
	M_s=2.0	-	-	-	-		0.0995	-2.17	-3.02	2.90

Table B.2: Tuning formulas for PI and PID controllers for integrating processes.

Appendix C – Some of the Matlab routines used

C.1)

```
function [Procvect,ProcvectInt] = procgen

% Genererar processerna och lagger dem i vektorer (med och utan integratorer)
% [procvect,procvectInt] = procgen;

Procvect = {};
%Type 1
T = [0.1 0.3 0.5 0.7 1.0 1.3 1.5 2 4 6 8 10];
for i = 1:12,
  G = tf([1],[T(i)^2,2*T(i),1],'Td',1);
  Procvect = {Procvect{:},G};
end
%Type 2
  Gn=tf([1],[1 1]);
  G3=Gn*Gn*Gn;
  G4=Gn*Gn*Gn*Gn;
  G8=Gn*Gn*Gn*Gn*Gn*Gn*Gn*Gn;
  Procvect = {Procvect{:},G3};
  Procvect = {Procvect{:},G4};
  Procvect = {Procvect{:},G8};
%Type 3
  alfa = [0.2 0.5 0.7];
  for i=1:3,
    G = tf([1],conv(conv([1 1],[alfa(i) 1]),conv([alfa(i)^2 1],[alfa(i)^3 1])));
    Procvect = {Procvect{:},G};
  end
%Type 4
  alfa = [0.1 0.2 0.5 1 2];
  for i=1:5,
    G = tf([-alfa(i) 1],conv(conv([1 1],[1 1]),[1 1]));
     Procvect = {Procvect{:},G};
  end

%------------ med integratorer-----------

ProcvectInt = {};
%Type 1
T = [0.1 0.3 0.5 0.7 1.0 1.3 1.5 2 4 6 8 10];
for i = 1:12,
  G = tf([1],conv([1 0],[T(i)^2,2*T(i),1]),'Td',1);
  ProcvectInt = {ProcvectInt{:},G};
end
%Type 2
  Gn=tf([1],[1 1]);
  Int=tf([1],[1 0]);
  G3=Int*Gn*Gn*Gn;
  G4=Int*Gn*Gn*Gn*Gn;
  G8=Int*Gn*Gn*Gn*Gn*Gn*Gn*Gn*Gn;
  ProcvectInt = {ProcvectInt{:},G3};
  ProcvectInt = {ProcvectInt{:},G4};
  ProcvectInt = {ProcvectInt{:},G8};
%Type 3
  alfa = [0.2 0.5 0.7];
  for i=1:3,
    G = tf([1],conv([1 0],conv(conv([1 1],[alfa(i) 1]),conv([alfa(i)^2
1],[alfa(i)^3 1]))));
    ProcvectInt = {ProcvectInt{:},G};
  end
%Type 4
  alfa = [0.1 0.2 0.5 1 2];
  for i=1:5,
    G = tf([-alfa(i) 1],conv([1 0],conv(conv([1 1],[1 1]),[1 1])));
     ProcvectInt = {ProcvectInt{:},G};
  end
```

C.2)

```
function [Ku,Tu,kappa]=calcFreqData(Gp)
```

```
% Frekvenssvars data
% [Ku,Tu,kappa]=calcFreqData(Gp);
% Parametrar enligt def. artikel "New tuning methods........"

% wu ungefarligt lage
wapprox = 0:0.1:20;
[Magn,Phase]=bode(Gp,wapprox);
wuapprox = wapprox(min(find(Phase(:)<=-180)));   %Phase ar en 3-dim matris

% Storre precision
wstart = wuapprox -0.1;
wstop = wuapprox;
w = wstart:0.00001:wstop;
[Magn,Phase]=bode(Gp,w);
index = min(find(Phase(:)<=-180));
wu = w(index);

% Parametrar enligt def.
Ku = 1/Magn(index);
Tu = (2*pi)/wu;
kappa = 1/Ku;   % Statisk forstarkning = ett
```

C.3)

```
function [L,T,tau,a]=calcStepData(Gp)

% Stegsvars data
% [L,T,tau,a]=calcStepData(Gp)
% Parametrar enligt def. artikel "New tuning methods........"



% Hitta ungefarligt lage pa maxder forst
tapprox = 0:0.1:50;
yimp = impulse(Gp,tapprox);
[maxder,index] = max(yimp);
tstart = tapprox(index)-0.1;
tstop = tapprox(index)+0.1;
% Laget med battre precision
t = tstart:0.0001:tstop;
yimp = impulse(Gp,t);
ystep = step(Gp,t);
[maxder,index] = max(yimp);
L = t(index) - ystep(index)/maxder;      %Tangentens skarning med x-axel

% Ungefarligt lage 63%-niva
ystep = step(Gp,tapprox);
T63 = tapprox(min(find(ystep>=(1-exp(-1)))));      %Tid da stegsvar 63%
tstart = T63-0.1;
tstop = T63;
% Battre precision
t = tstart:0.0001:tstop;
ystep = step(Gp,t);
T63 = t(min(find(ystep>=(1-exp(-1)))));      %Tid da stegsvar 63%

% Berakna parametrar
T = T63-L;
tau = L/(L+T);
a = 1*L/T;                               %Statisk forstarkning = ett
```

C.4)

```
function [datavect]=datagen(procvect,Ms1,Ms2)

% Obs! Specialfall processer 8,15,19,2 (se kod)
% Genererar strukturerad datavektor med data for tva olika Ms-varde
% [datavect]=datagen(procvect,Ms1,Ms2)


datavect = {};       %Tom cellarray

for i = 1:size(procvect,2),

  Gp = procvect{i};

% Process data
```

```
  [L,T,tau,a]=calcStepData(Gp);
  [Ku,Tu,kappa]=calcFreqData(Gp);
  process = struct('L',L,'T',T,'tau',tau,'a',a,'Ku',Ku,'Tu',Tu,'kappa',kappa);

% Regulator parametrar for forsta Ms-vardet
  [K,Ti,b,wMs]=optpi(Gp.num{1},Gp.den{1},Gp.Td,Ms1);
  piMs1 = struct('K',K,'Ti',Ti,'b',b);
  PIpar.K=K;PIpar.Ti=Ti;PIpar.wMs=wMs;
% Special
%  if i == 8,[PIpar.K,PIpar.Ti,PIpar.b,PIpar.wMs] =
optpi(Gp.num{1},Gp.den{1},Gp.Td,1.3);end,
%  if i == 15,[PIpar.K,PIpar.Ti,PIpar.b,PIpar.wMs] =
optpi(Gp.num{1},Gp.den{1},Gp.Td,2.2);end,
%  if i == 19,[PIpar.K,PIpar.Ti,PIpar.b,PIpar.wMs] =
optpi(Gp.num{1},Gp.den{1},Gp.Td,2.75);end,
%  if i == 9,[PIpar.K,PIpar.Ti,PIpar.b,PIpar.wMs] =
optpi(Gp.num{1},Gp.den{1},Gp.Td,1.9);end,

  [K,Ti,Td,b,wMs]=optpid(Gp.num{1},Gp.den{1},Gp.Td,Ms1,PIpar);
  pidMs1 = struct('K',K,'Ti',Ti,'Td',Td,'b',b);

% Med det andra Ms-vardet
  [K,Ti,b,wMs]=optpi(Gp.num{1},Gp.den{1},Gp.Td,Ms2);
  piMs2 = struct('K',K,'Ti',Ti,'b',b);
  PIpar.K=K;PIpar.Ti=Ti;PIpar.wMs=wMs;
% Special
%  if i == 10,[PIpar.K,PIpar.Ti,PIpar.b,PIpar.wMs] =
optpi(Gp.num{1},Gp.den{1},Gp.Td,1.9);end,
%  if i == 11,[PIpar.K,PIpar.Ti,PIpar.b,PIpar.wMs] =
optpi(Gp.num{1},Gp.den{1},Gp.Td,1.9);end,
%  if i == 12,[PIpar.K,PIpar.Ti,PIpar.b,PIpar.wMs] =
optpi(Gp.num{1},Gp.den{1},Gp.Td,1.9);end,

  [K,Ti,Td,b,wMs]=optpid(Gp.num{1},Gp.den{1},Gp.Td,Ms2,PIpar);
  pidMs2 = struct('K',K,'Ti',Ti,'Td',Td,'b',b);

% Lagg in allt i en record
  Data =
struct('process',process,'piMs1',piMs1,'pidMs1',pidMs1,'piMs2',piMs2,'pidMs2',pidM
s2);

% Lagg in i datavektorn
  datavect = {datavect{:},Data};
end
```

C.5)

```
function [datavectInt]=datagenInt(procvect,procvectInt,Ms1,Ms2)


% Genererar strukturerad datavektor med data for tva olika Ms-varde
% [datavectInt]=datagenInt(procvect,procvectInt,Ms1,Ms2)


datavectInt = {};       %Tom cellarray

for i = 1:size(procvectInt,2),

% Prim betyder storheter for processen utan integrator

% Process data
  [Lprim,Tprim,tauprim,aprim]=calcStepData(procvect{i});
  L = Lprim + Tprim;                                    % se def. sid 205
  %T existerar ej for Int. proc.                        % PID - boken
  tau = tauprim;
  a = 1*(Lprim + Tprim);
  [Ku,Tu,kappa]=calcFreqData(procvectInt{i});           % Samma som icke
Int.proc.
  process = struct('L',L,'tau',tau,'a',a,'Ku',Ku,'Tu',Tu,'kappa',kappa);

% Regulator parametrar for forsta Ms-vardet
  [K,Ti,b,wMs,Go,Gc]=optimalpi(procvectInt{i},Ms1);
  piMs1 = struct('K',K,'Ti',Ti,'b',b);
  [K,Ti,Td,b,wMs,Go,Gc]=optimalpid2(procvectInt{i},Ms1);
  pidMs1 = struct('K',K,'Ti',Ti,'Td',Td,'b',b);

% Med det andra Ms-vardet
```

```
  [K,Ti,b,wMs,Go,Gc]=optimalpi(procvectInt{i},Ms2);
  piMs2 = struct('K',K,'Ti',Ti,'b',b);
  [K,Ti,Td,b,wMs,Go,Gc]=optimalpid2(procvectInt{i},Ms2);
  pidMs2 = struct('K',K,'Ti',Ti,'Td',Td,'b',b);

% Lagg in allt i en record
  Data =
struct('process',process,'piMs1',piMs1,'pidMs1',pidMs1,'piMs2',piMs2,'pidMs2',pidM
s2);

% Lagg in i datavektorn
  datavectInt = {datavectInt{:},Data};
end
```

C.6)

```
function [PIvstau,PIDvstau,PIvskappa,PIDvskappa] = plotdata(datavect)


% Genererar plotdata.I utdata matriserna ar varje rad en kurva
% [PIvstau,PIDvstau,PIvskappa,PIDvskappa] = plotdata(datavect);

PIvstau = [];
PIDvstau = [];
PIvskappa = [];
PIDvskappa = [];

for i = 1:size(datavect,2)

  D=datavect{i};
  a=D.process.a; L=D.process.L; T=D.process.T; tau=D.process.tau;

% PI-regulator vs. tau
  pivect = [a*D.piMs1.K  a*D.piMs2.K  D.piMs1.b  D.piMs2.b  D.piMs1.Ti/L
D.piMs2.Ti/L  D.piMs1.Ti/T  D.piMs2.Ti/T  tau]';
  PIvstau = [PIvstau pivect];

% PID-regulator vs. tau
  pidvect = [a*D.pidMs1.K  a*D.pidMs2.K  D.pidMs1.b  D.pidMs2.b  D.pidMs1.Ti/L
D.pidMs2.Ti/L  D.pidMs1.Ti/T  D.pidMs2.Ti/T  D.pidMs1.Td/L  D.pidMs2.Td/L
D.pidMs1.Td/T  D.pidMs2.Td/T  tau]';
  PIDvstau = [PIDvstau pidvect];

  Ku=D.process.Ku; Tu=D.process.Tu; kappa=D.process.kappa;

% PI-regulator vs. kappa
  pivect = [D.piMs1.K/Ku  D.piMs2.K/Ku  D.piMs1.b  D.piMs2.b  D.piMs1.Ti/Tu
D.piMs2.Ti/Tu  kappa]';
  PIvskappa = [PIvskappa pivect];

% PID-regulator vs. kappa
  pidvect = [D.pidMs1.K/Ku  D.pidMs2.K/Ku  D.pidMs1.b  D.pidMs2.b  D.pidMs1.Ti/Tu
D.pidMs2.Ti/Tu  D.pidMs1.Td/Tu  D.pidMs2.Td/Tu  kappa]';
  PIDvskappa = [PIDvskappa pidvect];

end
```

C.7)

```
function [PIvstau,PIDvstau,PIvskappa,PIDvskappa] = plotdataInt(datavect)


% Genererar plotdata.I utdata matriserna ar varje rad en kurva
% [PIvstauInt,PIDvstauInt,PIvskappaInt,PIDvskappaInt] = plotdataInt(datavectInt);

PIvstau = [];
PIDvstau = [];
PIvskappa = [];
PIDvskappa = [];

for i = 1:size(datavect,2)

  D=datavect{i};
  a=D.process.a; L=D.process.L; tau=D.process.tau;

% PI-regulator vs. tau
```

```
  pivect = [a*D.piMs1.K  a*D.piMs2.K  D.piMs1.b  D.piMs2.b  D.piMs1.Ti/L
D.piMs2.Ti/L  tau]';
  PIvstau = [PIvstau pivect];

% PID-regulator vs. tau
  pidvect = [a*D.pidMs1.K  a*D.pidMs2.K  D.pidMs1.b  D.pidMs2.b  D.pidMs1.Ti/L
D.pidMs2.Ti/L  D.pidMs1.Td/L  D.pidMs2.Td/L  tau]';
  PIDvstau = [PIDvstau pidvect];

  Ku=D.process.Ku; Tu=D.process.Tu; kappa=D.process.kappa;

% PI-regulator vs. kappa
  pivect = [D.piMs1.K/Ku  D.piMs2.K/Ku  D.piMs1.b  D.piMs2.b  D.piMs1.Ti/Tu
D.piMs2.Ti/Tu  kappa]';
  PIvskappa = [PIvskappa pivect];

% PID-regulator vs. kappa
  pidvect = [D.pidMs1.K/Ku  D.pidMs2.K/Ku  D.pidMs1.b  D.pidMs2.b  D.pidMs1.Ti/Tu
D.pidMs2.Ti/Tu  D.pidMs1.Td/Tu  D.pidMs2.Td/Tu  kappa]';
  PIDvskappa = [PIDvskappa pidvect];

end
```

C.8)

```
function [A]=showplotLine(Matrix,LineMatrix)

% Plottar data matrisen, Matrix, genererad av plotdata
% Kurvanpassning till data i matrisen LineMatrix
% [A]=showplotLine(Matrix,LineMatrix);

Lastrow = size(Matrix,1);
NbrOfFig = 0.5 * (size(Matrix,1) - 1);
clf;
figure(1)
x1=[0:0.001:0.13];
x2=[0.13:0.001:1];
A=zeros(2*NbrOfFig,4);

for i = 1:NbrOfFig
 if i == 2   % b plott
   subplot(ceil(0.5*NbrOfFig),2,i);
   plot(Matrix(Lastrow,:),Matrix(2*i-1,:),'ko'); hold on;
   A(2*i-1,:)=polyfit(LineMatrix(Lastrow,:),LineMatrix(2*i-1,:),3);
   plot(x1,polyval(A(2*i-1,:),x1),'k:');
   plot(x2,polyval(A(2*i-1,:),x2),'k');
   plot(Matrix(Lastrow,:),Matrix(2*i,:),'kx');
   A(2*i,:)=polyfit(LineMatrix(Lastrow,:),LineMatrix(2*i,:),3);
   plot(x1,polyval(A(2*i,:),x1),'k:');
   plot(x2,polyval(A(2*i,:),x2),'k');hold off;
 else   % resterande i log skala
   subplot(ceil(0.5*NbrOfFig),2,i)
   semilogy(Matrix(Lastrow,:),Matrix(2*i-1,:),'ko'); hold on;
   A(2*i-1,:)=polyfit(LineMatrix(Lastrow,:),log(LineMatrix(2*i-1,:)),3);
   semilogy(x1,exp(polyval(A(2*i-1,:),x1)),'k:');
   semilogy(x2,exp(polyval(A(2*i-1,:),x2)),'k');
   semilogy(Matrix(Lastrow,:),Matrix(2*i,:),'kx');
   A(2*i,:)=polyfit(LineMatrix(Lastrow,:),log(LineMatrix(2*i,:)),3);
   semilogy(x1,exp(polyval(A(2*i,:),x1)),'k:');
   semilogy(x2,exp(polyval(A(2*i,:),x2)),'k');hold off;
 end;
end
A(:,4)=exp(A(:,4));A(3,4)=log(A(3,4));A(4,4)=log(A(4,4));
```