On the design of digital motor controllers with low sampling rates

Cloudt, R.P.M.

Award date:
2004

Eindhoven University of Technology
Department of Electrical Engineering
Control Systems

Océ-Technologies B.V.
Research & Development
Group Research 3

On the Design of Digital Motor Controllers with Low Sampling Rates

by R.P.M. Cloudt

Master of Science thesis

Project period: January 2004 – October 2004
Report Number: 04A/07
Commissioned by: Prof.dr.ir. P.P.J. van den Bosch

Supervisors:
Dr.ir. A.A.H. Damen (TU Eindhoven)
Ir. P.F.A. van den Bosch (Océ-Technologies B.V.)

Additional Commission members:
Prof.dr.ir. P.P.J. van den Bosch (TU Eindhoven)
Dr.ir. M.J.G. van de Molengraft (TU Eindhoven)

Abstract

The design of digital DC motor controllers is considered in this report. A structured manner of designing motor controllers for different sampling rates is presented. The proposed design method allows to trade-off performance against a low sampling rate. The method involves a controller optimisation procedure which regards the continuous-time performance of the mixed discrete-time/continuous-time control system. The independence of the criteria on the sampling rate allows a fair comparison of controllers with different sampling rates.
An extensive analysis of sampled-data control systems (control systems with a discrete-time controller and a continuous-time plant) is provided. It exposes capabilities and limitations of sampled-data control. Computable frequency gains are being used to study the response to arbitrary signals, whereas continuous-time tracking errors quantify the tracking performance with respect to prespecified setpoint profiles. The intuitive frequency gains are combined with an integral quadratic tracking error criterion in an optimisation algorithm. The proposed controller optimisation procedure is applied to a representative motor control case in an Océ copier.

Preface

The sampling rate selection of digital controllers poses a problem. Rules of the thumb for choosing the sampling rate exists, but few systematic approaches are known. In designing DC motor controllers, often a sampling frequency is stipulated, which may or may not be based on former experience of an engineer. The controller is then designed using continuous-time techniques, and it is implemented digitally by means of an approximation technique which emulates the continuous-time controller. This way of designing discrete-time controllers is popular because many engineers are not really acquainted with the phenomena occurring in digital control systems.
I felt that the only way of providing a systematic approach to designing DC motor controllers with a low sampling rate, was to study the continuous-time behaviour of the combined discrete-time/continuous-time control system. The continuous-time performance of such a hybrid system can be objectively evaluated, independent of the sampling rate. The study of continuous-time performance of digital control systems is the challenging area of sampled-data control.
At Océ-Technologies B.V. in Venlo, I have been able to combine theoretical work with some practical experimentation. I experienced my period in the industrial environment at Océ as both instructive and pleasant. I would like to thank my 'colleagues' for the pleasant interaction. Special thanks goes out to Ad Damen, my supervisor at Eindhoven University of Technology, for the fruitful discussions, and to Peter van den Bosch, who was my supervisor during the 9 months at Océ.

Robert Cloudt.

Contents

1. Project Description

1.1 Assignment

Due to the low costs of digital hardware and because of flexibility reasons, the majority of (motor) controllers are nowadays implemented on a digital microprocessor. However, the design of motor controllers is usually PID based and takes place in the continuous-time domain. The continuous-time PID controller is converted to discrete-time by means of a transformation in order to be implemented on a digital microprocessor. This transformation requires high sampling rates in order to cope with undesired sampling effects. For the implementation of motor control software it is advantageous when the sampling rate is lowered (more relaxed timing). Lower sampling rates could be used when it's acknowledged that the controller is to be implemented on a discrete-time platform.

The goals of the project are:
1. Show that the sampling rate of a motor controller can be lowered by using a properly designed digital controller which obtains (at least) the same performance as the present PID controller.
2. Derive design rules or a general method to select an appropriate sample rate and digital controller.
3. In order to convince everybody of the advantages of digital controller design: apply digital controller design to a representative simulation model and an experimental set-up.

1.2 Strategy

It is a challenge to find out how much the sampling rate can be lowered. This challenge asks for a certain approach to the problem; it requires optimal control strategies. In order to find the lowest possible sampling rate at which some criteria are still met, one has to be able to judge whether a controller is optimal (in a certain sense) for a fixed sampling rate. Furthermore, the criterion in which one is actually interested, is the continuous-time performance of the controlled system. Therefore the problem has a hybrid nature, it involves both continuous-time and discrete-time dynamics.
The main goal of lowering the sampling rate of a controller is an efficiency improvement in the controller software. If the sampling rate is lowered, a running program is less frequently interrupted and less task switching (with a significant processing overhead) occurs. This improvement allows the processor to run more tasks, or a cheaper, slower processor can be used. This cost reduction is canceled if extra hardware is necessary. Therefore the sampling rate has to be lowered without necessary changes in the hardware.
For a fair comparison, the complexity of the control algorithm has to be comparable to the present PID-like motor controllers. It would be convincing to show that a well designed low order controller (possibly a PID-like controller) achieves good performance while the sampling rate is reduced.
The problem of finding a well performing controller with minimal sampling rate is an optimisation problem. Some insight is desired into the limitations and compromises in digital control of a continuous-time process, in order not 'to ask the impossible' of the optimisation routine.
It is believed that the effect of a relatively long sampling period on the controller performance is more drastic than the effect of implementation issues like quantisation, finite accuracy of computations, and sampling time jitter in a real-time software environment. Therefore, in this study the focus is on the effect of the sampling rate. In practice, careful implementation of the control algorithm can reduce the effect of the implementation issues.

1.3 About this report

This report starts with an introduction to digital control and digital controller design methods, in the next chapter. An extensive frequency domain analysis of mixed continuous-time/discrete-time control systems is presented in chapter 3. This yields some interesting insights, but the analysis is unsuited for numerical analysis of a sampled-data control system. Some computable frequency domain characteristics are presented in chapter 4. Time-domain criteria are introduced in chapter 5, in order to study the tracking performance with respect to tracking prespecified setpoint profiles. These time domain criteria and frequency domain characteristics are combined in an controller optimisation algorithm presented in chapter 7, but first, a treatise on modelling a DC motor drive is given in chapter 6. In chapter 8, the controller optimisation scheme is applied to a representative motor control case. Both simulations and practical experimentation are described. Conclusions and recommendations are provided in chapters 9 and 10. Two appendices are appended to this report. Appendix A is about several computations and derivations, whereas MATLAB code implementing these computations is given in appendix B.

2. Introduction to Digital Control

2.1 Introduction

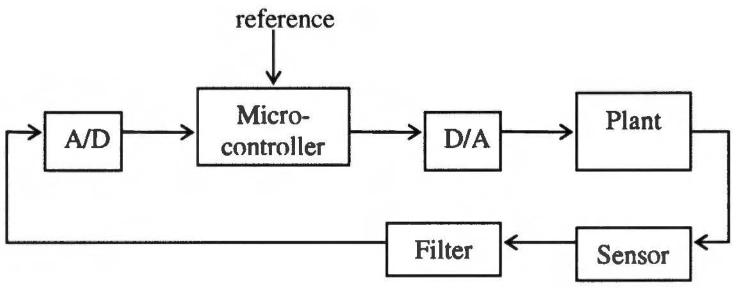


Figure 2.1 Digital control set-up

Figure 2.1 shows a typical setup for a digital control loop. A reference sequence is applied to some digital device, typically a microcontroller. This microcontroller produces a digital control signal which is converted to an analog signal by the digital to analog converter (D/A). The analog signal (usually a piecewise constant voltage) is applied to the plant which represent the physical process to be controlled. The sensor measures the quantities of interest and the sensor signal is filtered to remove the undesired high frequency components. Finally, the filtered signal is digitised by the analog to digital (A/D) converter, such that the microcontroller can use the digital signal to compute the control signal.
The block diagram in figure 2.1 is converted to figure 2.2.

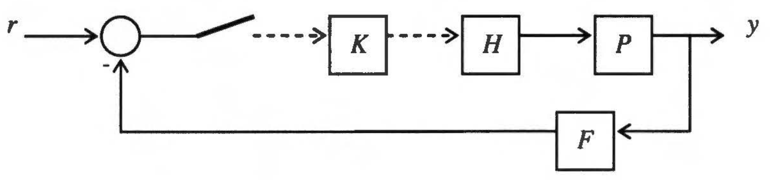


Figure 2.2 Digital control loop

The sampling action of the A/D converter is replaced with the switch in figure 2.2. The switch is closed at sampling instants *kh*, where *k* is an integer and *h* is the sampling period. The quantisation of the A/D converter is neglected. The dashed lines in the block diagram represent discrete-time signals, the solid lines represent continuous-time signals. Note that the reference signal is a continuous-time signal in figure 2.2. Although the reference is typically applied to the micro controller as a digital sequence, one actually wants the control loop to track a smooth, continuous-time reference signal resembling the digitised reference sequence. Therefore, the reference in the block diagram above is in continuous-time. *K* denotes the discrete-time controller. *H* is the hold device which transforms the discrete-time control signal to continuous-time, just as the D/A converter does. The zero order hold (ZOH) is usually used as the hold device. It produces a piecewise constant signal of which the signal level during a sampling interval corresponds to the discrete-time sample (the ZOH applies zeroth order extrapolation, which explains the name). *P* represents the continuous-time plant. The filter is depicted through the system *F*.
In the remaining part of this chapter, three classes of digital controller design methods are treated:
- continuous-time design;
- discrete-time design;
- sampled-data design.

The continuous-time design and discrete-time design use approximations or discretisations in the design of a discrete-time controller. The sampled-data design method is a direct method which does not involve approximation. In figure 2.3 is portrayed how the three design methods relate to each other.

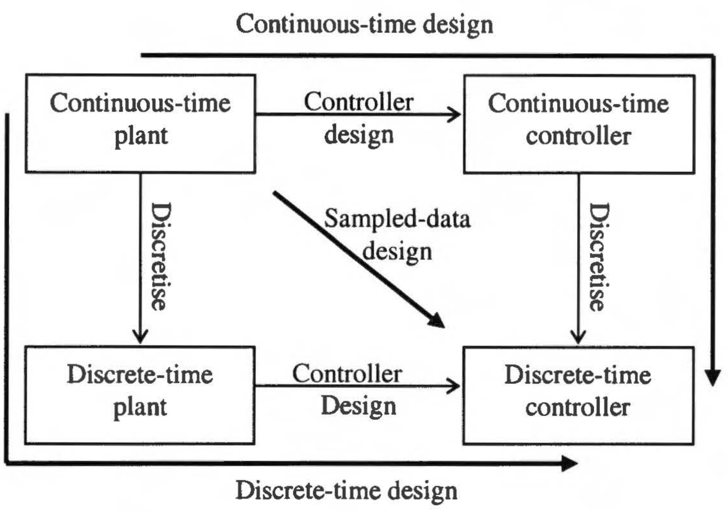


Figure 2.3 Digital controller design methods

The continuous-time design method is the easiest, because well-known continuous-time techniques can be used. The discrete-time design method has the potential to reduce the necessary sampling rate, but it can cause intersample oscillations. The sampled-data design takes the continuous-time performance of the hybrid system into account, but it is more complex. While treating the three classes of controller design methods, some typical side effects of the digital implementation of the controller will be elucidated.

2.2 Continuous-time design

The easiest way of designing a digital controller is applying continuous-time design. In this design procedure, a continuous-time controller is designed for the continuous-time process, and then it is converted to a discrete-time controller by means of some transformation. This transformation replaces the differential equation describing the action of the continuous-time controller with a difference equation. Some examples of such transforms are:

- **Backward Euler transform**
  This transform replaces a differentiation action with the difference equation:

$$\frac{de(t)}{dt} \to \frac{e(kh) - e(kh-h)}{h} \tag{2.1}$$

  or equivalently, the Laplace variable *s* is replaced by a function of *z*:

$$s \to \frac{z-1}{hz} = \frac{1-z^{-1}}{h} \tag{2.2}$$

  (For the moment, *z* might be interpreted as the sampling time advance operator).

- **Tustin transform**
  The Tustin transform (or bilinear transform) is related to the trapezoid integration rule as integration

$$u(t) = \int e(t)dt \tag{2.3}$$

  is replaced by:

$$u(kh+h)=u(kh)+e(kh)+\tfrac{h}{2}\left(e(kh+h)-e(kh)\right)=u(kh)+\tfrac{h}{2}\left(e(kh+h)+e(kh)\right) \tag{2.4}$$

or equivalently:

$$s \rightarrow \frac{2(z-1)}{h(z+1)}. \tag{2.5}$$

- **ZOH transform**
  The zero order impulse response equivalent transform (or step invariant transform) transforms a continuous-time system to a discrete-time system by replacing the continuous-time system with the same continuous-time system prepended with a sampler and appended with a zero order hold. See the figure below.

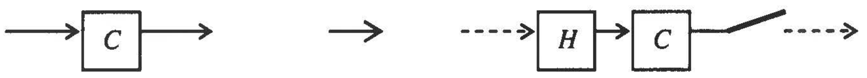


Figure 2.4 ZOH transform

The Tustin transform is quite popular because it yields a good approximation and it maps the stable part of the s-plane onto the stable part of the z-plane. More on these kind of transforms can be found in [1].
The quality of an approximation of a differential equation by a difference equation depends on the step size or sampling time h. If h increases the approximation gets worse. Therefore, the sampling rate $1/h$ has to be high compared to the bandwidth of the system. In textbooks on digital control one might find the rule of the thumb:

Sampling rate $\frac{1}{h}$ > *20* x *closed loop bandwidth (in Hz).*

This rule might be relaxed if the *phase loss* of the zero order hold is accounted for in the continuous-time plant model. The Laplace domain transfer function of the zero order hold is given by:

$$H(s)=\frac{1-e^{-sh}}{s}. \tag{2.6}$$

By replacing s by $j\omega$ and computing the complex argument one can find that

$$\arg H(j\omega)=\frac{\omega h}{2} \tag{2.7}$$

i.e. the zero order hold introduces a delay of half a sample time. The phase loss is identified as one of the major performance limitations in digital control. If this delay is included in the plant model, and the controller is designed for the delayed plant, the performance of the controller will generally be better than when the controller was designed for the undelayed plant [1]. This might give the opportunity to relax the sample rate constraint in the rule of the thumb.

2.3 Discrete-time design

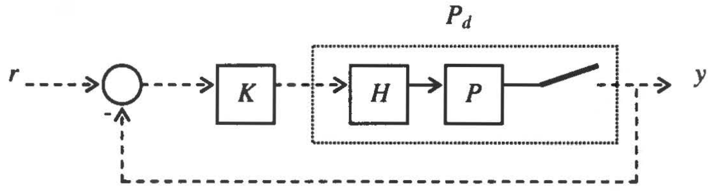


Figure 2.5 Block diagram for discrete-time design

Figure 2.5 shows the block diagram used in discrete-time controller design. It differs from the block diagram in figure 2.2. The reference and output signals are in discrete-time, the filter appears to be removed from the block diagram and the sampling takes place in a different position. So, designing a controller using the block diagram in figure 2.5 is a different problem than finding a controller using the block diagram in figure 2.2.
There are however good reasons for applying the block diagram above. The control loop in figure 2.5. is completely in discrete-time. P_d is the ZOH transform of plant P. When the control loop is considered on sampling instants only, the system is a discrete-time linear time-invariant (LTI) system. This property allows the use of transfer functions in the z-domain. Background information on the z-transform can be found in [2]. Furthermore, the famous Shannon-Nyquist sampling theorem states that the original continuous-time output signal of the combination PF can be reconstructed from the sampled signal if PF is band-limited to half the sampling frequency. In practice, however, this is not the case. Moreover, the reconstruction of the continuous-time signal requires non-causal filtering which is physically unrealisable. A more cautious choice on the sampling rate has to be used instead. It is wise to choose the sampling rate of at least 5 times the required closed loop bandwidth [1], but the performance limitations due to the phase loss of the zero order hold might further increase the necessary sampling rate.
A drawback of the discrete-time design method is the loss of intersample information. The discrete-time output y offers only information of the plant output at sampling times. Whether or not intersample oscillation occurs, depends on the sampling rate and the discrete-time controller K. There are however guidelines on how to choose the controller and the corresponding closed-loop pole positions in the z-plane to reduce the chance of disturbing intersample ripples (see [2]).

A word on the filter F is in place. It suits a more elaborate role than filtering just any high frequency components. By eliminating all frequency components above half the sampling frequency, it makes sure there is a one-to-one correspondence between the continuous-time signal and the sampled signal. This needs further explanation. Suppose a sampler is sampling the continuous-time signal $e(t)$ which is a sum of (complex) sinusoids of radial frequency $\omega + n\omega_s$ ($\omega_s = \frac{2\pi}{h}$, the sampling frequency in rad/s):

$$e(t) = \sum_{n=-\infty}^{\infty} e_n e^{j(\omega + n\omega_s)t}, \quad e_n \in \mathbb{C}. \tag{2.8}$$

After sampling, the signal is represented by

$$e[k] = \sum_{n=-\infty}^{\infty} e_n e^{j(\omega+n\omega_s)kh} = \sum_{n=-\infty}^{\infty} e_n e^{j(\omega kh + 2\pi kn)} = e^{j\omega kh} \sum_{n=-\infty}^{\infty} e_n . \tag{2.9}$$

($e[k]$ is shorthand notation for $e(kh)$).
This last equation exposes one of the effects of sampling. After sampling, it is impossible to distinguish between the fundamental component and the higher frequency components. All

continuous-time signals of frequency $\omega + n\omega_s$ with $-\frac{\omega_s}{2} < \omega < \frac{\omega_s}{2}$, are mapped on a single discrete-time sinusoid with frequency $-\frac{\omega_s}{2} < \omega < \frac{\omega_s}{2}$.

$e[k]$, $k \in \mathbb{N}$ is just a sequence of numbers. In signal analysis, signals have a notion of energy. For signals with energy it makes sense to work with them in the frequency domain. For this reason, the sampling action is modelled in the frequency domain by impulse modulation; the continuous-time signal is multiplied with a 'pulse train' like in figure 2.6. The resulting signal is a series of Dirac pulses with energy corresponding to the samples $e[k]$. Applying such a signal to the zero order hold yields the piecewise constant output signal of the D/A converter, as the impulse response of the ZOH is a unit pulse of duration h.

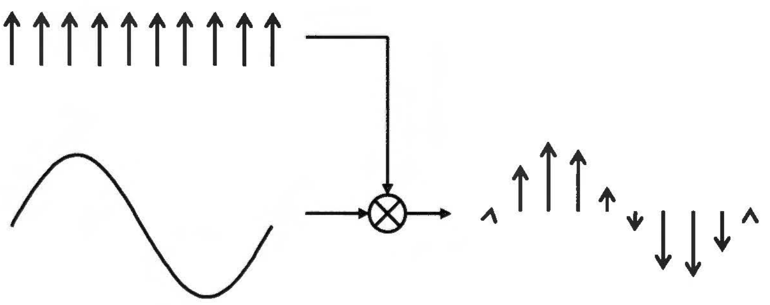


Figure 2.6 Impulse modualtion

Let $\tilde{e}(t)$ denote the resulting pulse train signal. It can be expressed as:

$$\tilde{e}(t) = e(t) \cdot \sum_{k=0}^{\infty} \delta(t - kh) = \sum_{k=0}^{\infty} e[k]\delta(t - kh). \tag{2.10}$$

The pulse train $\Sigma\delta(t - kh)$ is a periodic signal of period h so it can be represented by its Fourier series expansion:

$$\sum_{k=0}^{\infty} \delta(t - kh) = \frac{1}{h} \sum_{n=-\infty}^{\infty} e^{jn\omega_s t}. \tag{2.11}$$

Now, suppose $e(t)$ is a single (complex) sinusoid with frequency $-\frac{\omega_s}{2} \le \omega < \frac{\omega_s}{2}$:

$$e(t) = e_0 e^{j\omega t} \tag{2.12}$$

then substituting (2.11) and (2.12) into (2.10) yields:

$$\tilde{e}(t) = e_0 e^{j\omega t} \cdot \frac{1}{h} \sum_{n=-\infty}^{\infty} e^{jn\omega_s t} = e_0 \frac{1}{h} \sum_{n=-\infty}^{\infty} e^{j(\omega + n\omega_s)t}. \tag{2.13}$$

This equation shows that sampling (modelled in the frequency domain by impulse modulation) introduces components with frequencies $\omega + n\omega_s$, $n \in \mathbb{Z}$. These components are called *aliasing components* or *aliases* of the signal $e(t) = e_0 e^{j\omega t}$. As the sampling operation is linear (!), the same holds for a sum of sinusoids: for each frequency component, an infinite amount of aliasing components is introduced after digital signal processing.
Equation 2.9 showed that after sampling, the original signal and its aliases are indistinguishable. The filter F has the task to suppress all the aliasing components, such that the sampled signal corresponds

to the components in the frequency band $-\frac{\omega_s}{2} \leq \omega < \frac{\omega_s}{2}$ and not to the aliasing components. The filter *F* is usually called an *anti-aliasing filter*. Ideally, it is a brick wall low pass filter with a pass band of $\frac{\omega_s}{2}$. If this is not the case, some components of frequency $\omega > \frac{\omega_s}{2}$ are mapped onto a discrete-time sinusoid of frequency $\omega - n\omega_s$ with n such that $-\frac{\omega_s}{2} < \omega - n\omega_s < \frac{\omega_s}{2}$. This phenomena is called *frequency folding*.

2.4 Sampled-data design

The class of sampled-data design methods includes all the digital controller design methods which do not apply any approximations or discretisations like is done in the continuous-time design and discrete-time design classes. Sampled-data design methods are concerned with continuous performance criteria and take the hybrid nature of the mixed continuous-time/discrete-time system into account.

The name "sampled-data" is taken from the publications in the 90's concerning these kind of design methods. The nomenclature is not that clear. The sampled-data techniques considered here are sometimes called direct SD or direct digital design methods, whereas in some works (mainly before the 90's) the term "sampled-data" is used for discrete-time design methods. In this report, the term "sampled-data" is used whenever the hybrid nature of the digital control system is taken into account.

The difficulty in sampled-data design methods lies in the hybrid nature of the systems. When the behaviour of the digital control loop in figure 2.2 is considered in continuous-time, the closed loop system is linear, periodically time-varying (LPTV) with period *h*. Suppose the continuous-time reference signal $r(t)$ is applied to a digital controller. The digital controller produces a piecewise constant signal $u_0(kh)$ (the ZOH is assumed as the hold device) which is derived from $r(kh)$, the sampled version of the reference signal. Now consider the delayed reference signal $r(t - \eta)$. The sampled version $r(kh - \eta)$ is generally different from $r(kh)$ and the generated control sequence $u_\eta(kh)$ will differ too, i.e. $u_\eta(kh) \neq u_0(kh - \eta)$. However, if $\eta = \ell h$, the correspondence $u_\eta(kh) = u_0(kh - \eta)$ holds. This means that the system is linear, periodically time-varying (LPTV), and the classical concept of a transfer function cannot be applied.

The majority of the literature on sampled-data control is on optimal (in a H_2 or H_∞ sense) sampled-data control. Several general approaches to optimal sampled-data control have been developed:

- a 'lifting' approach which transforms the hybrid problem into an equivalent discrete-time problem [3];
- an approach based on polynomial methods [4];
- a game-theoretic approach resulting in hybrid Riccati equations [5];
- an approach based on frequency domain operators [6].

In chapters 3 and 4 some frequency domain analysis is applied to sampled-data systems. In chapter 5, time-domain measures are introduced which are related to sampled-data H_2 optimal control problems. Appendix A.4 treats the lifiting approach to optimal sampled-data control from a general point of view. The derivations in appendices A.5 and A.6 are inspired on this lifting approach.

3. Analysis of Sampled-data Feedback Control Systems

3.1 Introduction

Modelling the sampling action in a digital control system by impulse modulation allows frequency domain analysis of sampled-data systems. In this chapter, the influence of the discrete-time controller on the control constraints:
- stability;
- robust stability;
- avoidance of actuator saturation;

and the control aims:
- disturbance rejection;
- tracking;
- sensor noise avoidance;
- and robust performance;

is studied. Furthermore, some fundamental limitations of sampled-data feedback are presented. The analysis is mainly being carried out in the Fourier domain. The frequency domain analysis technique is based on ideas presented in [7].

3.2 Preliminaries

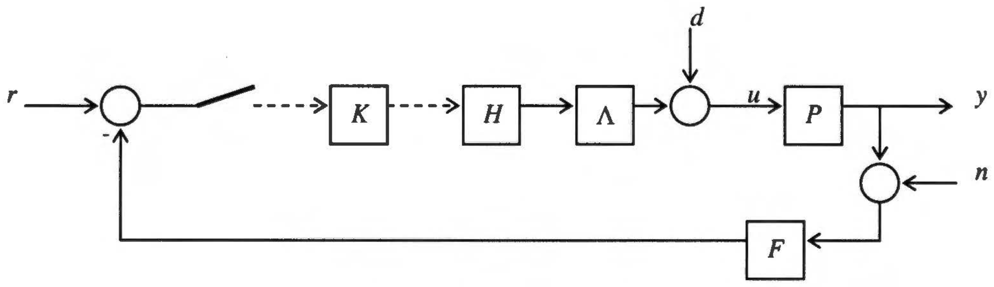


Figure 3.1 Block diagram under study

The properties of the general sampled-data system in figure 3.1 are being studied. *P* represents the continuous-time plant to be controlled. *K* is the discrete-time controller with sampling time *h*. *H* is the zero order hold and *F* the anti-aliasing filter. Λ represents the computation delay $0 \leq \lambda < h$ which is the time between the sampling instant and the output of a new control sample in the digital control algorithm. The signals *r*, *u*, *y*, *d* and *n* represent the reference signal, the control signal, the output signal, a disturbance and the measurement noise respectively.
Before being able to derive the various transfers in the block diagram above, one has to focus on the use of the *z*-transform and study the effect of sampling in the Fourier domain. Suppose a signal *v(t)* is sampled resulting in a series of Dirac pulses $\tilde{v}(t)$ due to impulse modulation. The question is, how $V(j\omega)$, the Fourier transform of *v(t)* relates to $\tilde{V}(j\omega)$, the Fourier transform of $\tilde{v}(t)$. Straightforward application of the definition of the Fourier transform yields:

$$\tilde{V}(j\omega)= \int_{-\infty}^{\infty}\tilde{v}(t)e^{-j\omega t}dt = \int_{-\infty}^{\infty}\left(v(t)\sum_{k=0}^{\infty}\delta(t-kh)\right)e^{-j\omega t}dt. \tag{3.1}$$

Expressing the pulse train $\Sigma\delta(t-kh)$ in terms of its Fourier series expansion yields:

$$\tilde{V}(j\omega)=\int_{-\infty}^{\infty} v(t)\frac{1}{h}\sum_{n=-\infty}^{\infty} e^{jn\omega_s t}e^{-j\omega t}dt=\frac{1}{h}\sum_{n=-\infty}^{\infty}\int_{-\infty}^{\infty} v(t)e^{-j(\omega-n\omega_s)t}dt. \qquad (3.2)$$

Recognising the shift in frequency and applying the definition of the Fourier transform learns:

$$\tilde{V}(j\omega)=\frac{1}{h}\sum_{n=-\infty}^{\infty} V(j\omega-jn\omega_s). \qquad (3.3)$$

This important sampling formula shows how the spectrum of the sampled signal is composed of infinitely many shifted copies of the spectrum of the continuous-time signal.
Now, let's study the use of the *z*-transform. The definition of the *z*-transform of a discrete-time signal *v*[*k*] is:

$$V(z)=\sum_{k=0}^{\infty} v[k]z^{-k}. \qquad (3.4)$$

Consider equation 3.1 again. Exploiting the sieve property of the Dirac pulses results in:

$$\tilde{V}(j\omega)=\sum_{k=0}^{\infty} v(kh)e^{-j\omega kh}=\sum_{k=0}^{\infty} v[k]e^{-j\omega hk}=V(z)\Big|_{z=e^{j\omega h}}. \qquad (3.5)$$

So, the *z*-transform of a discrete-time signal is related to the Fourier transform of the impulse modulated signal through the correspondence $z=e^{j\omega h}$. This correspondence shows that *z* might indeed be interpreted as the sample time advance operator, as $e^{j\omega h}$ represents a negative delay of duration *h* in the Fourier domain. The correspondence allows an easy conversion from a difference equation describing a discrete-time system, to a transfer function in the *z*-domain. Suppose the discrete-time system *K* maps the discrete-time signal *e*[*k*] to the signal *x*[*k*]. In the *z*-domain, this operation can be expressed as *X*(*z*) = *K*(*z*)*E*(*z*), which corresponds to the convolution sum in discrete-time domain

$$x[k]=\sum_{\ell=-\infty}^{\infty} k[\ell]e[k-\ell] \qquad (3.6)$$

where $k[\ell]$ denotes the impulse response of discrete-time system *K*.

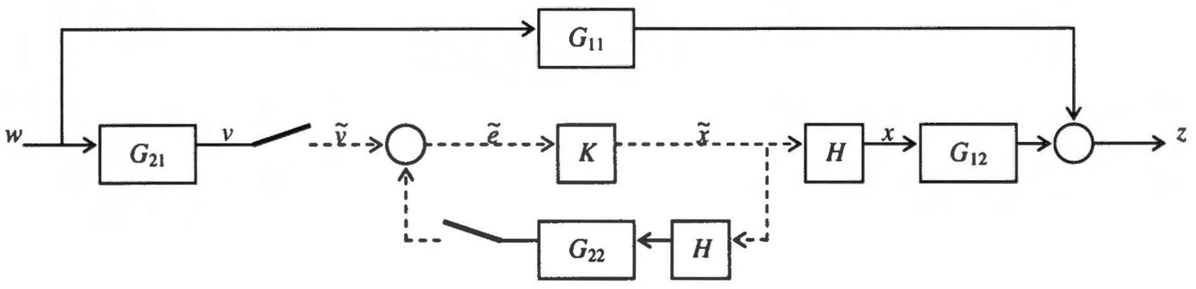


Figure 3.2 Universal block diagram for a sampled-data system

The block diagram in figure 3.2 allows the calculation of every transfer of interest in figure 3.1 by a proper choice of the continuous-time systems G_{11} to G_{22}. First, the Fourier transform of the output z has to be expressed as a function of the Fourier transform of input w.
The Fourier transform of z is:

$$Z(j\omega)=G_{11}(j\omega)W(j\omega)+G_{12}(j\omega)H(j\omega)\tilde{X}(j\omega) \tag{3.7}$$

where $\tilde{X}(j\omega)$ is equal to the z-transform of x with $e^{j\omega h}$ substituted for z. The discrete-time part of the figure 3.2 is recognised as a feedback loop with K in the feedforward path and the ZOH transform of G_{22} as (positive) feedback. Straightforward block diagram algebra relates $X(z)$ to the z-transform of v:

$$X(z)=K(z)(I-G_{22d}(z)K(z))^{-1}V(z)=R_d(z)V(z). \tag{3.8}$$

G_{22d} denotes the ZOH transform of G_{22}. $V(z)\big|_{z=e^{j\omega h}}$ equals the Fourier transform of the pulse train signal $\tilde{v}$ which in turn can be expressed as (by using (3.3)):

$$\tilde{V}(j\omega)=\frac{1}{h}\sum_{n=-\infty}^{\infty}G_{21}(j\omega-jn\omega_s)W(j\omega-jn\omega_s). \tag{3.9}$$

Combing these facts eventually yields:

$$Z(j\omega)=G_{11}(j\omega)W(j\omega)+G_{12}(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)\cdot\frac{1}{h}\sum_{n=-\infty}^{\infty}G_{21}(j\omega-jn\omega_s)W(j\omega-jn\omega_s). \tag{3.10}$$

The transfer function in (3.8) is deliberately named $R_d(z)$ because it is related to the control sensitivity function of the discretised system. Table 3.1 shows how G_{11} to G_{22} should be chosen to calculate the transfers of interest in figure 3.1.
The asymptotic response to a complex (possibly multivariable) sinusoid $w(t)=w_0e^{j\omega_0 t}$ can be obtained through inverse Fourier transform of (3.10) where $W(j\omega)=2\pi\delta(\omega-\omega_0)$:

$$\begin{aligned}z(t)&=\frac{1}{2\pi}\int_{-\infty}^{\infty}Z(j\omega)e^{j\omega t}\,d\omega=\\&G_{11}(j\omega_0)w_0e^{j\omega_0 t}+\left[\frac{1}{h}\sum_{n=-\infty}^{\infty}G_{12}(j\omega_0-jn\omega_s)H(j\omega_0-jn\omega_s)e^{j(\omega_0-n\omega_s)t}\right]\cdot R_d\left(e^{j\omega_0 h}\right)G_{21}(j\omega_0)w_0\end{aligned} \tag{3.11}$$

This equation exposes the aliasing effect: the response to a sinusoid of frequency ω_0 consists of infinitely many sinusoidal components with frequencies $\omega_0-n\omega_s$, $n\in\mathbb{Z}$, i.e. z can be written as:

$$z(t)=\sum_{n=-\infty}^{\infty}z_ne^{j(\omega_0+n\omega_s)t}\ ,\ z_n\in\mathbb{C}. \tag{3.12}$$

In the forthcoming analysis the distinction is made between the transfer of the fundamental component and the aliasing components. The fundamental transfer reflects the desired behaviour (when compared with a continuous-time controller), whereas the aliasing components are undesired side effects of the discrete-time implementation of the controller.

Table 3.1 Overview of transfers

w	z	G_{11}	G_{12}	G_{21}	G_{22}
r	y	0	$P\Lambda$	I	$-FP\Lambda$
r	u	0	Λ	I	$-FP\Lambda$
d	y	P	$P\Lambda$	$-FP$	$-FP\Lambda$
d	u	I	Λ	$-FP$	$-FP\Lambda$
n	y	0	$P\Lambda$	$-F$	$-FP\Lambda$
n	u	0	Λ	$-F$	$-FP\Lambda$

3.3 Stability

The stability of sampled-data system is studied in state space. This state space approach is due to [8], one of the earliest work on sampled-data systems. The concept of *internal stability* for the sampled-data system in figure 3.1 is defined as: *The sampled-data system is internally stable if for every initial controller state and every initial plant state, both states approach zero asymptotically when there are no exogenous inputs applied.* The autonomous response to initial states of the sampled-data system, considered on sampling times only, is completely described by the discrete-time system below.

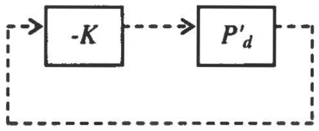


Figure 3.3. Autonomous discrete-time system

P'_d is the ZOH transform of $FP\Lambda$. Its state space representation is:

$$\begin{aligned} x_{P'_d}[k+1] &= A_{P'_d} x_{P'_d}[k] + B_{P'_d} u[k] \\ y[k] &= C_{P'_d} x_{P'_d}[k] + D_{P'_d} u[k] \end{aligned} \quad . \tag{3.13}$$

The state space representation of the discrete-time controller is given by:

$$\begin{aligned} x_K[k+1] &= A_K x_K[k] + B_K y[k] \\ u[k] &= C_K x_K[k] + D_K y[k] \end{aligned} \quad . \tag{3.14}$$

The response of the combined discrete states is governed by the equation:

$$\begin{pmatrix} x_{P'_d} \\ x_K \end{pmatrix}[k+1] = \begin{pmatrix} A_{P'_d} - B_{P'_d} D_K C_{P'_d} & -B_{P'_d} C_K \\ B_K C_{P'_d} & A_K \end{pmatrix} \begin{pmatrix} x_{P'_d} \\ x_K \end{pmatrix}[k]. \tag{3.15}$$

If all the eigenvalues of the matrix in the expression above have modulus less than one, then the combined discrete state will converge asymptotically to zero as time passes, regardless of the initial states. As $x_{P'_d}[k]$ converges to zero, so does $y[k]$. This sample is fed back to the discrete-time controller to generate a control sample $u[k]$ which in turn also converges to zero as $k \to \infty$. The evolution of the combined state of the continuous-time subsystem can be obtained through integration of the continuous-time state space equations (see [8]). The state of FP for $kh \le t < kh + h$ is governed by the equation:

$$
\begin{aligned}
x_{FP}(t) &= e^{A_{FP}(t-kh)}x_{FP}(kh) + \int_{kh}^{t} e^{A_{FP}(t-\tau)}B_{FP}u(\tau)d\tau \\
&= e^{A_{FP}(t-kh)}x_{FP}[k] + \begin{cases} \int_{kh}^{t} e^{A_{FP}(t-\tau)}B_{FP}d\tau \cdot u[k-1], & t-kh \le \lambda \\ \int_{kh}^{kh+\lambda} e^{A_{FP}(t-\tau)}B_{FP}d\tau \cdot u[k-1] + \int_{kh+\lambda}^{t} e^{A_{FP}(t-\tau)}B_{FP}d\tau \cdot u[k], & t-kh > \lambda \end{cases}
\end{aligned} \quad . \tag{3.16}
$$

$x_{FP}[k]$ is a subvector of $x_{P_d'}[k]$ because $x_{P_d'}[k]$ can be written as

$$
x_{P_d'}[k] = \begin{pmatrix} x_{FP}[k] \\ u[k-1] \end{pmatrix}
$$

by adopting the delay in the state space equations. While $x_{P_d'}[k]$ coverges to zero, so does $x_{FP}[k]$ and so do $u[k]$ and $u[k-1]$. Equation 3.16 reveals that the continuous-time state also converges to zero as time passes. This means that the sampled-data system is internally stable if the matrix

$$
\begin{pmatrix} A_{P_d'} - B_{P_d'}D_K C_{P_d'} & -B_{P_d'}C_K \\ B_K C_{P_d'} & A_K \end{pmatrix} \tag{3.17}
$$

has stable eigenvalues. This conclusion partly justifies the study of purely discrete-time systems where the plant is discretised by the ZOH transform, because the internal stability of the sampled-data system is equivalent to the stability of the discretised system.

3.4 Continuous-time tracking

Tracking in purely discrete-time feedback systems is studied by applying the final value theorem in the *z*-domain. The asymptotic tracking error for the control loop in figure 2.5 can be calculated to be:

$$
r[\infty] - y[\infty] = \lim_{z \to 1} \frac{z-1}{z} \cdot (I + P_d(z)K(z))^{-1} R(z) \tag{3.18}
$$

where *R*(*z*) is the *z*-transform of the reference signal. In this discrete-time framework one can conclude that (for single input, single output (SISO) systems):

- the discrete-time system is step tracking if there is at least one discrete-time integrator mode in $P_d(z)K(z)$;
- the discrete-time system is ramp tracking if there are at least two discrete-time integrator modes in $P_d(z)K(z)$.

This study however, considers only the response on sampling instants, while one is actually interested in the continuous-time tracking behaviour. One is forgiven to think that the result of the previous section implies that the steady-state error asymptotically reaches zero. This is indeed true for the response to an unknown initial state, but in the case of exogenous input signals, this is generally not the case.

The continuous-time asymptotic tracking error is calculated through applying the final value theorem in the Fourier-domain. Using (3.11), the response of $r - y$ to $r(t) = r_0 e^{j\omega t}$ is characterised by:

$$r(t) - y(t) = r_0 e^{j\omega t} - \frac{1}{h}\sum_{n=-\infty}^{\infty} P(j\omega - jn\omega_s)\Lambda(j\omega - jn\omega_s)H(j\omega - jn\omega_s)R_d\left(e^{j\omega h}\right)r_0 e^{j(\omega - n\omega_s)t}. \quad (3.19)$$

Separating the fundamental and aliasing components yields:

$$\begin{aligned} r(t) - y(t) &= \left(I - \frac{1}{h}P(j\omega)\Lambda(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)\right)r_0 e^{j\omega t} \\ &\quad - \frac{1}{h}\sum_{n\neq 0} P(j\omega - jn\omega_s)\Lambda(j\omega - jn\omega_s)H(j\omega - jn\omega_s)R_d\left(e^{j\omega h}\right)r_0 e^{j(\omega - n\omega_s)t} \end{aligned} \quad (3.20)$$

Let $z = r - y$. z has the structure of (3.12). The first term in (3.20) relates the fundamental components r_0 and z_0, and is a non-rational transfer function in the classical sense. The second term in (3.20) relates the aliasing components z_n to r_0. It is not a transfer function in the classical sense because the signals involved have unequal frequency, but for each n

$$\frac{1}{h}P(j\omega - jn\omega_s)\Lambda(j\omega - jn\omega_s)H(j\omega - jn\omega_s)R_d\left(e^{j\omega h}\right) \quad (3.21)$$

contains the gain and phase information relating z_n and r_0. So it makes sense to apply the final value theorem in the Fourier domain to (3.21). Let $R(j\omega)$ be the Fourier transform of the reference signal. The conditions for a zero asymptotic tracking error are:

$$\lim_{j\omega\to 0} j\omega\left(I - \frac{1}{h}P(j\omega)\Lambda(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)\right)R(j\omega) = 0 \quad (3.22)$$

and

$$\forall n \neq 0: \quad \lim_{j\omega\to 0} j\omega\frac{1}{h}P(j\omega - jn\omega_s)\Lambda(j\omega - jn\omega_s)H(j\omega - jn\omega_s)R_d\left(e^{j\omega h}\right)R(j\omega - jn\omega_s) = 0 \quad (3.23)$$

The zero order hold has zero transfer for frequencies $n\omega_s$, $n \neq 0$ and unity transfer for $\omega = 0$ (see figure 3.4), so condition 3.23 is satisfied when the zero order hold is used as a hold device. Λ has a DC gain of one, as

$$\Lambda(j\omega) = Ie^{-j\omega\lambda}. \quad (3.24)$$

Condition 3.22 reduces to:

$$\lim_{j\omega\to 0} j\omega\left(I - \frac{1}{h}P(j\omega)R_d\left(e^{j\omega h}\right)\right)R(j\omega) = 0. \quad (3.25)$$

For SISO systems without an anti-aliasing filter, the term of interest is:

$$P(j\omega)R_d\left(e^{j\omega h}\right) = \frac{P(j\omega)K\left(e^{j\omega h}\right)}{1 + P_d\left(e^{j\omega h}\right)K\left(e^{j\omega h}\right)} = \frac{P(j\omega)K\left(e^{j\omega h}\right)}{1 + \frac{1}{h}\sum_{n=-\infty}^{\infty} P(j\omega - jn\omega_s)\Lambda(j\omega - jn\omega_s)H(j\omega - jn\omega_s)K\left(e^{j\omega h}\right)}. \quad (3.26)$$

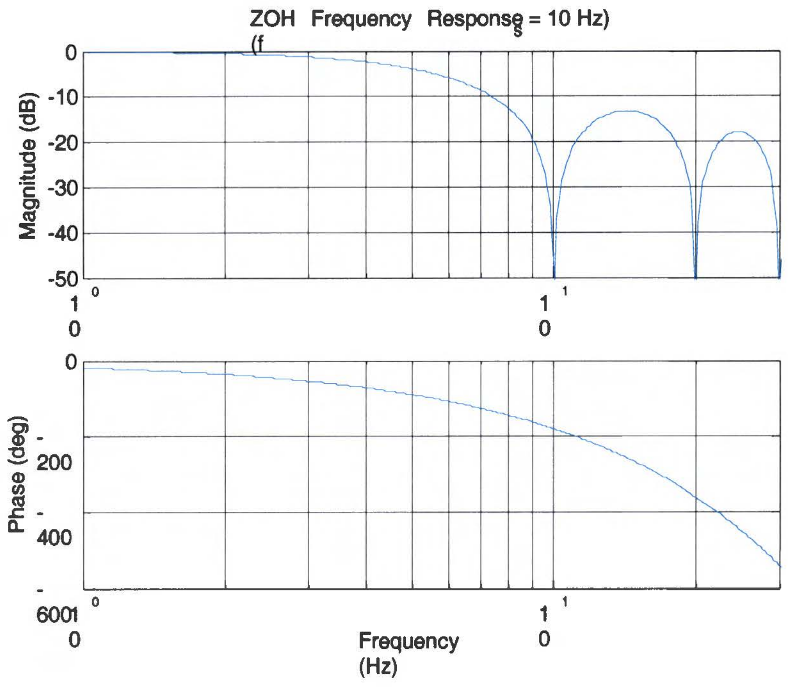


Figure 3.4 ZOH frequency response

It turns out (see also [3]) that the condition for step tracking ($R(j\omega)=\frac{1}{j\omega}$) is:

- **the sampled-data system is step tracking if the sum of integrator modes in the continuous-time plant and the discrete-time controller is at least one;**

and the condition for ramp tracking ($R(j\omega)=\frac{1}{-\omega^2}$):

- **the sampled-data system is ramp tracking if the continuous-time plant contains at least one integrator mode and the discrete-time controller contains at least one integrator mode.**

In general, to track the signal t^m with a zero asymptotic tracking error, the continuous-time plant has to contain m integrator modes. This is because the factors $(j\omega)^{m+1}$ due to the Fourier transform of the reference signal have to be compensated by m factors $j\omega$ descending from the continuous-time plant. The frequency response of the discrete-time controller is a function of $e^{j\omega h}$ and is not able to compensate a factor $j\omega$.

3.5 Hybrid disturbance rejection

The same steady state analysis as in the previous section can be applied to study the disturbance rejection properties of sampled-data systems. The asymptotic response to a sinusoidal disturbance $d(t)=d_0 e^{j\omega t}$ is given by:

$$\begin{aligned} y(t)=&\left(I-\frac{1}{h}P(j\omega)\Lambda(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)F(j\omega)\right)P(j\omega)d_0e^{j\omega t} \\ &-\frac{1}{h}\sum_{n\neq 0}P(j\omega-jn\omega_s)\Lambda(j\omega-jn\omega_s)H(j\omega-jn\omega_s)R_d\left(e^{j\omega_0 h}\right)F(j\omega)P(j\omega)d_0e^{j(\omega-n\omega_s)t} \end{aligned} \quad (3.27)$$

The conditions for asymptotic hybrid disturbance rejection are:

$$\lim_{j\omega\to 0} j\omega\left(I-\frac{1}{h}P(j\omega)\Lambda(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)F(j\omega)\right)P(j\omega)D(j\omega)=0 \tag{3.28}$$

and

$$\forall n\neq 0:\ \lim_{j\omega\to 0} j\omega\frac{1}{h}P(j\omega-jn\omega_s)\Lambda(j\omega-jn\omega_s)H(j\omega-jn\omega_s)R_d\left(e^{j\omega h}\right)F(j\omega)P(j\omega)D(j\omega-jn\omega_s)=0 \tag{3.29}$$

where $D(j\omega)$ is the Fourier transform of *d(t)*. A well known property of continuous-time feedback systems is that the response to a sinusoidal disturbance can be made arbitrary small, by applying a large feedback gain at that disturbance frequency. This is not the case for sampled-data feedback systems! Assume that $\left|K\left(e^{j\omega_0 h}\right)\right|\to\infty$ and $P(j\omega_0-jn\omega_s)\Lambda(j\omega_0-jn\omega_s)H(j\omega_0-jn\omega_s)\neq 0$ for some n and some ω_0, then

$$R_d\left(e^{j\omega_0 h}\right)\to R_{d\infty}\left(e^{j\omega_0 h}\right)=\left(\frac{1}{h}\sum_{n=-\infty}^{\infty}P(j\omega_0-jn\omega_s)\Lambda(j\omega_0-jn\omega_s)H(j\omega_0-jn\omega_s)\right)^{-1}. \tag{3.30}$$

Sampling formula 3.3 was used here to express the ZOH transform of $P\Lambda$ as an infinite summation in the frequency domain. The complex amplitude of the fundamental component in output y is:

$$y_0=\left(I-\frac{1}{h}P(j\omega_0)\Lambda(j\omega_0)H(j\omega_0)R_{d\infty}\left(e^{j\omega_0 h}\right)F(j\omega_0)\right)P(j\omega_0)d_0 \tag{3.31}$$

which in general is non-zero, so **in general the fundamental component of the disturbance response of a sampled-data feedback system cannot be reduced to arbitrary magnitude by applying a large feedback gain. However, if the disturbances are constant, the hold device is a zero order hold and the discrete-time controller contains an integrator, then the disturbance is completely rejected in steady state** (if the DC gain of the anti-aliasing filter is one).

3.6 Sensor noise avoidance

Using the result of section 3.2, the asymptotic response to a sinusoidal measurement error can be expressed as:

$$\begin{aligned} y(t)=&-\frac{1}{h}P(j\omega)\Lambda(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)F(j\omega)n_0e^{j\omega t}\\ &-\frac{1}{h}\sum_{n\neq 0}P(j\omega-jn\omega_s)\Lambda(j\omega-jn\omega_s)H(j\omega-jn\omega_s)R_d\left(e^{j\omega h}\right)F(j\omega)n_0e^{j(\omega-n\omega_s)t}\end{aligned}. \tag{3.32}$$

The measurement error induces some components in the output with frequencies $\omega+n\omega_s$, $n\in\mathbb{Z}$. The effect of measurement errors on the output increases if more feedback is applied. If $\left|K\left(e^{j\omega_0 h}\right)\right|\to\infty$ for some ω_0, then $R_d\left(e^{j\omega_0 h}\right)\to R_{d\infty}\left(e^{j\omega_0 h}\right)$ and $y(t)=-\sum_n F(j\omega)n_0e^{j(\omega_0-n\omega_s)t}$ (full measurement noise).

3.7 Actuator saturation avoidance

In order to avoid actuator saturation, the controller gain should not be too high, especially at frequencies above the bandwidth of the actuator. The actuator signal u is influenced by the exogenous inputs r, d and n. The asymptotic response to sinusoidal exogenous inputs is given by:

$$\begin{aligned}
u(t)=&\frac{1}{h}\Lambda(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)r_0e^{j\omega t}+\frac{1}{h}\sum_{n\neq 0}\Lambda(j\omega-jn\omega_s)H(j\omega-jn\omega_s)R_d\left(e^{j\omega h}\right)r_0e^{j(\omega-n\omega_s)t}\\
&-\frac{1}{h}\Lambda(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)F(j\omega)n_0e^{j\omega t}-\frac{1}{h}\sum_{n\neq 0}\Lambda(j\omega-jn\omega_s)H(j\omega-jn\omega_s)R_d\left(e^{j\omega h}\right)F(j\omega)n_0e^{j(\omega-n\omega_s)t}\\
&+\left(I-\frac{1}{h}\Lambda(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)F(j\omega)P(j\omega)\right)d_0e^{j\omega t}\\
&-\frac{1}{h}\sum_{n\neq 0}\Lambda(j\omega-jn\omega_s)H(j\omega-jn\omega_s)R_d\left(e^{j\omega h}\right)F(j\omega)P(j\omega)d_0e^{j(\omega-n\omega_s)t}\\
&. \qquad (3.33)
\end{aligned}$$

3.8 Robust stability

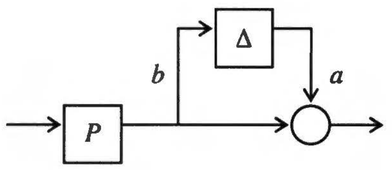


Figure 3.5 Multiplicative plant error

Suppose the plant in figure 3.1 is replaced by the perturbed plant in the figure above. Δ is an arbitrary operator with bounded L_2 induced norm, where L_2 represents the Lebesque space of square integrable functions. It represents a multiplicative model error. The block diagram in figure 3.1 with figure 3.5 replacing the nominal plant model can be manipulated into a form where all exogenous inputs are combined in a single vectorised input m as in the figure below.

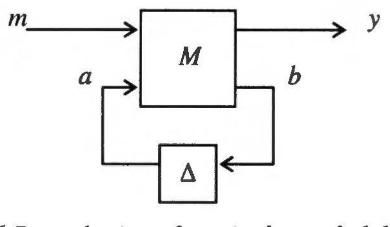


Figure 3.6 Perturbation of nominal sampled data system

The sampled-data system M can be partitioned as:

$$\begin{pmatrix} y \\ b \end{pmatrix}=\begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}\begin{pmatrix} m \\ a \end{pmatrix}. \qquad (3.34)$$

Suppose M_{11}, M_{12}, M_{21} and M_{22} are bounded linear operators on L_2 (in fact, this is implied by the internal stability of M, [3]). The mapping from $m\in L_2$ to $y\in L_2$ is bounded if the loopgain in the loop consisting of M_{22} and Δ is less then one (small gain theorem):

$$\|M_{22}\Delta\|<1. \qquad (3.35)$$

The norm is the induced L_2 norm (although mathematically not entirely correct, this norm might be interpreted as the maximal power gain). Applying the Schwarz inequality gives a sufficient condition for robust stability:

$$\|M_{22}\| < \frac{1}{\|\Delta\|}. \tag{3.36}$$

M_{22} is characterised by the closed loop transfer from a to b. This transfer can be calculated using the framework of section 3.2 with $G_{11} = 0$, $G_{12} = P\Lambda$, $G_{21} = -F$ and $G_{22} = -FP\Lambda$. The Fourier transforms of a and b are related through (see (3.10)):

$$B(j\omega) = P(j\omega)\Lambda(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)\cdot\frac{1}{h}\sum_{n=-\infty}^{\infty} -F(j\omega - jn\omega_s)A(j\omega - jn\omega_s). \tag{3.37}$$

Suppose $a(t)$ has the structure:

$$a(t) = \sum_{n=-\infty}^{\infty} a_n e^{j(\omega_0 - n\omega_s)t}\ ,\ a_n \in \mathbb{C} \tag{3.38}$$

then the Fourier transform of $a(t)$ is:

$$A(j\omega) = 2\pi\sum_{n=-\infty}^{\infty} a_n\delta(\omega - \omega_0 + n\omega_s). \tag{3.39}$$

Substituting (3.39) into (3.37) yields:

$$\begin{aligned} B(j\omega) &= P(j\omega)\Lambda(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)\cdot\frac{2\pi}{h}\sum_{n=-\infty}^{\infty} -F(j\omega - jn\omega_s)\sum_{m=-\infty}^{\infty} a_n\delta(\omega - n\omega_s - \omega_0 + n\omega_s) \\ &= P(j\omega)\Lambda(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)\cdot\frac{2\pi}{h}\sum_{n=-\infty}^{\infty} -F(j\omega - jn\omega_s)a_n\delta(\omega - n\omega_s - \omega_0) \end{aligned}. \tag{3.40}$$

Taking the inverse Fourier transform of (3.40) learns that $b(t)$ has the structure:

$$b(t) = \sum_{n=-\infty}^{\infty} b_n e^{j(\omega_0 - n\omega_s)t}\ ,\ b_n \in \mathbb{C} \tag{3.41}$$

and that the complex amplitudes a_n and b_n are related through:

$$b_n = -\frac{1}{h}P(j\omega_0 - jn\omega_s)\Lambda(j\omega_0 - jn\omega_s)H(j\omega_0 - jn\omega_s)R_d\left(e^{j\omega_0 h}\right)F(j\omega_0)\cdot\sum_{n=-\infty}^{\infty} a_n. \tag{3.42}$$

Note that $R_d\left(e^{j\omega_0 h}\right)$ is invariant under a frequency shift of $n\omega_s$ because $R_d\left(e^{j\omega h}\right)$ is a periodic function of ω with period ω_s.

The sampled-data system leaves the structure of the signals (3.38) and (3.41) invariant. When signal (3.38) is sampled, the continuous-time signal consisting of a series of (possibly multivariable) sinusoids is mapped onto one single (possibly multivariable) discrete time sinusoid with amplitude

$$\sum_{n=-\infty}^{\infty} a_n \tag{3.43}$$

and frequency $\omega_0 - m\omega_s$, where m is such that $-\frac{\omega_s}{2} \le \omega_0 - m\omega_s < \frac{\omega_s}{2}$. It is the same digital signal processing which introduces the aliasing components of frequency $\omega_0 + n\omega_s$, $n \in \mathbb{Z}$ again.
In general, an LPTV (linear periodically time-varying) system with period $h = \frac{2\pi}{\omega_s}$ leaves the set of signals with structure (3.38) invariant. If Δ is restricted to be LPTV with period h, then Δ maps b of form (3.41) to a of form (3.38). The (squared) L_2 induced norm of M_{22} can then be expressed as:

$$\|M_{22}\|^2 = \sup_{\omega_0} \sup_{a \in L_2, a \neq 0} \frac{\sum \|b_n\|^2}{\sum \|a_n\|^2} \tag{3.44}$$

by using Parseval's theorem to express the energy of $a(t)$ and $b(t)$. Although no explicit relation is obtained, it is clear that robust stability condition (3.36) imposes some constraints on the controller (through $R_d(e^{j\omega h})$).

In sampled-data H_∞ control, the uncertainty Δ is usually assumed to be LPTV. A discussion in [9] shows that the use of this class of uncertainty generally yields very conservative controllers. This can be explained by the fact that the sampled-data control system is essentially in open loop between two sampling instants. The class of LPTV uncertainties allows an operator Δ with a very 'wild' behaviour in between sampling instants and a timid character on sampling times. The discrete-time controller is designed such that condition 3.36 is satisfied for the worst-case uncertainty operator. This results in a conservative design, because the worst-case uncertainty operator introduces more uncertainty than necessary. A better uncertainty class would be the class of LTI operators, but the analysis for this class of uncertainty is much more complex. In [7] an analysis based on the structured singular value (μ) is presented. [10] treats robust stability under structured LTI uncertainties, as opposed to arbitrary norm-bounded unstructured uncertainties considered here.

3.9 Performance robustness

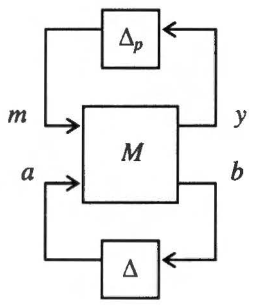


Figure 3.7. Set-up to study robust performance

Continuing to reason in the robust control framework of the previous section, a fancy performance feedback block is connected to the block diagram in figure 3.6 to study the performance robustness of the sampled-data system M (see figure 3.7). Standard arguments in robust control theory (in [11]) give a sufficient condition for robust performance (in a sense that the induced power gain from m to y is less than one for all Δ, $\|\Delta\| \le 1$):

$$\|M\| \le 1 \tag{3.45}$$

where the norm is the L_2 induced norm. This approach to guarantee robust performance results in an even more conservative controller design, because the special diagonal structure of the uncertainty:

$$\begin{pmatrix} \Delta_p & 0 \\ 0 & \Delta \end{pmatrix} \tag{3.46}$$

is not taken into account. The approach of [10] can be employed to facilitate non-conservative designs, but it is very complicated and not yet suited for reliable computation.
In the remaining part of this section, a sensitivity analysis is being carried out to study performance robustness. It yields some interesting insight into sampled-data systems.
If the performance of the sampled-data feedback control system is measured in terms of tracking and disturbance rejection, then

$$I-\frac{1}{h}P(j\omega)\Lambda(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)F(j\omega) \tag{3.47}$$

is an important term (see (3.27) and (3.20) and realise that in a practical situation $F(j\omega_0)r_0e^{j\omega_0 t} \approx r_0e^{j\omega_0 t}$ because $\omega_0 << \frac{\omega_s}{2}$). Denote (3.47) by $S_0(j\omega)$. This term relates the fundamental components of the reference signal and the tracking error, and the fundamental components of the disturbance and the output. The deviation of $S_0(j\omega)$ as a result of plant perturbations at frequencies $\omega + n\omega_s$, $n \in \mathbb{Z}$ is given by (first order approximation for SISO systems):

$$\delta S_0(j\omega)=\frac{\partial S_0(j\omega)}{\partial P(j\omega+n\omega_s)}\delta P(j\omega+n\omega_s). \tag{3.48}$$

The relative changes are characterised by:

$$\frac{\delta S_0(j\omega)}{S_0(j\omega)}=\left(\frac{P(j\omega+jn\omega_s)}{S_0(j\omega)}\frac{\partial S_0(j\omega)}{\partial P(j\omega+jn\omega_s)}\right)\cdot\frac{\delta P(j\omega+jn\omega_s)}{P(j\omega+jn\omega_s)}. \tag{3.49}$$

Straightforward differentiation yields:

$$\frac{P(j\omega+jn\omega_s)}{S_0(j\omega)}\frac{\partial S_0(j\omega)}{\partial P(j\omega+jn\omega_s)}=\begin{cases}-T^0(j\omega) & \text{for } n=0\\ \dfrac{T^0(j\omega)}{S^0(j\omega)}T^0(j\omega+jn\omega_s) & \text{for } n\neq 0\end{cases} \tag{3.50}$$

where:

$$T_0(j\omega)=\frac{1}{h}P(j\omega)\Lambda(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)F(j\omega). \tag{3.51}$$

From these equations it is clear that the performance of a sampled-data feedback system suspect to a sinusoidal exogenous input, is dependent on the plant uncertainty on infinitely many points in frequency. This in contrary to continuous-time feedback systems.

3.10 Limitations of sampled-data feedback control

Define the three hybrid sensitivity functions:

- *fundamental complementary sensitivity function*

$$T_0(j\omega)=\frac{1}{h}P(j\omega)\Lambda(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)F(j\omega) \tag{3.52}$$

- *fundamental sensitivity function*

$$S_0(j\omega)=I-\frac{1}{h}P(j\omega)\Lambda(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)F(j\omega) \tag{3.53}$$

- *n^th alias response function*

$$R_n(j\omega)=\frac{1}{h}\Lambda(j\omega-jn\omega_s)H(j\omega-jn\omega_s)R_d\left(e^{j\omega h}\right)F(j\omega). \tag{3.54}$$

The responses (3.20), (3.27), (3.32) and (3.33) can be expressed in terms of these hybrid sensitivity functions:

- *tracking*

$$r(t)-y(t)\approx S_0(j\omega)r_0e^{j\omega t}-\sum_{n\neq 0}P(j\omega-jn\omega_s)R_n(j\omega)r_0e^{j(\omega-n\omega_s)t} \tag{3.55}$$

- *disturbance rejection*

$$y(t)=S_0(j\omega)P(j\omega)d_0e^{j\omega t}-\sum_{n\neq 0}P(j\omega-jn\omega_s)R_n(j\omega)P(j\omega)d_0e^{j(\omega-n\omega_s)t} \tag{3.56}$$

- *measurement error*

$$y(t)=-T_0(j\omega)n_0e^{j\omega t}-\sum_{n\neq 0}P(j\omega-jn\omega_s)R_n(j\omega)n_0e^{j(\omega-n\omega_s)t} \tag{3.57}$$

- *actuator saturation*

$$\begin{aligned}u(t)\approx\ & P^{-1}(j\omega)T_0(j\omega)r_0e^{j\omega t}+\sum_{n\neq 0}R_n(j\omega)r_0e^{j(\omega-n\omega_s)t}\\ &-P^{-1}(j\omega)T_0(j\omega)n_0e^{j\omega t}-\sum_{n\neq 0}R_n(j\omega)n_0e^{j(\omega-n\omega_s)t}\\ &+P^{-1}(j\omega)S_0(j\omega)P(j\omega)d_0e^{j\omega t}-\sum_{n\neq 0}R_n(j\omega)P(j\omega)d_0e^{j(\omega-n\omega_s)t}\end{aligned} \tag{3.58}$$

.

Note again that $r_0e^{j\omega t}\approx F(j\omega)r_0e^{j\omega t}$ because the bandwidth of the reference signal is generally well below half the sampling frequency, the cut-off frequency of the anti-aliasing filter *F*. By the definition of $S_0(j\omega)$ and $T_0(j\omega)$, the fundamental property

$$S_0(j\omega)+T_0(j\omega)=I \tag{3.59}$$

is recovered. The control goals can be subdivided into two classes. The class of control goals that benefits from a 'small' (note that hybrid sensitivity functions are complex quantities) fundamental sensitivity can be listed as:
- tracking;
- and disturbance rejection.

The class which benefits from a 'small' fundamental complementary sensitivity function contains the control goals:
- avoidance of measurement noise;
- avoidance of actuator saturation (due to the reference and measurement noise).

The robust stability (with respect to multiplicative uncertainties) constraint can also be added to this last list. Although no explicit form was obtained, it is clear that condition 3.44 combined with (3.42) imposes a constraint on the fundamental complementary sensitivity function.
Restriction 3.59 implies that the ultimate control solution is necessarily a compromise between the control goals of the two classes. Moreover, it can be used to show that high performance and a high degree of performance robustness don't go together. For a good performance in terms of tracking and disturbance rejection, it is required to have a 'small' fundamental sensitivity function. A 'small' fundamental sensitivity function implies a relatively 'large' fundamental complementary sensitivity function, according to (3.59). Equation 3.50 learns that a 'large' fundamental complementary sensitivity results in a large sensitivity of the performance to plant perturbations. So the control goal performance robustness is conflicting with disturbance rejection and tracking.
Every control aim profits from a 'small' n^{th} alias response function, because then the effect of aliasing components is reduced, and the control system acts more like a continuous-time system. However, the transfer function of the discrete-time controller, $K(z)\big|_{z=e^{j\omega h}}$, is a periodic function of the frequency with period ω_s, and it shows an even symmetry because the impulse response of *K*(*z*) is real. This means that the discrete-time controller *K* cannot be designed to achieve independent control aims for the fundamental frequency range $\left[0, \frac{\omega_s}{2}\right)$ and the rest of the spectrum.

A well known limitation in continuous-time SISO feedback control is the Bode sensitivity integral. Its discrete-time counterpart is:

$$\int_0^{\frac{\omega_s}{2}} \ln\left|S_d\left(e^{j\omega h}\right)\right| d\omega = 0 \tag{3.60}$$

where

$$S_d\left(e^{j\omega h}\right) = \left.\frac{1}{1 + P_d(z)K(z)}\right|_{z=e^{j\omega h}} \tag{3.61}$$

and P_d is the ZOH transform of continuous-time plant *P*. Here it was assumed that S_d is strictly proper and contains no unstable poles or zeros. The Bode sensitivity integral states that the frequency band wherein disturbances are suppressed and the asymptotic tracking error is less than one, has to be complemented by a band where disturbances are amplified. The derivation of the Bode sensitivity integral is based on complex function analysis. Similar analysis can be performed for the fundamental sensitivity function. Perhaps it is not surprising that this analysis results in a integral constraint on the fundamental sensitivity function. Under some conditions holds:

$$\int_0^{\infty} \ln\left|S_0(j\omega)\right| d\omega = 0 \tag{3.62}$$

The derivation is given in [7].

3.11 Summary

In this chapter, control aims were studied for a general sampled-data feedback control system. Many results have a direct analogue in the continuous-time case (compare with chapter 3 from [11]), but some peculiar phenomena have no counterpart in the continuous-time feedback case:

- In general, it is not possible to asymptotically suppress a sinusoidal disturbance to an arbitrary degree by applying feedback, as is the case in continuous-time feedback control systems.
- The performance expressed in tracking and disturbance rejection abilities is sensitive to plant variations on an infinite amount of points in frequency.

The ultimate digital controller has to compromise between competing classes of control goals, and it has to make a well balanced trade-off between aims for the fundamental frequency range $\left[0, \frac{\omega_s}{2}\right)$ and the rest of the spectrum. The trade-off is portrayed in figure 3.8.

The fundamental property

$$S_0(j\omega) + T_0(j\omega) = I \tag{3.63}$$

and the Bode integral are identified as fundamental limitations in sampled-data feedback control.

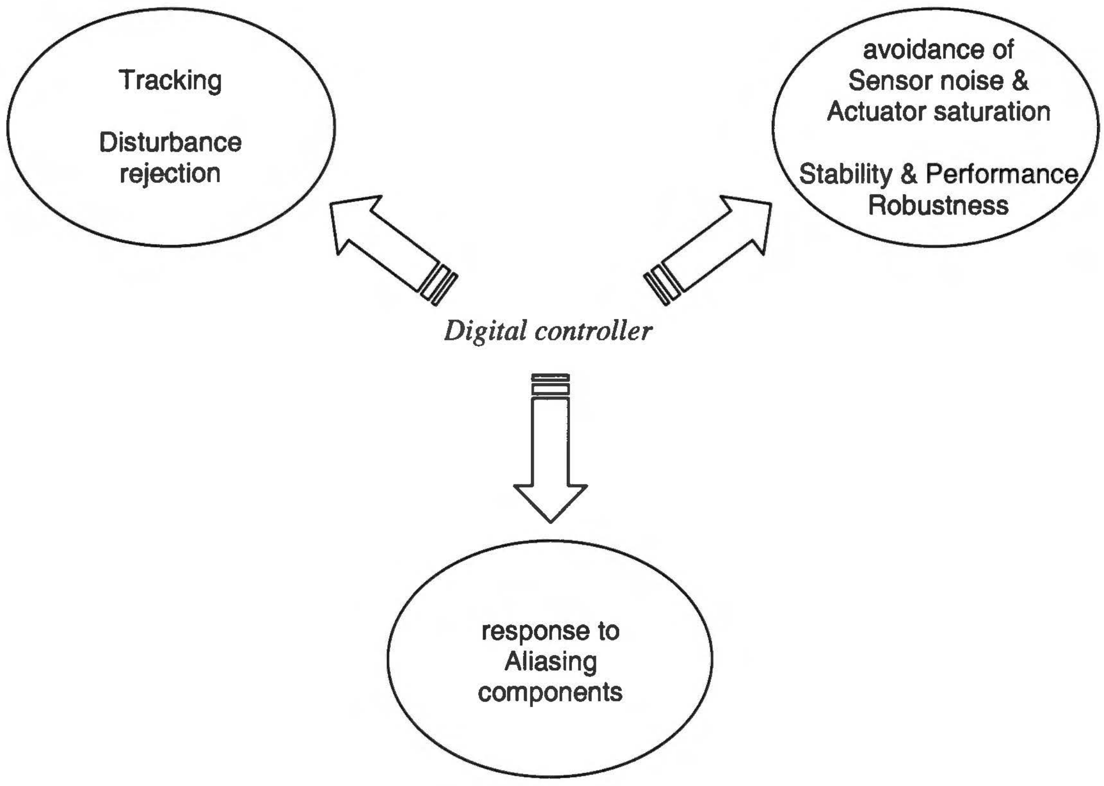

Tracking
Disturbance rejection
avoidance of Sensor noise & Actuator saturation
Stability & Performance Robustness
Digital controller
response to Aliasing components

Figure 3.8 Digital controller trade-off

4. Frequency Gains for Sampled-data Systems

4.1 Introduction

Although the frequency domain analysis of the previous chapter offers nice theoretical insights into the properties of sampled-data systems, it is not suited to numerically evaluate a sampled-data control system in the frequency domain. This is mainly due to the infinite summation over aliasing components in the derivations. This chapter introduces some computable frequency gains for the analysis of sampled-data systems, mainly due to [12]. The various frequency gains somehow relate (a portion of) the input power to (a portion of) the output power of the universal sampled-data control system in figure 4.1. Table 3.1 can be used to compute the frequency gains of interest in the block diagram of figure 3.1.

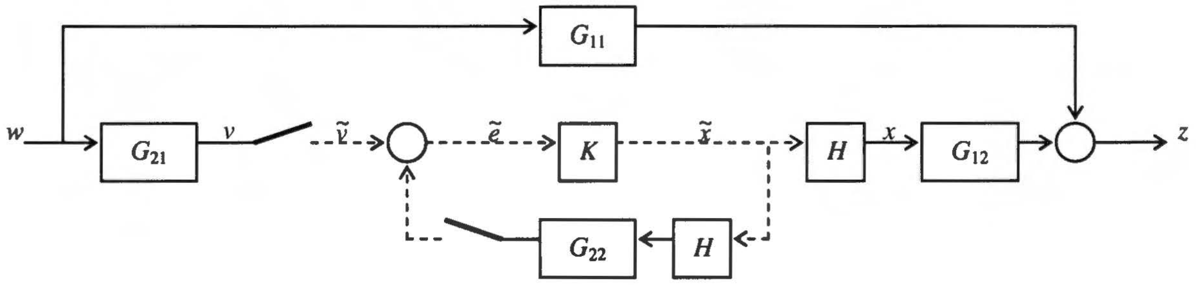


Figure 4.1 Universal block diagram for a sampled-data system

4.2 Discrete frequency gain (DFG)

The first notion of a frequency gain is the discrete frequency gain (*DFG*). It is actually the frequency response of a purely discrete-time system. If the system in figure 4.1 is appended with a sampler and prepended with a zero order hold, then the discrete-time transfer from *w*[*k*] to *z*[*k*] is given in the *z*-domain by:

$$Z(z)=[G_{11d}(z)+G_{12d}(z)R_d(z)G_{21d}(z)]W(z) \tag{4.1}$$

where

$$R_d(z)=K(z)(I-G_{22d}(z)K(z))^{-1} \tag{4.2}$$

and G_{11d} to G_{22d} are ZOH transforms of G_{11} to G_{22}. How to compute the ZOH transform of a general continuous-time system is explained in appendix A.2. The *DFG* is obtained by replacing z by $e^{j\omega h}$ and taking the maximum singular value (the maximum singular value represents the maximum gain of a MIMO system):

$$DFG_{zw}(\omega)=\sigma_{\max}\left(G_{11d}\left(e^{j\omega h}\right)+G_{12d}\left(e^{j\omega h}\right)R_d\left(e^{j\omega h}\right)G_{21d}\left(e^{j\omega h}\right)\right). \tag{4.3}$$

For SISO systems this reduces to:

$$DFG_{zw}(\omega)=\left|G_{11d}\left(e^{j\omega h}\right)+G_{12d}\left(e^{j\omega h}\right)R_d\left(e^{j\omega h}\right)G_{21d}\left(e^{j\omega h}\right)\right|. \tag{4.4}$$

Note that the *DFG* is a periodic function of frequency with period ω_s. **As the *DFG* is defined using the sampled output of a sampled-data system, it does not account for the intersample behaviour.** If $w[k]$ is restricted to:

$$w[k] = w_0 e^{j\omega kh} \tag{4.5}$$

then $DFG_{zw}(\omega)$ relates the input and output power through:

$$DFG_{zw}(\omega) = \max_{w \neq 0} \frac{P(z[k])}{P(w[k])} = \max_{\|w_0\|=1} P(z[k]) \tag{4.6}$$

where the power of a discrete-time signal is defined as:

$$P(z[k]) = \lim_{N\to\infty} \sqrt{\frac{1}{2N} \sum_{k=-N}^{N} \|z[k]\|^2}\ . \tag{4.7}$$

$A{\cdot}\omega_0 \rightarrow A,\omega_0$

4.3 Fundamental frequency gain (FFG)

The response of a sampled-data system to a continuous-time sinusoid $w(t) = w_0 e^{j\omega t}$ has the structure:

$$z(t) = \sum_{n=-\infty}^{\infty} z_n e^{j(\omega - n\omega_s)t}\ . \tag{4.8}$$

The fundamental frequency gain (*FFG*) is the ratio of the amplitudes of the fundamental components:

$$FFG_{zw}(\omega) = \frac{\|z_0\|}{\|w_0\|}\ . \tag{4.9}$$

Using (3.11), FFG_{zw} can be expressed as:

$$FFG_{zw}(\omega) = \sigma_{\max}\left(G_{11}(j\omega) + \frac{1}{h} G_{12}(j\omega) H(j\omega) R_d\left(e^{j\omega h}\right) G_{21}(j\omega) \right). \tag{4.10}$$

For SISO systems this reduces to:

$$FFG_{zw}(\omega) = \left| G_{11}(j\omega) + \frac{1}{h} G_{12}(j\omega) H(j\omega) R_d\left(e^{j\omega h}\right) G_{21}(j\omega) \right|. \tag{4.11}$$

If $w(t)$ is restricted to:

$$w(t) = w_0 e^{j\omega t} \tag{4.12}$$

then the *FFG* can be expressed in terms of power, as the power of the fundamental component in the response z divided by the power in the input signal w.
Note that the fundamental sensitivity function and the fundamental complementary sensitivity of the previous chapter can be expressed as:

$$S_0(j\omega) = FFG_{yd}(\omega) \cdot P^{-1}(j\omega) \approx FFG_{er}(\omega) \tag{4.13}$$
$$T_0(j\omega) = -FFG_{yn}(\omega) \approx FFG_{yr}(\omega). \tag{4.14}$$

response to sinusoid

4.4 Performance frequency gain (PFG) & alias response power (ARP)

In the case of the performance frequency gain (*PFG*) the exogenous input w is again restricted to (4.12), a single (possible multivariable) sinusoid. The *PFG* compares the *total* response power to the power of a sinusoidal exogenous input:

$$PFG_{zw}(\omega) = \max_{w \neq 0} \frac{P(z(t))}{P(w(t))} = \max_{\|w_0\|=1} P(z(t)) \tag{4.15}$$

$A_i^{\omega_0} \rightarrow \underline{\Sigma A_j^{h\omega_0}}$ all

where the power of a continuous-time signal is defined as:

$$P(z(t)) = \lim_{T \to \infty} \sqrt{\frac{1}{2T} \int_{-T}^{T} \|z(t)\|^2 dt} \,. \tag{4.16}$$

The *PFG* is a suited quantity to study the asymptotic tracking error of a sampled-data control system due to a sinusoidal reference, or the asymptotic rejection of a sinusoidal disturbance, because it takes the aliasing components into account! The portion of the power which corresponds to the aliasing components in the response, is called the *alias response power* (*ARP*):

$$ARP_{zw}(\omega) = PFG_{zw}(\omega) - FFG_{zw}(\omega) \,. \tag{4.17}$$

In appendix A.3, an algorithm is presented to compute the *PFG* for an arbitrary sampled-data system.

4.5 Robustness frequency gain (RFG)

The stability robustness analysis of section 3.8 learned that applying an exogenous signal of structure:

$$w(t) = \sum_{n=-\infty}^{\infty} w_n e^{j(\omega + n\omega_s)t} \,, \; w_n \in \mathbb{C} \tag{4.18}$$

to a sampled-data system results in a response of the same structure:

$$z(t) = \sum_{n=-\infty}^{\infty} z_n e^{j(\omega + n\omega_s)t} \,, \; z_n \in \mathbb{C}. \tag{4.19}$$

The robustness frequency gain (*RFG*) is defined as:

$$RFG_{zw}(\omega) = \sup_{w \neq 0} \frac{P(z(t))}{P(w(t))} \tag{4.20}$$

where w and z are assumed to be structured as (4.18) and (4.19). The *RFG* is suited to study robustness properties. For example, the sampled-data system in figure 3.6 is robustly stable if for every frequency:

$$\frac{1}{RFG_{ba}(\omega)} \leq RFG_{\Delta}(\omega) \tag{4.21}$$

where RFG_{ba} denotes the *RFG* from input *a* to output *b* of system *M*, and RFG_{Δ} is the *RFG* of LPTV operator Δ. The supremum of the *RFG* over frequency is equal to the L_2 induced norm of a sampled-data system. The *RFG* is a periodic function of frequency with period ω_s because of the structure of the signals being compared ((4.18) and (4.19)), so the supremum just has to be taken over one period. The computation of the *RFG* for the general case is quite complicated [13]. It involves a bisection search for every frequency in a frequency grid. For systems with $G_{11} = 0$, the computation simplifies, and a method is given in [14]. This assumption seems restrictive but it allows to study robustness against additive and multiplicative plant uncertainties. If one further assumes that $G_{21} = I$, then the *RFG* equals the *PFG*. This fact is easy to verify in the case of the example in section 3.8. First, note that both RFG_{ba} and PFG_{ba} can be expressed as:

$$\sup_{P(a(t))=1} P(b(t)) \tag{4.22}$$

(whether the supremum or the maximum has to be used is not considered here).
The difference is in what kind of signal *a(t)* is used. For the *PFG* case, *a* is a single sinusoid. In the *RFG* case, *a* is a series of sinusoids as in (3.38) or (4.18). Assume that the transfer from *a* to *b* is SISO. The power of the response *b* is obtained by using Parseval's theorem in conjunction with (3.42) where $F(j\omega)=1$:

$$\begin{aligned} P(b(t)) &= \sqrt{\sum_{n=-\infty}^{\infty} |b_n|^2} \\ &= \frac{1}{h}\left|R_d\left(e^{j\omega h}\right)\right| \cdot \sqrt{\sum_{n=-\infty}^{\infty} \left|P(j\omega - jn\omega_s)\Lambda(j\omega - jn\omega_s)H(j\omega - jn\omega_s)\right|^2} \cdot \sqrt{\sum_{n=-\infty}^{\infty} |a_n|^2} \\ &= \frac{1}{h}\left|R_d\left(e^{j\omega h}\right)\right| \cdot \sqrt{\sum_{n=-\infty}^{\infty} \left|P(j\omega - jn\omega_s)\Lambda(j\omega - jn\omega_s)H(j\omega - jn\omega_s)\right|^2} \cdot P(a(t)) \end{aligned} \tag{4.23}$$

.

As for both cases the supremum is taken over *a*, *P*(*a*(*t*)) = 1, the *PFG* and *RFG* are equal.

Mathematically, the sampled-data H_∞ control framework is very appealing, because it studies the mapping from L_2 signals to L_2 signals, just as in the continuous-time H_∞ control framework. If some kind of loop shaping procedure is applied in the sampled-data H_∞ control framework, the frequency gain being shaped is the *RFG*. However, in designing sampled-data (servo) control systems, one is actually interested in how well references of a certain frequency can be tracked, and how well disturbances of a certain frequency can be rejected. This makes the *PFG* the frequency gain of interest. One is not concerned with the mapping of *arbitrary* L_2 signals from the input to the output. This makes sampled-data H_∞ control unsuited for the design of tracking controllers. It can be used to study robustness issues, but it might turn out to be very conservative, as was pointed out in sections 3.8 and 3.9. For continuous-time systems, the role of the *FFG, PFG* and *RFG* is fulfilled by one single function: the ordinary continuous-time frequency response.

4.6 Fidelity index

Consider the standard sampled-data feedback control system in figure 3.1 and assume that all the blocks represent SISO transfers. Let's study the frequency gains DFG_{yr} and PFG_{yr}. The first frequency gain can be calculated from (4.4) and data from table 3.1 as:

$$DFG_{yr}(\omega)=\left|R_d\left(e^{j\omega h}\right)P_d\left(e^{j\omega h}\right)\right|=\frac{1}{h}\left|R_d\left(e^{j\omega h}\right)\right|\cdot\left|\sum_{n=-\infty}^{\infty}P(j\omega-jn\omega_s)\Lambda(j\omega-jn\omega_s)H(j\omega-jn\omega_s)\right| \quad (4.24)$$

where sampling formula 3.3 was used to write the ZOH transform of $P\Lambda$ as an infinite summation. PFG_{yr} can be calculated by observing (3.19) and applying Parseval's theorem.

$$PFG_{yr}(\omega)=\frac{\sqrt{\sum|y_n|^2}}{|r_0|}=\frac{1}{h}\left|R_d\left(e^{j\omega h}\right)\right|\cdot\sqrt{\sum_{n=-\infty}^{\infty}\left|P(j\omega-jn\omega_s)\Lambda(j\omega-jn\omega_s)H(j\omega-jn\omega_s)\right|^2} \quad (4.25)$$

Due to the Schwarz inequality:

$$\left|\sum_{n=-\infty}^{\infty}P(j\omega-jn\omega_s)\Lambda(j\omega-jn\omega_s)H(j\omega-jn\omega_s)\right|^2\leq\sum_{n=-\infty}^{\infty}\left|P(j\omega-jn\omega_s)\Lambda(j\omega-jn\omega_s)H(j\omega-jn\omega_s)\right|^2. \quad (4.26)$$

From this one can conclude that DFG_{yr} is a lower bound on PFG_{yr}.
Define the fidelity index Φ_d as:

$$\Phi_d\left(e^{j\omega h}\right)=\sqrt{\frac{\sum_{n=-\infty}^{\infty}\left|P(j\omega-jn\omega_s)\Lambda(j\omega-jn\omega_s)H(j\omega-jn\omega_s)\right|^2}{\left|\sum_{n=-\infty}^{\infty}P(j\omega-jn\omega_s)\Lambda(j\omega-jn\omega_s)H(j\omega-jn\omega_s)\right|^2}}\geq 1 \quad \neq \quad \sqrt{\frac{PFG^2}{DFG}} \quad \approx DFG^2 \quad (4.27)$$

then

$$PF\overset{G}{\cancel{R}}_{yr}(\omega)=\Phi_d\left(e^{j\omega h}\right)\cdot DF\overset{G.}{\cancel{R}}_{yr}(\omega). \qquad \text{R or G?.} \quad (4.28)$$

The fidelity index is a measure for how close the PFR and DFR are, i.e. a measure for intersample activity. **Note that the fidelity index is independent of the controller!**

4.7 Examples

Example 1: Intersample oscillation

Consider the continuous-time plant

$$P(s)=\frac{1}{s(s+1)}. \quad (4.29)$$

As it turns out, this plant is a very simplified model of a position controlled DC motor (see chapter 6). The plant is being sampled with a sampling rate of 10 Hz. No anti-aliasing filter is applied. A digital controller is being designed using a discrete-time design method.
The ZOH transform of the plant is given by:

$$P_d(z)=\frac{0.0048374\cdot(z+0.9672)}{(z-1)(z-0.9048)}. \qquad (4.30)$$

The discrete-time controller K compensates the plant zero and pole inside the unit circle. The gain is chosen such that the resulting closed-loop pole is placed at the origin. This kind of design is called a dead beat controller. The design yields:

$$K(z)=\frac{206.7265(z-0.9048)}{z+0.9672}. \qquad (4.31)$$

The closed loop transfer from reference input to plant output is given by z^{-1}, a sample time delay. Figure 4.2 shows the step response of the closed loop system. Indeed, the discrete-time step signal is being tracked with a delay of one sampling time, but the continuous-time response shows a severe intersample oscillation. If one had studied the frequency gains in figure 4.3, one would be warned by the oscillatory peaks in FFG_{yr} and PFG_{yr} and the discrepancy between the discrete frequency gain and the frequency gains accounting for the intersample behaviour (*FFG* and *PFG*).

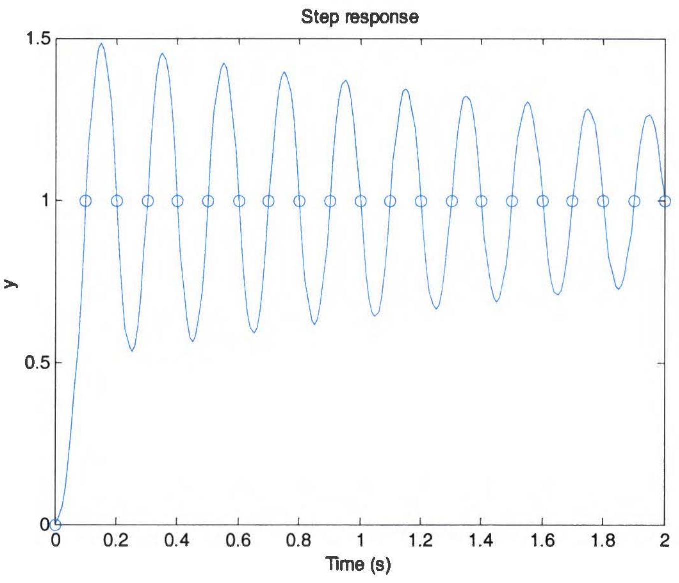
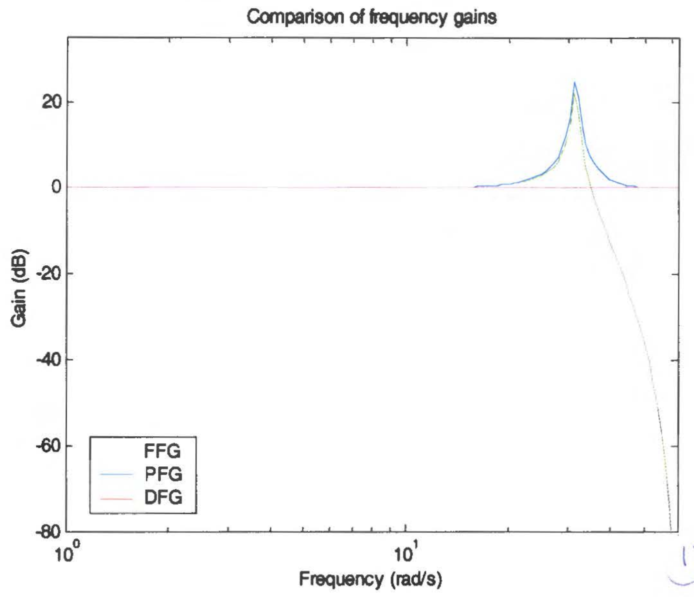


Figure 4.2 Step response

In this case, the intersample oscillation is caused by the compensation of the negative real axis zero of the discretised plant. The controller pole in $z = -0.9672$ corresponds to a heavily oscillating response. The zero of the discretised plant makes this behaviour unobservable on sampling instants, but the fluctuating actuator signal has its effect on the continuous-time response. For this reason, it is advised in [2] not to compensate any negative real axis zeros, in discrete-time design.



10 Hz =

62 rad/s

Figure 4.3 Warning for intersample oscillation

Example 2: Disturbance rejection

An integrator plant is compensated by the discrete-time controller

$$K(z)=\frac{9.1888\cdot\left(z^2-1.239z+0.5468\right)}{z^2+0.6442z+0.9995} \tag{4.32}$$

with sampling rate 10 Hz, in order to suppress a sinusoidal disturbance of 3 Hz. The frequency response $K\left(e^{j\omega h}\right)$ is shown in figure 4.4. A large feedback gain is applied at 3 Hz in order to suppress the disturbance.

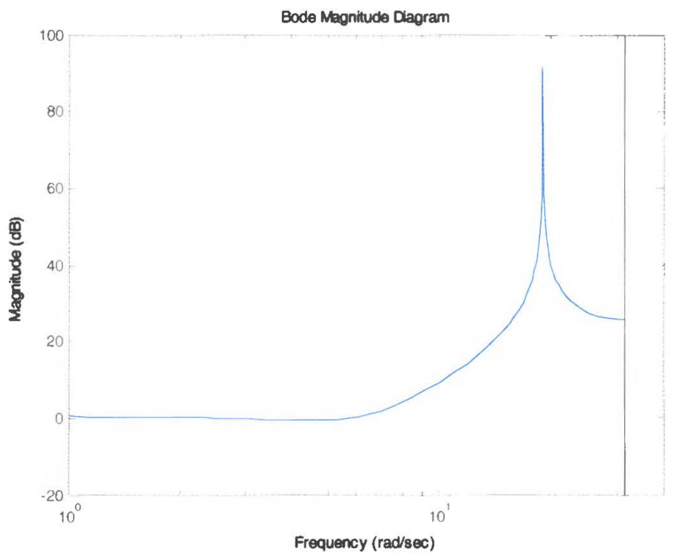


Figure 4.4 Frequency response of compensator

The sampled-data system was simulated. Figure 4.5 shows the asymptotic response to a sinusoidal disturbance of 3 Hz with amplitude one. Clearly, the disturbance is attenuated, but the effect of aliasing components is apparent.

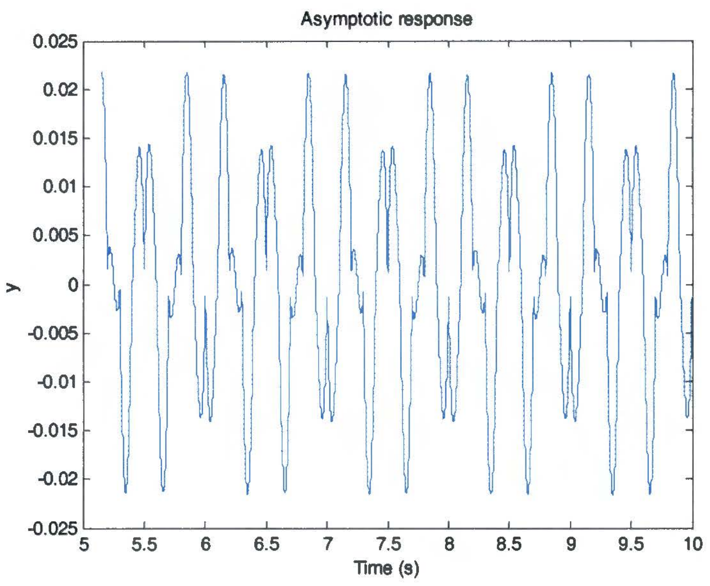


Figure 4.5 Asymptotic response to sinusoidal disturbance

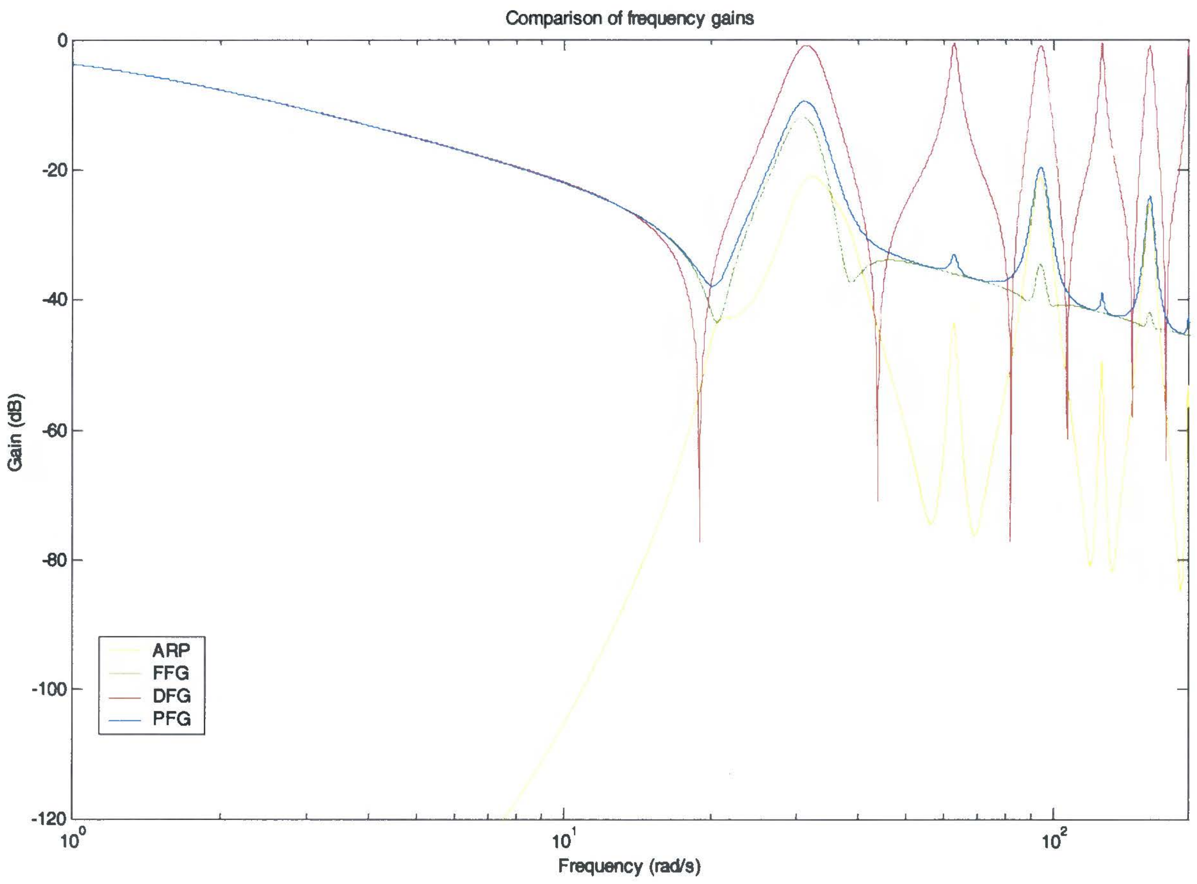


Figure 4.6 Comparison of frequency gains

It is revealing to study the frequency gains in the figure above. The shape of PFG_{yd} is quite different from DFG_{yd}. As was pointed out in section 3.5, a sinusoidal disturbance can not be attenuated to an arbitrary degree by applying a large feedback gain at that frequency. However, there is a small notch in the FFG_{yd} and PFG_{yd} about 3 Hz, but it is small compared to the notch in the *DFG* and at a slightly different frequency! The high frequency content in the asymptotic response is explained by the significant value of the *ARP* in the frequency range around 3 Hz. Further, it is striking that there are no notches in PFG_{yd} at frequencies 13Hz, 23 Hz,... , but PFG_{yd} shows an overall decay of 20 dB per decade. This is due to the factor $P(j\omega)$ in both terms of (3.56). This is a nice property, because it assures the rejection of high frequency disturbances for every plant, as all physical plants are essentially low pass.

5 Time Domain Criteria for Sampled-data Systems

5.1 Integral quadratic criteria

The performance frequency gain derived in the previous section is a suited criterion to study the asymptotic tracking performance and disturbance rejection properties of a sampled-data control system. Regarding the tracking behaviour, one is often not really interested in the asymptotic behaviour, but one wants to know how well the control system can track a certain setpoint profile (especially in motion control systems). If the tracking performance is insufficient, the closed-loop bandwidth can be increased. However, the tracking performance is dependent on the whole shape of a frequency response and not merely on the bandwidth, a point where the response crosses a certain level. Some kind of time domain criterion is better suited. An often encountered time domain criterion is an integral quadratic criterion. It has nice computational properties.
For purely discrete-time systems, the cumulative quadratic error on sampling instants is expressed as:

$$J_1 = \sum_{k=0}^{\infty} \left(y(kh) - r(kh)\right)^2 \tag{5.1}$$

where y is the plant output and r the reference signal. This criterion is related to the discrete H_2 norm of a discrete-time system. The H_2 norm describes the energy of the impulse response of a system. Criterion 5.1 equals the discrete H_2 norm of the system in figure 5.1, where the response of R to a Kronecker delta pulse represents the reference signal r.

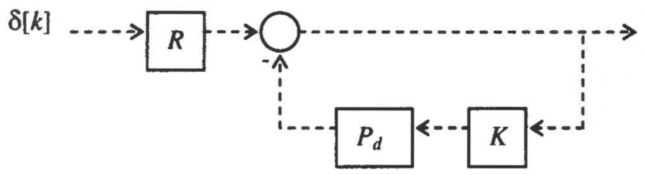


Figure 5.1 Set-up for discrete H_2 norm

The discrete-time controller K which minimises criterion 5.1 is given by the solution to the discrete H_2 optimal control problem (see [3]). This solution involves the computation of the solution to two algebraic Ricatti equations. It can be regarded as the deterministic counterpart of linear quadratic Gaussian (LQG) control. The order of the optimal linear controller is equal to the order of P_d and R combined.
For example, let P_d be the ZOH transform of

$$P(s) = \frac{1}{(10s+1)(25s+1)} \tag{5.2}$$

with sampling time h = 1. The optimal discrete-time controller which minimises criterion 5.1 for a step response ($R(z) = \frac{1}{z-1}$), achieves the response which is plotted as the red curve in figure 5.2. The optimal controller shows dead beat behaviour and a severe intersample oscillation. The step response of the system with the optimal controller minimising

$$J_2 = \int_0^{\infty} \left(y(t) - r(t)\right)^2 dt \tag{5.3}$$

for a step reference, is plotted as the blue curve. This controller achieves a better response because it minimises the actual criterion of interest: the continuous-time integral quadratic tracking error. This criterion is equal to the energy of the response of the system in figure 5.3 to a Dirac delta pulse. The

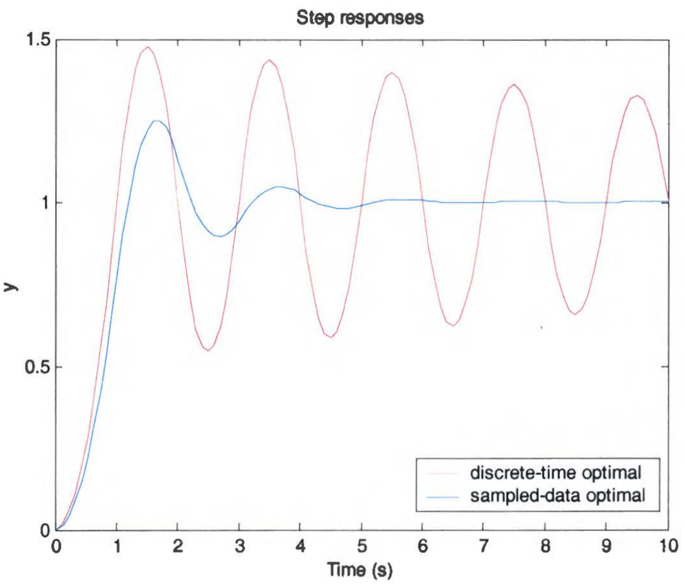


Figure 5.2 Step responses for optimal discrete and simple sampled-data H_2 controllers

amount of energy characterises the 'simple' H_2 measure of the sampled-data system in figure 5.3. The nomenclature is adopted from [3].

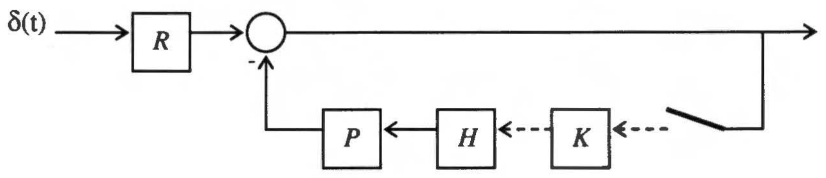


Figure 5.3 Setup for simple sampled-data H_2 measure

Although the simple H_2 measure is an interesting criterion, the computation requires that the Dirac delta pulse is applied at $t = 0$. As the mixed continuous-time/discrete-time system is periodically time-varying, the responses to a signal $r(t-\eta)$ differ for $\eta \in [0,h)$. A natural extension of an integral quadratic criterion to linear periodically time-varying (LPTV) systems with period h is:

$$J_3 = \frac{1}{h}\int_0^h \int_0^\infty (y(t) - r(t-\eta))^2 \, dt d\eta \, . \tag{5.4}$$

This criterion averages the continuous-time integral quadratic tracking error over the period h. It is equal to the generalised H_2 measure, or the average energy of the response to $\delta(t-\eta)$, $\eta \in [0,h)$.

A second example shows the difference between the two sampled-data H_2 measures. Consider the set-up in figure 5.4. A discrete-time controller with sampling time $h = 1$ is designed such that the output y mimics a triangular input r as closely as possible. The reference signal starts at a time during the first sampling interval (see figure 5.5). The optimal solution for criterion (5.4) shows a better response than the optimal solution for criterion (5.3), because the latter solution assumes the start of the reference signal to be synchronised to the sampling action. In the computations, the triangular reference profile was approximated by the impulse response of system with a rational transfer function.

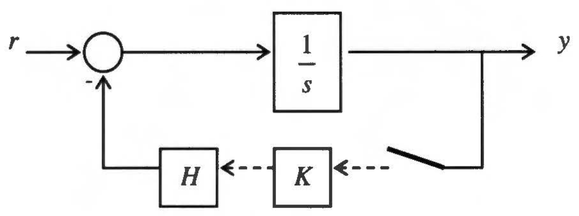


Figure 5.4 Control setup

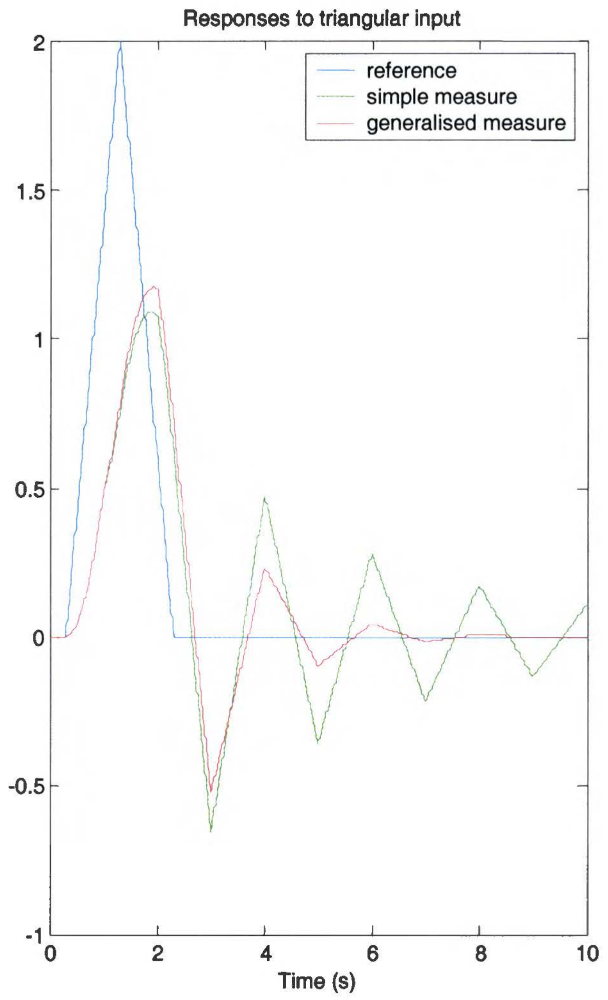


Figure 5.5 Optimal responses to a triangular input

The examples in this section were taken from [3]. They are also included in the DirectSD toolbox for MATLAB [19]. This toolbox is dedicated to optimal H_2 and H_∞ control using polynomial methods [4]. Examples of solutions using the lifting approach are included for comparison. In the lifting approach, the sampled-data optimal control problem is transformed to a norm-equivalent discrete-time optimal control problem. The model structures considered in this toolbox are too restrictive to be usefull for motor controller design. Own MATLAB tools have been developed, which are devoted to the computation of integral quadratic tracking errors for discrete-time motor controllers.
In chapter 7, two kinds of integral quadratic tracking error criteria for a fixed structure motor controller are considered. One criterion is used for calculating the continuous-time integral quadratic tracking error with respect to a setpoint profile obtained trough discrete-time integration (synchronous case). The second criterion computes the average continuous-time integral quadratic tracking error as result from applying an arbitrary continuous-time setpoint profile (asynchronous case). This latter case allows an independent design of the setpoint profiles, as opposed to the synchronous case where the profiles are dependent on the sampling time. The derivations in appendix A.5 and A.6 are based on the lifting approach to optimal sampled-data control.

5.2 Discussion

H_2 optimal sampled-data control offers a nice framework for designing controllers which are optimal with respect to some integral quadratic criterion. The lifting solution presented in [3] allows to transform the sampled-data problem to a norm-equivalent discrete-time problem. The well known solution to this problem is given in terms of the solutions to two algebraic Ricatti equations. The order of this optimal linear controller, however, equals the order of the plant P and the reference generator R combined. Typically, the system R is of high order due to the rational approximation. The high order controllers might be reduced using model reduction techniques, but this indirect approach sacrifices the optimality of the controller. In chapter 7 the direct approach is being applied: optimisation of a fixed structure controller with respect to an integral quadratic criterion.
The integral quadratic criteria presented in section 5.1 only involve the continuous-time tracking error. Optimal controller designs for these kind of criteria might be too optimistic because no errors were taken into account. The H_2 control framework allows the inclusion of stochastic components like white measurement noise (LQG control!) in the problem definition. The criterion becomes a mixed deterministic/stochastic criterion, like:

$$J = \int_0^\infty e^2(t)dt + \rho E\{e^2\}. \tag{5.5}$$

For sampled-data systems however, there are some problems with the interpretation of this kind of criteria. For example: what is the stochastic interpretation of the generalised H_2 norm of a sampled-data system? In [16] the interpretation as an asymptotic average of the variance is motivated. A second problem is, how to choose the scaling parameter ρ. There is no single sensible explanation on how to choose this scaling [17].
In [18] the existence of an optimal sampling rate is investigated, when the criterion is extended with a term describing the cost of sampling. An iterative algorithm is presented which can be used to compute the optimal sampling rate. The exogenous input signal, however, is restricted to a discrete-time sequence. The integral quadratic criterion can further be extended with a term weighing the actuator signal, to prevent actuator saturation.
The main problem in the practical application of these kind of extended cost functions is the weighing of the deterministic and stochastic components, the relative importance of the actuator signal and potentially, the cost of sampling. A practical design process would consist of iteratively adjusting the relative weights until a satisfactionary controller design is obtained. The *relative* weights have to be considered, because one can hardly judge if the controller is doing any good on the basis of the absolute values of the components in the cost function. In chapter 7, the explainability of the frequency gains is combined with the suitability of the continuous-time integral quadratic error criterion to optimisation, in a controller optimisation procedure.
The continuous-time integral quadratic tracking error criterion is also a suited criterion to optimise the feedforward part of a controller. The H_2 optimal solution allows the (offline) computation of an optimal feedforward signal. If the setpoint profiles are constant (repetitive), this signal can be stored in the micro controller memory. If the desired setpoint profile is constant and known in advance, one might even try to boost the performance by allowing a non-causal feedforward signal (preview control) [31]. Examples of sampled-data preview control are included in the DirectSD toolbox. The repetitive nature of the setpoint profiles allows the incorporation of an iterative learning or repititve control scheme. Such a scheme can be interpreted in the sampled-data framework too [34].
The sampled-data H_2 control design framework offers an elegant computational solution and opportunities to combine several control aims in one criterion. However, the application and interpretation of combined deterministic/stochastic criteria for hybrid continuous-time/discrete-time systems needs further research.

6. Modelling a DC Motor Drive

6.1 Introduction

For small servo drive applications, a permanent magnet DC (PMDC) motor is frequently used, because of its control ease. In this kind of motor, the stator flux is produced by permanent magnets. To keep the motor turning, the current directions in the rotor have to be changed during a revolution (commutation). This is done through a commutator structure with carbon brushes. The main disadvantage of a PMDC motor is the wear of the brushes and the need to replace them.
Usually, a switch-mode servo amplifier is used to supply the DC motor, because of its energy efficiency. Position feedback is applied by an optical encoder disk, because it is cheap and it offers a convenient interface to the digital control hardware.
The goal of this chapter is to show how to obtain a (linear) model of a DC motor drive application, and to discuss some typical parts and phenomena occurring in these kind of drives.

6.2 DC motor

Consider the single current conducting rotor winding in a magnetic field in the figure below. A torque is produced according to Ampère's law and the winding starts turning. Once the winding is in vertical position, the current sign has to be changed. This is done by a commutator structure like the one in figure 6.2. The rotor currents are transferred from the stationary frame to the rotor through brushes. The copper segments on the commutator are each connected to one end of a rotor winding. The commutator is placed on the rotor shaft, so as the rotor turns, the commutator switches the rotor currents in the right direction and the rotor will turn continuously.

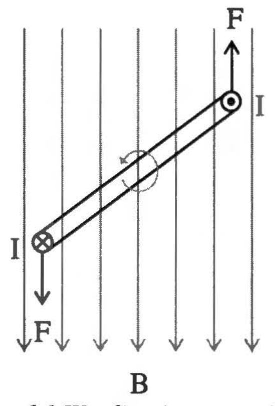


Figure 6.1 Winding in magnetic field

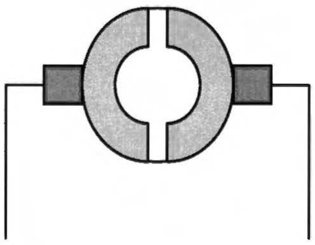


Figure 6.2 Commutator struture

The torque acting on the winding in figure 6.1 is given by:

$$T_{em} = 2B\ell Ir\cos\theta_m \tag{6.1}$$

where r is the radius of the rotor winding, ℓ is the rotor length and θ_m the angle between the normal of the plane of the winding and the magnetic field lines. Due to the current commutation, the torque during one revolution is:

$$T_{em} = \left|2B\ell Ir\cos\theta_m\right|. \tag{6.2}$$

When a winding rotates through the stator field it experiences a varying flux which generates a voltage across the ends of the winding. This voltage is called the back electro-motive force (back EMF). It can be calculated using Faraday's law:

$$e = -\frac{d\Phi}{dt}. \tag{6.3}$$

A single rotating winding in a uniform magnetic field like the one in figure 6.1 experiences a flux

$$\Phi = 2B\ell r\cos\theta_m. \tag{6.4}$$

The induced voltage is obtained as:

$$e = -\frac{d\Phi}{dt} = 2B\ell r\sin\theta_m \cdot \frac{d\theta_m}{dt} = 2B\ell r\omega_m\sin\theta_m \tag{6.5}$$

where ω_m is the angular speed of the rotor winding. Because of commutation the polarity is changed every half a revolution and the voltage across the brushes is given by:

$$E = \left|2B\ell r\omega_m\sin\theta_m\right|. \tag{6.6}$$

(6.2) and (6.6) are approximated by:

$$T_{em} = k_T I \tag{6.7}$$
$$E = k_E\omega_m. \tag{6.8}$$

k_E and k_T are dependent on the construction of the motor. They are numerically equal, which can be concluded from equating the electrical power and the mechanical power in steady state. Practical DC motors have multiple poles. The approximations get better as more commutations occur during one revolution. However, ripples in the torque, current and back EMF remain.
(6.7) and (6.8) describe the idealised DC motor in the equivalent circuit in figure 6.3. R models the resistance of the leads, the brushes and the rotor windings. L corresponds to the inductance of the rotor windings.
Motor parameters may vary heavily under manufacturing tolerances and temperature changes. Especially the temperature effects are large. For ferrite motors, the torque constant decreases with 2% per 10 K temperature drop. The electrical resistance increases with temperature; 4% per 10 K.

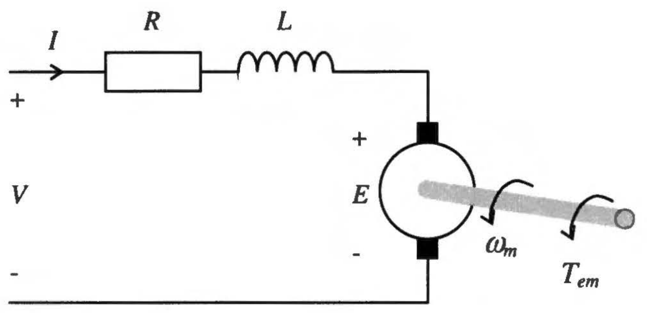


Figure 6.3 Equivalent circuit of a DC motor

6.3 Transmission & load

All the inertia of the drive is transformed onto the motor shaft by using the energy conservation law. In this way, the inertia of the rotor, pulleys, gears and load are incorporated in one total inertia (*J*). This allows a simple electromechanical model for the drive system, see figure 6.4. *B* represents the total physical damping coefficient.

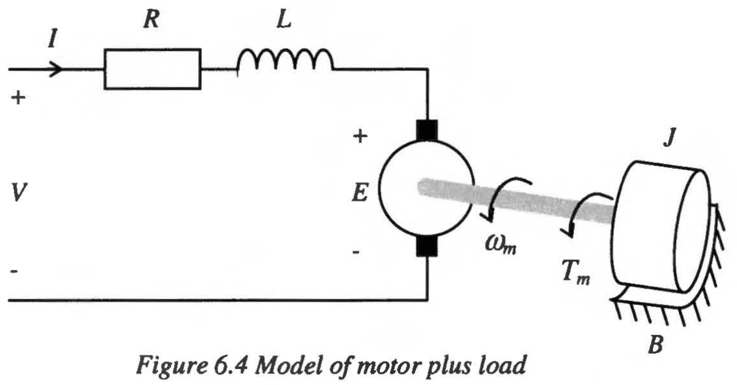


Figure 6.4 Model of motor plus load

The transfer from input voltage to angular speed in the figure above can be expressed in the Laplace domain as:

$$\frac{\omega_m(s)}{V(s)} = \frac{k_T}{(Js+b)(Ls+R)+k_E k_T}. \tag{6.9}$$

This transfer function is expressed in terms of the physical parameters of the drive system. A second often encountered transfer function has the form:

$$\frac{\omega_m(s)}{V(s)} = \frac{1}{k_E(s\tau_m+1)(s\tau_e+1)}. \tag{6.10}$$

Here, τ_m and τ_e are experimental parameters of the drive system. The electrical time constant τ_e corresponds to the time it takes the current to reach 63% of the end value after a voltage step is applied, assuming a constant rotor speed. It is simply the time constant of the *LR* combination in figure 6.4:

$$\tau_e = \frac{L}{R}. \tag{6.11}$$

Usually $\tau_e \ll \tau_m$, so the electrical time constant can be neglected resulting in *I(t)* = *V(t)/R*. In this case, the differential equation governing the drive system can be expressed as:

$$J\dot{\omega}_m = k_T \frac{V - k_E\omega_m}{R} - B\omega_m = k_T \frac{V}{R} - \left(\frac{k_T k_E}{R} + B\right)\omega_m = k_T \frac{V}{R} - D\omega_m . \tag{6.12}$$

A voltage step of magnitude ΔV results in a velocity response:

$$\omega_m(t) = \frac{k_T}{DR}\left(1 - e^{-\frac{t}{\tau_m}}\right)\Delta V \tag{6.13}$$

where

$$\tau_m = \frac{J}{D}. \tag{6.14}$$

If the physical damping coefficient B is small, then

$$D \approx \frac{k_T k_E}{R} \tag{6.15}$$

and (6.13) reduces to:

$$\omega_m(t) \approx \frac{1}{k_E}\left(1 - e^{-\frac{t}{\tau_m}}\right)\Delta V . \tag{6.16}$$

D is an artificial damping coefficient (the unit is also Nm·s/rad). In some datasheets D is called the *zero source impedance constant* and B is called the *infinite source impedance constant*.
If the angular position of the motor is of interest, an integrator has to be added to the transfer, like is done in the block diagram below. The block diagram also shows where the disturbing torques (T_d) are coupled in. If the electrical time constant is neglected, the disturbing torque has the same effect as a voltage error:

$$T_d = k_T \frac{\delta V}{R} \quad \Leftrightarrow \quad \delta V = \frac{R}{k_T} T_d . \tag{6.17}$$

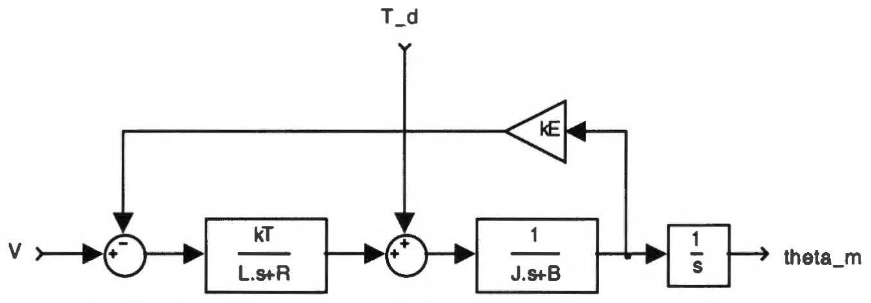


Figure 6.5 Block diagram for motor plus load

The model derived in this section is a basic general model for a DC motor drive system. Some extensions may be necessary to model particular dynamics of the load. There might be for example flexibilities in the transmission (flexible drive belt) or the load, which introduce some marginally damped modes whose frequency might vary due to wear. Eccentricity of pulleys may cause a sinusoidal disturbing torque and position error of the same frequency as the angular speed of the eccentric axle. Load variations contribute to the disturbing torques too. If the servo drive changes direction, non-linear effects due to Coulomb friction are encountered. This effect can fairly easy be compensated through applying a direction dependent feedforward voltage.

6.4 Servo amplifier

A high performance DC motor servo drive application requires four quadrant operation of the motor and the servo amplifier. Four quadrant operation means that the motor can drive and (actively) break in both directions. It requires that the servo amplifier can generate positive and negative voltages regardless of the current direction. The full-bridge DC-DC converter in figure 6.6 supports four quadrant operation.

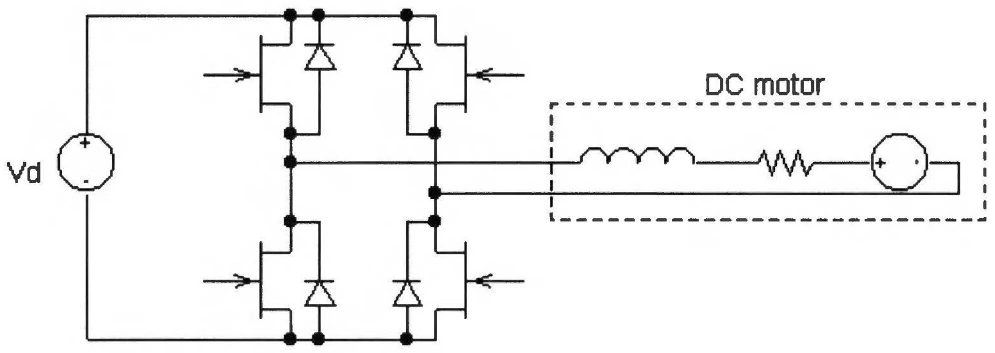


Figure 6.6 Full-bridge DC-DC converter

The output voltage of the full-bridge DC-DC converter is generated through pulse width modulation (PWM). The average output voltage of the full-bridge DC-DC converter depends on the duty ratio of the switching devices (it is assumed that the switching devices (typically MOSFETs) act like ideal switches). This manner of generating a variable voltage is much more efficient than a linear power supply where the semiconductor devices are operated in their linear operating region (where they act like controlled current sources). The full-bridge DC-DC converter in figure 6.6 can generate voltages between $-V_d$ and V_d.
Ideally, there is a linear relation between the control signal and the (average) output signal of a servo amplifier. If the switching devices in each leg of the full-bridge DC-DC converter are not simultaneously in the off state, the output voltage does only depend on the duty cycles of the switches in the two legs. If this condition does not hold, then the output voltage of the full-bridge DC-DC converter is dependent on the current sign which determines which of the anti-parallel diodes is conducting. This introduces an undesired non-linearity. More on these switching topologies can be found in [20].
Usually the switching frequency in the converter is well above the bandwidth of the DC motor, such that the motor reacts as if the average output voltage of the full-bridge DC-DC converter was applied. Typically, the switching frequency is chosen above 20 kHz, such that no induced torque ripple results in an audible noise. Although the switching character of the servo amplifier generally has no effect on the dynamics of the DC motor drive, the quantisation of the control signal to the servo amplifier can have an effect.

6.4 Optical encoder

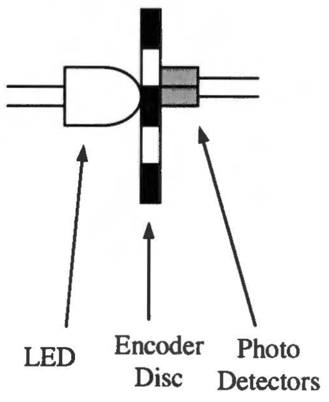


Figure 6.7 Principle of operation of position encoder

The principle of operation of an optical position encoder is depicted in figure 6.7. The light emitted by the light emitting diode (LED) is interrupted by the bars on the spinning encoder disc. The signals form the photo detectors are in quadrature (see figure 6.8). This allows determination of the direction. If the phase of channel *A* leads channel the phase of channel *B* then the direction is clockwise, if the channel *A* lags channel *B* then the direction is counter clockwise. Moreover, because the signals of the two channels are in quadrature, the resolution of the position measurement is enhanced to 4 times the number of encoder wheel slits per revolution.

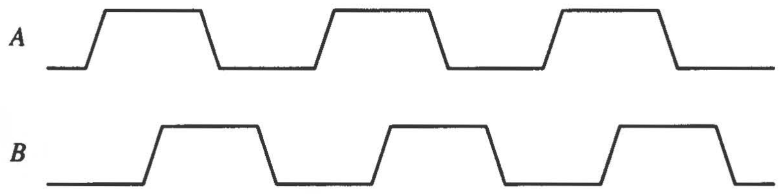


Figure 6.8 Quadrature signals

The pulses generated by the position encoder are counted on a microprocessor. The counter value represents the position of the encoder shaft. It is clear that in this way the position can only be represented by a number of discrete values. This introduces a measurement error. This quantisation error can be modelled as a white measurement noise with variance $q^2/12$, where q is the quantisation interval [1]. Another source of error is introduced when the encoder shaft or disc is eccentric. This causes a sinusoidal disturbance with a frequency equal to the angular speed of the encoder shaft.
Note that there is no possibility to include an anti-aliasing filter in the feedback path. The only way to include the filter would be to implement a kind of mechanical filter on encoder shaft, as the measured quantity has to be filtered before it is sampled. The sampling takes place when the counter value is read. The counter value is directly dependent on the encoder position. No filtering action is being applied.

7. Motor Controller Synthesis through Optimisation

7.1 Introduction

The frequency gains of chapter 4 and the integral quadratic criteria in chapter 5 can be used as criteria for controller optimisation. The definition of this optimisation problem requires the characterisation of:

- a parameterisation of the controller;
- a criterion for optimisation;
- and a series of constraints which the optimal solution has to satisfy.

The motor controller structure and parameterisation will be treated in section 7.3. A discussion on possible criteria and constraints is presented in section 7.4.
The ultimate goal is to find a controller which achieves a certain level of performance and robustness, and has minimal sampling rate. The performance generally degrades if the sampling rate is lowered. So finding that minimal sampling rate is essentially a multi-objective optimisation problem. However, if the robustness and performance goals are imposed as constraints, an optimal sampling rate can be searched for.
The sampling rate can be included in the parameter vector to facilitate minimisation of the sampling rate. This brings about some technical problems in the optimisation process, which are discussed in section 7.5. If the sampling rate is lowered, one has to reconsider the way in which setpoint profiles are generated, as this has very important design implications (see section 7.2). The choice on the tracking performance criterion is also affected by the way setpoint profiles are generated.
This chapter concludes with the derivation of a general design method for discrete-time motor controllers.

7.2 Reference generation

When expressing the tracking performance of a sampled-data control system, one is generally interested in how well some smooth, continuous-time reference signal can be tracked. However, as was pointed out in section 2.1, this reference signal is applied to the digital controller as a digital sequence. Somehow, the continuous-time reference signal has to be translated to a discrete-time signal. Two methods will be distinguished here, which will be called the *synchronous case* and the *asynchronous case*.
An often encountered reference signal in motion control systems is the second order position profile [22]. This kind of profile has a piecewise constant acceleration profile like in figure 7.1. The velocity profile is obtained by integrating the acceleration profile once, the second order position profile is obtained by integrating twice. If this integration is carried inside a digital device synchronised with the sampling rate, the block diagram ought to look like figure 7.2. The two discrete-time transfer functions are ZOH discretisations (step invariant discretisations) of $1/s$ and $1/s^2$, where h is the sampling time. Note that

$$\frac{h}{z-1}\cdot\frac{h}{z-1}\neq\frac{h^2(z+1)}{2(z-1)^2} \tag{7.1}$$

i.e. applying the forward Euler integration rule two times to the acceleration profile will not yield a position profile that is equal to the continuous-time profile on sampling times (see also [22]).

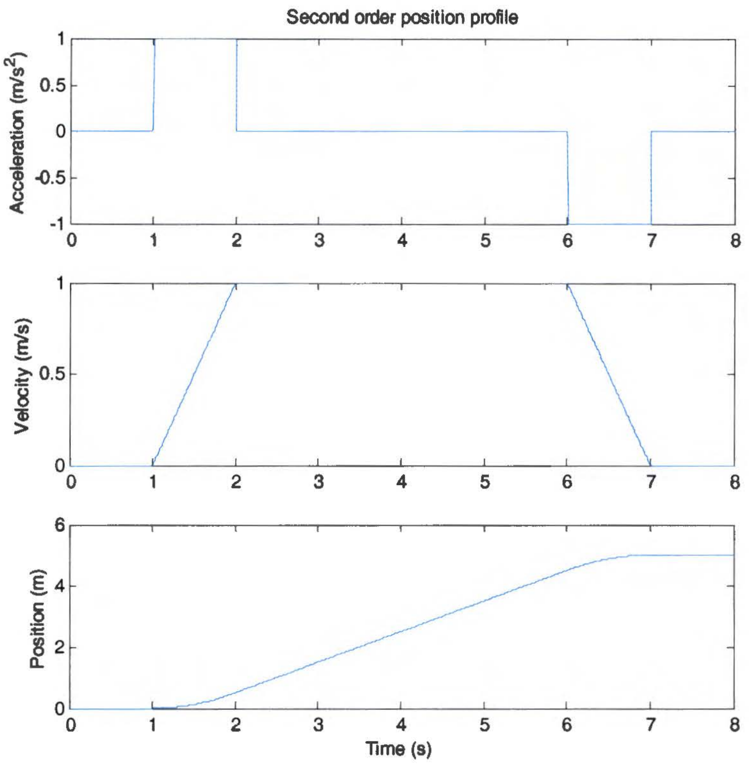


Figure 7.1 Second order position profile

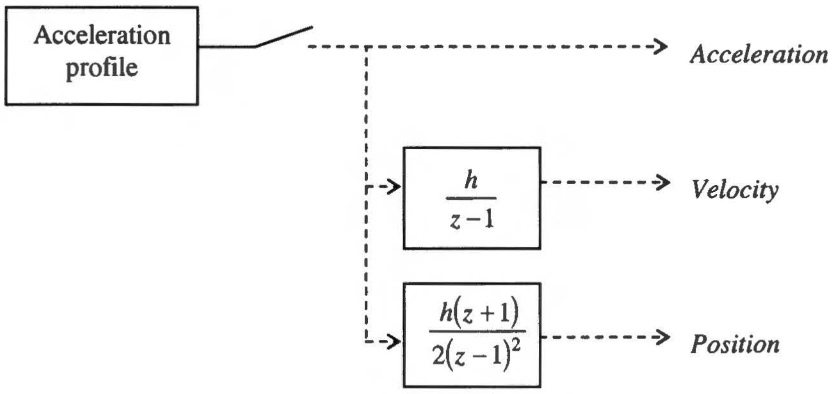


Figure 7.2 Integration block diagram

If the error due to the finite accuracy of the number representation inside the digital device is neglected, then the discrete-time profiles are equal to the continuous-time profiles on sampling times, if the acceleration is constant between two sampling instants. Let's call this case the *synchronous case*.

In the *asynchronous case*, the acceleration profile is still piecewise constant, but it is not synchronised to the sampling rate. It is easy to see that the block diagram in figure 7.2 is not applicable, by considering an extreme example where the acceleration is zero on every sampling time, but is nonzero between two sampling times. A correct way of calculating the reference position for time $t = kh$ is, to fill in the current time instant t in the equations of motion:

$$\begin{aligned} x(t) &= \tfrac{1}{2}a(t)t^2 + v(t)t + x_0 \\ v(t) &= a(t)t + v_0 \end{aligned} \quad (7.2)$$

This computation can be carried out off-line. The calculated profiles then have to be stored in the microcontroller memory. If the current time t is available in the digital controller subroutine, the calculation can be carried out online.
The synchronous implementation of profile generation through discrete-time integration may be favourable, because it is straightforward. Besides, if the sampling frequency is high (relative to the bandwidth of the profiles), the error between the profiles generated by the block diagram in figure 7.2 and an arbitrary desired continuous-time profiles is small. **Allowing the profiles to be *asynchronous* with the sampling rate is beneficial because it makes the design of the reference profiles and the choice on the sampling rate independent.** This is the preferred situation for paperpath design at Océ. It allows the independent design of sheet transport profiles and the digital controller.

7.3 Controller structure & parameterisation

In industry, the majority of motor controllers seems to be PID based. There is no need for a complex controller structure, as a well tuned PID-like controller achieves a sufficient level of performance. Velocity and acceleration feedforward is often added to the feedback controller in order to boost the tracking performance with respect to tracking second order position profiles [22]. Many engineers are familiar with this kind of control structure. It would be convincing to show that a well tuned controller with this structure can achieve a good level of performance while the sampling time is lowered. Ergo, the strategy is to optimise a fixed structure controller, with a low sampling rate, while maintaining a certain level of performance.
One of the design constraints is to achieve nominal stability. The stability of a controlled system does only depend on the feedback controller and is independent of the feedforward components (provided that these are stable themselves). By a clever parameterisation, one can assure that every feedback controller considered in the optimisation procedure stabilises a nominal plant.
Consider a second order model for a position controlled DC motor:

$$\frac{\theta_m(s)}{V(s)}=\frac{1}{k_E s(s\tau_m+1)}. \tag{7.3}$$

Let $G(z)$ be the ZOH transform of (7.3). It has the structure:

$$G(z)=\frac{k(z-a)}{(z-1)(z-b)}. \tag{7.4}$$

The discrete-time feedback controller should contain an integrator mode, in order to track a linearly increasing position reference signal (see section 3.4). For the moment, the remaining part of the controller is left free, resulting in the structure:

$$K(z)=\frac{R(z)}{S(z)(z-1)}. \tag{7.5}$$

The loopgain of the feedback loop is:

$$L(z)=G(z)K(z)=\frac{k(z-a)}{(z-1)^2(z-b)}\cdot\frac{R(z)}{S(z)}=\frac{P(z)}{Q(z)}\cdot\frac{R(z)}{S(z)}. \tag{7.6}$$

The characteristic equation of the closed-loop transfer is:

$$1+L(z)=0 \quad\Leftrightarrow\quad C(z)=Q(z)S(z)+P(z)R(z)=0. \tag{7.7}$$

The roots of this characteristic equation are the closed-loop poles. The equation:

$$C(z)=Q(z)S(z)+P(z)R(z) \tag{7.8}$$

is called a Diophantine equation. It has a unique solution in terms of $R(z)$ and $S(z)$ if

$$\deg(C(z))=\max(\deg(Q(z))+\deg(S(z)),\deg(P(z))+\deg(R(z))). \tag{7.9}$$

This follows from equating the coefficients of the powers of z for both sides of the Diophantine equation. For the case with the second order motor model and the controller structure in (7.5), it turns out that the Diophantine equation has a unique solution if deg($R(z)$) = 2 and deg($S(z)$) = 1. This controller structure corresponds to the biproper structure of a discretised PID controller with a roll-off pole to filter some high frequency components (see section 8.3)
Discrete-time controllers which stabilise nominal plant (7.3) are now parameterised by four stable closed loop poles, the roots of $C(z)$ (remember from section 3.3 that a discrete-time controller stabilises a continuous-time plant if the discrete-time controller stabilises the ZOH transform of the continuous-time plant). As the characteristic equation has four roots, three cases of possible pole configurations have to be considered. They are enumerated in table 7.1. The table shows how the characteristic equation can be constructed from four parameters $\theta_1,\ldots,\theta_4\in[-1,1]$, such that every possible stable closed-loop pole configuration is covered.
Once the characteristic equation $C(z)$ is known, the controller can be determined as a function of the parameters, through solving the Diophantine equation. This is done by equating the coefficients of the powers of z of both sides of the Diophantine equation. Let $R(z)=r_2z^2+r_1z+r_0$, $S(z)=s_1z+s_0$ and $C(z)=c_4z^4+\ldots+c_1z^1+c_0$, then equating the coefficients of the powers of z yields the set of linear equations:

$$\begin{aligned}
c_0 &= -kar_0-bs_0\\
c_1 &= kr_0-kar_1+(2b+1)s_0-bs_1\\
c_2 &= kr_1-kar_2-(b+2)s_0+(2b+1)s_1\;.\\
c_3 &= kr_2+s_0-(b+2)s_1\\
c_4 &= s_1
\end{aligned} \tag{7.10}$$

This set of five equations in five unknowns has a unique solution. The general solution to a Diophantine equation is the solution to matrix equation 7.11.

Table 7.1 Possible pole parameterisations

Pole configuration	Characteristic equation
two complex pole pairs ($\theta_1\geq 0$ and $\theta_3\geq 0$)	$(z-\lvert\theta_2\rvert e^{j\pi\theta_1})(z-\lvert\theta_2\rvert e^{-j\pi\theta_1})(z-\lvert\theta_4\rvert e^{j\pi\theta_3})(z-\lvert\theta_4\rvert e^{-j\pi\theta_3})$
one complex pole pair, two real poles ($\theta_1\geq 0$ and $\theta_3<0$, or $\theta_1<0$ and $\theta_3\geq 0$)	$(z-\lvert\theta_2\rvert e^{j\pi\theta_1})(z-\lvert\theta_2\rvert e^{-j\pi\theta_1})\cdot$ $(z+(1-2\theta_3)(1-\lvert\theta_4\rvert)+\lvert\theta_4\rvert)(z+(1-2\theta_3)(1-\lvert\theta_4\rvert)-\lvert\theta_4\rvert)$ or $(z+(1-2\theta_1)(1-\lvert\theta_2\rvert)+\lvert\theta_2\rvert)(z+(1-2\theta_1)(1-\lvert\theta_2\rvert)-\lvert\theta_2\rvert)\cdot$ $(z-\lvert\theta_4\rvert e^{j\pi\theta_3})(z-\lvert\theta_4\rvert e^{-j\pi\theta_3})$
four real poles ($\theta_1<0$ and $\theta_3<0$)	$(z+(1-2\theta_1)(1-\lvert\theta_2\rvert)+\lvert\theta_2\rvert)(z+(1-2\theta_1)(1-\lvert\theta_2\rvert)-\lvert\theta_2\rvert)\cdot$ $(z+(1-2\theta_3)(1-\lvert\theta_4\rvert)+\lvert\theta_4\rvert)(z+(1-2\theta_3)(1-\lvert\theta_4\rvert)-\lvert\theta_4\rvert)$

$$\left(\begin{array}{cccc|cccc} p_0 & 0 & \ddots & 0 & q_0 & 0 & \ddots & 0 \\ \vdots & p_0 & \ddots & 0 & \vdots & q_0 & \ddots & 0 \\ p_{np} & \vdots & \ddots & 0 & q_{nq} & \vdots & \ddots & 0 \\ 0 & p_{np} & \ddots & p_0 & 0 & q_{nq} & \ddots & q_0 \\ \vdots & 0 & \ddots & \vdots & \vdots & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & p_{np} & \vdots & \vdots & \ddots & q_{nq} \end{array}\right) \left(\begin{array}{c} r_0 \\ \vdots \\ r_{nr} \\ \hline s_0 \\ \vdots \\ s_{ns} \end{array}\right) = \left(\begin{array}{c} c_0 \\ c_1 \\ \vdots \\ \vdots \\ \vdots \\ c_{nc} \end{array}\right) \tag{7.11}$$

Here *np*, *nq*, *nr*, *ns* and *nc* respectively denote the degrees of the polynomials *P*(*z*), *Q*(*z*), *R*(*z*), *S*(*z*) and *C*(*z*), and the matrix entries denote the appropriate coefficients of the polynomials.
This result allows the parameterisation of stabilising controllers for arbitrary nominal plants (for example, a velocity controlled DC motor). If the Diophantine equation has no unique solution, extra controller poles can be added to ensure that (7.9) holds. These controller poles can be parameterised in a similar way as was done in table 7.1. The acceleration and velocity feedforward constants are also included in the parameter vector. **For the position controlled DC motor, this results in a parameter vector with 6 entries: 4 parameters characterising the feedback controller and 2 parameters for the velocity and acceleration feedforward constants.**
Note that the controller for the position controlled DC motor is biproper. In many academic examples this is not allowed, because in a practical implementation there is a computation delay between sampling and the output of a new control sample. This computation delay is neglected in the parameterisation, but it is taken into account in the computation of the frequency gains and the integral quadratic tracking criteria. It is possible to use the ZOH transform of the delayed plant in the parameterisation of the controller, but it would require different degrees for the polynomials *R*(*z*) and *S*(*z*), which would spoil the correspondence between the parameterised controller and the present PID-like controller.

7.4 Criteria & constraints

In the optimisation of motor controllers, both the frequency gains and the integral quadratic criteria will be used. The frequency gains have a clear interpretation, therefore they will be imposed as constraints. The integral quadratic tracking errors are suited criteria to quantify the performance with respect to tracking a second order position profile. However, it is usually impossible to give an upperbound on this integral quadratic criterion, because its numerical value lacks interpretation. The integral quadratic tracking error will be the optimisation criterion.
It is important to recognise that, in order to be able to execute a fair minimisation of the sampling rate, the constraints should be independent of the sampling rate! If a controller satisfies the constraints, a trade-off has to be made between a low sampling rate and a good tracking performance expressed as an integral quadratic tracking error. Which integral quadratic tracking error criterion has to be used depends on how the reference signals are generated. For a fair comparison between controllers of a different sampling rate, the asynchronous case should be considered. However, if one is willing to implement synchronous setpoint generation, the performance of a controller might be increased if the optimisation criterion is the synchronous one, instead of the asynchronous one (this fact is being motivated by example 12.4.2 in [3]).
In the remaining part of this section is illustrated how the control goals of chapter 3 can be incorporated in the optimisation criteria and constraints.

- **(Robust) stability**
  All the controllers inhibit some degree of nominal stability due to the parameterisation in section 7.3 (the computation delay is neglected in the parameterisation). One could use the approach of section 4.5 to ensure robust stability, but this will yield a too conservative design. A pragmatic approach is taken instead: the sampled-data control system is called robustly stable if the discrete-

time feedback controller stabilises every plant in a plant set. The plant set represents the plant uncertainty, and it should contain the 'worst-case' plant deviation. The phrase "worst-case" is quoted because in practice it may be hard to judge whether a set of physical plant parameters represents a worst-case (in what sense should a certain plant template be 'worst-case', anyway?). The problem of selecting plant templates which constitute the plant set is left to the expertise of the (control) engineer. It is expected that this pragmatic approach will yield controllers which show the desired degree of robustness.
If one really insists, one can constrain the gain and phase margin of the combination $P_d(z)K(z)$, where $P_d(z)$ is the ZOH transform of the plant including the computation delay. Lower bounds on the gain and phase margins are available as 'good practice' margins.

- **Tracking**
  Two types of tracking performances are distinguished:
  - the performance with respect to tracking arbitrary reference signals;
  - and the performance with respect to tracking fixed position profiles.

  An upper bound on the *PFG* of the transfer from reference signal to tracking error (PFG_{er}) can be used to enforce a certain level of tracking performance for the first class of reference signals, just as is done in continuous-time loopshaping. The PFG_{er} is computed for a grid of frequencies. The constraint for tracking the unknown reference signal is violated if for at least one plant template in the plant set, and at least one frequency in the grid, the PFG_{er} exceeds the upper bound.
  An integral quadratic tracking error is a suited criterion to express the performance with respect to tracking a certain position profile. In appendices A.5 and A.6 is shown how integral quadratic tracking error criteria are computed for motor controllers with acceleration and velocity feedforward, tracking a second order position profile. Both the synchronous case and the asynchronous case are considered. In the synchronous case, the acceleration profile corresponds to the impulse response of a discrete-time system. In the asynchronous case, the acceleration profile is approximated by an impulse response of a continuous-time rational transfer function.
  The interpretation of the generalised H_2 measure (see section 5.1) in the asynchronous case is the average integral quadratic tracking error. The average is taken over the sampling time, the period of the periodically time-varying sampled-data control system. Usually, one is actually interested in the worst-case tracking performance, i.e. the response to a setpoint profile with a time shift $\eta \in [0, h)$ for which the integral quadratic tracking error is worst. This worst-case criterion lacks the computational benefits which the average integral quadratic tracking error possesses.
  In section 5.2 it was already pointed out that the controller design achieving the lowest possible integral quadratic tracking error might be a too optimistic design, as disturbances and model errors are not taken into account. Some level of robust performance is pursued through optimising the worst-case tracking performance with respect to the plant templates in the plant set. **It is advised to include some frequency domain constraint even if the performance with respect to unknown reference signals is not of interest. In this way, resonances and an excessive bandwidth (poor rejection of high frequency measurement noise) which might appear in the optimal solution, are restricted.**

- **Disturbance rejection**
  The *PFG* from disturbance input to plant output (PFG_{yd}) is the frequency gain of interest when disturbance rejection is considered. Note, that PFG_{yd} and PFG_{er} differ a factor $P(j\omega)$ (compare (3.55) and (3.56)). This means that a constraint on PFG_{yd} can be translated to a constraint on PFG_{er}. So, the frequency gain PFG_{er} is used to express both tracking constraints and disturbance rejection constraints.
  One might be tempted to impose an upper bound on PFG_{er} such that the controller has a broad tracking band and excellent disturbance rejection properties. A controller optimisation problem with these constraints will probably turn out to be unfeasible, because the Bode sensitivity integral cannot be satisfied within these tight constraints. **Care has to be taken, that enough room is left for the Bode sensitivity integral constraint.**

- **Sensor noise avoidance**
  In order to decrease the effect of measurement noise, the controller bandwidth has to be decreased. It is possible to achieve this by constraining FFG_{yn}, DFG_{yn} or PFG_{yn}, but if one of these frequency gains is constrained in conjunction with PFG_{er}, there is a fair chance that the fundamental property

$$S_0(j\omega)+T_0(j\omega)=I \tag{7.12}$$

  is violated. The fundamental sensitivity function is given a high pass character by the upper bound on PFG_{er}. (7.12) generally ensures the low pass character of the fundamental complementary sensitivity function and it implicitly restrict the bandwidth of the controller.
  It can be shown that PFG_{yn} is a periodic function of the frequency, DFG_{yn} is a lower bound on PFG_{yn} and that $DFG_{yn} \approx PFG_{yn}$ for systems with a low fidelity index. So if one decides to put a low pass upper bound on DFG_{yn} or PFG_{yn}, the upper bound should be defined on the fundamental frequency range $\left[0,\frac{\omega_s}{2}\right)$ only, and not on the whole frequency axis, otherwise the optimisation problem will definitely be unfeasible!

- **Avoidance of actuator saturation**
  A sensible way of avoiding actuator saturation as effect of a reference signal is to constrain FFG_{ur}. FFG_{ur} is the ratio of the amplitude of the fundamental component in the actuator signal, to the amplitude of a sinusoidal reference signal. Generally, the bandwidth of the reference signal is low compared with the sampling frequency. This roughly means that the contribution of aliasing components to the actuator signal is low, which makes the *FFG* the appropriate frequency gain. However, the *FFG* describes only the amplitude ratios. If the reference signal is decomposed as a series of sinewaves, the *FFG* describes the amplitude gain of each of the frequency components, but it does not contain any phase information. The chance exists that the phases of the sinewaves are such that, in some point in time, all the sinewaves are at its maximum, resulting in a large peak in the actuator signal. This observation makes it hard to define an upper bound on *FFG* (and for the same reason, the other frequency gains) a priori. The constraint accounting for the avoidance of actuator saturation can be adjusted iteratively, after simulation or measurements have shown that actuator saturation occurs. Of course, one could find out if actuator saturation occurs, through running a simulation for every set of controller parameters occuring in the optimisation problem, but this will probably be too time consuming.
  If one decides to use an upper bound on FFG_{ur}, one has to make sure that fundamental limitation (7.12) can hold. Actuator saturation due to the feedforward voltage can be prevented through defining upper and lower bounds on the acceleration and velocity feedforward constants.

7.5 Optimisation algorithm

Ideally, a controller optimisation procedure would yield a single controller which has minimal sampling time and exhibits a certain level of performance. However, the performance level generally decreases if the sampling rate is lowered, making the optimisation problem multi-objective. There is no single optimal solution to a multi-objective optimisation problem, as every solution is necessarily a compromise between two or more criteria. One can try to find a set of solutions for which an improvement on one of the criteria implies a deterioration of the others. The complete set of these solutions is called the Pareto set or the Pareto front. For the optimisation problem considered here, the Pareto front may look like figure 7.3. Note that the Pareto front can be constructed through computing an optimal controller for every single sampling rate.
A different way of generating the Pareto front is through the use of a genetic algorithm (GA). A genetic algorithm is an optimisation algorithm which is inspired on the natural concept of evolution and the principle of 'survival of the fittest'. The genetic algorithm works with a population of individuals, each representing a controller. The individuals are assigned a fitness value which depends on the performance of the controller. The fittest individuals are selected for reproduction.

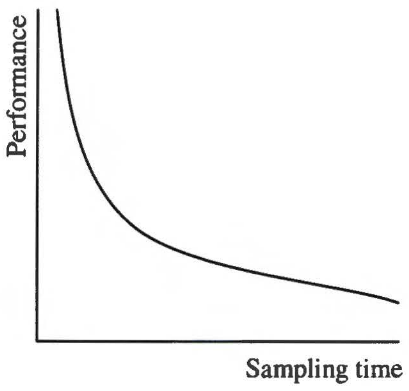


Figure 7.3 Pareto front

Reproduction occurs through applying a crossover on genes at a random position (see figure 7.4). A gene is a bit string representing an individual. The bit string is a binary representation of the parameter vector. Once every while, a mutation occurs where one of the bits changes value (see figure 7.4). Through breeding the fittest individuals, it is tried to achieve fitness improvement over subsequent generations.

Crossover

11010101|00110 → 11010101|10101
11000011|10101   11000011|00110

Mutation

1101010100110 → 1101000100110

Figure 7.4 Crossover & mutation

In order to approximate the Pareto front, diversity with respect to the sampling rate has to be maintained. This is done through a technique called fitness sharing [23]: if some individuals' sampling rates are close, their fitness is scaled down with an amount proportional to the amount of individuals with a sampling rate that is close.
The genetic algorithm is adapted to deal with constraints expressing robust stability and robust performance. The population is divided into three classes; a class of nominal stable controllers (NS), a class of robustly stable controllers (RS) and a class of controllers which achieve robust performance (and are robustly stable) (RP). Every controller is nominally stable because of the parameterisation presented in section 7.3. Once a controller stabilises every plant in the plant set, it is promoted to the RS class. The robustly stable controllers which satisfy a performance constraint (typically an upper bound on PFG_{er}) for every plant template in the plant set, are part of the RP class.
The population is ranked. Every individual in the RP class has a higher rank than an individual in the RS class, which in turn has a higher rank than every individual in the NS class. The NS class is sorted according to the maximum modulus of the eigenvalues, the individual with the lowest modulus gets the highest rank. The RS class is sorted according to the performance constraint violation, the individual with the lowest constraint violation earns highest rank. The RP class is sorted according to a performance criterion (typically, an integral quadratic tracking error criterion). The individuals are assigned a fitness value according to a ranking function with a parameter called the selective pressure [24]. The selective pressure determines the ratio of the probability that the best individual is chosen to reproduce, to the probability that an average individual is selected. It sets the balance between exploration and exploitation of the search space.
Although the introduction of the class system seems a sensible way to introduce some constraints, it turns out to be unsuited for the determination of the Pareto front. Experiments have pointed out that (depending on the constraints) only a couple of individuals reach the RP class, and they are likely to take over the whole population. The selective pressure can be reduced in order to try to overcome this problem, but then elitism has to be introduced to save the fittest individual of each generation, as this

individual has a fair change not to be selected for reproduction. Unfortunately, some experiments have pointed out that this adjustment isn't very fruitful either.
The problem seems to be the formation of lethals, unfit offspring. As the motor is open-loop stable, it is easy to find a stabilising controller with a low sampling rate, because in between sampling the system is essentially in open-loop. It turns out that almost any controller parameterised as in section 7.3 and having a low sampling rate, is robustly stabilising. For higher sampling frequencies it seems harder to find robustly stabilising controllers. The opposite is true for robust performance. Generally, controllers with a low sampling rate show a lower level of performance. So, if an individual representing a controller with a low sampling rate is crossbreeded with another individual, then the offspring is unlikely to exhibit robust performance. On the other hand, if a controller with a high sampling rate is selected for reproduction, there is a fair chance that the offspring won't be robustly stabilising. Once an individual reaches the RP class, and crossing this individual produces lethals, inbreeding occurs: a small group of individuals takes over the whole population. This problem might be overcome through the introduction of a mating selection scheme, or some other advanced genetic operations from [23] and [27].

Instead of generating the whole Pareto front at once, a different approach is being applied in which a controller is optimised for a fixed sampling rate. The approach is a state-of-the-art non-linear programming technique called sequential quadratic programming (SQP). Although the genetic algorithm can be used for optimising a controller with a fixed sampling rate, the sequential quadratic programming technique generally beats the genetic algorithm on convergence speed and consistency.
The sequential quadratic programming technique is based on analytical results rather than an analogue to a natural phenomena. The main idea behind sequential quadratic programming algorithm is explained in the steps below.

1. The non-linear optimisation problem with non-linear constraints is being transformed to a simpler well-posed problem (using the concept of Langrange multipliers).
2. This simpler well-posed problem is implicitly assumed to be a quadratic problem. A search direction is obtained through a quadratic programming subproblem. This subproblem is being derived from the quadratic approximation of the well posed simpler problem of the first step.
3. A one-dimensional line-search is executed, finding the maximum improvement in the search direction, respecting the constraints.
4. The quadratic approximation of the simpler well-posed problem is updated using curvature information obtained in the previous steps. The next iteration proceeds with the direction finding subproblem in step 2.

The quadratic subproblem is typical for sequential quadratic programming. The solution to the quadratic direction finding subproblem gives an optimal search direction. Moreover, an accurate initial guess for the one-dimensional search can be given, because the problem was assumed to be quadratic.
An introduction to sequential quadratic programming and other numerical optimisation techniques can be found in [28]. The function `fmincon` in MATLABs Optimization Toolbox offers an implementation of sequential quadratic programming. Implementation details can be found in [29].

7.6 A general motor controller design method

In this section a general motor controller design method is presented. It is an optimisation process which includes an intuitive frequency gain as a performance constraint, and a trade-off between the sampling rate and a time-domain integral quadratic tracking error criterion. Robustness is enforced by using a set of plant templates.
The design methods comprises the steps:

1. Define a nominal plant which is being used to parameterise nominal stabilising controllers, according to section 7.3.
2. Define a set of plant templates. They should represent all possible plant perturbations and model uncertainties.

3. Define a grid of frequencies which is dense enough to capture all the important behaviour in the frequency region of interest.
4. Define a frequency domain performance constraint, which is independent of the sampling frequency. Usually it suffices to use an upper bound on PFG_{er} for every frequency in the grid.
5. Introduce an auxiliary performance criterion which is traded off with the sampling frequency. This can be the average integral quadratic tracking error criterion (asynchronous case), if a fair comparison for different sampling rates is desired. The tracking error is computed with respect to tracking some setpoint profile.
6. Choose an initial sampling frequency.
7. Optimise a controller with the current sampling rate, such that:
   - the controller stabilises every plant in the plant set;
   - the controller satisfies the frequency domain performance constraint for every plant template in the plant set, and for every frequency in the frequency grid;
   - the auxiliary performance criterion is optimal for the current sampling rate and the worst-case plant template.
8. Evaluate the controller. This can be done through computing some frequency gains, simulation, or any other method for controller validation.
9. If the performance is satisfactionary, repeat from step 7 with a lower sampling rate. Otherwise, choose a higher sampling rate and proceed with step 7.

This is the outline of the design procedure. Some iterative adjustment of the frequency domain performance constraint may be necessary if the optimisation problem turns out to be infeasible.
For convergence to the minimal sampling rate, it is necessary that the constrained optimal auxiliary performance criterion is a monotonously decreasing function of the sampling rate. The existence of an optimal sampling rate in a context of LQG control was studied in [18]. It was shown that an LQG-like cost function augmented with a term representing the cost of sampling is not a convex function of the sampling frequency if *pathological sampling* (the definition is given in [3]) occurs. This is for example the case, if a sinewave of 1 Hz is being sampled with a sampling rate of 1 Hz. The sampled signal is then a series of samples with an equal magnitude. The oscillatory mode becomes unobservable. Although a different problem than in [18] is being considered here, it is not expected that the design procedure leads to optimal sampling rates. It is hard to believe that the imposed constraints make the optimisation problem any easier.
In spite of searching for the optimal sampling rate, the design procedure can be used to optimise a couple of controllers for different sampling rates, and then select the controller which achieves a sufficient level of performance and has the lowest sampling rate. Note however, that the design procedure gives a structured approach to the search for a suited sampling rate, as the constraints and criteria are independent of the sampling rate, and they take the continuous-time behaviour into account.
A similar approach was suggested in [30], but the authors use different constraints and the *PFG* is being approximated with the *DFG* in the optimisation. Furthermore, they don't use any time-domain criteria, like an integral quadratic tracking error criterion. The authors mention about the minimisation of a cost function involving the *PFG* as a performace criterion: "Note that the minimization is non-convex and ill-conditioned..."

8. A Motor Control Case

8.1 The case

The case regards the control of a motor in the paperpath of an Océ copier/printer. The motor being considered is the last one in the paperpath, before the sheet enters the transfuse where the toner image is transferred onto the sheet. A high degree of registration accuracy is required. This imposes some tight control requirements on the motor controller.

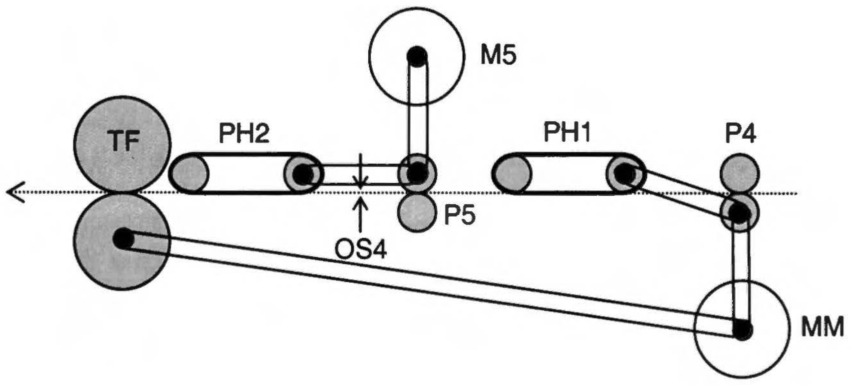


Figure 8.1 Control case

The relevant part of the paperpath is depicted in the figure above. The motor to be controlled is motor 5 (M5). It drives the last pinch (P5) before the transfuse (TF) and preheater belt 2 (PH2) which warms up the paper before entering the transfuse. When the leading edge of an A4 sheet triggers the optical sensor (OS4), the trailing edge is still in the first preheater (PH1) which is driven at a constant speed by the main motor (MM). The image at the transfuse belt is travelling at an approximately constant speed.
Nominally, a sheet triggering the optical sensor is too late with respect to image position on the transfuse. That is, a small distance has to be catched up while the sheet is in pinch 5 and the second preheater. However, when a sheet enters pinch 5, the trailing edge of the previous sheet is still in the second preheater. While the previous sheet is still in the preheater, motor 5 is not allowed to accelerate. In fact, motor 5 is then controlled by an open loop velocity controller, allowing the previous sheet to pass the transfuse smoothly. The transfuse runs at a slightly higher speed than PH2. The open loop control limits the pulling forces on the sheet.
After the previous sheet has left the second preheater (roughly 200 ms after OS4 is triggered), a correction profile is applied to a position controlled motor 5, in order to catch up with the image on the transfuse. The necessary correction profile is calculated from the time a sheet triggers OS4 and the position of the image on the transfuse. The worst-case correction profile is depicted in figure 8.2. The maximal sheet velocity is 488 mm/s, which is 20% higher than the nominal process speed at 403 mm/s. The duration of the correction profile is 200 ms at maximum, which corresponds to 80 mm of sheet transport. These bounds on velocities and profile duration have been chosen such that a sheet stays long enough in the preheater. The maximal acceleration and deceleration are 5 m/s^2 and 15 m/s^2 respectively. These bounds are related to the power budget, mechanical forces and the lifetime of drive components. The maximum position correction is equal to the shaded area in figure 8.2.
15 mm of paper transport ($\approx$ 30 ms) has been reserved as a settling time after the downward slope of the correction profile. After 30 ms the position error has to be within $\pm$ 50 μm, which is the error budget for the controller. During this last 30 ms before the sheet enters the transfuse, the controller has to track the toner image position variations. This image position signal is derived from the velocity of the transfuse (which is approximately constant). The bandwidth of the image position variations have been estimated to be 30 Hz. The significant components lie in the range from 0.5 to 5

Hz and have an amplitude of maximal 300 μm. These signals have to be tracked with an accuracy of 50 μm.

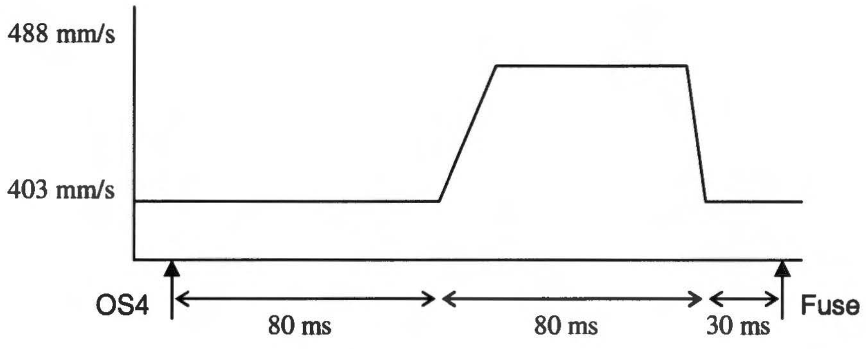


Figure 8.2 Worst case correction profile

Besides tracking reference signals, the controller should suppress disturbances. The main source of disturbing torques is identified as the eccentricity of the main pulley (see figure 8.3). The eccentricity introduces a sinusoidal disturbing torque with an estimated frequency of 4.5 Hz and an amplitude of $11 \cdot 10^{-4}$ Nm. The eccentricity also introduces a sinusoidal position error for pinch 5. This error cannot be reduced by a feedback controller, as the position encoder disc is mounted on the motor shaft. Therefore, this error is not subtracted from the error budget of the controller. Other sources of disturbances are the interaction of the paper and the stick/slip of the preheater belt. These disturbances are assumed to be of minor effect because of the relatively high transmission ratio (1:5) from the pinch to the motor shaft.

The control requirements can be summarised as:

- After tracking the worst-case correction profile, the tracking error should be settled in a band of ± 50 μm within 30 ms.
- The controller should be able to track components of amplitude 300 μm in the range 0.5 to 5 Hz with an accuracy of 50 μm.
- The controller has to suppress a sinusoidal disturbing torque with amplitude $11 \cdot 10^{-4}$ Nm and frequency 4.5 Hz.

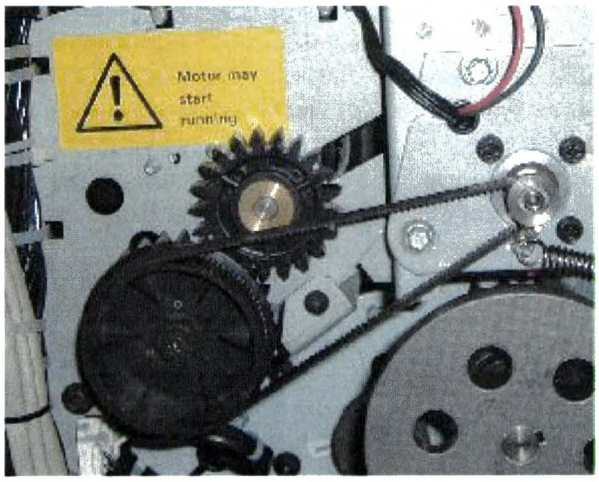


Figure 8.3 Motor 5 driving the main pulley

8.2 Simulation model

A model for the motor 5 drive application that was already available at Océ has been adapted for the purpose of controller simulation. The Simulink (MATLABs simulation tool) model is shown in figure 8.4. The block modelling the motor and load is depicted in figure 8.5.

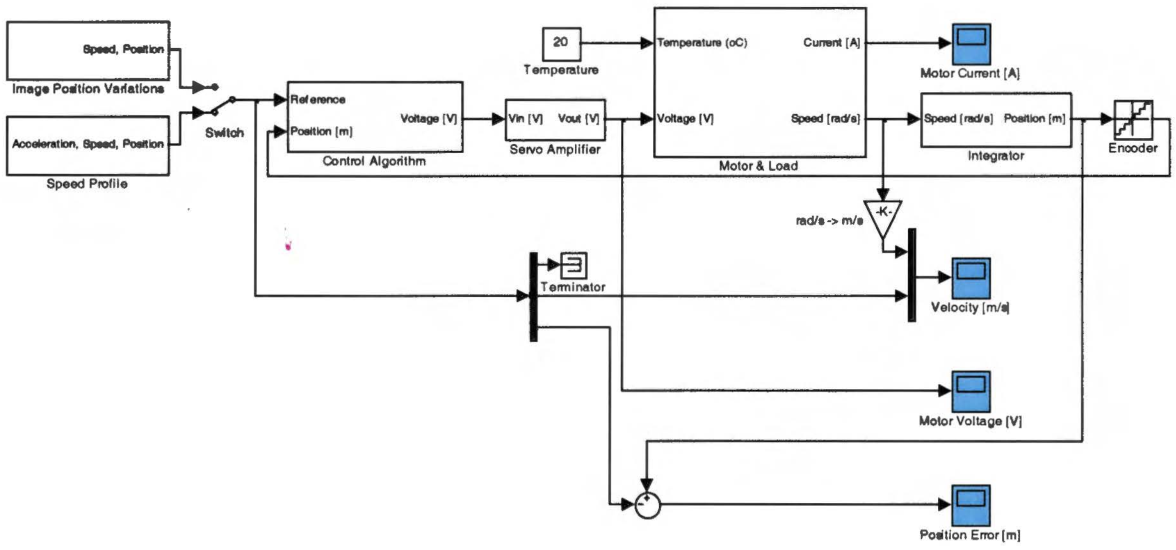


Figure 8.4 Simulink model

The model accounts for:
- third order dynamics of the motor plus load (electrical pole is modelled);
- temperature changes;
- a disturbance torque due to eccentricity of the main pulley;
- quantisation of the position measurement due to the optical encoder disc;
- quantisation and saturation of the actuator voltage;
- and a computation delay (in control algorithm).

Some phenomena which are not modelled are: the switching behaviour of the servo amplifier and the interaction of other pinches when the sheet is at two pinches at once. Two signals are available for simulation: the worst-case position profile and a signal representing the image position variations. This last signal is generated through filtering a random signal with a 4th order Butterworth filter with cut-off frequency 5 Hz. The position signals are in meters paper transport and the actuator signal is in Volts. The model parameters are enumerated in table 8.1. The parameter dependence on temperature is obtained through linear interpolation between the parameters for cold and hot conditions. The transmission ratio defines the ratio between the paper displacement in meters and the angular movement of the motor shaft in radians

Table 8.1 Model parameters

Temperature (T)	20	75	°C
Damping coefficient (B)	$3.617\cdot10^{-5}$	$4.02\cdot10^{-5}$	Nm·s/rad
Inertia (J)	$2.21\cdot10^{-5}$	$2.21\cdot10^{-5}$	kg·m^2
Motor constant (k_E or k_T)	0.046	0.0413	V·s/rad or Nm/A
Resistance (R)	1.1	1.2	Ω
Inductance (L)	$65\cdot10^{-5}$	$65\cdot10^{-5}$	H
Computation delay (λ)	$150\cdot10^{-6}$	$150\cdot10^{-6}$	s
Transmission ratio (i)	0.00285	0.00285	m/rad

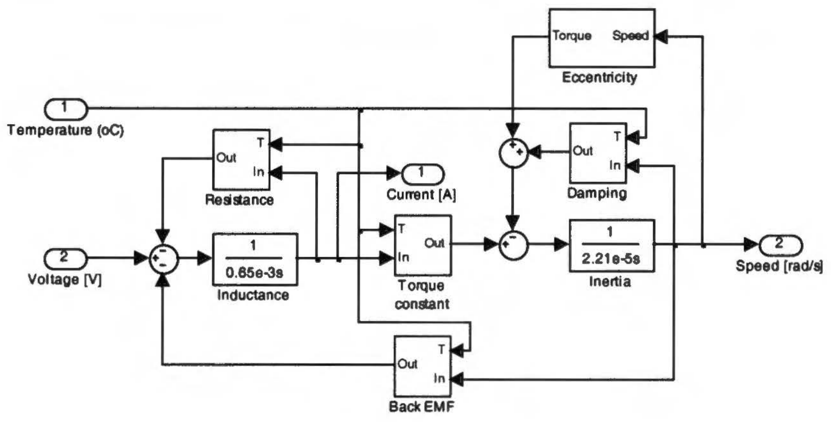


Figure 8.5 Simulink model for motor plus load

8.3 Evaluation of the present controller using the simulation model

The present position controller for motor 5 is a discrete-time equivalent of a PID controller. Its control action is described by the expression:

$$u[k] = K_P e[k] + K_I h \sum_{\ell=0}^{k} e[\ell] + K_D \frac{1}{h}\left(e[k] - e[k-1]\right) \tag{8.1}$$

where u is the actuator signal, e the deviation of the controlled value from the setpoint and h the sampling time. It corresponds to the z-domain transfer:

$$\frac{U(z)}{E(z)} = K_P + K_I \frac{hz}{z-1} + K_D \frac{z-1}{hz} \tag{8.2}$$

which is the backward Euler transform (see section 2.2) of the continuous-time PID controller transfer:

$$\frac{U(s)}{E(s)} = K_P + \frac{K_I}{s} + K_D s\,. \tag{8.3}$$

The output of the discrete-time PID controller is smoothed according to:

$$u_{filtered}[k] = \frac{a}{a+1} u_{filtered}[k-1] + \frac{1}{a+1} u[k]\,. \tag{8.4}$$

This filtering action has to prevent the overamplification of high frequency (measurement) noise due to the derivative action. a is selected to be $e^{-1/4}$, yielding a pole at a quarter of the sampling frequency, which for this PID controller is 1 kHz. The PID controller parameters are given in the table below. Again, the position quantity is the sheet displacement in meters.

Table 8.2 PID controller parameters

Parameter	Value	Unit
Sampling frequency (f_s)	1000	Hz
K_P	200	V/m
K_I	14000	V/s/m
K_D	1.2	V·s/m
Filter pole	250	Hz

Using the information form tables 8.1 and 8.2, the nominal plant model (at 20 °C) and the discrete-time controller transfer function are obtained.

$$P(s)=\frac{9126.3488}{s(s+1600)(s+93.79)}e^{-150\cdot 10^{-6}\cdot s} \tag{8.5}$$

$$K(z)=\frac{52224.9994\left(z^2-1.839z+0.8487\right)}{(z-1)(z-0.4378)} \tag{8.6}$$

P defines the transfer from motor voltage to sheet displacement (in meters), *K* defines the transfer from the position error in meters to the motor voltage.
It is instructive to study some frequency gains of the controller and plant combination before simulation. Figure 8.6 shows the *FFG*, *DFG* and *PFG* for the transfers from the reference input to the plant output (8.6a), the reference input to the tracking error (8.6b), the disturbance input to the plant output (8.6c) and the measurement noise to the plant output (8.6d). Figure 8.6b is used to study the asymptotic tracking properties. The PFG_{er} is approximately –37 dB at 5 Hz, which corresponds to a reduction of a factor 70 which is much higher than the required reduction of a factor 6 (300 µm image position variations have to be tracked with an accuracy of 50 µm). Figure 8.6c shows that the system is rather insensitive to high frequency torque disturbances. Note also, that the effect of the reference signal and the measurement noise on the output is the same, due to the absence of an anti-aliasing filter.

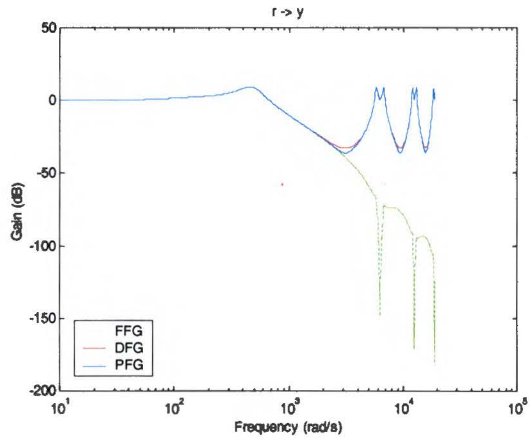


Figure 8.6a FFG_{yr}, DFG_{yr} *and* PFG_{yr}

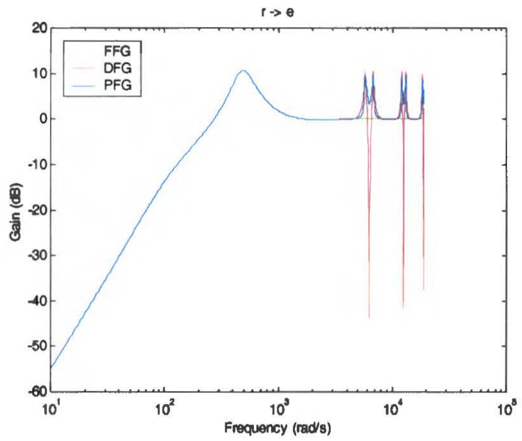


Figure 8.6b FFG_{er}, DFG_{er} *and* PFG_{er}

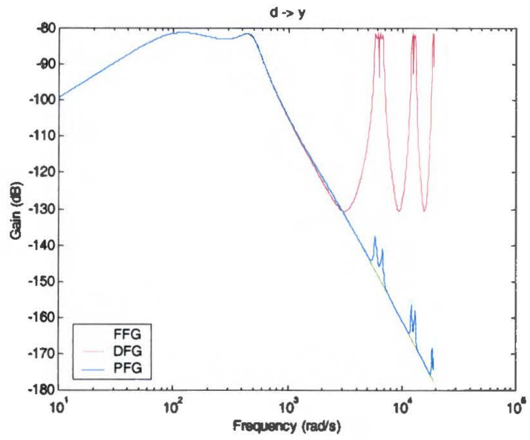


Figure 8.6c FFG_{yd}, DFG_{yd} *and* PFG_{yd}

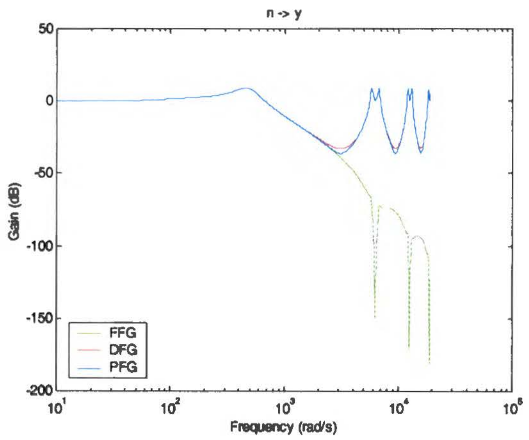


Figure 8.6d FFG_{yn}, DFG_{yn} *and* PFG_{yn}

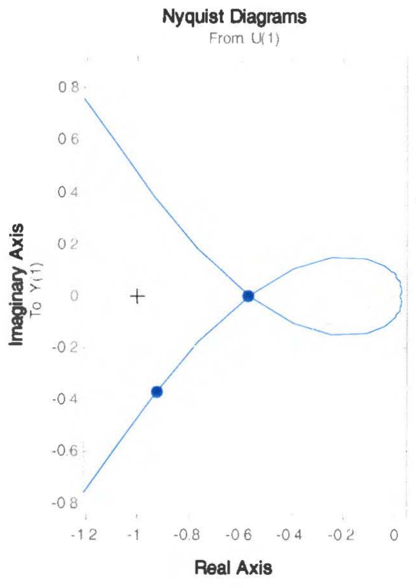


Figure 8.7 Nyquist diagram for PID controller

Figure 8.7 shows a part of the Nyquist diagram for the discrete-time PID controller. The controller has a phase margin of 21.8° and a gain margin of 4.85 dB.
For the simulation of the control system, values for the velocity an acceleration feedforward constants are needed. In the copier, they are obtained through some kind of calibration procedure. For the simulation, they are obtained from the inverse of a simple second order plant model:

$$P(s)=\frac{i}{k_E s(s\tau_m+1)} \quad \Rightarrow \quad P^{-1}(s)=\frac{k_E\tau_m}{i}s^2+\frac{k_E}{i}s\,. \tag{8.7}$$

Let x be the position of a sheet in the paperpath, and let V be the applied motor voltage. The action of the inverse plant can be expressed in time-domain as:

$$V(t)=\frac{k_E\tau_m}{i}\cdot\frac{d^2x(t)}{dt^2}+\frac{k_E}{i}\cdot\frac{dx(t)}{dt}=\frac{k_E\tau_m}{i}\cdot a(t)+\frac{k_E}{i}\cdot v(t) \tag{8.8}$$

where v is the sheet velocity and a is the sheet acceleration. By choosing the acceleration and velocity feedforward constant as:

$$K_a=\frac{k_E\tau_m}{i}=\frac{k_E J}{i\left(\frac{k_E k_T}{R}+B\right)}\approx 0.18 \tag{8.9}$$

$$K_v=\frac{k_E}{i}\approx 16 \tag{8.10}$$

the action of the inverse plant is being mimicked.
Four simulations were executed: the response to the image position signal for the hot and the cold motor, and the response to the worst-case correction profile (the profile achieving the maximal sheet position correction). The tracking error for the image position variation signal is depicted in figure 8.8. The simulation was carried out for the cold motor. The tracking error for the hot situation does not differ very much. Clearly the present PID-like controller is being able to track the image position signal with the desired accuracy, even in the presence of the disturbing torque due to the eccentricity of the main pulley.

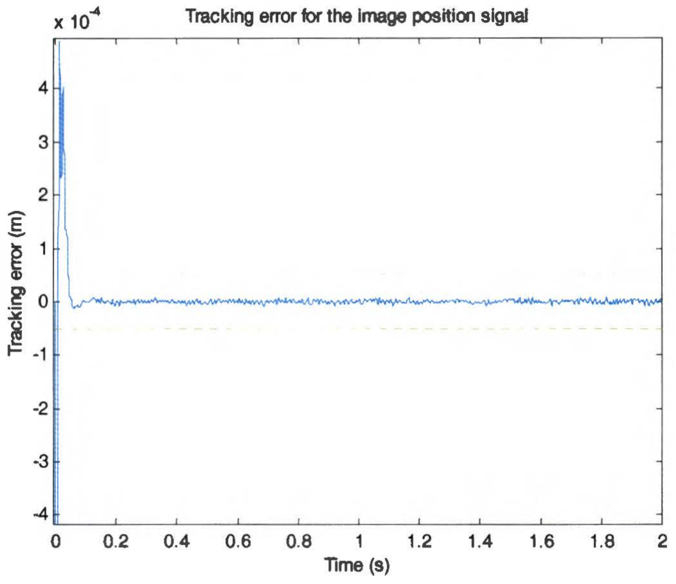


Figure 8.8 Tracking error of the PID controller for the image position signal

Figures 8.9 and 8.10 show the tracking error resulting from applying the worst-case correction profile and the sheet velocity responses. The simulation results for both the cold (blue) and the hot (red) motor are plotted. The actuator signal for the cold case is reflected in figure 8.11. The tracking error has to be within the range ± 50 μm after 30 ms of the downward slope of the velocity profile (which is at time t = 1.43 s for the simulation). The bounds are depicted through the green dotted lines in figure 8.9. The present discrete-time PID controller with a sampling frequency of 1 kHz satisfies these demands even with the occurrence of the disturbing eccentricity torque.

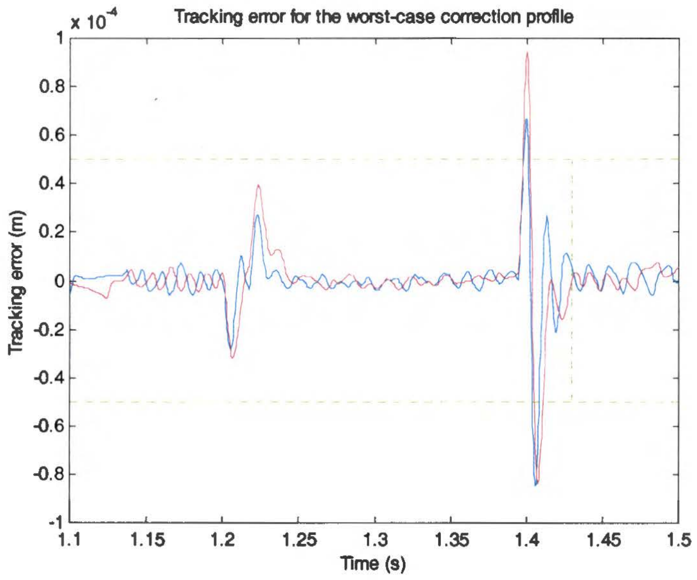


Figure 8.9 Tracking error for worst-case correction profile

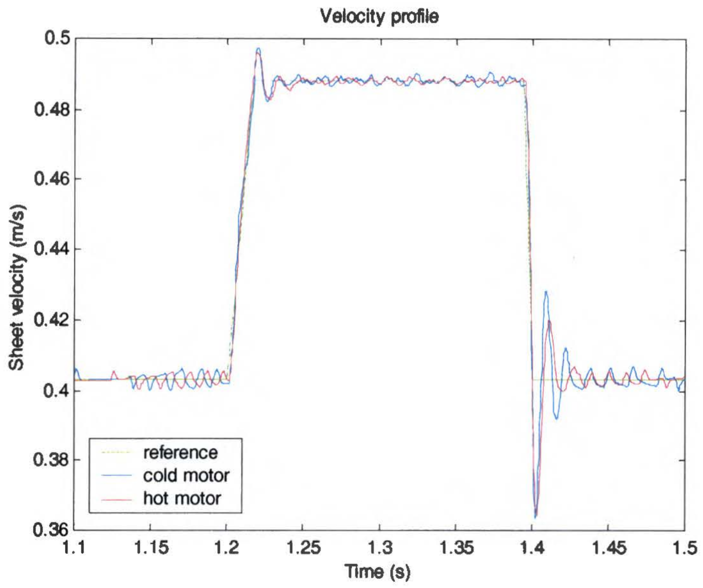


Figure 8.10 Sheet velocity response

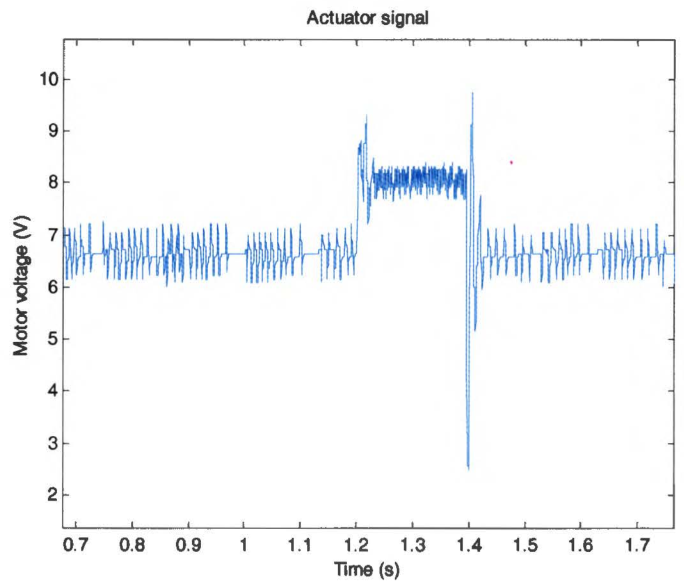


Figure 8.11 Actuator signal

8.4 Controller design for the simulation model

A new controller with a lower sampling rate is being designed for the simulation model. This is done using the method suggested in section 7.6.
First, a set of plant templates is defined for which the controller should demonstrate some level of performance and which it should stabilise. The nominal model, a second order plant model used to parameterise stabilising controllers, is included in the plant set. Two other plant templates are obtained from a plant model at 0 °C and a plant model at 100 °C. Although the thermal operating range for the motor is assumed to be 20 °C to 75 °C, the extra temperature deviations are used to enforce extra robustness against fabrication tolerances for example.
An upper bound on the PFG_{er} is being used as a frequency domain constraint in the optimisation algorithm. The shape is being determined by control demands:

- the PFG_{er} should be below –15 dB (a factor 6) for frequencies below 5 Hz, in order to get a sufficient tracking performance with respect to tracking image position variations and a sound amount of disturbance suppression at 4.5 Hz;

- it should be below 6 dB in the range from 5 to 30 Hz in order not to over-amplify some minor components in the image position signal.

Furthermore, it is desirable to give an upper bound on PFG_{er} for the frequencies above 30 Hz, because it prevents the possible occurrence of resonant peaks originating from the H_2 optimisation (remember the discussion in section 7.4). Because the PFG_{er} for the discrete-time PID controller has a maximum of 10 dB, this upper bound for higher frequencies is set to 10 dB.

As a fair comparison of controllers with a different sampling rate is desired, the worst-case correction profile is kept the same for every sampling frequency. This means that one has to consider the asynchronous case of reference generation and the average integral quadratic tracking error as optimisation criterion.

The worst-case average integral quadratic tracking error is being optimised under the constraints that the controller should stabilise every plant in the plant set, and that the upper bound on PFG_{er} is not violated for any plant template. "Worst-case" is with respect to the plant templates in the plant set and the average is taken over the sampling time h.

The tracking error is optimised with respect to tracking the downward slope of the worst-case correction velocity profile. The focus is on the downward slope, because the only constraint on correction profile tracking is on the settling time after the deceleration of the sheet. The piecewise constant deceleration of 15 m/s^2 during 6 ms is approximated with a fourth order Padé approximation, in order to compute the average integral quadratic tracking error. The approximation of the acceleration, velocity and position profiles is depicted in figure 8.12. The approximation error is negligible in the position profile, as the error is 'smeared out' due to the double integration of the acceleration profile. The acceleration and velocity information is being used to optimise the feedforward constants. Acceleration and velocity feedforward has the potential of improving the servo performance over feedback control alone, even if the acceleration and velocity feedforward constants aren't that accurate (compared to the second order inverse plant). For this reason, the approximation error in the acceleration and velocity profiles is admissible.

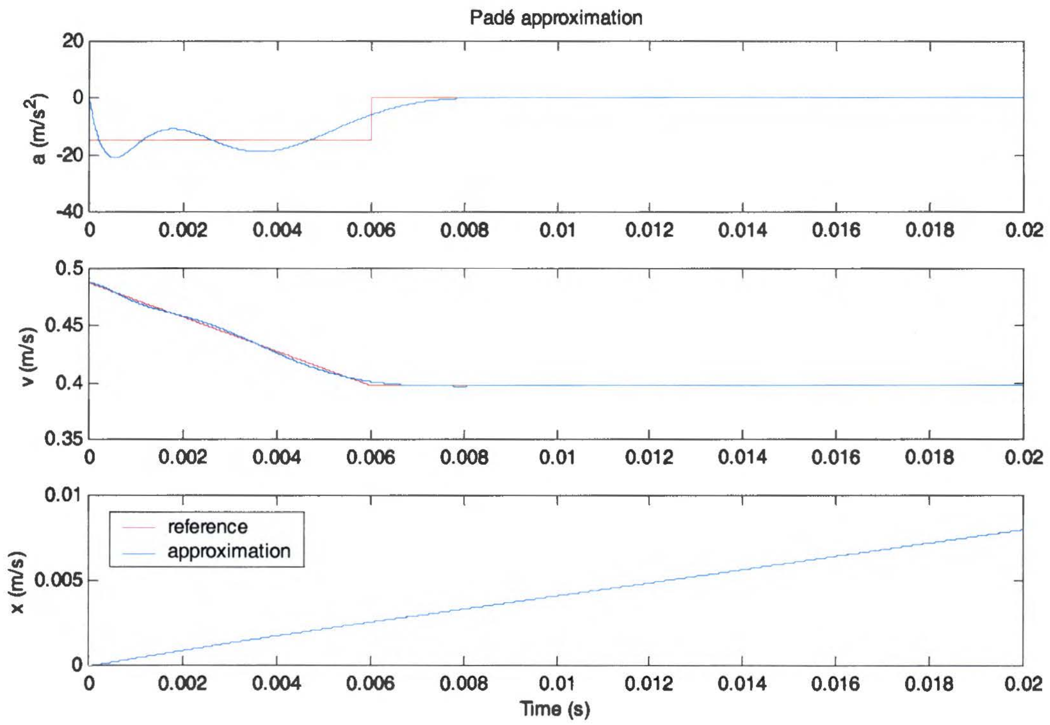


Figure 8.12 Profile approximation

The optimisation is being executed for the sampling frequencies 100, 150, 200, 250 and 300 Hz, using the sequential quadratic programming implementation of MATLABs Optimization Toolbox. As an

initial guess for the parameters, all closed-loop poles are placed in $z = 0.5$ and the feedforward is based on the inverse second order plant.
In figure 8.13, the achieved average integral quadratic error criterion is plotted against the sampling frequency. For $f_s = 100$ Hz, the problem turns out to be infeasible, as no solution satisfying the constraints was found. The plot shows that the constrained optimal average integral quadratic tracking error is not a monotonously decreasing function of the sampling rate.

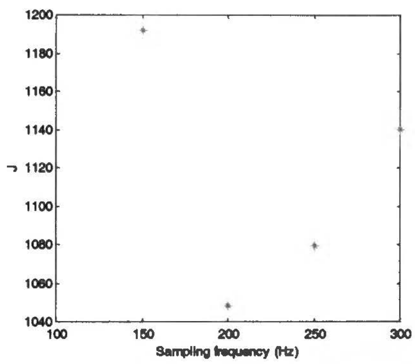


Figure 8.13 Constrained average integral quadratic tracking error vs. sampling rate

However, one is not really interested in the numerical value of the integral quadratic tracking error, but in the settling time after the worst-case correction profile. Simulation has turned out, that the settling time constraint is not satisfied by the solutions for $f_s = 150$ Hz and $f_s = 200$ Hz. The controller solution for $f_s = 250$ Hz is the solution with the lowest sampling rate, which satisfies the settling time constraint for both motor temperatures.
The optimal solution for $f_s = 250$ Hz is found to be:

$$K(z)=\frac{30298.7603(z-0.8544)(z-0.5359)}{(z-1)(z+0.7282)} \qquad (8.11)$$

$$K_a = 0.0317 \qquad (8.12)$$

$$K_v = 10.4481. \qquad (8.13)$$

Figure 8.14 shows PFG_{er} for the three plant templates and the upper bound on PFG_{er}. Figure 8.15 shows the tracking error response to the image position signal, obtained through simulation, and figure 8.16 shows a sheet velocity response. Four tracking error responses of the system at 20 °C are depicted in figure 8.17. One of the responses is for the case that the start of the deceleration period is synchronised with a sampling action, the other responses are tracking error responses to a delayed setpoint profile, where the delay η is 1, 2 or 3 milliseconds (remember that the asynchronous case is considered here).
Although the tracking error response to the correction profile shows a larger overshoot for the new controller, all the control demands are satisfied. The sampling rate is reduced with a factor four. The gain and phase margin of the new controller with respect to the delayed third order model at 20 °C are 2.9 dB and 21.2°. This seems a deterioration when compared with the robustness margins for the PID controller. The new controller is guaranteed to stabilise the plant templates in the plant set. If the plant templates truly reflect the maximal uncertainty of the motor parameters, there is no need to study the robustness margins. Moreover, this different approach to robustness ensures a level of robust performance, because the controller is optimised for the worst-case plant template. Realise that constraining the phase and gain margins generally does not yield a controller which achieves robust performance. A different plant set would have yielded a controller with different, possibly larger, stability margins.
It is interesting to regard the root locus diagram for the controller and the undelayed second order plant model in figure 8.18 (the root locus method is not suited for delayed plants). It shows a closed-loop pole configuration that was certainly not obtained if discrete-time controller design was applied according to the guidelines in [2]. The negative real closed-loop eigenvalue corresponds to a heavily

fluctuating response, which changes sign every sampling time. This might upset the plant, and for this reason it is advised not to place any closed-loop poles on the negative real axis. However, in the design procedure presented here, the continuous-time plant dynamics are taken into account. Apparently, the imposed bounds on the control system allow a closed-loop pole configuration like in figure 8.18.

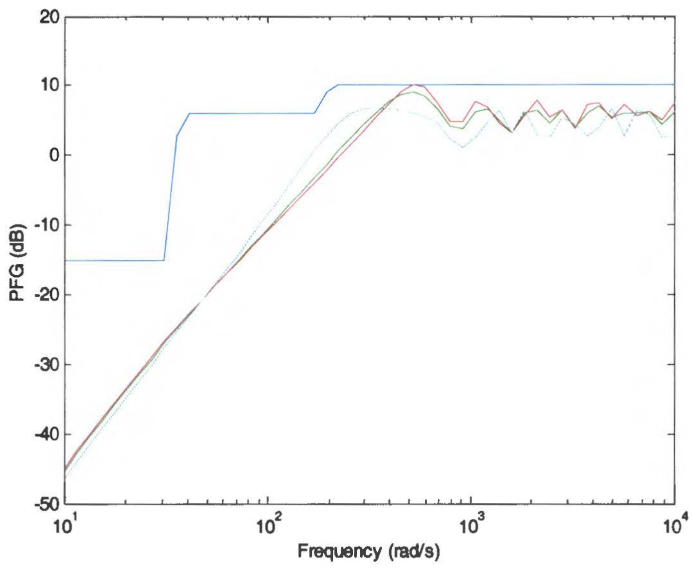


Figure 8.14 PFG_{er} for the different plant templates

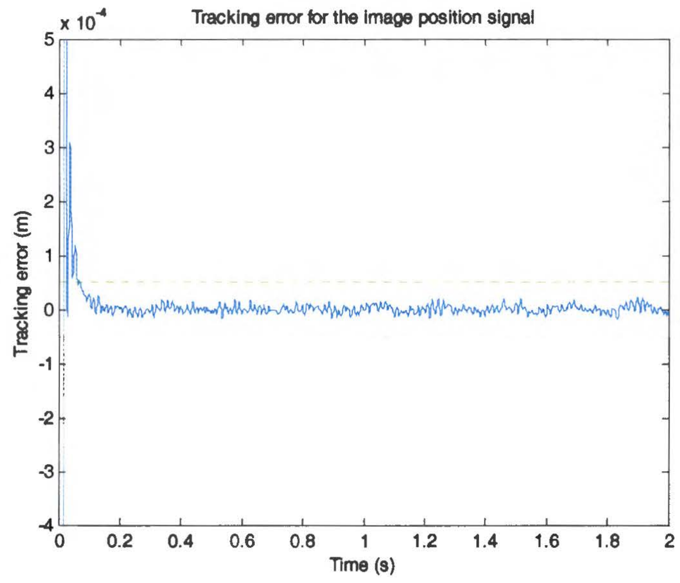


Figure 8.15 Tracking error of new controller for the image position signal

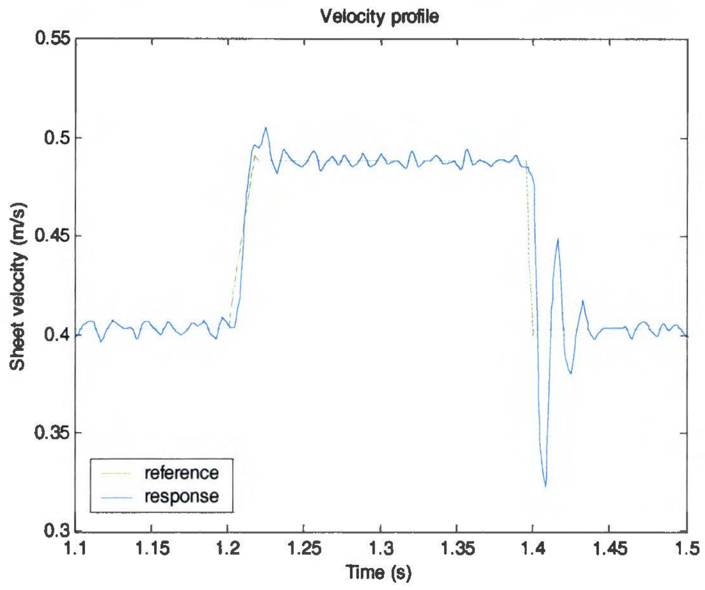


Figure 8.16 Sheet velocity response

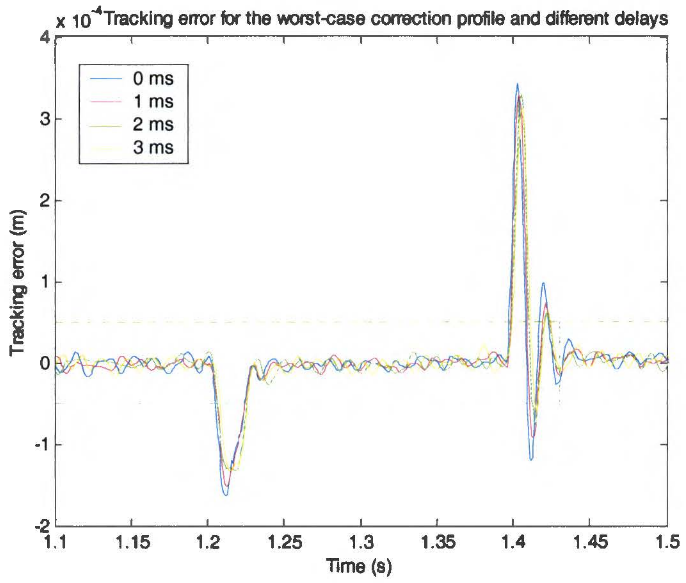


Figure 8.17 Tracking error for worst-case correction profile and different delays

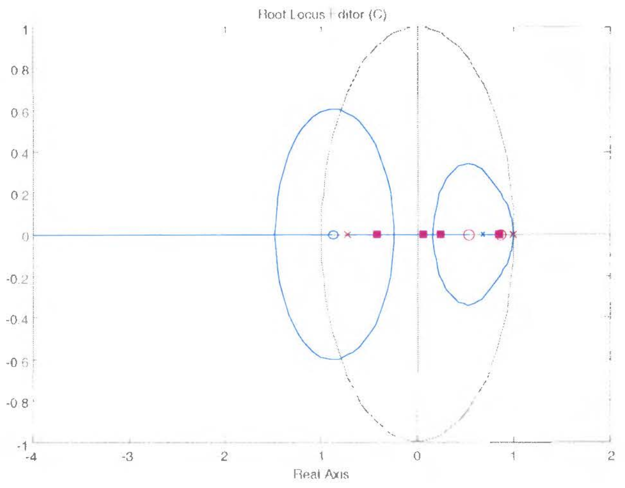


Figure 8.18 Root locus

8.5 The experimental set-up

A fully functional Océ copier (see figure 8.19a) is available for experimental purposes. The motor 5 controller software runs as a task under the real-time operating system VxWorks on a standardised Océ processor board with I/O functionality (figure 8.19c). The I/O functionality is used to generate PWM signals for one of the full-bridge DC-DC converters on a standardised power electronics board (figure 8.19d) driving motor 5. For experimentation purposes, the power electronics board is decoupled and connected to a separate processor board. This processor board is connected through a network connection with a PC running a development environment. Compiled C-code can be downloaded into the processor board memory after which it can be executed using the VxWorks command shell which is accessible from the PC. During an experiment, the copier is not in operation. The only motor being driven is motor 5 and there are no sheets passing through the paperpath.

Figure 8.19a Océ copier

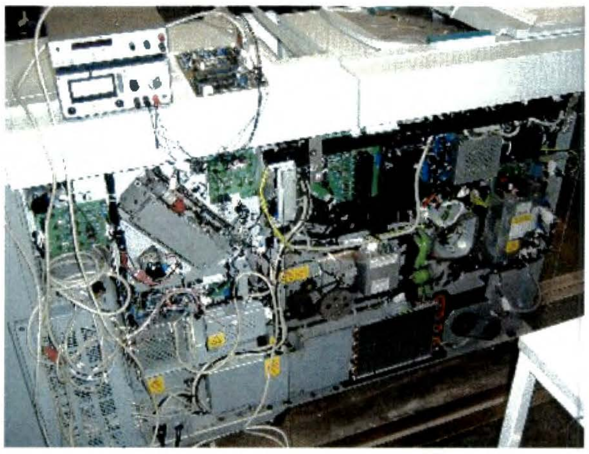


Figure 8.19b Open back end of copier

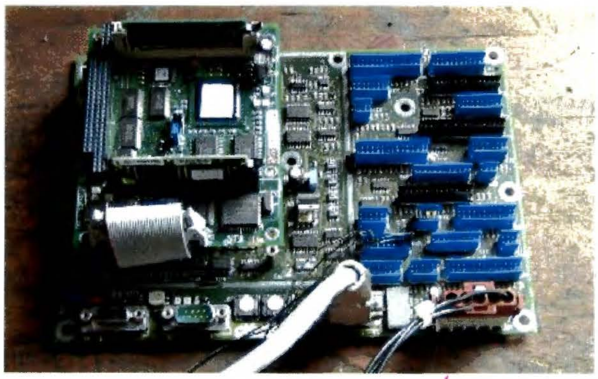


Figure 8.19c Processor board

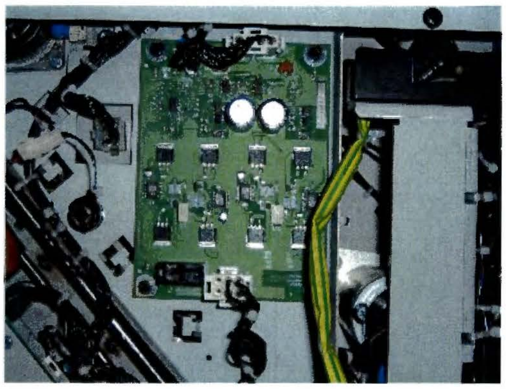


Figure 8.219d Power electronics board

8.6 Identification of experimental set-up

Some identification experiments are carried out in order to validate the simulation model. Using a small program running on the processor board, a random signal with a sampling rate of 4 kHz is applied to the motor during 15 seconds. The range of the motor voltage corresponds roughly to the range of the actuator signal in figure 8.11. The random signal and the position data from the encoder disc is recorded in an array. After the experiment, the contents of the array is being written to a file on the PC. The data is imported in MATLAB where the Identification Toolbox is being used to derive a numerical model using pem.

A model with the structure:

$$\begin{pmatrix}\ddot{\theta}\\ \dot{\theta}\end{pmatrix}=\begin{pmatrix}-\frac{1}{\tau_m} & 0\\ 1 & 0\end{pmatrix}\begin{pmatrix}\dot{\theta}\\ \theta\end{pmatrix}+\begin{pmatrix}\frac{k_T}{JR}\\ 0\end{pmatrix}V \qquad (8.11)$$

is being searched for. The model parameters are $-\tau_m$ and $\frac{k_T}{JR}$. The parameterised model is implicitly discretised through the ZOH transform with a sampling rate of 1/4000 s. Depending on the noise model, the output error or the prediction error is optimised through non-linear optimisation. In the case of the prediction error, a white noise source modelling the measurement errors and unmodelled dynamics, affects both the output and the state of the discretised system. For the case of the output error, only the output is disrupted by white noise.
The optimisation algorithm requires an initial guess for the model parameters. The data in table 8.1 can be used to form an initial estimate based on the simulation model:

$$-\frac{1}{\tau_m}=-\left(\frac{k_T k_E}{R}-B\right)\Big/J=-85 \qquad (8.12)$$

$$\frac{k_T}{JR}=1892\,. \qquad (8.13)$$

The model obtained for these initial guesses is validated through comparing its (differentiated) step response (the red curve in figure 8.20) with a measured voltage step response (cyan). The DC gain from applied motor voltage to steady state velocity shows a severe mismatch. Actually, the DC gain from the simulation model shows a discrepancy with the measurements too. The model minimising the output error gives a reliable estimate for the DC gain, but the estimate of the mechanical time constant is worst (according to the green curve in figure 8.20).
In a second try, the prediction error minimisation algorithm is equipped with initial estimates obtained from the voltage step response of the motor. The second model parameter can be expressed in terms of the DC gain $1/k_E$ and the mechanical time constant (if the damping is neglected):

$$\frac{k_T}{JR}=\frac{k_T}{\tau_m DR}\approx\frac{k_T}{\tau_m\left(\frac{k_T k_E}{R}\right)R}=\frac{1}{\tau_m}\cdot\frac{1}{k_E}\,. \qquad (8.14)$$

With the initial guesses obtained from step response data, the model parameters converge to:

$$-\frac{1}{\tau_m}=-55.3364 \qquad (8.15)$$

$$\frac{k_T}{JR}=1057.8\,. \qquad (8.16)$$

The model with these parameter values turns out to give a more accurate solution for the DC gain and the mechanical time constant (compare the black and cyan curves in figure 8.20). If the algorithm minimising the output error is equipped with the initial parameters obtained from the step response, then the solution converges to the one obtained with the initial values from the simulation model (see the green curve in figure 8.20). When the model mismatch between the simulation model and the identified model is attributed to a misfit in the inertia only, this misfit is about a factor 1.5 to 1.8.
In general, the output error is the preferred criterion for parameter estimation. It focuses on the evolution of the system behaviour over time, whereas the prediction error method puts all its effort in deriving models with excellent prediction capabilities. The latter approach uses the k^{th} output sample to predict the next output sample at time k + 1. As a sample is usually highly dependent on the

previous sample, the prediction error method focuses on fast, high frequency phenomena. The output error does not focus on any particular part of the spectrum, which makes it the preferred method for a general parameter estimation problem. However, the prediction error method turns out to give a more accurate (in terms of step response resemblance) model in this particular identification problem. For a thorough treatment of the differences between the output error and prediction method, the reader is referred to [33].

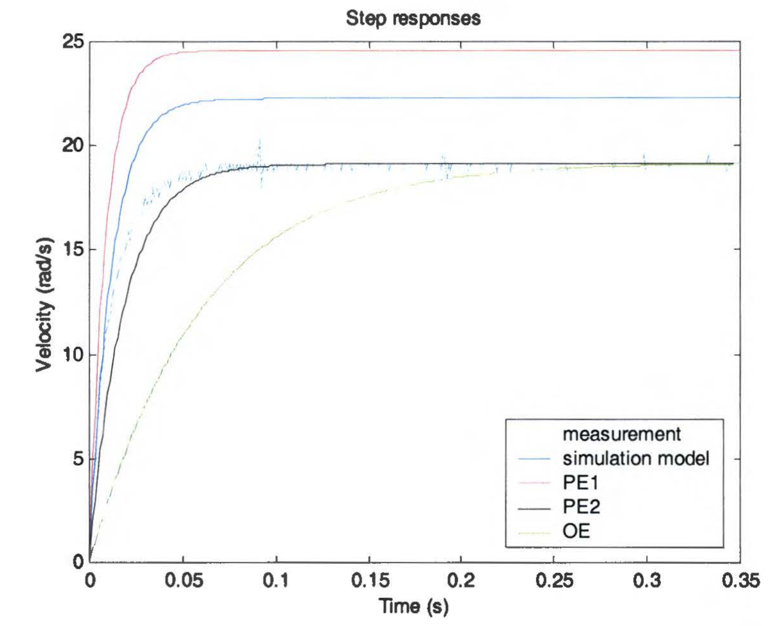


Figure 8.20 Model validation using step responses

Figure 8.21 shows an identified frequency response obtained through spectral analysis of data from an earlier conducted closed-loop identification experiment. The response suggests that there is at least one flexible mode apparent. It is probably introduced by the drive belt (see figure 8.3). The natural frequencies of the complex zero pair and the complex pole pair descendent from the flexible mode, are estimated from the figure as 63 Hz and 73 Hz respectively. The prediction error method is being used to estimate the damping.

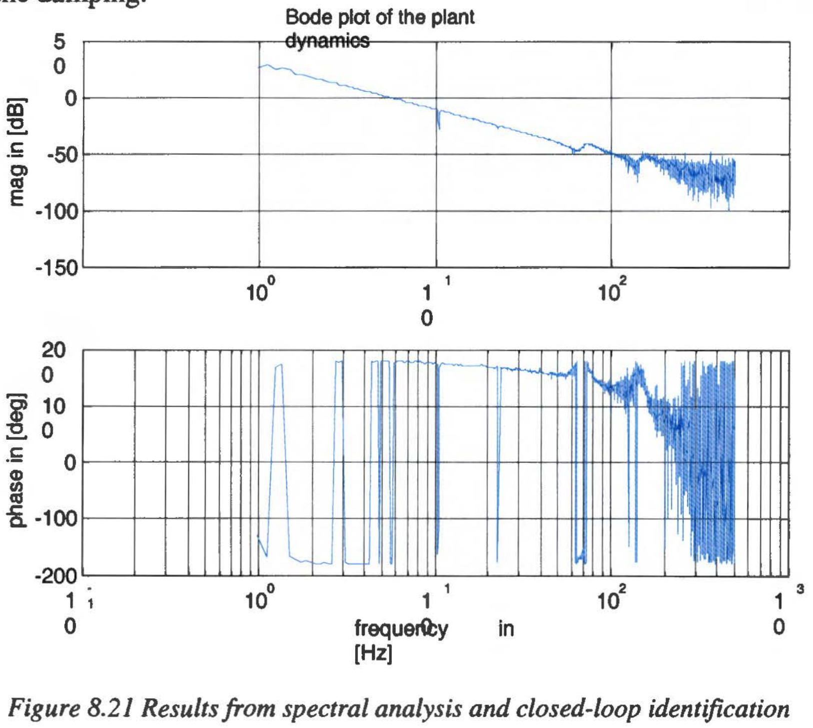


Figure 8.21 Results from spectral analysis and closed-loop identification

$$\begin{tikzcd}
V \arrow[r] & \boxed{\dfrac{1}{k_E(s\tau_m+1)}} \arrow[r, "\dot{\theta}"] & \boxed{\dfrac{\frac{\omega_n^2}{\omega_z^2}\left(s^2+\xi\omega_z s+\omega_z^2\right)}{s\left(s^2+\zeta\omega_n s+\omega_n^2\right)}} \arrow[r] & \varphi
\end{tikzcd}$$

Figure 8.22 Model for position controlled DC motor with flexible mode

Let a position controlled DC motor with a flexible mode somewhere in the transmission, be modelled by the block diagram in figure 8.22, where ω_n is the natural frequency and ζ the damping ratio of the complex pole pair and where the complex zero pair is given by the roots of $s^2+\xi\omega_z s+\omega_z^2$. The first transfer block is governed by the differential equation:

$$\ddot{\theta}=-\frac{1}{\tau_m}\dot{\theta}+\frac{1}{\tau_m k_E}V\,. \tag{8.17}$$

The integrator mode is included in the second transfer block. This allows the conversion from transfer function to control canonical state space form:

$$\begin{aligned}\dot{x}&=\begin{pmatrix}-\zeta\omega_n & -\omega_n^2 & 0\\ 1 & 0 & 0\\ 0 & 1 & 0\end{pmatrix}x+\begin{pmatrix}1\\0\\0\end{pmatrix}\dot{\theta}\,.\\ \varphi&=\begin{pmatrix}\frac{\omega_n^2}{\omega_z^2} & \frac{\omega_n^2}{\omega_z}\xi & \omega_n^2\end{pmatrix}x\end{aligned} \tag{8.18}$$

Combination of (8.17) and (8.18) yields the model structure:

$$\begin{aligned}\begin{pmatrix}\ddot{\theta}\\ \dot{x}\end{pmatrix}&=\begin{pmatrix}-\frac{1}{\tau_m} & 0 & 0 & 0\\ 1 & -\zeta\omega_n & -\omega_n^2 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\end{pmatrix}\begin{pmatrix}\dot{\theta}\\ x\end{pmatrix}+\begin{pmatrix}\frac{1}{\tau_m k_E}\\ 0\\ 0\\ 0\end{pmatrix}V\,.\\ \varphi&=\begin{pmatrix}0 & \frac{\omega_n^2}{\omega_z^2} & \frac{\omega_n^2}{\omega_z}\xi & \omega_n^2\end{pmatrix}\begin{pmatrix}\dot{\theta}\\ x\end{pmatrix}+0\cdot V\end{aligned} \tag{8.19}$$

The two entries containing ζ and ξ are regarded as model parameters. τ_m and $1/k_E$ are obtained from the identified second order model, and ω_z and ω_n are fixed to $2\pi\cdot 63$ and $2\pi\cdot 73$.
Again the prediction error method is sensitive to initial guesses. However, if the initial guesses $\zeta=0.25$ and $\zeta=0.71$ are supplied, the solution converges to a set of parameters which are almost equal to the initial guesses. This gives some confidence in the parameters, as the solution seems to be an (local) optimum.
The reader interested in identification is referred to [25], [26] and [33].

8.7 Controller design & implementation for the experimental set-up

A controller is designed for the experimental set-up, along the same lines as was done for the simulation model. The new plant set consists of three models: the identified second order model (the nominal model), the identified fourth order model accounting for the flexible mode, and a fourth order perturbed model. In the perturbed model the motor constant is decreased with 10% and the mechanical time constant is increased with 20%. This perturbation corresponds roughly to a temperature increase of 50 °C. Further, the natural frequency of the flexible mode is increased with 10 Hz (actually, a decrease of frequency would be more challenging for controller design), modelling the uncertainty in the resonance frequency (caused by wear of the belt, for example). The Bode diagrams of the plant templates are depicted in figure 8.23. A controller with a sampling rate of 250 Hz is optimised for the new plant set, as was done in section 8.4.

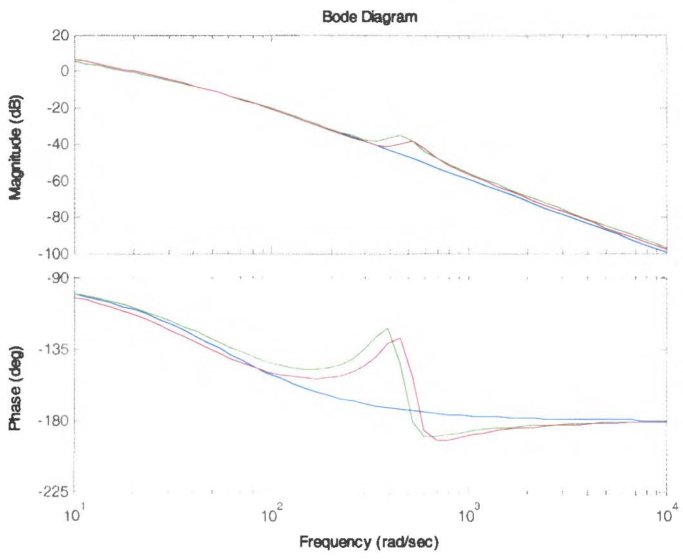


Figure 8.23 Bode diagrams of plant templates

The controller is implemented on the processor board. The computations are executed in floating point arithmic. As the processor platform does not support floating point numbers in hardware, the floating point type has to be emulated, which results in a relatively high computation time: 210 μs. The computation delay was measured as the time between sampling and the output of a new actuator sample. This computation delay was accounted for in the optimisation.
For implementation, the controller is converted to control canonical state space form. This allows an easy implementation of integrator antiwindup and an accurate representation of the integrator mode. If the controller was implemented as a difference equation, inaccuracies in the number representation may cause the unit eigenvalue of the integrator mode to deviate. A sample of C code implementing the actual controller action is given below. It gives an impression of the implementation of the controller.

```
/* actual controller code */
U = 0.4172*X1 - 1.781*X2 + 3.438*E + 3.6729*(velocity[counter] -
     36.0) + 70*acceleration[counter];

/* actuator saturation */
if (U > 359.0)
{
     U = 359.0;
}
else if (U < -359.0)
{
     U = -359.0;
}
set_pwm(- (int) U);

if ((U == 359.0) || (U == -359.0))
     X1 = X1;              /* integrator antiwindup */
else
     X1 = X1 + 0.02493*E;
```

```
X2 = -0.2398*X2 + 1.803*E;
```

The reference trajectory is computed offline with a resolution of 1 ms, and it is stored in memory. The controller's sampling frequency is 250 Hz. This means that the setpoint profile can be shifted with respect to sampling instant in four steps (asynchronous case). The measured tracking error response for these four cases is depicted in figure 8.23. The measurements were taken with a sampling rate of 1 kHz. The black curve shows the measured tracking error for the present PID controller.

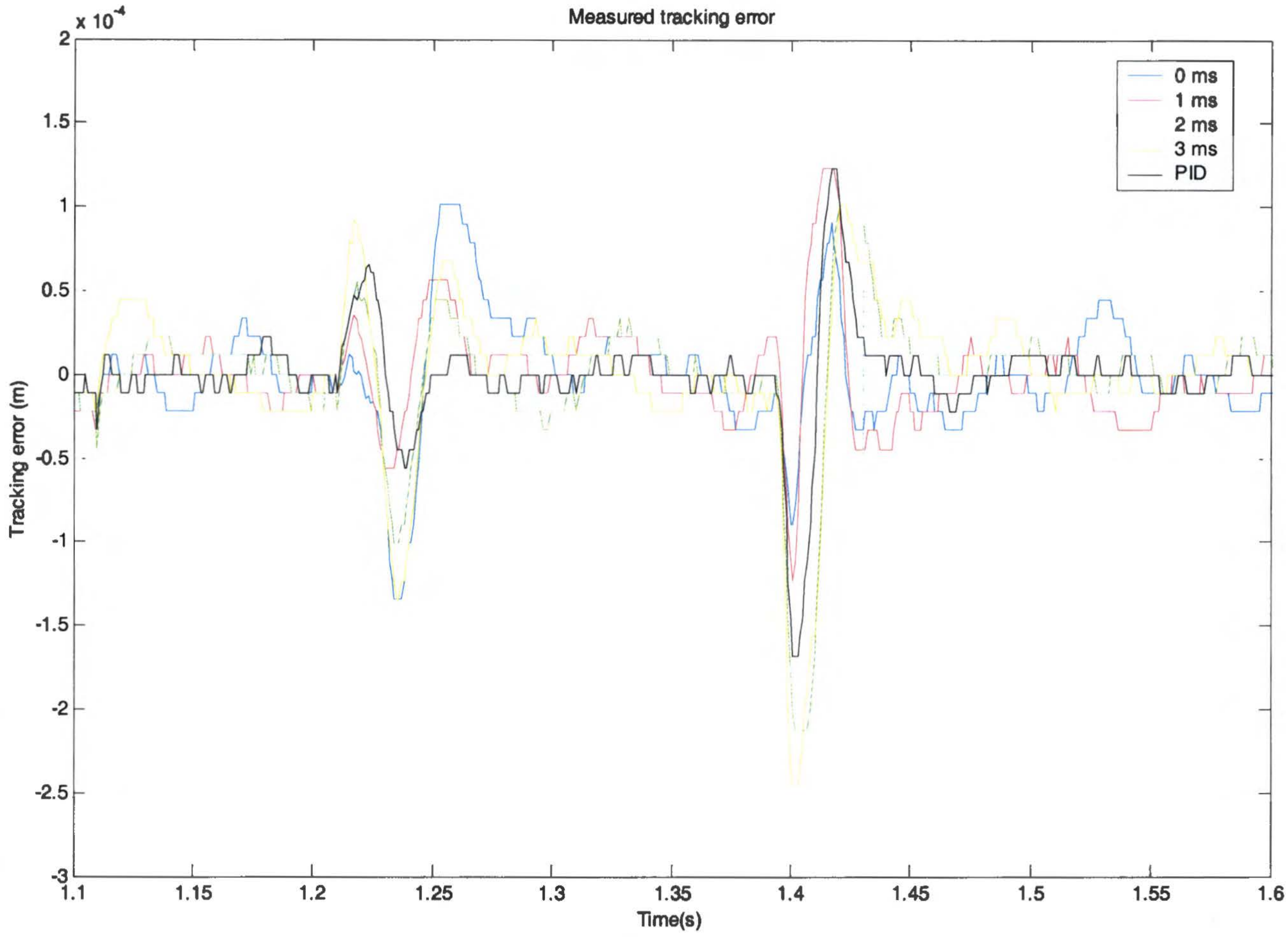


Figure 8.23 Measured tracking error

The bounds on the tracking error are expressed by the dotted cyan lines in figure 8.23. In order to satisfy the settling time constraint, the tracking error has to be within a bound of 50 μm after 1.43 seconds. The PID controller achieves this demand, and so does the new 250 Hz controller for the setpoint delays of 0 and 1 ms. For the other two setpoint delays the settling time constraint is violated. But as the tracking error responses for the PID controller and the new controller with a sampling rate of 250 Hz are comparable, one can say that roughly a factor 4 on sampling rate reduction can be achieved. The new control algorithm is not complexer than the discrete-time PID control algorithm
The results in figure 8.17 are incomparable to the results in the figure above. This is due to the fact that different controllers are obtained for the simulation model and the experimental set-up, and different plant set are used. The tracking error responses for the discrete-time PID controller at 1 kHz differ in the figures 8.9 and 8.23, due to the discrepancy between the simulation model and the identified model, and unmodelled dynamics.
If one is willing to modify the setpoint profile, a controller can be optimised for the synchronous case. In this case, the acceleration profile has to be constant between sampling times. The worst-case correction profile can be approximated with a profile which has constant acceleration during a sampling period of 4 ms. In this case, the 15 m/s^2 deceleration period of 6 ms is approximated with a 15 m/s^2 deceleration during 4 ms followed by a period of 4 ms, 7.5 m/s^2 deceleration. A 250 Hz

sampling rate controller optimised for this setpoint profile with a synchronous acceleration profile might achieve a tracking error response like in figure 8.24.

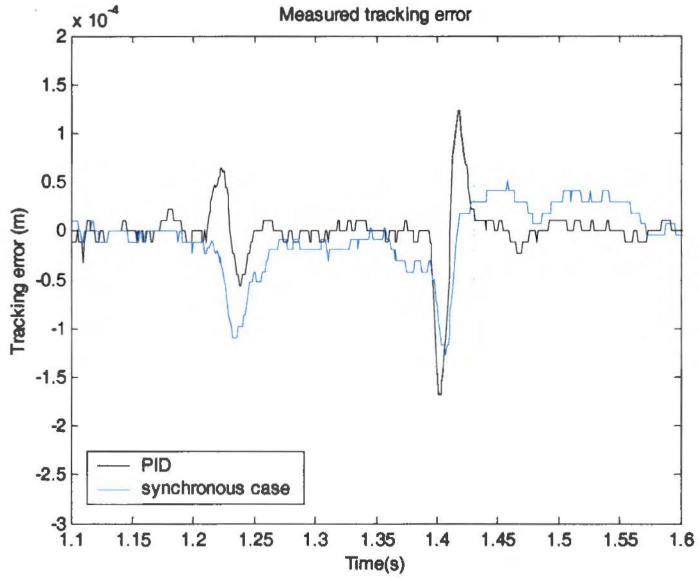


Figure 8.24 Tracking error response of optimal controller for the synchronous case

9. Conclusion

A systematic approach to the design of digital motor controllers is presented. The proposed design method is a controller optimisation procedure which regards the continuous-time performance of the hybrid, mixed discrete-time/continuous-time control system. The explainability of frequency domain performance constraints is combined with the suitability of integral quadratic time-domain constraints to optimisation. The sampling rate independence of the criteria and constraints allows a fair comparison of controllers with different sampling rates.
Useful insights into the capabilities and limitations of sampled-data control are obtained through frequency domain analysis. Frequency gain concepts pose a useful tool for the numerical analysis of sampled-data control systems.
The proposed controller design method is applied to a representative motor control case in an Océ copier. Simulation and experimental results show that a sampling rate reduction from 1kHz to 250 Hz can be accomplished while maintaining the controller complexity, provided that a reasonable accurate model is available. So, with little effort, a significant amount of efficiency improvement can be achieved.

10. Recommendations

- H_2 optimal control of mixed continuous-time/discrete-time and mixed stochastic/deterministic systems offers a challenging direction for further research. If a sensible method for weighing the terms in the cost function is found, the sampled-data H_2 control design framework might offer a useful framework for motor controller design. It has considerable computational advantages over the non-linear optimisation with frequency domain constraints. Controller reduction techniques can be applied to reduce the typical high order H_2 optimal controller.

- A lot of work can be done on adjusting the optimisation algorithm to the optimisation task. A study on the applicability of genetic algorithms to finding the set of Pareto optimal controllers is useful, for instance. Probably, the use of a reduced order H_2 optimal controller as an initial guess for the non-linear optimisation algorithm, would reduce the amount of iterations necessary to arrive at an optimal solution. It is also interesting to know, if a combination of sampled-data and discrete-time H_∞ optimisation techniques can be applied to shape the PFG_{er} of a sampled-data system. For well behaved systems (systems with a low fidelity index) the DFG_{er} might be used to approximate the PFG_{er} for low frequencies, and the RFG_{er} generally approximates the PFG_{er} for high frequencies [12]. These suggestions might offer directions for further research.

- Although the design method presented in section 7.6 offers a structured way of designing motor controllers with low sampling rates, the optimisation procedure may be quite cumbersome. It would be advantageous for acceptation and practical application, if some design guidelines for motor controller design could be derived, using the presented design framework. Moreover, some work is still to be done on the interpretation of the optimisation results. It is desired to relate the closed-loop pole placement, resulting from the optimisation process, with the upper bound on PFG_{er} and the level of the integral quadratic tracking error.

- To increase the robustness of the optimisation procedure, the numerical reliability of the various computations have to be increased. The computation of the average integral quadratic tracking error suffers from numerical problems, especially if the order of the used Padé approximation is high. The perturbation of the integrator mode, ε (see appendix A.6), has also an effect on the outcome. The application of multirate lifting [32] can overcome the necessity of the Padé approximation, which seems to cause the numerical problems. The asynchronous acceleration profile can be approximated to any desired degree of accuracy, with an impulse response of a discrete-time system with a sampling time N times higher than the sampling rate of the controller. Multirate lifting transforms a multirate sampled-data system to a norm-equivalent single rate discrete-time system.
  The computation of the *DFG* can be inaccurate for low frequencies, due to the computation of the matrix exponent in MATLAB. Similar inaccuracies are encountered when Bode diagrams of delayed systems are plotted.

Appendix A. Computations

A.1 Computing integrals involving the matrix exponential

Integrals involving the matrix exponentials are often encountered in sampled-data control theory. It turns out that a certain form of integrals involving the matrix exponential can be computed by calculating a matrix exponential. This lemma is used in the computation of the ZOH transform, the computation of the *PFG* and the derivations of integral quadratic measures for sampled-data control. The matrix exponential of a square matrix A is defined by the series:

$$e^{A\tau} = I + A\tau + \frac{1}{2!}A^2\tau^2 + \frac{1}{3!}A^3\tau^3 + \dots . \tag{A.1}$$

Let A_1 and A_2 be square matrices. If

$$\begin{pmatrix} F_1 & G_1 \\ 0 & F_2 \end{pmatrix} = \exp\left(\begin{pmatrix} A_1 & B_1 \\ 0 & A_2 \end{pmatrix}\tau\right) \tag{A.2}$$

then

$$F_1 = e^{A_1\tau}, \quad F_2 = e^{A_2\tau} \tag{A.3}$$

this follows directly from the block upper triangular structure of the matrices, and

$$G_1 = \int_0^\tau e^{A_1(\tau-\sigma)} B_1 e^{A_2\sigma} d\sigma . \tag{A.4}$$

This follows from the fact that if (A.2) is differentiated one gets:

$$\frac{d}{d\tau}\begin{pmatrix} F_1 & G_1 \\ 0 & F_2 \end{pmatrix} = \begin{pmatrix} A_1 & B_1 \\ 0 & A_2 \end{pmatrix}\begin{pmatrix} F_1 & G_1 \\ 0 & F_2 \end{pmatrix} \tag{A.5}$$

so

$$\frac{d}{d\tau}G_1 = A_1 G_1 + B_1 F_2 = A_1 G_1 + B_1 e^{A_2\tau} . \tag{A.6}$$

Solving this differential equation yields:

$$G_1 = \int_0^\tau e^{A_1(\tau-\sigma)} B_1 e^{A_2\tau} d\sigma . \tag{A.7}$$

In a similar way one can derive (see [15]):

$$\exp\left(\begin{pmatrix} A_1 & B_1 & C_1 \\ 0 & A_2 & B_2 \\ 0 & 0 & A_3 \end{pmatrix}\tau\right) = \begin{pmatrix} F_1 & G_1 & H_1 \\ 0 & F_2 & G_2 \\ 0 & 0 & F_3 \end{pmatrix} \tag{A.8}$$

where

$$F_j = e^{A_j\tau}, \quad j = 1,2,3 \tag{A.9}$$

$$G_j = \int_0^{\tau} e^{A_j(\tau-\sigma)} B_j e^{A_{j+1}\sigma} d\sigma, \quad j = 1,2 \tag{A.10}$$

$$H_j = \int_0^{\tau} e^{A_j(\tau-\sigma)} C_j e^{A_{j+2}\sigma} d\sigma + \int_0^{\tau}\int_0^{\sigma} e^{A_j(\tau-\sigma)} B_j e^{A_{j+1}(\sigma-\rho)} B_{j+1} e^{A_{j+2}\rho} d\rho d\sigma, \quad j = 1 \tag{A.11}$$

and $\tau \geq 0$.

A.2 Computing the ZOH transform

Let continuous-time plant *P* have state space matrices *A*, *B*, *C* and *D* and a delay $0 \leq \lambda < h$, where *h* is the sampling time. Let *u* be the input of the plant and *y* the output, *x* is the internal state of *P*. The evolution of the plant from sampling time to sampling time is obtained by integrating the continuous-time state space equations from *kh* to *kh* + *h*, where *k* is a positive integer. This results in (see also [8]):

$$\begin{aligned} x(kh+h) &= e^{Ah}x(kh) + \int_0^{\lambda} e^{A(\lambda-\tau)} B d\tau \cdot u(kh-h) + \int_{\lambda}^{h} e^{(h-\tau)} B d\tau \cdot u(kh) \\ y(kh) &= Ce^{Ah}x(kh) + \int_0^{\lambda} Ce^{A(\lambda-\tau)} B d\tau \cdot u(kh-h) + \left(\int_{\lambda}^{h} Ce^{(h-\tau)} B d\tau + D \right) \cdot u(kh) \end{aligned} . \tag{A.12}$$

If the delayed input sample *u*(*kh* – *h*) is adopted into the new state vector:

$$x_d[k] = \begin{pmatrix} x(kh) \\ u(kh-h) \end{pmatrix} \tag{A.13}$$

then the state space equations of the ZOH transform of *P* can be written as:

$$\begin{aligned} x_d[k+1] &= A_d x_d[k] + B_d u[k] \\ y[k] &= C_d x_d[k] + D_d u[k] \end{aligned} \tag{A.14}$$

where:

$$A_d = \begin{pmatrix} e^{Ah} & \int_0^{\lambda} e^{A(\lambda-\tau)} d\tau \\ 0 & 1 \end{pmatrix} \tag{A.15}$$

$$B_d = \int_0^{h-\lambda} e^{A\tau} B d\tau \tag{A.16}$$

$$C_d = \begin{pmatrix} Ce^{Ah} & \int_0^{\lambda} Ce^{A(\lambda-\tau)} B d\tau \end{pmatrix} \tag{A.17}$$

$$D_d = \int_0^{h-\lambda} Ce^{A\tau}Bd\tau + D. \tag{A.18}$$

Using the results of section A.1, the integrals can be written as matrix exponentials:

$$A_d = \exp\left(\begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix}\lambda\right)\begin{pmatrix} e^{A(h-\lambda)} & 0 \\ 0 & 1 \end{pmatrix} \tag{A.19}$$

$$B_d = \begin{pmatrix} I & 0 \end{pmatrix}\exp\left(\begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix}(h-\lambda)\right)\begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{A.20}$$

$$C_d = \begin{pmatrix} C & 0 \end{pmatrix}\exp\left(\begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix}\lambda\right)\begin{pmatrix} e^{A(h-\lambda)} & 0 \\ 0 & 1 \end{pmatrix} \tag{A.21}$$

$$D_d = \begin{pmatrix} C & 0 \end{pmatrix}\exp\left(\begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix}(h-\lambda)\right)\begin{pmatrix} 0 \\ 1 \end{pmatrix} + D. \tag{A.22}$$

A.3 Computing the PFG

Parseval's theorem gives an expression for the power of a signal *z*(*t*) in the frequency domain:

$$P^2(z(t)) = \frac{1}{2\pi}\int_{-\infty}^{\infty} Z^*(j\omega)Z(j\omega)d\omega \tag{A.23}$$

where * denotes the conjugate transpose. Substituting general form (3.10) into (A.23) gives:

$$\begin{aligned} P^2(z(t)) = &\frac{1}{2\pi}\int_{-\infty}^{\infty} W^*(j\omega)G_{11}^*(j\omega)G_{11}(j\omega)W(j\omega)d\omega \\ &+\frac{1}{2\pi}\int_{-\infty}^{\infty} W^*(j\omega)G_{11}^*(j\omega)G_{12}(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)\cdot\frac{1}{h}\sum_{n=-\infty}^{\infty}G_{21}(j\omega-jn\omega_s)W(j\omega-jn\omega_s)d\omega \\ &+\frac{1}{2\pi}\int_{-\infty}^{\infty}\frac{1}{h}\sum_{n=-\infty}^{\infty}W^*(j\omega-jn\omega_s)G_{21}^*(j\omega-jn\omega_s)R_d^*\left(e^{j\omega h}\right)H^*(j\omega)G_{12}^*(j\omega)G_{11}(j\omega)W(j\omega)d\omega \\ &+\frac{1}{2\pi}\int_{-\infty}^{\infty}\frac{1}{h}\sum_{n=-\infty}^{\infty}W^*(j\omega-jn\omega_s)G_{21}^*(j\omega-jn\omega_s)R_d^*\left(e^{j\omega h}\right)H^*(j\omega)G_{12}^*(j\omega)\cdot \\ &\quad G_{12}(j\omega)H(j\omega)R_d\left(e^{j\omega h}\right)\cdot\frac{1}{h}\sum_{n=-\infty}^{\infty}G_{21}(j\omega-jn\omega_s)W(j\omega-jn\omega_s)d\omega \end{aligned}$$

$$. \tag{A.24}$$

For the *PFG* case, *w*(*t*) is a single (possibly multivariable) sinusoid, so:

$$W(j\omega) = 2\pi w_0\delta(\omega-\omega_0) \tag{A.25}$$

Substitution of (A.25) into (A.24) and exploiting the sieve property of the Dirac delta pulse yields:

$$\begin{aligned}
P^2(z(t)) &= w_o^* G_{11}^*(j\omega_0) G_{11}(j\omega_0) w_0 + w_o^* \frac{1}{h} G_{11}^*(j\omega_0) G_{12}(j\omega_0) H(j\omega_0) R_d\left(e^{j\omega_0 h}\right) G_{21}(j\omega_0) w_0 \\
&+ w_o^* \frac{1}{h} G_{21}^*(j\omega_0) R_d^*\left(e^{j\omega_0 h}\right) H^*(j\omega_0) G_{12}^*(j\omega_0) G_{11}(j\omega_0) w_0 \\
&+ w_o^* \frac{1}{h} G_{21}^*(j\omega_0) R_d^*\left(e^{j\omega_0 h}\right) \cdot \\
&\left[\frac{1}{h}\sum_{n=-\infty}^{\infty} H^*(j\omega_0 - jn\omega_s) G_{12}^*(j\omega_0 - jn\omega_s) G_{12}(j\omega_0 - jn\omega_s) H(j\omega_0 - jn\omega_s)\right] \cdot \\
&R_d\left(e^{j\omega_0 h}\right) G_{21}(j\omega_0) w_0 \\
&= w_0^* A(\omega_0) w_0
\end{aligned} \tag{A.26}$$

By construction, $A(\omega_0)$ is a symmetric positive definite matrix, so it has positive singular values. The maximum power gain is given by the square root of the maximum singular value:

$$PFG_{zw}(\omega) = \sqrt{\sigma_{\max}(A(\omega))}. \tag{A.27}$$

Note that the term in (A.26) containing the infinite summation describes the ZOH transform of the system in figure A.1.

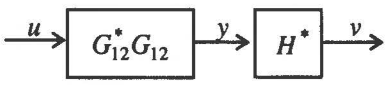


Figure A.1 System for computing infinite summation

Suppose G_{12} has system matrices A, B, C and D. Its transfer function is:

$$G_{12}(j\omega) = C(j\omega I - A)B + D. \tag{A.28}$$

The adjoint system G_{12}^* has transfer function:

$$G_{12}^*(j\omega) = G_{12}^T(-j\omega) = B^T\left(-j\omega I - A^T\right)C^T + D^T = -B^T\left(j\omega I - \left(-A^T\right)\right)C^T + D^T. \tag{A.29}$$

So G_{12}^* has system matrices $-A^T$, C^T, $-B^T$, D^T. It is easy to derive that the series connection $G_{12}^* G_{12}$ has state space representation:

$$\begin{aligned}
\dot{x}(t) &= \begin{pmatrix} A & 0 \\ C^T C & -A^T \end{pmatrix} x(t) + \begin{pmatrix} B \\ C^T D \end{pmatrix} u(t) \\
y(t) &= \begin{pmatrix} D^T C & -B^T \end{pmatrix} x(t) + D^T D u(t)
\end{aligned}. \tag{A.30}$$

Note that if G_{12} contains a delay, there exists no state space representation for G_{12} as it is non-rational. However, the delay is cancelled in $G_{12}^* G_{12}$, so the state space matrices of the undelayed system G_{12} can be used instead.

The adjoint of the zero order hold has transfer function:

$$H^*(j\omega)=\frac{1-e^{j\omega h}}{-j\omega}=\frac{e^{j\omega h}-1}{j\omega} \tag{A.31}$$

Let $h^*(t)$ denote the impulse response of the conjugated hold operator. It is obtained from (A.31) through inverse Fourier transform.

$$h^*(t)=\begin{cases}1 & -h\le t\le 0\\ 0 & elsewhere\end{cases} \tag{A.32}$$

If the system in figure A.1 is prepended with a zero order hold, then the signal y is described by the equations:

$$\textit{for } kh\le t<kh+h,\ k\in\mathbb{N}:$$
$$\begin{cases}x(t)=\exp\left(\begin{pmatrix}A & 0\\ C^TC & -A^T\end{pmatrix}(t-kh)\right)x(kh)+\int\limits_{kh}^{t}\exp\left(\begin{pmatrix}A & 0\\ C^TC & -A^T\end{pmatrix}(t-\tau)\right)d\tau\cdot\begin{pmatrix}B\\ C^TD\end{pmatrix}u(kh)\\ y(t)=\begin{pmatrix}D^TC & -B^T\end{pmatrix}x(t)+D^TDu(kh)\end{cases} \tag{A.33}$$

where x is the internal state of $G_{12}^*G_{12}$. $v(t)$ is the convolution of $h^*(t)$ with $y(t)$ which yields:

$$\textit{for } kh\le t<kh+h,\ k\in\mathbb{N}:$$
$$\begin{aligned}v(t-h)&=\int\limits_{kh}^{t}y(\sigma)d\sigma=\begin{pmatrix}D^TC & -B^T\end{pmatrix}\int\limits_{kh}^{t}x(\sigma)d\sigma+D^TDu(kh)\\ &=\begin{pmatrix}D^TC & -B^T\end{pmatrix}\int\limits_{kh}^{t}\exp\left(\begin{pmatrix}A & 0\\ C^TC & -A^T\end{pmatrix}(\sigma-kh)\right)d\sigma\cdot x(kh)\\ &+\begin{pmatrix}D^TC & -B^T\end{pmatrix}\int\limits_{kh}^{t}\int\limits_{kh}^{\sigma}\exp\left(\begin{pmatrix}A & 0\\ C^TC & -A^T\end{pmatrix}(\sigma-\tau)\right)d\tau d\sigma\cdot\begin{pmatrix}B\\ C^TD\end{pmatrix}u(kh)+D^TDu(kh)\end{aligned} \tag{A.34}$$

for $t=kh+h$ this yields:

$$\begin{aligned}v(kh)&=\begin{pmatrix}D^TC & -B^T\end{pmatrix}\int\limits_{kh}^{kh+h}\exp\left(\begin{pmatrix}A & 0\\ C^TC & -A^T\end{pmatrix}(\sigma-kh)\right)d\sigma\cdot x(kh)\\ &+\begin{pmatrix}D^TC & -B^T\end{pmatrix}\int\limits_{kh}^{kh+h}\int\limits_{kh}^{\sigma}\exp\left(\begin{pmatrix}A & 0\\ C^TC & -A^T\end{pmatrix}(\sigma-\tau)\right)d\tau d\sigma\cdot\begin{pmatrix}B\\ C^TD\end{pmatrix}u(kh)+D^TDu(kh)\end{aligned} \tag{A.35}$$

Substitution of variables gives:

$$\begin{aligned}v(kh)&=\begin{pmatrix}D^TC & -B^T\end{pmatrix}\int\limits_{0}^{h}\exp\left(\begin{pmatrix}A & 0\\ C^TC & -A^T\end{pmatrix}\sigma\right)d\sigma\cdot x(kh)\\ &+\begin{pmatrix}D^TC & -B^T\end{pmatrix}\int\limits_{0}^{h}\int\limits_{0}^{\sigma}\exp\left(\begin{pmatrix}A & 0\\ C^TC & -A^T\end{pmatrix}(\sigma-\tau)\right)d\tau d\sigma\cdot\begin{pmatrix}B\\ C^TD\end{pmatrix}u(kh)+D^TDu(kh)\end{aligned} \tag{A.36}$$

Now it is possible to write the infinite summation in (A.26) as a transfer function of a discrete-time system;

$$C_d\left(e^{j\omega h}I - A_d\right)B_d + D_d \tag{A.37}$$

where

$$A_d = \exp\left(\begin{pmatrix} A & 0 \\ C^T C & -A^T \end{pmatrix} h\right) \tag{A.38}$$

$$B_d = \int_0^h \exp\left(\begin{pmatrix} A & 0 \\ C^T C & -A^T \end{pmatrix} \tau\right) d\tau \cdot \begin{pmatrix} B \\ C^T D \end{pmatrix} \tag{A.39}$$

$$C_d = \begin{pmatrix} D^T C & -B^T \end{pmatrix} \int_0^h \exp\left(\begin{pmatrix} A & 0 \\ C^T C & -A^T \end{pmatrix} \sigma\right) d\sigma \tag{A.40}$$

$$D_d = \begin{pmatrix} D^T C & -B^T \end{pmatrix} \int_0^h \int_0^\sigma \exp\left(\begin{pmatrix} A & 0 \\ C^T C & -A^T \end{pmatrix} (\sigma - \tau)\right) d\tau d\sigma \cdot \begin{pmatrix} B \\ C^T D \end{pmatrix} + D^T D. \tag{A.41}$$

Using the results of section A.1, the state space matrices can be computed through the computation of one matrix exponential:

$$\left(\begin{array}{c:c:c} 1 & C_d & D_d \\ \hdashline 0 & A_d & B_d \\ \hdashline 0 & 0 & 1 \end{array}\right) = \exp\left(\left(\begin{array}{c:cc:c} 0 & D^T C & -B^T & D^T D \\ \hdashline 0 & A & 0 & B \\ 0 & C^T C & -A^T & C^T D \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right) h\right). \tag{A.42}$$

This result allows the exact computation of the *PFG.*

A.4 The lifting approach to optimal sampled-data control

Three steps can be distinguished in applying the lifting approach to a sampled-data optimal control problem.

1. The sampled-data system is 'lifted' to a norm-equivalent infinite-dimensional time invariant discrete-time system.
2. The infinite-dimensional discrete-time problem is converted to a norm-equivalent finite-dimensional discrete-time problem.
3. The optimal control problem is solved for the norm-equivalent finite-dimensional discrete-time system.

The first two steps will be explained in this section.

The lifting action 'chops' a $L_2[0,\infty)$ signal x in parts, where each part $\hat{x}[k]$ is an element of $L_2[0,h)$ and h is the sampling time. This operation is illustrated in figure A.2. The lifted signal $\hat{x}$ is the sequence $\hat{x}[k]$, $k \in \mathbb{N}$. The l_2 norm of such a lifted signal is defined as:

$$\|\hat{x}\|_{l_2} = \left(\sum_{k=0}^{\infty} \|\hat{x}[k]\|_{L_2[0,h)}^2\right)^{\frac{1}{2}} \tag{A.43}$$

and it is equal to the L_2 norm of the continuous signal;

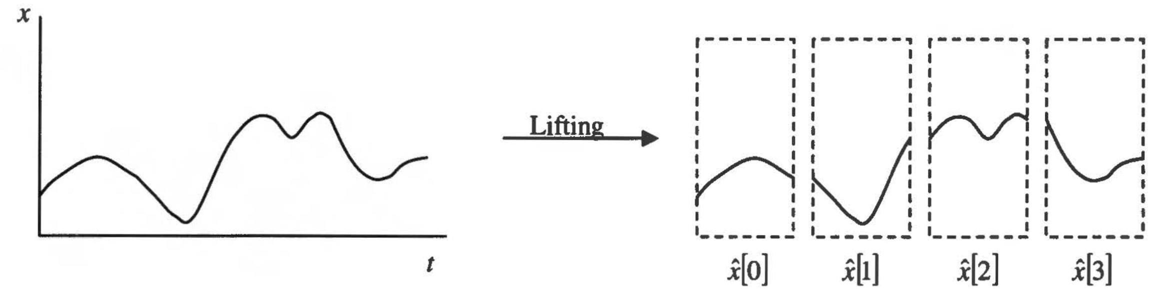


Figure A.2 The lifting operation

$$\|\hat{x}\|_{l_2}^2 = \sum_{k=0}^{\infty} \|\hat{x}[k]\|_{L_2[0,h)}^2 = \sum_{k=0}^{\infty} \left(\left(\int_0^h |x(kh+\tau)|^2 d\tau \right)^{\frac{1}{2}} \right)^2 = \sum_{k=0}^{\infty} \int_0^h |x(kh+\tau)|^2 d\tau = \int_0^{\infty} |x(t)|^2 dt = \|x\|_{L_2}^2 . \tag{A.44}$$

Besides lifting signals, the lifting operation can be applied to systems. Consider an ordinary continuous-time LTI state space system. The state of this system evolves from one sampling time to the next according to:

$$x(kh+h) = e^{Ah} x(kh) + \int_0^h e^{A(h-\tau)} Bu(kh+\sigma) d\sigma \tag{A.45}$$

and the output y of the plant between sampling instants is described by:

$$y(kh+\tau) = Ce^{A\tau} x(kh) + \int_0^{\tau} Ce^{A(\tau-\sigma)} Bu(kh+\sigma) d\sigma + Du(kh+\tau), \quad \tau \in [0,h). \tag{A.46}$$

These equations were obtained through integration of the continuous-time state space equations, see also [8]. The lifted system is described by the discrete-time state space equations:

$$\begin{aligned} \hat{x}[k+1] &= \hat{A}\hat{x}[k] + \hat{B}\hat{u}[k] \\ \hat{y}[k] &= \hat{C}\hat{x}[k] + \hat{D}\hat{u}[k] \end{aligned} \tag{A.47}$$

where $\hat{x}[k] = x(kh) \in \mathbb{R}^n$, $\hat{u}[k] \in L_2[0,h)$ and $\hat{y}[k] \in L_2[0,h)$ (the dimensions of the input and output signals are omitted). The operators in (A.47) are defined as:

$$\hat{A} : \mathbb{R}^n \to \mathbb{R}^n, \qquad \hat{A} = e^{Ah} \tag{A.48}$$

$$\hat{B} : L_2[0,h) \to \mathbb{R}^n, \qquad \hat{B}\hat{u}[k] = \int_0^h e^{A(h-\sigma)} Bu(kh+\sigma) d\sigma \tag{A.49}$$

$$\hat{C} : \mathbb{R}^n \to L_2[0,h), \qquad \{\hat{C}\hat{x}[k]\}(\tau) = Ce^{A\tau} \hat{x}[k] \tag{A.50}$$

$$\hat{D} : L_2[0,h) \to L_2[0,h), \qquad \{\hat{D}\hat{u}[k]\}(\tau) = \int_0^{\tau} Ce^{A(\tau-\sigma)} Bu(kh+\sigma) d\sigma + Du(kh+\tau) \tag{A.51}$$

with $\tau \in [0,h)$. Technically, (A.47) is an operator valued state space equation, which defines an infinite-dimensional discrete-time system, as $L_2[0,h)$ is an infinite-dimensional Hilbert space. The lifted system has the same L_2 induced norm, because the l_2 norm of the input and output signals of the lifted system are equal to the L_2 norms of the continuous-time system. The lifting operation can also be used to derive a norm-equivalent infinite-dimensional discrete-time system for a mixed continuous-time/discrete-time system or even multirate control systems.

The second step in the lifting approach to optimal sampled-data control is a transformation from an infinite-dimensional problem to a finite-dimensional one. Note that operator $\hat{A}$ has already an n-dimensional matrix representation. Operators $\hat{B}$ and $\hat{C}$ are infinite-dimensional operators, but they have finite rank (maximal n). As $L_2[0,h)$ is a Hilbert space, a fairly general result allows the derivation of a norm-equivalent finite-dimensional operator.
The definition of the norm and inner product associated with Hilbert space $L_2[0,h)$ are:

- inner product: $\langle x, y\rangle = \int_0^h x^T(\sigma)y(\sigma)d\sigma$; (A.52)
- norm: $\|x\|_{L_2[0,h)} = \left(\int_0^h |x(\sigma)|^2 d\sigma\right)^{\frac{1}{2}} = \left(\int_0^h x^T(\sigma)x(\sigma)dt\right)^{\frac{1}{2}} = \sqrt{\langle x, x\rangle}$. (A.53)

Let Ψ be an operator $\Psi : \mathbb{R}^n \rightarrow L_2[0,h)$. The adjoint operator is the unique operator $\Psi^* : L_2[0,h) \rightarrow \mathbb{R}^n$ satisfying $\langle \Psi x, y\rangle = \langle x, \Psi^* y\rangle$, where $x \in \mathbb{R}^n$ and $y \in L_2[0,h)$ (also a Hilbert space with the Euclidian norm and vector dot product as inner product). The composite operator $\Psi^*\Psi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ (see figure A.3) has an $n \times n$ matrix representation $M^T M$. M is a norm-equivalent finite-dimensional operator for Ψ as:

$$\|\Psi x\|^2_{L_2[0,h)} = \langle \Psi x, \Psi x\rangle = \langle x, \Psi^*\Psi x\rangle = \langle x, M^T M x\rangle = x^T M^T M x = \langle Mx, Mx\rangle = \|Mx\|^2 . \qquad \text{(A.54)}$$

Once an expression for the matrix $M^T M = \Psi^*\Psi$ is obtained, a (non-unique) matrix M can be obtained through a Cholesky-like factorisation (if the matrix $M^T M$ has full rank, the Cholesky factorisation can be used, otherwise the singular value decomposition (SVD) can be applied to obtain a suited factorisation).
A dual lemma exists for operators $L_2[0,h) \rightarrow \mathbb{R}^n$. The operator $\hat{D}$ (A.51) imposes a difficulty because it has infite rank. However, in sampled-data H_2 optimal control, the response to a delta pulse $u(t) = u_0\delta(t)$ or $u[k] = u_0\delta[k]$ is studied, where $u_0 \in \mathbb{R}$. In this case, the operator $\hat{D}$ can be regarded as an operator from $\mathbb{R}$ to $L_2[0,h)$, and the same trick can be applied to derive a norm-equivalent finite-dimensional matrix operator. In sampled-data H_∞ control, the problem with the infinite rank operator $\hat{D}$ is circumvented through a loop-shifting transformation which preserves norm and stability [21].

Solving the optimal control problem for the norm-equivalent finite-dimensional discrete-time system is the third step in the lifting approach. In section A.5 and A.6, a sampled-data H_2 norm is computed for two kinds of digital motor control set-ups. The third step in these computations is solving a discrete-time Lyapunov equation (the factorisations in step 2 of the lifting approach are not necessary in these computations).

$$\begin{tikzcd}
\mathbb{R} \arrow[r, shift left=2, "\Psi"] \arrow[loop right, "\Psi^*\Psi"] & L_2[0, h) \arrow[l, shift left=2, "\Psi^*"]
\end{tikzcd}$$

Figure A.3 Commutative diagram

A.5 Computing the integral quadratic tracking error for the synchronous case

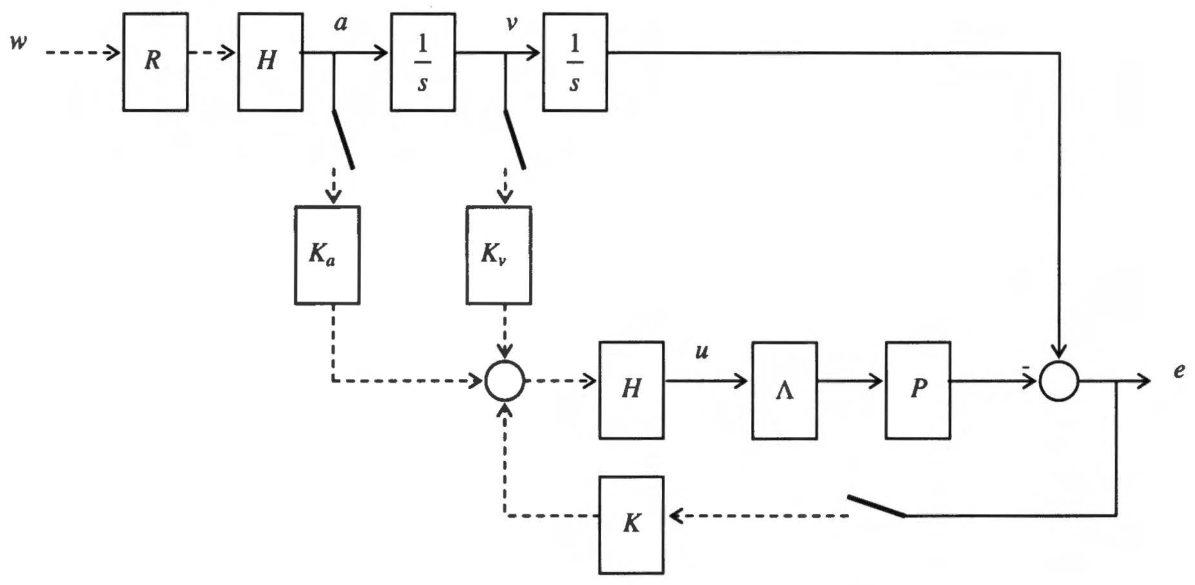


Figure A.4 Block diagram for synchronous case

The integral quadratic tracking error with respect to tracking a second order position profile is equal to the simple H_2 measure of the sampled-data system depicted in the figure above, where the impulse response of discrete-time system R represents the desired acceleration profile. The acceleration profile is constant between two sampling instants. The second order position profile is obtained through applying a continuous-time integration operation ($1/s$) twice. K is a discrete-time feedback controller and P is a continuous-time model for a position controlled motor. Λ represents the computation delay $0 \leq \lambda < h$, where h is the sampling time. K_a and K_v are the acceleration and velocity feedforward constants, and H denotes the zero order hold operation. For the computation of the simple H_2 measure, the following assumptions are made:

- the discrete-time system R is stable;
- P is strictly proper and contains an integrator mode, $\frac{P'(s)}{s} = P(s)$;
- K contains an integrator mode such that the sampled-data system is asymptotically ramp tracking;
- and K stabilises the loop in figure A.4.

The derivation of the simple H_2 measure for the sampled-data system is based on the lifting approach, which was discussed in appendix A.4. First, the sampled-data system is lifted to a norm-equivalent discrete-time system. Infinite-dimensional operators are converted to norm-equivalent finite-dimensional operators. The calculations heavily depend on the results of appendix A.1. After a norm-equivalent finite-dimensional discrete-time system representation has been obtained, a Lyapunov equation is solved to compute the H_2 norm of the discrete-time system, which is equal to simple H_2 measure for the sampled data system.

The computation of a (positive definite) solution to the discrete-time Lyapunov equation requires that all the eigenvalues of the A matrix involved lie within the unit circle. Integrator modes are not allowed, so the block diagram in figure A.4 is reconfigured as in figure A.5. ε is a small negative perturbation and $\frac{P'(s)}{s} = P(s)$. Note that one integrator is shifted into the loop.

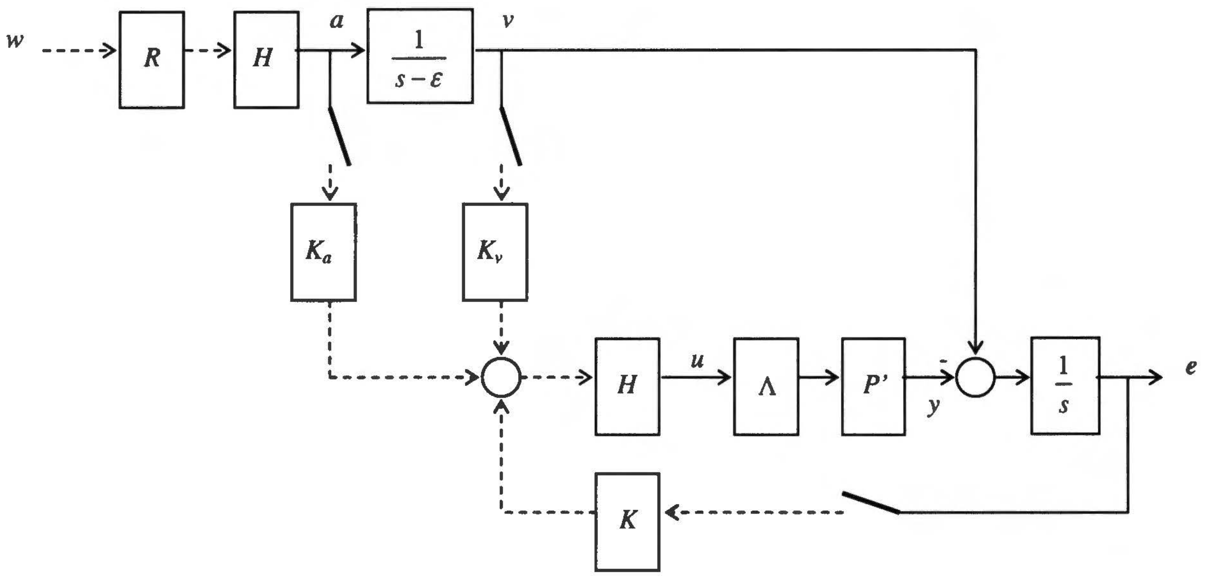


Figure A.5 Reconfigured block diagram for synchronous case

Let (A_R, B_R, C_R, D_R) be a state-space representation of R and denote the internal state of R by x_R. P' has state-space description $(A_{P'}, B_{P'}, C_{P'}, D_{P'})$ and internal state $x_{P'}$. K has system matrices (A_K, B_K, C_K, D_K) and state x_K.
The piecewise constant acceleration a is described by:

$$a(kh) = C_R x_R(kh) + D_R w(kh) \tag{A.55}$$
$$x_R(kh) = A_R x_R(kh) + B_R w(kh). \tag{A.56}$$

The evolution of the continuous-time velocity signal is given by:
(obtained through integration of state-space differential equation (see also [8]))

$$v(kh+\tau) = e^{\varepsilon\tau} v(kh) + \int_0^\tau e^{\varepsilon(\tau-\sigma)} d\sigma \cdot a(kh), \quad \tau \in [0, h). \tag{A.57}$$

The discrete-time actuator signal is characterised by:

$$u(kh) = K_v v(kh) + K_a a(kh) + D_K e(kh) + C_K x_K(kh). \tag{A.58}$$

The signal y depends on u in the following way:

$$y(kh+\tau) = \begin{cases} C_{P'} e^{A_{P'}\tau} x_{P'}(kh) + \left(\int_0^\tau C_{P'} e^{A_{P'}(\tau-\sigma)} B_{P'} d\sigma + D_{P'} \right) u(kh-h), & \tau \le \lambda \\ C_{P'} e^{A_{P'}\tau} x_{P'}(kh) + \int_0^\lambda C_{P'} e^{A_{P'}(\lambda-\sigma)} B_{P'} d\sigma \cdot u(kh-h) + \left(\int_\lambda^\tau C_{P'} e^{A_{P'}(\tau-\sigma)} B_{P'} d\sigma + D_{P'} \right) u(kh), & \tau > \lambda \end{cases}$$

. (A.59)

And the error e evolves according to:

$$\begin{aligned}
e(kh+\tau) &= e(kh) + \int_0^\tau \big(v(kh+\sigma) - y(kh+\sigma)\big)d\sigma \\
&= e(kh) + \int_0^\tau e^{\varepsilon\sigma}d\sigma \cdot v(kh) + \int_0^\tau\int_0^\sigma e^{\varepsilon(\sigma-\rho)}d\rho d\sigma \cdot a(kh) - \int_0^\tau C_{P'}e^{A_{P'}\sigma}d\sigma \cdot x_{P'}(kh) \\
&\quad - \begin{cases} \left(\int_0^\tau\int_0^\sigma C_{P'}e^{A_{P'}(\sigma-\rho)}B_{P'}d\rho d\sigma + \int_0^\tau D_{P'}d\sigma\right)u(kh-h), & \tau \le \lambda \\ \lambda\left(\int_0^\lambda C_{P'}e^{A_{P'}(\sigma-\rho)}B_{P'}d\sigma + D_{P'}\right)\cdot u(kh-h) + \left(\int_0^{\tau-\lambda}\int_0^\sigma C_{P'}e^{A_{P'}(\sigma-\rho)}B_{P'}d\rho d\sigma + \int_0^{\tau-\lambda} D_{P'}d\sigma\right)u(kh), & \tau > \lambda \end{cases}
\end{aligned}$$

. (A.60)

Define the combined state vector $\hat{x}$ as

$$\hat{x}(kh) = \begin{pmatrix} e(kh) \\ v(kh) \\ x_R(kh) \\ x_{P'}(kh) \\ x_K(kh) \\ u(kh-h) \end{pmatrix}, \tag{A.61}$$

then, by substituting (A.55) and (A.57) into (A.58), (A.58) into (A.59) and using (A.8) to write the integrals in terms of a matrix exponential, (A.60) can be rewritten as:

$$\begin{aligned}
e(kh+\tau) &= \begin{pmatrix}1 & 0 & 0\end{pmatrix}\exp\left(\begin{pmatrix}0 & 1 & 0 \\ 0 & \varepsilon & 1 \\ 0 & 0 & 0\end{pmatrix}\tau\right)\begin{pmatrix}1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_R & 0 & 0 & 0\end{pmatrix}\hat{x}(kh) \\
&\quad - \begin{cases} \begin{pmatrix}1 & 0 & 0\end{pmatrix}\exp\left(\begin{pmatrix}0 & C_{P'} & D_{P'} \\ 0 & A_{P'} & B_{P'} \\ 0 & 0 & 0\end{pmatrix}\tau\right)\begin{pmatrix}0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1\end{pmatrix}\hat{x}(kh), & \tau \le \lambda \\ \begin{pmatrix}1 & 0 & 0\end{pmatrix}\exp\left(\begin{pmatrix}0 & C_{P'} & D_{P'} \\ 0 & A_{P'} & B_{P'} \\ 0 & 0 & 0\end{pmatrix}(\tau-\lambda)\right)\begin{pmatrix}0 & 0 & 0 & 0 & 0 & D_{Pd}^{0\lambda} \\ 0 & 0 & 0 & e^{A_{P'}\lambda} & 0 & 0 \\ D_K & K_v & K_aC_R & 0 & C_K & 0\end{pmatrix}\hat{x}(kh), & \tau > \lambda \end{cases} \\
&\quad + \begin{pmatrix}1 & 0 & 0\end{pmatrix}\exp\left(\begin{pmatrix}0 & 1 & 0 \\ 0 & \varepsilon & 1 \\ 0 & 0 & 0\end{pmatrix}\tau\right)\begin{pmatrix}0 \\ 0 \\ D_R\end{pmatrix}w(kh) \\
&\quad - \begin{cases} 0, & \tau \le \lambda \\ \begin{pmatrix}1 & 0 & 0\end{pmatrix}\exp\left(\begin{pmatrix}0 & C_{P'} & D_{P'} \\ 0 & A_{P'} & B_{P'} \\ 0 & 0 & 0\end{pmatrix}(\tau-\lambda)\right)\begin{pmatrix}0 \\ 0 \\ K_aD_R\end{pmatrix}w(kh), & \tau > \lambda \end{cases}
\end{aligned}$$

(A.62)

where

$$D_{Pd}^{0\lambda} = \lambda\left(\int_0^{\lambda} C_{P'} e^{A_{P'}(\sigma-\rho)} B_{P'} d\sigma + D_{P'}\right). \tag{A.63}$$

Equation A.62 can further be reduced to:

$$e(kh+\tau) = \begin{cases} H_1 e^{F_1\tau} G_1 \hat{x}(kh) + H_3 e^{F_3\tau} G_3 w(kh), & \tau \le \lambda \\ H_2 e^{F_2(\tau-\lambda)} G_2 \hat{x}(kh) + H_4 e^{F_4(\tau-\lambda)} G_4 w(kh), & \tau > \lambda \end{cases} \tag{A.64}$$

where

$$H_1 = \begin{pmatrix}1 & 0 & 0 & -1 & 0 & 0\end{pmatrix},\ F_1 = \begin{pmatrix} 0 & 1 & 0 & & & \\ 0 & \varepsilon & 1 & & & \\ 0 & 0 & 0 & & & \\ & & & 0 & C_{P'} & D_{P'} \\ & & & 0 & A_{P'} & B_{P'} \\ & & & 0 & 0 & 0 \end{pmatrix},\ G_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_R & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.65}$$

$$H_2 = \left(\begin{pmatrix}1 & 0 & 0\end{pmatrix} \exp\left(\begin{pmatrix} 0 & 1 & 0 \\ 0 & \varepsilon & 1 \\ 0 & 0 & 0 \end{pmatrix} \lambda \right) \quad -1 \quad 0 \quad 0 \right),\ F_2 = \begin{pmatrix} 0 & 1 & 0 & & & \\ 0 & \varepsilon & 1 & & & \\ 0 & 0 & 0 & & & \\ & & & 0 & C_{P'} & D_{P'} \\ & & & 0 & A_{P'} & B_{P'} \\ & & & 0 & 0 & 0 \end{pmatrix},$$

$$G_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_R & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & D_{Pd}^{0\lambda} \\ 0 & 0 & 0 & e^{A_{P'}\lambda} & 0 & 0 \\ D_K & K_v & K_a C_R & 0 & C_K & 0 \end{pmatrix} \tag{A.66}$$

$$H_3 = \begin{pmatrix}1 & 0 & 0\end{pmatrix},\ F_3 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & \varepsilon & 1 \\ 0 & 0 & 0 \end{pmatrix},\ G_3 = \begin{pmatrix} 0 \\ 0 \\ D_R \end{pmatrix} \tag{A.67}$$

$$H_4 = H_2,\ F_4 = F_2,\ G_4 = \begin{pmatrix} 0 \\ 0 \\ D_R \\ 0 \\ 0 \\ K_a D_R \end{pmatrix}. \tag{A.68}$$

The state of *P*' evolves from one sampling time to next according to:

$$\begin{aligned} x_{P'}(kh+h) &= e^{A_{P'}h}x_{P'}(kh)+\int_0^{\lambda} e^{A_{P'}(\lambda-\sigma)}B_{P'}d\sigma\cdot u(kh-h)+\int_{\lambda}^{h} e^{A_{P'}(h-\sigma)}B_{P'}d\sigma\cdot u(kh) \\ &= e^{A_{P'}h}x_{P'}(kh)+B_{P'd}^{0\lambda}u(kh-h)+B_{P'd}^{\lambda h}\cdot u(kh) \end{aligned} \quad . \tag{A.69}$$

Note that (A.57) can be written as:

$$v(kh+\tau)=\begin{pmatrix}1 & 0\end{pmatrix}\exp\left(\begin{pmatrix}\varepsilon & C_R\\ 0 & 0\end{pmatrix}h\right)\begin{pmatrix}v(kh)\\ x_R(kh)\end{pmatrix}+\begin{pmatrix}1 & 0\end{pmatrix}\exp\left(\begin{pmatrix}\varepsilon & D_R\\ 0 & 0\end{pmatrix}h\right)\begin{pmatrix}0\\ 1\end{pmatrix}w(kh). \tag{A.70}$$

Combining equations A.55, A.56, A.58, A.60, A.69, A.70 and the state-space equations of K yields the discrete-time state equation:

$$\hat{x}(kh+h)=\hat{A}\hat{x}(kh)+\hat{B}w(kh) \tag{A.71}$$

with

$$\hat{A}=\left(\begin{array}{c:c:cccc} \multicolumn{6}{c}{H_2e^{F_2(h-\lambda)}G_2} \\ \hdashline 0 & \begin{pmatrix}1 & 0\end{pmatrix}\exp\left(\begin{pmatrix}\varepsilon & C_R\\ 0 & 0\end{pmatrix}h\right) & 0 & 0 & 0 & \\ \hdashline 0 & \begin{matrix}0 & A_R\end{matrix} & 0 & 0 & 0 & \\ \hdashline B_{P'd}^{\lambda h}D_K & B_{P'd}^{\lambda h}K_v \quad B_{P'd}^{\lambda h}K_aC_R & e^{A_{P'}h} & B_{P'd}^{\lambda h}C_K & B_{P'd}^{0\lambda} & \\ B_K & 0 \quad\quad 0 & 0 & A_K & 0 & \\ D_K & K_v \quad\quad K_aC_R & 0 & C_K & 0 & \end{array}\right) \tag{A.72}$$

$$\hat{B}=\begin{pmatrix} H_4e^{F_4(h-\lambda)}G_4 \\ \begin{pmatrix}1 & 0\end{pmatrix}\exp\left(\begin{pmatrix}\varepsilon & D_R\\ 0 & 0\end{pmatrix}h\right)\begin{pmatrix}0\\ 1\end{pmatrix} \\ B_R \\ B_{P'd}^{\lambda h}K_aD_R \\ 0 \\ K_aD_R \end{pmatrix} \tag{A.73}$$

where

$$B_{P'd}^{0\lambda}=\int_0^{\lambda} e^{A_{P'}(\lambda-\sigma)}B_{P'}d\sigma=\begin{pmatrix}I & 0\end{pmatrix}\exp\left(\begin{pmatrix}A_{P'} & B_{P'}\\ 0 & 0\end{pmatrix}\lambda\right)\begin{pmatrix}0\\ 1\end{pmatrix} \tag{A.74}$$

$$B_{P'd}^{\lambda h}=\int_{\lambda}^{h} e^{A_{P'}(h-\sigma)}B_{P'}d\sigma=\begin{pmatrix}I & 0\end{pmatrix}\exp\left(\begin{pmatrix}A_{P'} & B_{P'}\\ 0 & 0\end{pmatrix}(h-\lambda)\right)\begin{pmatrix}0\\ 1\end{pmatrix}. \tag{A.75}$$

(A.71) denotes the finite-dimensional part of the state space equations of the norm-equivalent discrete-time system. Calculations involving the infinite-dimensional part of the state space equations follow now.

Define the operator $C:\mathbb{R}^n\rightarrow L_2[0,h)$,

$$C\hat{x} = \begin{cases} H_1 e^{F_1\tau} G_1 \hat{x}(kh), & \tau \le \lambda \\ H_2 e^{F_2(\tau-\lambda)} G_2 \hat{x}(kh), & \tau > \lambda \end{cases} \quad \text{for } \tau \in [0,h) \tag{A.76}$$

and the adjoint operator $C^* : L_2[0, h) \to \mathbb{R}^n$,

$$C^* e = \int_0^\lambda G_1^T e^{F_1^T \tau} H_1^T e(\tau) d\tau + \int_\lambda^h G_2^T e^{F_2^T \tau} H_2^T e(\tau) d\tau . \tag{A.77}$$

The operator $C^*C : \mathbb{R}^n \to \mathbb{R}^n$ has an n dimensional matrix representation:

$$\hat{C}^T \hat{C} = \int_0^\lambda G_1^T e^{F_1^T \tau} H_1^T H_1 e^{F_1\tau} G_1 d\tau + \int_\lambda^h G_2^T e^{F_2^T \tau} H_2^T H_2 e^{F_2\tau} G_2 d\tau . \tag{A.78}$$

This matrix can be computed as two matrix exponentials. Let

$$\begin{pmatrix} P_{11} & P_{12} \\ 0 & P_{22} \end{pmatrix} = \exp\left(\begin{pmatrix} -F_2^T & H_2^T H_2 \\ 0 & F_2 \end{pmatrix} (h-\lambda) \right) \text{ and } \begin{pmatrix} Q_{11} & Q_{12} \\ 0 & Q_{22} \end{pmatrix} = \exp\left(\begin{pmatrix} -F_1^T & H_1^T H_1 \\ 0 & F_1 \end{pmatrix} \lambda \right) \tag{A.79}$$

then by using (A.2):

$$\hat{C}^T \hat{C} = G_1^T Q_{22}^T G_{12} G_1 + G_2^T P_{22}^T P_{12} G_2 . \tag{A.80}$$

Similarly, define the operator $D : \mathbb{R} \to L_2[0, h)$,

$$Dw = \begin{cases} H_3 e^{F_3\tau} G_3 w(kh), & \tau \le \lambda \\ H_4 e^{F_4(\tau-\lambda)} G_4 w(kh), & \tau > \lambda \end{cases} \tag{A.81}$$

and the adjoint operator:

$$D^* w = \int_0^\lambda G_3^T e^{F_3^T \tau} H_3^T w(\tau) d\tau + \int_\lambda^h G_4^T e^{F_4^T \tau} H_4^T w(\tau) d\tau . \tag{A.82}$$

The operator $D^*D : \mathbb{R} \to \mathbb{R}$ can be represented by a scalar:

$$\hat{D}^T \hat{D} = \int_0^\lambda G_3^T e^{F_{31}^T \tau} H_3^T H_3 e^{F_3\tau} G_3 d\tau + \int_\lambda^h G_4^T e^{F_4^T \tau} H_4^T H_4 e^{F_4\tau} G_4 d\tau \tag{A.83}$$

which can be computed as:

$$\hat{D}^T \hat{D} = G_3^T R_{22}^T R_{12} G_1 + G_4^T S_{22}^T S_{12} G_4 \tag{A.84}$$

where:

$$\begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} = \exp\left(\begin{pmatrix} -F_3^T & H_3^T H_3 \\ 0 & F_3 \end{pmatrix}\lambda\right) \text{ and } \begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix} = \exp\left(\begin{pmatrix} -F_4^T & H_4^T H_4 \\ 0 & F_4 \end{pmatrix}(h-\lambda)\right). \tag{A.85}$$

Consider the L_2 norm of e with support on $[0, h)$; it can be expressed in terms of the inproduct on $L_2[0, h)$ because $L_2[0, h)$ is a Hilbert space.

$$\|e\|^2_{L_2[0,h)} = \left(\int_0^h |e(\tau)|^2 d\tau\right)^2 = \langle C\hat{x}, C\hat{x}\rangle = \langle \hat{x}, C^* C\hat{x}\rangle = \langle \hat{x}, \hat{C}^T \hat{C}\hat{x}\rangle = \hat{x}^T \hat{C}^T \hat{C}\hat{x} \tag{A.86}$$

Now, suppose that a Kronecker delta pulse is applied to input w. The $\hat{D}$ term only has a contribution during the first sampling period, where $w = 1$. The remaining energy is due to the $\hat{C}$ term, which contributes from the second sampling period on, when zero initial state is assumed. The energy of the impulse response can now be expressed as:

$$\begin{aligned} \|e\|^2_{L_2} &= \hat{D}^T\hat{D} + \sum_{k=1}^{\infty} \|e\|^2_{L_2[0,h)} = \hat{D}^T\hat{D} + \sum_{k=1}^{\infty} \hat{x}^T(kh)\hat{C}^T\hat{C}\hat{x}(kh) = \hat{D}^T\hat{D} + \hat{B}^T \sum_{k=1}^{\infty} \left(\hat{A}^T\right)^k \hat{C}^T\hat{C}\hat{A}^k\hat{B} \\ &= \hat{D}^T\hat{D} + \hat{B}^T \sum_{k=0}^{\infty} \left(\hat{A}^T\right)^k \hat{C}^T\hat{C}\hat{A}^k\hat{B} = \hat{D}^T\hat{D} + \hat{B}^T M\hat{B} \end{aligned} . \tag{A.87}$$

This is the well known characterisation of the H_2 norm of a discrete-time system. M is the observability gramian:

$$M = \sum_{k=0}^{\infty} \left(\hat{A}^T\right)^k \hat{C}^T\hat{C}\hat{A}^k \tag{A.88}$$

which can be computed through:

$$\begin{aligned} M &= \hat{C}^T\hat{C} + \sum_{k=1}^{\infty} \left(\hat{A}^T\right)^k \hat{C}^T\hat{C}\hat{A} = \hat{C}^T\hat{C} + \hat{A}^T\left(\sum_{k=0}^{\infty} \left(\hat{A}^T\right)^k \hat{C}^T\hat{C}\hat{A}\right)\hat{A} \\ &= \hat{C}^T\hat{C} + \hat{A}^T M\hat{A} \end{aligned} . \tag{A.89}$$

This resulting equation is a discrete-time Lyapunov equation. If $\hat{A}$ has all eigenvalues inside the unit disk, the sum (A.88) converges and M is the unique positive definite solution to the discrete-time Lyapunov equation:

$$\hat{A}^T M\hat{A} + \hat{C}^T\hat{C} = M . \tag{A.90}$$

This ends the derivation of the simple H_2 measure for the synchronous case. The computation is quite straightforward. It only requires the computation of some matrix exponentials and a solution to a Lyapunov equation. Both operations are standard operations in MATLAB and the Control Toolbox.

A.6 Computing the average integral quadratic tracking error for the asynchronous case

The integral quadratic tracking error for the asynchronous case is equal to the generalised H_2 measure of the sampled-data system in figure A.6. The derivation of the generalised H_2 measure is to some extended quite similar to the derivation of the simple H_2 measure for the synchronous case. There are however some differences:

- The acceleration profile is generated by a continuous-time system as opposed to a discrete-time system in the synchronous case. The piecewise constant acceleration signal has to be approximated with an impulse response of a rational continuous-time transfer function (the Padé approximation might be applied).
- the generalised H_2 measure is calculated as the integral of the squared output after applying a Dirac pulse at $t=\eta$, $\eta\in[0,h)$, averaged over the interval [0, h). Recall that the simple H_2 measure for the synchronous case was calculated after a Kronecker delta pulse was applied. The averaging action causes an extra integration in the derivation.

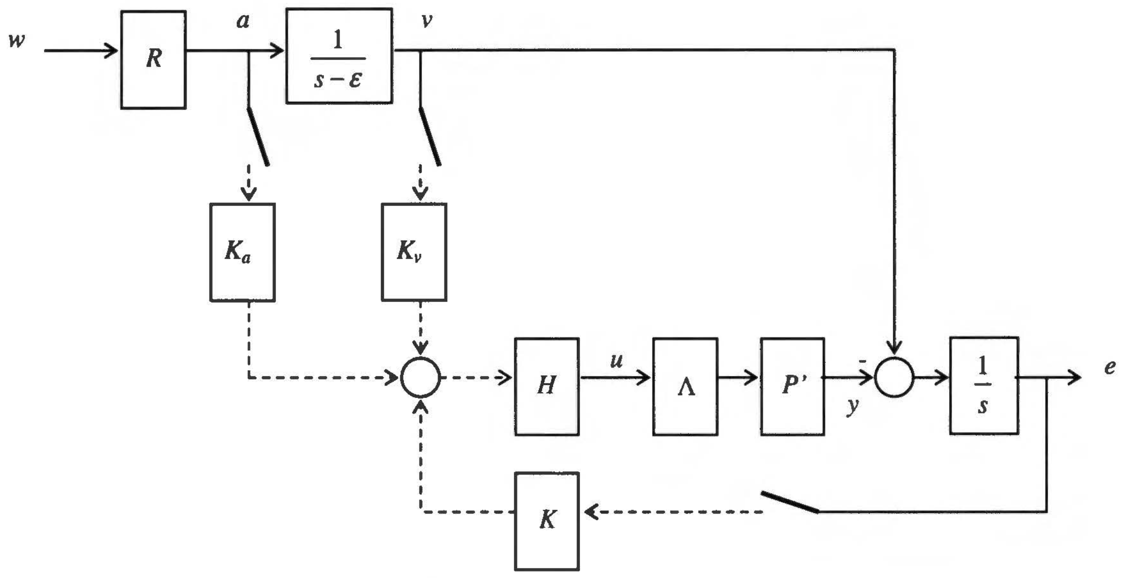


Figure A.6 Block diagram for asynchronous case

The assumptions for the asynchronous case are:

- the continuous-time system R is strictly proper and has all its poles in the open left half plane;
- the continuous-time plant P is strictly proper and contains an integrator mode, $\frac{P'(s)}{s}=P(s)$;
- K contains an integrator mode such that the sampled-data system is asymptotically ramp tracking;
- the computation delay λ is smaller than the sampling time h;
- and the discrete-time feedback controller K stabilises the feedback loop.

Again, the first step in the derivation is to gather the equations describing the combined discrete-time dynamics of the hybrid system.
The signal a evolves as:

$$a(kh+\tau)=C_R e^{A_R\tau}x_R(kh)+\int_0^\tau C_R e^{A_R(\tau-\sigma)}B_R w(kh+\sigma)d\sigma \tag{A.91}$$

$$x_R(kh+h)=e^{A_R h}x_R(kh)+\int_0^h e^{A_R(h-\sigma)}B_R w(kh+\sigma)d\sigma \tag{A.92}$$

(of course $\tau\in[0,h)$).
The continuous-time velocity signal v is related to a by:

$$v(kh+\tau)=e^{\varepsilon\tau}v(kh)+\int_0^\tau e^{\varepsilon(\tau-\sigma)}a(kh+\sigma)d\sigma\,. \tag{A.93}$$

The undelayed actuator signal is:

$$u(kh)=K_v v(kh)+K_a C_R x_R(kh)+D_K e(kh)+C_K x_K(kh). \tag{A.94}$$

P' generates

$$y(kh+\tau)=\begin{cases} C_{P'}e^{A_{P'}\tau}x_{P'}(kh)+\left(\int_0^\tau C_{P'}e^{A_{P'}(\tau-\sigma)}B_{P'}d\sigma+D_{P'}\right)u(kh-h), & \tau\le\lambda \\ C_{P'}e^{A_{P'}\tau}x_{P'}(kh)+\int_0^\lambda C_{P'}e^{A_{P'}(\lambda-\sigma)}B_{P'}d\sigma\cdot u(kh-h)+\left(\int_\lambda^\tau C_{P'}e^{A_{P'}(\tau-\sigma)}B_{P'}d\sigma+D_{P'}\right)u(kh), & \tau>\lambda\end{cases} \tag{A.95}$$

with

$$\begin{aligned} x_{P'}(kh+h)&=e^{A_{P'}h}x_{P'}(kh)+\int_0^\lambda e^{A_{P'}(\lambda-\sigma)}B_{P'}d\sigma\cdot u(kh-h)+\int_\lambda^h e^{A_{P'}(h-\sigma)}B_{P'}d\sigma\cdot u(kh) \\ &=e^{A_{P'}h}x_{P'}(kh)+B_{P'd}^{0\lambda}u(kh-h)+B_{P'd}^{\lambda h}\cdot u(kh)\end{aligned} \tag{A.96}$$

which results in a tracking error:

$$\begin{aligned} e(kh+\tau)&=e(kh)+\int_0^\tau (v(kh+\sigma)-y(kh+\sigma))d\sigma= \\ &e(kh)+\int_0^\tau e^{\varepsilon\sigma}d\sigma\cdot v(kh)+\int_0^\tau\int_0^\sigma e^{\varepsilon(\sigma-\rho)}C_R e^{A_R\rho}d\rho d\sigma\cdot x_R(kh)+\int_0^\tau\int_0^\sigma\int_0^\rho e^{\varepsilon(\sigma-\rho)}C_R e^{A_R(\rho-\vartheta)}B_R w(kh+\vartheta)d\vartheta d\rho d\sigma \\ &-\int_0^\tau C_{P'}e^{A_{P'}\sigma}d\sigma\cdot x_{P'}(kh) \\ &-\begin{cases}\left(\int_0^\tau\int_0^\sigma C_{P'}e^{A_{P'}(\sigma-\rho)}B_{P'}d\rho d\sigma+\int_0^\tau D_{P'}d\sigma\right)u(kh-h), & \tau\le\lambda \\ \lambda\left(\int_0^\lambda C_{P'}e^{A_{P'}(\sigma-\rho)}B_{P'}d\rho d\sigma+D_{P'}\right)u(kh-h)+\left(\int_0^{\tau-\lambda}\int_0^\sigma C_{P'}e^{A_{P'}(\sigma-\rho)}B_{P'}d\rho d\sigma+\int_0^{\tau-\lambda}D_{P'}d\sigma\right)u(kh), & \tau>\lambda\end{cases} \\ &. \end{aligned} \tag{A.97}$$

This equation can be written in terms of matrix exponentials as:

$$e(kh+\tau)=\int_0^\tau\int_0^\sigma\int_0^\rho e^{\varepsilon(\sigma-\rho)}C_R e^{A_R(\rho-\vartheta)}B_R w(kh+\vartheta)d\vartheta d\rho d\sigma+\begin{cases}H_1 e^{F_1\tau}G_1\hat{x}(kh), & \tau\le\lambda \\ H_2 e^{F_2(\tau-\lambda)}G_2\hat{x}(kh), & \tau>\lambda\end{cases} \tag{A.98}$$

where

$$H_1 = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 \end{pmatrix},\quad F_1 = \begin{pmatrix} 0 & 1 & 0 & & & \\ 0 & \varepsilon & C_R & & & \\ 0 & 0 & A_R & & & \\ & & & 0 & C_{P'} & D_{P'} \\ & & & 0 & A_{P'} & B_{P'} \\ & & & 0 & 0 & 0 \end{pmatrix},\quad G_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.99}$$

$$H_2 = \left(\begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \exp\left(\begin{pmatrix} 0 & 1 & 0 \\ 0 & \varepsilon & C_R \\ 0 & 0 & A_R \end{pmatrix} \lambda \right) \quad -1 \quad 0 \quad 0 \right),\quad F_2 = \begin{pmatrix} 0 & 1 & 0 & & & \\ 0 & \varepsilon & C_R & & & \\ 0 & 0 & A_R & & & \\ & & & 0 & C_{P'} & D_{P'} \\ & & & 0 & A_{P'} & B_{P'} \\ & & & 0 & 0 & 0 \end{pmatrix},$$

$$G_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & D_{Pd}^{0\lambda} \\ 0 & 0 & 0 & e^{A_{P'}\lambda} & 0 & 0 \\ D_K & K_v & K_a C_R & 0 & C_K & 0 \end{pmatrix} \tag{A.100}$$

and $D_{Pd}^{0\lambda}$ is given by (A.63). The combined discrete-time state vector $\hat{x}(kh)$ is again composed as in (A.61).

Combining equations A.91, A.92, A.93, A.94, A.96, A.97 and the state-space equations of K, yields the discrete-time combined state equation:

$$\hat{x}(kh+h) = \hat{A}\hat{x}(kh) + \begin{pmatrix} \int_0^h \int_0^\sigma \int_0^\rho e^{\varepsilon(\sigma-\rho)} C_R e^{A_R(\rho-\vartheta)} B_R w(kh+\vartheta)\, d\vartheta\, d\rho\, d\sigma \\ \int_0^h \int_0^\sigma e^{\varepsilon(h-\sigma)} C_R e^{A_R(\sigma-\rho)} B_R w(kh+\rho)\, d\rho\, d\sigma \\ \int_0^h e^{\varepsilon(h-\sigma)} B_R w(kh+\sigma)\, d\sigma \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{A.101}$$

with

$$\hat{A} = \left(\begin{array}{cccccc} \multicolumn{6}{c}{H_2 e^{F_2(h-\lambda)} G_2} \\ \hline \begin{array}{c} 0 \\ 0 \end{array} & \multicolumn{2}{|c|}{\exp\left(\begin{pmatrix} \varepsilon & C_R \\ 0 & A_R \end{pmatrix} h \right)} & \begin{array}{c} 0 \\ 0 \end{array} & \begin{array}{c} 0 \\ 0 \end{array} & \begin{array}{c} 0 \\ 0 \end{array} \\ \hline B_{P'd}^{\lambda h} D_K & B_{P'd}^{\lambda h} K_v & B_{P'd}^{\lambda h} K_a C_R & e^{A_{P'}h} & B_{P'd}^{\lambda h} C_K & B_{P'd}^{0\lambda} \\ B_K & 0 & 0 & 0 & A_K & 0 \\ D_K & K_v & K_a C_R & 0 & C_K & 0 \end{array} \right). \tag{A.102}$$

Note that the $\hat{A}$ matrix for the synchronous case and the asynchronous case are equal if the discrete-time system $R(z)$ equals the ZOH transform of the continuous-time system $R(s)$. The real difference is in the effect of w on the state. The computation of the finite dimensional matrix $\hat{C}^T\hat{C}$ is similar to the synchronous case.

Let $w(t)=w\delta(t-\eta)$, where δ represents the Dirac pulse, and let

$$D_\delta(\eta)w=\int_0^\tau\int_0^\sigma\int_0^\rho e^{\varepsilon(\sigma-\rho)}C_R e^{A_R(\rho-\vartheta)}B_R w\delta(\vartheta-\eta)d\vartheta d\rho d\sigma=\begin{cases}0, & \tau\le\eta\\ \int_\eta^\tau\int_0^\sigma e^{\varepsilon(\sigma-\rho)}C_R e^{A_R(\rho-\eta)}B_R w d\rho d\sigma, & \tau>\eta\end{cases}$$

$$=\begin{cases}0, & \tau\le\eta\\ \begin{pmatrix}1&0&0\end{pmatrix}\exp\left(\begin{pmatrix}0&1&0\\0&\varepsilon&C_R\\0&0&A_R\end{pmatrix}(\tau-\eta)\right)\begin{pmatrix}0\\0\\B_R\end{pmatrix}w, & \tau>\eta\end{cases}\tag{A.103}$$

and

$$B_\delta(\eta)w=\begin{pmatrix}\int_0^h\int_0^\sigma\int_0^\rho e^{\varepsilon(\sigma-\rho)}C_R e^{A_R(\rho-\vartheta)}B_R w\delta(\vartheta-\eta)d\vartheta d\rho d\sigma\\ \int_0^h\int_0^\sigma e^{\varepsilon(h-\sigma)}C_R e^{A_R(\sigma-\rho)}B_R w(\rho-\eta)d\rho d\sigma\\ \int_0^h e^{\varepsilon(h-\sigma)}B_R w(\sigma-\eta)d\sigma\\0\\0\\0\end{pmatrix}=\begin{pmatrix}\int_0^h\int_0^\sigma e^{\varepsilon(\sigma-\rho)}C_R e^{A_R(\rho-\eta)}B_R w\delta d\rho d\sigma\\ \int_0^h e^{\varepsilon(h-\sigma)}C_R e^{A_R(\sigma-\eta)}B_R w d\sigma\\ e^{\varepsilon(h-\eta)}B_R w\\0\\0\\0\end{pmatrix}$$

$$=\begin{pmatrix}1&0&0\\0&1&0\\0&0&I\\0&0&0\\0&0&0\\0&0&0\end{pmatrix}\exp\left(\begin{pmatrix}0&1&0\\0&\varepsilon&C_R\\0&0&A_R\end{pmatrix}(h-\eta)\right)\begin{pmatrix}0\\0\\B_R\end{pmatrix}w\tag{A.104}$$

.

These two operators describe the effect of a Dirac pulse on the output and the combined discrete-time state. Let

$$A_x=\begin{pmatrix}0&1&0\\0&\varepsilon&C_R\\0&0&A_R\end{pmatrix}.\tag{A.105}$$

The $D_\delta(\eta)$ operator defines a mapping from $w\in\mathbb{R}$ to $e\in L_2[0,h)$. The adjoint operator is given by:

$$D_\delta^*(\eta)e = \int_\eta^h \begin{pmatrix} 0 & 0 & B_R^T \end{pmatrix} e^{A_x^T(\tau-\eta)} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} e(\tau)d\tau . \tag{A.106}$$

Operator $D_\delta^* D_\delta(\eta)$ has scalar representation:

$$\int_\eta^h \begin{pmatrix} 0 & 0 & B_R^T \end{pmatrix} e^{A_x^T(\tau-\eta)} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} e^{A_x(\tau-\eta)} \begin{pmatrix} 0 \\ 0 \\ B_R \end{pmatrix} d\tau . \tag{A.107}$$

The Dirac delta pulse is applied at time $t=\eta$, $\eta \in [0,h)$ to input w in figure A.6. The operator $D_\delta^* D_\delta(\eta)$ describes the $L_2[0,\, h)$ norm of output e during the first sampling period. The cumulative integral quadratic error over the remaining sampling intervals is given by:

$$\sum_{k=1}^{\infty} \|e\|^2_{L_2[0,h)} = \sum_{k=1}^{\infty} \hat{x}^T(kh)\hat{C}^T\hat{C}\hat{x}(kh) = B_\delta^T(\eta)\sum_{k=1}^{\infty}\left(\hat{A}^T\right)^k \hat{C}^T\hat{C}\hat{A}^k B_\delta(\eta) = B_\delta^T(\eta)MB_\delta(\eta) \tag{A.108}$$

where M is the positive definite solution to the discrete-time Lyapunov equation:

$$\hat{A}^T M\hat{A} + \hat{C}^T\hat{C} = M . \tag{A.109}$$

The (squared) generalised H_2 norm is now characterised as:

$$J^2 = \frac{1}{h}\int_0^h \left(D_\delta^T D_\delta(\eta) + B_\delta^T(\eta)MB_\delta(\eta)\right)d\eta . \tag{A.110}$$

It is the average of the energy of the response to $w(t) = \delta(t-\eta)$, where the average is taken over $\eta \in [0,h)$.The first part of integral (A.110) is computed as:

$$\begin{aligned} \frac{1}{h}\int_0^h D_\delta^T D_\delta(\eta)d\eta &= \frac{1}{h}\int_0^h\int_\eta^h \begin{pmatrix} 0 & 0 & B_R^T \end{pmatrix} e^{A_x^T(\tau-\eta)} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} e^{A_x(\tau-\eta)} \begin{pmatrix} 0 \\ 0 \\ B_R \end{pmatrix} d\tau d\eta \\ &= \frac{1}{h}\begin{pmatrix} 0 & 0 & B_R^T \end{pmatrix} R_{33}^T R_{13} \begin{pmatrix} 0 \\ 0 \\ B_R \end{pmatrix} \end{aligned} \tag{A.111}$$

where

$$\begin{pmatrix} R_{11} & R_{12} & R_{13} \\ & R_{22} & R_{23} \\ & & R_{33} \end{pmatrix} = \exp\left(\left(\begin{array}{c:c:c} -A_x^T & I & 0 \\ \hdashline & -A_x^T & \begin{matrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \\ \hdashline & & A_x \end{array}\right)h\right) . \tag{A.112}$$

This follows from substitution of variables in the integral expression and using (A.8). The second part of integral (A.110) is computed as:

$$\begin{aligned}\frac{1}{h}\int_0^h B_\delta^T(\eta)MB_\delta(\eta)d\eta &= \frac{1}{h}\int_0^h \begin{pmatrix}0 & 0 & B_R^T\end{pmatrix}e^{A_x^T(h-\eta)}O^T MOe^{A_x(h-\eta)}\begin{pmatrix}0\\0\\B_R\end{pmatrix}d\eta \\ &= \frac{1}{h}\begin{pmatrix}0 & 0 & B_R^T\end{pmatrix}T_{22}^T T_{12}\begin{pmatrix}0\\0\\B_R\end{pmatrix}\end{aligned} \qquad (A.113)$$

where

$$\begin{pmatrix}T_{11} & T_{12}\\ & T_{22}\end{pmatrix} = \exp\left(\begin{pmatrix}-A_x^T & O^T MO\\ 0 & A_x\end{pmatrix}h\right) \qquad (A.114)$$

and

$$O = \begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & I\\ 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{pmatrix}. \qquad (A.115)$$

This ends the derivation of the generalised H_2 measure of the sampled-data system in figure A.6. Again, the computation involves some matrix exponentials and a discrete-time Lyapunov equation.

Appendix B. MATLAB Tools

B.1 Overview

Some MATLAB tools were devellopped during this graduation project. They include the computation of the frequency gains for a general system, and the computation of the integral quadratic tracking errors for a motion control system with velocity and acceleration feedforward. An example controller parameterisation function is included, and a convenient interface to the sequential quadratic programming implementation of MATLABs Optimization Toolbox is provided. Genetic algorithm code is presented too.
The MATLAB code is provided for the functions enumerated below.

FFG	-	Fundamental frequency gain
DFG	-	Discrete frequency gain
PFG	-	Performance frequency gain
simpleH2	-	Integral quadratic tracking error for a motion control system with acceleration and velocity feedforward and synchronous acceleration profile
genH2	-	Average integral quadratic tracking error for a motion control system with acceleration and velocity feedforward and asynchronous acceleration profile
sqp	-	Interface to sequential quadratic programming in Optimization Toolbox
ga	-	Genetic algorithm
param	-	Example of a parameterisation function, parameterising second order controllers for a position controlled DC motor

B.2 FFG.m

```
% FFG  Fundamental Frequency Gain for a sampled-data system
%
% p = FFG(G11, G12, G21, G22, K, w) returns the FFG at frequencies (in rad/s)
% specified in vector w of the general sampled-data feedback control system
% specified by the continuous-time LTI systems G11, G12, G21, G22 and
% discrete-time feedback controller K.
%
% 11-6-2004
% Robert Cloudt

function p = PFR(G11, G12, G21, G22, K, w)

h = get(K, 'Ts');

% Form discrete control sensitivity
G22d = c2d(G22, h, 'zoh');
Rd = feedback(K, G22d, +1);
[nout, nin] = size(G12);

% Compute frequency responses
G11resp = freqresp(G11, w);
G12resp = freqresp(G12, w);
G21resp = freqresp(G21, w);
G22resp = freqresp(G22, w);
Rdresp = freqresp(Rd, w);
for k = 1:length(w)
    Hresp(1:nin, 1:nin, k) = (1 - exp(-j*w(k)*h))/(j*w(k))*eye(nin);
```

```
end;

% Compute FFR
for k = 1:length(w)
    p(k) = max(svd(G11resp(:, :, k) + 1/h*G12resp(:, :, k)*Hresp(:, :, k)*...
               Rdresp(:, :, k)*G21resp(:, :, k)));
end;
```

B.3 DFG.m

```
% DFG  Discrete Frequency Gain for a sampled-data system
%
% p = DFG(G11, G12, G21, G22, K, w) returns the DFG at frequencies (in rad/s)
% specified in vector w of the general sampled-data feedback control system
% specified by the continuous-time LTI systems G11, G12, G21, G22 and
% discrete-time feedback controller K.
%
% 11-6-2004
% Robert Cloudt

function p = DFG(G11, G12, G21, G22, K, w)

h = get(K, 'Ts');

% Form ZOH transformed systems
G11d = c2d(G11, h, 'zoh');
G12d = c2d(G12, h, 'zoh');
G21d = c2d(G21, h, 'zoh');
Rd = feedback(K, G22d, +1);

% Construct interconnection
M = G11d + G21d*Rd*G12d;

% Compute DFR
Mresp = freqresp(M, w);
for k = 1:length(w)
    p(k) = max(svd(Mresp(:, :, k)));
end;
```

B.4 PFG.m

```
% PFG  Performance Frequency Gain for a sampled-data system
%
% p = PFG(G11, G12, G21, G22, K, w) returns the PFG at frequencies (in rad/s)
% specified in vector w of the general sampled-data feedback control system
% specified by the continuous-time LTI systems G11, G12, G21, G22 and
% discrete-time feedback controller K.
%
% 11-6-2004
% Robert Cloudt

function p = PFR(G11, G12, G21, G22, K, w)

h = get(K, 'Ts');

% Compute infinite summation term in PFG
[A, B, C, D] = ssdata(G12);
n = size(A, 1);
nin = size(B, 2);
F = [A, zeros(size(A)); C'*C, -A'];
G = [B; C'*D];
H = [D'*C, -B'];
J = D'*D;
```

```
E = expm([zeros(nin, 1), H, J;
          zeros(2*n, 1), F, G;
          zeros(1, (2*n + nin + 1))]*h);
Ad = E((nin + 1):(nin + 2*n), 2:(2*n + 1));
Bd = E((nin + 1):(nin + 2*n) ,(2*n + 2):(2*n + nin + 1));
Cd = E(1:nin, 2:(2*n + 1));
Dd = E(1:nin, (2*n + 2):(2*n + nin + 1));
infsumsys = ss(Ad, Bd, Cd, Dd, h);

% Form discrete control sensitivity
G22d = c2d(G22, h, 'zoh');
Rd = feedback(K, G22d, +1);

% Compute frequency responses
G11resp = freqresp(G11, w);
G12resp = freqresp(G12, w);
G21resp = freqresp(G21, w);
infsumsysresp = freqresp(infsumsys, w);
Rdresp = freqresp(Rd, w);
for k = 1:length(w)
    Hresp(1:nin, 1:nin, k) = (1 - exp(-j*w(k)*h))/(j*w(k))*eye(nin);
end;

% Compute PFR
for k = 1:length(w)
    Aw(:, :, k) = (G11resp(:, :, k))'*G11resp(:, :, k) +...
                  1/h*(G11resp(:, :, k))'*G12resp(:, :, k)*Hresp(:, :, k)*...
                     Rdresp(:, :, k)*G21resp(:, :, k) +...
              1/h*(G21resp(:, :, k))'*(Rdresp(:, :, k))'*(Hresp(:, :, k))'*...
                 (G12resp(:, :, k))'*G11resp(:, :, k) +...
              1/h*(G21resp(:, :, k))'*(Rdresp(:, :, k))'*...
                 infsumsysresp(:, :, k)*Rdresp(:, :, k)*G21resp(:, :, k);
   % take real part becuase of numerical inaccuracies
   p(k) = sqrt(max(real(svd(Aw(k)))));
end;
```

B.5 simpleH2.m

```
% simpleH2  Simple sampled-data H2 measure for motion control system
%           with acceleration and velocity feedforward
%
% J = simpleH2(R, P, Kfb, Ka, Kv, lambda) returns the simple sampled-data H2
% measure of a continuous-time plant P controlled by discrete-time feedback
% controller Kfb  and (discrete-time) feedforward specified by Ka and Kv. Lambda
% defines the computation delay. R is a discrete-time system whose impulse
% response specifies the desired acceleration profile. Kfb and R have the same
% sampling time.
%
% 19-7-2004
% Robert Cloudt

function J = simpleH2(R, P, Kfb, Ka, Kv, lambda)

% Perturbation size
e = -1e-3;

% Remove integrator from P
[z, p, k] = zpkdata(P, 'v');
p = esort(p);
p = p(2:length(p));
Pquote = zpk(z, p, k);

% Extract system matrices and sizes
[Ak, Bk, Ck, Dk, h] = ssdata(Kfb);
[Ar, Br, Cr, Dr] = ssdata(minreal(R));
[Apquote, Bpquote, Cpquote, Dpquote] = ssdata(Pquote);
```

```
nk = size(Ak, 1);
nr = size(Ar, 1);
npquote = size(Apquote, 1);

% Construct CChat
Dpd0lambda = lambda*[Cpquote, 0]*expm([Apquote, Bpquote;
                                        zeros(1, (npquote + 1))]*lambda)*...
                [zeros(npquote, 1); 1] + lambda*Dpquote;
H1 = [1, 0, 0, -1, zeros(1, (npquote + 1))];
F1 = blkdiag([0, 1, 0;
              0, e, 1;
              0, 0, 0], [0, Cpquote, Dpquote;
                         zeros(npquote, 1), Apquote, Bpquote;
                         zeros(1, (npquote + 2))]);
G1 = [1, zeros(1, (nr + npquote + nk + 2));
      0, 1, zeros(1, (nr + npquote + nk + 1));
      0, 0, Cr, zeros(1, (npquote + nk + 1));
      zeros(1, (nr + npquote + nk + 3));
      zeros(npquote, (nr + 2)), eye(npquote), zeros(npquote, (nk + 1));
      zeros(1, (nr + npquote + nk + 2)), 1];
H2 = [[1, 0, 0]*expm([0, 1, 0;
                      0, e, 1;
                      0, 0, 0]*lambda), -1, zeros(1, npquote), 0];
F2 = F1;
G2 = [1, zeros(1, (nr + npquote + nk + 2));
      0, 1, zeros(1, (nr + npquote + nk + 1));
      0, 0, Cr, zeros(1, (npquote + nk + 1));
      zeros(1, (nr + npquote + nk + 2)), Dpd0lambda;
      zeros(npquote , (nr + 2)), expm(Apquote*lambda), zeros(npquote, (nk + 1));
      Dk, Kv, Ka*Cr, zeros(1, npquote), Ck, 0];
Q = expm([-F1', H1'*H1; zeros(npquote + 5), F1]*lambda);
Q12 = Q(1:(npquote + 5), (npquote + 6):(2*npquote + 10));
Q22 = Q((npquote + 6):(2*npquote + 10), (npquote + 6):(2*npquote + 10));
P = expm([-F2', H2'*H2; zeros(npquote + 5), F2]*(h - lambda));
P12 = P(1:(npquote + 5), (npquote + 6):(2*npquote + 10));
P22 = P((npquote + 6):(2*npquote + 10), (npquote + 6):(2*npquote + 10));
CChat = G1'*Q22'*Q12*G1 + G2'*P22'*P12*G2;

% Construct DDhat
H3 = [1, 0, 0];
F3 = [0, 1, 0;
      0, e, 1;
      0, 0, 0];
G3 = [0; 0; Dr];
H4 = H2;
F4 = F2;
G4 = [0; 0; Dr; 0; zeros(npquote, 1); Ka*Dr];
R = expm([-F3', H3'*H3; zeros(3), F3]*lambda);
R12 = R(1:3, 4:6);
R22 = R(4:6, 4:6);
S = expm([-F4', H4'*H4; zeros(npquote + 5), F2]*(h - lambda));
S12 = S(1:(npquote + 5), (npquote + 6):(2*npquote + 10));
S22 = S((npquote + 6):(2*npquote + 10), (npquote + 6):(2*npquote + 10));
DDhat = G3'*R22'*R12*G3 + G4'*S22'*S12*G4;

% Construct Ahat
Bpquotedlambdah = [eye(npquote), zeros(npquote, 1)]*...
                      expm([Apquote, Bpquote;
                            zeros(1, (npquote + 1))]*(h - lambda))*...
                               [zeros(npquote, 1); 1];
Bpquoted0lambda = [eye(npquote), zeros(npquote, 1)]*...
                      expm([Apquote, Bpquote;
                            zeros(1, (npquote + 1))]*lambda)*[zeros(npquote, 1); 1];
Ahat = [H2*expm(F2*(h - lambda))*G2;
        0, [1, zeros(1, nr)]*...
           expm([e, Cr;
                 zeros(nr, (nr + 1))]*h), zeros(1, (npquote + nk + 1));
        zeros(nr, 2), Ar, zeros(nr, (npquote + nk + 1));
        Bpquotedlambdah*Dk, Bpquotedlambdah*Kv, Bpquotedlambdah*Ka*Cr,
           expm(Apquote*h), Bpquotedlambdah*Ck, Bpquoted0lambda;
        Bk, zeros(nk, (nr + npquote + 1)), Ak, zeros(nk, 1);
        Dk, Kv, Ka*Cr, zeros(1, npquote), Ck, 0];

if max(abs(eig(Ahat))) > 1
    warning('Kfb does not stabilize P');
end;

% Construct Bhat
Bhat = [H4*expm(F4*(h - lambda))*G4;
        [1, 0]*expm([e, Dr;
                     0, 0]*h)*[0; 1];
        Br;
        Bpquotedlambdah*Ka*Dr
        zeros(nk, 1);
        Ka*Dr];

% Solve Lyapunov equation
M = dlyap(Ahat', CChat);

% Return norm
J = sqrt(DDhat + Bhat'*M*Bhat);
```

B.6 genH2.m

```
% genH2  Generalised sampled-data H2 measure for motion control system
%        with acceleration and velocity feedforward
%
% J = genH2(R, P, Kfb, Ka, Kv, lambda) returns the generalised H2 measure
% for continuous-time plant P controlled by discrete-time feedback
% controller Kfb and (discrete-time) feedforward specified by Ka  and Kv.
% Lambda defines the computation delay. R is a continuous-time system whose
% impulse response specifies the desired acceleration profile.
%
% 19-7-2004
% Robert Cloudt

function J = genH2(R, P, Kfb, Ka, Kv, lambda)

% Perturbation size
e = -1e-2;

% Remove integrator from P
[z, p, k] = zpkdata(P, 'v');
p = esort(p);
p = p(2:length(p));
Pquote = zpk(z, p, k);

% Extract system matrices and sizes
[Ak, Bk, Ck, Dk, h] = ssdata(Kfb);
[Ar, Br, Cr, Dr] = ssdata(minreal(R));
[Apquote, Bpquote, Cpquote, Dpquote] = ssdata(Pquote);
nk = size(Ak, 1);
nr = size(Ar, 1);
npquote = size(Apquote, 1);

% Construct CChat
AA = [0, 1, zeros(1, nr);
      0, e, Cr;
      zeros(nr, 2), Ar];
Dpd0lambda = lambda*[Cpquote, 0]*expm([Apquote, Bpquote;
                                        zeros(1, (npquote + 1))]*lambda)*...
                   [zeros(npquote, 1); 1] + lambda*Dpquote;
H1 = [1, zeros(1, (nr + 1)), -1, zeros(1, (npquote + 1))];
F1 = blkdiag(AA, [0, Cpquote, Dpquote;
                    zeros(npquote, 1), Apquote, Bpquote;
                    zeros(1, (npquote + 2))]]);
G1 = [1, zeros(1, (nr + npquote + nk + 2));
      0, 1, zeros(1, (nr + npquote + nk + 1));
      zeros(nr, 2), eye(nr), zeros(nr, (npquote + nk + 1));
      zeros(1, (nr + npquote + nk + 3));
      zeros(npquote, (nr + 2)), eye(npquote), zeros(npquote, (nk + 1));
      zeros(1, (nr + npquote + nk + 2)), 1];
H2 = [[1, zeros(1, (nr + 1))]*expm(AA*lambda), -1, zeros(1, (npquote + 1))];
F2 = F1;
G2 = [1, zeros(1, (nr + npquote + nk + 2));
      0, 1, zeros(1, (nr + npquote + nk + 1));
      zeros(nr, 2), eye(nr), zeros(nr, (npquote + nk + 1));
      zeros(1, (nr + npquote + nk + 2)), Dpd0lambda;
      zeros(npquote , (nr + 2)), expm(Apquote*lambda), zeros(npquote, (nk + 1));
      Dk, Kv, Ka*Cr, zeros(1, npquote), Ck, 0];
Q = expm([-F1', H1'*H1; zeros(nr + npquote + 4), F1]*lambda);
Q12 = Q(1:(nr + npquote + 4), (nr + npquote + 5):(2*nr + 2*npquote + 8));
Q22 = Q((nr + npquote + 5):(2*nr + 2*npquote + 8),
        (nr + npquote + 5):(2*nr + 2*npquote + 8));
P = expm([-F2', H2'*H2; zeros(nr + npquote + 4), F2]*(h - lambda));
P12 = P(1:(nr + npquote + 4), (nr + npquote + 5):(2*nr + 2*npquote + 8));
P22 = P((nr + npquote + 5):(2*nr + 2*npquote + 8),
        (nr + npquote + 5):(2*nr + 2*npquote + 8));
CChat = G1'*Q22'*Q12*G1 + G2'*P22'*P12*G2;

% Construct Ahat
Bpquotedlambdah = [eye(npquote), zeros(npquote, 1)]*...
                       expm([Apquote, Bpquote;
                             zeros(1, (npquote + 1))]*(h - lambda))*...
                                [zeros(npquote, 1); 1];
Bpquoted0lambda = [eye(npquote), zeros(npquote, 1)]*...
                       expm([Apquote, Bpquote;
                             zeros(1, (npquote + 1))]*lambda)*[zeros(npquote, 1); 1];
Ahat = [H2*expm(F2*(h - lambda))*G2;
        zeros((nr + 1), 1), expm([e, Cr;
                                   zeros(nr, 1), Ar]*h), zeros((nr + 1),
                                      (npquote + nk + 1));
        Bpquotedlambdah*Dk, Bpquotedlambdah*Kv, Bpquotedlambdah*Ka*Cr,
           expm(Apquote*h), Bpquotedlambdah*Ck, Bpquoted0lambda;
        Bk, zeros(nk, (nr + npquote + 1)), Ak, zeros(nk, 1);
        Dk, Kv, Ka*Cr, zeros(1, npquote), Ck, 0];

if max(abs(eig(Ahat))) > 1
    warning('Kfb does not stabilize P');
    J = 1e10;
    return;
end;

% Solve Lyapunov equation
M = dlyap(Ahat', CChat);

% Compute integral of BMB
T = expm([-AA', [eye(nr + 2); zeros((npquote + nk + 1), (nr + 2))]'*M*[eye(nr + 2);
zeros((npquote + nk + 1), (nr + 2))];
                  zeros(nr + 2), AA]*h);
T12 = T(1:(nr + 2), (nr + 3):(2*nr + 4));
T22 = T((nr + 3):(2*nr + 4), (nr + 3):(2*nr + 4));
iBMB = [0, 0, Br']*T22'*T12*[0; 0; Br];

% Compute integral of DD
R = expm([-AA', eye(nr + 2), zeros(nr + 2);
          zeros(nr + 2), -AA', blkdiag(1, zeros(nr + 1));
          zeros((nr + 2), (2*nr + 4)), AA]*h);
R13 = R(1:(nr + 2), (2*nr + 5):(3*nr + 6));
R33 = R((2*nr + 5):(3*nr + 6), (2*nr + 5):(3*nr + 6));
iDD = [0, 0, Br']*R33'*R13*[0; 0; Br];
```

```
% Return norm
J = sqrt(iDD/h + iBMB/h);
```

B.7 sqp.m

```
% SQP  Sequential Quadratic Programming for controller optimisation
%
% [thetaopt, hopt] = SQP(ub, lb, @param, @stab, @perf, @aux, h, theta0, h0, N)
% can  be used to optimise controllers with respect to some performance and
% stability criteria. The controller is parameterized by a parameter vector
% with size equal to the size of ub and lb.
%
% ub and lb are equal length row vectors which specify the upper and lower
% bounds on the parameter vector.
%
% @param is a funtion handle to the parameterisation function. This function
% has to accept the vector of real parameters and a sample time. It returns
% a structure containing a characterisation of the controller. The structure
% has to contain all the information necessary for controller evaluation.
%
% @stab is a function handle to the function which specifies the stability
% criteria. This function should accept the structure returned by the
% parameterisation function and it has to return a scalar value quantifying
% the stability constraint violation. A value less than or equal to zero
% indicates (robust) stability.
%
% @perf is a function handle to a function which specifies the performance
% criteria. This function should accept the structure returned by the
% parameterisation function and it has to return a scalar value quantifying
% the performance constraint violation. A value less than or equal to zero
% indicates that the performance constraints are met.
%
% @aux is a function handle to an auxiliary function which is used to sort
% individuals which satisfy stability and performance criteria. The function
% should accept the structure returned by the parameterisation function. The
% individuals are sorted in ascending order with respect to the value returned
% by the auxiliary function.
%
% h specifies the sampling time of the discrete-time controller being optimised.
% h can be a real scalar or a two-element vector [hhigh, hlow] which specifies
% the upper and lower bound on the sampling time.
%
% theta0 and h0 specifify the initial guesses for controller perameters and
% sampling time. If h is a scalar, then h0 has to be equal to h.
%
% N is the maximum number of iterations in the SQP algorithm. The default is 20.
%
% The algrithm returns the optimal vector of real parameters thetaopt and the
% optimal sampling time hopt. If h is a scalar, then h = hopt.
%
% 20-7-2004
% Robert Cloudt

function [thetaopt, hopt] = sqp(ub, lb, param, stab, perf, aux, h, theta0, h0, N)

if (nargin == 9)
    N = 20;
end;

% Determine size of parameter vector
if (size(ub) ~= size(lb))
    error('ub and lb have unequal length');
end;
n = length(ub);
if (size(h, 2) == 2)
```

```
    nh = 1;
    ubx = [h(1), ub];
    lbx = [h(2), lb];
    x0 = [h0, theta0];
else
    nh = 0;
    ubx = ub;
    lbx = lb;
    x0 = theta0;
end;

% Set optimisation options
options = optimset('LargeScale', 'off', 'Display', 'Iter', 'MaxIter', N, 'TolFun',
             1e-10, 'MaxFunEvals', 1000);

% Start optimisation
x = fmincon(@fun, x0, [], [], [], [], lbx, ubx, @nonlcon, options, param, stab,
       perf, aux, h);

% Return optimal solution
if nh == 1
    hopt = x(1);
    thetaopt = x(2:(n + 1));
else
    hopt = h;
    thetaopt = x;
end;
return;



% Interface between objective function used in fmincon and auxiliary function
function f = fun(x, param, stab, perf, aux, h)
if (size(h, 2) == 2)
    hparam = x(1);
    theta = x(2:length(x));
else
    hparam = h;
    theta = x;
end;
Cstruct = feval(param, theta, hparam);
f = feval(aux, Cstruct);
return;


% Interface between nonlinear constraint function in fmincon and the performance
and stability functions
function [c, ceq] = nonlcon(x, param, stab, perf, aux, h)
ceq = 0;
if (size(h, 2) == 2)
    hparam = x(1);
    theta = x(2:length(x));
else
    hparam = h;
    theta = x;
end;
Cstruct = feval(param, theta, hparam);
c(1) = feval(stab, Cstruct);
c(2) = feval(perf, Cstruct);
return;
```

B.8 ga.m

```
% GA  Genetic Algorithm for controller optimisation
%
% [thetaopt, hopt] = GA(ub, lb, @param, @stab, @perf, @aux, h, N) can be used
% to optimise controllers with respect to some performance and stability
% criteria. The controller is parameterized by a parameter vector with size
% equal to the size of ub and lb.
%
% ub and lb are equal length row vectors which specify the upper and lower
% bounds on the parameter vector.
%
% @param is a funtion handle to the parameterisation function. This function
% has to accept the vector of real parameters and a sample time. It returns
% a structure containing a characterisation of the controller. the structure
% has to contain all the information necessary for controller evaluation.
%
% @stab is a function handle to the function which specifies the stability
% criteria. This function should accept the structure returned by the
% parameterisation function and it has to return a scalar value quantifying
% the stability constraint violation. A value less than or equal to zero
% indicates (robust) stability.
%
% @perf is a function handle to a function which specifies the performance
% criteria. This function should accept the structure returned by the
% parameterisation function and it has to return a scalar value quantifying
% the performance constraint violation. A value less than or equal to zero
% indicates that the performance constraints are met.
%
% @aux is a function handle to an auxiliary function which is used to sort
% individuals which satisfy stability and performance criteria. The function
% should accept the structure returned by the parameterisation function. The
% individuals are sorted in ascending order with respect to the value returned
% by the auxiliary function.
%
% h specifies the sampling time of the discrete-time controller being optimised.
% h can be a real scalar or a two-element vector [hhigh, hlow] which specifies
% the upper and lower bound on the sampling time.
%
% N is the number of generations which the genetic algorithm is allowed to
% run. The default is 10.
%
% The algrithm returns the optimal vector of real parameters thetaopt and the
% optimal sampling time hopt. If h was a scalar, then h = hopt.
%
% 20-7-2004
% Robert Cloudt

function [thetaopt, hopt] = ga(ub, lb, param, stab, perf, aux, h, Ngen)

if (nargin == 7)
    Ngen = 10;
end;

% Determine size of parameter vector
if (size(ub) ~= size(lb))
    error('ub and lb have unequal length');
end;
n = length(ub);
if (size(h, 2) == 2)
    nh = 1;
else
    nh = 0;
end;

% Genetic algorithm parameters
```

```
N = 50;                 % population size
Pcross = 0.7;           % crossover probability
Pmut = 0.03;            % mutation probability
pres = 1.15;            % selective pressure
M = 16;                 % number of bits used in binary representation of each parameter

% Generate random initial population
pop = (rand(N, (n*M + nh*M)) > 0.5);

% Evaluate initial individuals
NS = [];
RP = [];
RS = [];
[popreal, hparam] = decode(pop, h, ub, lb, M);
for k = 1:N
    Cstruct = feval(param, popreal(k, :), hparam(k));
    stabcv = feval(stab, Cstruct);
    if stabcv > 0
        % no (robust) stability
        NS = [NS; k, stabcv];
    else
        perfcv = feval(perf, Cstruct);
        if perfcv > 0
            % (robust) stability
            RS = [RS; k, perfcv];
        else
            % (robust) performance
            RP = [RP; k, feval(aux, Cstruct)];
        end;
    end;
end;

elite = [];
% Save elite (robust) performing individual
    if ~isempty(RP)
        RP = sortrows(RP, 2);
        elite = pop(RP(1, 1), :);
    end;

% Apply ranking and sharing in the NS and RS classes to maintain diversity of the
sampling rate
if (nh == 1)
    if ~isempty(NS)
        NS = sortrows(NS, 2);
        NS(:, 2) = (size(NS, 1):-1:1)';
        sigma = (h(1) - h(2))/size(NS, 1);
        for i = 1:size(NS, 1)
            nichecount = 0;
            for j = 1:size(NS, 1)
                dist = abs(hparam(NS(i, 1)) - hparam(NS(j, 1)));
                if (dist < sigma)
                    nichecount = nichecount + 1 - dist/sigma;
                end;
            end;
            NS(i, 2) = NS(i, 2)/nichecount;
        end;
    end;
    if ~isempty(RS)
        RS = sortrows(RS, 2);
        RS(:, 2) = (size(RS, 1):-1:1)';
        sigma = (h(1) - h(2))/size(RS, 1);
        for i = size(RS, 1)
            nichecount = 0;
            for j = size(RS, 1)
                dist = abs(hparam(RS(i, 1)) - hparam(RS(j, 1)));
                if (dist < sigma)
                    nichecount = nichecount + 1 - dist/sigma;
                end;
            end;
            RS(i, 2) = RS(i, 2)/nichecount;
        end;
    end;
end;

% Rank population
if ~isempty(NS)
    NS = sortrows(NS, 2);
    rank(NS(:, 1)) = N:-1:(N - size(NS, 1) + 1);
    NS = sortrows(NS, 2);
end;
if ~isempty(RS)
    RS = sortrows(RS, 2);
    rank(RS(:, 1)) = (N - size(NS, 1)):-1:(N - size(NS, 1) - size(RS, 1) + 1);
    RS = sortrows(RS, 2);
end;
if ~isempty(RP)
    RP = sortrows(RP, 2);
    rank(RP(:, 1)) = 1:(N - size(NS, 1) - size(RS, 1));
end;

% Assign initial dummy fitness value through linear ranking
for k = 1:N
    f(k) = 2 - pres + 2*(pres - 1)*(N - rank(k))/(N - 1);
end;



% MAIN LOOP
for g = 1:Ngen

    % Selection through Stochastic Universal Sampling
    step = sum(f)/N;
    ptr = rand*step;
    partsum  = 0;
    sel = [];
    for k = 1:N
        partsum = partsum + f(k);
        while (ptr < partsum)
            sel = [sel; k];
            ptr = ptr + step;
        end;
    end;

    % Random permutation of selection
    perm = randperm(N);
    pop = pop(sel(perm), :);

    % Perform crossover & mutation
    for k = 1:2:floor(N/2)
        if rand < Pcross
            % do crossover
            pos = ceil(rand*(n*M + nh*M));
            if pos < (n*M + nh*M)
                % crossover between pos and pos + 1
                pop(k, :) = [pop(k, 1:pos), pop((k + 1), (pos + 1):(n*M + nh*M))];
                pop((k + 1), :) = [pop((k + 1), 1:pos),
                                     pop(k, (pos + 1):(n*M + nh*M))];
            else
                % possible mutation
                if rand < Pmut
                    % mutate individual k
                     pos = ceil(rand*(n*M + nh*M));
                     pop(k, pos) = ~pos(k, pos);
                elseif rand < Pmut
                    % mutate individual k + 1
                     pos = ceil(rand*(n*M + nh*M));
                    pop((k + 1), pos) = ~pos((k + 1), pos);
                end;
            end;
        end;
    end;

    % Reinsert elite individual if necessary
    found = 0;
    if ~isempty(elite)
        for k = 1:N
            found = found | isequal(pop(k, :), elite);
        end;
        if ~found
            NS = sortrows(NS, 2);
            pop(NS(end, 1), :) = elite;
        end;
    end;

    % Evaluate individuals
    NS = [];
    RP = [];
    RS = [];
    [popreal, hparam] = decode(pop, h, ub, lb, M);
    for k = 1:N
        Cstruct = feval(param, popreal(k, :), hparam(k));
        stabcv = feval(stab, Cstruct);
        if stabcv > 0
            % no (robust) stability
            NS = [NS; k, stabcv];
        else
            perfcv = feval(perf, Cstruct);
            if perfcv > 0
                % (robust) stability
                RS = [RS; k, perfcv];
            else
                % (robust) performance
                RP = [RP; k, feval(aux, Cstruct)];
            end;
        end;
    end;

    % Save elite (robust) performing individual
    if ~isempty(RP)
        RP = sortrows(RP, 2);
        elite = pop(RP(1, 1), :);
    end;

    % Apply ranking and sharing in the NS and RS classes to maintain diversity of
    % the sampling rate
   if (nh == 1)
        if ~isempty(NS)
            NS = sortrows(NS, 2);
            NS(:, 2) = (size(NS, 1):-1:1)';
            sigma = (h(1) - h(2))/size(NS, 1);
            for i = 1:size(NS, 1)
                nichecount = 0;
                for j = 1:size(NS, 1)
                    dist = abs(hparam(NS(i, 1)) - hparam(NS(j, 1)));
                    if (dist < sigma)
                        nichecount = nichecount + 1 - dist/sigma;
                    end;
                end;
                NS(i, 2) = NS(i, 2)/nichecount;
            end;
        end;
        if ~isempty(RS)
            RS = sortrows(RS, 2);
            RS(:, 2) = (size(RS, 1):-1:1)';
```

```
                sigma = (h(1) - h(2))/size(RS, 1);
                for i = size(RS, 1)
                    nichecount = 0;
                    for j = size(RS, 1)
                        dist = abs(hparam(RS(i, 1)) - hparam(RS(j, 1)));
                        if (dist < sigma)
                            nichecount = nichecount + 1 - dist/sigma;
                        end;
                    end;
                    RS(i, 2) = RS(i, 2)/nichecount;
                end;
            end;
    end;

    % Rank population
    if ~isempty(NS)
         NS = sortrows(NS, 2);
         rank(NS(:, 1)) = N:-1:(N - size(NS, 1) + 1);
         NS = sortrows(NS, 2);
    end;
    if ~isempty(RS)
         RS = sortrows(RS, 2);
         rank(RS(:, 1)) = (N - size(NS, 1)):-1:(N - size(NS, 1) - size(RS, 1) + 1);
         RS = sortrows(RS, 2);
    end;
    if ~isempty(RP)
         RP = sortrows(RP, 2);
         rank(RP(:, 1)) = 1:(N - size(NS, 1) - size(RS, 1));
    end;


     % Assign dummy fitness value through linear ranking
     for k = 1:N
         f(k)   = 2 - pres + 2*(pres - 1)*(N - rank(k))/(N - 1);
     end;


     if isempty(RP)
         fprintf('Generation %d: #stab = %d, #perf = %d\n', g, size(RS, 1),
           size(RP, 1));
     else
         fprintf('Generation %d: #stab = %d, #perf = %d, avg(aux) = %g, min(aux) =
            %g\n', g, size(RS, 1), size(RP, 1), mean(RP(:, 2)), RP(1, 2));
     end;
end;

% Return optimal solution
if ~isempty(RP)
    [thetaopt, hopt] = decode(pop(RP(1, 1), :), h, ub, lb, M);
else
    [thetaopt, hopt] = decode(pop(RS(1, 1), :), h, ub, lb, M);
end;
return;
```

```
% Decode binary representation to real representation
function [preal, hparam] = decode(p, h, ub, lb, M)

% Create column vector containing powers of 2
pow2 = 2.^(0:(M - 1))';

% Decode binary strings to real values in range [0, (2^M-1)]
for k = 1:size(p, 1)
```

```
    for m = 1:(size(p, 2)/M)
        preal(k, m) = p(k, (1 + (m - 1)*M):(M + (m - 1)*M))*pow2;
    end;
end;

% Scale real values
if (length(h) == 2)
    hparam = h(2) + (h(1) - h(2))*preal(:, 1)./(2^M - 1);
    for n = 2:size(preal, 2)
        preal(:, (n - 1)) = lb(n - 1) + (ub(n - 1) - lb(n - 1))*...
            preal(:, n)./(2^M - 1);
    end;
else
    hparam = h*ones(size(preal, 1), 1);
    for n = 1:size(preal, 2)
        preal(:, n) = lb(n) + (ub(n) - lb(n))*preal(:, n)./(2^M - 1);
    end;
end;
return;
```

B.9 param.m

```
% PARAM.M
%
% Second order controller parameterisation for a position controlled DC motor.
%
% Find an LTI controller representation from the parameter vector theta and the
% sampling time h.
% The four parameters in the parameter vector specify four closed-loop poles
% If the discretized plant has structure:
%
%            k(z-a)
% G(z) = ------------
%          (z-1)(z-b)
%
% and the controller has the structure:
%
%            R(z)
% K(z) = ----------- , where deg(R(z)) = 2 and deg(S(z)) = 1,
%          S(z)(z-1)
%
% then there exists a unique solution to the Diophantine equation:
%
% Q(z)S(z) + P(z)R(z) = C(z)
%
% where Q(z) = (z-b)(z-1)^2, P(z) = k(z-a) and C(z) is the desired characteristic
% equation. C(z) has  four roots equal to the closed-loop poles.
%
% 4-8-2004
% Robert Cloudt

function Cstruct = param(theta, h)

global Pnom;

% Extract a and b
[a, p, k] = zpkdata(c2d(Pnom, h), 'v');
p = sort(p);
b = p(1);

% Form polynomials P and Q
P = [k, -k*a];
Q = conv([1, -b], [1, -2, 1]);

% Form desired characteristic equation C
if (theta(1) >= 0)
```

```
    % complex pole pair
    C1 = conv([1, -abs(theta(2))*exp(j*pi*theta(1))],
            [1, -abs(theta(2))*exp(-j*pi*theta(1))]);
else
    % two real poles
    C1 = conv([1, (-(2*theta(1) + 1)*(1 - abs(theta(2))) + abs(theta(2)))],
            [1, (-(2*theta(1) + 1)*(1 - abs(theta(2))) - abs(theta(2)))]);
end;
if (theta(3) >= 0)
    % complex pole pair
    C2 = conv([1, -abs(theta(4))*exp(j*pi*theta(3))],
            [1, -abs(theta(4))*exp(-j*pi*theta(3))]);
else
    % two real poles
    C2 = conv([1, (-(2*theta(3) + 1)*(1 - abs(theta(4))) + abs(theta(4)))],
            [1, (-(2*theta(3) + 1)*(1 - abs(theta(4))) - abs(theta(4)))]);
end;
C = conv(C1, C2);

% Solve Diophantine equation through a set of linear equations
nr = 2 + 1;
ns = 1 + 1;
np = size(P, 2);
nq = size(Q, 2);
% nc = nr + ns;
for p = 1:nr
    A(:, p) = [zeros((p - 1), 1); flipud(P'); zeros((ns + nr - p + 1 - np), 1)];
end;
for q = 1:ns
    A(:, (nr + q)) = [zeros((q - 1), 1); flipud(Q');
                      zeros((ns + nr - q + 1 - nq), 1)];
end;
RS = A\flipud(C');
R = flipud(RS(1:nr))';
S = flipud(RS((nr + 1):(nr + ns)))';

% Form controller structure
Cstruct.Kfb = zpk([], [1], 1, h)*tf(R, S, h);
Cstruct.Ka = theta(5);
Cstruct.Kv = theta(6);
Cstruct.h = h;
```

List of abbreviations

A/D	Analog to digital
ARP	Alias response power
D/A	Digital to analog
DC	Direct current
DFG	Discrete frequency gain
EMF	Electro-motive force
FFG	Fundamental frequency gain
GA	Genetic algorithm
LED	Light emitting diode
LPTV	Linear periodically time-varying
LQG	Linear quadratic Gaussian
LTI	Linear time-invariant
M5	Motor 5
MIMO	Multiple input, multiple output
MM	Main motor
MOSFET	Metal, oxide, semiconductor, field effect transistor
NS	Nominal stability
OS4	Optical sensor 4
P5	Pinch 5
PFG	Performance frequency gain
PH1	Preheater belt 1
PH2	Preheater belt 2
PID	Proportional/integral/derivative
PMDC	Permanent magnet, direct current
RFG	Robustness frequency gain
RP	Robust performance
RS	Robust stability
SISO	Single input, single output
SQP	Sequential quadratic programming
TF	Transfuse
ZOH	Zero order hold

References

1. Franklin, G.F. and J.D. Powell, M.L. Workman
DIGITAL CONTROL OF DYNAMIC SYSTEMS, 2nd ed.
Amsterdam: Addison-Wesley, 1990.

2. Damen, A. A. H.
INTRODUCTION DIGITAL CONTROL.
Measurement and Control group, Department of Electrical Engineering, Eindhoven University of Technology, 2001.
Lecture notes. (available at: `http://www.cs.ele.tue.nl/ADamen/REGEL2.HTM`)

3. Chen, Tongwen and B.A. Francis
OPTIMAL SAMPLED-DATA CONTROL SYSTEMS.
Berlin: Springer, 1995.
Communications and Control Engineering Series.

4. Rosenwasser, E.N. and B.P. Lampe
COMPUTER CONTROLLED SYSTEMS: ANALYSIS AND DESIGN WITH PROCESS-ORIENTED MODELS.
Berlin: Springer, 2000.
Communications and Control Engineering Series.

5. Christiansson, A.K.
A GENERAL FRAMEWORK FOR HYBRID H_∞-CONTROL.
Licentiate Thesis Chalmers University of Technology Göteborg, 2000.

6. Hagiwara, Tomomichi and Mituhiko Araki
FR-OPERATOR APPRAOCH TO THE H_2 ANALYSIS AND SYNTHESIS OF SAMPLED-DATA SYSTEMS.
IEEE Transactions on Automatic Control, Vol. 40 (1995), No. 8, p. 1411-1421.

7. Braslavsky, J.H.
FREQUENCY DOMAIN ANALYSIS OF SAMPLED-DATA CONTROL SYSTEMS.
PhD Thesis University of Newcastle, 1995.

8. Kalman, R. E. and J. E. Bertram
A UNIFIED APPROACH TO THE THEORY OF SAMPLING SYSTEMS.
Journal of the Franklin Institute, Vol. 267 (1959), No. 5. P. 405-436.

9. Sågfors, M.
OPTIMAL SAMPLED-DATA AND MULTIRATE CONTROL.
Academic Dissertation Åbo Akademi University, 1998.

10. Dullerud, G. and K. Glover
ROBUST STABILIZATION OF SAMPLED-DATA SYSTEMS TO STRUCTURED LTI PERTURBATIONS.
IEEE Transactions on Automatic Control, Vol. 38 (1993), No. 10, p. 1497-1508.

11. Damen, A.A.H and S. Weiland
ROBUST CONTROL.
Measurement and Control group, Department of Electrical Engineering, Eindhoven University of Technology, 2002.
Lecture notes. (available at: `http://www.cs.ele.tue.nl/SWeiland/RCTRL.HTM`)

12. Lindgärde, O. and B. Lennartson
COMPARING FREQUENCY ANALYSIS METHODS FOR SAMPLED-DATA CONTROL.
Proc. of the 37th Conference on Decision & Control, Tampa, Florida USA, 16-18 December 1998.
New York: IEEE, 1998, Vol. 1, p. 829-834.

13. Yamamoto, Yutaka and P.P. Khargonekar
FREQUENCY RESPONSE OF SAMPLED-DATA SYSTEMS.
IEEE Transactions on Automatic Control, Vol. 41 (1996), No. 2, p. 166-176.

14. Lindgärde, O. and B. Lennartson
FREQUENCY ANALYSIS FOR SINGLE AND MULTIRATE SAMPLED-DATA CONTROL.
Proc. of the 34th Conference on Decision & Control, New Orleans, USA, 13-15 December 1995.
New York: IEEE, 1995, Vol. 2, p. 1770-1775.

15. van Loan, C. F.
COMPUTING INTEGRALS INVOLVING THE MATRIX EXPONENTIAL.
IEEE Transactions on Automatic Control, Vol. 23 (1978), No. 3, p. 395-404.

16. Bamieh, B. and J.B. Pearson
THE H^2 PROBLEM FOR SAMPLED-DATA SYSTEMS.
Systems & Control Letters, Vol. 19 (1992), No. 1, p. 1-12.

17. Mirkin, L. and Z.J. Palmor
MIXED DISCRETE/CONTINUOUS SPECIFICATIONS IN SAMPLED-DATA H^2-OPTIMAL CONTROL.
Automatica, Vol. 33 (1997), No. 11, p. 1997-2014.

18. Engwerda, J. and G. van Willigenburg
OPTIMAL SAMPLING-RATES AND TRACKING PROPERTIES OF DIGITAL LQ AND LQG TRACKING CONTROLLERS FOR SYSTEMS WITH AN EXOGENOUS COMPONENT AND COSTS ASSOCIATED TO SAMPLING.
Computational Economics, Vol. 8 (1995), No. 2, p. 107-125.

19. Polyakov, K.Yu. and Ye.N. Rosenwasser, B.P. Lampe
DIRECTSD – A TOOLBOX FOR DIRECT DESIGN OF SD SYSTEMS.
Proc. of the 1999 IEEE Internatinal Symposium on Computer-Aided Control System Design, Kohala Coast, Hawaii USA, 22-27 August 1999.
Piscataway, NY USA: IEEE, 1999, p. 357-362.
(The toolbox is available from `http://www-at.e-technik.uni-rostock.de/~bl`)

20. Mohan, N. and T.M. Undeland, W.P. Robbins
POWER ELECTRONICS. Converters, Applications, and Design, 3rd ed.
New York: Wiley, 2003.

21. Bamieh, B.A. and J.B. Pearson
A GENERAL FRAMEWORK FOR LINEAR PERIODIC SYSTEMS WITH APPLICATIONS TO H^∞ SAMPLED-DATA CONTROL.
IEEE Transactions on Automatic Control, Vol. 37 (1992), No. 4, p. 418-435.

22. Lambrechts, P.
TRAJECTORY PLANNING AND FEEDFORWARD DESIGN FOR ELECTROMECHANICAL MOTION SYSTEMS VERSION 2.
Dynamics and Control Technology group, Department of Mechanical Engineering, Eindhoven University of Technology, 2003.
Report nr. DCT 203-18.

23. Sareni, B. and L. Krähenbühl.
FITNESS SHARING AND NICHING METHODS REVISITED.
IEEE Transactions on Evolutionary Computation, Vol. 2 (1998), No.3, p. 97-106.

24. Whitley, D.
THE GENITOR ALGORITHM AND SELECTION PRESSURE: WHY RANK-BASED ALLOCATION OF REPRODUCTIVE TRIALS IS BEST.
Proc. of the 3th International Conference on Genetic Algorithms, Fairfax, Viriginia, USA, 4-7 June 1989.
San Fransisco: Morgan Kauffman, 1989, p. 116-121.

25. van den Bosch, P.P.J. and van der Klauw, A.C.
MODELLING, IDENTIFICATION AND SIMULATION OF DYNAMICAL SYSTEMS.
Control group, Department of Electrical Engineering, Delft University of Technology, 1992.
Lecture notes.

26. Ljung, L.
SYSTEM IDENTIFICATION TOOLBOX USER'S GUIDE.
The MathWorks Inc., 1995.

27. Goldberg, D.E.
GENETIC ALGORITHMS IN SEARCH, OPTIMIZATION , AND MACHINE LEARNING.
Reading: Addison-Wesley, 1989.

28. Vanderplaats, G.N.
NUMERICAL OPTIMIZATION TECHNIQUES FOR ENGINEERING DESIGN: WITH APPLICATIONS.
New York: McGraw-Hill, 1984.
McGraw-Hill Series in Mechanical Engineering.

29. Coleman, T. and M.A. Branch, A. Grace
OPTIMIZATION TOOLBOX USER'S GUIDE.
The MathWorks Inc., 1999.

30. Peng, Xiao-long and B. Lennartson, C.M. Fransson
ON THE CHOICE OF SAMPLING PERIOD AND ROBUST POLE PLACEMENT.
Proc. of the 15th IFAC World Congress on Automatic Control, Barcelona, Spain, 21-26 July 2002.

31. Hoover, D.N. and R. Longchamp, J. Rosentahl
TWO-DEGREE-OF-FREEDOM ℓ_2-OPTIMAL TRACKING WITH PREVIEW.
Automatica, Vol. 40 (2004), No. 1, p. 155-162.

32. Vélez, C.M. and J. Salt
SIMULATION OF MULTIRATE CONTROL SYSTEMS IN MATLAB/SIMULINK.
Proc. of the 2nd World Manufacturing Congress, Durham, United Kingdom, 27-30 September 1999.
Sliedrecht: ICSC-NAISO Academic Press, 1999. p. 84-90.

33. Damen, A.A.H
PHYSIOLOGICAL PROCESSES AND PARAMETER ESTIMATION.
Measurement and Control group, Department of Electrical Engineering, Eindhoven University of Technology, 2003.
Lecture notes. (available at: `http://www.cs.ele.tue.nl/ADamen/Fyspar.htm`)

34. Yamamoto, Yutaka
A RETROSPECTIVE VIEW ON SAMPLED-DATA – CONTROL SYSTEMS.
CWI Quarterly, Vol. 9 (1996), No. 3, p. 261-276.