

# TOPICS IN UNDERGRADUATE CONTROL SYSTEMS DESIGN

João P. Hespanha

April 24, 2010

Disclaimer: This is an early draft and probably contains many typos.

Comments and information about typos are welcome. Please contact the author at ([hespanha@ece.ucsb.edu](mailto:hespanha@ece.ucsb.edu)).

© Copyright to João Hespanha. Please do not distribute this document without the author's consent.



# Contents

<b>I</b>	<b>System Identification</b>	<b>1</b>
	<b>System Identification</b>	<b>3</b>
<b>1</b>	<b>Computer-Controlled Systems</b>	<b>5</b>
1.1	Computer control . . . . .	5
1.2	Continuous-time systems . . . . .	6
1.3	Discrete-time systems . . . . .	8
1.4	Discrete vs. continuous-time transfer functions . . . . .	10
1.5	MATLAB® hints . . . . .	12
1.6	To probe further . . . . .	13
1.7	Exercise . . . . .	14
<b>2</b>	<b>Parametric identification using least-squares</b>	<b>15</b>
2.1	Parametric identification . . . . .	15
2.2	Least-squares line fitting . . . . .	15
2.3	Vector least-squares . . . . .	17
2.4	MATLAB® hints . . . . .	19
2.5	To probe further . . . . .	19
2.6	Exercises . . . . .	21
<b>3</b>	<b>Parametric identification of an ARX model</b>	<b>23</b>
3.1	ARX Model . . . . .	23
3.2	Identification of an ARX model . . . . .	24
3.3	Known parameters . . . . .	25
3.4	MATLAB® hints . . . . .	25
3.5	To probe further . . . . .	27
3.6	Exercises . . . . .	27
<b>4</b>	<b>Practical considerations in parametric identification</b>	<b>29</b>
4.1	Choice of inputs . . . . .	29
4.2	Signal scaling . . . . .	33
4.3	Choice of sampling frequency . . . . .	36
4.4	Choice of model order . . . . .	37
4.5	Combination of multiple experiments . . . . .	39
4.6	Exercises . . . . .	41
<b>5</b>	<b>Nonparametric identification</b>	<b>45</b>
5.1	Nonparametric methods . . . . .	45
5.2	Time-domain identification . . . . .	45
5.3	Frequency response identification . . . . .	47
5.4	MATLAB® hints . . . . .	48
5.5	To probe further . . . . .	49

5.6 Exercises . . . . .	50
<b>II Robust Control</b>	<b>51</b>
<b>Robust Control</b>	<b>53</b>
<b>6 Robust stability</b>	<b>55</b>
6.1 Model uncertainty . . . . .	55
6.2 Nyquist stability criterion . . . . .	58
6.3 Small gain condition . . . . .	59
6.4 MATLAB hints . . . . .	61
6.5 Exercises . . . . .	62
<b>7 Control design by loop-shaping</b>	<b>65</b>
7.1 The loop-shaping design method . . . . .	65
7.2 Open-loop vs. closed-loop specifications . . . . .	65
7.3 Open-loop gain shaping . . . . .	69
7.4 Exercises . . . . .	69
<b>III LQG/LQR Controller Design</b>	<b>73</b>
<b>LQG/LQR Controller Design</b>	<b>75</b>
<b>8 Review of State-space models</b>	<b>77</b>
8.1 State-space models . . . . .	77
8.2 Input-output relations . . . . .	78
8.3 Realizations . . . . .	79
8.4 Controllability and observability . . . . .	79
8.5 Stability . . . . .	80
8.6 MATLAB hints . . . . .	80
<b>9 Linear Quadratic Regulation (LQR)</b>	<b>83</b>
9.1 Feedback configuration . . . . .	83
9.2 Optimal Regulation . . . . .	84
9.3 State-Feedback LQR . . . . .	85
9.4 Stability and Robustness . . . . .	86
9.5 Loop-shaping control using LQR . . . . .	88
9.6 MATLAB hints . . . . .	90
9.7 To probe further . . . . .	91
9.8 Exercises . . . . .	92
<b>10 LQG/LQR Output Feedback</b>	<b>93</b>
10.1 Output Feedback . . . . .	93
10.2 Full-order observers . . . . .	93
10.3 LQG estimation . . . . .	94
10.4 LQG/LQR output feedback . . . . .	95
10.5 Separation Principle . . . . .	96
10.6 Loop-gain recovery . . . . .	96
10.7 MATLAB hints . . . . .	98
10.8 Exercises . . . . .	98

<b>11 Set-point control</b>	<b>99</b>
11.1 Nonzero equilibrium state and input . . . . .	99
11.2 State-feedback . . . . .	100
11.3 Output-feedback . . . . .	101
11.4 To probe further . . . . .	101
<b>IV Nonlinear Control</b>	<b>103</b>
<b>Nonlinear Control</b>	<b>105</b>
<b>12 Feedback linearization controllers</b>	<b>107</b>
12.1 Feedback linearization . . . . .	107
12.2 Generalized model for mechanical systems . . . . .	108
12.3 Feedback linearization of mechanical systems . . . . .	110
12.4 Exercises . . . . .	111
<b>13 Lyapunov stability</b>	<b>113</b>
13.1 Lyapunov stability . . . . .	113
13.2 Lyapunov Stability Theorem . . . . .	115
13.3 Exercise . . . . .	116
13.4 LaSalle’s Invariance Principle . . . . .	116
13.5 Liénard equation and generalizations . . . . .	117
13.6 To probe further . . . . .	119
13.7 Exercises . . . . .	119
<b>14 Lyapunov-based designs</b>	<b>121</b>
14.1 Lyapunov-based controller . . . . .	121
14.2 Exercises . . . . .	122
14.3 Application to mechanical systems . . . . .	123
14.4 Exercises . . . . .	124

**Attention!** When a marginal note finishes with “► p. XXX,” more information about that topic can be found on page XXX.



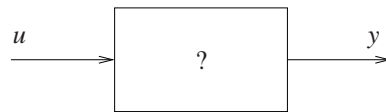
**Part I**

**System Identification**



# Introduction to System Identification

The goal of *system identification* is to utilize input/output experimental data to determine a system's model. For example, one may apply to the system a specific input  $u$ , measure the observed discrete-time output  $y$  and try to determine the system's transfer function (cf. Figure 1).



**Figure 1.** System identification from input/output experimental data

## Pre-requisites

1. Laplace transform, continuous-time transfer functions, stability, and frequency response (briefly reviewed here).
2.  $z$ -transform, discrete-time transfer functions, stability, and impulse/step/frequency responses (briefly reviewed here).
3. Computer-control system design (briefly reviewed here).
4. Familiarity with basic vector and matrix operations.
5. Knowledge of MATLAB<sup>®</sup>/Simulink.



# Chapter 1

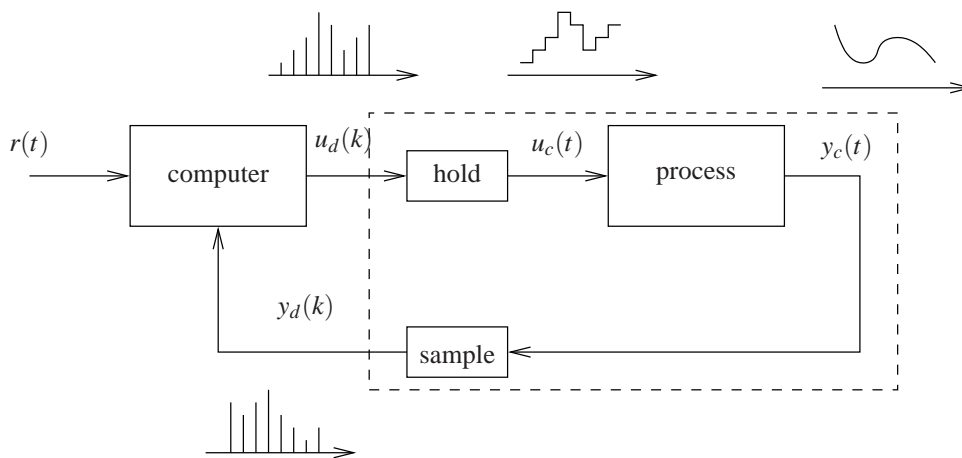
## Computer-Controlled Systems

### Contents

1.1	Computer control	5
1.2	Continuous-time systems	6
1.3	Discrete-time systems	8
1.4	Discrete vs. continuous-time transfer functions	10
1.5	MATLAB® hints	12
1.6	To probe further	13
1.7	Exercise	14

### 1.1 Computer control

Figure 1.1 show the typical block diagram of a control system implemented using a digital computer. Although the output  $y_c(t)$  of most physical systems vary continuously as a function of time, it can



**Figure 1.1.** Computer control architecture. The components in the dashed boxed can be regarded as a discrete-time process to be controlled.

only be measured at discrete time instants. Typically, it is *sampled* periodically as shown in the left plot of Figure 1.2. Moreover, the control signal  $u_c(t)$  cannot be changed continuously and typically is *held* constant between sampling times as shown in the right plot of Figure 1.2. It is therefore sometimes convenient to regard the process to be controlled as a *discrete-time system* whose inputs

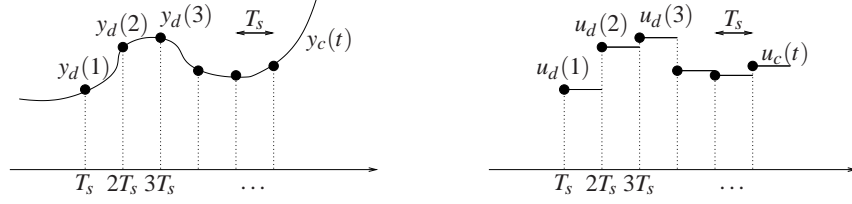


Figure 1.2. Sampling (left) and holding (right)

and outputs are the discrete-time signals

$$y_d(k) = y_c(kT_s), \quad u_d(k) = u_c(kT_s), \quad k \in \{1, 2, 3, \dots\}.$$

The identification techniques that we will study allow us to estimate the transfer function of this discrete-time system. We can then use standard techniques from digital control to (approximately) recover the underlying continuous-time transfer function. This is discussed in Section 1.4 below.

## 1.2 Continuous-time systems

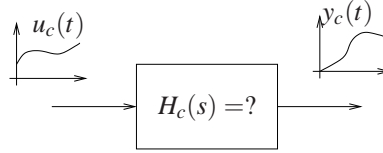


Figure 1.3. Continuous-time system

Consider the continuous-time system in Figure 1.3 and assume that the input and output signals satisfy the following differential equation:

**Notation 1.** We denote by  $\dot{y}_c(t)$  and  $\ddot{y}_c(t)$  the first and second time derivatives of the signal  $y_c(t)$ . Higher-order derivatives of order  $k \geq 3$  are denoted by  $y_c^{(k)}(t)$ .

$$\begin{aligned} y_c^{(n)} + \beta_{n-1}y_c^{(n-1)} + \dots + \beta_2\ddot{y}_c + \beta_1\dot{y}_c + \beta_0y_c \\ = \alpha_m u_c^{(m)} + \alpha_{m-1}u_c^{(m-1)} + \dots + \alpha_2\ddot{u}_c + \alpha_1\dot{u}_c + \alpha_0u_c. \end{aligned} \quad (1.1)$$

**Note 1.** The (unilateral) Laplace transform of a signal  $x(t)$  is given by

$$X(s) := \int_0^{\infty} e^{-st}x(t)dt.$$

See [4, Appendix A] for a review of Laplace transforms.

We recall that, given a signal  $x(t)$  with Laplace transform  $X(s)$ , the Laplace transform of the  $\ell$ th derivative of  $x(t)$  is given by

$$s^\ell X(s) - s^{\ell-1}x(0) - s^{\ell-2}\dot{x}(0) - \dots - x^{(\ell-1)}(0).$$

In particular, when  $x$  and all its derivatives are zero at time  $t = 0$ , the Laplace transform of the  $\ell$ th derivative  $x^{(\ell)}(t)$  simplifies to

$$s^\ell X(s).$$

Taking Laplace transforms to both sides of (1.1) and assuming that both  $y$  and  $u$  are zero at time zero (as well as all their derivatives), we obtain

$$\begin{aligned} s^n Y_c(s) + \beta_{n-1}s^{n-1}Y_c(s) + \dots + \beta_2s^2Y_c(s) + \beta_1sY_c(s) + \beta_0Y_c(s) \\ = \alpha_ms^m U_c(s) + \alpha_{m-1}s^{m-1}U_c(s) + \dots + \alpha_2s^2U_c(s) + \alpha_1sU_c(s) + \alpha_0U_c(s). \end{aligned}$$

This leads to

$$Y_c(s) = H_c(s)U_c(s),$$

where

$$H_c(s) := \frac{\alpha_m s^m + \alpha_{m-1} s^{m-1} + \dots + \alpha_1 s + \alpha_0}{s^n + \beta_{n-1} s^{n-1} + \dots + \beta_1 s + \beta_0}, \quad (1.2)$$

the (*continuous-time*) *transfer function* of the system. The roots of the denominator are called the *poles* of the system and the roots of the numerator are called the *zeros* of the system.

The system is (*BIBO stable*) if all poles have negative real part. In this case, for every input bounded signal  $u_c(t)$ , the output  $y_c(t)$  remains bounded.

### 1.2.1 Steady-state response to sine waves

Suppose that we apply a sinusoidal input with amplitude  $A$  and angular frequency  $\omega$ , given by

$$u_c(t) = A \cos(\omega t), \quad \forall t \geq 0,$$

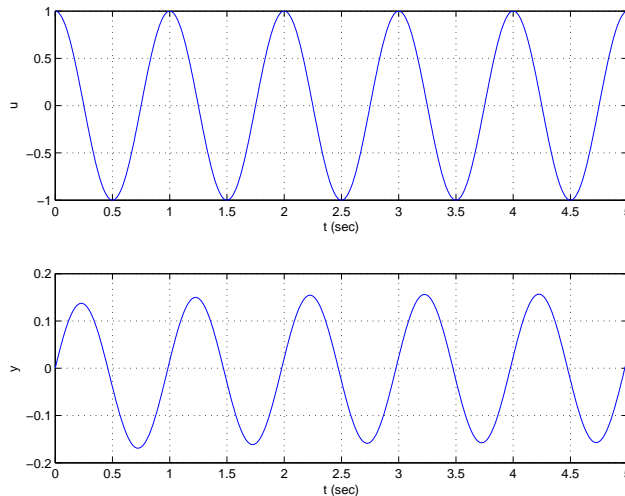
to the system (1.1) with transfer function (1.2). Assuming that the system is BIBO stable, the corresponding output is of the form

$$y_c(t) = \underbrace{gA \cos(\omega t + \phi)}_{\text{steady-state}} + \underbrace{\varepsilon(t)}_{\text{transient}}, \quad \forall t \geq 0, \quad (1.3)$$

where the gain  $g$  and the phase  $\phi$  are given by the following formulas

$$g := |H_c(j\omega)|, \quad \phi := \angle H_c(j\omega), \quad (1.4)$$

and  $\varepsilon(t)$  is a *transient signal* that decays to zero at  $t \rightarrow \infty$  (cf. Figure 1.4).



**Figure 1.4.** Response to a sinusoidal input with frequency  $f = 1\text{Hz}$ , corresponding to  $\omega = 2\pi$  rad/sec. The Bode plot of the system is shown in Figure 1.5.

The *Bode plot* of a transfer function  $H_c(s)$  depicts

1. the norm of  $H_c(j\omega)$  in decibels [dB], which is given by  $20 \log_{10} |H_c(j\omega)|$ ; and
2. the phase of  $H_c(j\omega)$

both as a function of the angular frequency  $\omega$  (in a logarithmic scale). In view of (1.3)–(1.4), the Bode plot provides information about how much a sinusoidal signal is amplified and by how much its phase is changed.

**MATLAB® Hint 1.**  
`tf(num,den)` creates a continuous-time transfer function with numerator and denominator specified by num, den... ▶ p. 12

**MATLAB® Hint 2.**  
`zpk(z,p,k)` creates a continuous-time transfer function with zeros, poles, and gain specified by z, p, k... ▶ p. 12

**MATLAB® Hint 3.**  
`pole(sys)` and `zero(sys)` compute the poles and zeros of a system, respectively.  
**Note 2.** Since there are other notions of stability, one should use the term *BIBO stable* to clarify that it refers to the property that Bounded-Inputs lead to Bounded-Outputs.

**Note 3.** The angular frequency  $\omega$  is measured in radian per second, and is related to the (regular) frequency  $f$  by the formula  $\omega = 2\pi f$ , where  $f$  is given in Hertz or cycles per second.

**MATLAB® Hint 4.**  
`bode(sys)` draws the Bode plot of the system sys... ▶ p. 13

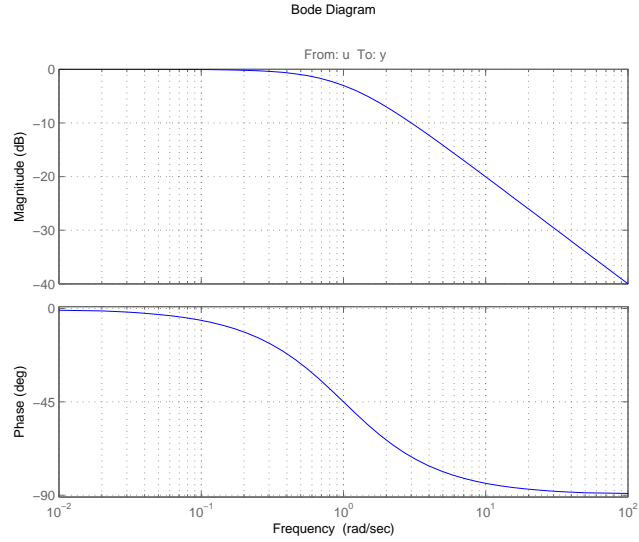


Figure 1.5. Bode plot of a system with transfer function  $H_c(s) = \frac{1}{s+1}$ .

### 1.3 Discrete-time systems

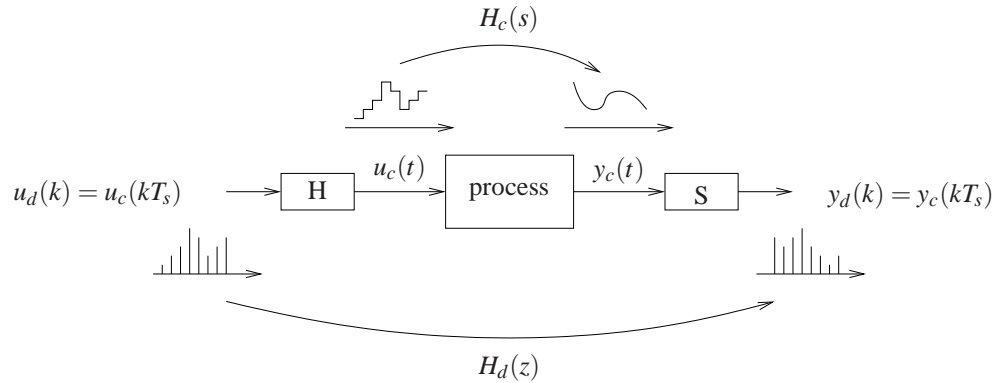


Figure 1.6. Discrete-time system

**Note 4.** Since the output cannot depend on future inputs, we must have  $n \geq m$  for (1.5) to be physically realizable.

Consider now the discrete-time system in Figure 1.6 and assume that the output can be computed recursively by the following difference equation:

$$y_d(k+n) = -\beta_{n-1}y_d(k+n-1) - \dots - \beta_2y_d(k+2) - \beta_1y_d(k+1) - \beta_0y_d(k) + \alpha_mu_d(k+m) + \alpha_{m-1}u_d(k+m-1) + \dots + \alpha_2u_d(k+2) + \alpha_1u_d(k+1) + \alpha_0u_d(k). \quad (1.5)$$

To obtain an equation that mimics (1.1), we can re-write (1.5) as

$$y_d(k+n) + \beta_{n-1}y_d(k+n-1) + \dots + \beta_2y_d(k+2) + \beta_1y_d(k+1) + \beta_0y_d(k) = \alpha_mu_d(k+m) + \alpha_{m-1}u_d(k+m-1) + \dots + \alpha_2u_d(k+2) + \alpha_1u_d(k+1) + \alpha_0u_d(k). \quad (1.6)$$

We recall that, given a signal  $x(k)$  with  $z$ -transform  $X(z)$ , the  $z$ -transform of  $x(k+\ell)$  is given by

$$z^\ell X(z) - z^\ell x(0) - z^{\ell-1}x(1) - \dots - zx(\ell-1).$$

In particular, when  $x$  is equal to zero before time  $k = \ell$ , the  $z$ -transform of  $x(k+\ell)$  simplifies to

$$z^\ell X(z).$$

**Note 5.** The (unilateral)  $z$ -transform of a signal  $x(k)$  is given by

$$X(z) := \sum_{k=0}^{\infty} z^{-k}x(k).$$

See [4, Section 8.2] for a review of Laplace  $z$ -transforms.

Taking  $z$ -transforms to both sides of (1.6) and assuming that both  $y_d$  and  $u_d$  are zero before time  $n \geq m$ , we obtain

$$\begin{aligned} z^n Y_d(z) + \beta_{n-1} z^{n-1} Y_d(z) + \cdots + \beta_2 z^2 Y_d(z) + \beta_1 z Y_d(z) + \beta_0 Y_d(z) \\ = \alpha_m z^m U_d(z) + \alpha_{m-1} z^{m-1} U_d(z) + \cdots + \alpha_2 z^2 U_d(z) + \alpha_1 z U_d(z) + \alpha_0 U_d(z). \end{aligned}$$

This leads to

$$Y_d(z) = H_d(z) U_d(z),$$

where

$$H_d(z) := \frac{\alpha_m z^m + \alpha_{m-1} z^{m-1} + \cdots + \alpha_1 z + \alpha_0}{z^n + \beta_{n-1} z^{n-1} + \cdots + \beta_1 z + \beta_0}, \quad (1.7)$$

is called the (*discrete-time*) *transfer function* of the system. The roots of the denominator are called the *poles* of the system and the roots of the numerator are called the *zeros* of the system.

The system is (*BIBO*) *stable* if all poles have magnitude strictly smaller than one. In this case, for every input bounded signal  $u(k)$ , the output  $y(k)$  remains bounded.

### 1.3.1 Steady-state response to sine waves

Suppose that we apply a sinusoidal input with amplitude  $A$  and discrete-time angular frequency  $\Omega \in [0, \pi]$ , given by

$$u_d(k) = A \cos(\Omega k), \quad \forall k \in \{0, 1, 2, \dots\},$$

to the system (1.1) with transfer function (1.2). Assuming that the system is BIBO stable, the corresponding output is of the form

$$y_d(k) = \underbrace{gA \cos(\Omega k + \phi)}_{\text{steady-state}} + \underbrace{\varepsilon(k)}_{\text{transient}}, \quad \forall k \in \{0, 1, 2, \dots\}, \quad (1.8)$$

where the gain  $g$  and phase  $\phi$  are given by

$$g := |H_d(e^{j\Omega})|, \quad \phi := \angle H_d(e^{j\Omega}), \quad (1.9)$$

and  $\varepsilon(k)$  is a *transient signal* that decays to zero as  $k \rightarrow \infty$  (cf. Figure 1.4).

The *Bode plot* of a transfer function  $H_d(z)$  depicts

1. the norm of  $H_d(e^{j\Omega})$  in decibels [dB], which is given by  $20 \log_{10} |H_d(e^{j\Omega})|$ ; and
2. the phase of  $H_d(e^{j\Omega})$

as a function of the angular frequency  $\Omega$  (in a logarithmic scale). In view of (1.8)–(1.9), the Bode plot provides information about how much a sinusoidal signal is amplified (magnitude) and by how much its phase is changed.

### 1.3.2 Step and impulse responses

Suppose that we apply a discrete-time impulse, given by

$$u(k) = \delta(k) := \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases}$$

#### MATLAB® Hint 5.

`tf(num,den,Ts)` creates a discrete-time transfer function with sampling time  $T_s$  and numerator and denominator specified by `num, den...` ▶ p. 12

#### MATLAB® Hint 6.

`zpk(z,p,k,Ts)` creates a transfer function with sampling time  $T_s$  zeros, poles, and gain specified by `z, p, k...` ▶ p. 12

**Note 6.** Since there are other notions of stability, one should use the term *BIBO stable* to clarify that it refers to the property that Bounded-Inputs lead to Bounded-Outputs.

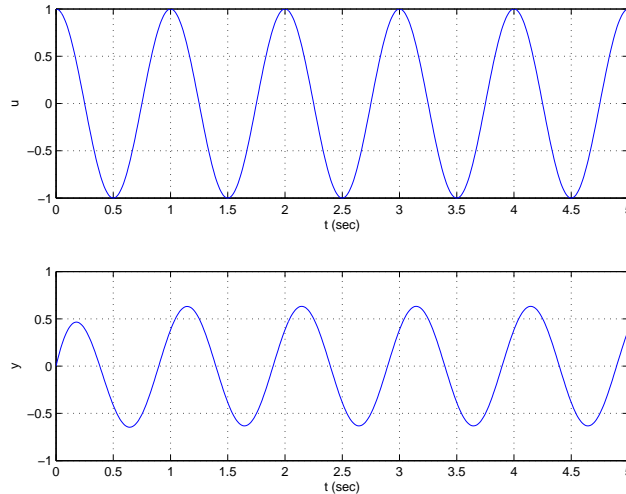
**Note 7.** Discrete-time angular frequencies take values from 0 to  $\pi$ . In particular,  $\Omega = \pi$  corresponds to the “fastest” discrete-time signal:

$$u_d(k) = A \cos(\pi k) = A(-1)^k. \quad \dots \quad \text{▶ p. 13}$$

**Attention!** For discrete-time systems the argument to  $H$  is  $e^{j\Omega}$  and not just  $j\Omega$ , as in continuous-time systems. This means that  $H(z)$  will be evaluated over the unit circle, instead of the imaginary axis.

#### MATLAB® Hint 7.

`bode(sys)` draws the Bode plot of the system `sys...` ▶ p. 13



**Figure 1.7.** Response to a sinusoidal input with frequency  $f = 1\text{Hz}$ ,  $\omega = 2\pi$  rad/sec sampled with a period equal to  $T_s = .01$  sec. The corresponding discrete-time frequency is given by  $\Omega = .02\pi$  and the Bode plot of the system is shown in Figure 1.8.

to the system (1.1) with transfer function (1.2). The corresponding output  $h(k)$  is called the *impulse response* and its  $z$ -transform is precisely the system's transfer function:

$$H_d(z) = \sum_{k=0}^{\infty} z^{-k} h(k).$$

Consider now an input equal to a discrete-time step with magnitude  $\alpha$ , i.e.,

$$u(k) = s(k) := \begin{cases} 0 & k < 0 \\ \alpha & k \geq 0, \end{cases}$$

and let  $y(k)$  be the corresponding output, which is called the *step response*. Since the impulse  $\delta(k)$  can be obtained from the step  $s(k)$  by

$$\delta(k) = \frac{s(k) - s(k-1)}{\alpha}, \quad \forall k \in \{0, 1, 2, \dots\},$$

the impulse response  $h(k)$  (which is the output to  $\delta(k)$ ) can be obtained from the output  $y(k)$  to  $s(k)$  by

$$h(k) = \frac{y(k) - y(k-1)}{\alpha}, \quad \forall k \in \{0, 1, 2, \dots\}.$$

This is a consequence of linearity (cf. Figure 1.9).

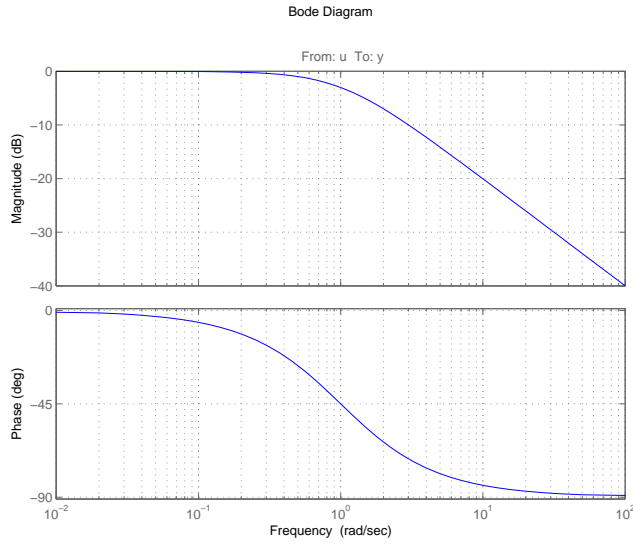
## 1.4 Discrete vs. continuous-time transfer functions

When the sampling time  $T_s$  is small, one can easily go from continuous- to discrete-time transfer functions shown in Figure 1.10. To understand how this can be done, consider a continuous-time integrator

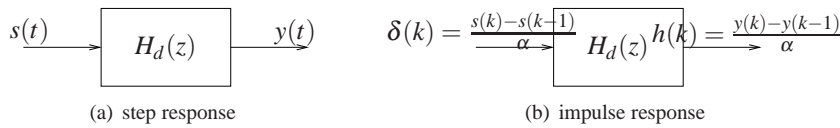
$$\dot{y}_c = u_c \tag{1.10}$$

with transfer function

$$H_c(s) = \frac{Y_c(s)}{U_c(s)} = \frac{1}{s}. \tag{1.11}$$



**Figure 1.8.** Bode plot of a system with transfer function  $H_d(s) = \frac{-0.05}{z - 0.95}$ .



**Figure 1.9.** Impulse versus step responses

Suppose that the signals  $y$  and  $u$  are *sampled* every  $T_s$  time units as shown in Figure 1.2 and that we define the discrete-time signals

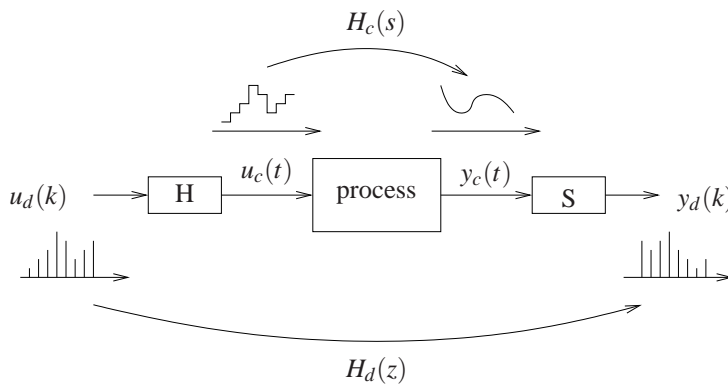
$$y_d(k) = y_c(kT_s), \quad u_d(k) = u_c(kT_s), \quad k \in \{1, 2, 3, \dots\}.$$

Assuming that  $u_c(t)$  is approximately constant over an interval of length  $T_s$ , we can take a finite differences approximation to the derivative in (1.10), which leads to

$$\frac{y_c(t + T_s) - y_c(t)}{T_s} = u_c(t).$$

In particular, for  $t = kT_s$  we obtain

$$\frac{y_d(k + 1) - y_d(k)}{T_s} = u_d(k). \tag{1.12}$$



**Figure 1.10.** Discrete vs. continuous-time transfer functions

**Note 8.** In discrete-time, an integrator has a pole at  $z = 1$ , whereas in continuous-time the pole is at  $s = 0$ .

**MATLAB<sup>®</sup> Hint 8.** `c2d` and `d2c` convert transfer functions from continuous- to discrete-time and vice-versa... ▶ p. 13

**Note 9.** Why?... ▶ p. 14

**Note 10.** The Euler transformation preserves the number of poles and the number of zeros since it replaces a polynomial of degree  $n$  in  $s$  by a polynomial of the same degree in  $z$  and vice-versa. However, the Tustin transformation only preserves the number of poles and can make discrete-time zeros appear, for systems that did not have continuous-time zeros. E.g., the integrator system  $\frac{1}{s}$  becomes  $\frac{T_s(z+1)}{2(z-1)}$ ... ▶ p. 14

Taking the  $z$ -transform we conclude that

$$\frac{zY_d(z) - Y_d(z)}{T_s} = U_d(z) \Leftrightarrow H_d(z) = \frac{Y_d(z)}{U_d(z)} = \frac{1}{\frac{z-1}{T_s}}. \quad (1.13)$$

Comparing (1.11) with (1.13), we observe that we can go directly from a continuous-time transfer function to the discrete-time one using the so called *Euler* transformations:

$$s \mapsto \frac{z-1}{T_s} \Leftrightarrow z \mapsto 1 + sT_s.$$

A somewhat better approximation is obtained by taking the right-hand-side of (1.12) to be the average of  $u_d(k)$  and  $u_d(k+1)$ . This leads to the so called *Tustin or bilinear* transformation:

$$s \mapsto \frac{2(z-1)}{T_s(z+1)} \Leftrightarrow z \mapsto \frac{2 + sT_s}{2 - sT_s}.$$

## 1.5 MATLAB<sup>®</sup> hints

**MATLAB<sup>®</sup> Hint 1, 5** (`tf`). The command `sys_tf=tf(num,den)` assigns to `sys_tf` a MATLAB<sup>®</sup> continuous-time transfer function. `num` is a vector with the coefficients of the numerator of the system's transfer function and `den` a vector with the coefficients of the denominator. The last coefficient must always be the zero-order one. E.g., to get  $\frac{2s}{s^2+3}$  you should use `num=[2 0];den=[1 0 3]`;

The command `sys_tf=tf(num,den,Ts)` assigns to `sys_tf` a MATLAB<sup>®</sup> discrete-time transfer function, with sampling time equal to `Ts`.

For transfer matrices, `num` and `den` are cell arrays. Type `help tf` for examples.

Optionally, one can specify the names of the inputs, outputs, and state to be used in subsequent plots as follows:

```
sys_tf=tf(num,den,...
'InputName',{ 'input1', 'input2', ... },...
'OutputName',{ 'output1', 'output2', ... },...
'StateName',{ 'input1', 'input2', ... })
```

The number of elements in the bracketed lists must match the number of inputs, outputs, and state variables. □

**MATLAB<sup>®</sup> Hint 2, 6** (`zpk`). The command `sys_tf=zpk(z,p,k)` assigns to `sys_tf` a MATLAB<sup>®</sup> continuous-time transfer function. `z` is a vector with the zeros of the system, `p` a vector with its poles, and `k` the gain. E.g., to get  $\frac{2s}{(s+1)(s+3)}$  you should use `z=0;p=[1,3];k=2`;

The command `sys_tf=zpk(z,p,k,Ts)` assigns to `sys_tf` a MATLAB<sup>®</sup> discrete-time transfer function, with sampling time equal to `Ts`.

For transfer matrices, `z` and `p` are cell arrays and `k` a regular array. Type `help zpk` for examples.

Optionally, one can specify the names of the inputs, outputs, and state to be used in subsequent plots as follows:

```
sys_tf=zpk(z,p,k,...
'InputName',{ 'input1', 'input2', ... },...
'OutputName',{ 'output1', 'output2', ... },...
'StateName',{ 'input1', 'input2', ... })
```

The number of elements in the bracketed lists must match the number of inputs, outputs, and state variables. □

**MATLAB® Hint 4, 7** (bode). The command `bode(sys)` draws the Bode plot of the system `sys`. To specify the system one can use:

1. `sys=tf(num,den)`, where `num` is a vector with the coefficients of the numerator of the system's transfer function and `den` a vector with the coefficients of the denominator. The last coefficient must always be the zero-order one. E.g., to get  $\frac{2s}{s^2+3}$  one should use `num=[2 0];den=[1 0 3]`;
2. `sys=zpk(z,p,k)`, where `z` is a vector with the zeros of the system, `p` a vector with its poles, and `k` the gain. E.g., to get  $\frac{2s}{(s+1)(s+3)}$  one should use `z=0;p=[1,3];k=2`;
3. `sys=ss(A,B,C,D)`, where `A,B,C,D` are a realization of the system. □

**MATLAB® Hint 8** (`c2d` and `d2c`). The command `c2d(sysc,Ts,'tustin')` converts the continuous-time LTI model `sysc` to a discrete-time model with sampling time `Ts` using the Tustin transformation. To specify the continuous-time system `sysc` one can use:

1. `sysc=tf(num,den)`, where `num` is a vector with the coefficients of the numerator of the system's transfer function and `den` a vector with the coefficients of the denominator. The last coefficient must always be the zero-order one. E.g., to get  $\frac{2s}{s^2+3}$  you should use `num=[2 0];den=[1 0 3]`;
2. `sysc=zpk(z,p,k)`, where `z` is a vector with the zeros of the system, `p` a vector with its poles, and `k` the gain. E.g., to get  $\frac{2s}{(s+1)(s+3)}$  you should use `z=0;p=[1,3];k=2`;
3. `sysc=ss(A,B,C,D)`, where `A,B,C,D` are a realization of the system.

The command `d2c(sysd,'tustin')` converts the discrete-time LTI model `sysd` to a continuous-time model, using the Tustin transformation. To specify the discrete-time system `sysd` one can use:

1. `sysd=tf(num,den,Ts)`, where `Ts` is the sampling time, `num` is a vector with the coefficients of the numerator of the system's transfer function and `den` a vector with the coefficients of the denominator. The last coefficient must always be the zero-order one. E.g., to get  $\frac{2s}{s^2+3}$  you should use `num=[2 0];den=[1 0 3]`;
2. `sysd=zpk(z,p,k,Ts)`, where `Ts` is the sampling time, `z` is a vector with the zeros of the system, `p` a vector with its poles, and `k` the gain. E.g., to get  $\frac{2s}{(s+1)(s+3)}$  you should use `z=0;p=[1,3];k=2`;
3. `sysd=ss(A,B,C,D,Ts)`, where `Ts` is the sampling time, `A,B,C,D` are a realization of the system. □

## 1.6 To probe further

**Note 7** (Discrete- vs. continuous-time frequencies). When operating with a sample period  $T_s$ , to generate a discrete-time signal  $u_d(k)$  that corresponds to the sampled version of the continuous-time signal

$$u_c(t) = A \cos(\omega t), \quad t \geq 0$$

we must have

$$u_d(k) = u_c(kT_s) = A \cos(\omega kT_s) = A \cos(\Omega k),$$

where the last equality holds as long as we select  $\Omega = \omega T_s = 2\pi f T_s$ .

Discrete-time angular frequencies take values from 0 to  $\pi$ . In particular,  $\Omega = \pi$  results in the “fastest” discrete-time signal:

$$u_d(k) = A \cos(\pi k) = A(-1)^k,$$

which corresponds to a continuous-time frequency  $f = \frac{1}{T_s}$  equal to half the sampling rate, also known as the Nyquist frequency.  $\square$

**Note 9** (Tustin transformation). Consider a continuous-time integrator in (1.10) and assume that  $u(t)$  is approximately linear on the interval  $[kT_s, (k+1)T_s]$ , i.e.,

$$u(t) = \frac{(k+1)T_s - t}{T_s} u_d(k) + \frac{t - kT_s}{T_s} u_d(k+1), \quad t \in [kT_s, (k+1)T_s].$$

Then, for  $y_d(k+1)$  to be exactly equal to the integral of  $u(t)$  at time  $t = (k+1)T_s$ , we need to have

$$\begin{aligned} y_d(k+1) &= y_d(k) + \int_{kT_s}^{(k+1)T_s} \left( \frac{(k+1)T_s - t}{T_s} u_d(k) + \frac{t - kT_s}{T_s} u_d(k+1) \right) dt \\ &= y_d(k) + \frac{T_s}{2} u_d(k) + \frac{T_s}{2} u_d(k+1). \end{aligned}$$

Taking the  $z$ -transform we conclude that

$$\frac{zY_d(z) - Y_d(z)}{T_s} = \frac{U_d(z) + zU_d(z)}{2} \Leftrightarrow H_d(z) = \frac{Y_d(z)}{U_d(z)} = \frac{T_s(1+z)}{2(z-1)}.$$

Comparing this with (1.11), we observe that we can go directly from a continuous-time transfer function to the discrete-time one using the so called *Tustin or bilinear* transformation:

$$s \mapsto \frac{2(z-1)}{T_s(z+1)} \Leftrightarrow z \mapsto \frac{2+sT_s}{2-sT_s}. \quad \square$$

**Note 10** (Number of poles and zeros). The Euler transformation preserves the number of poles and the number of zeros since the rule

$$s \mapsto \frac{z-1}{T_s}$$

replaces a polynomial of degree  $n$  in  $s$  by a polynomial of the same degree in  $z$  and vice-versa. However, the Tustin transformation

$$s \mapsto \frac{2(z-1)}{T_s(z+1)}$$

replaces a polynomial of degree  $n$  in  $s$  by a *ratio* of polynomials of the same degree in  $z$ . This means that it preserves the number of poles, but it can make discrete-time zeros appear, for systems that did not have continuous-time zeros. E.g., the integrator system  $\frac{1}{s}$  becomes

$$\frac{T_s(z+1)}{2(z-1)},$$

which has a pole at  $z = 1$  (as expected), but also a zero at  $z = -1$ .  $\square$

## 1.7 Exercise

**1.1.** Suppose that you sample at 1KHz a unit-amplitude continuous-time sinusoid  $u_c(t)$  with frequency 10Hz. Write the expression for the corresponding discrete-time sinusoid  $u_d(k)$ .

## Chapter 2

# Parametric identification using least-squares

### Contents

2.1	Parametric identification	15
2.2	Least-squares line fitting	15
2.3	Vector least-squares	17
2.4	MATLAB <sup>®</sup> hints	19
2.5	To probe further	19
2.6	Exercises	21

## 2.1 Parametric identification

In *parametric identification* one attempts to determine a finite number of unknown parameters that characterize the model of the system. The following is a typical problem in this class:

**Problem 2.1** (Parametric transfer function identification). Determine the coefficients  $\alpha_i$  and  $\beta_i$  of the system's *rational transfer function*

$$H(z) = \frac{\alpha_m z^m + \alpha_{m-1} z^{m-1} + \cdots + \alpha_1 z + \alpha_0}{z^n + \beta_{n-1} z^{n-1} + \cdots + \beta_1 z + \beta_0}.$$

The number of poles  $n$  and the number of zeros  $m$  are assumed known.

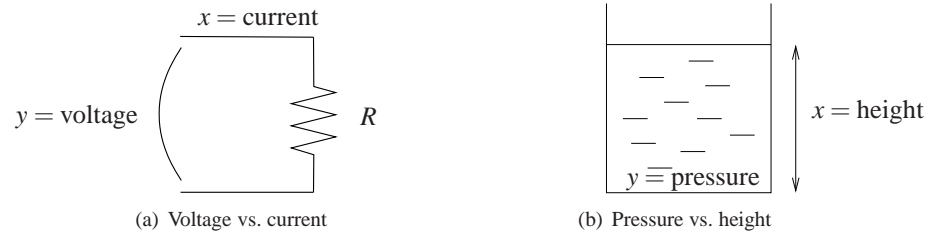
This can be done using the *methods of least-squares*, which we introduce next, starting from a simple line fitting problem and eventually building up to Problem 2.1 in Chapter 3.

The methods of least-squares is generally attributed to Gauss [5] but the American mathematician Adrain [1] seems to have arrived at the same results independently [6]. A detailed treatment of least-squares estimation can be found, e.g., in [7, Chapter 7].

## 2.2 Least-squares line fitting

Suppose that we have reasons to believe that two physical *scalar variables*  $x$  and  $y$  are *approximately related by a linear equation* of the form

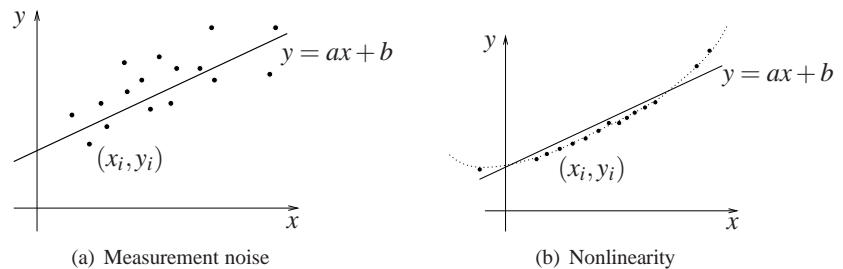
$$y = ax + b, \tag{2.1}$$



**Figure 2.1.** Linear models

but we do not know the value of the constants  $a$  and  $b$ . The variables  $y$  and  $x$  could be, e.g., a voltage and a current in a simple electric circuit, or the height of a fluid in a tank and the pressure at the bottom of the tank (cf. Figure 2.1).

Numerical values for  $a$  and  $b$  can be *determined by conducting several experiments* and measuring the values obtained for  $x$  and  $y$ . We will denote by  $(x_i, y_i)$  the measurements obtained on the  $i$ th experiment of a total of  $N$  experiments. As illustrated in Figure 2.2, it is unlikely that we have



**Figure 2.2.** Main causes for discrepancy between experimental data and linear model

exactly

$$y_i = ax_i + b, \quad \forall i \in \{1, 2, \dots, N\}, \quad (2.2)$$

because of measurement errors and also because often the model (2.1) is only approximate.

Least-squares identification attempts to find values for  $a$  and  $b$  for which the left- and the right-hand-sides of (2.2) *differ by the smallest possible error*. More precisely, the values of  $a$  and  $b$  leading to the smallest possible sum of squares for the errors over the  $N$  experiments. This motivates the following problem:

**Note 11.** Statistical interpretation of least-squares... [► p. 19](#)

**Problem 2.2** (Basic Least-Squares). Find values for  $a$  and  $b$  that minimize the following Sum-of-Squares Error (SSE):

$$SSE := \sum_{i=1}^N (ax_i + b - y_i)^2.$$

*Solution to Problem 2.2.* This problem can be solved using standard calculus. All we need to do is solve

$$\begin{cases} \frac{\partial SSE}{\partial a} = \sum_{i=1}^N 2x_i(ax_i + b - y_i) = 0 \\ \frac{\partial SSE}{\partial b} = \sum_{i=1}^N 2(ax_i + b - y_i) = 0 \end{cases}$$

for the unknowns  $a$  and  $b$ . This is a simple task because it amounts to solving a system of linear equations:

$$\begin{cases} 2\left(\sum_{i=1}^N x_i^2\right)a + 2\left(\sum_{i=1}^N x_i\right)b = 2\sum_{i=1}^N x_i y_i \\ 2\left(\sum_{i=1}^N x_i\right)a + 2Nb = 2\sum_{i=1}^N y_i \end{cases} \Leftrightarrow \begin{cases} \hat{a} = \frac{N(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{N(\sum_i x_i^2) - (\sum_i x_i)^2} \\ \hat{b} = \frac{(\sum_i x_i^2)(\sum_i y_i) - (\sum_i x_i y_i)(\sum_i x_i)}{N(\sum_i x_i^2) - (\sum_i x_i)^2} \end{cases}$$

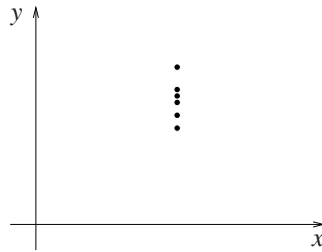
In the above expressions, we used  $\hat{a}$  and  $\hat{b}$  to denote the solution to the least-squares minimization. These are called the *least-squares estimates* of  $a$  and  $b$ , respectively. □

**Attention!** The above estimates for  $a$  and  $b$  are not valid when

$$N(\sum_i x_i^2) - (\sum_i x_i)^2 = 0, \tag{2.3}$$

because this would lead to a division by zero. It turns out that (2.3) only holds when the  $x_i$  obtained in *all experiments* are exactly the same, as shown in Figure 2.3. Such set of measurements does not

**Note 12.** Why?... ► p. 20



**Figure 2.3.** Singularity in least-squares line fitting due to poor data.

provide enough information to estimate the parameter  $a$  and  $b$  that characterize the line defined by (2.1). □

### 2.3 Vector least-squares

The least-square problem defined in Section 2.2 can be extended to a more general linear model, where  $y$  is a scalar,  $z := [z_1 \ z_2 \ \dots \ z_m]$  an  $m$ -vector, and these variables are related by a linear equation of the form

$$y = \sum_{j=1}^m z_j \theta_j = z \cdot \theta, \tag{2.4}$$

where  $\theta := [\theta_1 \ \theta_2 \ \dots \ \theta_m]$  is an  $m$ -vector with parameters whose values we would like to determine.

To determine the parameter vector  $\theta$ , we conduct  $N$  experiments from which we obtain measurements  $(z(i), y(i))$ ,  $i \in \{1, 2, \dots, N\}$ , where each  $z(i)$  denotes one  $m$ -vector and each  $y(i)$  a scalar. The least-squares problem is now formulated as follows:

**Problem 2.3** (Vector Least-Squares). Find values for  $\theta := [\theta_1 \ \theta_2 \ \dots \ \theta_m]$  that minimize the following Sum-of-Squares Error (SSE):

$$SSE := \sum_{i=1}^N (z(i) \cdot \theta - y(i))^2.$$

**Notation 2.**  $a \cdot b$  denotes the inner product of  $a$  and  $b$ ... ► p. 19

**Note 13.** The line fitting problem is a special case of this one for  $z = [x \ 1]$  and  $\theta = [a \ b]$ .

**Attention!** We are now using  $(i)$  to denote the  $i$ th experiment since the subscript in  $z_j$  is needed to denote the  $j$ th entry of the vector  $z$ .

*Solution to Problem 2.3.* This problem can also be solved using standard calculus. Essentially we have to solve

$$\frac{\partial SSE}{\partial \theta_1} = \frac{\partial SSE}{\partial \theta_2} = \dots = \frac{\partial SSE}{\partial \theta_m} = 0,$$

for the unknown  $\theta_i$ . The equation above can also be re-written as

$$\nabla_{\theta} SSE = \left[ \frac{\partial SSE}{\partial \theta_1} \quad \frac{\partial SSE}{\partial \theta_2} \quad \dots \quad \frac{\partial SSE}{\partial \theta_m} \right] = 0.$$

However, to simplify the computation it is convenient to write the SSE in *vector notation*. Defining  $E$  to be an  $N$ -vector obtained by stacking all the errors  $z(i) \cdot \theta - y(i)$  on top of each other, i.e.,

$$E := \begin{bmatrix} z(1) \cdot \theta - y(1) \\ z(2) \cdot \theta - y(2) \\ \vdots \\ z(N) \cdot \theta - y(N) \end{bmatrix}_{N \times 1},$$

we can write

$$SSE = \|E\|^2 = E'E.$$

Moreover, by viewing  $\theta$  as a column vector, we can write

$$E = Z\theta - Y,$$

where  $Z$  denotes a  $N \times m$  matrix obtained by stacking all the row vectors  $z(i)$  on top of each other, and  $Y$  an  $m$ -vector obtained by stacking all the  $y(i)$  on top of each other, i.e.,

$$Z = \begin{bmatrix} z(1) \\ z(2) \\ \vdots \\ z(N) \end{bmatrix}_{N \times m}, \quad Y = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix}_{N \times 1}.$$

Therefore

$$SSE = (\theta'Z' - Y')(Z\theta - Y) = \theta'Z'Z\theta - 2Y'Z\theta + Y'Y. \quad (2.5)$$

It is now straightforward to compute the gradient of the  $SSE$  and find the solution to  $\nabla_{\theta} SSE = 0$ :

$$\nabla_{\theta} SSE = 2\theta'Z'Z - 2Y'Z = 0 \Leftrightarrow \theta' = Y'Z(Z'Z)^{-1}, \quad (2.6)$$

which yields the following *least-squares estimate* for  $\theta$ :

$$\hat{\theta} = (Z'Z)^{-1}Z'Y. \quad (2.7)$$

Since  $Z'Z = \sum_i z(i)'z(i)$  and  $Z'Y = \sum_i z(i)'y(i)$ , we can re-write the above formula as

$$\hat{\theta} = R^{-1}f, \quad R := \sum_{i=1}^N z(i)'z(i), \quad f := \sum_{i=1}^N z(i)'y(i).$$

One can gauge the *quality of the fit* by computing the  $SSE$  achieved by the estimate (i.e., the minimum achievable squared-error). Replacing the estimate (2.7) in (2.5) we obtain

$$\frac{SSE}{\|Y\|^2} = \frac{\|Z\hat{\theta} - Y\|^2}{\|Y\|^2} = 1 - \frac{Y'Z(Z'Z)^{-1}Z'Y}{Y'Y} = 1 - \frac{Y'Z\hat{\theta}}{Y'Y}. \quad (2.8)$$

When this *error-to-signal ratio* is much smaller than one, the mismatch between the left- and right-hand-sides of (2.4) has been made significantly smaller than the output.  $\square$

**Attention!** The above estimate for  $\theta$  is not valid when the  $m \times m$  matrix  $R$  is not invertible. Singularity of  $R$  is a situation similar to (2.3) in line fitting and it means that the  $z(i)$  are not sufficiently rich to estimate  $\theta$ . In practice, even when  $R$  is invertible, poor estimates are obtained if  $R$  is close to a singular matrix.  $\square$

**Notation 3.**  $\nabla_x f(x)$  denotes the gradient of  $f(x)$ . . .  $\blacktriangleright$  p. 19

**Note 14.** Vector notation is convenient both for algebraic manipulations and for efficient MATLAB<sup>®</sup> computations.

**Notation 4.** Given a matrix  $M$ , we denote the transpose of  $M$  by  $M'$ .

**Note 15.** The gradient of a quadratic function  $f(x) = x'Qx + cx + d$ , is given by  $\nabla_x f(x) = x'(Q+Q') + c$ . . .  $\blacktriangleright$  p. 20

**MATLAB<sup>®</sup> Hint 9.**  $Z \setminus Y$  computes directly  $(Z'Z)^{-1}Z'Y$  in a very efficient way, which avoids inverting  $Z'Z$  by solving directly the linear system of equations in (2.6). This is desirable, because this operation is often ill-conditioned.

**MATLAB<sup>®</sup> Hint 10.**  $\text{svd}(A)$  provides a measure of how close  $A$  is to being singular. . .  $\blacktriangleright$  p. 19

## 2.4 MATLAB<sup>®</sup> hints

**MATLAB<sup>®</sup> Hint 10** (svd). The command `svd(A)` computes the singular values of the matrix  $A$ . The *singular values* of a matrix  $A$  are the square-roots of the eigenvalues of  $A'A$ . Singular values are always real and nonnegative.

1. The largest singular value (written  $\sigma_{\max}[A]$ ) provides a measure of how much a matrix can amplify a vector. In particular,

$$\|Ax\| \leq \sigma_{\max}[A]\|x\|.$$

2. The smallest singular value (written  $\sigma_{\min}[A]$ ) provides a measure of how little a matrix can amplify a vector. In particular,

$$\|Ax\| \geq \sigma_{\min}[A]\|x\|.$$

For singular matrices  $\sigma_{\min}[A] = 0$ .

3. The ratio between the largest and smallest singular values is called the *condition number* of a matrix:

$$\rho[A] := \frac{\sigma_{\max}[A]}{\sigma_{\min}[A]}.$$

For singular matrices, the condition number is equal to  $\infty$ . Finite but large condition numbers correspond to matrices that are nonsingular but “close” to being singular, in the sense that they can become singular by making small changes in their entries. Because of this property, the condition number provides a measure of *how close a matrix is to being singular*.

Inverting matrices with large condition numbers is very difficult and often results in large numerical errors due to round-off.  $\square$

## 2.5 To probe further

**Notation 2** (Inner product). Given two  $m$ -vectors,  $a = [a_1 \ a_2 \ \cdots \ a_m]$  and  $b = [a_1 \ a_2 \ \cdots \ a_m]$ ,  $a \cdot b$  denotes the inner product of  $a$  and  $b$ , i.e.,

$$a \cdot b = \sum_{i=1}^m a_i b_i.$$

$\square$

**Notation 3** (Gradient). Given a scalar function of  $n$  variables  $f(x_1, \dots, x_m)$ ,  $\nabla_x f$  denotes the gradient of  $f$ , i.e.,

$$\nabla_x f(x_1, x_2, \dots, x_m) = \left[ \frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \cdots \quad \frac{\partial f}{\partial x_m} \right]. \quad \square$$

**Note 11** (Statistical interpretation of least-squares). Suppose that the mismatch between left- and the right-hand sides of (2.2) is due to uncorrelated zero-mean noise. In particular, that

$$y_i = ax_i + b + n_i, \quad \forall i \in \{1, 2, \dots, n\},$$

where all the  $n_i$  are uncorrelated zero-mean random variables with the same variance  $\sigma^2$ , i.e.,

$$E[n_i] = 0, \quad E[n_i^2] = \sigma^2, \quad E[n_i n_j] = 0, \forall i, j \neq i.$$

The Gauss-Markov Theorem stated below justifies the wide use of least-squares estimation.

**Theorem 2.1** (Gauss-Markov). *The best linear unbiased estimator (BLUE) for the parameters  $a$  and  $b$  is the least-squares estimator.*

Some clarification is needed to understand this theorem

1. The Gauss-Markov Theorem 2.1 compares all linear estimators, i.e., all estimators of the form

$$\hat{a} = \alpha_1 y_1 + \cdots + \alpha_n y_n, \quad \hat{b} = \beta_1 y_1 + \cdots + \beta_n y_n,$$

where the  $\alpha_i, \beta_i$  are coefficients that may depend on the  $x_i$ .

2. It then asks what is the choice for the  $\alpha_i, \beta_i$  that lead to an estimator that is “best” in the sense that it is (i) unbiased, i.e., for which

$$E[\hat{a}] = a, \quad E[\hat{b}] = b,$$

and (ii) results in the smallest possible variance for the estimation error, i.e., that minimizes

$$E[(\hat{a} - a)^2 + (\hat{b} - b)^2].$$

The conclusion is that the least-squares estimator satisfies these requirements.

*Unbiasedness*, means that when we repeat the identification procedure many time, although we may never get  $\hat{a} = a$  and  $\hat{b} = b$ , the estimates obtained will be clustered around the true values. *Minimum variance*, means that the clusters so obtained will be as “narrow” as possible.  $\square$

**Note 12** (Singularity of line fit). The estimates for  $a$  and  $b$  are not valid when

$$N(\sum_i x_i^2) - (\sum_i x_i)^2 = 0.$$

To understand when this can happen let us compute

$$N \sum_i (x_i - \mu)^2 = N \sum_i x_i^2 - 2N\mu \sum_i x_i + n^2 \mu^2$$

but  $N\mu = \sum_i x_i$ , so

$$N \sum_i (x_i - \mu)^2 = N \sum_i x_i^2 - (\sum_i x_i)^2.$$

This means that we run into trouble when

$$\sum_i (x_i - \mu)^2 = 0,$$

which can only occur when all the  $x_i$  are the same and equal to  $\mu$ .  $\square$

**Note 15** (Gradient of a quadratic function). Given a  $m \times m$  matrix  $Q$  and a  $m$ -row vector  $c$ , the gradient of the quadratic function

$$f(x) = x'Qx + cx = \sum_{i=1}^m \sum_{j=1}^m q_{ij} x_i x_j + \sum_{i=1}^m c_i x_i.$$

To determine the gradient of  $f$ , we need to compute:

$$\frac{\partial f(x)}{\partial x_k} = \sum_{i=1}^m \sum_{j=1}^m q_{ij} x_i \frac{\partial x_j}{\partial x_k} + \sum_{i=1}^m \sum_{j=1}^m q_{ij} x_j \frac{\partial x_i}{\partial x_k} + \sum_{i=1}^m c_i \frac{\partial x_i}{\partial x_k}.$$

However,  $\frac{\partial x_i}{\partial x_k}$  and  $\frac{\partial x_j}{\partial x_k}$  are zero whenever  $i \neq k$  and  $j \neq k$ , respectively, and 1 otherwise. Therefore, the summations above can be simplified to

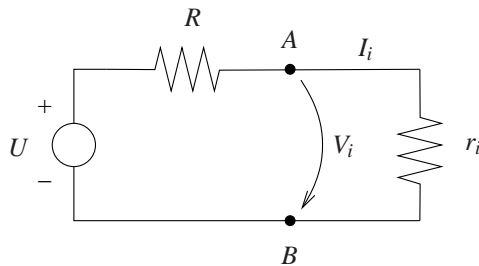
$$\frac{\partial f(x)}{\partial x_k} = \sum_{i=1}^m q_{ik}x_i + \sum_{j=1}^m q_{kj}x_j + c_k = \sum_{i=1}^m (q_{ik} + q_{ki})x_i + c_k.$$

Therefore

$$\begin{aligned} \nabla_x f(x) &= \left[ \frac{\partial f(x)}{\partial x_1} \quad \frac{\partial f(x)}{\partial x_2} \quad \dots \quad \frac{\partial f(x)}{\partial x_m} \right] \\ &= \left[ \sum_{i=1}^m (q_{i1} + q_{1i})x_i + c_1 \quad \dots \quad \sum_{i=1}^m (q_{im} + q_{mi})x_i + c_m \right] \\ &= x'(Q + Q') + c. \end{aligned} \quad \square$$

## 2.6 Exercises

**2.1** (Electrical circuit). Consider the electrical circuit in Figure 2.4, where the voltage  $U$  across the source and the resistor  $R$  are unknown. To determine the values of  $U$  and  $R$  you place several test



**Figure 2.4.** Electrical circuit

resistors  $r_i$  between the terminals  $A, B$  and measure the voltages  $V_i$  and currents  $I_i$  across them.

1. Write a MATLAB<sup>®</sup> script to compute the voltages  $V_i$  and currents  $I_i$  that would be obtained when  $U = 10V$ ,  $R = 1k\Omega$ , and the  $r_i$  are equally spaced in the interval  $[100\Omega, 10k\Omega]$ . Add to the “correct” voltage and current values measurement noise. Use for the currents and for the voltages Gaussian distributed noise with zero mean and standard deviation  $.1mA$  and  $10mV$ , respectively.

The script should take as input the number  $N$  of test resistors.

2. Use least-squares to determine the values of  $R$  and  $U$  from the measurements for  $N \in \{5, 10, 50, 100, 1000\}$ . Repeat each procedure 5 times and plot the average error that you obtained in your estimates versus  $N$ . Use a logarithmic scale. What do you conclude?  $\square$

**2.2** (Nonlinear spring). Consider a nonlinear spring with restoring force

$$F = -(x + x^3),$$

where  $x$  denotes the spring displacement. Use least-squares to determine linear models for the spring of the form

$$F = ax + b.$$

Compute values for the parameters  $a$  and  $b$  for

1. forces evenly distributed in the interval  $[-.1, .1]$ ,
2. forces evenly distributed in the interval  $[-.5, .5]$ , and

3. forces evenly distributed in the interval  $[-1, 1]$ .

For each case

1. calculate the *SSE*,
2. plot the actual and estimated forces vs.  $x$ , and
3. plot the estimation error vs.  $x$ .

□

# Chapter 3

## Parametric identification of an ARX model

### Contents

3.1 ARX Model . . . . .	23
3.2 Identification of an ARX model . . . . .	24
3.3 Known parameters . . . . .	25
3.4 MATLAB® hints . . . . .	25
3.5 To probe further . . . . .	27
3.6 Exercises . . . . .	27

### 3.1 ARX Model

Suppose that we want to determine the transfer function  $H(z)$  of the discrete-time system in Figure 3.1. In least-squares identification, one converts the problem of estimating  $H(z)$  into the vector

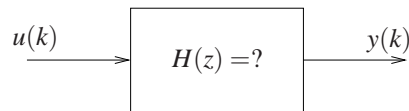


Figure 3.1. System identification from input/output experimental data

least-squares problem considered in Section 2.3. This is done using the ARX model that will be constructed below.

The  $z$ -transforms of the input and output of the system in Figure 3.1 are related by

$$\frac{Y(z)}{U(z)} = H(z) = \frac{\alpha_m z^m + \alpha_{m-1} z^{m-1} + \dots + \alpha_1 z + \alpha_0}{z^n + \beta_{n-1} z^{n-1} + \dots + \beta_1 z + \beta_0}, \tag{3.1}$$

where the  $\alpha_i$  and the  $\beta_i$  denote the coefficients of the numerator and denominator polynomials of  $H(z)$ . Multiplying the numerator and denominator of  $H(z)$  by  $z^{-n}$ , we obtain the transfer function expressed in *negative powers of  $z$* :

$$\begin{aligned} \frac{Y(z)}{U(z)} &= \frac{\alpha_m z^{-n+m} + \alpha_{m-1} z^{-n+m-1} + \dots + \alpha_1 z^{-n+1} + \alpha_0 z^{-n}}{1 + \beta_{n-1} z^{-1} + \dots + \beta_1 z^{-n+1} + \beta_0 z^{-n}} \\ &= z^{-(n-m)} \frac{\alpha_m + \alpha_{m-1} z^{-1} + \dots + \alpha_1 z^{-m+1} + \alpha_0 z^{-m}}{1 + \beta_{n-1} z^{-1} + \dots + \beta_1 z^{-n+1} + \beta_0 z^{-n}} \end{aligned}$$

and therefore

$$Y(z) + \beta_{n-1}z^{-1}Y(z) + \cdots + \beta_1z^{-n+1}Y(z) + \beta_0z^{-n}Y(z) = \alpha_mz^{-n+m}U(z) + \alpha_{m-1}z^{-n+m-1}U(z) + \cdots + \alpha_1z^{-n+1}U(z) + \alpha_0z^{-n}U(z).$$

Taking inverse  $z$ -transforms we obtain

**Note 16.** We can view the difference  $n - m$  between the degrees of the denominator and numerator of (3.1) as a delay, because the output  $y(k)$  at time  $k$  in (3.2) only depends on the value of the input  $u(k - (n - m))$  at time  $k - (n - m)$  and on previous values of the input.

$$y(k) + \beta_{n-1}y(k-1) + \cdots + \beta_1y(k-n+1) + \beta_0y(k-n) = \alpha_mu(k-n+m) + \alpha_{m-1}u(k-n+m-1) + \cdots + \alpha_1u(k-n+1) + \alpha_0u(k-n). \quad (3.2)$$

This can be re-written compactly as

$$y(k) = \varphi(k) \cdot \theta \quad (3.3)$$

where the  $(n + m + 1)$ -vector  $\theta$  contains the coefficient of the transfer function and the vector  $\varphi(k)$  the past inputs and outputs, i.e.,

$$\theta := [-\beta_{n-1} \quad \cdots \quad -\beta_1 \quad -\beta_0 \quad \alpha_m \quad \alpha_{m-1} \quad \cdots \quad \alpha_1 \quad \alpha_0]$$

$$\varphi(k) := [y(k-1) \quad \cdots \quad y(k-n) \quad u(k-n+m) \quad \cdots \quad u(k-n)].$$

The vector  $\varphi(k)$  is called the *regression vector* and the equation (3.3) is called an *ARX model*, a short form of Auto-Regression model with eXogenous inputs.

## 3.2 Identification of an ARX model

We are now ready to solve the system identification Problem 2.1 introduced in Chapter 2, by applying the least-squares method to the ARX model:

*Solution to Problem 2.1.*

1. Apply a probe input signal  $u(k)$ ,  $k \in \{1, 2, \dots, N\}$  to the system.
2. Measure the corresponding output  $y(k)$ ,  $k \in \{1, 2, \dots, N\}$ .
3. Determine the values for the parameter  $\theta$  that minimize the discrepancy between the left- and the right-hand-sides of (3.3) in a least-squares sense.

According to Section 2.3, the least-squares estimate of  $\theta$  is given by

$$\hat{\theta} = (\Phi' \Phi)^{-1} \Phi' Y, \quad (3.4)$$

where

$$\Phi := \begin{bmatrix} \varphi(1) \\ \varphi(2) \\ \vdots \\ \varphi(N) \end{bmatrix} = \begin{bmatrix} y(0) & \cdots & y(1-n) & u(1-n+m) & \cdots & u(1-n) \\ y(1) & \cdots & y(2-n) & u(2-n+m) & \cdots & u(2-n) \\ \vdots & & \vdots & \vdots & & \vdots \\ y(N-1) & \cdots & y(N-n) & u(N-n+m) & \cdots & u(N-n) \end{bmatrix}, \quad Y := \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix},$$

or equivalently

$$\hat{\theta} = R^{-1} f, \quad R := \Phi' \Phi = \sum_{k=1}^N \varphi(k)' \varphi(k), \quad f := \Phi' Y = \sum_{k=1}^N \varphi(k)' y(k).$$

The *quality of the fit* can be assessed by computing the error-to-signal ratio

$$\frac{SSE}{\|Y\|^2} = 1 - \frac{Y' \Phi \hat{\theta}}{Y' Y}.$$

**MATLAB<sup>®</sup> Hint 11.** `PHI\Y` computes  $\hat{\theta}$  directly, from the matrix `PHI=Φ` and the vector `Y=Y`.

**Attention!** When the system is at rest before  $k = 1$ ,  $u(k) = y(k) = 0 \forall k \leq 0$ . Otherwise, if the past inputs are unknown, one needs to remove from  $\Phi$  and  $Y$  all rows with unknown data.

**MATLAB<sup>®</sup> Hint 12.** `arx` is very convenient to perform least-squares identification of ARX models because it does not require the need to explicitly construct the matrix  $\Phi$  and the vector  $Y$ ... ► p. 25

When this quantity is much smaller than one, the mismatch between the left- and right-hand-sides of (3.3) has been made significantly smaller than the output. □

**MATLAB® Hint 13.** compare is useful to validate the quality of the fit... ► p. 27

**Attention!** Two common causes for errors in least-squares identifications of ARX models are:

1. incorrect construction of the matrix  $\Phi$  and/or vector  $Y$ ;
2. incorrect construction of the identified transfer function from the entries in the least-squares estimate  $\hat{\theta}$ .

Both errors can be avoided using the MATLAB® command `arx`.

### 3.3 Known parameters

Due to physical considerations, one often knows one or more poles/zeros of the process. For example:

1. one may know that the process has an integrator, which corresponds to a continuous-time pole at  $s = 0$  and consequently to a discrete-time pole at  $z = 1$ ; or
2. that the process has a differentiator, which corresponds to a continuous-time zero at  $s = 0$  and consequently to a discrete-time zero at  $z = 1$ .

In this case, it suffices to identify the remaining poles/zeros, which can be done as follows: Suppose that it is known that the transfer function  $H(z)$  has a pole at  $z = \lambda$ , i.e.,

$$H(z) = \frac{1}{z - \lambda} \tilde{H}(z),$$

where  $\tilde{H}(z)$  corresponds to the unknown portion of the transfer function. In this case, the  $z$ -transforms of the input and output of the system are related by

$$\frac{Y(z)}{U(z)} = \frac{1}{z - \lambda} \tilde{H}(z),$$

and therefore

$$\frac{\bar{Y}(z)}{U(z)} = \tilde{H}(z),$$

where

$$\bar{Y}(z) := (z - \lambda)Y(z) \quad \Rightarrow \quad \bar{y}(k) = y(k+1) - \lambda y(k). \quad (3.5)$$

This means that we can directly estimate  $\tilde{H}(z)$  by computing  $\bar{y}(k)$  prior to identification and then regarding this variable as the output, instead of  $y(k)$ .

**Attention!** To obtain the original transfer function  $H(z)$ , one needs to multiply the identified function  $\tilde{H}(z)$  by the term  $\frac{1}{z - \lambda}$ . □

**Note 17.** The transfer function  $\tilde{H}(z)$  is proper only if  $H(z)$  is strictly proper. However, even if  $\tilde{H}(z)$  happens not to be proper, this introduces no difficulties for this method of system identification.

**MATLAB® Hint 14.** One must be careful in constructing the vector  $\bar{y}$  in (3.5) to be used by the function `arx`... ► p. 26

### 3.4 MATLAB® hints

**MATLAB® Hint 12** (`arx`). The command `arx` from the identification toolbox performs least-squares identification of ARX models. To use this command one must

1. Create a data object that encapsulates the input/output data using:

```
data=iddata(y,u,Ts);
```

where  $u$  and  $y$  are vectors with the input and output data, respectively, and  $T_s$  is the sampling interval.

Data from multiple experiments can be combined using the command `merge` as follows:

```
dat1=iddata(y1,u1,Ts);
dat2=iddata(y1,u1,Ts);
data=merge(dat1,dat2);
```

2. Compute the estimated model using:

```
model=arx(data,[na,nb,nk])
```

where `data` is the object with input/output data; `na`, `nb`, `nk` are integers that define the degrees of the numerator and denominator of the transfer function (see below); and `model` a MATLAB<sup>®</sup> object with the coefficients of the transfer function. The coefficient are returned for *negative powers of  $z$* . In particular, the estimated transfer function is given by

$$\frac{Y(z)}{U(z)} = \frac{z^{-nk}(\text{model.a}(1) + \text{model.a}(2)z^{-1} + \dots + \text{model.a}(\text{na} + 1)z^{-\text{na}})}{\text{model.b}(1) + \text{model.b}(2)z^{-1} + \dots + \text{model.b}(\text{nb} + 1)z^{-\text{nb}}}. \quad (3.6)$$

The estimated discrete-time transfer function can be recovered using the MATLAB<sup>®</sup> command

```
sysd = tf(model,'Measured')
```

The object `model` contains a lot more information about the estimated transfer function, which includes

1. `model.da` and `model.db` give the standard deviations associated with the estimates of the coefficient.

You should compare the value of each parameter with its standard deviation. A large value in one or more of the standard deviations indicates that the matrix  $R := \Phi' \Phi$  is close to singular and points to little confidence on the estimated value of that parameter.

2. `model.noise` provides a measure of the quality of fit and is essentially the value of *SSE* divided by the number of samples. However, this measure is not normalized by the norm-square of  $Y$ , opposite to (2.8).

You can type `idprops idpoly` at the MATLAB<sup>®</sup> prompt to obtain more information about other information that is available in `model`.

The MATLAB<sup>®</sup> command `arx` uses a more sophisticated algorithm than the one described in these notes so the results obtained using `arx` may not match exactly the ones obtained using the formula (4.8).  $\square$

**MATLAB<sup>®</sup> Hint 14** (Known parameters). One must be careful in constructing the vector  $\bar{y}$  in (3.5) to be used by the function `arx`. In particular, its first element must be equal to

$$y(2) - \lambda y(1),$$

as suggested by (3.5). Moreover, if we had values of  $y(k)$  and  $u(k)$  for  $k \in \{1, 2, \dots, N\}$ , then  $\bar{y}(k)$  will only have values for  $k \in \{1, 2, \dots, N-1\}$ , because the last element of  $\bar{y}(k)$  that we can construct is

$$\bar{y}(N-1) = y(N) - y(N-1).$$

Since the function `iddata` only takes input/output pairs of the same length, this means that we need to discard the last input data  $u(N)$ . The following MATLAB<sup>®</sup> commands could be used construct  $\bar{y}$  from  $y$  and also to discard the last element of  $u$ :

**Note 19.** MATLAB<sup>®</sup> is an object oriented language and `model.a` is the property `a` of the object `model`.

**Note 20.** Typically, one chooses `nb` equal to the (expected) number of poles, `nk` equal to the (expected) input delay, and `na = nb - nk`, which would be the corresponding number of zeros. In this case,  $y(k)$  is a function of  $y(k-1), \dots, y(k-nb)$  and of  $u(k-nk), \dots, u(k-nb)$ . See equations (3.1)–(3.2)...  $\blacktriangleright$  p. 23

**Note 21.** Obtaining a standard deviation for one parameter that is comparable or larger than its estimated value typically arises in one of three situations...

`bary=y(2:end)-lambda*y(1:end-1); u=u(1:end-1);` □

**MATLAB<sup>®</sup> Hint 13** (`compare`). The command `compare` from the identification toolbox allows one to compare experimental outputs with simulated values from an estimated model. This is useful to validate model identification. The command

`compare(data,model)`

plots the measured output from the input/output data object `data` and the predictions obtained from the estimated model `model`. See MATLAB<sup>®</sup> hints 12, 21 for details on these objects. □

### 3.5 To probe further

**Note 21** (Large standard deviations for the parameter estimates). Obtaining a standard deviation for one parameter that is comparable or larger than its estimated value typically arises in one of three situations:

1. The data collected is not sufficiently rich. This issue is discussed in detail in Section 4.1. It can generally be resolved by choosing a different input signal  $u$  for the identification procedure.
2. One has hypothesized an incorrect value for the number of poles, the number of zeros, of the system delay. This issue is discussed in detail in Section 4.4. It can generally be resolved by one of three options:
  - When encounters small estimated value for parameters corresponding to the terms in the *denominator* of (3.6) with the *most negative powers in  $z$* , this may be resolved by selecting a smaller value for  $n_b$ ;
  - When encounters small estimated value for parameters corresponding to the terms in the *numerator* of (3.6) with the *most negative powers in  $z$* , this may be resolved by selecting a smaller value for  $n_a$ ;
  - When encounters small estimated value for parameters corresponding to the terms in the *numerator* of (3.6) with the *least negative powers in  $z$* , this may be resolved by selecting a large value for  $n_k$ .

3. One is trying to identify a parameter whose value is actually zero or very small.

When the parameter is one of the leading/trailing coefficients of the numerator/denominator polynomials, this can be addressed as discussed in the previous bullet. Otherwise, generally there is not much one can do about it during system identification. However, since there is a fair chance that the value of this parameter has a large percentage error (perhaps even the wrong sign), we must make sure that whatever controller we design for this system is robust with respect to errors in such parameter. This issue is addressed in Chapters 6–7. □

### 3.6 Exercises

**3.1** (Known zero). Suppose that the process is known to have a zero at  $z = \gamma$ , i.e., that

$$H(z) = (z - \gamma)\bar{H}(z),$$

where  $\bar{H}(z)$  corresponds to the unknown portion of the transfer function. How would you estimate  $\bar{H}(z)$ ? What if you know that the process has both a zero at  $\gamma$  and a pole at  $\lambda$ ? □

**3.2** (Selected parameters). The data provided was obtained from a system with transfer function

$$H(z) = \frac{z - .5}{(z - .3)(z - p)},$$

where  $p$  is unknown. Use the least-squares method to determine the value of  $p$ . □



# Chapter 4

## Practical considerations in parametric identification

### Contents

4.1	Choice of inputs	29
4.2	Signal scaling	33
4.3	Choice of sampling frequency	36
4.4	Choice of model order	37
4.5	Combination of multiple experiments	39
4.6	Exercises	41

### 4.1 Choice of inputs

The quality of least-squares estimates depends significantly on the input  $u(k)$  used. The choice of inputs should take into account the following requirements:

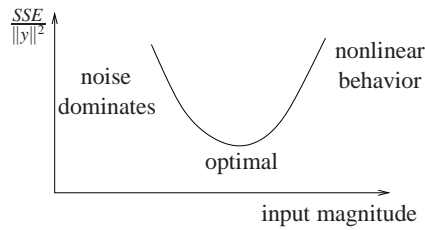
1. The input should be *sufficiently rich* so that the matrix  $R$  is nonsingular (and is well conditioned).
  - (a) Single sinusoids should be avoided because a sinusoid of frequency  $\Omega$  will not allow us to distinguish between different transfer functions with exactly the same value at the point  $z = e^{j\Omega}$ .
  - (b) Good inputs include: square-waves, the sum of many sinusoids, or a chirp signal (i.e., a signal with time-varying frequency).

2. The *amplitude of the resulting output*  $y(k)$  should be much larger than the measurement noise. In particular, large inputs are needed when one wants to probe the system at frequencies for which the noise is large.

However, when the process is nonlinear but is being approximated by a linear model, large inputs may be problematic because the linearization may not be valid. As shown in Figure 4.1, there is usually an “optimal” input level that needs to be determined by trial-and-error.  $\square$

3. The input should be *representative* of the class of inputs that are expected to appear in the feedback loop.
  - (a) The input should have strong components in the range of frequencies at which the system will operate.
  - (b) The magnitude of the input should be representative of the magnitudes that are expected in the closed loop.

**Note 22.** Why?  $H_1(z) = \frac{1}{z-1}$  and  $H_2(z) = \frac{z+2}{z-3}$  have the same value for  $z = e^{j\pi/2} = j$ . Therefore they will lead to the same  $y(k)$  when  $u(k)$  is a sinusoid with frequency  $\pi/2$ . This input will not allow us to distinguish between  $H_1$  and  $H_2$ .

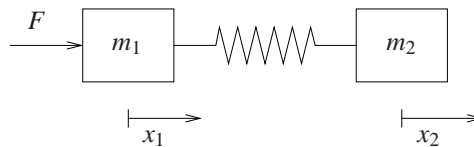


**Figure 4.1.** Optimal choice of input magnitude

**Validation.** The choice of input is perhaps the most critical aspect in good system identification. After any identification procedure the following steps should be followed to make sure that the data collected is adequate:

1. Check if the matrix  $R$  is far from singular. If not a different “richer” input should be used.
2. Check the quality of fit by computing  $\frac{SSE}{\|Y\|^2}$ . If this quantity is not small, one of the following is probably occurring:
  - (a) There is too much noise and the input magnitude needs to be increased.
  - (b) The inputs are outside the range for which the process is approximately linear and the input magnitude needs to be decreased.
  - (c) The assumed degrees for the numerator and denominator are incorrect (see Section 4.4).
3. After an input signal  $u(k)$  passed the checks above, repeat the identification experiment with the input  $\alpha u(k)$  with  $\alpha = -1$ ,  $\alpha = .5$ . If the process is in the linear region, the measured outputs should roughly be equal to  $\alpha y(k)$  and the two additional experiments should approximately result in the same transfer function. When one obtains larger gains in the experiment with  $\alpha = .5$ , this typically means that the process is saturating and one needs to decrease the magnitude of the input.

**Example 4.1** (Two-cart with spring). To make these concepts concrete, we will use as a running example the identification of the two-carts with spring apparatus shown in Figure 4.2. From Newton’s



**Figure 4.2.** Two-mass with spring

law one concludes that

$$m_1 \ddot{x}_1 = k(x_2 - x_1) + f, \quad m_2 \ddot{x}_2 = k(x_1 - x_2), \quad (4.1)$$

where  $x_1$  and  $x_2$  are the positions of both carts,  $m_1$  and  $m_2$  the masses of the carts, and  $k$  the spring constant. The force  $f$  is produced by an electrical motor driven by an applied voltage  $v$  according to

$$f = \frac{K_m K_g}{R_m r} \left( v - \frac{K_m K_g}{r} \dot{x}_1 \right), \quad (4.2)$$

where  $K_m$ ,  $K_g$ ,  $R_m$ , and  $r$  are the motor parameters.

For this system, we are interested in taken the control input to be the applied voltage  $u := v$  and the measured output to be the position  $y := x_2$  of the second cart. To determine the system’s transfer

**MATLAB® Hint 12.** When using the `arx` command, the singularity of the matrix  $R$  can be inferred from the standard deviations associated with the parameter estimates... ▶ p. 25,27

**MATLAB® Hint 12.** When using the `arx` command, the quality of fit can be inferred from the `noise` field in the estimated model... ▶ p. 25

function, we replace  $f$  from (4.2) into (4.1) and take Laplace transforms to conclude that

$$\begin{aligned}
 & \begin{cases} m_1 s^2 X_1(s) = k(Y(s) - X_1(s)) + \frac{K_m K_g}{R_m r} \left( U(s) - \frac{K_m K_g}{r} s X_1(s) \right) \\ m_2 s^2 Y(s) = k(X_1(s) - Y(s)) \end{cases} \\
 & \Rightarrow \begin{cases} \left( m_1 s^2 + \frac{K_m^2 K_g^2}{R_m r^2} s + k \right) X_1(s) = kY(s) + \frac{K_m K_g}{R_m r} U(s) \\ (m_2 s^2 + k) Y(s) = kX_1(s) \end{cases} \\
 & \Rightarrow (m_2 s^2 + k) \left( m_1 s^2 + \frac{K_m^2 K_g^2}{R_m r^2} s + k \right) Y(s) = k^2 Y(s) + \frac{k K_m K_g}{R_m r} U(s) \\
 & \qquad \qquad \qquad \frac{Y(s)}{U(s)} = \frac{\frac{k K_m K_g}{R_m r}}{(m_2 s^2 + k) \left( m_1 s^2 + \frac{K_m^2 K_g^2}{R_m r^2} s + k \right) - k^2} \quad (4.3)
 \end{aligned}$$

where the large caps signals denote the Laplace transforms of the corresponding small caps signals.

From the first principles model derived above, we expect this continuous-time system to have 4 poles and no zero. Moreover, a close look at the denominator of the transfer function in (4.3) reveals that there is no term of order zero, since the denominator is equal to

$$(m_2 s^2 + k) \left( m_1 s^2 + \frac{K_m^2 K_g^2}{R_m r^2} s + k \right) - k^2 = m_2 s^2 \left( m_1 s^2 + \frac{K_m^2 K_g^2}{R_m r^2} s + k \right) + k m_1 s^2 + \frac{k K_m^2 K_g^2}{R_m r^2} s$$

and therefore the system should have one pole at zero, i.e., it should contain one pure integrator.

Our goal now is to determine the system's transfer function (4.3) using system identification from input/output measurements. The need for this typically stems from the fact that

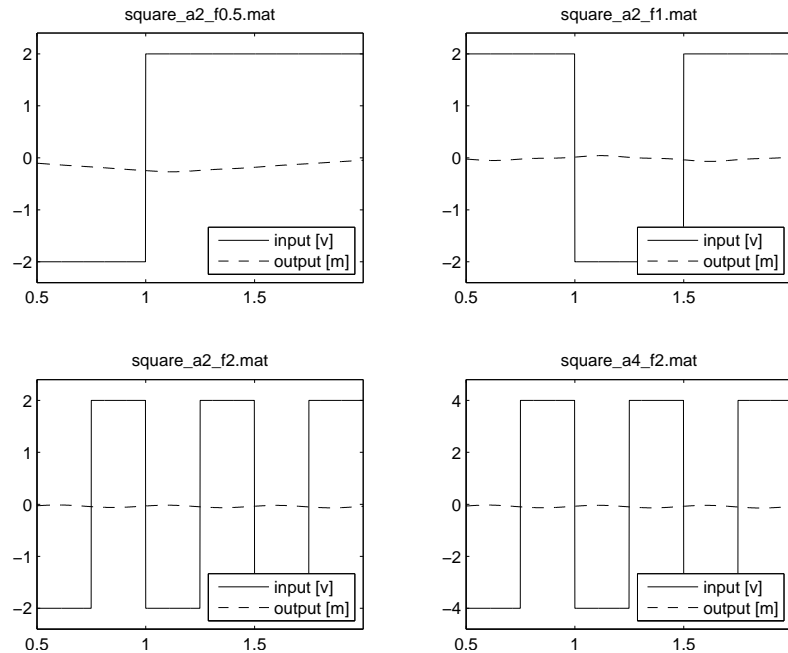
1. we may not know the values of the system parameters  $m_1$ ,  $m_2$ ,  $k$ ,  $k_m$ ,  $K_g$ ,  $R_m$ , and  $r$ ; and
2. the first-principles model (4.1)–(4.2) may be ignoring factors that turn out to be important, e.g., the mass of the spring itself, friction between the masses and whatever platform is supporting them, the fact that the track where the masses move may not be perfectly horizontal, etc.

Figure 4.3(a) shows four sets of input/output data and four discrete-time transfer functions identified from these data sets.

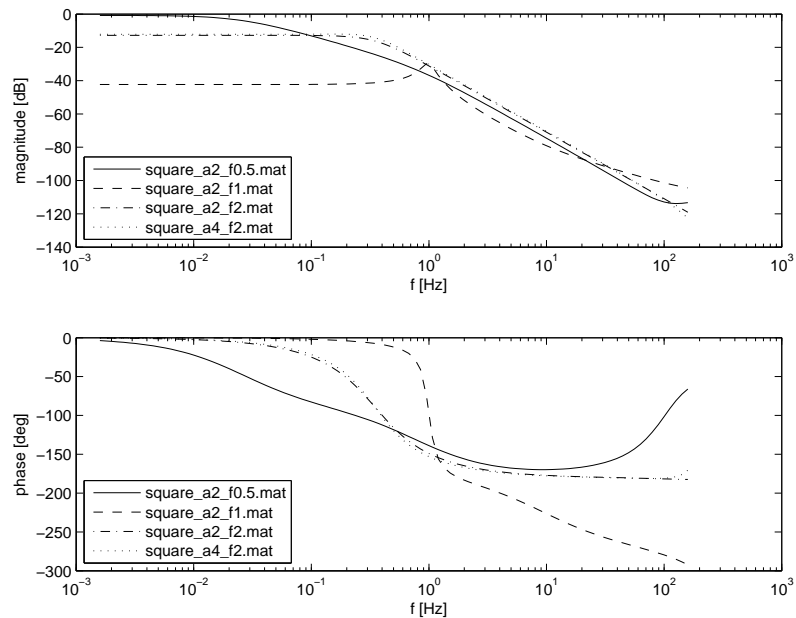
**Key observations.** A careful examination of the outputs of the `arx` command and the plots in Figure 4.3 reveals the following:

1. The quality of fit appears to be very good since `arx` reports a very small value for the `noise` parameter (around  $10^{-10}$ ), which is suspiciously small when compared to the average value of  $y^2$  (around  $10^{-2}$ ).
2. The standard deviations associated with the denominator parameters are small when compared to the parameter estimates (ranging from 2 to 100 times smaller than the estimates).
3. The standard deviations associated with the numerator parameters appear to be worse, often exceeding the estimate, which *indicate problems in the estimated transfer functions*.
4. While difficult to see in Figure 4.3(a), it turns out that the data collected with similar input signals (square wave with a period of 2Hz) but 2 different amplitudes (2v and 4v) resulted in roughly the same shape of the output, but scaled appropriately. However, the fact that this is *difficult to see in Figure 4.3(a) because of scaling is actually an indication of trouble*, as we shall discuss shortly.
5. The integrator that we were expecting in the transfer functions is not there.

**Note 23.** We can make use of the knowledge that the continuous-time system has 4 poles, because the corresponding discrete-time transfer function will have the same number of poles. However, the absence of zeros in the continuous-time transfer functions tell us little about the number of zeros in the corresponding discrete-time transfer function because the Tustin transformation does not preserve the number of zeros (cf. Note 10)... ► p. 14



(a) Four sets of input/output data used to estimate the two-mass system in Figure 4.2. In all experiments the input signal  $u$  is a square wave with frequencies .5Hz, 1Hz, 2Hz, and 2Hz, respectively, from left to right and top to bottom. In the first three plots the square wave switches from  $-2v$  to  $+2v$ , whereas in the last plot it switches from  $-4v$  to  $+4v$ . All signals were sampled at 1KHz.



(b) Four discrete-time transfer functions estimated from the four sets of input/output data in (a), sampled at 1KHz. The transfer functions were obtained using the MATLAB<sup>®</sup> command `arx` with  $n_a = 4$ ,  $n_b = 3$ , and  $n_k = 1$ , which reflects an expectation of 4 poles and a delay of one sampling period. The one period delay from  $u(k)$  to  $y(k)$  is natural since a signal applied at the sampling time  $k$  will not affect the output at the same sampling time  $k$  (See Note 20, p. 26). The labels in the transfer functions refer to the titles of the four sets of input/output data in (a).

**Figure 4.3.** Initial attempt to estimate the discrete-time transfer function for the two-mass system in Figure 4.2.

6. The four sets of data input/output data resulted in *dramatically different identified transfer functions*.

The last three items are, of course, of great concern, and a clear indication that we are not getting a good transfer function.

**Fixes.** The fact that we are not seeing an integrator in the identified transfer functions is not too surprising, since our probe input signals are periodic square waves, which has no component at the zero frequency. The input that has the most zero-frequency component is the square wave with frequency .5Hz (cf. top-left plot in Figure 4.3(a)), for which less than a full period of data was applied to the system. Not surprisingly, that data resulted in the transfer function that is closer to an integrator (i.e., it is larger at low frequencies).

Since we know that the system has a *structural pole* at  $z = 1$ , we should force it into the model using the technique seen in Section 3.3. Figure 4.4(a) shows the same four sets of input/output data used to estimate the discrete-time transfer function, but the signal labeled “output” now corresponds to the signal  $\bar{y}$  that we encountered in Section 3.3. The new identified transfer functions that appear in Figure 4.4(b) are now much more consistent, mostly exhibiting differences in phase equal to 360 deg. However, the standard deviations associated with the numerator parameters continue to be large, often exceeding the estimate.  $\square$

## 4.2 Signal scaling

For the computation of the least-squares estimate to be numerically well conditioned, it is important that the numerical values of both the inputs and the outputs have roughly the same order of magnitude. Because the units of these variable are generally quite different, this often requires scaling of these variables. It is good practice to scale both inputs and outputs so that *all variable take “normal” values in the same range*, e.g., the interval  $[-1, 1]$ .

**Attention!** After identification, one must then adjust the system gain to reflect the scaling performed on the signals used for identification.

Suppose that one takes the original input/output data set  $u, y$  and constructs a new scaled data set  $\bar{u}, \bar{y}$ , using

$$\bar{u}(k) = \alpha_u u(k), \quad \bar{y}(k) = \alpha_y y(k), \quad \forall k.$$

If one then uses  $\bar{u}$  and  $\bar{y}$  to identify the transfer function

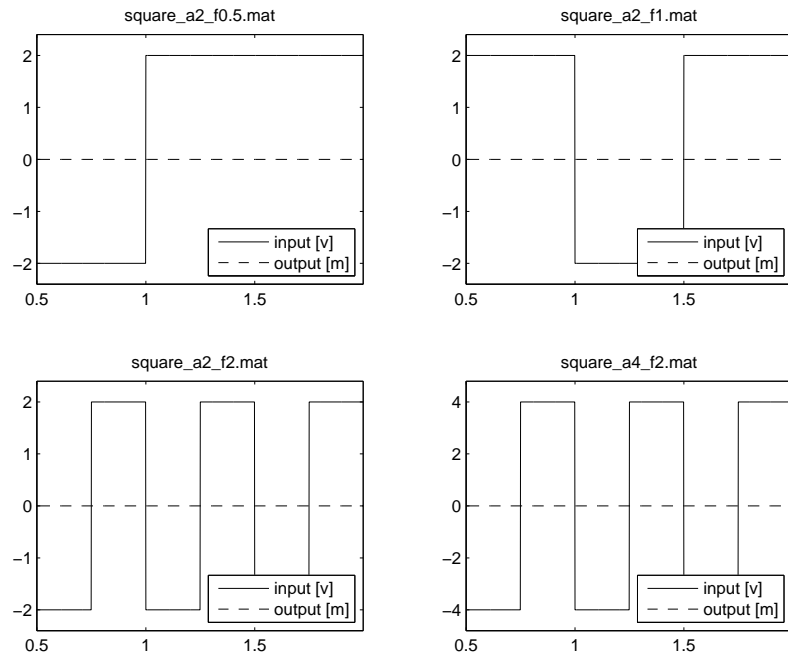
$$\bar{H}(z) = \frac{\bar{U}(z)}{\bar{Y}(z)} = \frac{\alpha_u U(z)}{\alpha_y Y(z)},$$

in order to obtain the original transfer function  $H(z)$  from  $u$  to  $y$ , one need to reverse the scaling:

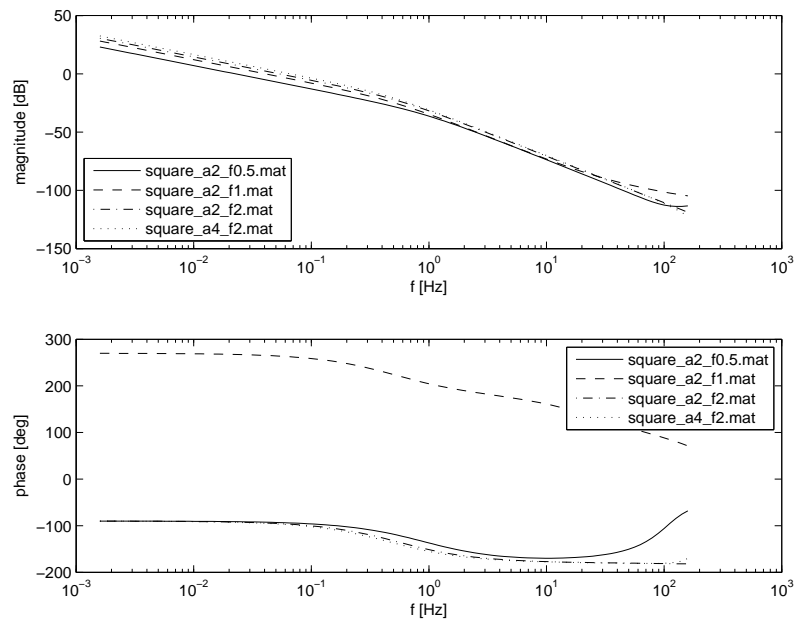
$$H(z) = \frac{U(z)}{Y(z)} = \frac{\alpha_y}{\alpha_u} \bar{H}(z). \quad \square$$

**Example 4.2** (Two-cart with spring (cont.)). We can see in Figure 4.4(a), that the input and output signals used for identification exhibit vastly different scales. In fact, when drawn in the same axis, the output signals appears to be identically zero.

**Fix.** To minimize numerical errors, one can scale the output signal by multiplying it by a sufficiently large number so that it becomes comparable with the input. Figure 4.4(a) shows the same four sets of input/output data used to estimate the discrete-time transfer function, but the signal labeled “output” now corresponds to a scaled version of the signal  $\bar{y}$  that we encountered in Section 3.3.

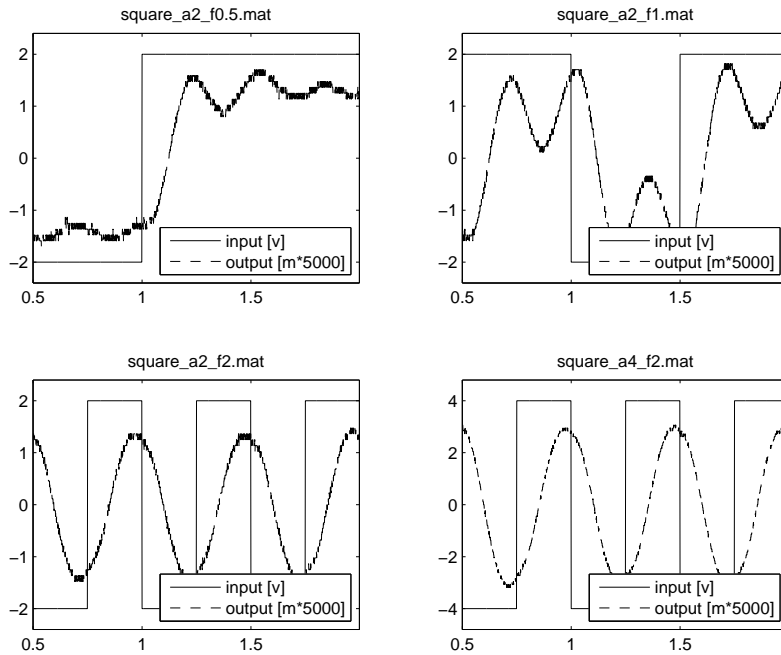


(a) Four sets of input/output data used to estimate the two-mass system in Figure 4.2. In all experiments the input signal  $u$  is a square wave with frequencies .5Hz, 1Hz, 2Hz, and 2Hz, respectively, from left to right and top to bottom. In the first three plots the square wave switches from  $-2v$  to  $+2v$ , whereas in the last plot it switches from  $-4v$  to  $+4v$ . The signal labeled “output” now corresponds to the signal  $\bar{y}$  that we encountered in Section 3.3. All signals were sampled at 1KHz.

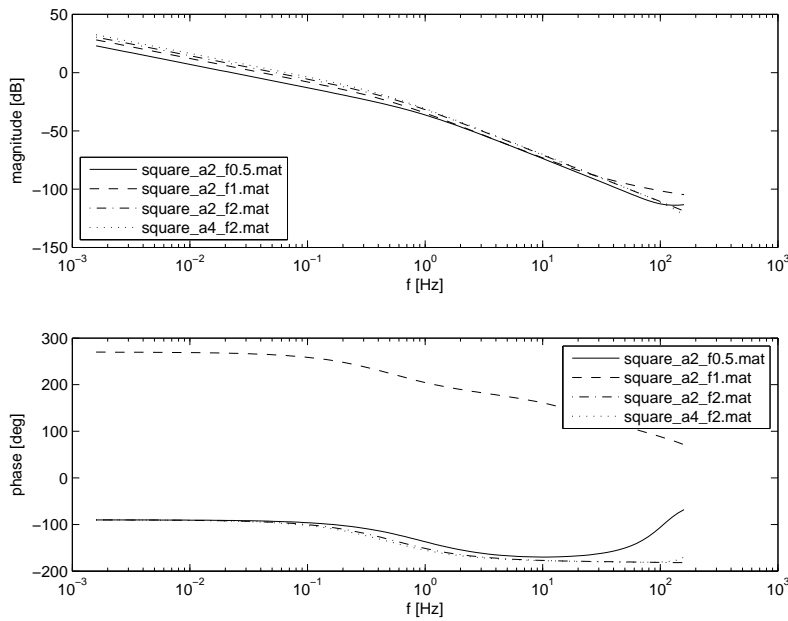


(b) Four discrete-time transfer functions estimated from the four sets of input/output data in (a), sampled at 1KHz. The transfer functions were obtained using the MATLAB<sup>®</sup> command `arx` with  $na = 3$ ,  $nb = 3$ , and  $nk = 0$ , which reflects an expectation of 3 poles in addition to the one at  $z = 1$  and no delay from  $u$  to  $\bar{y}$ . No delay should now be expected since  $u(k)$  can affect  $y(k+1)$ , which appears directly in  $\bar{y}(k)$  (See Note 20, p. 26). A pole at  $z = 1$  was inserted manually into the transfer function returned by `arx`. The labels in the transfer functions refer to the titles of the four sets of input/output data in (a).

**Figure 4.4.** Attempt to estimate the discrete-time transfer function for the two-mass system in Figure 4.2, forcing a pole at  $z = 1$ .



(a) Four sets of input/output data used to estimate the two-mass system in Figure 4.2. In all experiments the input signal  $u$  is a square wave with frequencies .5Hz, 1Hz, 2Hz, and 2Hz, respectively, from left to right and top to bottom. In the first three plots the square wave switches from -2v to +2v, whereas in the last plot it switches from -4v to +4v. The signal labeled “output” now corresponds to a scaled version of the signal  $\bar{y}$  that we encountered in Section 3.3. All signals were sampled at 1KHz.



(b) Four discrete-time transfer functions estimated from the four sets of input/output data in (a), sampled at 1KHz. The transfer functions were obtained using the MATLAB<sup>®</sup> command `arx` with `na = 3`, `nb = 3`, and `nk = 0`, which reflects an expectation of 3 poles in addition to the one at  $z = 1$  and no delay from  $u$  to  $\bar{y}$ . A pole at  $z = 1$  was inserted manually into the transfer function returned by `arx` and the output scaling was reversed back to the original units. The labels in the transfer functions refer to the titles of the four sets of input/output data in (a).

**Figure 4.5.** Attempt to estimate the discrete-time transfer function for the two-mass system in Figure 4.2, forcing a pole at  $z = 1$  and with appropriate output scaling.

**Key observation.** While the transfer functions identified do not show a significant improvements and the standard deviations associated with the numerator parameters continue to be large, Figure 4.4(a) now shows a clue to further problems: the output signals exhibits significant quantization noise.  $\square$

### 4.3 Choice of sampling frequency

For a discrete-time model to accurately capture the process' continuous-time dynamics, the sampling frequency should be as large as possible. However, this often leads to difficulties in system identification.

As we have seen before, least-squares ARX identification amounts to finding the values for the parameters that minimize the sum of squares difference between the two sides of the following equation:

$$y(k) = -\beta_{n-1}y(k-1) - \dots - \beta_1y(k-n+1) + \beta_0y(k-n) + \alpha_mu(k-n+m) + \alpha_{m-1}u(k-n+m-1) + \dots + \alpha_1u(k-n+1) + \alpha_0u(k-n). \quad (4.4)$$

In particular, one is looking for values of the parameters that are optimal at predicting  $y(k)$  based on inputs and outputs that for back to time  $k-n$ .

When the sampling period is very short, all the output values

$$y(k), y(k-1), \dots, y(k-n)$$

that appear in (4.4) are very close to each other and often their difference is smaller than one quantization interval of the analogue-to-digital converter (ADC) In this case, the pattern of output values that appear in (4.4) is mostly determined by the dynamics of the ADC converter and the least-squares ARX model will contain little information about the process transfer function.

As we increase the sampling time, the difference between the consecutive output values increases and eventually dominates the quantization noise. In practice, one wants to chose a sampling rate that is large enough so that the Tustin transformation is valid but sufficiently small so that the quantization noise does not dominate. A good rule of thumb is to sample the system at a frequency *20-40 times larger than the desired closed-loop bandwidth for the system.*

### Down-sampling

It sometimes happens that the hardware used to collect data allow us to sample the system at frequencies much larger than what is needed for identification—*oversampling*. In this case, one generally needs to *down-sample* the signals but this actually provides an *opportunity to remove measurement noise from the signals.*

Suppose that  $y(k)$  and  $u(k)$  have been sampled with a period  $T_{\text{low}}$  but one wants to obtain signals  $\bar{y}(k)$  and  $\bar{u}(k)$  sampled with period  $T_{\text{high}} := LT_{\text{low}}$  where  $L$  is some integer larger than one.

The simplest method to obtain  $\bar{y}(k)$  and  $\bar{u}(k)$  consists of extracting only one out of each  $L$  samples of the original signals:

$$\bar{y}(k) = y(Lk), \quad \bar{u}(k) = u(Lk). \quad (4.5)$$

Instead of discarding all the remaining samples, one can achieve some noise reduction by averaging, as in

$$\bar{y}(k) = \frac{y(Lk-1) + y(Lk) + y(Lk+1)}{3}, \quad (4.6)$$

$$\bar{u}(k) = \frac{u(Lk-1) + u(Lk) + u(Lk+1)}{3}, \quad (4.7)$$

**MATLAB<sup>®</sup> Hint 15.** A signal  $y$  can be down-sampled as in (4.5) using `bary=y(1:L:end)`. Down-sampling can also be achieved with the command `bary=downsample(y,L)`. This command requires MATLAB<sup>®</sup>'s signal processing toolbox.

**MATLAB<sup>®</sup> Hint 16.** A signal  $y$  can be down-sampled as in (4.6) using `bary=(y(1:L:end-2) + y(2:L:end-1) + y(3:L:end))/3`.

or even longer averaging. The down-sampled signals obtained in this way exhibit lower noise than the original ones because noise is being “averaged out.”

**Example 4.3** (Two-cart with spring (cont.)). We can see, e.g., in Figure 4.4(b) that the system appears to have an “interesting” dynamical behavior in the range of frequencies from .1 to 10Hz. “Interesting,” in the sense that the phase varies and the magnitude bode plot is not just a line. Since the sampling frequency of 1kHz lies far above this range, we have the opportunity to down-sample the measured signals and remove some of the measurement noise that was evident in Figure 4.4(a).

**Fix.** To remove some noise quantization noise, we down-sampled the input and output signals by a factor of 10 with the MATLAB<sup>®</sup> `resample` command. The new discrete-time frequency is therefore now sampled only at 100Hz, but this still appears to be sufficiently large. Figure 4.6(a) shows the same four sets of input/output data used to estimate the discrete-time transfer function, down-sampled from the original 1kHz to 100Hz. By comparing this figure with Figure 4.5(a), we can observe a significant reduction in the output noise level.

**Key observation.** The transfer functions identified show some improvement, especially because we are now starting to see some resonance, which would be expected in a system like this. More importantly, now *all the standard deviations associated with the numerator and denominator parameters are reasonably small when compared to the parameter estimates* (ranging from 2.5 to 30 times smaller than the estimates). □

## 4.4 Choice of model order

A significant difficulty in parametric identification of ARX models is that to construct the regression vector  $\varphi(k)$ , one needs to know the degree  $n$  of the denominator. In fact, an incorrect choice for  $n$  will generally lead to difficulties.

1. Selecting a value for  $n$  too small will lead to mismatch between the measured data and the model and the *SSE* will be large.
2. Selecting a value for  $n$  too large is called *over-parameterization* and it generally leads to  $R$  being close to singular. To understand why, suppose we have a transfer function

$$H(z) = \frac{1}{z-1},$$

but for estimation purposes we assumed that  $n = 2$  and therefore attempted to determine constants  $\alpha_i, \beta_i$  such that

$$H(z) = \frac{\alpha_2 z^2 + \alpha_1 z + \alpha_0}{z^2 + \beta_1 z + \beta_0}.$$

If the model was perfect, it should be possible to match the data with any  $\alpha_i, \beta_i$  such that

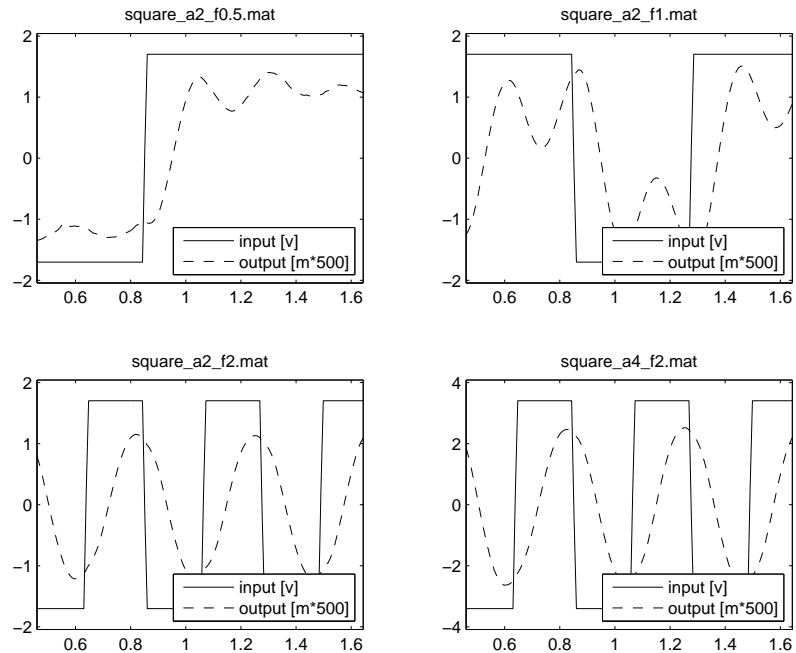
$$\frac{\alpha_2 z^2 + \alpha_1 z + \alpha_0}{z^2 + \beta_1 z + \beta_0} = \frac{z-p}{(z-1)(z-p)} \Leftrightarrow \begin{cases} \alpha_2 = 0, \alpha_1 = 1, \alpha_0 = -p, \\ \beta_1 = -1-p, \beta_0 = p, \end{cases}$$

where  $p$  can be *any number*. This means that the data is not sufficient to determine the values of the parameters  $\alpha_0, \beta_0, \beta_1$ , which translates into  $R$  being singular.

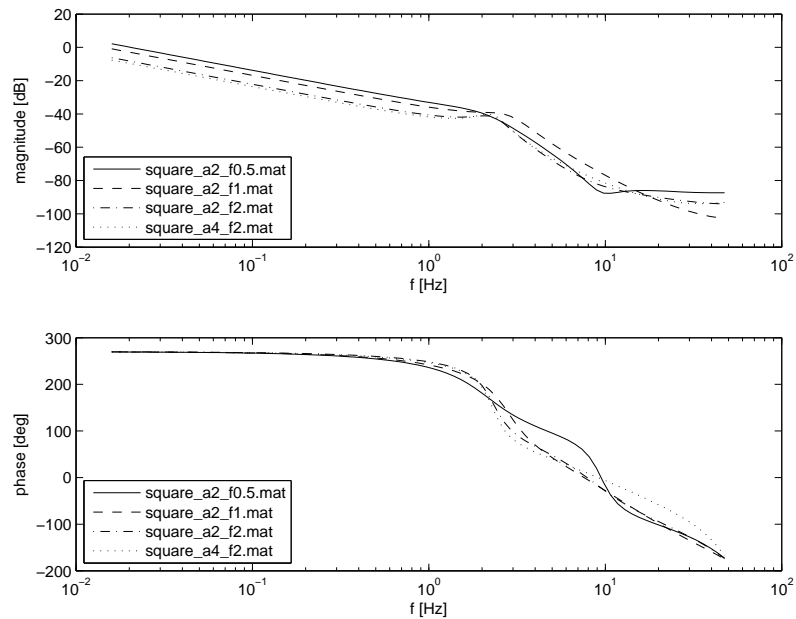
When there is noise, it will never be possible to perfectly explain the data and the smallest *SSE* will always be positive (either with  $n = 1$  or  $n = 2$ ). However, in general, different values of  $p$  will result in different values for *SSE*. In this case, least-squares estimation will produce the specific value of  $p$  that is better at “explaining the noise,” which is not physically meaningful.

**MATLAB<sup>®</sup> Hint 17.** Down-sampling with more sophisticated (and longer) filtering can be achieved with the command `resample`, e.g., `bary=resample(y,1,L,1)`. This command requires MATLAB<sup>®</sup>'s signal processing toolbox.

**MATLAB<sup>®</sup> Hint 18.** When using the `arx` MATLAB<sup>®</sup> command, singularity of  $R$  can be inferred from standard deviations for the parameters that are large when compared with the estimated parameter values... ► p. 27



(a) Four sets of input/output data used to estimate the two-mass system in Figure 4.2. In all experiments the input signal  $u$  is a square wave with frequencies .5Hz, 1Hz, 2Hz, and 2Hz, respectively, from left to right and top to bottom. In the first three plots the square wave switches from  $-2v$  to  $+2v$ , whereas in the last plot it switches from  $-4v$  to  $+4v$ . The signal labeled “output” corresponds to a scaled version of the signal  $\bar{y}$  that we encountered in Section 3.3. All signals were down-sampled from 1 KHz to 100Hz.



(b) Four discrete-time transfer functions estimated from the four sets of input/output data in (a), sampled at 1KHz. The transfer functions were obtained using the MATLAB<sup>®</sup> command `arx` with  $n_a = 3$ ,  $n_b = 3$ , and  $n_k = 0$ , which reflects an expectation of 3 poles in addition to the one at  $z = 1$  and no delay from  $u$  to  $\bar{y}$ . A pole at  $z = 1$  was inserted manually into the transfer function returned by `arx` and the output scaling was reversed back to the original units. The labels in the transfer functions refer to the titles of the four sets of input/output data in (a).

**Figure 4.6.** Attempt to estimate the discrete-time transfer function for the two-mass system in Figure 4.2, forcing a pole at  $z = 1$ , with appropriate output scaling, and with the signals down-sampled to 100Hz.

When one is uncertain about which values to choose for  $m$  and  $n$ , the following procedure should be followed

1. Select for  $n$  the smallest possible integer that is physically meaningful (conceivably 0).
2. Do least-square estimation with  $m = n$ .
  - (a) If the  $SSE$  is large and  $R$  is far from singular increase  $n$  and repeat the identification.
  - (b) If  $R$  is close to singular or the  $SSE$  did not decrease significantly, then  $n$  got too large, go back to the previous  $n$  and stop.

**Note 24.** When the number of poles  $n$  is chosen correctly, a number of zeros  $m$  too large will generally not lead to trouble so one may as well choose the largest one that is physically meaningful. If it is known that there is a delay of at least  $d$ , one should make  $m = n - d$  [cf. equation (3.2)]. . . ▶ p. 24

One needs to exercise judgment in deciding when “the  $SSE$  is large” or when “ $R$  is far/close to singular.” Physical knowledge about the model should play a major role in deciding model orders. Moreover, one should always be very concerned about identifying noise, as opposed to actually identifying the process model.

**Example 4.4** (Two-cart with spring (cont.)). Figure 4.7 shows results obtained for the input/output data set shown in the bottom-right plot of Figure 4.6(a). The procedure to estimate the discrete-time transfer functions was similar to that used to obtain the those in Figure 4.6(b), but we let the parameter  $n_a = n_b$  that defines the number of poles (excluding the integrator at  $z = 1$ ) range from 1 to 16. The delay parameter  $n_k$  was kept equal to 0.

As expected, the  $SSE$  error decreases as we increase  $n_a$ , but the standard deviation of the coefficients increases as we increase  $n_a$ , and rapidly becomes unacceptably large. The decrease of the standard deviation from  $n_a = 4$  to  $n_a = 5$  is suspicious and indicates that, for  $n_a \geq 4$ , one is most likely identifying noise than the system dynamics. These results confirm our expectation that  $n_a = 3$  is a good choice. The corresponding transfer function is shown in Figure 4.7(b).

**Key observation.** While the results improved dramatically with respect to the original ones, the *standard deviations for the parameter estimates are still relatively high*. We can see from Figure 4.7(a) that even for  $n_a = 3$ , at least one standard deviation is still only about 40% of the value of the corresponding parameter. This means that the data used is still not sufficiently rich to achieve a reliable estimate for the transfer function. ◻

## 4.5 Combination of multiple experiments

As discussed in Section 4.1, the input used for system identification should be sufficiently rich to make sure that the matrix  $R$  is nonsingular and also somewhat *representative of the class of all inputs that are likely to appear in the feedback loop*. To achieve this, one could use a single very long input signal  $u(k)$  that contains a large number of frequencies, steps, chirp signals, etc. In practice, this is often difficult so an easier approach is to conduct multiple identification experiments, each with a different input signal  $u_i(k)$ . These experiments result in multiple sets of input/output data that can then be combined to identify a single ARX model.

Suppose, for example, that one wants to identify the following ARX model with two poles and two zeros

$$\frac{Y(z)}{U(z)} = \frac{\alpha_2 z^2 + \alpha_1 z + \alpha_0}{z^2 + \beta_1 z + \beta_0},$$

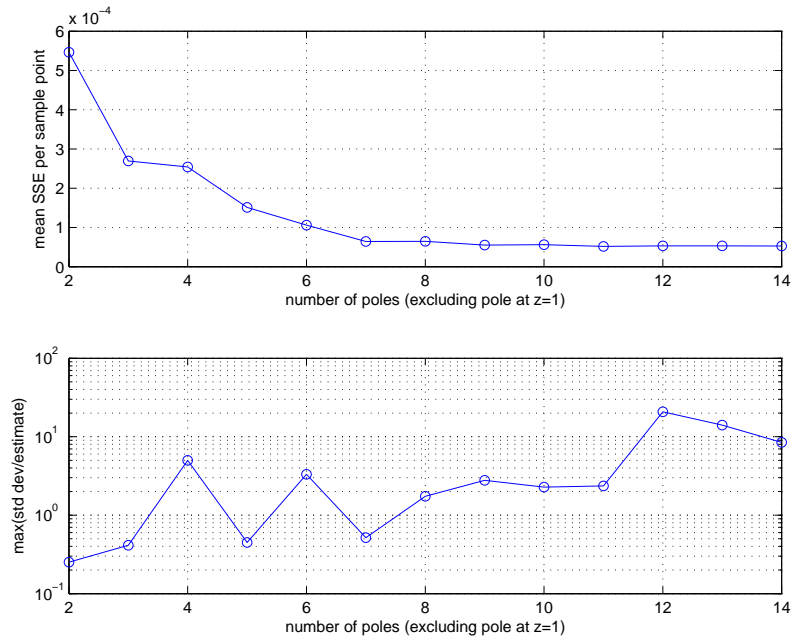
and that we want to accomplish this using on an input/output pair

$$\{(u_1(k), y_1(k)) : k = 1, 2, \dots, N_1\}$$

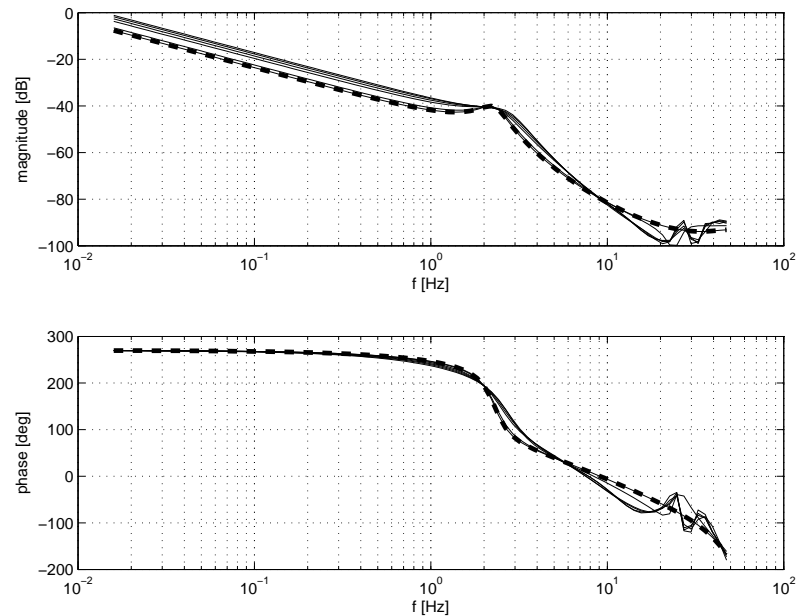
of length  $N_1$  and another input/output pair

$$\{(u_2(k), y_2(k)) : k = 1, 2, \dots, N_2\}$$

**MATLAB® Hint 12.** `merge` allows one to combine data for this type of multi-input processing. . . ▶ p. 25



(a) The y-axis of the top plot shows the average *SSE* error per sample and the y-axis of the bottom plot shows the highest (worst) ratio between a parameter standard deviation and its value. Each point in the plots corresponds to the results obtained for a particular choice of the model order. In particular, to obtain these two plots we varied from 1 to 16 the parameter  $n_a=n_b$  (shown in the x-axis) that defines the number of poles (excluding the integrator at  $z = 1$ ). The delay parameter  $n_k$  was kept the same equal to 0.



(b) Transfer functions corresponding to the identification experiments in (a). The transfer function obtained for  $n_a = 3$ ,  $n_b = 3$ , and  $n_k = 0$  is shown in a thick dashed line.

**Figure 4.7.** Choice of model order. All results in this figure refer to the estimation of the discrete-time transfer function for the two-mass system in Figure 4.2, using the set of input/output data shown in the bottom-right plot of Figure 4.6(a), forcing a pole at  $z = 1$ , with appropriate output scaling, and with the signals down-sampled to 100Hz.

of length  $N_2$ . One would construct

$$\Phi = \begin{bmatrix} y_1(2) & y_1(1) & u_1(3) & u_1(2) & u_1(1) \\ y_1(3) & y_1(2) & u_1(4) & u_1(3) & u_1(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_1(N_1-1) & y_1(N_1-2) & u_1(N_1) & u_1(N_1-1) & u_1(N_1-2) \\ y_2(2) & y_2(1) & u_2(3) & u_2(2) & u_2(1) \\ y_2(3) & y_2(2) & u_2(4) & u_2(3) & u_2(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_2(N_2-1) & y_2(N_2-2) & u_2(N_2) & u_2(N_2-1) & u_2(N_2-2) \end{bmatrix}, \quad Y = \begin{bmatrix} y_1(3) \\ y_1(4) \\ \vdots \\ y_1(N_1) \\ y_2(3) \\ y_2(4) \\ \vdots \\ y_2(N_2) \end{bmatrix}$$

and, according to Section 3.2, the least-squares estimate of

$$\theta := [-\beta_1 \quad -\beta_0 \quad \alpha_m \quad \alpha_2 \quad \alpha_1 \quad \alpha_0]$$

using *all the available data* is still given by

$$\hat{\theta} = (\Phi' \Phi)^{-1} \Phi' Y. \tag{4.8}$$

**MATLAB® Hint 19.** `PHI\Y` computes  $\hat{\theta}$  directly, from the matrix `PHI = Φ` and the vector `Y = Y`.

**Example 4.5** (Two-cart with spring (cont.)). As noted before, we can see in Figure 4.7 that even for  $n_a = 3$ , at least one standard deviation is still only about 40% of the value of the corresponding parameter. This is likely caused by the fact that the all the results in this figure were obtained for the input/output data set shown in the bottom-right plot of Figure 4.6(a). This input data will be excellent to infer the response of the system to square waves of 2Hz, and possibly to other periodic inputs in this frequency range. However, this data set is relatively poor in providing information on the system dynamics below and above this frequency.

**Fix.** By combining the data from the four sets of input/output data shown in Figure 4.6(a), we should be able to decrease the uncertainty regarding the model parameters.

Figure 4.8 shows results obtained by combining all four sets of input/output data shown in Figure 4.6(a). Aside from this change, the results shown follow from the same procedure used to construct the plots in Figure 4.7.

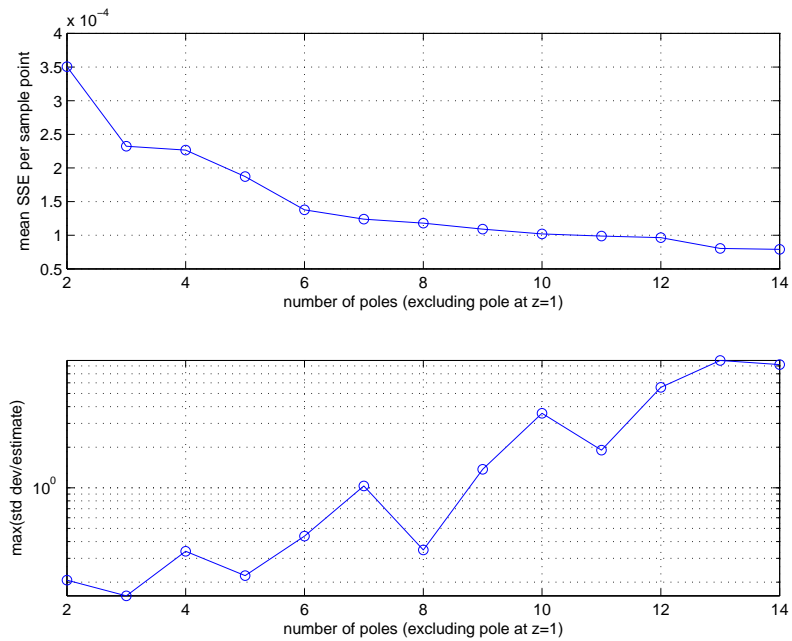
**Key Observation.** As expected, the *standard deviations for the parameter estimates decreased significantly* and, for  $n_a = 3$ , all standard deviations are well below 20% of the values of the corresponding parameter. However, one still needs to combine more inputs to obtain a high-confidence model. In particular, the inputs considered provide relatively little data on the system dynamics above 2-4Hz since the 2Hz square wave contains very little energy above its 2nd harmonic. One may also want to use a longer time horizon to get inputs with more energy at low frequencies.  $\square$

## 4.6 Exercises

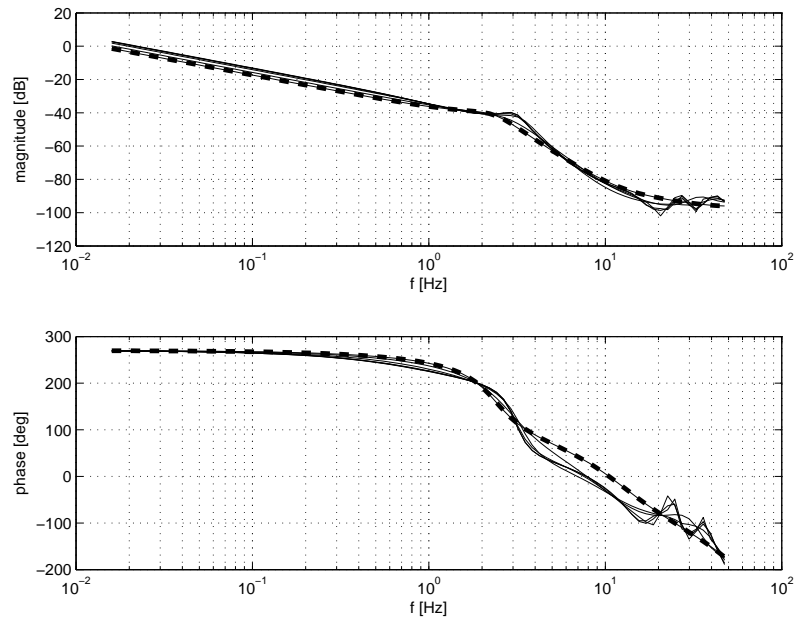
**4.1** (Input magnitude). A Simulink block that models a nonlinear spring-mass-damper system is provided.

1. Use the Simulink block to generate the system's response to step inputs with amplitude 0.25 and 1.0 and no measurement noise.
2. For each set of data, use the least-squares method to estimate the systems transfer function. Try a few values for the degrees of the numerator and denominator polynomials  $m$  and  $n$ . Check the quality of the model by following the validation procedure outlines above.

**Important:** write MATLAB® scripts to automate these procedures. These scripts should take as inputs the simulation data  $u(k)$ ,  $y(k)$ , and the integers  $m$ ,  $n$ .



(a) The y-axis of the top plot shows the average *SSE* error per sample and the y-axis of the bottom plot shows the highest (worst) ratio between a parameter standard deviation and its value. Each point in the plots corresponds to the results obtained for a particular choice of the model order. In particular, to obtain these two plots we varied from 1 to 16 the parameter  $n_a = n_b$  (shown in the x-axis) that defines the number of poles (excluding the integrator at  $z = 1$ ). The delay parameter  $n_k$  was kept the same equal to 0.



(b) Transfer functions corresponding to the identification experiments in (a). The transfer function obtained for  $n_a = 3$ ,  $n_b = 3$ , and  $n_k = 0$  is shown in a thick dashed line.

**Figure 4.8.** Choice of model order. All results in this figure refer to the estimation of the discrete-time transfer function for the two-mass system in Figure 4.2, using all four sets of input/output data shown in Figure 4.6, forcing a pole at  $z = 1$ , with appropriate output scaling, and with the signals down-sampled to 100Hz.

- Use the Simulink block to generate the system's response to step inputs and measurement noise with intensity 0.01.

For the best values of  $m$  and  $n$  determined above, plot the  $SSE$  vs. the step size. Which step-size leads to the best model?

**4.2 (Model order).** Use the data provided to identify the transfer function of the system. Use the procedure outlined above to determine the order of the numerator and denominator polynomials. Plot the largest and smallest singular value of  $R$  and the  $SSE$  as a function of  $n$ .

**Important:** write MATLAB<sup>®</sup> scripts to automate this procedure. These scripts should take as inputs the simulation data  $u(k)$ ,  $y(k)$ , and the integers  $m$ ,  $n$ .

**4.3 (Sampling frequency).** Consider a continuous-time system with transfer function

$$P(s) = \frac{4\pi^2}{s^2 + \pi s + 4\pi^2}.$$

- Build a Simulink model for this system and collect input/output data for an input square wave with frequency .25Hz and unit amplitude for two sampling frequencies  $T_s = .25\text{sec}$  and  $T_s = .0005\text{sec}$ .
- Identify the system's transfer function without down-sampling.
- Identify the system's transfer function using the data collected with  $T_s = .0005\text{sec}$  but down-sampled.

**Important:** write MATLAB<sup>®</sup> scripts to automate this procedure. These scripts should take as inputs the simulation data  $u(k)$ ,  $y(k)$ , the integers  $m$ ,  $n$ , and the down-sampling period  $L$ .

- Briefly comment on the results.



# Chapter 5

## Nonparametric identification

### Contents

5.1	Nonparametric methods	45
5.2	Time-domain identification	45
5.3	Frequency response identification	47
5.4	MATLAB <sup>®</sup> hints	48
5.5	To probe further	49
5.6	Exercises	50

### 5.1 Nonparametric methods

Nonparametric identification attempts to directly determine the model of a system, *without assuming that its transfer function is rational and that we know the number of poles or zeros*. The following are typical problems in this class:

**Problem 5.1** (Nonparametric frequency response identification). Determine the *frequency response*  $H(e^{j\Omega})$  over a range of frequencies  $\Omega \in [\Omega_{\min}, \Omega_{\max}]$ .

or

**Problem 5.2** (Nonparametric time response identification). Determine the *impulse response*  $h(k)$  from time  $k = 0$  to  $k = N$ .

Problem 5.1 is useful for controller design based on loop-shaping or the Nyquist criterion, whereas Problem 5.2 is useful for controller design based on the Ziegler-Nichols rules to tune PID controllers [4]. For  $N$  large, one can also recover the transfer function from the impulse response by taking the  $z$ -transform:

$$H(z) = \mathcal{Z}[h(k)] = \sum_{k=0}^{\infty} h(k)z^{-k} \approx \sum_{k=0}^N h(k)z^{-k},$$

assuming that  $N$  is large enough so that  $h(k) \approx 0$  for  $k > N$ .

Throughout this chapter we will assume that the *system is BIBO stable*, i.e., that all poles of the transfer function  $H(z)$  have magnitude smaller than one or equivalently that  $h(k)$  converges to zero exponentially fast.

A detailed treatment of this subject can be found, e.g., in [7, Chapter 7].

**MATLAB<sup>®</sup> Hint 20.** `fft(h)` computes the fast-Fourier-transform (FFT) of the signal  $h(k)$ ,  $k \in \{1, 2, \dots, N\}$ , which provides the values of  $H(e^{j\Omega})$  for equally spaced frequencies  $\Omega = \frac{2\pi k}{N}$ ,  $k \in \{0, 1, \dots, N-1\}$ . However, since we only care for values of  $\Omega$  in the interval  $[0, \pi]$ , we should discard the second half of `fft`'s output.

### 5.2 Time-domain identification

We start by discussing a few methods used to solve the time response identification Problem 5.2.

### 5.2.1 Impulse response method

**Note 25.** The main *weakness* of the impulse-response method is that impulses are rarely representative of typical inputs that appear in closed loop so, especially for nonlinear systems, we estimate a regimen that may be far from the dynamics that will appear in closed loop.

The impulse response of a system can be determined directly by starting with the system at rest and applying an *impulse at the input*:

$$u(k) = \begin{cases} \alpha & k = 0 \\ 0 & k \neq 0. \end{cases}$$

The output will be a (scaled) version of the impulse response, possibly corrupted by some measurement noise  $n(k)$ :

$$y(k) = \alpha h(k) + n(k).$$

Therefore

$$h(k) = \frac{y(k)}{\alpha} - \frac{n(k)}{\alpha}$$

Hopefully the noise term  $n(k)/\alpha$  is small when compared to  $h(k)$  and one can use the following estimate for the impulse response:

$$\hat{h}(k) = \frac{y(k)}{\alpha}, \quad k \in \{1, 2, \dots, N\}.$$

**Attention!** The choice of  $\alpha$  is generally critical to obtain a good estimate for the impulse response:

- (i)  $|\alpha|$  should be *large* to make sure that  $n(k)/\alpha$  is indeed negligible when compared to  $h(k)$ ;
- (ii)  $|\alpha|$  should be *small* to make sure that the process does not leave the region where a linear model is valid and where it is safe to operate it open-loop.

**Note 26.** Note that the estimate  $\hat{h}(k)$  differs from the true value  $h(k)$ , precisely by:

$$\hat{h}(k) = h(k) + \frac{n(k)}{\alpha}.$$

As one increases  $\alpha$ , a simple practical test to check if one is leaving the linear region of operation is to do identification both with some  $\alpha > 0$  and  $-\alpha < 0$ . In the linear region of operation, the identified impulse-response does not change. See also the discussion in Section 4.1.  $\square$

### 5.2.2 Step response

**Note 27.** In general, steps are more representative than pulses in feedback loops so, in that respect, the step response is a better method than the impulse response.

The impulse response of a system can also be determined by starting with the system at rest and applying an *step at the input*:

$$u(k) = \begin{cases} \alpha & k \geq 0 \\ 0 & k < 0. \end{cases}$$

The  $z$ -transform of the output will be given by

$$Y(z) = H(z)U(z) + N(z) = H(z)\frac{\alpha}{1-z^{-1}} + N(z),$$

where  $N(z)$  denotes the  $z$ -transform of measurement noise. Solving for  $H(z)$  we obtain:

$$H(z) = \frac{Y(z) - z^{-1}Y(z)}{\alpha} - \frac{N(z) - z^{-1}N(z)}{\alpha}$$

Taking inverse  $z$ -transforms, we conclude that

$$h(k) = \frac{y(k) - y(k-1)}{\alpha} - \frac{n(k) - n(k-1)}{\alpha}.$$

Hopefully the noise term  $\frac{n(k)-n(k-1)}{\alpha}$  is small when compared to  $h(k)$ , and we can use following estimate for the impulse response:

$$\hat{h}(k) = \frac{y(k) - y(k-1)}{\alpha}, \quad k \in \{1, 2, \dots, N\}.$$

**Attention!** The main *weakness* of the step-response method is that it can amplify noise because in the worst case  $\frac{n(k)-n(k-1)}{\alpha}$  can be twice as large as  $\frac{n(k)}{\alpha}$  (when  $n(k-1) = -n(k)$ ). Although this may seem unlikely, it is not unlikely to have all the  $n(k)$  independent and identically distributed with zero mean and standard deviation  $\sigma$ . In this case,

$$\text{StdDev} \left[ \frac{n(k)}{\alpha} \right] = \frac{\sigma}{\alpha}, \quad \text{StdDev} \left[ \frac{n(k) - n(k-1)}{\alpha} \right] = \frac{1.41 \sigma}{\alpha},$$

which means that the noise is amplified by approximately 41%.  $\square$

**Note 28.** Why?...  $\blacktriangleright$  p. 49

### 5.2.3 Other inputs

One can determine the impulse response of a system using any input and not just a pulse- or step-input. Take an arbitrary input  $u(k)$ , with  $z$ -transform  $U(z)$ . The  $z$ -transform of the output will be given by

$$Y(z) = H(z)U(z) + N(z),$$

where  $N(z)$  denotes the  $z$ -transform of measurement noise. Solving for  $H(z)$  we obtain:

$$H(z) = U^{-1}(z)(Y(z) - N(z))$$

Taking inverse  $z$ -transforms, we conclude that

$$h(k) = \mathcal{Z}^{-1}[U^{-1}(z)] * (y(k) - n(k)),$$

where the  $*$  denotes the convolution operation. Hopefully the noise term is small when compared to  $h(k)$ , and we can use following estimate for the impulse response:

$$\hat{h}(k) = \mathcal{Z}^{-1}[U^{-1}(z)] * y(k), \quad k \in \{1, 2, \dots, N\}.$$

**MATLAB<sup>®</sup> Hint 21.** The identification toolbox command `impz` performs impulse response estimation for arbitrary inputs...  $\blacktriangleright$  p. 48

## 5.3 Frequency response identification

We now consider a couple of methods used to solve the frequency response identification Problem 5.1.

### 5.3.1 Sine-wave testing

Suppose that one applies an sinusoidal input of the form,

$$u(k) = \alpha \cos(\Omega k), \quad \forall k \in \{1, 2, \dots, N\}. \quad (5.1)$$

Since we are assuming that  $H(z)$  is BIBO stable, the measured output is given by

$$y(k) = \alpha A_{\Omega} \cos(\Omega k + \phi_{\Omega}) + \varepsilon(k) + n(k), \quad (5.2)$$

where

1.  $A_{\Omega} := |H(e^{j\Omega})|$  and  $\phi_{\Omega} := \angle H(e^{j\Omega})$  are the magnitude and phase of the transfer function  $H(z)$  at  $z = e^{j\Omega}$ ;
2.  $\varepsilon(k)$  is a transient signal that converges to zero as fast as  $c\gamma^k$ , where  $\gamma$  is the magnitude of the pole of  $H(z)$  with largest magnitude and  $c$  some constant; and
3.  $n(k)$  corresponds to measurement noise.

When  $n(k)$  is much smaller than  $\alpha A_{\Omega}$  and for  $k$  sufficiently large so that  $\varepsilon(k)$  is negligible, we have

$$y(k) \approx \alpha A_{\Omega} \cos(\Omega k + \phi_{\Omega}).$$

This allows us to recover both  $A_{\Omega} := |H(e^{j\Omega})|$  and  $\phi_{\Omega} := \angle H(e^{j\Omega})$  from the magnitude and phase of  $y(k)$ . Repeating this experiment for several inputs (5.2) with distinct frequencies  $\Omega$ , one obtains several points in the Bode plot of  $H(z)$  and, eventually, one estimates the frequency response  $H(e^{j\Omega})$  over the range of frequencies of interest.

**Note sb:d-t-frequency.** Discrete-time angular frequencies  $\Omega$  take values from 0 to  $\pi$ . When operating with a sample period  $T_s$ , to generate a discrete-time signal  $u_d(k)$  that corresponds to the sampled version of the continuous-time signal

$$u_c(t) = A \cos(\omega t), \quad t \geq 0,$$

we should select  $\Omega = \omega T_s = 2\pi f T_s \dots$   $\blacktriangleright$  p. 13

**Note 29.** To minimize the errors caused by noise, the amplitude  $\alpha$  should be large. However, it should still be sufficiently small so that the process does not leave the linear regime.

### 5.3.2 Correlation method

Especially when there is noise, it may be difficult to determine the amplitude and phase of  $y(k)$  by inspection. The *correlation method* aims at accomplishing this task.

Suppose that the input  $u(k)$  in (5.1) was applied, resulting in the measured output  $y(k)$  given by (5.2). In the correlation method we compute

$$K_{\Omega} := \frac{1}{T} \sum_{k=1}^T e^{-j\Omega k} y(k) = \frac{1}{T} \sum_{k=1}^T \left( \cos(\Omega k) - j \sin(\Omega k) \right) y(k).$$

Using the expression for  $y(k)$  in (5.2), we conclude after fairly straightforward algebraic manipulations that

$$K_{\Omega} = \frac{\alpha}{2} H(e^{j\Omega}) + \frac{\alpha}{2T} H^*(e^{j\Omega}) \frac{e^{-2j\Omega} - e^{-2j\Omega(T+1)}}{1 - e^{-2j\Omega}} + \frac{1}{T} \sum_{k=1}^T e^{-j\Omega k} (\varepsilon(k) + n(k)).$$

As  $T \rightarrow \infty$ , the second term converges to zero and the summation converges to the  $z$ -transform of  $\varepsilon(k) + n(k)$  at the point  $z = e^{-j\Omega}$ . We thus conclude that

$$K_{\Omega} \rightarrow \frac{\alpha}{2} H(e^{j\Omega}) + \lim_{T \rightarrow \infty} \frac{1}{T} (E(e^{-j\Omega}) + N(e^{-j\Omega})),$$

where  $E(z)$  and  $N(z)$  denote the  $z$ -transforms of  $\varepsilon(k)$  and  $n(k)$ , respectively. As long as  $n(k)$  and  $\varepsilon(k)$  do not contain pure sinusoidal terms of frequency  $\Omega$ , the  $z$ -transforms are finite and therefore the limit is equal to zero. This leads to the following estimate for  $H(e^{j\Omega})$

$$\widehat{H(e^{j\Omega})} = \frac{2}{\alpha} K_{\Omega},$$

which is valid for large  $T$ .

**Attention!** The main *weaknesses* of the frequency response methods are:

- (i) To estimate  $H(e^{j\Omega})$  at a single frequency, one still needs a long input (i.e.,  $T$  large). In practice, this leads to a long experimentation period to obtain the Bode plot over a wide range of frequencies.
- (ii) It requires that we apply very specific inputs (sinusoids) to the process. This may not be possible (or safe) in certain applications. Especially for processes that are open-loop unstable.
- (iii) We do not get a parametric form of the transfer function; and, instead, we get the Bode plot directly. This prevents the use of control design methods that are not directly based on the process' Bode plot.

Its key *strengths* are

- (i) It is typically very robust with respect to measurement noise since it uses a large amount of data to estimate a single point in the Bode plot.
- (ii) It requires very few assumptions on the transfer function, which may not even be rational.  $\square$

## 5.4 MATLAB<sup>®</sup> hints

**MATLAB<sup>®</sup> Hint 21** (impulse). The command `impulse` from the identification toolbox performs impulse response estimation from data collected with arbitrary inputs. To use this command one must

1. Create a data object that encapsulates the input/output data using:

**Note 30.** Why?...  $\blacktriangleright$  p. 49

**Note 31.** Since the process is bounded,  $\varepsilon(k)$  converges to zero and therefore it has no pure sinusoids. However, a periodic noise term with frequency exactly equal to  $\Omega$  would lead to trouble.

**Note 32.** Even though the term due to  $\varepsilon(k)$  does not lead to trouble as  $T \rightarrow \infty$ , it is still a good idea to start collecting data only after the time at which  $y(k)$  appears to have stabilized into a pure sinusoid.

**Note 33.**  $K_{\Omega}$  is a complex number so we get estimates for the amplitude and phase of  $H(e^{j\Omega})$ .

data=iiddata(y,u),

where u and y are vectors with the input and output data.

2. Plot the estimated impulse response using:

impulse(dat),

or compute the response using

model=impulse(data),

where m is a MATLAB<sup>®</sup> object with an impulse-response model. The impulse response is given by model.b.  $\square$

## 5.5 To probe further

**Note 28** (Noise in step-response method). When  $n(k)$  and  $n(k-1)$  are independent, we have

$$\text{Var}[n(k) - n(k-1)] = \text{Var}[n(k)] + \text{Var}[n(k-1)].$$

Therefore

$$\begin{aligned} \text{StdDev} \left[ \frac{n(k) - n(k-1)}{\alpha} \right] &= \sqrt{\text{Var} \left[ \frac{n(k) - n(k-1)}{\alpha} \right]} \\ &= \sqrt{\frac{\text{Var}[n(k)] + \text{Var}[n(k-1)]}{\alpha^2}} \end{aligned}$$

When, both variables have the same standard deviation  $\sigma$ , we obtain

$$\text{StdDev} \left[ \frac{n(k) - n(k-1)}{\alpha} \right] = \sqrt{\frac{2\sigma^2}{\alpha^2}} = \frac{\sqrt{2}\sigma}{\alpha}. \quad \square$$

**Note 30** (Correlation method). In the correlation method one computes

$$K_{\Omega} := \frac{1}{T} \sum_{k=1}^T e^{-j\Omega k} y(k) = \frac{1}{T} \sum_{k=1}^T (\cos(\Omega k) - j \sin(\Omega k)) y(k)$$

Using the expression for  $y(k)$  in (5.2), we conclude that

$$K_{\Omega} = \frac{\alpha}{T} A_{\Omega} \sum_{k=1}^T \left( \cos(\Omega k) \cos(\Omega k + \phi_{\Omega}) - j \sin(\Omega k) \cos(\Omega k + \phi_{\Omega}) \right) + \frac{1}{T} \sum_{k=1}^T e^{-j\Omega k} (\varepsilon(k) + n(k)).$$

Applying the trigonometric formulas  $\cos a \cos b = \frac{1}{2}(\cos(a-b) + \cos(a+b))$  and  $\sin a \cos b = \frac{1}{2}(\sin(a-b) + \sin(a+b))$ , we further conclude that

$$\begin{aligned} K_{\Omega} &= \frac{\alpha A_{\Omega}}{2T} \sum_{k=1}^T \left( \cos \phi_{\Omega} + \cos(2\Omega k + \phi_{\Omega}) + j \sin \phi_{\Omega} - j \sin(2\Omega k + \phi_{\Omega}) \right) + \frac{1}{T} \sum_{k=1}^T e^{j\Omega k} (\varepsilon(k) + n(k)) \\ &= \frac{\alpha A_{\Omega}}{2T} \sum_{k=1}^T e^{j\phi_{\Omega}} + \frac{\alpha A_{\Omega}}{2T} \sum_{k=1}^T e^{-2j\Omega k - j\phi_{\Omega}} + \frac{1}{T} \sum_{k=1}^T e^{-j\Omega k} (\varepsilon(k) + n(k)) \\ &= \frac{\alpha}{2} H(e^{j\Omega}) + \frac{\alpha}{2T} H^*(e^{j\Omega}) \sum_{k=1}^T e^{-2j\Omega k} + \frac{1}{T} \sum_{k=1}^T e^{-j\Omega k} (\varepsilon(k) + n(k)), \end{aligned}$$

where we used the fact that  $A_{\Omega} e^{j\phi_{\Omega}}$  is equal to the value  $H(e^{j\Omega})$  of the transfer function at  $z = e^{j\Omega}$  and  $A_{\Omega} e^{-j\phi_{\Omega}}$  is equal to its complex conjugate  $H^*(e^{j\Omega})$ . By noticing that the second term is the summation of a geometric series, we can simplify this expression to

$$K_{\Omega} = \frac{\alpha}{2} H(e^{j\Omega}) + \frac{\alpha}{2T} H^*(e^{j\Omega}) \frac{e^{-2j\Omega} - e^{-2j\Omega(T+1)}}{1 - e^{-2j\Omega}} + \frac{1}{T} \sum_{k=1}^T e^{-j\Omega k} (\varepsilon(k) + n(k)) \quad \square$$

**Note 34.** Recall that the sum of a geometric series is given by

$$\sum_{k=1}^T r^k = \frac{r - r^{T+1}}{1 - r}$$

## 5.6 Exercises

**5.1** (Step response). A Simulink block that models a pendulum with viscous friction is provided.

1. Use the Simulink block (without noise) to estimate the system's impulse response  $\hat{h}(k)$  using both the impulse and step response methods.

What is the largest value  $\alpha$  of the input magnitude for which the system still remains within the approximately linear region of operation?

*Hint:* to make sure that the estimate is good you can compare the impulses response from steps and impulses.

2. Turn on the noise in the Simulink block and repeat the identification procedure above for a range of values for  $\alpha$ . For both methods, plot the error  $\sum_k \|\hat{h}_\alpha(k) - h(k)\|$  vs.  $\alpha$ , where  $\hat{h}_\alpha(k)$  denotes the estimate obtained for an input with magnitude  $\alpha$ . What is your conclusion?

Take the estimate  $\hat{h}(k)$  determined in 1 for the noise-free case as the ground truth  $h(k)$ .  $\square$

**5.2** (Correlation method). Modify the Simulink block provided for Exercise 5.1 to accept a sinusoidal input.

1. Estimate the system's frequency response  $\widehat{H}(e^{j\Omega})$  at a representative set of (log-spaced) frequencies  $\Omega_i$  using the correlation method.

Select an input amplitude  $\alpha$  for which the system remains within the approximately linear region of operation.

2. Turn on the noise in the Simulink block and repeat the identification procedure above for a range of values for  $\alpha$ . Plot the error  $\sum_i \|H_\alpha(e^{j\Omega_i}) - H(e^{j\Omega_i})\|$  vs.  $\alpha$ , where  $H_\alpha(e^{j\Omega_i})$  denotes the estimate obtained for an input with magnitude  $\alpha$ . What is your conclusion?

Take the estimate  $\widehat{H}(e^{j\Omega})$  determined in 1 for the noise-free case as the ground truth  $H(e^{j\Omega})$ .

3. Compare your best frequency response estimate  $\widehat{H}(e^{j\Omega})$ , with the frequency responses that you would obtain by taking the Fourier transform of the best impulse response  $\hat{h}(k)$  that you obtained in Exercise 5.1.

*Hint:* Use the MATLAB<sup>®</sup> command `fft` to compute the Fourier transform. This command gives you  $H(e^{j\Omega})$  from  $\Omega = 0$  to  $\Omega = 2\pi$  so you should discard the second half of the Fourier transform (corresponding to  $\Omega > \pi$ ).  $\square$

**Part II**

**Robust Control**



# Introduction to Robust Control

For controller design purposes it is convenient to imagine that we *know* an accurate model for the process, e.g., its transfer function. In practice, this is hardly ever the case:

1. When process models are derived from *first principles*, they always exhibit constants that can only be determined up to some error.  
E.g., the precise values of masses, moments of inertia, and friction coefficients in models derived from Newton's laws; or resistances, capacitances, and gains, in electrical circuits.
2. When one *identifies a model experimentally*, noise and disturbances generally lead to different results as the identification experiment is repeated multiple times. Which experiment gave the true model? The short answer is none, all models obtained have some error.
3. Processes change due to wear and tear so even if a process was perfectly identified before starting operation, its model will soon exhibit some mismatch with respect to the real process.

The goal of this chapter is to learn how to take process model uncertainty into account, while designing a feedback controller.

## Pre-requisites

1. Laplace transform, continuous-time transfer functions, frequency responses, and stability.
2. Classical continuous-time feedback control design using loop-shaping.
3. Knowledge of MATLAB/Simulink.



# Chapter 6

## Robust stability

### Contents

6.1	Model uncertainty	55
6.2	Nyquist stability criterion	58
6.3	Small gain condition	59
6.4	MATLAB hints	61
6.5	Exercises	62

### 6.1 Model uncertainty

Suppose we want to control the spring-mass-damper system in Figure 6.1, which has the following

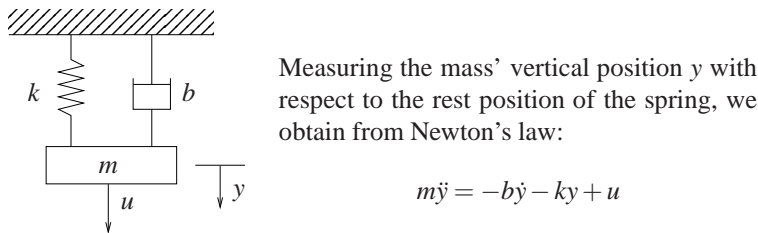


Figure 6.1. Spring-mass-damper system.

transfer function from the applied force  $u$  to the spring position  $y$

$$P(s) = \frac{1}{ms^2 + bs + k}. \tag{6.1}$$

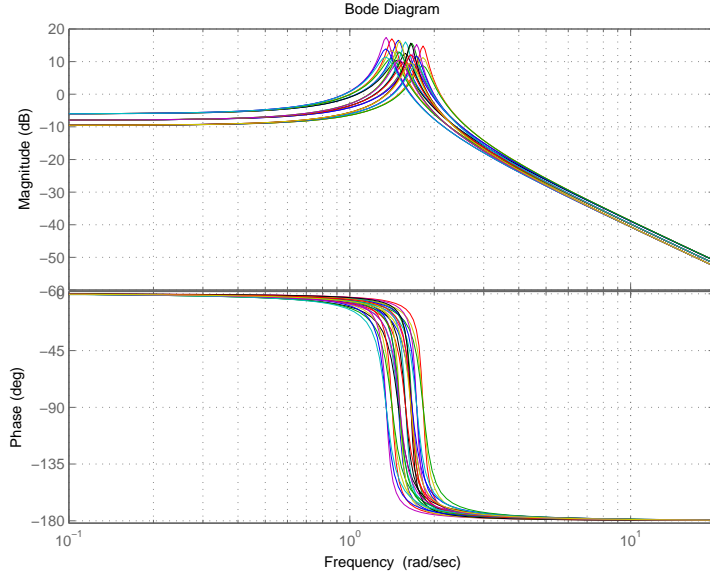
Typically, the mass  $m$ , the friction coefficient  $b$ , and the spring constant  $k$  would be identified experimentally (or taken from some specifications sheet) leading to *confidence intervals* for these parameters and not just a single value:

$$m \in [m_0 - \delta_1, m_0 + \delta_1], \quad b \in [b_0 - \delta_2, b_0 + \delta_2], \quad k \in [k_0 - \delta_3, k_0 + \delta_3].$$

The values with subscript  $0$  are called the *nominal values* for the parameters and the  $\delta_i$  are called the *maximum deviations* from the nominal value.

In practice, this means that there are *many admissible transfer functions for the process*—one for each possible combination of  $m$ ,  $b$ , and  $k$  in the given intervals. Figure 6.2 shows the Bode plot of (6.1) for different values of the parameters  $m$ ,  $b$ , and  $k$ . The idea behind *robust control* is to design a single controllers that achieves acceptable performance (or at least stability) for every admissible process transfer function.

**MATLAB® Hint 22.**  
bode(sys) draws the Bode plot of the system sys... ► p. 61



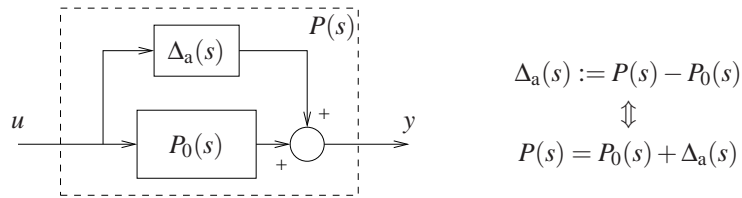
**Figure 6.2.** Bode plot of  $P(s) = \frac{1}{ms^2 + bs + k}$ , for different values of  $m \in [.9, 1.1]$ ,  $b \in [.1, .2]$ , and  $k \in [2, 3]$ . Note how the different values of the parameters lead to different values for the resonance frequency. One can see in the phase plot that it may be dangerous to have the cross-over frequency near the “uncertain” resonance frequency since the phase margin may vary a lot depending on the system parameters. In particular, one process transfer function may lead to a large phase margin, whereas another one to an unstable closed-loop system.

### 6.1.1 Additive uncertainty

In robust control one starts by characterizing the uncertainty in terms of the process’ frequency response. To this effect one selects a *nominal process transfer function*  $P_0(s)$  and, for each admissible process transfer function  $P(s)$ , one defines:

$$\Delta_a(s) := P(s) - P_0(s), \quad (6.2)$$

which measures how much  $P(s)$  deviates from  $P_0(s)$ . This allows us to express any admissible transfer function  $P(s)$  as in Figure 6.3. Motivated by the diagram in Figure 6.3,  $\Delta_a(s)$  is called an *additive uncertainty block*.  $P_0(s)$  should correspond to the “most likely” transfer function, so that



**Figure 6.3.** Additive uncertainty

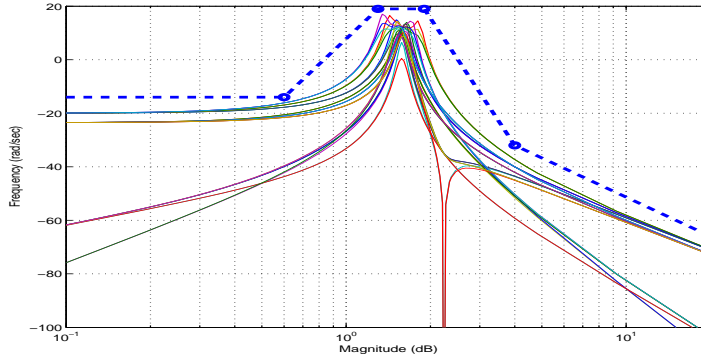
the additive uncertainty block is as small as possible. In the example above, one would typically choose the transfer function corresponding to the nominal parameter values:

$$P_0(s) = \frac{1}{m_0s^2 + b_0s + k_0}.$$

To obtain a characterization of the uncertainty *purely in the frequency domain*, one needs to specify how large  $\Delta_a(j\omega)$  may be for each frequency  $\omega$ . This is done by determining a function  $\ell_a(\omega)$  sufficiently large so that for every admissible process transfer function  $P(s)$  we have

$$|\Delta_a(j\omega)| = |P(j\omega) - P_0(j\omega)| \leq \ell_a(\omega), \quad \forall \omega.$$

When one has available all the admissible  $P(s)$  (or a representative set of them—such as in Figure 6.2), one can determine  $\ell_a(\omega)$  by simply plotting  $|P(j\omega) - P_0(j\omega)|$  vs.  $\omega$  for all the  $P(s)$  and choosing for  $\ell_a(\omega)$  a function larger than all the plots. Since in general it is not feasible to plot *all*  $|P(j\omega) - P_0(j\omega)|$ , one should provide some “safety-cushion” when selecting  $\ell_a(\omega)$ . Figure 6.4 shows  $\ell_a(\omega)$  for the Bode plots in Figure 6.2.



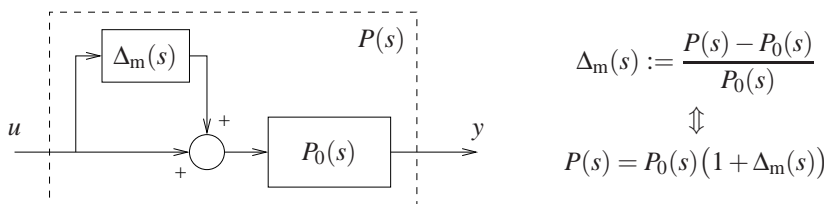
**Figure 6.4.** Additive uncertainty bounds for the process Bode plots in Figure 6.2 with  $P_0(s) := \frac{1}{m_0s^2 + b_0s + k_0}$ ,  $m_0 = 1$ ,  $b_0 = 1.5$ ,  $k_0 = 2.5$ . The solid lines represent possible plots for  $|P(j\omega) - P_0(j\omega)|$  and the dashed one represents the uncertainty bound  $\ell_a(\omega)$ . Note the large uncertainty bound near the resonance frequency. We shall show shortly how to design a controller that stabilizes *all* processes for which the additive uncertainty falls below the dashed line.

### 6.1.2 Multiplicative uncertainty

The additive uncertainty in (6.2) measures the difference between  $P(s)$  and  $P_0(s)$  in absolute terms and may seem misleadingly large when both  $P(s)$  and  $P_0(s)$  are large—e.g., at low frequencies when the process has a pole at the origin. To overcome this difficulty one often defines instead

$$\Delta_m(s) := \frac{P(s) - P_0(s)}{P_0(s)}, \tag{6.3}$$

which measures how much  $P(s)$  deviates from  $P_0(s)$ , relative to the size of  $P_0(s)$ . We can now express any admissible transfer function  $P(s)$  as in Figure 6.5, which motivates calling  $\Delta_m(s)$  a *multiplicative uncertainty block*. To characterize a multiplicative uncertainty one determines a function

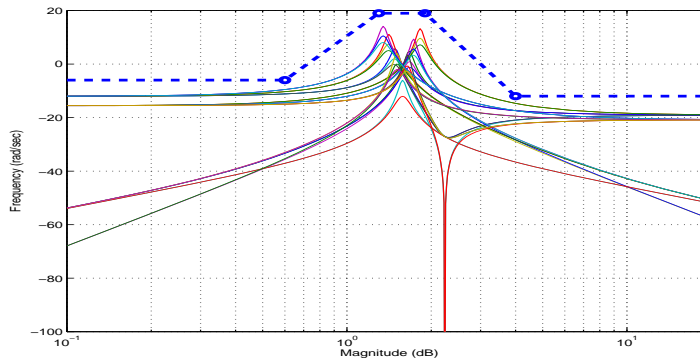


**Figure 6.5.** Multiplicative uncertainty

$\ell_m(\omega)$  sufficiently large so that for every admissible process transfer function  $P(s)$  we have

$$|\Delta_m(j\omega)| = \frac{|P(s) - P_0(s)|}{|P_0(s)|} \leq \ell_m(\omega), \quad \forall \omega.$$

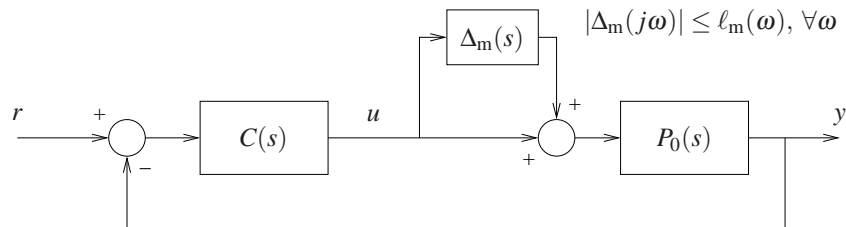
One can determine  $\ell_m(\omega)$  by plotting  $\frac{|P(s) - P_0(s)|}{|P_0(s)|}$  vs.  $\omega$  for all admissible  $P(s)$  (or a representative set of them) and choosing  $\ell_m(\omega)$  to be larger than all the plots. Figure 6.6 shows  $\ell_m(\omega)$  for the Bode plots in Figure 6.2.



**Figure 6.6.** Multiplicative uncertainty bounds for the process Bode plots in Figure 6.2 with  $P_0(s) := \frac{1}{m_0 s^2 + b_0 s + k_0}$ ,  $m_0 = 1$ ,  $b_0 = 1.5$ ,  $k_0 = 2.5$ . The solid lines represent possible plots for  $\frac{|P(j\omega) - P_0(j\omega)|}{|P_0(j\omega)|}$  and the dashed one represents the uncertainty bound  $\ell_m(\omega)$ . Note the large uncertainty bound near the resonance frequency. We shall show shortly how to design a controller that stabilizes *all* processes for which the multiplicative uncertainty falls below the dashed line.

## 6.2 Nyquist stability criterion

The first question we address is: Given a specific feedback controller  $C(s)$ , how can we verify that it stabilizes every admissible process  $P(s)$ . When the admissible processes are described in terms of a multiplicative uncertainty block, this amounts to verifying that the closed-loop system in Figure 6.7 is stable *for every*  $\Delta_m(j\omega)$  with norm smaller than  $\ell_m(\omega)$ . This can be done using the Nyquist

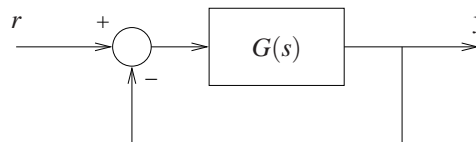


**Figure 6.7.** Unity feedback configuration with multiplicative uncertainty

stability criterion, which we review next.

The Nyquist criterion is used to investigate the stability of the negative feedback connection in Figure 6.8. We briefly summarize it here. The textbook [4, Section 6.3] provides a more detailed

**Note 35.** Figure 6.8 can represent the closed-loop system in Figure 6.7 if we choose  $G(s) := (1 + \Delta_m(s))P(s)C(s)$ .



**Figure 6.8.** Negative feedback

description of it with several examples.

### 6.2.1 The Nyquist Plot

The first step consists of drawing the *Nyquist plot*, which is done by evaluating  $G(j\omega)$  from  $\omega = -\infty$  to  $\omega = +\infty$  and plotting it in the complex plane. This leads to a closed-curve that is always symmetric with respect to the real axis. This curve should be annotated with arrows indicating the direction corresponding to increasing  $\omega$ .

**MATLAB® Hint 23.**  
nyquist(sys) draws the Nyquist plot of the system sys...  
► p. 62

### 6.2.2 The Nyquist criterion

Count the number #ENC of clockwise encirclements by the Nyquist plot of the point  $-1$ . To do this, we draw a ray from  $-1$  to  $\infty$  in *any* direction and add one each time the Nyquist plot crosses the ray in the clockwise direction (with respect to the origin of the ray) and subtract one each time it crosses the ray in the counter-clockwise direction. The final count gives #ENC.

**Nyquist Stability Criterion.** *The total number of unstable (i.e., in the right-hand-side plane) closed-loop poles (#CUP) is given by*

$$\#CUP = \#ENC + \#OUP,$$

where #OUP denotes the number of unstable (open-loop) poles of  $G(s)$ . To have a stable closed-loop one thus needs

$$\#ENC = -\#OUP. \quad \square$$

**Attention!** Any poles of  $G(s)$  on the imaginary axis should be moved slightly to the left of the axis to avoid divisions by zero. E.g.,

$$\begin{aligned} G(s) &= \frac{s+1}{s(s-3)} & \longrightarrow & G_\varepsilon(s) \approx \frac{s+1}{(s+\varepsilon)(s-3)} \\ G(s) &= \frac{s}{s^2+4} = \frac{s}{(s+2j)(s-2j)} & \longrightarrow & G_\varepsilon(s) \approx \frac{s}{(s+\varepsilon+2j)(s+\varepsilon-2j)} = \frac{s}{(s+\varepsilon)^2+4}, \end{aligned}$$

for a small  $\varepsilon > 0$ . The criterion should then be applied to the “perturbed” transfer function  $G_\varepsilon(s)$ . If we conclude that the closed-loop is stable for  $G_\varepsilon(s)$  with very small  $\varepsilon$ , then the closed-loop with  $G(s)$  will also be stable and vice-versa.  $\square$

Figure 6.9 shows the Nyquist plot of

$$G_0(s) = C(s)P_0(s), \quad (6.4)$$

for the nominal process model

$$P_0(s) := \frac{1}{m_0s^2 + b_0s + k_0}, \quad m_0 = 1, b_0 = 1.5, k_0 = 2.5, \quad (6.5)$$

(used in Figures 6.4 and 6.6) and a PID controller

$$C(s) := \frac{10}{s} + 15 + 5s, \quad (6.6)$$

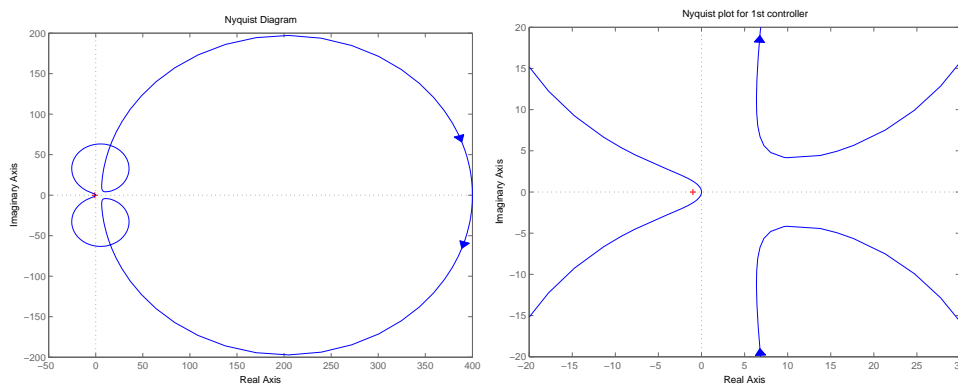
To obtain this plot, we moved the single controller pole on the imaginary axis to the left-hand side of the complex plane. Since this led to an open loop gain with no unstable poles ( $\#OUP = 0$ ) and there are no encirclements of  $-1$  ( $\#ENC = 0$ ), we conclude that the closed-loop system is stable. This means that the given PID controller  $C(s)$  stabilizes, at least, the nominal process  $P_0(s)$ . It remains to check if it also stabilizes every admissible process model with multiplicative uncertainty.

### 6.3 Small gain condition

Consider the closed-loop system in Figure 6.7 and suppose that we are given a controller  $C(s)$  that stabilizes the nominal process  $P_0(s)$ , i.e., the closed-loop is stable when  $\Delta_m(s) = 0$ . Our goal is to find out if the closed-loop remains stable for every  $\Delta_m(j\omega)$  with norm smaller than  $\ell_m(\omega)$ .

Since  $C(s)$  stabilizes  $P_0(s)$ , we know that the Nyquist plot of the nominal (open-loop) transfer function

$$G_0(s) = C(s)P_0(s),$$



**Figure 6.9.** Nyquist plot for the (open-loop) transfer function in (6.4). The right figure shows a zoomed view of the origin. To avoid a pole over the imaginary axis, in these plots we moved the pole of the controller from 0 to  $-.01$ .

has the “right” number of encirclements ( $\#ENC = -\#OUP$ , where  $\#OUP$  is the number of unstable poles of  $G_0(s)$ ). To check is the closed-loop is stable from some admissible process transfer function

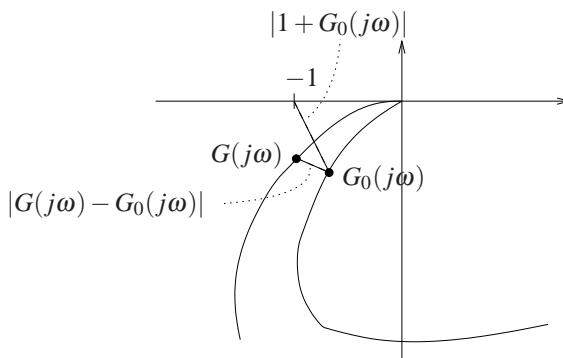
$$P(s) = P_0(s)(1 + \Delta_m(s)),$$

we need to draw the Nyquist plot of

$$G(s) = C(s)P(s) = C(s)P_0(s)(1 + \Delta_m(s)) = G_0(s) + G_0(s)\Delta_m(s)$$

and verify that we still get the same number of encirclements.

**Note 36.** We are assuming that  $G(s)$  and  $G_0(s)$  have the same number of unstable poles  $\#OUP$  and therefore stability is achieved for the same number of encirclements  $\#ENC = -\#OUP$ . In practice this means that the uncertainty should not change the stability of any pole.



**Figure 6.10.** Nyquist plot derivation of the small-gain conditions

The Nyquist plots of  $G(s)$  and  $G_0(s)$  differ by

$$|G(j\omega) - G_0(j\omega)| = |G_0(j\omega)\Delta_m(j\omega)| \leq |G_0(j\omega)| \ell_m(\omega),$$

A simple way to make sure that  $G(j\omega)$  and  $G_0(j\omega)$  have the same number of encirclements is to ask that the maximum difference between the two be smaller than the distance from  $G_0(j\omega)$  to the point  $-1$ , i.e.,

$$|1 + G_0(j\omega)| > |G_0(j\omega)| \ell_m(\omega) \Leftrightarrow \frac{|G_0(j\omega)|}{|1 + G_0(j\omega)|} < \frac{1}{\ell_m(\omega)} \quad \forall \omega.$$

This leads to the so called *small-gain condition*:

**Small-gain Condition.** The closed-loop system in Figure 6.7 is stable for every  $\Delta_m(j\omega)$  with norm smaller than  $\ell_m(\omega)$ , provided that

$$\left| \frac{C(j\omega)P_0(j\omega)}{1+C(j\omega)P_0(j\omega)} \right| < \frac{1}{\ell_m(\omega)}, \quad \forall \omega. \tag{6.7}$$

The transfer function on the left-hand-side of (6.7) is precisely the *complementary sensitivity function*:

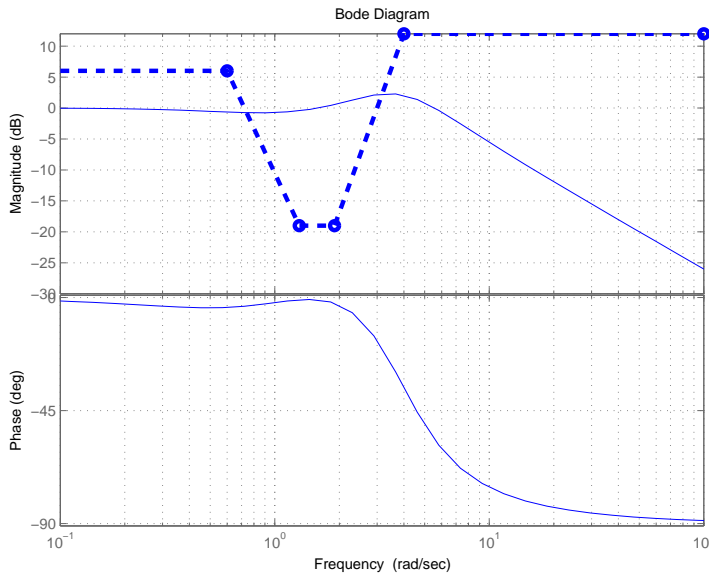
$$T_0(s) := 1 - S_0(s), \quad S_0(s) := \frac{1}{1+C(s)P_0(s)}$$

for the nominal process. So (6.7) can be interpreted as requiring the norm of the nominal complementary sensitivity function to be smaller than  $1/\ell_m(\omega)$ ,  $\forall \omega$ . For this reason (6.7) is called a *small-gain condition*.

Figure 6.11 shows the Bode plot of the complementary sensitivity function for the nominal process in (6.5) and the PID controller in (6.6). In the same plot we can see the  $20 \log_{10} \frac{1}{\ell_m(\omega)} = -20 \log_{10} \ell_m(\omega)$ , for the  $\ell_m(\omega)$  in Figure 6.6. Since the magnitude plot of  $T_0(j\omega)$  is not always below that of  $\frac{1}{\ell_m(\omega)}$ , we conclude that the system may be unstable for some admissible processes. However, if we redesign our controller to consist of an integrator with two lags

$$C(s) = \frac{.005(s+5)^2}{s(s+.5)^2}, \tag{6.8}$$

the magnitude plot of  $T_0(j\omega)$  is now always below that of  $\frac{1}{\ell_m(\omega)}$  and we conclude that we have stability for every admissible process. In this case, the price to pay was a low bandwidth. This example illustrates a common problem in the control of systems with uncertainty: *it is not possible to get good reference tracking over ranges of frequencies for which there is large uncertainty*.



**Figure 6.11.** Verification of the small gain condition for the nominal process (6.5), the multiplicative uncertainty in Figure 6.6, and the PID controller (6.6). The solid line corresponds to the Bode plot of the complementary sensitivity function  $T_0(s)$  and the dashed line to  $\frac{1}{\ell_m(\omega)}$  (both in dB).

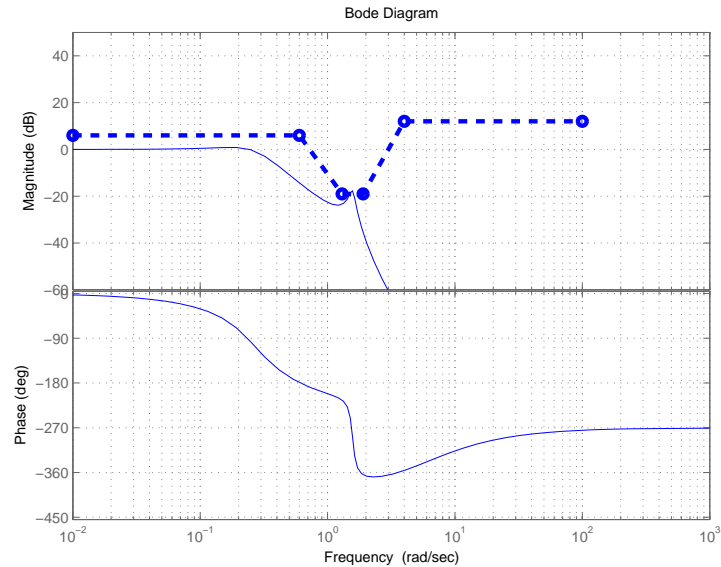
## 6.4 MATLAB hints

**MATLAB® Hint 22** (bode). The command `bode(sys)` draws the Bode plot of the system `sys`. To specify the system one can use:

**Note 37.** A mnemonic to remember (6.7) is that the transfer function whose norm needs to be small is precisely the transfer function “seen” by the  $\Delta_m$  block in Figure 6.7. This mnemonic also “works” for additive uncertainty... ► p. 63

**MATLAB® Hint 24.** To check if (6.7) holds for a specific system, draw  $20 \log_{10} \frac{1}{\ell_m(\omega)} = -20 \log_{10} \ell_m(\omega)$  on top of the magnitude Bode plot of the complementary sensitivity function and see if the latter always lies below the former.

**Note 38.** Recall that the complementary sensitivity function is also the closed-loop transfer function for the reference  $r$  to the output  $y$ . Therefore good tracking requires this function to be close to 1, whereas to reject large uncertainty we need this function to be much smaller than 1.



**Figure 6.12.** Verification of the small gain condition for the nominal process (6.5), the multiplicative uncertainty in Figure 6.6, and the integrator with 2 lags controller (6.8). The solid line corresponds to the Bode plot of the complementary sensitivity function  $T_0(s)$  and the dashed line to  $\frac{1}{\ell_m(\omega)}$  (both in dBs).

1. `sys=tf(num,den)`, where `num` is a vector with the coefficients of the numerator of the system's transfer function and `den` a vector with the coefficients of the denominator. The last coefficient must always be the zero-order one. E.g., to get  $\frac{2s}{s^2+3}$  one should use `num=[2 0]`; `den=[1 0 3]`;
2. `sys=zpk(z,p,k)`, where `z` is a vector with the zeros of the system, `p` a vector with its poles, and `k` the gain. E.g., to get  $\frac{2s}{(s+1)(s+3)}$  one should use `z=0`; `p=[1, 3]`; `k=2`;
3. `sys=ss(A,B,C,D)`, where `A,B,C,D` are a realization of the system. □

**MATLAB<sup>®</sup> Hint 23** (`nyquist`). The command `nyquist(sys)` draws the Nyquist plot of the system `sys`. To specify the system you can use any of the commands in Matlab hint 22.

Especially when there are poles very close to the imaginary axis (e.g., because they were actually on the axis and you moved them slightly to the left), the automatic scale may not be very good because it may be hard to distinguish the point  $-1$  from the origin. In this case, you can use then zoom features of MATLAB to see what is going on near  $-1$ : Try clicking on the magnifying glass and selecting a region of interest; or try left-clicking on the mouse and selecting “zoom on  $(-1,0)$ ” (without the magnifying glass selected.) □

## 6.5 Exercises

**6.1** (Unknown parameters). Suppose one want to control the orientation of a satellite with respect to its orbital plane by applying a thruster's generated torque. The system's transfer function is give by

$$P(s) := \frac{10(bs+k)}{s^2(s^2+11(bs+k))},$$

where the values of the parameters  $b$  and  $k$  are not exactly known, but it is known that

$$.09 \leq k \leq .4 \quad .006 \leq b \leq .03.$$

Find a nominal model and compute the corresponding additive and multiplicative uncertainty bounds. □

**6.2** (Noisy identification). The Simulink block provided corresponds to a discrete-time system with transfer function

$$H(z) = \frac{\alpha_1 z + \alpha_0}{z^2 + \beta_1 z + \beta_0}.$$

1. Use least-squares to estimate the 4 coefficients of the transfer function. Repeat the identification experiment 25 times to obtain 25 estimates of the system's transfer functions.
2. Convert the discrete-time transfer functions so obtained to continuous-time using the Tustin transformation.

*Hint:* Use the MATLAB command `d2c`.

3. Select a nominal model and compute the corresponding additive and multiplicative uncertainty bounds. □

**6.3** (Small gain). For the nominal process model and multiplicative uncertainty that you obtained in Exercise 6.2, use the small gain condition to verify if the following controllers achieve stability for all admissible process models:

$$C_1(s) = .3 \quad C_2(s) = \frac{2}{s + .1} \quad C_3(s) = \frac{20}{s - 30} \quad C_4(s) = \frac{5(s + 1)}{(s + 2)(s + 3)} \quad C_5(s) = \frac{2(s + 1)}{(s + .3)^2}$$

Justify your answers with appropriate Bode plots. □

**6.4** (Robustness vs. performance). Justify the statement: "It is not possible to get good reference tracking over ranges of frequencies for which there is large uncertainty." □

**6.5** (Additive uncertainty). Derive a small-gain condition similar to (6.7) for additive uncertainty. Check if the mnemonic in Sidebar 37 still applies.

*Hint:* With additive uncertainty, the open-loop gain is given by

$$G(s) = C(s)P(s) = C(s)(P_0(s) + \Delta_a(s)) = G_0(s) + C(s)\Delta_a(s),$$

which differs from  $G_0(s)$  by  $C(s)\Delta_a(s)$ . □



# Chapter 7

## Control design by loop-shaping

### Contents

7.1 The loop-shaping design method . . . . .	65
7.2 Open-loop vs. closed-loop specifications . . . . .	65
7.3 Open-loop gain shaping . . . . .	69
7.4 Exercises . . . . .	69

### 7.1 The loop-shaping design method

The goal of this chapter is to briefly review the loop-shaping control design method for SISO systems. The basic idea behind loop shaping is to convert the desired specifications on the closed-loop

**Note.** The loop-shaping design method is covered extensively, e.g., in [4].

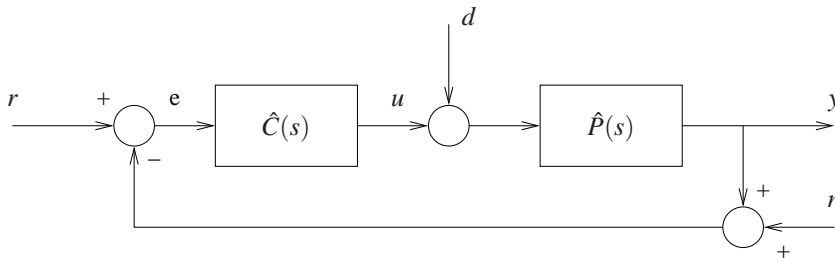


Figure 7.1. Closed-loop system

system in Figure 7.1 into constraints on the open-loop gain

$$G_0(s) := C(s)P_0(s).$$

The controller  $C(s)$  is then designed so that the open-loop gain  $G_0(s)$  satisfies these constraints. The shaping of  $G_0(s)$  can be done using the classical methods briefly mentioned in Section 7.3 and explained in much greater detail in [4, Chapter 6.7]. However, it can also be done using LQR state feedback, as discussed in Section 9.5, or using LQG/LQR output feedback controllers, as we shall see in Section 10.6.

### 7.2 Open-loop vs. closed-loop specifications

We start by discussing how several closed-loop specifications can be converted into constraints on the open-loop gain  $G_0(s)$ .

**Stability.** Assuming that the open-loop gain has no unstable poles, the stability of the closed-loop system is guaranteed as long as the phase of the open-loop gain is above  $-180^\circ$  at the cross-over frequency  $\omega_c$ , i.e., at the frequency for which

$$|G_0(j\omega_c)| = 1.$$

**Overshoot.** Larger phase margins generally correspond to a smaller overshoot for the step response of the closed-loop system. The following rules of thumb work well when the open-loop gain  $G_0(s)$  has a pole at the origin, an additional real pole, and no zeros:

$$G_0(s) = \frac{k}{s(s+p)}, \quad p > 0,$$

The following rules of thumb are useful to determine the value of the phase margin needed to obtain the desired overshoot:

Phase margin	overshoot
$65^\circ$	$\leq 5\%$
$60^\circ$	$\leq 10\%$
$45^\circ$	$\leq 15\%$

**Reference tracking.** Suppose that one wants the tracking error to be at least  $k_T \ll 1$  times smaller than the reference, over the range of frequencies  $[0, \omega_T]$ . In the frequency domain, this can be expressed by

$$\frac{|E(j\omega)|}{|R(j\omega)|} \leq k_T, \quad \forall \omega \in [0, \omega_T], \quad (7.1)$$

where  $E(s)$  and  $R(s)$  denote the Laplace transforms of the tracking error  $e := r - y$  and the reference signal  $r$ , respectively, in the absence of noise and disturbances. For the closed-loop system in Figure 7.1,

$$E(s) = \frac{1}{1 + G_0(s)} R(s).$$

Therefore (7.1) is equivalent to

$$\frac{1}{|1 + G_0(j\omega)|} \leq k_T, \quad \forall \omega \in [0, \omega_T] \quad \Leftrightarrow \quad |1 + G_0(j\omega)| \geq \frac{1}{k_T}, \quad \forall \omega \in [0, \omega_T].$$

This condition is guaranteed to hold by requiring that

$$\boxed{|G_0(j\omega)| \geq \frac{1}{k_T} + 1, \quad \forall \omega \in [0, \omega_T].} \quad (7.2)$$

**Disturbance rejection.** Suppose that one wants input disturbances to appear in the output attenuated at least  $k_D \ll 1$  times, over the range of frequencies  $[\omega_{D_1}, \omega_{D_2}]$ . In the frequency domain, this can be expressed by

$$\frac{|Y(j\omega)|}{|D(j\omega)|} \leq k_D, \quad \forall \omega \in [\omega_{D_1}, \omega_{D_2}], \quad (7.3)$$

where  $Y(s)$  and  $D(s)$  denote the Laplace transforms of the output  $y$  and the input disturbance  $d$ , respectively, in the absence of reference and measurement noise. For the closed-loop system in Figure 7.1,

$$Y(s) = \frac{P_0(s)}{1 + G_0(s)} D(s),$$

**Notation.** The distance between the phase of  $G_0(j\omega_c)$  and  $-180^\circ$  is called the *phase margin*.

**Note 39.** Additional zeros and poles at frequencies significantly above the cross over generally have little effect and can be ignored. However, additional dynamics below or around the cross-over frequency typically affect the overshoot; and make determining the needed phase margin to achieve a desired overshoot a trial and error process.

**Note 40.** Typically one wants to track low frequency references, which justifies the requirement for equation (7.1) to hold in an interval of the form  $[0, \omega_T]$ .

**Note 41.** Typically one wants to reject low-frequency disturbances and therefore  $\omega_{D_1}$  and  $\omega_{D_2}$  in (7.3) generally take low values.

and therefore (7.3) is equivalent to

$$\frac{|P_0(j\omega)|}{|1 + G_0(j\omega)|} \leq k_D, \quad \forall \omega \in [\omega_{D_1}, \omega_{D_2}] \Leftrightarrow |1 + G_0(j\omega)| \geq \frac{|P_0(j\omega)|}{k_D}, \quad \forall \omega \in [\omega_{D_1}, \omega_{D_2}].$$

This condition is guaranteed to hold as long as one requires that

$$\boxed{|G_0(j\omega)| \geq \frac{|P_0(j\omega)|}{k_D} + 1, \quad \forall \omega \in [\omega_{D_1}, \omega_{D_2}].} \quad (7.4)$$

**Noise rejection.** Suppose that one wants measurement noise to appear in the output attenuated at least  $k_N \ll 1$  times, over the range of frequencies  $[\omega_N, \infty)$ . In the frequency domain, this can be expressed by

$$\frac{|Y(j\omega)|}{|N(j\omega)|} \leq k_N, \quad \forall \omega \in [\omega_N, \infty), \quad (7.5)$$

**Note 42.** Typically one needs to reject high frequencies noise, which justifies the requirement for equation (7.5) to hold in an interval of the form  $[\omega_N, \infty)$ .

where  $Y(s)$  and  $N(s)$  denote the Laplace transforms of the output  $y$  and the measurement noise  $n$ , respectively, in the absence of reference and disturbances. For the closed-loop system in Figure 7.1,

$$Y(s) = -\frac{G_0(s)}{1 + G_0(s)}N(s), \quad (7.6)$$

and therefore (7.5) is equivalent to

$$\frac{|G_0(j\omega)|}{|1 + G_0(j\omega)|} \leq k_N, \quad \forall \omega \in [\omega_N, \infty) \Leftrightarrow \left|1 + \frac{1}{G_0(j\omega)}\right| \geq \frac{1}{k_N}, \quad \forall \omega \in [\omega_N, \infty).$$

This condition is guaranteed to hold as long as one requires that

$$\left|\frac{1}{G_0(j\omega)}\right| \geq \frac{1}{k_N} + 1, \quad \forall \omega \in [\omega_N, \infty) \Leftrightarrow \boxed{|G_0(j\omega)| \leq \frac{k_N}{1 + k_N}, \quad \forall \omega \in [\omega_N, \infty).} \quad (7.7)$$

**Robustness with respect to multiplicative uncertainty** Suppose that one wants the closed-loop to remain stable for every multiplicative uncertainty  $\Delta_m(j\omega)$  with norm smaller than  $\ell_m(\omega)$ . This can be expressed by

$$\frac{|G_0(j\omega)|}{|1 + G_0(j\omega)|} \leq \frac{1}{\ell_m(\omega)}, \quad \forall \omega, \quad (7.8)$$

which is equivalent to

$$\left|1 + \frac{1}{G_0(j\omega)}\right| \geq \ell_m(\omega), \quad \forall \omega. \quad (7.9)$$

We have *two options* to make sure that (7.9) hold:

1. As in the discussion for noise rejection, we can require  $|G_0(j\omega)|$  to be small. In particular, reasoning as before we may require that

$$\left|\frac{1}{G_0(j\omega)}\right| \geq \ell_m(\omega) + 1, \quad \Leftrightarrow \boxed{|G_0(j\omega)| \leq \frac{1}{1 + \ell_m(\omega)}} \quad (7.10)$$

This condition will hold for frequencies for which we can make  $|G_0(j\omega)|$  small, typically high frequencies.

**Note 43.** While this condition is similar to the one appearing in (7.6), we now need it to hold *for every frequency* and we no longer have the “luxury” of simply requiring  $|G_0(j\omega)|$  to be small for every frequency.

2. Alternatively, (7.9) may hold even when  $|G_0(j\omega)|$  large, provided that  $\ell_m(\omega)$  is small. In particular, since

$$\left|1 + \frac{1}{G_0(j\omega)}\right| \geq 1 - \left|\frac{1}{G_0(j\omega)}\right|,$$

the condition (7.9) holds, provided that we require that

$$1 - \left|\frac{1}{G_0(j\omega)}\right| \geq \ell_m(\omega) \quad \Leftrightarrow \quad \boxed{|G_0(j\omega)| \geq \frac{1}{1 - \ell_m(\omega)}} \quad (7.11)$$

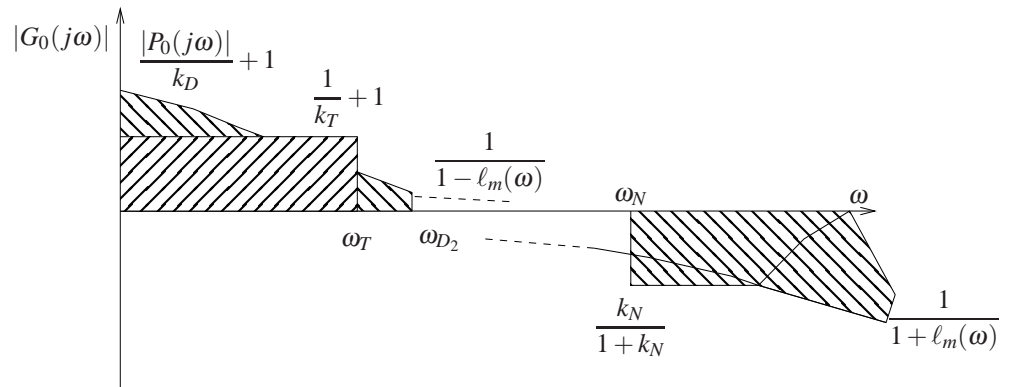
This condition will hold for frequencies for which  $|G_0(j\omega)|$  is large, typically low frequencies.

From the two conditions (7.10)–(7.11), one generally needs to be mostly concerned about (7.10). This is because when  $\ell_m(\omega)$  is small, (7.9) will generally hold. However, when  $\ell_m(\omega)$  is large for (7.9) to hold we need  $G_0(j\omega)$  to be small, which corresponds to the condition (7.10). Hopefully,  $\ell_m(\omega)$  will only be large at high frequencies, for which we do not need (7.2) or (7.4) to hold.

Table 7.1 and Figure 7.2 summarize the constraints on the open-loop gain  $G_0(j\omega)$  discussed above.

closed-loop specification	open-loop constraint
overshoot $\leq 10\%$ ( $\leq 5\%$ )	phase margin $\geq 60$ deg ( $\geq 65$ deg)
$\frac{ E(j\omega) }{ R(j\omega) } \leq k_T, \quad \forall \omega \in [0, \omega_T]$	$ G_0(j\omega)  \geq \frac{1}{k_T} + 1, \quad \forall \omega \in [0, \omega_T]$
$\frac{ Y(j\omega) }{ D(j\omega) } \leq k_D, \quad \forall \omega \in [\omega_{D1}, \omega_{D2}]$	$ G_0(j\omega)  \geq \frac{ P_0(j\omega) }{k_D} + 1, \quad \forall \omega \in [\omega_{D1}, \omega_{D2}]$
$\frac{ Y(j\omega) }{ N(j\omega) } \leq k_N, \quad \forall \omega \in [\omega_N, \infty)$	$ G_0(j\omega)  \leq \frac{k_N}{1 + k_N}, \quad \forall \omega \in [\omega_N, \infty)$
$\frac{ G_0(j\omega) }{ 1 + G_0(j\omega) } \leq \frac{1}{\ell_m(\omega)}, \quad \forall \omega$	$ G_0(j\omega)  \leq \frac{1}{1 + \ell_m(\omega)}$ ( or $ G_0(j\omega)  \geq \frac{1}{1 - \ell_m(\omega)}$ )

**Table 7.1.** Summary of the relationship between closed-loop specifications and open-loop constraints for the loop shaping design method



**Figure 7.2.** Typical open-loop specifications for the loop shaping design method

**Attention!** The conditions derived above for the open-loop gain  $G_0(j\omega)$  are *sufficient* for the original closed-loop specifications to hold, but they are not *necessary*. When the open-loop gain “almost” verifies the conditions derived, it may be worth it to check directly if it verifies the original closed-loop conditions.

This is actually crucial for the conditions (7.10)–(7.11) that arise from robustness with respect to multiplicative uncertainty, because around the crossover frequency  $|G_0(j\omega_c)| \approx 1$  will not satisfy either of these conditions, but it will generally satisfy the original condition (7.8) (as long as  $\ell_m(\omega)$  is not too large).  $\square$

### 7.3 Open-loop gain shaping

In classical lead/lag compensation, one starts with a basic unit-gain controller

$$C(s) = 1$$

and “adds” to it appropriate blocks to shape the desired open-loop gain

$$G_0(s) := C(s)P_0(s),$$

so that it satisfies the appropriate open-loop constraints. This shaping can be achieved using three basic tools.

1. *Proportional gain.* Multiplying the controller by a constant  $k$  moves the magnitude Bode plot up and down, without changing its phase.
2. *Lead compensation.* Multiplying the controller by a lead block with transfer function

$$C_{\text{lead}}(s) = \frac{Ts + 1}{\alpha Ts + 1}, \quad \alpha < 1$$

increases the phase margin when placed at the cross-over frequency. Figure 7.3(a) shows the Bode plot of a lead compensator.

3. *Lag compensation.* Multiplying the controller by a lag block with transfer function

$$C_{\text{lag}}(s) = \frac{s/z + 1}{s/p + 1}, \quad p < z$$

decreases the high-frequency gain. Figure 7.3(b) shows the Bode plot of a lag compensator.

**Note.** One actually does not “add” to the controller. To be precise, one *multiplies* the controller by appropriate gain, lead, and lag blocks. However, this does correspond to additions in the magnitude (in dBs) and phase Bode plots.

**Note.** A lead compensator also increases the cross-over frequency, so it may require some trial and error to get the peak of the phase right at the cross-over frequency.

**Note.** A lag compensator also increases the phase, so it can decrease the phase margin. To avoid this, one should only introduce lag compensation away from the cross-over frequency.

### 7.4 Exercises

**7.1 (Loop-shape 1).** Consider again the nominal process model and multiplicative uncertainty that you obtained in Exercise 6.2. Design a controller for this process that achieves stability for all admissible process models and that exhibits:

1. zero steady-state error to a step input,
2. Phase Margin no smaller than 60degrees,
3. steady-state error for sinusoidal inputs with frequencies  $\omega < 0.1\text{rad/sec}$  smaller than 1/50 (−34dB), and
4. Rise time no slower than .3sec.  $\square$

**7.2 (Loop-shape 2).** Consider the following nominal transfer function and uncertainty bound:

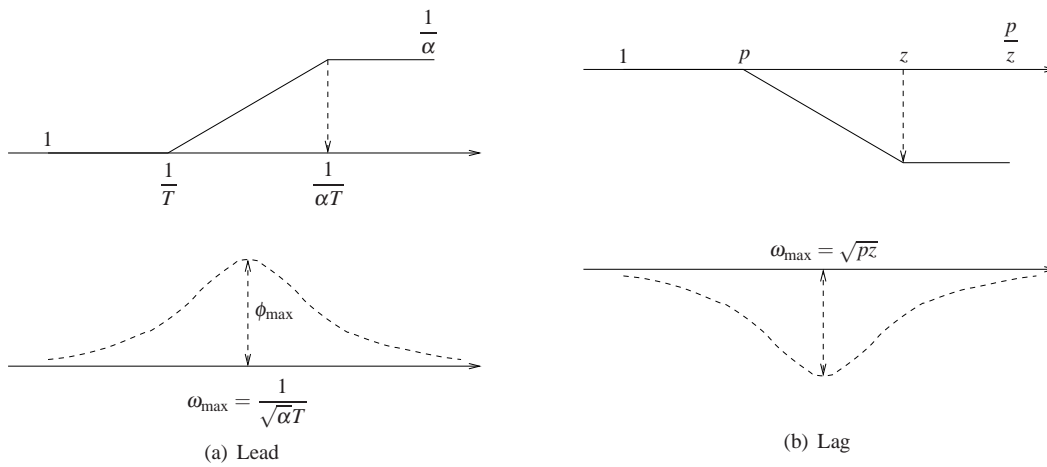
$$P_0(s) = \frac{1}{s(1+s/5)(1+s/20)}, \quad \ell_m(\omega) = |L(j\omega)|,$$

where

$$L(s) := \frac{2.5}{(1+s/20)^2}.$$

Use loop-shaping to design a controller that achieves stability for all admissible process models and that exhibits:

1. steady-state error to a ramp input no larger than .01,
2. Phase Margin no smaller than 45degrees,
3. steady-state error for sinusoidal inputs with frequencies  $\omega < 0.2\text{rad/sec}$  smaller than  $1/250$  ( $-50\text{dB}$ ), and
4. attenuation of measurement noise by at least a factor of 100 ( $-40\text{dB}$ ) for frequencies greater than  $200\text{rad/sec}$ . □



**Figure 7.3.** Bode plots of lead/lag compensators. The maximum lead phase angle is given by  $\phi_{\max} = \arcsin \frac{1-\alpha}{1+\alpha}$ ; therefore, to obtain a desired given lead angle  $\phi_{\max}$  one sets  $\alpha = \frac{1-\sin \phi_{\max}}{1+\sin \phi_{\max}}$ .



## **Part III**

# **LQG/LQR Controller Design**



# Introduction to LQG/LQR controller design

In *optimal control* one attempts to find a controller that provides the best possible performance with respect to some given *measure of performance*. E.g., the controller that uses the least amount of control-signal energy to take the output to zero. In this case the measure of performance (also called the *optimality criterion*) would be the control-signal energy.

In general, optimality with respect to some criterion is not the only desirable property for a controller. One would also like stability of the closed-loop system, good gain and phase margins, robustness with respect to unmodeled dynamics, etc.

In this section we study controllers that are optimal with respect to energy-like criteria. These are particularly interesting because the minimization procedure *automatically produces controllers that are stable and somewhat robust*. In fact, the controllers obtained through this procedure are generally so good that we often use them *even when we do not necessarily care about optimizing for energy*. Moreover, this procedure is applicable to *multiple-input/multiple-output* processes for which classical designs are difficult to apply.

## Pre-requisites

1. Basic knowledge of state-space models (briefly reviewed here)
2. Familiarity with basic vector and matrix operations.
3. Knowledge of MATLAB/Simulink.



# Chapter 8

## Review of State-space models

### Contents

8.1	State-space models	77
8.2	Input-output relations	78
8.3	Realizations	79
8.4	Controllability and observability	79
8.5	Stability	80
8.6	MATLAB hints	80

### 8.1 State-space models

Consider the system in Figure 8.1 with  $m$  inputs and  $k$  outputs. A *state-space* model for this system

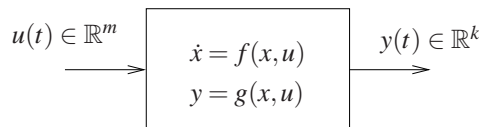


Figure 8.1. System with  $m$  inputs and  $k$  outputs

relates the input and output of a system using the following first-order vector ordinary differential equation

$$\dot{x} = f(x, u), \quad y = g(x, u). \quad (8.1)$$

where  $x \in \mathbb{R}^n$  is called the state of system. In this Chapter we restrict our attention to *linear time-invariant (LTI)* systems for which the functions  $f(\cdot, \cdot)$  and  $g(\cdot, \cdot)$  are linear. In this case, (8.1) has the special form

$$\dot{x} = Ax + Bu, \quad y = Cx + Du, \quad (8.2)$$

where  $A$  is a  $n \times n$  matrix,  $B$  a  $n \times m$  matrix,  $C$  a  $k \times n$  matrix, and  $D$  a  $k \times m$  matrix.

**Example 8.1** (Aircraft roll-dynamics). Figure 8.2 shows the roll-angle dynamics of an aircraft [9, p. 381]. Defining

$$x := [\theta \quad \omega \quad \tau]'$$

we conclude that

$$\dot{x} = Ax + Bu$$

**MATLAB<sup>®</sup> Hint 25.**  
`ss(A,B,C,D)` creates a LTI state-space model with realization (8.2)... **► p. 80**

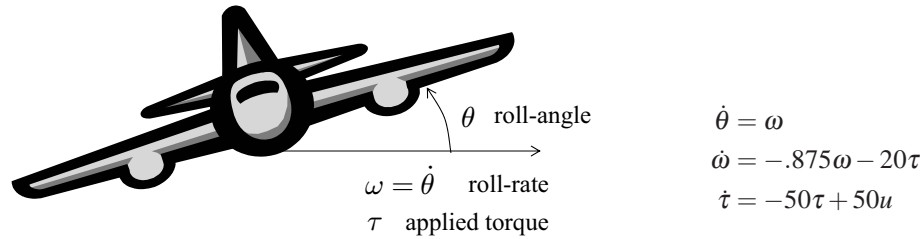


Figure 8.2. Aircraft roll-angle dynamics

with

$$A := \begin{bmatrix} 0 & 1 & 0 \\ 0 & -.875 & -20 \\ 0 & 0 & -50 \end{bmatrix}, \quad B := \begin{bmatrix} 0 \\ 0 \\ 50 \end{bmatrix}.$$

If we have both  $\theta$  and  $\omega$  available for control, we can define

$$y := [\theta \quad \omega]' = Cx + Du$$

with

$$C := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad D := \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \quad \square$$

## 8.2 Input-output relations

**Note 1.** The (unilateral) Laplace transform of a signal  $x(t)$  is given by

$$X(s) := \int_0^{\infty} e^{-st} x(t) dt.$$

See [4, Appendix A] for a review of Laplace transforms.

The transfer-function of this system can be found by taking Laplace transforms of (8.2):

$$\begin{cases} \dot{x} = Ax + Bu, \\ y = Cx + Du, \end{cases} \xrightarrow{\mathcal{L}} \begin{cases} sX(s) = AX(s) + BU(s), \\ Y(s) = CX(s) + DU(s), \end{cases}$$

where  $X(s)$ ,  $U(s)$ , and  $Y(s)$  denote the Laplace transforms of  $x(t)$ ,  $u(t)$ , and  $y(t)$ . Solving for  $X(s)$ , we get

$$(sI - A)X(s) = BU(s) \Leftrightarrow X(s) = (sI - A)^{-1}BU(s)$$

and therefore

$$Y(s) = C(sI - A)^{-1}BU(s) + DU(s) = (C(sI - A)^{-1}B + D)U(s).$$

Defining

$$T(s) := C(sI - A)^{-1}B + D,$$

we conclude that

$$Y(s) = T(s)U(s). \quad (8.3)$$

To emphasize the fact that  $T(s)$  is a  $k \times m$  matrix, we call it the *transfer-matrix* of the system (8.2).

The relation (8.3) between the Laplace transforms of the input and the output of the system is only valid for zero initial conditions, i.e., when  $x(0) = 0$ . The general solution to the system (8.2) in the *time domain* is given by

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-s)}Bu(s)ds, \quad (8.4)$$

$$y(t) = Ce^{At}x(0) + \int_0^t Ce^{A(t-s)}Bu(s)ds + Du(t), \quad \forall t \geq 0. \quad (8.5)$$

### MATLAB<sup>®</sup> Hint 1.

`tf(num,den)` creates a transfer-function with numerator and denominator specified by `num`, `den`... ► p. 12

### MATLAB<sup>®</sup> Hint 2.

`zpk(z,p,k)` creates a transfer-function with zeros, poles, and gain specified by `z`, `p`, `k`... ► p. 12

### MATLAB<sup>®</sup> Hint 26.

`tf(sys_ss)` and `zpk(sys_ss)` compute the transfer-function of the state-space model `sys_ss`... ► p. 80

Equation (8.4) is called the *variation of constants formula*.

**Example 8.2** (Aircraft roll-dynamics). The transfer-function for the state-space model in Example 8.1 is given by:

$$T(s) = \begin{bmatrix} \frac{-1000}{s(s+.875)(s+50)} \\ \frac{-1000}{(s+.875)(s+50)} \end{bmatrix}$$

□

**MATLAB® Hint 27.** `expm` computes the exponential of a matrix... ▶ p. 80

## 8.3 Realizations

Consider a transfer-matrix

$$T(s) = \begin{bmatrix} T_{11}(s) & T_{12}(s) & \cdots & T_{1m}(s) \\ T_{21}(s) & T_{22}(s) & \cdots & T_{2m}(s) \\ \vdots & \vdots & \ddots & \vdots \\ T_{k1}(s) & T_{k2}(s) & \cdots & T_{km}(s) \end{bmatrix},$$

where all the  $T_{ij}(s)$  are given by a ratio of polynomials with the degree of the numerator smaller than or equal to the degree of the denominator. It is always possible to find matrices  $A, B, C, D$  such that

$$T(s) = C(sI - A)^{-1}B + D.$$

This means that it is always possible to find a state-space model like (8.2) whose transfer-matrix is precisely  $T(s)$ . The model (8.2) is called a *realization* of  $T(s)$ .

**MATLAB® Hint 28.** `ss(sys_tf)` computes a realization of the transfer-function `sys_tf`... ▶ p. 80

**Attention!** Realizations are not unique, i.e., several state-space models may have the same transfer function.

## 8.4 Controllability and observability

The system (8.2) is said to be *controllable* when given any initial state  $x_i \in \mathbb{R}^n$ , any final state  $x_f \in \mathbb{R}^n$ , and any finite time  $T$ , one can find an input signal  $u(t)$  that takes the state of (8.2) from  $x_i$  to  $x_f$  in the interval of time  $0 \leq t \leq T$ , i.e., when there exists an input  $u(t)$  such that

$$x_f = e^{AT}x_i + \int_0^T e^{A(T-s)}Bu(s)ds.$$

To determine if a system is controllable, one can compute the *controllability matrix*, which is defined by

$$\mathcal{C} := [B \quad AB \quad A^2B \quad \cdots \quad A^{n-1}B].$$

The system is controllable if and only if this matrix has rank equal to the size  $n$  of the state vector.

The system (8.2) is said to be *observable* when one can determine the initial condition  $x(0)$  by simply looking at the input and output signals  $u(t)$  and  $y(t)$  on a certain interval of time  $0 \leq t \leq T$ , i.e., one can solve

$$y(t) = Ce^{At}x(0) + \int_0^t Ce^{A(t-s)}Bu(s)ds + Du(t), \quad \forall 0 \leq t \leq T,$$

uniquely for the unknown  $x(0)$ . To determine if a system is observable, one can compute the *observability matrix*, which is defined by

$$\mathcal{O} := \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}.$$

The system is observable if and only if this matrix has rank equal to the size  $n$  of the state vector.

**MATLAB® Hint 29.** `ctrb(sys)` computes the controllability matrix of the state-space system `sys`. Alternatively, one can use directly `ctrb(A,B)`... ▶ p. 81

**MATLAB® Hint 30.** `rank(M)` computes the rank of a matrix `M`.

**MATLAB® Hint 31.** `obsv(sys)` computes the observability matrix of the state-space system `sys`. Alternatively, one can use directly `obsv(A,C)`... ▶ p. 81

**Example 8.3** (Aircraft roll-dynamics). The controllability and observability matrices for the state-space model in Example 8.1 are given by:

$$\mathcal{C} = \begin{bmatrix} 0 & 0 & -1000 \\ 0 & -1000 & 50875 \\ 50 & -2500 & 125000 \end{bmatrix}, \quad \mathcal{O} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & -.875 & -20 \\ 0 & -.875 & -20 \\ 0 & .7656 & 1017.5 \end{bmatrix}.$$

Both matrices have rank 3 so the system is both controllable and observable.  $\square$

## 8.5 Stability

The system (8.2) is *asymptotically stable* when all eigenvalues of  $A$  have negative real parts. In this case, for any bounded input  $u(t)$  the output  $y(t)$  and the state  $x(t)$  are also bounded, i.e.,

$$\|u(t)\| \leq c_1, \forall t \geq 0 \Rightarrow \|y(t)\| \leq c_2, \|x(t)\| \leq c_3 \quad \forall t \geq 0.$$

Moreover, if  $u(t)$  converges to zero as  $t \rightarrow \infty$ , then  $x(t)$  and  $y(t)$  also converge to zero as  $t \rightarrow \infty$ .

**Example 8.4** (Aircraft roll-dynamics). The eigenvalues of the matrix the  $A$  matrix for the state-space model in Example 8.1 are  $\{0, -.875, -50\}$  so the system is not asymptotically stable.  $\square$

## 8.6 MATLAB hints

**MATLAB<sup>®</sup> Hint 25** (ss). The command `sys_ss=ss(A,B,C,D)` assigns to `sys_ss` a MATLAB LTI state-space model with realization

$$\dot{x} = Ax + Bu, \quad y = Cx + Du.$$

Optionally, one can specify the names of the inputs, outputs, and state to be used in subsequent plots as follows:

```
sys_ss=ss(A,B,C,D,...
'InputName',{ 'input1', 'input2', ... },...
'OutputName',{ 'output1', 'output2', ... },...
'StateName',{ 'input1', 'input2', ... })
```

The number of elements in the bracketed lists must match the number of inputs, outputs, and state variables.  $\square$

**MATLAB<sup>®</sup> Hint 26** (tf). The commands `tf(sys_ss)` and `zpk(sys_ss)` compute the transfer-function of the state-space model `sys_ss` specified as in Matlab Hint 25.

`tf(sys_ss)` stores (and displays) the transfer function as a ratio of polynomials on  $s$ .

`zpk(sys_ss)` stores (and displays) the polynomials factored as the product of monomials (for the real roots) and binomials (for the complex roots). This form highlights the zeros and poles of the system.  $\square$

**MATLAB<sup>®</sup> Hint 28** (ss). The command `ss(sys_tf)` computes the state-space model of the transfer function `sys` specified as in Matlab Hints 1 or 2.  $\square$

**MATLAB<sup>®</sup> Hint 27** (expm). The command `expm(M)` computes the matrix exponential  $e^M$ . With the symbolic toolbox, this command can be used to compute  $e^{At}$  symbolically as follows:

**MATLAB<sup>®</sup> Hint 32.** `eig(A)` computes the eigenvalues of the matrix  $A$ ...  $\blacktriangleright$  p. 81

```
syms t
expm(A*t)
```

The first command defines  $t$  as a symbolic variable and the second computes  $e^{At}$  (assuming that the matrix  $A$  has been previously defined). □

**MATLAB® Hint 29** (`ctrb`). The command `ctrb(sys)` computes the controllability matrix of the system `sys`. The system must be specified by a state-space model using, e.g., `sys=ss(A,B,C,D)`, where  $A, B, C, D$  are a realization of the system. Alternatively, one can use directly `ctrb(A,B)`. □

**MATLAB® Hint 31** (`obsv`). The command `obsv(sys)` computes the observability matrix of the system `sys`. The system must be specified by a state-space model using, e.g., `sys=ss(A,B,C,D)`, where  $A, B, C, D$  are a realization of the system. Alternatively, one can use directly `obsv(A,C)`. □

**MATLAB® Hint 32** (`eig`). The command `eig(A)` computes the eigenvalues of the matrix  $A$ . Alternatively, `eig(sys)` computes the eigenvalues of the  $A$  matrix for a state-space system `sys` specified by `sys=ss(A,B,C,D)`, where  $A, B, C, D$  are a realization of the system. □



# Chapter 9

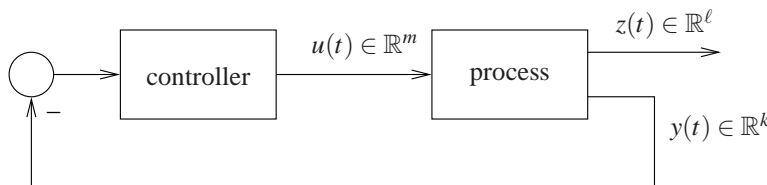
## Linear Quadratic Regulation (LQR)

### Contents

9.1	Feedback configuration	83
9.2	Optimal Regulation	84
9.3	State-Feedback LQR	85
9.4	Stability and Robustness	86
9.5	Loop-shaping control using LQR	88
9.6	MATLAB hints	90
9.7	To probe further	91
9.8	Exercises	92

### 9.1 Feedback configuration

Figure 9.1 shows the feedback configuration for the *Linear quadratic regulation (LQR) problem*.



**Figure 9.1.** Linear quadratic regulation (LQR) feedback configuration. Note the *negative feedback* and the *absence of a reference signal*. Reference signals will be introduced in Chapter 11.

In this configuration, the state-space model of the process is of the form

$$\dot{x} = Ax + Bu, \quad y = Cx, \quad z = Gx + Hu. \quad (9.1)$$

and has two distinct outputs:

1. The *measured output*  $y(t) \in \mathbb{R}^k$  corresponds to the signal(s) that can be measured and are therefore available for control. If the controller transfer-matrix is  $C(s)$ , we have

$$Y(s) = -C(s)U(s),$$

where  $Y(s)$  and  $U(s)$  denote the Laplace transforms of the process input  $u(t)$  and the measured output  $y(t)$ , respectively.

2. The *controlled output*  $z(t) \in \mathbb{R}^\ell$  corresponds to a signal that one would like to make as small as possible in the shortest possible amount of time.

**Note 44.** In the context of Example 8.1, one could imagine that, while both  $\theta$  and  $\omega$  can be measured, one may mostly want to regulate the roll angle  $\theta$ ... ► p. 77

Sometimes  $z(t) = y(t)$ , which means that our control objective is to make the whole measured output very small. However, when the output  $y(t)$  is a vector, often one simply needs to make one of the measured outputs  $y_1(t)$  small. In this case, one chooses  $z(t) = y_1(t)$ .

In some situations one chooses

$$z(t) = \begin{bmatrix} y_1(t) \\ \dot{y}_1(t) \end{bmatrix},$$

which means that we want to make both the measured output  $y_1(t)$  and its derivative  $\dot{y}_1(t)$  very small. Many other options are possible.

The choice of  $z$  should be viewed as a design parameter. In Section 9.5 we will study the impact of this choice in the performance of the closed-loop.

## 9.2 Optimal Regulation

The LQR problem is defined as follows:

**Problem 9.1** (Optimal LQR). Find the controller transfer-matrix  $C(s)$  that makes the following criterion as small as possible

$$J_{\text{LQR}} := \int_0^{\infty} \|z(t)\|^2 + \rho \|u(t)\|^2 dt, \quad (9.2)$$

**Notation 5.** Given an  $m$ -vector,

$$v = [v_1 \ v_2 \ \dots \ v_m],$$

$\|v\|$  denotes the Euclidean norm of  $v$ , i.e.,

$$\|v\| = \sqrt{v'v} = \left( \sum_{i=1}^m v_i^2 \right)^{\frac{1}{2}}.$$

where  $\rho$  is a positive constant. □

The term

$$\int_0^{\infty} \|z(t)\|^2 dt$$

corresponds to the *energy of the controlled output* and the term

$$\int_0^{\infty} \|u(t)\|^2 dt$$

to the *energy of the control signal*. In LQR one seeks a controller that minimizes both energies. However, decreasing the energy of the controlled output will require a large control signal and a small control signal will lead to large controlled outputs. The role of the constant  $\rho$  is to establish a trade-off between these conflicting goals:

1. When we chose  $\rho$  very large, the most effective way to decrease  $J_{\text{LQR}}$  is to use little control, at the expense of a large controlled output.
2. When we chose  $\rho$  very small, the most effective way to decrease  $J_{\text{LQR}}$  is to obtain a very small controlled output, even if this is achieved at the expense of a large control input.

Often the *optimal LQR problem* is defined more generally and consists of finding the controller transfer-matrix  $C(s)$  that minimizes

$$J_{\text{LQR}} := \int_0^{\infty} z(t)' Q z(t) + \rho u'(t) R u(t) dt, \quad (9.3)$$

**Note 45.** The most general form for the quadratic criteria is

$$\int_0^{\infty} x' \bar{Q} x + u' \bar{R} u + 2x' \bar{N} u dt$$

...

► p. 91

where  $Q$  is an  $\ell \times \ell$  symmetric positive-definite matrix,  $R$  an  $m \times m$  symmetric positive-definite matrix, and  $\rho$  a positive constant.

**Bryson's rule** A first choice for the matrices  $Q$  and  $R$  in (9.3) is given by the *Bryson's rule* [4, p. 537]: select  $Q$  and  $R$  diagonal with

$$Q_{ii} = \frac{1}{\text{maximum acceptable value of } z_i^2}, \quad i \in \{1, 2, \dots, \ell\}$$

$$R_{jj} = \frac{1}{\text{maximum acceptable value of } u_j^2}, \quad j \in \{1, 2, \dots, m\},$$

which corresponds to the following criteria

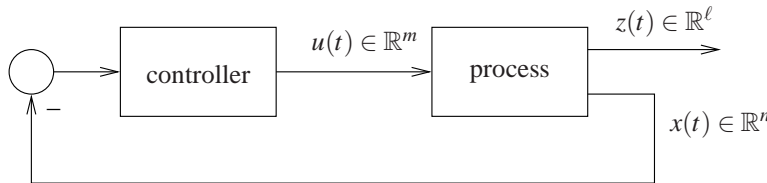
$$J_{\text{LQR}} := \int_0^\infty \left( \sum_{i=1}^{\ell} Q_{ii} z_i(t)^2 + \rho \sum_{j=1}^m R_{jj} u_j(t)^2 \right) dt.$$

In essence the Bryson's rule scales the variables that appear in  $J_{\text{LQR}}$  so that the *maximum acceptable value for each term is one*. This is especially important when the units used for the different components of  $u$  and  $z$  make the values for these variables numerically very different from each other.

Although Bryson's rule sometimes gives good results, often it is just the starting point to a trial-and-error iterative design procedure aimed at obtaining desirable properties for the closed-loop system. In Section 9.5 we will discuss systematic methods to choose the weights in the LQR criterion.

### 9.3 State-Feedback LQR

In the *state-feedback* version of the LQR problem (Figure 9.2), we assume that the whole state  $x$  can be measured and therefore it is available for control.



**Figure 9.2.** Linear quadratic regulation (LQR) with state feedback

*Solution to the optimal state-feedback LQR Problem 9.1.* The optimal state-feedback LQR controller for the criteria (9.3) is a simple matrix gain of the form

$$u = -Kx \quad (9.4)$$

where  $K$  is the  $m \times n$  matrix given by

$$K = (H'QH + \rho R)^{-1}(B'P + H'QG)$$

and  $P$  is the unique positive-definite solution to the following equation

$$A'P + PA + G'QG - (PB + G'QH)(H'QH + \rho R)^{-1}(B'P + H'QG) = 0,$$

known as the *Algebraic Riccati Equation (ARE)*. □

**Note 46.** Other choices for the matrices  $Q$  and  $R$  are possible but one should always choose both matrices to be positive-definite. We recall that a symmetric  $q \times q$  matrix  $M$  is *positive-definite* if  $x'Mx > 0$ , for every nonzero vector  $x \in \mathbb{R}^q$  ... ► p. 91

**MATLAB® Hint 33.** `lqr` computes the optimal state-feedback controller gain  $K$  ... ► p. 90

## 9.4 Stability and Robustness

**Note 47.** A system  $\dot{x} = Ax + Bu$  is *asymptotically stable* when all eigenvalues of  $A$  have negative real parts. See Section 8.5... ▶ p. 80

**Note 48.** The definitions and tests for controllability and observability are reviewed in Section 8.4... ▶ p. 79

**Attention!** When selecting the measured output  $z$ , it is important to verify that the observability condition is satisfied.

The state-feedback control law (9.4), results in a closed-loop system of the form

$$\dot{x} = (A - BK)x.$$

A crucial property of LQR controller design is that this *closed-loop is asymptotically stable* (i.e., all the eigenvalues of  $A - BK$  have negative real part) as long as the following two conditions hold:

1. The system (9.1) is controllable.
2. The system (9.1) is observable when we ignore  $y$  and regard  $z$  as the sole output .

Perhaps even more important is the fact that LQR controllers are *inherently robust with respect to process uncertainty*. To understand why, consider the open-loop transfer-matrix from the process' input  $u$  to the controller's output  $\bar{u}$  (Figure 9.3). The state-space model from  $u$  to  $\bar{u}$  is given by

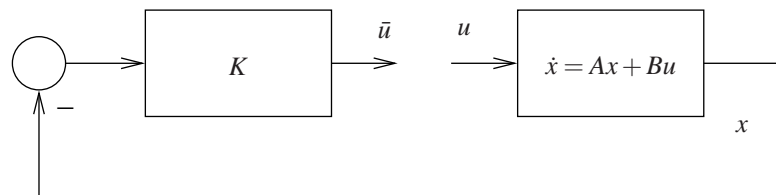


Figure 9.3. State-feedback open-loop gain

$$\dot{x} = Ax + Bu, \quad \bar{u} = -Kx,$$

which corresponds to the following open-loop *negative feedback*  $m \times m$  transfer-matrix

$$G_0(s) = K(sI - A)^{-1}B.$$

**Note 49.** LQR controllers also exhibit robustness properties for multiple-input processes. However, in this case  $G_0(s)$  is a  $m \times m$  transfer-matrix and one needs a *multi-variable Nyquist criterion*.

We focus our attention in single-input processes ( $m = 1$ ), for which  $G_0(s)$  is a scalar transfer-function and the following holds:

**Kalman's Inequality.** When  $H'G = 0$ , the Nyquist plot of  $G_0(j\omega)$  does not enter a circle of radius one around  $-1$ , i.e.,

$$|1 + G_0(j\omega)| \geq 1, \quad \forall \omega \in \mathbb{R}. \quad \square$$

Kalman's Inequality is represented graphically in Figure 9.4 and has several significant implications, which are discussed next.

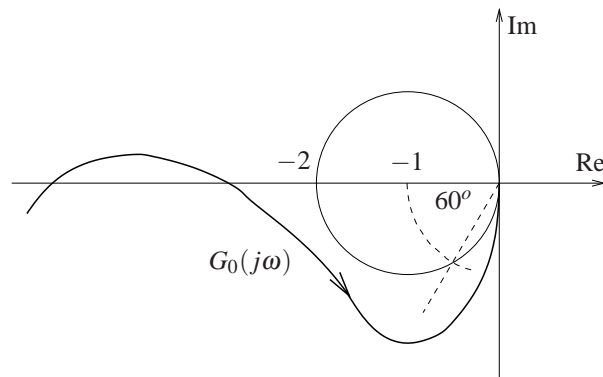
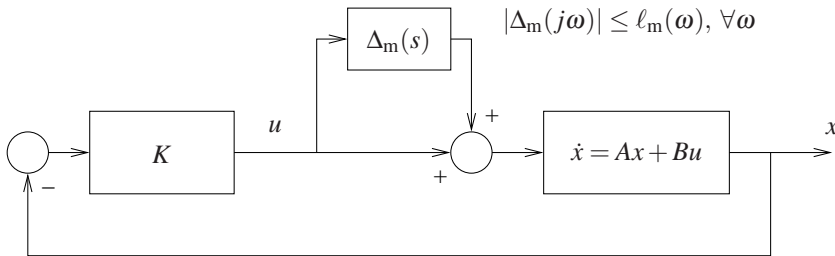


Figure 9.4. Nyquist plot for a LQR state-feedback controller

**Positive gain margin** If the process' gain is multiplied by a constant  $k > 1$ , its Nyquist plot simply expands radially and therefore the number of encirclements does not change. This corresponds to a *positive gain margin of  $+\infty$* .

**Negative gain margin** If the process' gain is multiplied by a constant  $.5 < k < 1$ , its Nyquist plot contracts radially but the number of encirclements still does not change. This corresponds to a *negative gain margin of  $20\log_{10}(.5) = -6\text{dB}$* .

**Phase margin** If the process' phase increases by  $\theta \in [-60, 60]$  degrees, its Nyquist plots rotates by  $\theta$  but the number of encirclements still does not change. This corresponds to a *phase margin of  $\pm 60$  degrees*.



**Figure 9.5.** Unity feedback configuration with multiplicative uncertainty

**Multiplicative uncertainty** Kalman's inequality guarantees that

$$\left| \frac{G_0(j\omega)}{1 + G_0(j\omega)} \right| \leq 2. \quad (9.5)$$

**Note 50.** Why?...

► p. 91

Since, we know that the closed-loop system in Figure 9.5 remains stable for every multiplicative uncertainty block  $\Delta_m(j\omega)$  with norm smaller than  $\ell_m(\omega)$ , as long as

$$\left| \frac{G_0(j\omega)}{1 + G_0(j\omega)} \right| < \frac{1}{\ell_m(\omega)}, \quad (9.6)$$

we conclude that an LQR controller provides *robust stability with respect to any multiplicative uncertainty with magnitude smaller than  $\frac{1}{2}$* , because we then have

$$\left| \frac{G_0(j\omega)}{1 + G_0(j\omega)} \right| \leq 2 < \frac{1}{\ell_m(\omega)}.$$

However, much larger additive uncertainties may be admissible: e.g., when  $G_0(j\omega) \gg 1$ , (9.6) will hold for  $\ell_m(\omega)$  almost equal to 1; and when  $G_0(j\omega) \ll 1$ , (9.6) will hold for  $\ell_m(\omega)$  almost equal to  $1/G_0(j\omega) \gg 1$ .

**Attention!** Kalman's inequality is only valid when  $H'G = 0$ . When this is not the case, LQR controllers can be significantly less robust. This limits to some extent the controlled outputs that can be placed in  $z$ .

For example, consider the process  $\dot{x} = Ax + Bu$  and suppose that we want to regulate a particular output  $y_1 = C_1x$ . Choosing

$$z = y_1 = C_1x$$

leads to  $G = C_1$  and  $H = 0$  and therefore  $H'G = 0$ , for which Kalman's inequality holds. However, choosing

$$z = \begin{bmatrix} y_1 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} C_1x \\ C_1\dot{x} \end{bmatrix} = \begin{bmatrix} C_1x \\ C_1Ax + C_1Bu \end{bmatrix} = \begin{bmatrix} C_1 \\ C_1A \end{bmatrix} x + \begin{bmatrix} 0 \\ C_1B \end{bmatrix} u,$$

leads to

$$G = \begin{bmatrix} C_1 \\ C_1 A \end{bmatrix}, \quad H = \begin{bmatrix} 0 \\ C_1 B \end{bmatrix},$$

and therefore

$$H'G = B'C_1' C_1 A,$$

which may not be equal to zero. □

## 9.5 Loop-shaping control using LQR

Although Bryson's rule sometimes gives good results, it may not suffice to satisfy tight control specifications. We will see next a few other rules that allow us to actually do loop-shaping using LQR. We restrict our attention to the single-input case ( $m = 1$ ) and  $R = 1$ ,  $Q = I$ , which corresponds to

$$J_{\text{LQR}} := \int_0^\infty \|z(t)\|^2 + \rho u(t)^2 dt.$$

**Low-frequency open-loop gain** For the range of frequencies for which  $|G_0(j\omega)| \gg 1$  (typically low frequencies), we have that

$$|G_0(j\omega)| \approx \frac{\|P_z(j\omega)\|}{\sqrt{H'H + \rho}}$$

where

$$P_z(s) := G(sI - A)^{-1}B + H$$

is the transfer function from the control signal  $u$  to the controlled output  $z$ . To understand the implications of this formula, it is instructive to consider two fairly typical cases:

1. When  $z = y_1$ , with  $y_1 := C_1 x$  scalar, we have

$$|G_0(j\omega)| \approx \frac{|P_1(j\omega)|}{\sqrt{H'H + \rho}}$$

where

$$P_1(s) := C_1(sI - A)^{-1}B$$

is the transfer function from the control input  $u$  to the output  $y_1$ . In this case,

- (a) the “shape” of the magnitude of the open-loop gain  $G_0(j\omega)$  is determined by the magnitude of the transfer function from the control input  $u$  to the output  $y_1$ ;
  - (b) the parameter  $\rho$  moves the magnitude Bode plot up and down (more precisely  $H'H + \rho$ ).
2. When  $z = [y_1 \quad \gamma \dot{y}_1]'$ , we can show that

$$|G_0(j\omega)| \approx \frac{|1 + j\gamma\omega| |P_1(j\omega)|}{\sqrt{H'H + \rho}}. \quad (9.7)$$

In this case the low-frequency open-loop gain mimics the process transfer function from  $u$  to  $y$ , with an extra zero at  $1/\gamma$  and scaled by  $\frac{1}{\sqrt{H'H + \rho}}$ . Thus

- (a)  $\rho$  moves the magnitude Bode plot up and down (more precisely  $H'H + \rho$ ),
- (b) large values for  $\gamma$  lead to a low-frequency zero and generally result in a larger phase margin (above the minimum of 60 degrees) and smaller overshoot in the step response. However, this is often achieved at the expense of a slower response.

**Note 51.** If the transfer function from  $u$  to  $y_1$  has two more poles than zeros, then one can show that  $C_1 B = 0$  and  $H = 0$ . In this case, Kalman's inequality holds also for this choice of  $z$ .

**MATLAB<sup>®</sup> Hint 34.**  
`sigma(sys)` draws the norm-Bode plot of the system `sys...` ▶ p. 90

**Note 52.** Although the magnitude of  $G_0(j\omega)$  mimics the magnitude of  $P_1(j\omega)$ , the phase of the open-loop gain  $G_0(j\omega)$  always leads to a stable closed-loop with an appropriate phase margin.

**Note 53.** Why?... ▶ p. 91

**High-frequency open-loop gain** For  $\omega \gg 1$ , we have that

$$|G_0(j\omega)| \approx \frac{c}{\omega\sqrt{\rho}},$$

for some constant  $c$ . We thus conclude the following:

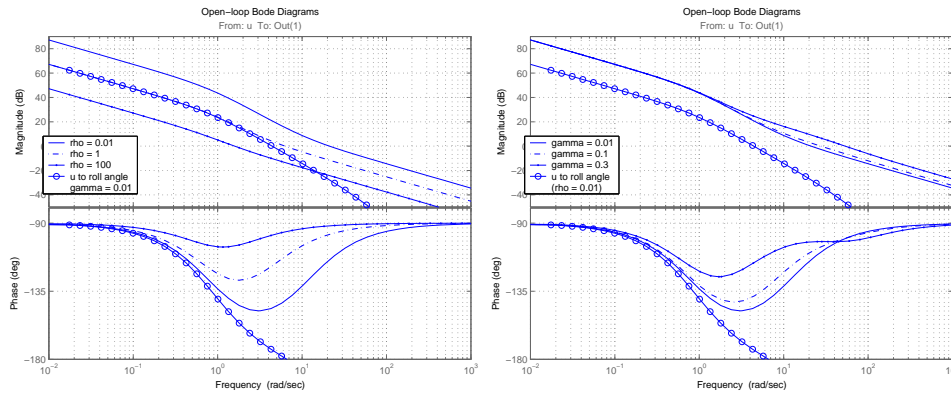
1. LQR controllers always exhibit a high-frequency magnitude decay of  $-20\text{dB/decade}$ .
2. The cross-over frequency is approximately given by

$$\frac{c}{\omega_{\text{cross}}\sqrt{\rho}} \approx 1 \Leftrightarrow \omega_{\text{cross}} \approx \frac{c}{\sqrt{\rho}},$$

which shows that the cross-over frequency is proportional to  $1/\sqrt{\rho}$  and generally small values for  $\rho$  result in faster step responses.

**Attention!** The (slow)  $-20\text{dB/decade}$  magnitude decrease is the main shortcoming of state-feedback LQR controllers because it may not be sufficient to clear high-frequency upper bounds on the open-loop gain needed to reject disturbances and/or for robustness with respect to process uncertainty. We will see in Section 10.6 that this can actually be improved with output-feedback controllers.  $\square$

**Example 9.1** (Aircraft roll-dynamics). Figure 9.6 shows Bode plots of the open-loop gain  $G_0(s) = K(sI - A)^{-1}B$  for several LQR controllers obtained for the aircraft roll-dynamics in Example 8.1. The controlled output was chosen to be  $z := [\theta \ \gamma\theta]'$ , which corresponds to



(a) Open-loop gain for several values of  $\rho$ . This parameter allows us to move the whole magnitude Bode plot up and down. (b) Open-loop gain for several values of  $\gamma$ . Larger values for this parameter result in a larger phase margin.

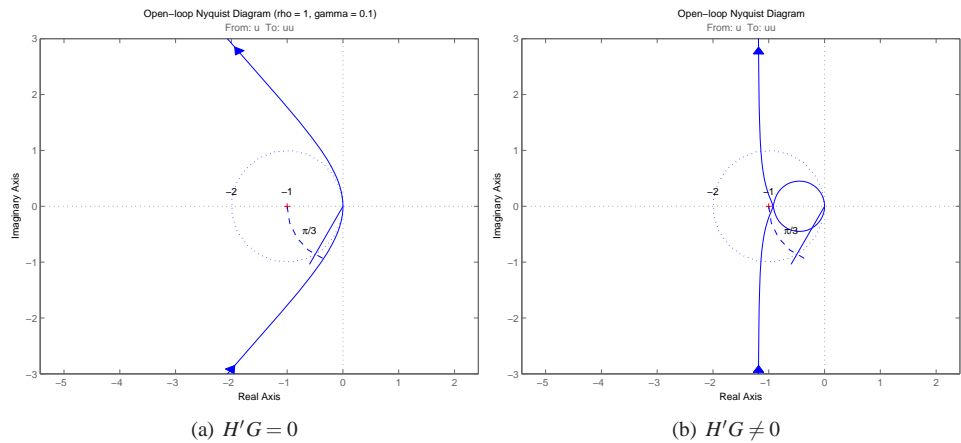
**Figure 9.6.** Bode plots for the open-loop gain of the LQR controllers in Example 9.1. As expected, for low frequencies the open-loop gain magnitude matches that of the process transfer function from  $u$  to  $\theta$  (but with significantly lower/better phase) and at high-frequencies the gain's magnitude falls at  $-20\text{dB/decade}$ .

$$G := \begin{bmatrix} 1 & 0 & 0 \\ 0 & \gamma & 0 \end{bmatrix}, \quad H := \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The controllers were obtained with  $R = 1$ ,  $Q = I_{2 \times 2}$ , and several values for  $\rho$  and  $\gamma$ . Figure 9.6(a) shows the open-loop gain for several values of  $\rho$  and Figure 9.6(b) shows the open-loop gain for several values of  $\gamma$ .

Figure 9.7 shows Nyquist plots of the open-loop gain  $G_0(s) = K(sI - A)^{-1}B$  for different choices of the controlled output  $z$ . In Figure 9.7(a)  $z := [\theta \ \dot{\theta}]'$ , which corresponds to

$$G := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad H := \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$



**Figure 9.7.** Bode plots for the open-loop gain of the LQR controllers in Example 9.1

In this case,  $H'G = [0 \ 0 \ 0]$  and Kalman's inequality holds as can be seen in the Nyquist plot. In Figure 9.7(b), the controlled output was chosen to be  $z := [\theta \ \dot{\tau}]'$ , which corresponds to

$$G := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -50 \end{bmatrix}, \quad H := \begin{bmatrix} 0 \\ 50 \end{bmatrix}.$$

In this case we have  $H'G = [0 \ 0 \ -2500]$  and Kalman's inequality does not hold. We can see from the Nyquist plot that the phase and gain margins are very small and there is little robustness with respect to unmodeled dynamics since a small perturbation in the process can lead to an encirclement of the point  $-1$ .  $\square$

## 9.6 MATLAB hints

**MATLAB<sup>®</sup> Hint 33** (`lqr`). The command `[K,S,E]=lqr(A,B,QQ,RR,NN)` computes the optimal state-feedback LQR controller for the process

$$\dot{x} = Ax + Bu$$

with criteria

$$J := \int_0^{\infty} x(t)'Qx(t) + u'(t)Ru(t) + 2x'(t)Nu(t)dt.$$

For the criterion in (9.2) one should select

$$QQ = G'G, \quad RR = H'H + \rho I, \quad NN = G'H$$

and for the criterion (9.3)

$$QQ = G'QG, \quad RR = H'QH + \rho R, \quad NN = G'QH.$$

(cf. Sidebar 45). This command returns the optimal state-feedback matrix  $K$ , the solution  $P$  to the corresponding Algebraic Riccati Equation, and the poles  $E$  of the closed-loop system.  $\square$

**MATLAB<sup>®</sup> Hint 34** (`sigma`). The command `sigma(sys)` draws the norm-Bode plot of the system `sys`. For scalar transfer functions this command plots the usual magnitude Bode plot but for vector transfer function it plots the norm of the transfer function versus the frequency.  $\square$

## 9.7 To probe further

**Note 45** (General LQR). The most general form for a quadratic criteria is

$$J := \int_0^{\infty} x(t)' \bar{Q}x(t) + u'(t) \bar{R}u(t) + 2x'(t) \bar{N}u(t) dt. \quad (9.8)$$

Since  $z = Gx + Hu$ , the criterion in (9.2) is a special form of (9.8) with

$$\bar{Q} = G'G, \quad \bar{R} = H'H + \rho I, \quad \bar{N} = G'H$$

and (9.3) is a special form of (9.8) with

$$\bar{Q} = G'QG, \quad \bar{R} = H'QH + \rho R, \quad \bar{N} = G'QH.$$

For this criteria, the optimal state-feedback LQR controller is still of the form

$$u = -\bar{K}x$$

but now  $\bar{K}$  is given by

$$\bar{K} = \bar{R}^{-1}(B'\bar{P} + \bar{N})$$

and  $\bar{P}$  is a solution to the following *Algebraic Riccati Equation (ARE)*

$$A'P + PA + \bar{Q} - (\bar{P}B + \bar{N})\bar{R}^{-1}(B'\bar{P} + \bar{N}') = 0. \quad \square$$

**Note 46.** A symmetric  $k \times k$  matrix  $M$  is *positive-definite* if

$$x'Mx > 0,$$

for every nonzero vector  $x \in \mathbb{R}^k$ . Positive-definite matrices are always nonsingular and their inverses are also positive-definite.

To test if a matrix is positive definite one can compute its eigenvalues. If they are all positive the matrix is positive-definite, otherwise it is not. □

**MATLAB® Hint 32.** `eig(A)` computes the eigenvalues of the matrix  $A$ . . . ▶ p. 81

**Note 50** (Multiplicative uncertainty). Since the Nyquist plot of  $G_0(j\omega)$  does not enter a circle of radius one around  $-1$ , we have that

$$|1 + G_0(j\omega)| \geq 1 \Rightarrow \left| \frac{1}{1 + G_0(j\omega)} \right| = \left| 1 - \frac{G_0(j\omega)}{1 + G_0(j\omega)} \right| \leq 1 \Rightarrow \left| \frac{G_0(j\omega)}{1 + G_0(j\omega)} \right| \leq 2. \quad \square$$

**Note 53** (Equation (9.7)). When  $z = [y \quad \gamma\dot{y}]'$ , we have that

$$z = \begin{bmatrix} y \\ \gamma\dot{y} \end{bmatrix} = \begin{bmatrix} Cx \\ \gamma CAx + \gamma CBu \end{bmatrix} \Rightarrow G = \begin{bmatrix} C \\ \gamma CA \end{bmatrix}, \quad H = \begin{bmatrix} 0 \\ \gamma CB \end{bmatrix}.$$

In this case,

$$P_z(s) = \begin{bmatrix} P_y(s) \\ \gamma s P_y(s) \end{bmatrix} = \begin{bmatrix} 1 \\ \gamma s \end{bmatrix} P_y(s),$$

where  $P_y(s) := C(sI - A)^{-1}B$ , and therefore

$$|G_0(j\omega)| \approx \frac{\sqrt{1 + \gamma^2 \omega^2} |P_y(j\omega)|}{\sqrt{H'H + \rho}} = \frac{|1 + j\gamma\omega| |P_y(j\omega)|}{\sqrt{H'H + \rho}}. \quad \square$$

## 9.8 Exercises

**9.1.** Verify using the diagram in Figure 11.1 that, for the single-input case ( $m = 1$ ), the closed-loop transfer function  $T_u(s)$  from the reference  $r$  to the process input  $u$  is given by

$$T_u(s) = \frac{1}{1 + G_0}(KF + N),$$

where  $G_0(s) = K(sI - A)^{-1}B$ , and the closed-loop transfer function  $T_z$  from the reference  $r$  to the controlled output  $z$  is given by

$$T_z(s) = \frac{1}{1 + G_0}P_z(KF + N),$$

where  $P_z(s) = G(sI - A)^{-1}B + H$ . □

**9.2.** Consider an inverted pendulum operating near the upright equilibrium position, with linearized model given by

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{g}{\ell} & -\frac{b}{m\ell^2} \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m\ell^2} \end{bmatrix} T$$

where  $T$  denotes an applied torque,  $\theta$  the pendulum's angle with a vertical axis pointing up, and  $\ell = 1\text{m}$ ,  $m = 1\text{Kg}$ ,  $b = .1\text{N/m/s}$ ,  $g = 9.8\text{m/s}^2$ .

1. Design a PD controller using LQR
2. Design a PID controller using LQR

*Hint: Consider an "augmented" process model with state  $\theta, \dot{\theta}$ ,  $z(t) = \int_0^t \theta(s)ds$ .* □

# Chapter 10

## LQG/LQR Output Feedback

### Contents

---

10.1 Output Feedback . . . . .	93
10.2 Full-order observers . . . . .	93
10.3 LQG estimation . . . . .	94
10.4 LQG/LQR output feedback . . . . .	95
10.5 Separation Principle . . . . .	96
10.6 Loop-gain recovery . . . . .	96
10.7 MATLAB hints . . . . .	98
10.8 Exercises . . . . .	98

---

### 10.1 Output Feedback

The state-feedback LQR formulation considered in Chapter 9.3 suffered from the drawback that the optimal control law

$$u(t) = -Kx(t) \tag{10.1}$$

required the whole state  $x$  of the process to be measurable. An possible approach to overcome this difficulty is to estimate the state of the process based solely on the measured output  $y$ , and use

$$u(t) = -K\hat{x}(t)$$

instead of (10.1), where  $\hat{x}(t)$  denotes an estimate of the process' state  $x(t)$ . In this chapter we consider the problem of constructing state estimates.

### 10.2 Full-order observers

Consider a process with state-space model

$$\dot{x} = Ax + Bu, \quad y = Cx, \tag{10.2}$$

where  $y$  denotes the measured output,  $u$  the control input. We assume that  $x$  cannot be measured and our goal to estimate its value based on  $y$ .

Suppose we construct the estimate  $\hat{x}$  by replicating the process dynamics as in

$$\dot{\hat{x}} = A\hat{x} + Bu. \tag{10.3}$$

To see if this would generate a good estimate for  $x$ , we can define the state estimation error  $e := x - \hat{x}$  and study its dynamics. From (10.2) and (10.3), we conclude that

$$\dot{e} = Ax - A\hat{x} = Ae.$$

This shows that when the matrix  $A$  is asymptotically stable the error  $e$  converges to zero for any input  $u$ , which is good news because it means that  $\hat{x}$  eventually converges to  $x$  as  $t \rightarrow \infty$ . However, when  $A$  is not stable  $e$  is unbounded and  $\hat{x}$  grow further and further apart from  $x$  as  $t \rightarrow \infty$ . To avoid this, one includes a correction term in (10.3):

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - \hat{y}), \quad \hat{y} = C\hat{x}, \quad (10.4)$$

where  $\hat{y}$  should be viewed as an estimate of  $y$  and  $L$  a given  $n \times k$  matrix. When  $\hat{x}$  is equal (or very close) to  $x$ , then  $\hat{y}$  will be equal (or very close) to  $y$  and the correction term  $L(y - \hat{y})$  plays no role. However, when  $\hat{x}$  grows away from  $x$ , this term will (hopefully!) correct the error. To see how this can be done, we re-write the estimation error dynamics now for (10.2) and (10.4):

$$\dot{e} = Ax - A\hat{x} - L(Cx - C\hat{x}) = (A - LC)e.$$

Now  $e$  converges to zero as long as  $A - LC$  is asymptotically stable. It turns out that, even when  $A$  is unstable, in general we will be able to select  $L$  so that  $A - LC$  is asymptotically stable. The system (10.4) can be re-write as

$$\dot{\hat{x}} = (A - LC)\hat{x} + Bu + Ly, \quad (10.5)$$

and is called a *full-order observer* for the process. Full-order observers have two inputs—the process' control input  $u$  and its measured output  $y$ —and a single output—the state estimate  $\hat{x}$ . Figure 10.1 shows how a full-order observer is connected to the process.

**Note 54.** “Full-order” refers to the fact that the size of its state  $\hat{x}$  is equal to the size of the process' state  $x$ .

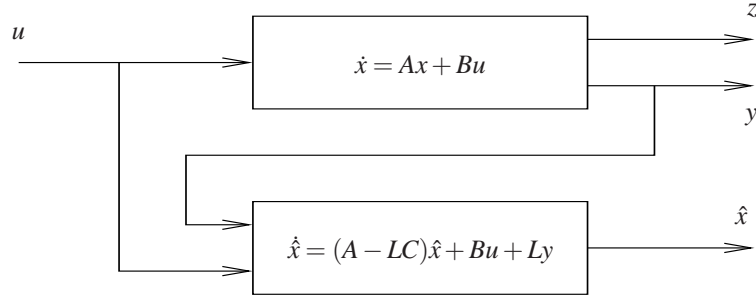


Figure 10.1. Full-order observer

### 10.3 LQG estimation

Any choice of  $L$  in (10.4) for which  $A - LC$  is asymptotically stable will make  $\hat{x}$  converge to  $x$ , as long as the process dynamics is given by (10.2). However, in general the output  $y$  is affected by measurement noise and the process dynamics are also affected by disturbance. In light of this, a more reasonable model for the process is

$$\dot{x} = Ax + Bu + \bar{B}d, \quad y = Cx + n, \quad (10.6)$$

where  $d$  denotes a disturbance and  $n$  measurement noise. In this we need to re-write the estimation error dynamics for (10.6) and (10.4), which leads to

$$\dot{e} = Ax + \bar{B}d - A\hat{x} - L(Cx + n - C\hat{x}) = (A - LC)e + \bar{B}d - Ln.$$

Because of  $n$  and  $d$ , the estimation error will generally not converge to zero, but we would still like it to remain small by appropriate choice of the matrix  $L$ . This motivates the so called *Linear-quadratic Gaussian (LQG) estimation problem*:

**Problem 10.1** (Optimal LQG). Find the matrix gain  $L$  that minimizes the asymptotic expected value of the estimation error:

$$J_{\text{LQG}} := \lim_{t \rightarrow \infty} \mathbb{E} [\|e(t)\|^2],$$

where  $d(t)$  and  $n(t)$  are zero-mean Gaussian noise processes (uncorrelated from each other) with power spectrum

$$S_d(\omega) = Q_N, \quad S_n(\omega) = R_N, \quad \forall \omega. \quad \square$$

*Solution to the optimal LQG Problem 10.1.* The optimal LQG estimator gain  $L$  is the  $n \times k$  matrix given by

$$L = PC'R_N^{-1}$$

and  $P$  is the unique positive-definite solution to the following Algebraic Riccati Equation (ARE)

$$AP + PA' + \bar{B}Q_N\bar{B}' - PC'R_N^{-1}CP = 0.$$

When one uses the optimal gain  $L$  in (10.5), this system is called the *Kalman-Bucy* filter.

A crucial property of this system is that  $A - LC$  is asymptotically stable as long as the following two conditions hold:

1. The system (10.6) is observable.
2. The system (10.6) is controllable when we ignore  $u$  and regard  $d$  as the sole input.

Different choices of  $Q_N$  and  $R_N$  result in different estimator gains  $L$ :

1. When  $R_N$  is *very small* (when compared to  $Q_N$ ), the measurement noise  $n$  is necessarily small so the optimal estimator interprets a large deviation of  $\hat{y}$  from  $y$  as an indication that the estimate  $\hat{x}$  is bad and needs to be correct. In practice, this lead to large matrices  $L$  and fast poles for  $A - LC$ .
2. When  $R_N$  is *very large*, the measurement noise  $n$  is large so the optimal estimator is much more conservative in reacting to deviations of  $\hat{y}$  from  $y$ . This generally leads to smaller matrices  $L$  and slow poles for  $A - LC$ .

We will return to the selection of  $Q_N$  and  $R_N$  in Section 10.6.

## 10.4 LQG/LQR output feedback

We now go back to the problem of designing an output-feedback controller for the process:

$$\dot{x} = Ax + Bu, \quad y = Cx, \quad z = Gx + Hu.$$

Suppose that we designed a state-feedback controller

$$u = -Kx \tag{10.7}$$

that solves an LQR problem and constructed an LQG state-estimator

$$\dot{\hat{x}} = (A - LC)\hat{x} + Bu + Ly.$$

We can obtain an output-feedback controller by using the estimated state  $\hat{x}$  in (10.7), instead of the true state  $x$ . This leads to the following output-feedback controller

$$\dot{\hat{x}} = (A - LC)\hat{x} + Bu + Ly = (A - LC - BK)\hat{x} + Ly, \quad u = -K\hat{x},$$

with *negative-feedback* transfer matrix given by

$$C(s) = K(sI - A + LC + BK)^{-1}L.$$

This is usually known as an *LQG/LQR output-feedback controller* and the resulting closed-loop is shown in Figure 10.2.

**Note 55.** A zero-mean white noise process  $n$  has an autocorrelation of the form

$$R_n(t_1, t_2) := \mathbb{E} [n(t_1)n'(t_2)] = Q_N \delta(t_1 - t_2).$$

Such process is *wide-sense stationary* in the sense that its mean is time-invariant and its auto-correlation  $R(t_1, t_2)$  only depends on the difference  $\tau := t_1 - t_2$ . Its power spectrum is frequency-independent and given by

$$S_n(\omega) := \int_{-\infty}^{\infty} R(\tau)e^{-j\omega\tau} d\tau = Q_N.$$

**MATLAB® Hint 35.** `kalman` computes the optimal LQG estimator gain  $L$ . . . ▶ p. 98

**Note 46.** A symmetric  $q \times q$  matrix  $M$  is *positive definite* if  $x'Mx > 0$ , for every nonzero vector  $x \in \mathbb{R}^q$ . . . ▶ p. 91

**MATLAB® Hint 36.** `reg(sys, K, L)` computes the LQG/LQR *positive* output-feedback controller for the process `sys` with regulator gain  $K$  and estimator gain  $L$ . . . ▶ p. 98

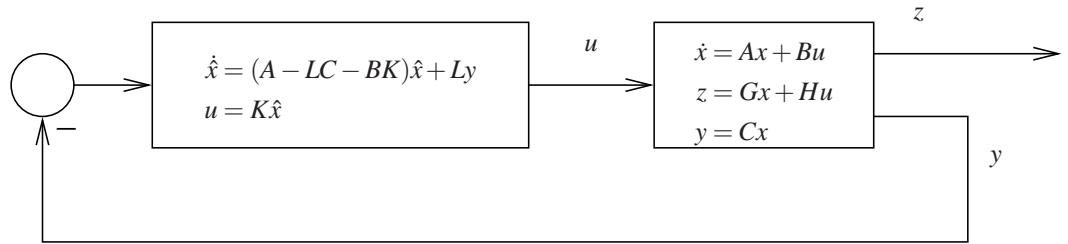


Figure 10.2. LQG/LQR output-feedback

## 10.5 Separation Principle

The first question to ask about an LQG/LQR controller is whether or not the closed-loop system will be stable. To answer this question we collect all the equations that defines the closed-loop system:

$$\dot{x} = Ax + Bu, \quad y = Cx, \quad (10.8)$$

$$\dot{\hat{x}} = (A - LC)\hat{x} + Bu + Ly, \quad u = -K\hat{x}. \quad (10.9)$$

To check the stability of this system it is more convenient to consider the dynamics of the estimation error  $e := x - \hat{x}$  instead of the the state estimate  $\hat{x}$ . To this effect we replace in the above equations  $\hat{x}$  by  $x - e$ , which yields:

$$\begin{aligned} \dot{x} &= Ax + Bu = (A - BK)x + BKe, & y &= Cx, \\ \dot{e} &= (A - LC)e, & u &= -K(x - e). \end{aligned}$$

This can be written in matrix notation as

$$\begin{bmatrix} \dot{\hat{x}} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} A - BK & BK \\ 0 & A - LC \end{bmatrix} \begin{bmatrix} x \\ e \end{bmatrix}, \quad y = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x \\ e \end{bmatrix}.$$

**Note 56.** Any eigenvalue of a block diagonal matrix must be an eigenvalue of one of the diagonal blocks.

**Separation Principle.** *The eigenvalues of the closed-loop system (10.8) are given by those of the state-feedback regulator dynamics  $A - BK$  together with those of state-estimator dynamics  $A - LC$ . In case these both matrices are asymptotically stable, then so is the closed-loop (10.8).*

## 10.6 Loop-gain recovery

We saw in Sections 9.4 and 9.5 that state-feedback LQR controllers have desirable robustness properties and that we can shape the open-loop gain by appropriate choice of the LQR weighting parameter  $\rho$  and the choice of the controlled output  $z$ . It turns out that we can, to some extent, recover the LQR open-loop gain for the LQG/LQR controller.

**Loop-gain recovery.** *Suppose that the process is single-input/single-output and has no zeros in the right half-plane. Selecting*

$$\bar{B} := B, \quad Q_N := 1, \quad R_N := \sigma, \quad \sigma > 0,$$

**Note 57.**  $\bar{B} = B$  corresponds to an input disturbance since the process becomes

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

*the open-loop gain for the output-feedback LQG/LQR controller converges to the open-loop gain for the state-feedback LQR state-feedback controller over a range of frequencies  $[0, \omega_{\max}]$  as we make  $\sigma \rightarrow 0$ , i.e.,*

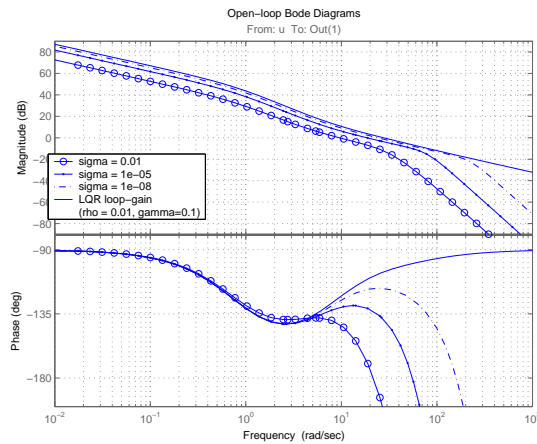
$$C(j\omega)P(j\omega) \xrightarrow{\sigma \rightarrow 0} K(j\omega I - 1)^{-1}B, \quad \forall \omega \in [0, \omega_{\max}]$$

*In general, the larger  $\omega_{\max}$  is, the smaller  $\sigma$  needs to be for the gains to match.*

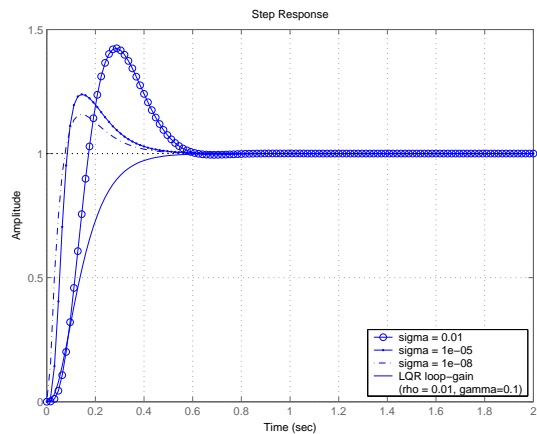
- Attention!** 1. To achieve loop-gain recovery we need to choose  $R_N := \sigma$ , even if this does not accurately describe the noise statistics. This means that the estimator may not be optimal for the actual noise.
2. One should not make  $\sigma$  smaller than necessary because we do not want to recover the (slow)  $-20\text{dB/decade}$  magnitude decrease at high frequencies. In practice we should make  $\sigma$  just small enough to get loop-recovery until just above or at cross-over. For larger values of  $\omega$ , the output-feedback controller may actually behave much better than the state-feedback one.
3. When the process has zeros in the right half-plane, loop-gain recovery will generally only work up to the frequencies of the nonminimum-phase zeros.
- When the zeros are in the left half-plane but close to the axis, the closed-loop will not be very robust with respect to uncertainty in the position of the zeros. This is because the controller will attempt to cancel these zeros. □

**Note 58.** We need loop-recovery up to the cross-over frequency to maintain the desired phase margin, otherwise the LQG/LQR controller may have a phase margin that is much smaller than that of the original LQR controller.

**Example 10.1** (Aircraft roll-dynamics). Figure 10.3(a) shows Bode plots of the open-loop gain for the state-feedback LQR state-feedback controller vs. the open-loop gain for several output-feedback LQG/LQR controller obtained for the aircraft roll-dynamics in Example 8.1. The LQR controller



(a) Open-loop gain



(b) Closed-loop step response

**Figure 10.3.** Bode plots and closed-loop step response for the open-loop gain of the LQR controllers in Examples 10.1, 11.2.

was designed using the controlled output  $z := [\theta \quad \gamma\dot{\theta}]'$ ,  $\gamma = .1$  and  $\rho = .01$ . For the LQG state-estimators we used  $\bar{B} = B$  and  $R_N = \sigma$  for several values of  $\sigma$ . We can see that, as  $\sigma$  decreases, the range of frequencies over which the open-loop gain of the output-feedback LQG/LQR controller

matches that of the state-feedback LQR state-feedback increases. Moreover, at high frequencies the output-feedback controllers exhibit much faster (and better!) decays of the gain's magnitude.  $\square$

## 10.7 MATLAB hints

**MATLAB<sup>®</sup> Hint 35** (`kalman`). The command `[est,L,P]=kalman(sys,QN,RN)` computes the optimal LQG estimator gain for the process

$$\dot{x} = Ax + Bu + BBd, \quad y = Cx + n,$$

where  $d(t)$  and  $n(t)$  are zero-mean Gaussian noise processes (uncorrelated from each other) with power spectrum

$$S_d(\omega) = QN, \quad S_n(\omega) = RN, \quad \forall \omega.$$

The system `sys` should be a state-space model defined by `sys=ss(A,[B BB],C,0)`. This command returns the optimal estimator gain `L`, the solution `P` to the corresponding Algebraic Riccati Equation, and a state-space model `est` for the estimator. The inputs to `est` are  $[u;y]$  and its outputs are  $[\hat{y};\hat{x}]$ .  $\square$

**MATLAB<sup>®</sup> Hint 36** (`reg`). The command `reg(sys,K,L)` computes a state-space model for a *positive* output-feedback LQG/LQR controller for the process with state-space model `sys` with regulator gain `K` and estimator gain `L`.  $\square$

## 10.8 Exercises

**10.1.** Consider an inverted pendulum on a cart operating near the upright equilibrium position, with linearized model given by

$$\begin{bmatrix} \dot{p} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -2.94 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 11.76 & 0 \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ .325 \\ 0 \\ -.3 \end{bmatrix} F$$

where  $F$  denotes a force applied to the cart,  $p$  the cart's horizontal position, and  $\theta$  the pendulum's angle with a vertical axis pointing up.

1. Design an LQG/LQR output-feedback controller that uses only the angle  $\theta$  and the position of the cart  $p$ .
2. Design an LQG/LQR output-feedback controller that uses the angle  $\theta$ , the angular velocity  $\dot{\theta}$ , the position of the cart  $p$ , and its derivative using LQG/LQR (full-state feedback).

Why use LQG when the state is accessible?  $\square$

# Chapter 11

## Set-point control

### Contents

11.1 Nonzero equilibrium state and input . . . . .	99
11.2 State-feedback . . . . .	100
11.3 Output-feedback . . . . .	101
11.4 To probe further . . . . .	101

### 11.1 Nonzero equilibrium state and input

Often one does not want to make  $z$  as small as possible, but instead make it converge as fast as possible to a given constant *set-point value*  $r$ . This can be achieved by making the state  $x$  and the input  $u$  of the process (9.1) converge to values  $x^*$  and  $u^*$  for which

$$Ax^* + Bu^* = 0, \quad r = Gx^* + Hu^*. \quad (11.1)$$

The right equation makes sure that  $z$  will be equal to  $r$  when  $x$  and  $u$  reach  $x^*$  and  $u^*$ , respectively. The left-equation makes sure that when these values are reached,  $\dot{x} = 0$  and therefore  $x$  will remain equal to  $x^*$ , i.e.,  $x^*$  is an *equilibrium state*.

Given the desired set-point  $r$  for  $x$ , *computing  $x^*$  and  $u^*$*  is straightforward because (11.1) is a system of linear equations and in general the solution to these equations is of the form

$$x^* = Fr, \quad u^* = Nr. \quad (11.2)$$

For example, when the number of inputs to the process  $m$  is equal to the number of controlled outputs  $\ell$ , we have

$$\begin{bmatrix} A & B \\ G & H \end{bmatrix} \begin{bmatrix} x^* \\ u^* \end{bmatrix} = \begin{bmatrix} 0 \\ r \end{bmatrix} \Leftrightarrow \begin{bmatrix} x^* \\ u^* \end{bmatrix} = \begin{bmatrix} A & B \\ G & H \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ r \end{bmatrix}, \quad (11.3)$$

and  $F$  is an  $n \times \ell$  matrix given by the top  $n$  rows and right-most  $\ell$  columns of  $\begin{bmatrix} A & B \\ G & H \end{bmatrix}^{-1}$  and  $N$  is an  $m \times \ell$  matrix given by the bottom  $m$  rows and right-most  $\ell$  columns of  $\begin{bmatrix} A & B \\ G & H \end{bmatrix}^{-1}$ .

**Attention!** When the number of process inputs  $m$  is larger than the number of controlled outputs  $\ell$  we have an *over-actuated system* and the system of equations (11.1) will generally have multiple solutions. One of them is

$$\begin{bmatrix} x^* \\ u^* \end{bmatrix} = \begin{bmatrix} A & B \\ G & H \end{bmatrix}' \left( \begin{bmatrix} A & B \\ G & H \end{bmatrix} \begin{bmatrix} A & B \\ G & H \end{bmatrix}' \right)^{-1} \begin{bmatrix} 0 \\ r \end{bmatrix}.$$

**Note 59.** We will see shortly how to accomplish this.

**Note 60.** When the process transfer-function has an integrator, one generally gets  $u^* = 0$ .

**Note 61.** The matrix  $\begin{bmatrix} A & B \\ G & H \end{bmatrix}$  is invertible unless the process' transfer-function from  $u$  to  $z$  has a zero at the origin. This derivative effect will always make  $z$  converge to zero when the input converges to a constant.

In this case, we can still express  $x^*$  and  $u^*$  as in (11.2).

When the number of process inputs  $m$  is smaller than the number of controlled outputs  $\ell$  we have an *under-actuated system* and the system of equations (11.1) may not have a solution. In fact, a solution will only exist for some specific references  $r$ . However, when it does exist, the solution can still express  $x^*$  and  $u^*$  as in (11.2).  $\square$

## 11.2 State-feedback

When one wants  $z$  to converge to a given *set-point value*  $r$ , the state-feedback controller should be

$$u = -K(x - x^*) + u^* = -Kx + (KF + N)r, \quad (11.4)$$

where  $x^*$  and  $u^*$  are given by (11.2) and  $K$  is the gain of the optimal regulation problem. The corresponding control architecture is shown in Figure 11.1. The state-space model for the closed-loop system is given by

$$\begin{aligned} \dot{x} &= Ax + Bu = (A - BK)x + B(KF + N)r \\ z &= Gx + Hu = (G - HK)x + H(KF + N)r. \end{aligned}$$

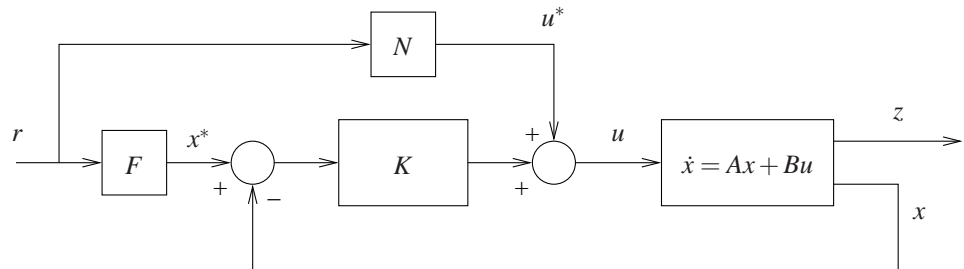


Figure 11.1. Linear quadratic set-point control with state feedback

**Example 11.1** (Aircraft roll-dynamics). Figure 11.2 shows step responses for the state-feedback LQR controllers in Example 9.1, whose Bode plots for the open-loop gain are shown in Figure 9.6. Figure 11.2(a) shows that smaller values of  $\rho$  lead to faster responses and Figure 11.2(b) shows that

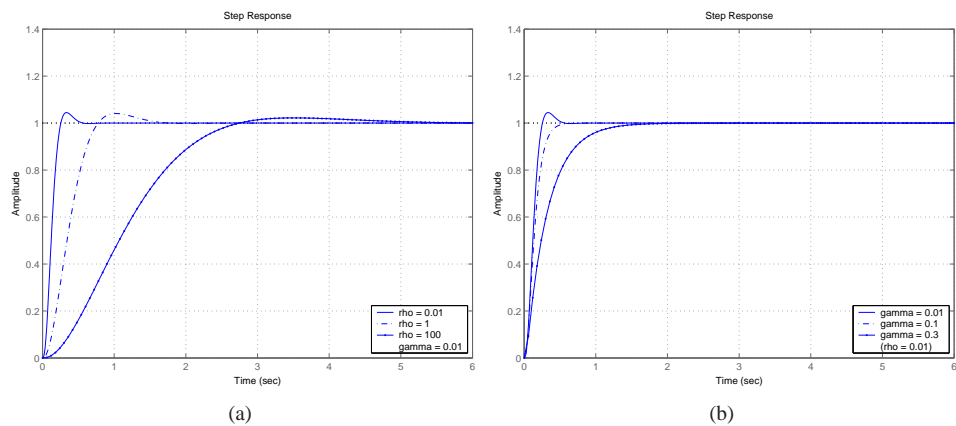


Figure 11.2. Step responses for the closed-loop LQR controllers in Example 11.1

larger values for  $\gamma$  lead to smaller overshoots (but slower responses).  $\square$

### 11.3 Output-feedback

When one wants  $z$  to converge to a given *set-point value*  $r$ , the output-feedback LQG/LQR controller should be

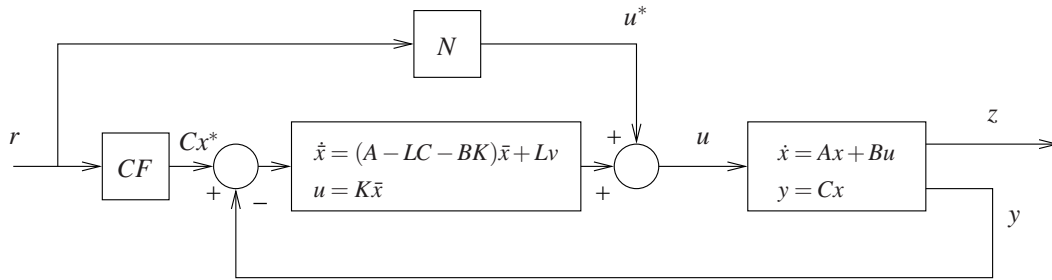
$$\dot{\bar{x}} = (A - LC - BK)\bar{x} + L(Cx^* - y), \quad u = K\bar{x} + u^*, \quad (11.5)$$

where  $x^*$  and  $u^*$  are given by (11.2). The corresponding control architecture is shown in Figure 11.3. The state-space model for the closed-loop system is given by

$$\begin{aligned} \begin{bmatrix} \dot{\bar{x}} \\ \dot{x} \end{bmatrix} &= \begin{bmatrix} Ax + B(K\bar{x} + u^*) \\ (A - LC - BK)\bar{x} + L(Cx^* - Cx) \end{bmatrix} = \begin{bmatrix} A & BK \\ -LC & A - LC - BK \end{bmatrix} \begin{bmatrix} x \\ \bar{x} \end{bmatrix} + \begin{bmatrix} BN \\ LCF \end{bmatrix} r \\ z &= Gx + H(K\bar{x} + u^*) = \begin{bmatrix} G & HK \end{bmatrix} \begin{bmatrix} x \\ \bar{x} \end{bmatrix} + HNr \end{aligned}$$

**Note 63.** Why?... ► p. 102

**Note 64.** When  $z = y$ , we have  $G = C, H = 0$  and in this case  $Cx^* = r$ . This corresponds to  $CF = 1$  in Figure 11.3. When the process has an integrator we get  $N = 0$  and obtain the usual unity-feedback configuration.



**Figure 11.3.** LQG/LQR set-point control

**Example 11.2** (Aircraft roll-dynamics). Figure 10.3(b) shows step responses for the output-feedback LQG/LQR controllers in Example 10.1, whose Bode plots for the open-loop gain are shown in Figure 10.3(a). We can see that smaller values of  $\sigma$  lead to a smaller overshoot mostly due to a larger gain margin. □

### 11.4 To probe further

**Note 62** (Set-point control with state-feedback). To understand why (11.4) works, suppose we define

$$\tilde{z} = z - r, \quad \tilde{x} = x - x^*, \quad \tilde{u} = u - u^*.$$

Then

$$\begin{aligned} \dot{\tilde{x}} &= Ax + Bu = A(x - x^*) + B(u - u^*) + Ax^* + Bu^* \\ \tilde{z} &= Gx + Hu - r = G(x - x^*) + H(u - u^*) + Gx^* + Hu^* - r \end{aligned}$$

and we conclude that

$$\dot{\tilde{x}} = A\tilde{x} + B\tilde{u}, \quad \tilde{z} = G\tilde{x} + H\tilde{u}. \quad (11.6)$$

By selecting the control signal in (11.4), we are setting

$$\tilde{u} = u - u^* = -K(x - x^*) = -K\tilde{x},$$

which is the optimal state-feedback LQR controller that minimizes

$$J_{LQR} := \int_0^\infty (\tilde{z}(t)' Q \tilde{z}(t)) + \rho (\tilde{u}'(t) R \tilde{u}(t)) dt,$$

This controller makes the system (11.6) asymptotically stable and therefore  $\tilde{x}, \tilde{u}, \tilde{z}$  all converge to zero as  $t \rightarrow \infty$ , which means that  $z$  converges to  $r$ . □

**Note 63** (Set-point control with output-feedback). To understand why (11.5) works suppose we define

$$\tilde{z} = z - r, \quad \tilde{x} = x - x^* + \bar{x}.$$

Then

$$\dot{\tilde{x}} = (A - LC)\tilde{x} \quad (11.7)$$

$$\dot{\bar{x}} = (A - BK)\bar{x} - LC\bar{x} \quad (11.8)$$

$$\tilde{z} = G(\tilde{x} - \bar{x}) + HK\bar{x} - r. \quad (11.9)$$

1. Since  $A - LC$  is asymptotically stable, we conclude from (11.7) that  $\tilde{x} \rightarrow 0$  as  $t \rightarrow \infty$ .

In practice, we can view the state  $\bar{x}$  of the controller as an estimate of  $x^* - x$ .

2. Since  $A - BK$  is asymptotically stable and  $\tilde{x} \rightarrow 0$  as  $t \rightarrow \infty$ , we conclude from (11.8) that  $\bar{x} \rightarrow 0$  as  $t \rightarrow \infty$ .

3. Since  $\tilde{x} \rightarrow 0$  and  $\bar{x} \rightarrow 0$  as  $t \rightarrow \infty$ , we conclude from (11.9) that  $z \rightarrow r$  as  $t \rightarrow \infty$ . □

**11.1.** Verify equations (11.7), (11.8), and (11.9). □

**Part IV**

**Nonlinear Control**



# Introduction to nonlinear control

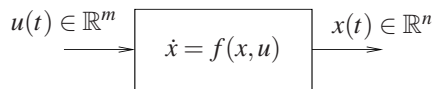
In this section we consider the control of nonlinear systems such as the one shown in Figure 11.4. Our goal is to construct a state-feedback control law of the form

$$u = k(x)$$

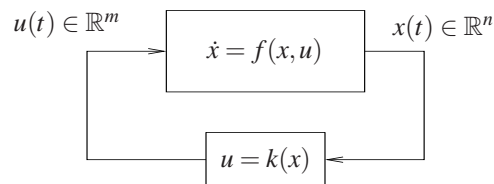
that results in adequate performance for the closed-loop system

$$\dot{x} = f(x, k(x)).$$

Typically, at least we want  $x$  to be bounded and converge to some desired reference value  $r$ .



**Figure 11.4.** Nonlinear process with  $m$  inputs



**Figure 11.5.** Closed-loop nonlinear system with state-feedback

**Note 65.** This control-law implicitly assumes that the whole state  $x$  can be measured.

## Pre-requisites

1. Basic knowledge of nonlinear ordinary differential equations.
2. Basic knowledge of continuous-time linear controller design.
3. Familiarity with basic vector and matrix operations.



# Chapter 12

## Feedback linearization controllers

### Contents

12.1 Feedback linearization . . . . .	107
12.2 Generalized model for mechanical systems . . . . .	108
12.3 Feedback linearization of mechanical systems . . . . .	110
12.4 Exercises . . . . .	111

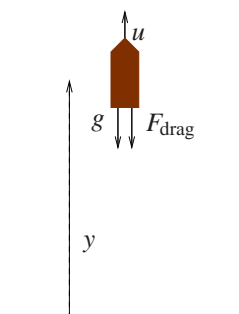
### 12.1 Feedback linearization

In feedback linearization control design, we decompose the control signal  $u$  into two components with distinct functions:

$$u = u_{nl} + v,$$

where

1.  $u_{nl}$  is used to “cancel” the process’ nonlinearities, and
2.  $v$  is used to control the resulting linear system.



From Newton’s law:

$$\begin{aligned} m\ddot{y} &= F_{\text{drag}} - mg + u \\ &= -\frac{1}{2}c\rho A\dot{y}|\dot{y}| - mg + u, \end{aligned}$$

where  $m$  is the vehicle’s mass,  $g$  gravity’s acceleration,  $F_{\text{drag}} = -\frac{1}{2}c\rho A\dot{y}|\dot{y}|$  the drag force, and  $u$  an applied force.

**Figure 12.1.** Dynamics of a vehicle moving vertically in the atmosphere

**Note 66.** For large objects moving through air, the air resistance is approximately proportional to the square of the velocity, with a drag force given by

$$F_{\text{drag}} = -\frac{1}{2}c\rho A\dot{y}|\dot{y}|$$

where  $\rho$  is the air density,  $A$  the cross-sectional area, and  $c$  the drag coefficient, which is 0.5 for a spherical object and can reach 2 for irregularly shaped objects [8].  $F_{\text{drag}}$  always points in the opposite direction of  $\dot{y}$ .

To understand how this is done, consider the vehicle shown in Figure 12.1 moving vertically in the atmosphere. By choosing

$$u = u_{\text{nl}}(\dot{y}) + v, \quad u_{\text{nl}}(\dot{y}) := \frac{1}{2}c\rho A\dot{y}|\dot{y}| + mg,$$

we obtain

$$m\ddot{y} = -\frac{1}{2}c\rho A\dot{y}|\dot{y}| - mg + (u_{\text{nl}}(\dot{y}) + v) = v \Rightarrow \ddot{y} = \frac{1}{m}v.$$

In practice, we transformed the original nonlinear process into a (linear) double integrator with transfer function from  $v$  to  $y$  given by

$$T(s) := \frac{1}{ms^2}.$$

We can now use linear methods to find a controller for  $v$  that stabilizes the closed-loop. E.g., a PD controller of the form

$$v = K_P e + K_D \dot{e}, \quad e := r - y.$$

Figure 12.2 shows a diagram of the overall closed-loop system. From an “input-output” perspective, the system in the dashed block behaves as a linear system with transfer function  $T(s) := \frac{1}{ms^2}$ .

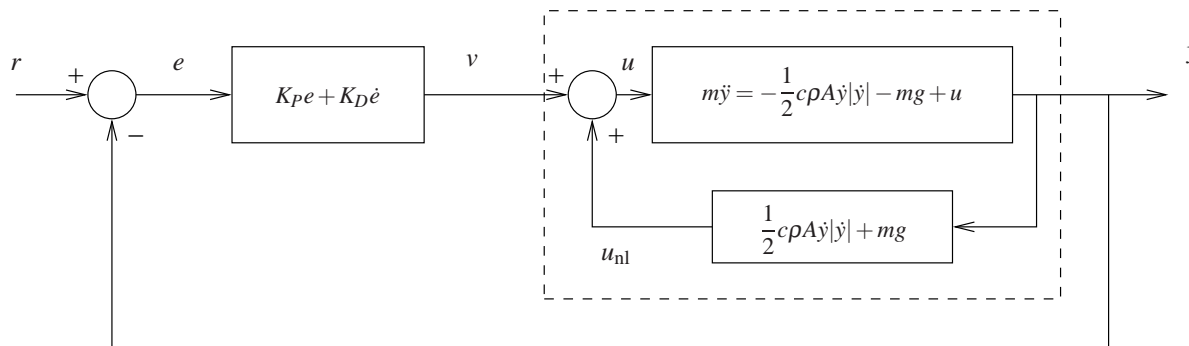


Figure 12.2. Feedback linearization controller.

## 12.2 Generalized model for mechanical systems

The equations of motion of many mechanical systems can be written in the following general form

$$M(q)\ddot{q} + B(q, \dot{q})\dot{q} + G(q) = F, \quad (12.1)$$

where

- $q \in \mathbb{R}^k$  is a  $k$ -vector with linear and/or angular positions called the *vector of generalized coordinates*;
- $F \in \mathbb{R}^k$  is a  $k$ -vector with applied forces and/or torques called the *vector of generalized forces*;
- $M(q)$  is a  $k \times k$  non-singular symmetric positive-definite matrix called the *mass matrix*;
- $B(q, \dot{q})$  is a  $k \times k$  matrix sometimes called the *centrifugal/Coriolis/friction matrix*, for systems with no friction we generally have

$$\dot{q}'B(q, \dot{q})\dot{q} = 0, \quad \forall \dot{q} \in \mathbb{R}^k$$

whereas for systems with friction

$$\dot{q}'B(q, \dot{q})\dot{q} \geq 0, \quad \forall \dot{q} \in \mathbb{R}^k,$$

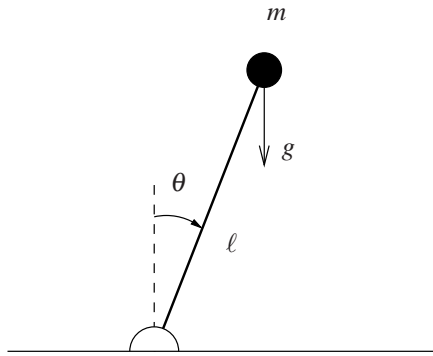
with equality only when  $\dot{q} = 0$ ;

- $G(q)$  is a  $k$ -vector sometimes called the *vector of conservative forces*.

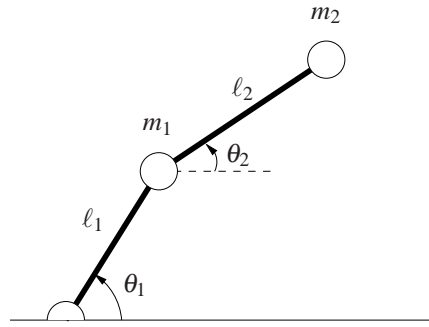
**Note 67.** These include robots, mobile robots, airplanes, helicopters, underwater vehicles, hovercraft, etc.

**Note 46.** A symmetric  $q \times q$  matrix  $M$  is *positive definite* if  $x'Mx > 0$ , for every nonzero vector  $x \in \mathbb{R}^q$ . . . ► p. 91

**Examples**



**Figure 12.3.** Inverted pendulum



**Figure 12.4.** Two-link robot manipulator

**Example 12.1** (Inverted pendulum). The dynamics of the inverted pendulum shown in Figure 12.3 are given by

$$m\ell^2\ddot{\theta} = mg\ell \sin \theta - b\dot{\theta} + T,$$

where  $T$  denotes a torque applied at the base, and  $g$  is the gravity’s acceleration. This equation can be identified with (12.1), provided that we define

$$q := \theta, \quad F := T, \quad M(q) := m\ell^2, \quad B(q) := b, \quad G(q) := -mg\ell \sin \theta,$$

When the pendulum is attached to a cart and the input is the cart’s acceleration  $\ddot{z}$ , we have  $T = -m\ell\ddot{z}\cos \theta$  but this makes the problem more difficult, especially around  $\theta = \pm\pi/2$ .  $\square$

**Example 12.2** (Robot arm). The dynamics of the robot-arm with two revolution joints shown in Figure 12.4 can be written as in (12.1), provided that we define

$$q := \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}, \quad F := \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}.$$

where  $\tau_1$  and  $\tau_2$  denote torques applied at the joints. For this system

$$\begin{aligned} M(q) &:= \begin{bmatrix} \ell_2^2 m_2^2 + 2\ell_1 \ell_2 \cos \theta_2 + \ell_1^2 (m_1 + m_2) & \ell_2^2 + \ell_1 \ell_2 m_2 \cos \theta_2 \\ \ell_2^2 m_2 + \ell_1 \ell_2 m_2 \cos \theta_2 & \ell_2^2 m_2 \end{bmatrix} \\ B(q, \dot{q}) &:= \begin{bmatrix} -2m_2 \ell_1 \ell_2 \dot{\theta}_2 \sin \theta_2 & -m_2 \ell_1 \ell_2 \dot{\theta}_2 \sin \theta_2 \\ m_2 \ell_1 \ell_2 \dot{\theta}_1 \sin \theta_2 & 0 \end{bmatrix} \\ G(q) &:= g \begin{bmatrix} m_2 \ell_2 \cos(\theta_1 - \theta_2) + (m_1 + m_2) \ell_1 \cos \theta_1 \\ m_2 \ell_2 \cos(\theta_1 - \theta_2) \end{bmatrix}. \end{aligned}$$

where  $g$  is the gravity’s acceleration [2, p. 202–205].  $\square$

**Example 12.3** (Hovercraft). The dynamics of the hovercraft shown in Figure 12.5 can be written as in (12.1), provided that we define

$$q := \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \quad F := \begin{bmatrix} (F_s + F_p) \cos \theta - F_\ell \sin \theta \\ (F_s + F_p) \sin \theta + F_\ell \cos \theta \\ \ell(F_s - F_p) \end{bmatrix},$$

where  $F_s$ ,  $F_p$ , and  $F_\ell$  denote the starboard, portboard, and lateral fan forces. The vehicle in the photograph does not have a lateral fan, which means that  $F_\ell = 0$ . It is therefore called *underactuated*

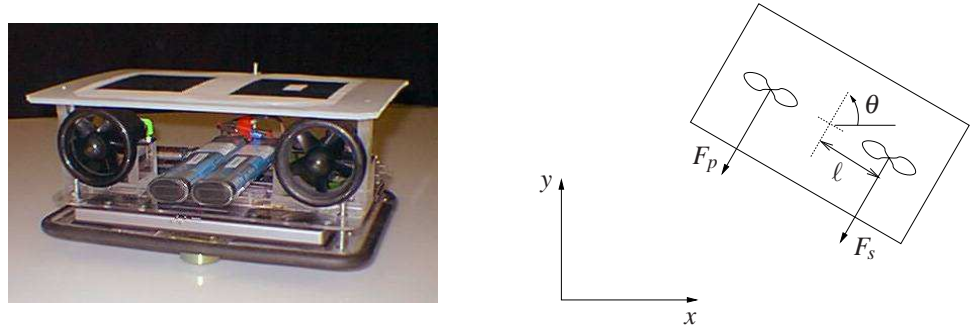


Figure 12.5. Hovercraft

because the number of controls ( $F_s$  and  $F_p$ ) is smaller than the number of degrees of freedom ( $x$ ,  $y$ , and  $\theta$ ). For this system

$$M(q) := \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix}, \quad B(q) := \begin{bmatrix} d_v & 0 & 0 \\ 0 & d_v & 0 \\ 0 & 0 & d_\omega \end{bmatrix},$$

where  $m = 5.5$  kg is the mass of the vehicle,  $J = 0.047$  kg m<sup>2</sup> its rotational inertia,  $d_v = 4.5$  the coefficient of linear friction,  $d_\omega = .41$  the coefficient of rotational friction, and  $\ell = 0.123$  m the moment arm of the forces. In these equations, the geometry and mass centers of the vehicle are assumed to coincide [3].  $\square$

### 12.3 Feedback linearization of mechanical systems

A mechanical system is called *fully actuated* when one has control over the whole vector or generalized forces. For such systems we regard  $u := F$  as the control input and we can use feedback linearization to design nonlinear controllers. In particular, choosing

$$F = u = u_{\text{nl}}(q, \dot{q}) + M(q)v, \quad u_{\text{nl}}(q, \dot{q}) := B(q, \dot{q})\dot{q} + G(q),$$

we obtain

$$M(q)\ddot{q} + B(q, \dot{q})\dot{q} + G(q) = B(q, \dot{q})\dot{q} + G(q) + M(q)v \Leftrightarrow \ddot{q} = v.$$

Expanding the  $k$ -vectors  $\ddot{q}$  and  $v$  in their components, we obtain

$$\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \vdots \\ \ddot{q}_k \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix}$$

and therefore we can select each  $v_i$  as if we were designing a controller for a double integrator

$$\ddot{q}_i = v_i.$$

**Attention!** Measurement noise can lead to problems in feedback linearization controllers. When the measurements of  $q$  and  $\dot{q}$  are affected by noise, we have

$$M(q)\ddot{q} + B(q, \dot{q})\dot{q} + G(q) = u_{\text{nl}}(q + n, \dot{q} + w) + M(q + n)v,$$

where  $n$  is measurement noise in the  $q$  sensors and  $w$  the measurement noise in the  $\dot{q}$  sensors. In this case

$$M(q)\ddot{q} + B(q, \dot{q})\dot{q} + G(q) = B(q + n, \dot{q} + w)(\dot{q} + w) + G(q + n) + M(q + n)v \quad (12.2)$$

and (with some work) one can show that

$$\ddot{q} = (I + \Delta)v + d, \quad (12.3)$$

where

$$\begin{aligned} \Delta &:= M(q)^{-1}(M(q+n) - M(q)), \\ d &:= M(q)^{-1} \left( (B(q+n, \dot{q}+w) - B(q, \dot{q}))\dot{q} + B(q+n, \dot{q}+w)w + G(q+n) - G(q) \right). \end{aligned}$$

Since  $\Delta$  and  $d$  can be very large, with feedback linearization controllers it is particularly important to make sure that the controller selected for  $v$  is robust with respect to multiplicative uncertainty and good at rejecting disturbances.

## 12.4 Exercises

**12.1.** Verify that (12.3) is indeed equivalent to (12.2), by solving the latter equation with respect to  $\ddot{q}$ . □

**12.2.** Design a feedback linearization controllers to drive the inverted pendulum in Example 12.1 to the up-right position. Use the following values for the parameters:  $\ell = 1$  m,  $m = 1$  kg,  $b = .1$  N m<sup>-1</sup>s<sup>-1</sup>, and  $g = 9.8$  m s<sup>-2</sup>. Verify the performance of your system in the presence of measurement noise using Simulink. □



# Chapter 13

## Lyapunov stability

### Contents

---

13.1 Lyapunov stability . . . . .	113
13.2 Lyapunov Stability Theorem . . . . .	115
13.3 Exercise . . . . .	116
13.4 LaSalle's Invariance Principle . . . . .	116
13.5 Liénard equation and generalizations . . . . .	117
13.6 To probe further . . . . .	119
13.7 Exercises . . . . .	119

---

### 13.1 Lyapunov stability

Although feedback linearization provides a simple method to design controllers for nonlinear systems. The resulting controllers are sometimes very sensitive to measurement noise because it prevents a perfect cancellation of the nonlinearities. It turns out that such cancellation is not always needed.

Consider again the vehicle shown in Figure 12.1 and suppose that we simply want to control its velocity  $\dot{y}$ . For simplicity, we assume that the applied force  $u$  is constant and larger than  $mg$  (i.e., there is an excess upward force) and that the units were chosen so that all constant coefficient are numerically equal to 1:

$$\ddot{y} = -\dot{y}|\dot{y}| + 1.$$

Since

$$-\dot{y}|\dot{y}| + 1 = 0 \quad \Leftrightarrow \quad \dot{y} = 1,$$

we conclude that 1 is the only *equilibrium-point* for the system, i.e.,

$$\dot{y}(t) = 1, \quad \forall t \geq 0,$$

is the only possible *constant trajectory* for the system. A question that may then arise is:

*Will  $\dot{y}(t)$  converge to 1 as  $t \rightarrow \infty$ , when  $\dot{y}(0) \neq 1$ ?*

To study this question we will consider an “error” system that looks at the difference from  $\dot{y}$  to the equilibrium-point 1. Defining  $x := \dot{y} - 1$ , we conclude that

$$\dot{x} = -(x+1)|x+1| + 1. \tag{13.1}$$

and the previous question could be reformulated as

Will  $x(t)$  converge to 0 as  $t \rightarrow \infty$ , when  $x(0) \neq 0$ ?

To address this question, suppose that we define

$$V(x) = x^2$$

and compute  $V$ 's derivative with respect to time using the chain rule:

$$\dot{V} = \frac{dV}{dt} = \frac{\partial V}{\partial x} \dot{x} = 2x(-(x+1)|x+1|+1) = \begin{cases} -2x^2(x+2) & x \geq -1 \\ 2x(x^2+2x+2) & x < -1. \end{cases} \quad (13.2)$$

Figure 13.1 shows the right-hand-side of the above equation as a function of  $x$ . Three conclusions

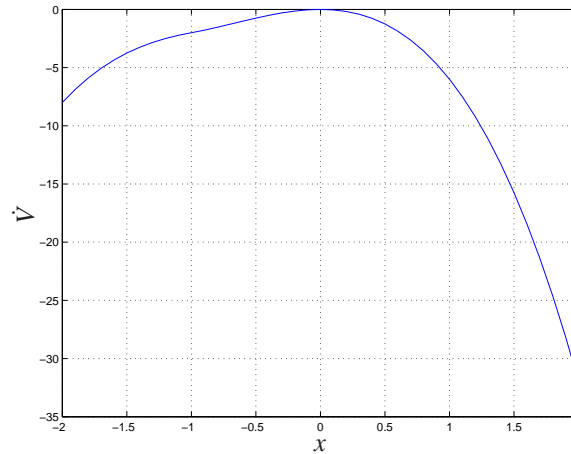


Figure 13.1.  $\dot{V}$  vs.  $x$

can be drawn from here:

**Boundedness** For every initial condition  $x(0)$ ,  $|x(t)|$  will remain bounded for all  $t \geq 0$ , i.e., it will not blow up. This is because  $V(x(t)) = x^2(t)$  cannot increase.

**Stability** If we start  $x(0)$  very close to zero, then  $x(t)$  will remain very close to zero for all times  $t \geq 0$ . This is because  $V(x(t)) = x^2(t)$  will always be smaller or equal than  $V(x(0))$ .

**Convergence**  $x(t) \rightarrow 0$  as  $t \rightarrow \infty$ . This is because the right-hand side of (13.2) is only equal to zero when  $x = 0$  and therefore  $\dot{V}$  will be strictly negative for any nonzero  $x$ .

This finally provides an answer to the previous questions: *Yes!*  $x \rightarrow 0$  as  $t \rightarrow \infty$ .

When a system exhibits these three properties we say that *the origin is globally asymptotically stable*. This notion of stability for nonlinear systems was originally proposed by Aleksandr Lyapunov and now carries his name.

**Definition 13.1** (Lyapunov stability). Given a system

$$\dot{x} = f(x), \quad t \geq 0, x \in \mathbb{R}^n, \quad (13.3)$$

we say that:

(i) the *trajectories are globally bounded* if for every initial condition  $x(0)$ , there exists a scalar  $\alpha(x(0)) \geq 0$  such that

$$x(t) \leq \alpha(x(0)), \quad \forall t \geq 0;$$

(ii) the *origin is globally stable* if the trajectories are globally bounded and  $\alpha(x(0)) \rightarrow 0$  as  $x(0) \rightarrow 0$ ;

(iii) the origin is globally asymptotically stable if it is globally stable and in addition  $x(t) \rightarrow 0$  as  $t \rightarrow \infty$ .

**Attention!** The requirement (ii) is rather subtle but very important. In essence, it says that if we choose the initial condition  $x(0)$  very close to the origin, then the upper bound  $\alpha(x(0))$  will also be very small and the whole trajectory stays very close to the origin.  $\square$

### 13.2 Lyapunov Stability Theorem

The technique used before to conclude that the origin is globally asymptotically stable for the system (13.1) is a special case of a more general method proposed by Aleksandr M. Lyapunov. His may contribution to nonlinear control was the realization that *an appropriate choice of the function  $V(x)$  could make proving stability very simple*. In the example above (and in fact in most one-dimensional systems)  $V(x) = x^2$  works well, but for higher dimensional systems the choice of  $V(x)$  is critical.

**Lyapunov Stability Theorem.** Suppose that there exists a scalar-valued function  $V : \mathbb{R}^n \rightarrow \mathbb{R}$  with the following properties:

- (i)  $V(x)$  is differentiable;
- (ii)  $V(x)$  is positive definite, which means that  $V(0) = 0$  and

$$V(x) > 0, \quad \forall x \neq 0.$$

- (iii)  $V(x)$  is radially unbounded, which means that  $V(x) \rightarrow \infty$  whenever  $x \rightarrow \infty$ ;
- (iv)  $\nabla_x V(x) \cdot f(x)$  is negative definite, which means that  $\nabla_x V(0) \cdot f(0) = 0$  and

$$\nabla_x V(x) \cdot f(x) < 0, \quad \forall x \neq 0.$$

Then the origin is globally asymptotically stable for the system (13.3). The function  $V(x)$  is called a Lyapunov function.  $\square$

The function  $V(x) = x^2$  considered before satisfies all the conditions of the Lyapunov Stability Theorem for the system (13.1) so our conclusion that the origin was globally asymptotically stable is consistent with this theorem.

**Note 68** (Lyapunov Stability Theorem). When the variable  $x$  is a scalar,

$$\dot{V} = \frac{dV}{dt} = \frac{\partial V}{\partial x} \dot{x} = \frac{\partial V}{\partial x} f(x),$$

but when  $x$  and  $f$  are  $n$ -vectors, i.e.,

$$\dot{x} = f(x) \iff \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix},$$

we have

$$\dot{V} = \frac{dV}{dt} = \sum_{i=1}^n \frac{\partial V}{\partial x_i} \dot{x}_i = \frac{\partial V}{\partial x_i} f_i(x) = \nabla_x V(x) \cdot f(x).$$

Condition (iv) thus requires  $V(x)$  to strictly decrease as long  $x \neq 0$ . Because of condition (ii),  $x \neq 0$  is equivalent to  $V(x) > 0$  so  $V(x)$  must decrease all the way to zero. Since  $V(x)$  is only zero at zero, this necessarily implies that  $x \rightarrow 0$  as  $t \rightarrow \infty$ .  $\square$

**Notation 6.** Given a scalar function of  $n$  variables  $f(x_1, \dots, x_m)$ ,  $\nabla_x f$  denotes the gradient of  $f$ , i.e.,

$$\nabla_x f(x_1, x_2, \dots, x_m) = \left[ \frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \dots \quad \frac{\partial f}{\partial x_m} \right].$$

**Note 68.** Why? ► p. 115

**Example 13.1.** Consider the following system

$$\ddot{y} + (1 + |\dot{y}|)(y + \dot{y}) = 0.$$

Defining  $x = [x_1 \ x_2]'$  :=  $[y \ \dot{y}]'$ , this system can be written as  $\dot{x} = f(x)$  as follows:

$$\begin{aligned}\dot{x}_1 &= x_2, \\ \dot{x}_2 &= -(1 + |x_2|)(x_1 + x_2).\end{aligned}$$

Suppose we want to show that the origin is globally asymptotically stable. A “candidate” Lyapunov function for this system is

$$V(x) := x_1^2 + (x_1 + x_2)^2.$$

This function satisfies the requirements (i)–(iii). Moreover,

$$\begin{aligned}\nabla_x V(x) \cdot f(x) &= 2x_1\dot{x}_1 + 2(x_1 + x_2)(\dot{x}_1 + \dot{x}_2) \\ &= 2x_1x_2 + 2(x_1 + x_2)(x_2 - (1 + |x_2|)(x_1 + x_2)) \\ &= 2x_1(-x_1 + x_1 + x_2) + 2(x_1 + x_2)(x_2 - (1 + |x_2|)(x_1 + x_2)) \\ &= -2x_1^2 + 2(x_1 + x_2)(x_1 + x_2 - (1 + |x_2|)(x_1 + x_2)) \\ &= -2x_1^2 - 2|x_2|(x_1 + x_2)^2 \leq 0.\end{aligned}$$

Since we only have equality to zero when  $x_1 = x_2 = 0$ , we conclude that (iv) also holds and therefore  $V(x)$  is indeed a Lyapunov function and the origin is globally asymptotically stable.  $\square$

### 13.3 Exercise

**13.1.** The average number of data-packets sent per second by any program that uses the TCP protocol (e.g., ftp) evolves according to an equation of the form:

$$\dot{r} = \frac{1}{RTT^2} - \frac{2}{3}p_{\text{drop}}r(r + d),$$

where  $r$  denotes the average sending rate,  $RTT$  denotes the round-trip time for one packet in seconds,  $p_{\text{drop}}$  the probability that a particular packet will be dropped inside the network, and  $d$  the average sending rate of other data-flows using the same connection. All rates are measured in packets per second. Typically one packet contains 1500bytes.

1. Find the equilibrium-point  $r_{\text{eq}}$  for the sending rate.

*Hint: When  $d = 0$ , your expression should simplify to*

$$r_{\text{eq}} = \frac{\sqrt{3/2}}{RTT\sqrt{p_{\text{drop}}}}.$$

*This is known as the “TCP-friendly” equation.*

2. Verify that the origin is globally asymptotically stable for the system with state  $x := r - r_{\text{eq}}$ .  $\square$

### 13.4 LaSalle’s Invariance Principle

Consider now the system

$$\ddot{y} + \dot{y}|\dot{y}| + y^3 = 0$$

**Note 69.** Finding a Lyapunov function is more an art than a science. Fortunately several “artists” already discovered Lyapunov for many systems of interest.

Defining  $x = [x_1 \ x_2]' := [y \ \dot{y}]'$ , this system can be written as  $\dot{x} = f(x)$  as follows:

$$\dot{x}_1 = x_2, \tag{13.4}$$

$$\dot{x}_2 = -x_2|x_2| - x_1^3. \tag{13.5}$$

A candidate Lyapunov function for this system is

$$V(x) := \frac{x_2^2}{2} + \frac{x_1^4}{4}.$$

This function satisfies the requirements (i)–(iii). However,

$$\nabla_x V(x) \cdot f(x) = x_2 \dot{x}_2 + x_1^3 \dot{x}_1 = -|x_2|x_2^2$$

and therefore it does not satisfy (iv) because

$$\nabla_x V(x) \cdot f(x) = 0 \quad \text{for} \quad x_2 = 0, x_1 \neq 0.$$

However,

$$\nabla_x V(x) \cdot f(x) = 0 \quad \Rightarrow \quad x_2 = 0$$

so  $V$  can only stop decreasing if  $x_2$  converges to zero. Moreover, if we go back to (13.5) we see that  $x_2(t) = 0, \forall t$  must necessarily imply that  $x_1(t) = 0, \forall t$ . So although  $V(x)$  is not a Lyapunov function, it can still be used to prove that the origin is globally asymptotically stable. The argument just used to prove that the origin is asymptotically stable is due to J. P. LaSalle:

**LaSalle’s Invariance Principle.** *Suppose that there exists a scalar-valued function  $V : \mathbb{R}^n \rightarrow \mathbb{R}$  that satisfies the conditions (i)–(iii) of Lyapunov Stability Theorem as well as*

(iv)  $\nabla_x V(x) \cdot f(x)$  is negative semi-definite, which means that

$$\nabla_x V(x) \cdot f(x) \leq 0, \quad \forall x,$$

and, in addition, for every solution  $x(t)$  for which

$$\nabla_x V(x(t)) \cdot f(x(t)) = 0, \quad \forall t \geq 0 \tag{13.6}$$

we must necessarily have that

$$x(t) = 0, \quad \forall t \geq 0. \tag{13.7}$$

Then the origin is globally asymptotically stable for the system (13.3). In this case the function  $V(x)$  is called a weak Lyapunov function. □

### 13.5 Liénard equation and generalizations

LaSalle’s Invariance principle is especially useful to prove stability for systems with dynamics described by the *Liénard equation*:

$$\ddot{y} + b(y, \dot{y})\dot{y} + \lambda(y) = 0,$$

where  $y$  is a scalar and  $b(y, \dot{y}), \lambda(y)$  are functions such that

$$b(y, \dot{y}) > 0, \quad \forall \dot{y} \neq 0 \tag{13.8}$$

$$\lambda(y) \neq 0, \quad \forall y \neq 0 \tag{13.9}$$

$$\Lambda(y) := \int_0^y \lambda(z) dz > 0, \quad \forall y \neq 0 \tag{13.10}$$

$$\lim_{y \rightarrow \infty} \Lambda(y) = \infty. \tag{13.11}$$

**Note 70.** In general, LaSalle’s Invariance Principle is stated in a more general form... ► p. 119

**Note 71.** The system considered in Section 13.4 was precisely of this form.

This type of equation arises in numerous mechanical systems from Newton's law.

Defining  $x = [x_1 \ x_2]' := [y \ \dot{y}]'$ , the Liénard equation can be written as  $\dot{x} = f(x)$  as follows:

$$\dot{x}_1 = x_2, \quad (13.12)$$

$$\dot{x}_2 = -b(x_1, x_2)x_2 - \lambda(x_1). \quad (13.13)$$

A candidate Lyapunov function for this system is

$$V(x) := \frac{x_2^2}{2} + \Lambda(x_1). \quad (13.14)$$

This function satisfies the requirements (i)–(iii) and

$$\nabla_x V(x) \cdot f(x) = x_2 \dot{x}_2 + \lambda(x_1) \dot{x}_1 = -b(x_1, x_2)x_2^2 \leq 0.$$

Moreover, from (13.8) we conclude that

$$\nabla_x V(x) \cdot f(x) = 0 \quad \Rightarrow \quad x_2 = 0.$$

Because of (13.13), any trajectory with  $x_2(t) = 0, \forall t$  necessarily must have  $\lambda(x_1(t)) = 0, \forall t$ , which in turn implies that  $x_1(t) = 0, \forall t$  because of (13.9). Therefore  $V(x)$  is a weak Lyapunov function and the origin is globally asymptotically stable.

The type of weak Lyapunov function used for the Liénard equation can also be used for higher-order dynamical system of the form

$$M(q)\ddot{q} + B(q, \dot{q})\dot{q} + Lq = 0, \quad (13.15)$$

where  $q$  is a  $k$ -vectors and  $M(q), B(q, \dot{q})$ , and  $L$  are  $k \times k$  matrices with  $M(q)$  and  $L$  symmetric and positive definite.

Defining

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} := \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \in \mathbb{R}^{2k},$$

the system (13.15) can be written as  $\dot{x} = f(x)$  as follows:

$$\dot{x}_1 = x_2,$$

$$\dot{x}_2 = -M(x_1)^{-1} \left( B(x_1, x_2)x_2 + Lx_1 \right).$$

A candidate Lyapunov function for this system is

$$V(x) := \frac{1}{2}x_2' M(x_1)x_2 + \frac{1}{2}x_1' Lx_1. \quad (13.16)$$

This function satisfies the requirements (i)–(iii) and

$$\begin{aligned} \nabla_x V(x) \cdot f(x) &= x_2' M(x_1) \dot{x}_2 + \frac{1}{2} x_2' \left( \frac{dM(x_1)}{dt} \right) x_2 + \dot{x}_1' Lx_1 \\ &= -x_2' \left( B(x_1, x_2) - \frac{1}{2} \frac{dM(x_1)}{dt} \right) x_2. \end{aligned}$$

Using LaSalle's Invariance Principle, we conclude that the origin is globally asymptotically stable as long as

$$B(x_1, x_2) - \frac{1}{2} \frac{dM(x_1)}{dt}$$

is positive definite. This will be used shortly to design controllers for mechanical systems.

**Note 72.** This is how we got the Lyapunov function for the system in Section 13.4. Verify!

**Note 73.** Using the product rule:  $\frac{d(x'Mx)}{dt} = \frac{dx'}{dt}Mx + x'M\frac{dx}{dt} + x'\frac{dM}{dt}x$ . But since  $\frac{dx'}{dt}Mx$  is a scalar, it is equal to its transpose and we get  $\frac{d(x'Mx)}{dt} = 2x'M\frac{dx}{dt} + x'\frac{dM}{dt}x$ .

### 13.6 To probe further

**Note 70** (LaSalle’s Invariance Principle). LaSalle’s Invariance Principle is generally stated in the following form.

**LaSalle’s Invariance Principle.** Suppose that there exists a scalar-valued function  $V : \mathbb{R}^n \rightarrow \mathbb{R}$  that satisfies the conditions (i)–(iii) of Lyapunov Stability Theorem as well as

(iv)  $\nabla_x V(x) \cdot f(x)$  is negative semi-definite, which means that

$$\nabla_x V(x) \cdot f(x) \leq 0, \quad \forall x.$$

Then the origin is globally stable for the system (13.3). Moreover, let  $E$  denote the set of all points for which  $\nabla_x V(x) \cdot f(x) = 0$ , i.e.,

$$E := \{x \in \mathbb{R}^n : \nabla_x V(x) \cdot f(x) = 0\},$$

and let  $M$  be the largest invariant set for the system (13.3) that is fully contained in  $E$ . Then every solution to (13.3) approaches  $M$  as  $t \rightarrow \infty$ . In case  $M$  only contains the origin, the origin is globally asymptotically stable for the system (13.3).

The condition (iv) that appeared in Section (iv), requires that any solution  $x(t)$  that stays inside the set  $E$  forever, must necessarily be the identically zero [see equation (13.6)]. This means that the set  $M$  can only contain the origin, because otherwise there would be another solution  $x(t)$  that would start inside  $M \subset E$  and stay inside this set forever. □

**Note 74.** A set  $M$  is said to be invariant for the system (13.3) if every trajectory that starts inside  $M$  will remain inside this set forever.

### 13.7 Exercises

**13.2.** For the following systems, show that the origin is asymptotically stable using the Lyapunov function provided.

$$\begin{cases} \dot{x} = -x + y - xy^2 \\ \dot{y} = -x - y - x^2y \end{cases} \quad V(x, y) = x^2 + y^2 \quad (13.17)$$

$$\begin{cases} \dot{x} = -x^3 + y^4 \\ \dot{y} = -y^3(x + 1) \end{cases} \quad V(x, y) = x^2 + y^2 \quad (13.18)$$

$$\begin{cases} \dot{x} = -x + 4y \\ \dot{y} = -x - y^3 \end{cases} \quad V(x, y) = x^2 + 4y^2 \quad (13.19)$$

$$\begin{cases} \dot{x} = x + 4y \\ \dot{y} = -2x - 5y \end{cases} \quad V(x, y) = ax^2 + bxy + cy^2 \quad (13.20)$$

For the last system, determine possible values for the constants  $a$ ,  $b$ , and  $c$  (recall that  $V$  must be positive definite). □

**13.3.** For the following systems, show that the origin is asymptotically stable using the Lyapunov function provided.

$$\begin{cases} \dot{x} = y \\ \dot{y} = -x - y \end{cases} \quad V(x, y) = x^2 + y^2 \quad (13.21)$$

$$\begin{cases} \dot{x} = y \\ \dot{y} = -y|y| - 3x \end{cases} \quad V(x, y) = ax^2 + by^2 \quad (13.22)$$

For the last system, determine possible values for the constants  $a$  and  $b$  (recall that  $V$  must be positive definite). □

**13.4.** Consider the hovercraft model in Example 12.3. Show that if the generalized force vector is set to  $F = -q$ , then the origin is globally asymptotically stable. □



# Chapter 14

## Lyapunov-based designs

### Contents

14.1 Lyapunov-based controller . . . . .	121
14.2 Exercises . . . . .	122
14.3 Application to mechanical systems . . . . .	123
14.4 Exercises . . . . .	124

### 14.1 Lyapunov-based controller

In Lyapunov-based designs one starts by selecting a candidate Lyapunov function and then chooses the control signal for which the trajectories of the closed-loop do indeed decrease along the chosen Lyapunov function.

To understand how this can be done, consider again the vehicle shown in Figure 12.1 with units chosen so that all constant coefficient are numerically equal to 1

$$\ddot{y} = -\dot{y}|\dot{y}| + 1$$

and suppose we want  $y$  to converge to some constant reference value  $r$ . Defining the tracking error  $e := y - r$ , we conclude that

$$\ddot{e} = -\dot{e}|\dot{e}| + 1.$$

Setting the state

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} := \begin{bmatrix} e \\ \dot{e} \end{bmatrix}$$

the dynamics of the system can be written as  $\dot{x} = f(x, u)$  as follows:

$$\dot{x}_1 = x_2 \tag{14.1}$$

$$\dot{x}_2 = -x_2|x_2| - 1 + u, \tag{14.2}$$

and we would like to make the origin globally asymptotically stable.

In view of what we saw for the Liénard equation, we will try to make

$$V(x) := \frac{x_2^2}{2} + \lambda \frac{x_1^2}{2}, \quad \lambda > 0$$

**Note 75.** Compare with equation (13.14).

a Lyapunov function for the system by appropriate choice of  $u$ . This function satisfies the requirements (i)–(iii) and

$$\nabla_x V(x) \cdot f(x, u) = x_2 \dot{x}_2 + \lambda x_1 \dot{x}_1 = -x_2^2 |x_2| + x_2(-1 + u + \lambda x_1).$$

A simple way to make the system Lyapunov stable is then to select

$$-1 + u + \lambda x_1 = 0 \quad \Leftrightarrow \quad u = 1 - \lambda x_1 = 1 - \lambda(y - r),$$

which leads to

$$\nabla_x V(x) \cdot f(x, u) = -x_2^2 |x_2|,$$

and therefore the candidate  $V(x)$  becomes a weak Lyapunov function for the closed-loop:

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -x_2 |x_2| - 1 + u(x_1) = -x_2 |x_2| - \lambda x_1. \end{aligned}$$

**Attention!** A feedback linearization controller for (14.2) would cancel both the nonlinearity  $-x_2 |x_2|$  and the  $-1$  terms, using a controller of the form

$$u(x_1, x_2) = 1 + x_2 |x_2| - K_P x_1 - K_D x_2, \quad K_P, K_D > 0.$$

However, in the presence of measurement noise this would lead to the following closed-loop system

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -x_2 |x_2| - 1 + u(x_1 + n_1, x_2 + n_2) = -K_P x_1 - K_D x_2 + d \end{aligned}$$

where  $d$  is due to the noise and is equal to

$$d := -K_P n_1 - K_D n_2 - x_2 |x_2| + x_2 |x_2 + n_2| + n_2 |x_2 + n_2|.$$

The main difficulty with this controller is that  $n_2$  appears in  $d$  multiplied by  $x_2$  so even with little noise,  $d$  can be large if  $x_2$  is large.

Consider now the Lyapunov-based controller

$$u(x_1) = 1 - \lambda x_1$$

also in the presence of noise:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -x_2 |x_2| - 1 + u(x_1 + n_1) = -x_2 |x_2| - \lambda x_1 + d, \quad d = -\lambda n_1. \end{aligned}$$

This controller is not affected at all by noise in the measurement of  $x_2 = \dot{y}$  and the noise in the measurement of  $x_1 = y - r$  is not multiplied by the state.  $\square$

## 14.2 Exercises

**14.1.** Design a controller for the system (14.2) using the following candidate Lyapunov function

$$V(x) := \frac{x_2^2}{2} + \lambda x_1 \arctan(x_1), \quad \lambda > 0.$$

What is the maximum value of  $u$  that this controller requires? Can you see an advantage of using this controller with respect to the one derived before?  $\square$

**14.2.** Consider again system (14.2) and the following candidate Lyapunov function

$$V(x) := \frac{x_2^2}{2} + \lambda x_1 \arctan(x_1), \quad \lambda > 0.$$

Find a Lyapunov-based control law  $u(x_1, x_2)$  that keeps

$$\nabla_x V(x) \cdot f(x, u) \leq -x_2^2 |x_2|,$$

always using the  $u$  with smallest possible norm. This type of controller is called a *point-wise min-norm controller* and is generally very robust with respect to process uncertainty.  $\square$

### 14.3 Application to mechanical systems

Consider again a fully actuated mechanical system of the following form

$$M(q)\ddot{q} + B(q, \dot{q})\dot{q} + G(q) = u \quad (14.3)$$

and suppose that we want to make  $q$  converge to some constant value  $r$ . Defining

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} := \begin{bmatrix} q - r \\ \dot{q} \end{bmatrix} \in \mathbb{R}^{2k},$$

the system (14.3) can be written as  $\dot{x} = f(x, u)$  as follows:

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -M(x_1 + r)^{-1} \left( B(x_1 + r, x_2)x_2 + G(x_1 + r) - u \right). \end{aligned}$$

Based on what we saw in Section 13.5, we will try to make

$$V(x) := \frac{1}{2}x_2' M(x_1 + r)x_2 + \frac{\gamma_1}{2}x_1'x_1, \quad \gamma_1 > 0$$

**Note 76.** Compare with equation (13.16).

a Lyapunov function for the system by appropriate choice of  $u$ . This function satisfies the requirements (i)–(iii) and

$$\begin{aligned} \nabla_x V(x) \cdot f(x) &= x_2' M(x_1 + r)\dot{x}_2 + \frac{1}{2}x_2' \left( \frac{dM(x_1 + r)}{dt} \right) x_2 + \gamma_1 x_1' x_1 \\ &= -x_2' \left( B(x_1 + r, x_2)x_2 - \frac{1}{2} \frac{dM(x_1 + r)}{dt} x_2 + G(x_1 + r) - u - \gamma_1 x_1 \right). \end{aligned}$$

Since in general  $x_2' B(x_1 + r, x_2)x_2 \leq 0$ , a simple way to make the system Lyapunov stable is to select

$$\begin{aligned} -\frac{1}{2} \frac{dM(x_1 + r)}{dt} x_2 + G(x_1 + r) - u - \gamma_1 x_1 &= \gamma_2 x_2 \\ \Leftrightarrow u &= -\frac{1}{2} \frac{dM(x_1 + r)}{dt} x_2 + G(x_1 + r) - \gamma_1 x_1 - \gamma_2 x_2 \\ &= -\frac{1}{2} \frac{dM(q)}{dt} \dot{q} + G(q) - \gamma_1 (q - r) - \gamma_2 \dot{q}, \end{aligned}$$

which leads to

$$\nabla_x V(x) \cdot f(x) = -x_2' B(x_1 + r, x_2)x_2 - \gamma_2 x_2' x_2.$$

and therefore the candidate  $V(x)$  becomes a weak Lyapunov function for the closed-loop.

**Attention!** For several mechanical system

$$x_2' \left( B(x_1 + r, x_2)x_2 - \frac{1}{2} \frac{dM(x_1 + r)}{dt} x_2 \right) \geq 0.$$

For those systems it suffices to select

$$\begin{aligned} G(x_1 + r) - u - \gamma_1 x_1 &= \gamma_2 x_2 \quad \Leftrightarrow \quad u = G(x_1 + r) - \gamma_1 x_1 - \gamma_2 x_2 \\ &= Gq - \gamma_1 (q - r) - \gamma_2 \dot{q}, \end{aligned}$$

which leads to

$$\begin{aligned} \nabla_x V(x) \cdot f(x) &= -x_2' \left( B(x_1 + r, x_2)x_2 - \frac{1}{2} \frac{dM(x_1 + r)}{dt} x_2 \right) x_2 - x_2' B(x_1 + r, x_2)x_2 - \gamma_2 x_2' x_2 \\ &\leq -\gamma_2 x_2' x_2. \end{aligned} \quad \square$$

## 14.4 Exercises

**14.3.** Design a Lyapunov based controller for the inverted pendulum considered in Exercise 12.2 and compare its noise rejection properties with the feedback linearization controller previously designed. □

**14.4.** Re-design the controller in Exercise 14.3 to make sure that the control signal always remains bounded. Investigate its noise rejection properties.

*Hint: Draw inspiration from Exercises 14.1 and 14.2.* □

# Bibliography

- [1] R. Adrain. Research concerning the probabilities of the errors which happen in making observations. *The Analyst*, I:93–109, 1808. Article XIV. [15](#)
- [2] J. J. Craig. *Introduction to Robotics Mechanics and Control*. Addison Wesley, Reading, MA, 1986. [109](#)
- [3] L. Cremean, W. Dumbar, D. van Gogh, J. Hickey, E. Klavins, J. Meltzer, and R. Murray. The Caltech multi-vehicle wireless testbed. In *Proc. of the 41st Conf. on Decision and Contr.*, Dec. 2002. [110](#)
- [4] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Prentice Hall, Upper Saddle River, NJ, 4th edition, 2002. [6](#), [8](#), [45](#), [58](#), [65](#), [78](#), [85](#)
- [5] C. F. Gauss. Theoria motus corporum coelestium in sectionibus conicis solem ambientum (the theory of the motion of heavenly bodies moving around the sun in conic sections). Hamburg, Germany, 1809. URL [http://134.76.163.65/agora\\_docs/137784TABLE\\_OF\\_CONTENTS.html](http://134.76.163.65/agora_docs/137784TABLE_OF_CONTENTS.html). [15](#)
- [6] B. Hayes. Science on the farther shore. *American Scientist*, 90(6):499–502, Nov. 2002. [15](#)
- [7] L. Ljung. *System Identification: Theory for the user*. Information and System Sciences Series. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 1999. [15](#), [45](#)
- [8] R. A. Serway and R. J. Beichner. *Physics for Scientists and Engineers*. Brooks Cole, 5th edition, 1999. [108](#)
- [9] J. V. Vegete. *Feedback Control Systems*. Prentice Hall, Englewood Cliffs, NJ, 3rd edition, 1994. [77](#)
- [10] K. Zhou, J. C. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice Hall, Englewood Cliffs, NJ, 1996.