Advanced Undergraduate Topics in Control Systems Design

João P. Hespanha

March 26, 2024

Contents

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 output y an try to determine the system's transfer function (cf. Figure 1).

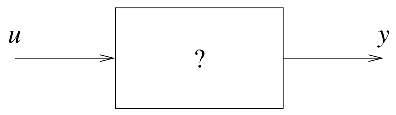


Figure 1. System identification from input/output experimental data

Pre-requisites

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

Further reading A more extensive coverage of system identification can be found, e.g., in [10].

Lecture 1

Computer-Controlled Systems

This lecture reviews basic concepts in computer-controlled systems:

Contents

1.1 Computer Control

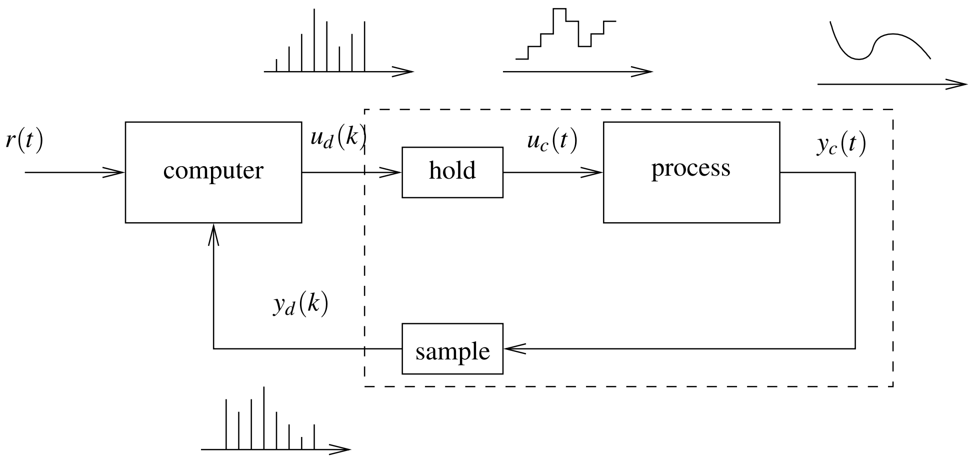


Figure 1.1. Computer control architecture. The components in the dashed boxed can be regarded as a discrete-time process to be controlled.

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 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:

$$u_c(t) = u_d(k), \qquad \forall t \in [kT_s, (k+1)T_s).$$

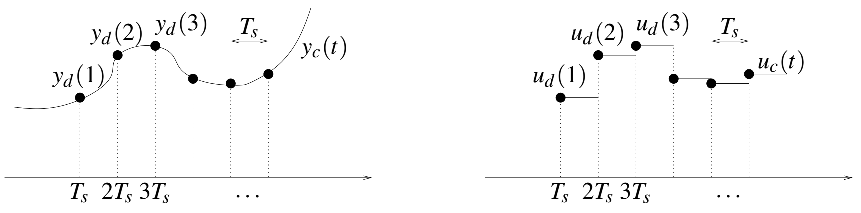


Figure 1.2. Sampling (left) and holding (right)

It is therefore sometimes convenient to regard the process to be controlled as a *discrete-time system* whose inputs and outputs are the discrete-time signals

Note 1. Many dynamical systems are inherently discrete and do not arise from sampling an underlying continuous process. ► p. 6

$$y_d(k) = y_c(kT_s), \qquad u_d(k) = u_c(kT_s), \qquad k \in \{0, 1, 2, \dots\}.$$

The identification techniques that we will study allow us to take either of the following approaches:

1. Estimate directly the transfer function from $u_c(t)$ to $y_c(t)$ of the original continuous-time process.

   Identifying the continuous-time transfer function will generally lead to better results when the sampling frequency is high and the underlying process is naturally modeled by a differential equation.

2. Estimate the transfer function from $u_d(k)$ to $y_d(k)$ of the discrete-time system and, if desired, recover the underlying continuous-time transfer function.

   When the sampling frequency is low (when compared to the natural frequency of the system) or the time variable is inherently discrete, one should do the identification directly in discrete-time.

Note. See Section 1.4 on how to (approximately) recover the underlying continuous-time transfer function from the discrete-time transfer function of a sampled system. ► p. 12

Note 1 (Inherently discrete systems)**.** Many dynamical systems are inherently discrete and do not arise from sampling an underlying continuous process. Examples of processes whose time variables are inherently discrete include population dynamics for which k refers of the number of generations or a stock daily minimum/maximum price for which k refers to a day. □

1.2 Continuous-time Systems

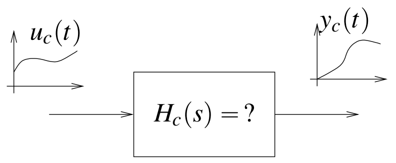


Figure 1.3. Continuous-time system

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 $k \geqslant 3$ are denoted by denote by $y^{(k)}(t)$.

$$y_c^{(n)} + \beta_{n-1} y_c^{(n-1)} + \dots + \beta_2 \ddot{y}_c + \beta_1 \dot{y}_c + \beta_0 y_c \\ = \alpha_m u_c^{(m)} + \alpha_{m-1} u_c^{(m-1)} + \dots + \alpha_2 \ddot{u}_c + \alpha_1 \dot{u}_c + \alpha_0 u_c. \quad (1.1)$$

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

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

Note 2. The *(unilateral) Laplace transform* of a signal $x(t)$ is given by

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

See [5, Appendix A] for a review of Laplace transforms.

In particular, when x and all its derivatives are zero at time $t = 0$, the Laplace transform of the ℓth 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

$$\begin{aligned} s^nY_c(s) + \beta_{n-1}s^{n-1}Y_c(s) + \cdots + \beta_2 s^2 Y_c(s) + \beta_1 sY_c(s) + \beta_0 Y_c(s) \\ = \alpha_m s^m U_c(s) + \alpha_{m-1}s^{m-1}U_c(s) + \cdots + \alpha_2 s^2 U_c(s) + \alpha_1 sU_c(s) + \alpha_0 U_c(s). \end{aligned}$$

This leads to

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

MATLAB® Hint 1. `tf(num,den)` creates a continuous-time transfer function with numerator and denominator specified by `num`, `den`. ► p. 13

where

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

MATLAB® Hint 2. `zpk(z,p,k)` creates a continuous-time transfer function with zeros, poles, and gain specified by `z`, `p`, `k`. ► p. 13

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.

Note. 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.

1.2.1 Steady-state Response to Sine Waves

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

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

Note. The angular frequency ω 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.

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

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

where the gain g and the phase ϕ are given by the following formulas

$$g := |H_c(j\omega)|, \qquad \phi := \angle H_c(j\omega), \tag{1.4}$$

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

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 $H_c(j\omega)$

MATLAB® Hint 3. `bode(sys)` draws the Bode plot of the system `sys`. ► p. 14

both as a function of the angular frequency ω (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.

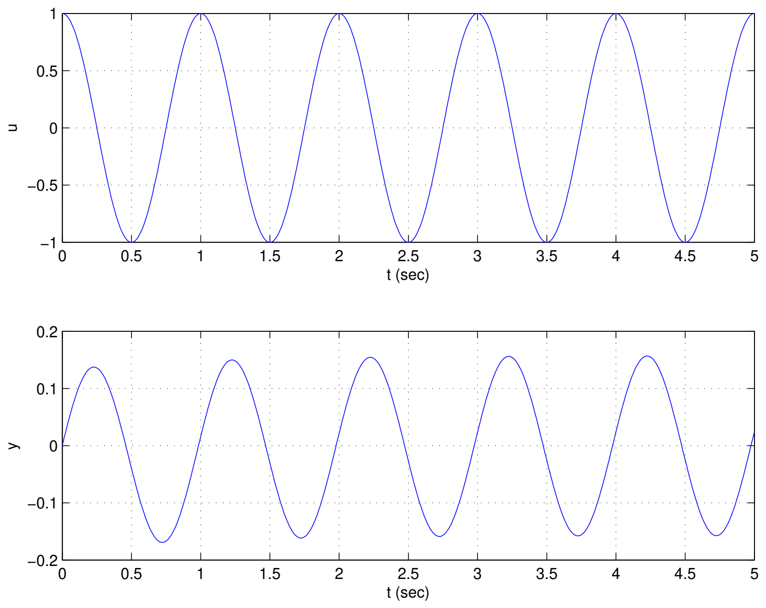


Figure 1.4. Response of a system with transfer function $H_c(s) = \frac{1}{s+1}$ to a sinusoidal input with frequency $f = 1$Hz, corresponding to $\omega = 2\pi$ rad/sec. The Bode plot of the system is shown in Figure 1.5.

1.2.2 Impulse and Step Responses

The *impulse response* of the system (1.1) is defined as the inverse Laplace transform of its transfer function (1.2):

$$h_c(t) = \mathscr{L}^{-1}\big[H_c(s)\big] := \frac{\alpha_m s^m + \alpha_{m-1}s^{m-1} + \cdots + \alpha_1 s + \alpha_0}{s^n + \beta_{n-1}s^{n-1} + \cdots + \beta_1 s + \beta_0},$$

and can be viewed as the system's response to a δ-Dirac impulse.

A δ-Dirac impulse is a signal that is zero everywhere except for $t = 0$ but integrates to one. Such signals are mathematical abstractions that cannot really be applied to real systems. However, one can apply a very narrow pulse, with a small duration $\varepsilon > 0$ and magnitude $1/\varepsilon$ that integrates to 1, as shown in Figure 1.6. Such input can produce an output that is a good approximation to the impulse response.

Note. We will find in Section 2.2.1 that there are better ways to obtain the impulse response of a system. ► p. 18

A δ-Dirac impulse can also be viewed as the derivative of a step

$$u_{\text{step}}(t) := \begin{cases} 0 & t < 0 \\ 1 & t \geqslant 0, \end{cases}$$

or conversely, the step is the integral of a δ-Dirac. Because of linearity, this means that the system's response to a step input will be the integral of the impulse response, or conversely, the impulse response is equal to the derivative of the step response (cf. Figure 1.7).

1.3 Discrete-time Systems

Consider now the discrete-time system in Figure 1.8. It turns out that the discrete-time inputs and outputs can be related by an equation of the following form that mimics (1.1):

$$y_d(k+n) + \beta_{n-1}y_d(k+n-1) + \cdots + \beta_2 y_d(k+2) + \beta_1 y_d(k+1) + \beta_0 y_d(k)$$

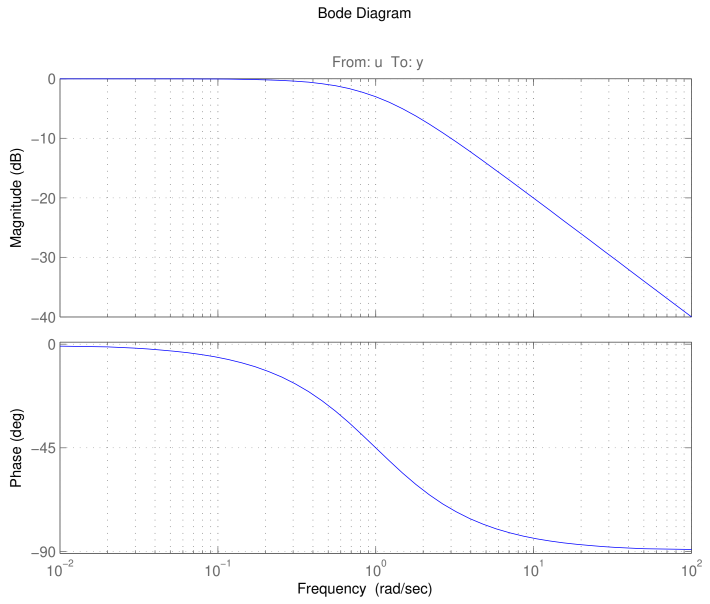


Figure 1.5. Bode plot of a system with transfer function $H_c(s) = \frac{1}{s+1}$.

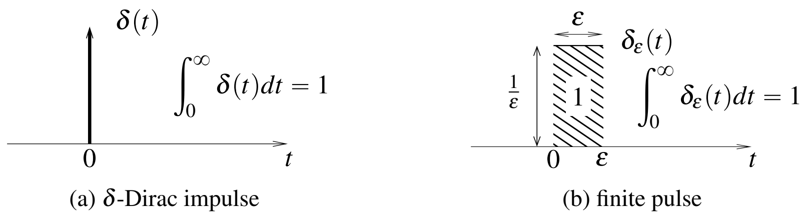


Figure 1.6. Practical approximation to a continuous-time δ-Dirac impulse. .

$$= \alpha_m u_d(k+m) + \alpha_{m-1} u_d(k+m-1) + \cdots + \alpha_2 u_d(k+2) + \alpha_1 u_d(k+1) + \alpha_0 u_d(k). \quad (1.5)$$

This equation shows that the discrete-time output can be computed recursively by the following equation:

$$y_d(k+n) = -\beta_{n-1} y_d(k+n-1) - \cdots - \beta_2 y_d(k+2) - \beta_1 y_d(k+1) - \beta_0 y_d(k)$$
$$+ \alpha_m u_d(k+m) + \alpha_{m-1} u_d(k+m-1) + \cdots + \alpha_2 u_d(k+2) + \alpha_1 u_d(k+1) + \alpha_0 u_d(k). \quad (1.6)$$

Note. Since the output cannot depend on future inputs, we must have $n \geqslant m$ for (1.6) to be physically realizable.

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

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

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

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

See [5, Section 8.2] for a review of Laplace z-transforms.

Taking z-transforms to both sides of (1.5) and assuming that both y_d and u_d are zero before time $n \geqslant m$, we obtain

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

MATLAB® Hint 4. `tf(num,den,Ts)` creates a discrete-time transfer function with sampling time `Ts` and numerator and denominator specified by `num`, `den`. ► p. 13

This leads to

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

MATLAB® Hint 5. `zpk(z,p,k,Ts)` creates a transfer function with sampling time `Ts` zeros, poles, and gain specified by `z`, `p`, `k`. ► p. 13

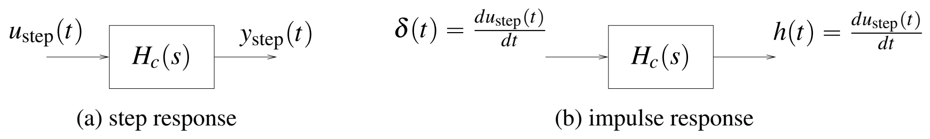


Figure 1.7. Impulse versus step responses

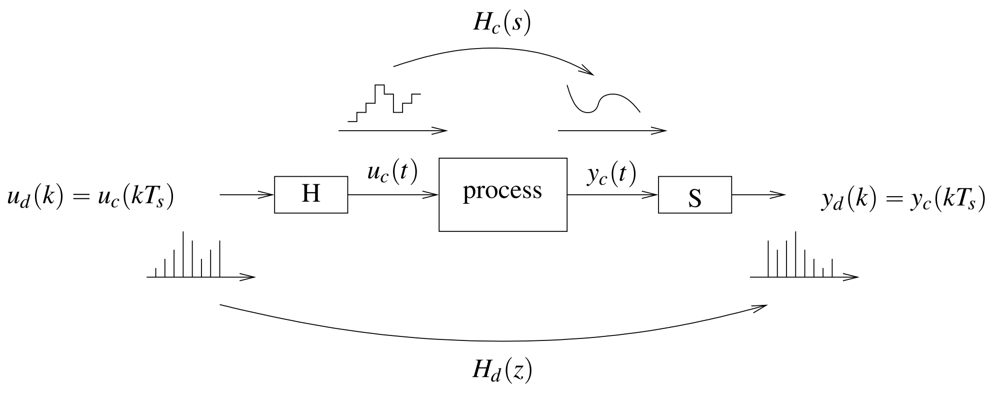


Figure 1.8. Discrete-time system

where

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

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.

Note. 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.

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

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

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

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

where the gain g and phase ϕ are given by

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

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

Note 4. Discrete-time angular frequencies take values from 0 to π. In particular, $\Omega = \pi$ corresponds to the "fastest" discrete-time signal $u_d(k) = A\cos(\pi k) = A(-1)^k$, which corresponds to a continuous-time frequency $f = \frac{1}{2T_s}$ Hz equal to half the sampling rate, or $\omega = \frac{\pi}{T_s}$ rad/sec , also known as the Nyquist frequency. ► p. 15
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 6. `bode(sys)` draws the Bode plot of the system `sys`. ► p. 14

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 Ω (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.

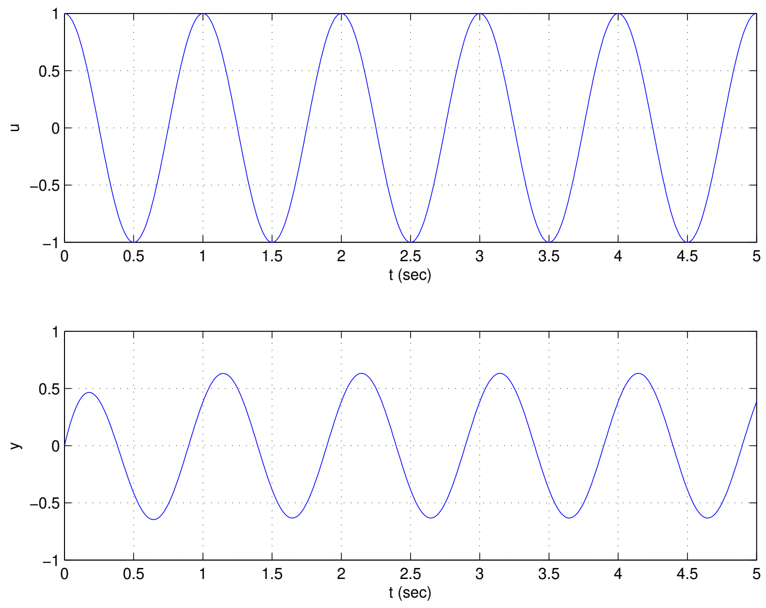


Figure 1.9. Response a system with transfer function $H_d(z) = \frac{0.05}{z-0.95}$ to a sinusoidal input with frequency $f = 1$Hz, $\omega = 2\pi$ rad/sec sampled with a period equal to $T_s = .01$ sec. The corresponding discrete-time frequency is given by $\Omega = .02\pi$ and the Bode plot of the system is shown in Figure 1.10.

1.3.2 Impulse and Step Responses

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

Note. In contrast to continuous-time δ-Dirac impulses, discrete-time steps can be applied to the input of a system.

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

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

$$u(k) = s(k) := \begin{cases} 0 & k < 0 \\ 1 & k \geqslant 0, \end{cases}$$

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

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

the impulse response $h(k)$ (which is the output to $\delta(k)$) can be obtained from the output $y_{\text{step}}(k)$ to $s(k)$ by

$$h(k) = y_{\text{step}}(k) - y_{\text{step}}(k-1), \quad \forall k \in \{0, 1, 2 \dots\}.$$

This is a consequence of linearity (cf. Figure 1.11).

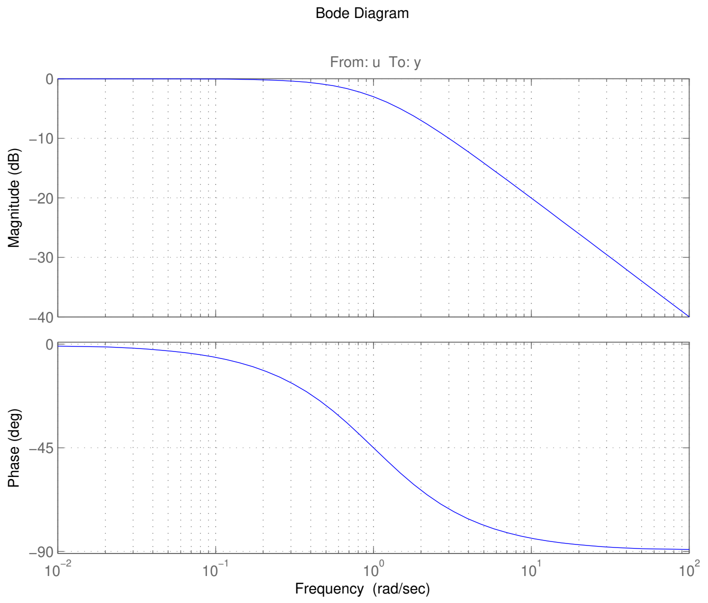
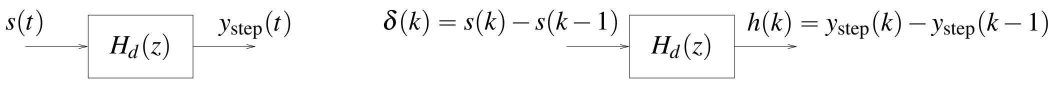


Figure 1.10. Bode plot of a system with transfer function $H_d(s) = \frac{.05}{z-.95}$


$s(t)$
$H_d(z)$
$y_{\text{step}}(t)$
$\delta(k) = s(k) - s(k-1)$
$H_d(z)$
$h(k) = y_{\text{step}}(k) - y_{\text{step}}(k-1)$

(a) step response (b) impulse response

Figure 1.11. Impulse versus step responses

1.4 Discrete-time 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 1.12. To understand how this can be done, consider a continuous-time integrator

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

with transfer function

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

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

$$y_d(k) = y_c(kT_s), \qquad u_d(k) = u_c(kT_s), \qquad k \in \{0, 1, 2, \dots\}.$$

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

Note 5. Equation (1.12) is exact when $u_c(t)$ remains constant throughout the whole sample time $[t, t+T_s]$, which is case when $u_c(t)$ is the output of a hold block.

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

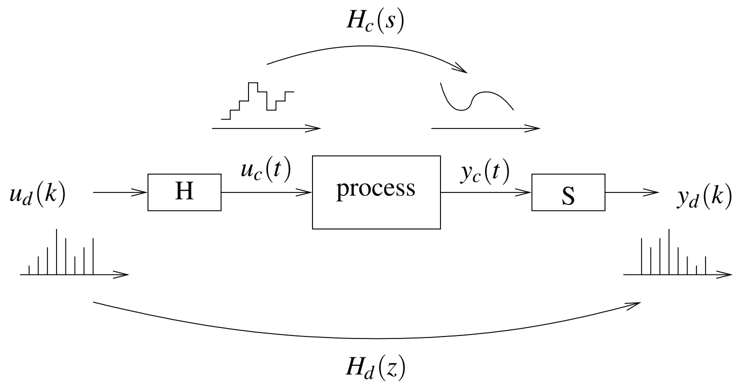


Figure 1.12. Discrete vs. continuous-time transfer functions

Taking the z-transform we conclude that

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

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

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:

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

MATLAB® Hint 7. `c2d` and `d2c` convert transfer functions from continuous- to discrete-time and vice-versa. ► p. 14

In general, 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:

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

Note 6. Why? ► p. 15

Note 7. 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. ► p. 16

1.5 MATLAB® Hints

MATLAB® Hint 1, 4 (`tf`)**.** The command `sys_tf=tf(num,den)` assigns to `sys_tf` a MATLAB® 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 $\frac{2s}{s^2+3}$ one should use `num=[2 0];den=[1 0 3];`

The command `sys_tf=tf(num,den,Ts)` assigns to `sys_tf` a MATLAB® 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® Hint 2, 5 (`zpk`)**.** The command `sys_tf=zpk(z,p,k)` assigns to `sys_tf` a MATLAB® 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 $\frac{2s}{(s+1)(s+3)}$ one should use `z=0;p=[1,3];k=2;`

The command `sys_tf=zpk(z,p,k,Ts)` assigns to `sys_tf` a MATLAB® 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 3, 6, 32 (`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 7 (`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 $\frac{2s}{s^2+3}$ one 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 $\frac{2s}{(s+1)(s+3)}$ one 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 $\frac{2s}{s^2+3}$ one 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 $\frac{2s}{(s+1)(s+3)}$ one 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 4 (Discrete-time 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

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

we must have

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

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 π. 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 $f = \frac{1}{2T_s}$ equal to half the sampling rate, also known as the Nyquist frequency, as shown in Figure 1.13. □

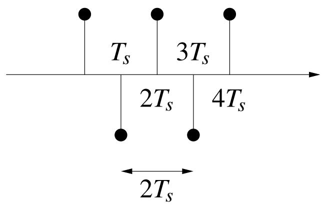


Figure 1.13. Discrete-time sinusoid at the Nyquist frequency $u_d(k) = \cos(\pi k) = (-1)^k$, which corresponds to $\Omega = \pi$. This signal would be obtained by sampling a continuous time sinusoid $u_c(t) = \cos\left(\frac{\pi}{T_s}t\right)$, with frequency $f = \frac{1}{2T_s}Hz$ or $\omega = \frac{\pi}{T_s}$ rad/sec.

Note 6 (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.,

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

(see Figure 1.14b). 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

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

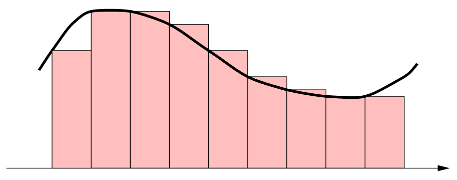
Taking the z-transform we conclude that

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

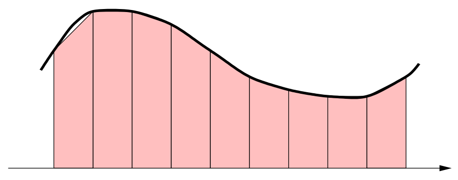
Comparing this with (1.11), we observe that we can go directly from a continuous-time transfer function to the discrete-time one using the so called *Tustin or bilinear* transformation:

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

Figure 1.14 compares the use of this transformation versus the Euler transformation in Section 1.4, to approximate a continuous-time integrator.



(a) A discrete-time integrator obtained using the Euler transformation approximates the integral by the sum of *piecewise constant* blocks: $y_d(k+1) = y_d(k) + T_s u_d(k)$.



(b) A discrete-time integrator obtained using the Tustin transformation approximates the integral by the sum of *piecewise linear* blocks: $y_d(k+1) = y_d(k) + T_s \frac{u_d(k+1)+u_d(k)}{2}$.

Figure 1.14. Converting a continuous-time integrator with transfer function $H_c(s) = \frac{1}{s}$ to discrete time using the Euler (a) or the Tustin (b) approximation results in discrete-time systems with transfer functions $H_d(z) = \frac{T_s}{z-1}$ and $H_d(z) = \frac{T_s(z+1)}{z-1}$, respectively, that approximate the integrator with different levels of accuracy.

Note 7 (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 $z = -1$. □

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 $u_d(k)$. □

Lecture 2

Non-parametric Identification

This lecture presents basic techniques for non-parametric identification.

Note. The terminology "non-parametric identification" will become clear in Lecture 3, when we introduce "parametric identification."

Contents

2.1 Non-parametric Methods

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

Problem 2.1 (Nonparametric continuous-time frequency response identification)**.** Determine the *frequency response* $H_c(j\omega)$ over a range of frequencies $\omega \in [\omega_{\min}, \omega_{\max}]$.

Problem 2.2 (Nonparametric discrete-time frequency response identification)**.** Determine the *frequency response* $H_d(e^{j\Omega})$ over a range of frequencies $\Omega \in [\Omega_{\min}, \Omega_{\max}]$.

Problem 2.3 (Nonparametric continuous-time impulse response identification)**.** Determine the *impulse response* $h_c(t) = \mathscr{L}^{-1}[H_c(s)]$ from time $t = 0$ to $t = T$ of a continuous-time system .

Problem 2.4 (Nonparametric discrete-time impulse response identification)**.** Determine the *impulse response* $h_d(k) = \mathscr{Z}^{-1}[H_d(z)]$ from time $k = 0$ to $k = N$ of a discrete-time system .

Problems 2.1 and 2.2 are useful for *frequency-domain* controller design methods like loop-shaping or the Nyquist criterion, whereas Problems 2.3 and 2.4 are useful for *time-domain* controller design methods like the Ziegler-Nichols rules to tune PID controllers [5]. However, one can also recover the transfer function from the impulse response by taking the Laplace or the z-transform:

MATLAB® Hint 8. `fft(h)` computes the values of $H(z)$ for $z = e^{j\Omega}$. ► p. 27

$$H_c(s) := \int_0^\infty e^{-st} h_c(t)dt \approx \int_0^T e^{-st} h_c(t)dt,$$
$$H_d(z) := \mathscr{Z}[h_d(k)] = \sum_{k=0}^{\infty} h_d(k)z^{-k} \approx \sum_{k=0}^{N} h_d(k)z^{-k},$$

assuming that T is sufficiently large to that $h_c(t) \approx 0$, $\forall t > T$ (in continuous time) or that N is large enough so that $h_d(k) \approx 0$, $\forall k > N$ (in discrete time). Therefore, impulse response identification can also be used for frequency-domain controller design methods.

Attention! Systems with an integrator (i.e., a continuous-time pole at $s = 0$ or a discrete-time pole at $z = 1$) are not BIBO stable. However, if all other discrete-time poles have magnitude strictly smaller than one, this difficulty can be overcome using a technique that will be discussed in Sections 4.5 and 6.3. ► p. 45

Throughout this chapter we will assume that the *system is BIBO stable*, i.e., that

1. all poles of the continuous-time transfer function $H_c(s)$ have real-part smaller than zero or equivalently that $h_c(t)$ converges to zero exponentially fast; and
2. all poles of the discrete-time transfer function $H_d(z)$ have magnitude smaller than one or equivalently that $h_d(k)$ converges to zero exponentially fast.

A detailed treatment of this subject can be found, e.g., in [10, Chapter 7].

2.2 Continuous-time Time-domain Identification

We start by discussing a few methods that can be used to solve the continuous-time impulse response identification Problem 2.3.

2.2.1 Impulse Response Method

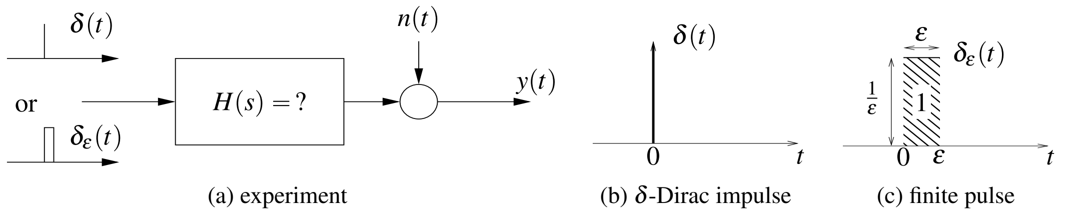


Figure 2.1. Impulse response method

If we were able to apply a δ-Dirac impulse to the input of a system, the measured output would be precisely the impulse response, possibly corrupted by some measurement noise $n(t)$:

$$y(t) = h(t) + n(t).$$

Hopefully, the noise term $n(t)$ is small when compared to $h(t)$ and one could take the measured output as an estimate for the impulse response:

$$\hat{h}(t) = y(t), \qquad \forall t \geqslant 0.$$

While the perfect δ-Dirac impulse cannot be applied, one could apply a finite pulse with unit area, as shown in Figure 2.1, which would provide a good approximation to the δ-Dirac impulse. However, such impulses are rarely representative of the 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.

Note. In computer control systems with sample and hold, impulses are held for a sample period T_s (recall discussion in Lecture 1). This means that the pulse duration will always be a multiple of the sample time, so we need to choose the pulse duration $\varepsilon = kT_s$ and the pulse magnitude $1/\varepsilon = 1/(kT_s)$, for some integer $k \geqslant 1$.

2.2.2 Step Response

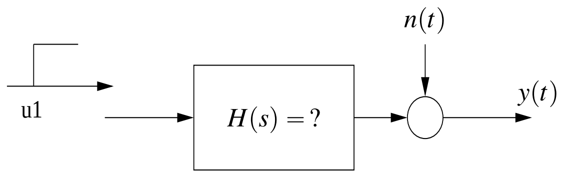


Figure 2.2. Step response method

While applying a δ-Dirac impulse to the input of a system is not possible, applying a scaled step is generally possible:

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

$$u(t) = \begin{cases} 0 & t < 0 \\ \alpha & t \geqslant 0 \end{cases} \quad \Rightarrow \quad \mathscr{L}\big[u(t)\big] = U(s) = \frac{\alpha}{s},$$

as in Figure 2.2. In this case, the Laplace transform of the output will be given by

$$Y(s) = H(s)U(s) + N(s) = H(s)\frac{\alpha}{s} + N(s),$$

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

$$H(s) = \frac{sY(s)}{\alpha} - \frac{sN(s)}{\alpha}$$

Taking inverse Laplace transforms, we conclude that

$$h(t) = \frac{1}{\alpha}\frac{dy(t)}{dt} - \frac{1}{\alpha}\frac{dn(t)}{dt}.$$

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

$$\hat{h}(t) = \frac{1}{\alpha}\frac{dy(t)}{dt}, \qquad \forall t \geqslant 0.$$

Attention! The choice of α is generally critical to obtain a good estimate for the impulse response:

(i) $|\alpha|$ should be *large* to make sure that $\frac{1}{\alpha}\frac{dn(t)}{dt}$ is indeed negligible when compared to $h(t)$;

(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.

As one increases α, 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 $\alpha/2$: in the linear region of operation, the identified impulse-response should not change.

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

$$\hat{h}(t) = h(t) + \frac{1}{\alpha}\frac{dn(t)}{dt}.$$

As shown in Figure 2.3, there is usually an "optimal" input level that needs to be determined by trial-and-error. See also the discussion in Section 5.1. □

Attention! The main *weakness* of the step-response method is that it can amplify noise because $\frac{1}{\alpha}\frac{dn(t)}{dt}$ can be much larger that $n(t)$ if the noise has small magnitude but with a strong high-frequency components, which is quite common. □

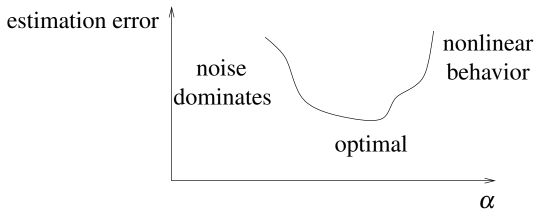


Figure 2.3. Optimal choice of input magnitude

2.2.3 Other Inputs

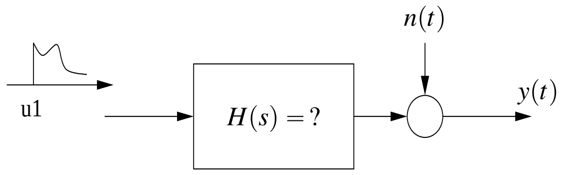


Figure 2.4. Impulse response from the response to a generic input

One can determine the impulse response of a system using any input and not just a pulse- or step-input, as shown in Figure 2.4. Take an arbitrary input $u(t)$, with Laplace transform $U(s)$. The Laplace transform of the output will be given by

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

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

$$H(s) = \frac{Y(s)}{U(s)} - \frac{N(s)}{U(s)}$$

Taking inverse Laplace transforms, we conclude that

$$h(t) = \mathscr{L}^{-1}\Big[\frac{Y(s)}{U(s)}\Big] - \mathscr{L}^{-1}\Big[\frac{N(s)}{U(s)}\Big].$$

MATLAB® Hint 9. The identification toolbox command `impulseest` performs impulse response estimation for arbitrary inputs. ► p. 27

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

$$\hat{h}(t) = \mathscr{L}^{-1}\Big[\frac{Y(s)}{U(s)}\Big], \qquad \forall t \geqslant 0.$$

The error for this estimate is given by

$$e(t) := \hat{h}(t) - h(t) = \mathscr{L}^{-1}\Big[\frac{N(s)}{U(s)}\Big],$$

showing that, in general, we want $U(s)$ to be much larger than $N(s)$, over the range of frequencies for which we care to get a good estimate of $h(t)$.

2.3 Discrete-time Time-domain Identification

We now discuss methods to solve the discrete-time impulse response identification Problem 2.4.

2.3.1 Impulse Response Method

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

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

Note. 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 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:

$$\hat{h}(k) = \frac{y(k)}{\alpha}, \qquad k \in \{0, 1, \ldots, N\}.$$

Attention! The choice of α 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. 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 α, 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. As shown in Figure 2.5, there is usually an "optimal" input level that needs to be determined by trial-and-error. See also the discussion in Section 5.1. □

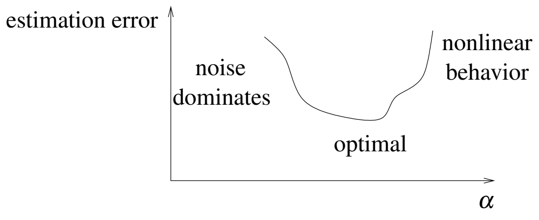


Figure 2.5. Optimal choice of input magnitude

2.3.2 Step 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*:

$$u(k) = \begin{cases} \alpha & k \geqslant 0 \\ 0 & k < 0 \end{cases} \quad \Rightarrow \quad \mathscr{Z}\big[u(k)\big] = U(z) = \frac{\alpha}{1 - z^{-1}}.$$

Note. 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.

In this case, 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:

$$\hat{h}(k) = \frac{y(k) - y(k-1)}{\alpha}, \qquad k \in \{0, 1, \ldots, N\}.$$

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 σ. In this case,

Note 8. Why? ► p. 28

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

which means that the noise is amplified by approximately 41%. □

2.3.3 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:

$$H(z) = \frac{Y(z)}{U(z)} - \frac{N(z)}{U(z)}$$

Taking inverse z-transforms, we conclude that

$$h(k) = \mathscr{Z}^{-1}\left[\frac{Y(z)}{U(z)}\right] - \mathscr{Z}^{-1}\left[\frac{N(z)}{U(z)}\right].$$

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

MATLAB® Hint 9. The identification toolbox command `impulseest` performs impulse response estimation for arbitrary inputs. ► p. 27

$$\hat{h}(k) = \mathscr{Z}^{-1}\left[\frac{Y(z)}{U(z)}\right], \qquad k \in \{0, 1, \ldots, N\}.$$

2.4 Continuous-time Frequency Response Identification

We now consider methods to solve the continuous-time frequency response identification Problem 2.1.

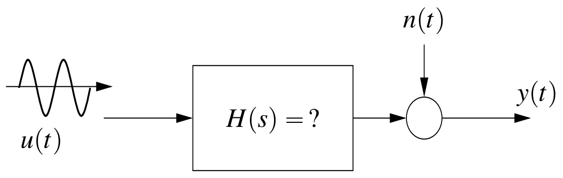


Figure 2.6. Sine wave testing

2.4.1 Sine-wave Testing

Suppose that one applies an sinusoidal input of the form

$$u(t) = \alpha \cos(\omega t), \qquad \forall t \in [0, T], \tag{2.1}$$

as in Figure 2.6. Since we are assuming that $H(s)$ is BIBO stable, the measured output is given by

$$y(t) = \alpha A_\omega \cos\left(\omega t + \phi_\omega\right) + \varepsilon(t) + n(t), \tag{2.2}$$

where

1. $A_\omega := |H(j\omega)|$ and $\phi_\omega := \angle H(j\omega)$ are the magnitude and phase of the transfer function $H(s)$ at $s = j\omega$;
2. $\varepsilon(t)$ is a transient signal that converges to zero as fast as $ce^{-\lambda t}$, where λ is the absolute value of the real past of the pole of $H(s)$ with largest (least negative) real part and c some constant; and
3. $n(t)$ corresponds to measurement noise.

When $n(t)$ is much smaller than αA_ω and for t sufficiently large so that $\varepsilon(t)$ is negligible, we have

$$y(t) \approx \alpha A_\omega \cos\left(\omega t + \phi_\omega\right).$$

This allows us to recover both $A_\omega := |H(j\omega)|$ and $\phi_\omega := \angle H(j\omega)$ from the magnitude and phase of $y(t)$. Repeating this experiment for several inputs of the form (2.1) with distinct frequencies ω, one obtains several points in the Bode plot of $H(s)$ and, eventually, one estimates the frequency response $H(j\omega)$ over the range of frequencies of interest.

Note. To minimize the errors caused by noise, the amplitude α should be large. However, it should still be sufficiently small so that the process does not leave the linear regime. A similar trade-off was discussed in Section 2.2.2, regarding the selection of the step magnitude. ► p. 19

2.4.2 Correlation Method

Especially when there is noise, it may be difficult to determine the amplitude and phase of $y(t)$ by inspection. The *correlation method* aims at solving this task, thus improving the accuracy of frequency response identification with sine-wave testing.

Suppose that the input $u(t)$ in (2.1) was applied, resulting in the measured output $y(t)$ given by (2.2), which was measured at the K sampling times $0, T_s, 2T_s, \ldots, (K-1)T_s$. In the correlation method we compute

$$Y_\omega := \frac{1}{K}\sum_{k=0}^{K-1} e^{-j\omega T_s k} y(T_s k) = \frac{1}{K}\sum_{k=0}^{K-1}\Big(\cos(\omega T_s k) - j\sin(\omega T_s k)\Big) y(T_s k).$$

Note. The expression for Y_ω very much resembles the definition of the Laplace transform, with the integral replaced by a finite summation. In essence, we are trying to estimate the Laplace transform of the output, from which we will get the amplitude and phase of the transfer function.

Using the expression for $y(t)$ in (2.5), we conclude after fairly straightforward algebraic manipulations that

$$Y_\omega = \frac{\alpha}{2}H(j\omega) + \frac{\alpha}{2K}H^*(j\omega)\frac{1-e^{-2j\omega T_s K}}{1-e^{-2j\omega T_s}} + \frac{1}{T_s K}\sum_{k=0}^{K-1} T_s e^{-j\omega T_s k}\big(\varepsilon(T_s k) + n(T_s k)\big).$$

Note 9. Why? ► p. 28

Note 10. Since the process is BIBO stable, $\varepsilon(t)$ converges to zero and therefore it has no pure sinusoids. However, a periodic noise term with frequency exactly equal to ω would lead to trouble. ► p. 24

Note 11. Y_ω is a complex number so we get estimates for the amplitude and phase of $H(j\omega)$. ► p. 24

As $K \to \infty$, the second term converges to zero and the summation at the end converges to the Laplace transform of $\varepsilon(t) + n(t)$ at the point $s = j\omega$. We thus conclude that

$$Y_\omega \to \frac{\alpha}{2} H(j\omega) + \lim_{K\to\infty} \frac{1}{T_s K}\big(E(j\omega) + N(j\omega)\big),$$

where $E(s)$ and $N(s)$ denote the Laplace transforms of $\varepsilon(t)$ and $n(t)$, respectively. As long as $n(t)$ and $\varepsilon(t)$ do not contain pure sinusoidal terms at the frequency ω, the Laplace transforms are finite and therefore the limit is equal to zero. This leads to the following estimate for $H(j\omega)$

$$\widehat{H(j\omega)} = \frac{2}{\alpha} Y_\omega, \tag{2.3}$$

which is accurate for large K.

Note 10 (Minimizing the impact of initial conditions and noise)**.** For a For a BIBO stable process, $\varepsilon(t)$ converges to zero and therefore it has no pure sinusoids. However, it is still a good idea to start collecting data only after the time at which $y(t)$ appears to have stabilized into a pure sinusoid. To make sure that the input is still of the form (2.1), we need to skip a number of *full periods of the input signal.*

The correlation method, relies heavily on the noise not having pure sinusoidal terms of the frequency ω. In case the noise does have pure sinusoidal terms, one should avoid estimating the transfer function at those frequencies and, if needed, rely on extrapolation from nearby frequencies. A common source of noise with pure sinusoids are analog-to-digital converters that try to "interpolate" between consecutive quantization values, by switching rapidly between those values. As illustrated in Figure 2.7, this results in large noise near the Nyquist frequency of $f = \frac{1}{2T_s}$. This means that one generally cannot trust results from the correlation method at frequencies close to the Nyquist frequency. □

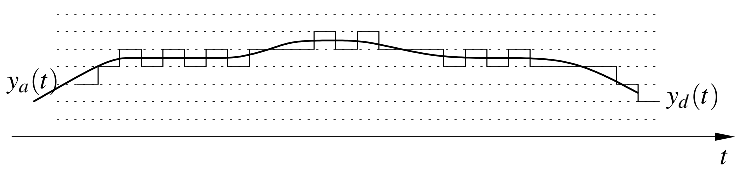


Figure 2.7. High-frequency noise caused by analog-to-digital conversion (ADC): the solid thick line shows the original analog signal $y_a(t)$ and the solid thin line the same signal after its conversion to digital values $y_d(t)$. The dotted lines show the quantization levels of the conversion to digital values.

Note 11 (Recovering amplitude and phase of a transfer function)**.** The correlation method provides estimates of $H(j\omega)$ as a complex number, from which the amplitude and phase of the transfer function needs to be extracted. Estimating the amplitude is straightforward, but the phase comes with an ambiguity of 2π as it is not possible to distinguish between the angles ϕ and $2\pi + \phi$ for a complex number.

Note. The *relative degree* of a SISO transfer function is equal to the number of poles (or the degree of the denominator polynomial) minus the number of zeros (or the degree of the numerator polynomial).

Determining the "initial phase" as $\omega \to 0$ can be discovered by looking at the magnitude plot. Specifically, as $\omega \to 0$ we will always have something like

$$\lim_{\omega\to 0} H(j\omega) = \lim_{\omega\to 0} \frac{k}{j^r \omega^r}$$

for some integer $r \geqslant 0$, which is equal to *relative degree*. This means that we basically only have a few options:

1. If $|H(j\omega)|$ converges to a finite constant as $\omega \to 0$, we must have $r = 0$ and the initial phase must be:

$$r = 0 \quad \Rightarrow \quad \begin{cases} \angle H(j\omega)_{\omega=0} = 0 & k > 0 \\ \angle H(j\omega)_{\omega=0} = \pi & k < 0 \end{cases}$$

2. If $|H(j\omega)|$ converges to infinite as $\omega \to 0$, we must have $r > 0$ the rate of increase determines the value of r:

$$
\begin{array}{lllll}
20 \text{ dB/decade} & \Rightarrow & r=1 & \Rightarrow & \begin{cases} \angle H(j\omega)_{\omega\to 0} = -\frac{\pi}{2} & k>0 \\ \angle H(j\omega)_{\omega\to 0} = -\frac{3\pi}{2} & k<0 \end{cases} \\
40 \text{ dB/decade} & \Rightarrow & r=2 & \Rightarrow & \begin{cases} \angle H(j\omega)_{\omega\to 0} = -\frac{5\pi}{2} & k>0 \\ \angle H(j\omega)_{\omega\to 0} = -\frac{7\pi}{2} & k<0 \end{cases} \\
& & \vdots & &
\end{array}
$$

Resolving the phase ambiguity for frequencies $\omega > 0$, can then be done by using the fact that the phase of a transfer function $H(j\omega)$ should be continuous with respect to ω. □

MATLAB® Hint 10. `phaseOut=unwrap(phaseIn)` takes an array of phases and adds/subtracts multiples of 2π to each element of the array to make the array "as smooth as possibles." The `unwrap` command does not take into account the transfer function magnitude so the initial phase still need to be "fixed" by using the approach outlined in the Note 11. ► p. 28

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

(i) To estimate $H(j\omega)$ at a single frequency, one still needs a long input (i.e., K 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 may prevent the use of control design methods that are not directly based on the process' Bode plot.

Note. For simple systems, it may be possible to estimate the locations of poles and zeros, directly from the Bode plot. This permits the use of many other control design methods.

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 by rational. □

2.5 Discrete-time Frequency Response Identification

We now consider methods to solve the discrete-time frequency response identification Problem 2.2.

2.5.1 Sine-wave Testing

Suppose that one applies an sinusoidal input of the form

$$u(k) = \alpha \cos(\Omega k), \qquad \forall k \in \{1, 2, \ldots, K\}. \tag{2.4}$$

Note 4. Discrete-time angular frequencies Ω take values from 0 to π. 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 $u_c(t) = A\cos(\omega t)$, $\forall t \geqslant 0$, we should select $\Omega = \omega T_s = 2\pi f T_s$. ► p. 15

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

$$y(k) = \alpha A_\Omega \cos\left(\Omega k + \phi_\Omega\right) + \varepsilon(k) + n(k), \tag{2.5}$$

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 γ is the magnitude of the pole of $H(z)$ with largest magnitude and c some constant; and

3. $n(k)$ corresponds to measurement noise.

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

When $n(k)$ is much smaller than αA_Ω and for k sufficiently large so that $\varepsilon(k)$ is negligible, we have

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

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 (2.4) with distinct frequencies Ω, 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.

2.5.2 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 solving this task, thus improving the accuracy of frequency response identification with sine-wave testing.

Suppose that the input $u(k)$ in (2.4) was applied, resulting in the measured output $y(k)$ given by (2.5) was measured at K times $k \in \{0, 1, \dots, K-1\}$. In the correlation method we compute

$$Y_\Omega := \frac{1}{K}\sum_{k=0}^{K-1} e^{-j\Omega k} y(k) = \frac{1}{K}\sum_{k=0}^{K-1}\Big(\cos(\Omega k) - j\sin(\Omega k)\Big) y(k).$$

Note 12. Why? ► p. 28

Using the expression for $y(k)$ in (2.5), we conclude after fairly straightforward algebraic manipulations that

$$Y_\Omega = \frac{\alpha}{2}H(e^{j\Omega}) + \frac{\alpha}{2K}H^*(e^{j\Omega})\frac{1-e^{-2j\Omega K}}{1-e^{-2j\Omega}} + \frac{1}{K}\sum_{k=0}^{K-1} e^{-j\Omega k}\big(\varepsilon(k) + n(k)\big).$$

Note. Since the process is stable, $\varepsilon(k)$ converges to zero and therefore it has no pure sinusoids. However, a periodic noise term with frequency exactly equal to Ω would lead to trouble. ► p. 24

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

$$Y_\Omega \to \frac{\alpha}{2}H(e^{j\Omega}) + \lim_{K\to\infty}\frac{1}{K}\big(E(e^{-j\Omega}) + N(e^{-j\Omega})\big),$$

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 Ω, the z-transforms are finite and therefore the limit is equal to zero. This leads to the following estimate for $H(e^{j\Omega})$

Note. Y_Ω is a complex number so we get estimates for the amplitude and phase of $H(e^{j\Omega})$. ► p. 24

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

which is accurate for large K.

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., K 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 prevent the use of control design methods that are not directly based on the process' Bode plot.

Note. For simple systems, it may be possible to estimate the locations of poles and zeros, directly from the Bode plot. This permits the use of many other control design methods.

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 by rational. □

2.6 MATLAB® Hints

MATLAB® Hint 8 (`fft`)**.** The command `fft(h)` computes the fast-Fourier-transform (FFT) of the signal $h(k)$, $k \in \{1,2,\ldots,K\}$, which provides the values of $H(e^{j\Omega})$ for equally spaced frequencies $\Omega = \frac{2\pi k}{K}$, $k \in \{0,1,\ldots,K-1\}$. However, since we only care for values of Ω in the interval $[0,\pi]$, we should discard the second half of `fft`'s output. □

MATLAB® Hint 9 (`impulseest`)**.** The command `impulseest` 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:

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

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

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

   ```
   dat1=iddata(y1,u1,Ts);
   dat2=iddata(y2,u2,Ts);
   data=merge(dat1,dat2);
   ```

2. Compute the estimated discrete-time impulse response using

   ```
   model=impulseest(data,N);
   ```

   where `N` is the length of the impulse response and `model` is a MATLAB® object with the result of the identification.

   The impulse response is given by

   ```
   impulse(model)
   ```

   and the estimated discrete-time transfer function can be recovered using

   ```
   sysd=tf(model);
   ```

   Additional information about the result of the system identification can be obtained in the structure `model.report`. Some of the most useful items in this structure include

   - `model.report.Parameters.ParVector` is a vector with all the parameters that have been estimated.
   - `model.report.Parameters.FreeParCovariance` is the error covariance matrix for the estimated parameters. The diagonal elements of this matrix are the variances of the estimation errors of the parameters and their square roots are the corresponding standard deviations.

     One should compare the value of each parameter with its standard deviation. A large value in one or more of the standard deviations points to little confidence on the estimated value of that parameter.
   - `model.report.Fit.MSE` is the mean-square estimation error, which measure of how well the response of the estimated model fits the estimation data. □

MATLAB® Hint 11. As of MATLAB® version R2018b, there is no command to directly estimate the continuous-time impulse response, but one can obtain the discrete-time response and subsequently convert it to continuous time.

MATLAB® Hint 12. MATLAB® is an object oriented language and `model.report` is the property `report` of the object `model`.

Note. The `unwrap` command does not take into account the transfer function magnitude so the initial phase still need to be "fixed" by using the approach outlined in the Note 11. ► p. 24

MATLAB® Hint 10 (`unwrap`)**.** The command `phaseOut=unwrap(phaseIn)` takes an array of phases and adds/subtracts multiples of 2π to each element of the array to minimize the difference between consecutive phases in the array, i.e., to make the array "as smooth as possibles." □

2.7 To Probe Further

Note 8 (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 σ, we obtain

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

Note 9 (Continuous-time correlation method)**.** In the continuous-time correlation method one computes

$$Y_\omega := \frac{1}{K}\sum_{k=0}^{K-1} e^{-j\Omega k} y(T_s k) = \frac{1}{K}\sum_{k=0}^{K-1}\big(\cos(\Omega k) - j\sin(\Omega k)\big) y(T_s k), \qquad \Omega := \omega T_s.$$

Using the expression for $y(T_s k)$ in (2.2), we conclude that

$$Y_\omega = \frac{\alpha A_\omega}{K}\sum_{k=0}^{K-1}\Big(\cos(\Omega k)\cos(\Omega k+\phi_\omega) - j\sin(\Omega k)\cos(\Omega k+\phi_\omega)\Big) + \frac{1}{K}\sum_{k=0}^{K-1} e^{-j\Omega k}\big(\varepsilon(T_s k)+n(T_s k)\big).$$

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

$$\begin{aligned}Y_\omega &= \frac{\alpha A_\omega}{2K}\sum_{k=0}^{K-1}\Big(\cos\phi_\omega + \cos\big(2\Omega k+\phi_\omega\big) + j\sin\phi_\omega - j\sin\big(2\Omega k+\phi_\omega\big)\Big) + \frac{1}{K}\sum_{k=0}^{K-1} e^{j\Omega k}\big(\varepsilon(T_s k)+n(T_s k)\big)\\ &= \frac{\alpha A_\omega}{2K}\sum_{k=0}^{K-1} e^{j\phi_\omega} + \frac{\alpha A_\omega}{2K}\sum_{k=0}^{K-1} e^{-2j\Omega k - j\phi_\omega} + \frac{1}{K}\sum_{k=0}^{K-1} e^{-j\Omega k}\big(\varepsilon(T_s k)+n(T_s k)\big)\\ &= \frac{\alpha}{2}H(j\omega) + \frac{\alpha}{2K}H^*(j\omega)\sum_{k=0}^{K-1} e^{-2j\Omega k} + \frac{1}{K}\sum_{k=0}^{K-1} e^{-j\Omega k}\big(\varepsilon(T_s k)+n(T_s k)\big),\end{aligned}$$

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

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

$$\sum_{k=0}^{K-1} r^k = \frac{1-r^K}{1-r}$$

.

$$Y_\omega = \frac{\alpha}{2}H(j\omega) + \frac{\alpha}{2K}H^*(j\omega)\frac{1-e^{-2j\Omega K}}{1-e^{-2j\Omega}} + \frac{1}{K}\sum_{k=0}^{K-1} e^{-j\Omega k}\big(\varepsilon(T_s k)+n(T_s k)\big)$$ □

Note 12 (Discrete-time correlation method)**.** In the discrete-time correlation method one computes

$$Y_\Omega := \frac{1}{K}\sum_{k=0}^{K-1} e^{-j\Omega k} y(k) = \frac{1}{K}\sum_{k=0}^{K-1}\big(\cos(\Omega k) - j\sin(\Omega k)\big) y(k)$$

Using the expression for $y(k)$ in (2.5), we conclude that

$$Y_\Omega = \frac{\alpha A_\Omega}{K}\sum_{k=0}^{K-1}\Big(\cos(\Omega k)\cos(\Omega k+\phi_\Omega) - j\sin(\Omega k)\cos(\Omega k+\phi_\Omega)\Big) + \frac{1}{K}\sum_{k=0}^{K-1} e^{-j\Omega k}\big(\varepsilon(k)+n(k)\big).$$

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

$$\begin{aligned} Y_\Omega &= \frac{\alpha A_\Omega}{2K}\sum_{k=0}^{K-1}\Big(\cos\phi_\Omega + \cos\left(2\Omega k+\phi_\Omega\right) + j\sin\phi_\Omega - j\sin\left(2\Omega k+\phi_\Omega\right)\Big) + \frac{1}{K}\sum_{k=0}^{K-1} e^{j\Omega k}\big(\varepsilon(k)+n(k)\big) \\ &= \frac{\alpha A_\Omega}{2K}\sum_{k=0}^{K-1} e^{j\phi_\Omega} + \frac{\alpha A_\Omega}{2K}\sum_{k=0}^{K-1} e^{-2j\Omega k - j\phi_\Omega} + \frac{1}{K}\sum_{k=0}^{K-1} e^{-j\Omega k}\big(\varepsilon(k)+n(k)\big) \\ &= \frac{\alpha}{2}H(e^{j\Omega}) + \frac{\alpha}{2K}H^*(e^{j\Omega})\sum_{k=0}^{K-1} e^{-2j\Omega k} + \frac{1}{K}\sum_{k=0}^{K-1} e^{-j\Omega k}\big(\varepsilon(k)+n(k)\big), \end{aligned}$$

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

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

$$\sum_{k=0}^{K-1} r^k = \frac{1-r^K}{1-r}$$

.

$$Y_\Omega = \frac{\alpha}{2}H(e^{j\Omega}) + \frac{\alpha}{2K}H^*(e^{j\Omega})\frac{1-e^{-2j\Omega K}}{1-e^{-2j\Omega}} + \frac{1}{K}\sum_{k=0}^{K-1} e^{-j\Omega k}\big(\varepsilon(k)+n(k)\big) \qquad \square$$

2.8 Exercises

2.1 (Impulse response)**.** A Simulink block that models a pendulum with viscous friction is provided. This Simulink model expects some variables to be defined. Please use:

```
Ts = 0.01;
tfinal = 300;
noise = 1e-5;
noiseOn = 0;
```

For part 2 of this problem, you must set `noiseOn = 1` to turn on the noise.

You will also need to define values for the height of the impulse and the height of the step through the variables:

```
alpha_step
alpha_impulse
```

For these variable, you select the appropriate values.

Once all these variables have been set, you can run a simulation using

```
sim('pendulum',tfinal)
```

after which the variables `t`, `u`, and `y` are created with time, the control input, and the measured output.

1. Use the Simulink block (without noise, i.e., with `noiseOn = 0`) to estimate the system's impulse response $\hat{h}(k)$ using the impulse response method in Section 2.3.1 and the step response method in Section 2.3.2.

   What is the largest value α 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 responses that you obtain from the two methods.

   This system has a very long time response, which `impulseest` does not handle very well so you should probably use directly the formulas in Sections 2.3.1 and 2.3.2.

2. Turn on the noise in the Simulink block (with `noiseOn = 1`) and repeat the identification procedure above for a range of values for α. For both methods, plot the error $\sum_k \|\hat{h}_\alpha(k) - h(k)\|$ vs. α, where $\hat{h}_\alpha(k)$ denotes the estimate obtained for an input with magnitude α and $h(k)$ the actual impulse response. What is your conclusion?

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

2.2 (Correlation method in continuous time)**.** Modify the Simulink block provided for Exercise 2.1 to accept a sinusoidal input and use the following parameters:

```
Ts = 0.01;
tfinal = 300;
noise = 1e-4;
noiseOn = 0;
```

Note the noise level larger than that used in Exercise 2.1.

1. Estimate the system's frequency response $\widehat{H(j\omega)}$ at a representative set of (log-spaced) frequencies ω_i using the correlation method in Section 2.4.2.

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

   Important: Write MATLAB® scripts to automate the procedure of taking as inputs the simulation data $u(t)$, $y(t)$ (from a set of `.mat` files in a given folder) and producing the Bode plot of the system.

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

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

3. Compare your best frequency response estimate $\widehat{H(j\omega)}$, 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 2.1.

   Note. See Note 4 on how to translate discrete-time to continuous-time frequencies... ► p. 15

   Hint: Use the MATLAB® 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 $\Omega > \pi$). □

2.3 (Correlation method in discrete-time)**.** Modify the Simulink block provided for Exercise 2.1 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 Ω_i using the correlation method.

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

   Important: Write MATLAB® scripts to automate the procedure of taking as inputs the simulation data $u(k)$, $y(k)$ (from a set of `.mat` files in a given folder) and producing the Bode plot of the system.

2. Turn on the noise in the Simulink block and repeat the identification procedure above for a range of values for α. Plot the error $\sum_i \|\widehat{H_\alpha(e^{j\Omega_i})} - H(e^{j\Omega_i})\|$ vs. α, where $\widehat{H_\alpha(e^{j\Omega_i})}$ denotes the estimate obtained for an input with magnitude α. 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 2.1.

   Hint: Use the MATLAB® 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 $\Omega > \pi$).

Note. See Note 4 on how to translate discrete-time to continuous-time frequencies. . .

► p. 15

Lecture 3

Parametric Identification using Least-Squares

This lecture presents the basic tools needed for parametric identification using the method of least-squares:

Contents

3.1 Parametric Identification

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

Problem 3.1 (Parametric continuous-time transfer function identification)**.** Determine the coefficients α_i and β_i of the system's *rational continuous-time transfer function*

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

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

Note. The number of poles and zeros is often not know, which will be addressed in Section 5.3. ► p. 57

Problem 3.2 (Parametric discrete-time transfer function identification)**.** Determine the coefficients α_i and β_i of the system's *rational discrete-time 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.

Note. The number of poles and zeros is often not know, which will be addressed in Section 7.4. ► p. 83

This can be done using the *method of least-squares*, which we introduce next, starting from a simple line fitting problem and eventually building up to the Problems 3.1 and 3.2 in Lectures 4 and 6, respectively.

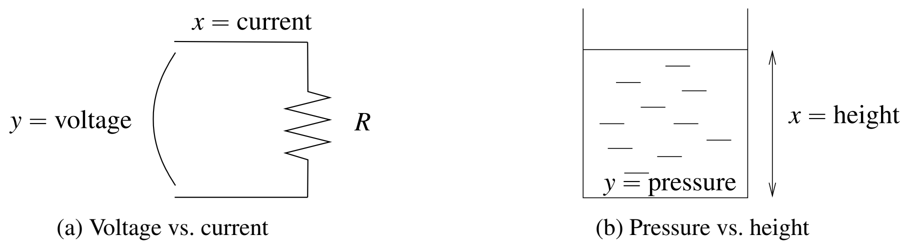
The methods of least-squares is generally attributed to Gauss [6] but the American mathematician Adrain [1] seems to have arrived at the same results independently [7]. A detailed treatment of least-squares estimation can be found, e.g., in [10, Chapter 7].

3.2 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

$$y = ax + b, \tag{3.1}$$

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 3.1).



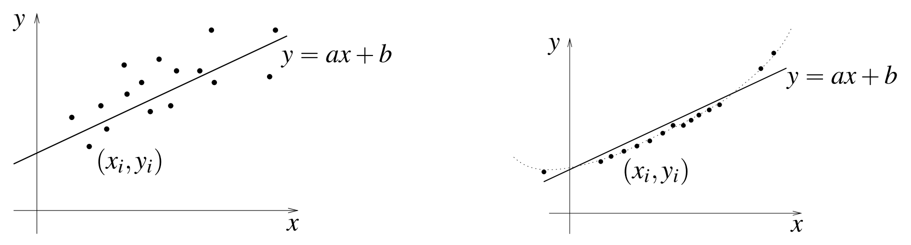
(a) Voltage vs. current (b) Pressure vs. height

Figure 3.1. Linear models

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 ith experiment of a total of N experiments. As illustrated in Figure 3.2, it is unlikely that we have exactly

$$y_i = ax_i + b, \qquad \forall i \in \{1, 2, \dots, N\}, \tag{3.2}$$

because of measurement errors and also because often the model (3.1) is only approximate.



(a) Measurement noise (b) Nonlinearity

Figure 3.2. Main causes for discrepancy between experimental data and linear model

Least-squares identification attempts to find values for a and b for which the left- and the right-hand-sides of (3.2) *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:

Problem 3.3 (Basic Least-Squares)**.** Find values for a and b that minimize the following Mean-Square Error (MSE):

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

Solution to Problem 3.3. This problem can be solved using standard calculus. All we need to do is solve

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

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

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

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. □

Note 13. Statistical interpretation of least-squares. ► p. 38

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

$$N\big(\sum_i x_i^2\big) - \big(\sum_i x_i\big)^2 = 0, \tag{3.3}$$

because this would lead to a division by zero. It turns out that (3.3) only holds when the x_i obtained in *all experiments* are exactly the same, as shown in Figure 3.3. Such set of measurements does not provide enough information to estimate the parameter a and b that characterize the line defined by (3.1). □

Note 14. Why does (3.3) only hold when all the x_i are equal to each other? ► p. 38

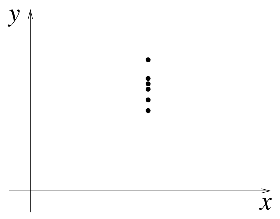


Figure 3.3. Singularity in least-squares line fitting due to poor data.

3.3 Vector Least-Squares

The least-square problem defined in Section 3.2 can be extended to a more general linear model, where *y is a scalar*, $z \coloneqq \begin{bmatrix} z_1 & z_2 & \cdots & z_m \end{bmatrix}$ *an m-vector*, and these variables are related by a linear equation of the form

Notation 2. $a \cdot b$ denotes the inner product of a and b. ► p. 37

$$y = \sum_{j=1}^{m} z_j \theta_j = z \cdot \theta, \tag{3.4}$$

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

where $\theta \coloneqq \begin{bmatrix} \theta_1 & \theta_2 & \cdots & \theta_m \end{bmatrix}$ is an m-vector with parameters whose values we would like to determine.

Attention! We are now using (i) to denote the ith experiment since the subscript in z_j is needed to denotes the jth entry of the vector z.

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

Problem 3.4 (Vector Least-Squares)**.** Find values for $\theta := \begin{bmatrix}\theta_1 & \theta_2 & \cdots & \theta_m\end{bmatrix}$ that minimize the following Mean-Square Error (MSE):

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

Solution to Problem 3.4. This problem can also be solved using standard calculus. Essentially we have to solve

$$\frac{\partial MSE}{\partial \theta_1} = \frac{\partial MSE}{\partial \theta_2} = \cdots = \frac{\partial MSE}{\partial \theta_m} = 0,$$

Notation 3. $\nabla_x f(x)$ denotes the gradient of $f(x)$. ► p. 37

for the unknown θ_i. The equation above can also be re-written as

$$\nabla_\theta MSE = \begin{bmatrix}\frac{\partial MSE}{\partial \theta_1} & \frac{\partial MSE}{\partial \theta_2} & \cdots & \frac{\partial MSE}{\partial \theta_m}\end{bmatrix} = 0.$$

Note. Vector notation is convenient both for algebraic manipulations and for efficient MATLAB® computations.

However, to simplify the computation it is convenient to write the MSE 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.,

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

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

we can write

$$MSE = \frac{1}{N}\|E\|^2 = \frac{1}{N}E'E.$$

Moreover, by viewing θ 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.,

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

Therefore

$$MSE = \frac{1}{N}(\theta' Z' - Y')(Z\theta - Y) = \frac{1}{N}\big(\theta' Z' Z\theta - 2Y' Z\theta + Y' Y\big). \tag{3.5}$$

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

$$\nabla_\theta MSE = \frac{1}{N}\big(2\theta' Z'Z - 2Y'Z\big) = 0 \quad \Leftrightarrow \quad \theta' = Y'Z(Z'Z)^{-1}, \tag{3.6}$$

Note 15. The gradient of a quadratic function $f(x) = x'Qx + cx + d$, is given by $\nabla_x f(x) = x'(Q+Q') + c$. ► p. 39

which yields the following *least-squares estimate* for θ:

$$\hat{\theta} = (Z'Z)^{-1}Z'Y. \tag{3.7}$$

MATLAB® Hint 13. Z\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 (3.6). This is desirable, because this operation is often ill-conditioned.

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

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

Quality of Fit: One should check the *quality of fit* by computing the Mean-Square Error MSE achieved by the estimate (i.e., the minimum achievable squared-error), normalized by the Mean-Square Output MSO. Replacing the estimate (3.7) in (3.5) we obtain

$$\frac{MSE}{MSO} = \frac{\frac{1}{N}\|Z\hat{\theta} - Y\|^2}{\frac{1}{N}\|Y\|^2} \tag{3.8}$$

When this *error-to-signal ratio* is much smaller than one, the mismatch between the left- and right-hand-sides of (3.4) has been made significantly smaller than the output.

Identifiability: The above estimate for θ is not valid when the $m \times m$ matrix R is not invertible. Singularity of R is a situation similar to (3.3) in line fitting and it means that the $z(i)$ are not sufficiently rich to estimate θ. In practice, even when R is invertible, poor estimates are obtained if R is close to a singular matrix, in which case we say that the model is not *identifiable* for the given measurements. One should check identifiability by computing the following estimate for the covariance of the estimation error:

Note. This estimate for the error covariance will be small if there is a good fit in the sense that $\|Z\hat{\theta} - Y\|$ is small and the data is sufficiently "rich" to make $(Z'Z)^{-1}$ also small.

$$E[(\theta - \hat{\theta})(\theta - \hat{\theta})'] \approx \frac{N}{N-m} MSE(Z'Z)^{-1} = \frac{1}{N-m}\|Z\hat{\theta} - Y\|^2 (Z'Z)^{-1}. \tag{3.9}$$

The diagonal elements of this matrix are the variances of the estimation errors of the parameter and their square roots are the corresponding standard deviations. A large value for one of these standard deviations (compared to the value of the parameter) means that the estimate of the parameter is likely inaccurate. □

3.4 To Probe Further

Notation 2 (Inner product)**.** Given two m-vectors, $a = \begin{bmatrix} a_1 & a_2 & \cdots & a_m \end{bmatrix}$ and $a = \begin{bmatrix} a_1 & a_2 & \cdots & a_m \end{bmatrix}$, $a \cdot b$ denotes the inner product of a and b, i.e.,

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

□

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

$$\nabla_x f(x_1, x_2, \ldots, x_m) = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_m} \end{bmatrix}.$$ □

Note 13 (Statistical interpretation of least-squares)**.** Suppose that the mismatch between left- and the right-hand sides of (3.2) is due to uncorrelated zero-mean noise. In particular, that

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

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

$$\mathrm{E}[n_i] = 0, \qquad \mathrm{E}[n_i^2] = \sigma^2, \qquad \mathrm{E}[n_i n_j] = 0, \forall i, j \neq i.$$

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

Theorem 3.1 (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 3.1 compares all linear estimators, i.e., all estimators of the form

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

   where the α_i, β_i are coefficients that may depend on the x_i.

2. It then asks what is the choice for the α_i, β_i that lead to an estimator that is "best" in the sense that it is (i) unbiased, i.e., for which

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

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

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

   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. □

Note 14 (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 μ. □

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

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

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

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

□

3.5 Exercises

3.1 (Electrical circuit)**.** Consider the electrical circuit in Figure 3.4, 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

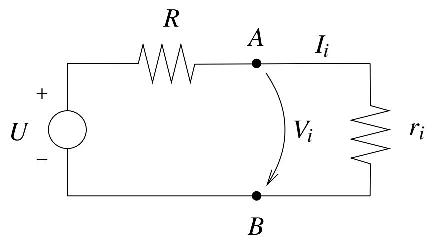


Figure 3.4. Electrical circuit

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

1. Write a MATLAB® 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? □

3.2 (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. $\square$

Lecture 4

Parametric Identification of a Continuous-Time ARX Model

This lecture explains how the methods of least-squares can be used to identify the transfer function of a continuous-time system.

Contents

4.1 CARX Model

Suppose that we want to determine the transfer function $H(s)$ of the SISO continuous-time system in Figure 6.1. In least-squares identification, one converts the problem of estimating $H(s)$ into the

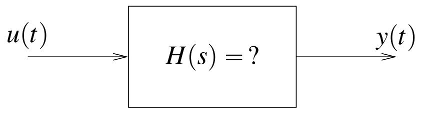


Figure 4.1. System identification from input/output experimental data

vector least-squares problem considered in Section 3.3. This is done using the CARX model that will be constructed below.

The Laplace transforms of the input and output of the system in Figure 6.1 are related by

$$\frac{Y(s)}{U(s)} = H(s) = \frac{\alpha(s)}{\beta(s)} = \frac{\alpha_m s^m + \alpha_{m-1} s^{m-1} + \cdots + \alpha_1 s + \alpha_0}{s^n + \beta_{n-1} s^{n-1} + \cdots + \beta_1 s + \beta_0}, \tag{4.1}$$

where

$$\alpha(s) := \alpha_m s^m + \alpha_{m-1} s^{m-1} + \cdots + \alpha_1 s + \alpha_0, \qquad \beta(s) := s^n + \beta_{n-1} s^{n-1} + \cdots + \beta_1 s + \beta_0,$$

denote the numerator and denominator polynomials of the system's transfer function. The relationship between $Y(s)$ and $U(s)$ in (4.1) can be equivalently expressed as

$$\beta(s)Y(s) = \alpha(s)U(s) \quad \Leftrightarrow$$
$$s^n Y(s) + \beta_{n-1}s^{n-1}Y(s) + \cdots + \beta_1 sY(s) + \beta_0 Y(s) =$$
$$\alpha_m s^m U(s) + \alpha_{m-1}s^{m-1}U(s) + \cdots + \alpha_1 sU(s) + \alpha_0 U(s). \quad (4.2)$$

Taking inverse Laplace transforms we obtain the so-called *Continuous-time Auto-Regression model with eXogeneous inputs (CARX)* model:

Note 16. Equation (4.3) can be conveniently written using the notation

$$\beta\Big(\frac{d}{dt}\Big)y = \alpha\Big(\frac{d}{dt}\Big)u,$$

where $\alpha(s)$ and $\beta(s)$ are the numerator and denominator polynomials of the transfer function.

$$\frac{d^n y(t)}{dt^n} + \beta_{n-1}\frac{d^{n-1}y(t)}{dt^{n-1}} + \cdots + \beta_1\frac{dy(t)}{dt} + \beta_0 y(t) =$$
$$\alpha_m\frac{d^m u(t)}{dt^m} + \alpha_{m-1}\frac{d^{m-1}u(t)}{dt^{m-1}} + \cdots + \alpha_1\frac{du(t)}{dt} + \alpha_0 u(t), \quad \forall t \geqslant 0. \quad (4.3)$$

This can be re-written compactly as

$$\frac{d^n y(t)}{dt^n} = \varphi(t)\cdot\theta, \quad \forall t \geqslant 0. \quad (4.4)$$

where the $(n+m+1)$-vector θ contains the coefficient of the transfer function and the vector $\varphi(t)$ is called the *regression vector* and includes the derivatives of the inputs and outputs, i.e.,

$$\theta := \begin{bmatrix}\alpha_m & \alpha_{m-1} & \cdots & \alpha_1 & \alpha_0 & \beta_{n-1} & \cdots & \beta_1 & \beta_0\end{bmatrix}$$
$$\varphi(t) := \begin{bmatrix}\frac{d^m u(t)}{dt^m} & \frac{d^{m-1}u(t)}{dt^{m-1}} & \cdots & u(t) & -\frac{d^{n-1}y(t)}{dt^{n-1}} & -\frac{d^{n-2}y(t)}{dt^{n-2}} & \cdots & -y(t)\end{bmatrix}.$$

4.2 Identification of a CARX Model

We are now ready to propose a tentative solution for the system identification Problem 3.1 introduced in Lecture 3, by applying the least-squares method to the CARX model:

Solution to Problem 3.1 (tentative).

1. Apply a probe input signal $u(t)$, $t \in [0,T]$ to the system.
2. Measure the corresponding output $y(t)$ at a set of times $t_0, t_1, \ldots, t_N \in [0,T]$.
3. Compute the first m derivatives of $u(t)$ and the first n derivatives of $y(t)$ at the times $t_0, t_1, \ldots, t_N \in [0,T]$.
4. Determine the values for the parameter θ that minimize the discrepancy between the left- and the right-hand-sides of (4.4) in a least-squares sense at the times $t_0, t_1, \ldots, t_N \in [0,T]$.

   According to Section 3.3, the least-squares estimate of θ is given by

MATLAB® Hint 14. PHI\Y computes $\hat\theta$ directly, from the matrix PHI$=\Phi$ and the vector Y$=Y$.

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

   where

$$\Phi := \begin{bmatrix}\varphi(t_0)\\ \varphi(t_1)\\ \vdots\\ \varphi(t_N)\end{bmatrix} = \begin{bmatrix}\frac{d^m u(t_0)}{dt^m} & \cdots & u(t_0) & -\frac{d^{n-1}y(t_0)}{dt^{n-1}} & \cdots & -y(t_0)\\ \frac{d^m u(t_1)}{dt^m} & \cdots & u(t_1) & -\frac{d^{n-1}y(t_1)}{dt^{n-1}} & \cdots & -y(t_1)\\ \vdots & & \vdots & \vdots & & \vdots\\ \frac{d^m u(t_N)}{dt^m} & \cdots & u(t_N) & -\frac{d^{n-1}y(t_N)}{dt^{n-1}} & \cdots & -y(t_N)\end{bmatrix}, \qquad Y := \begin{bmatrix}\frac{d^n y(t_0)}{dt^n}\\ \frac{d^n y(t_1)}{dt^n}\\ \vdots\\ \frac{d^n y(t_N)}{dt^n}\end{bmatrix}.$$

□

The problem with this approach is that very often the measurements of $y(t)$ have high frequency noise which will be greatly amplified by taking the n derivatives required to compute Y and the $n-1$ derivatives needed for Φ.

4.3 CARX Model with Filtered Data

Consider a polynomial

$$\omega(s) \coloneqq s^{\ell} + \omega_{\ell-1}s^{\ell-1} + \cdots + \omega_1 s + \omega_0$$

of order ℓ larger than or equal to the order n of the denominator of the transfer function in (4.1) that is "asymptotically stable" in the sense that all its roots have strictly negative real part and suppose that construct the "filtered" versions u_f and y_f of the input u and output y, respectively, using the transfer function $1/\omega(s)$:

Note. Since all $\omega(s)$ roots of $\omega(s)$ have strictly negative real part the transfer function $\frac{1}{\omega(s)}$ is asymptotically stable and therefore the effect of initial conditions will converge to zero.

$$Y_f(s) = \frac{1}{\omega(s)}Y(s), \qquad U_f(s) = \frac{1}{\omega(s)}U(s). \tag{4.5}$$

In view of these equations, we have that

$$Y(s) = \omega(s)Y_f(s), \qquad U(s) = \omega(s)U_f(s),$$

which can be combined with (4.2) to conclude that

$$\beta(s)\omega(s)Y_f(s) = \alpha(s)\omega(s)U_f(s) \quad \Leftrightarrow \quad \omega(s)\Big(\beta(s)Y_f(s) - \alpha(s)U_f(s)\Big) = 0.$$

Taking inverse Laplace transforms we now obtain

Note. To keep the formulas short, here we are using the notation introduced in Note 16. ► p. 42

$$\omega\Big(\frac{d}{dt}\Big)\Big(\beta\Big(\frac{d}{dt}\Big)y_f(t) - \alpha\Big(\frac{d}{dt}\Big)u_f(t)\Big) = 0.$$

Defining

$$e(t) \coloneqq \beta\Big(\frac{d}{dt}\Big)y_f(t) - \alpha\Big(\frac{d}{dt}\Big)u_f(t),$$

the above equation tell us that the "error" $e(t)$ is a solution to the differential equation

$$\omega\Big(\frac{d}{dt}\Big)e(t) = 0.$$

Moreover, since all roots of $\omega(s)$ have strictly negative real part, $e(t) \approx 0$ for sufficiently large t and therefore

$$\beta\Big(\frac{d}{dt}\Big)y_f(t) - \alpha\Big(\frac{d}{dt}\Big)u_f(t) \approx 0, \quad \forall t \geqslant T_f, \tag{4.6}$$

where T_f should be chosen sufficiently large so that the impulse response of $1/\omega(s)$ is negligible for $t \geqslant T_f$.

At this point, we got to an equation (4.6) that very much resembles the CARX model (4.3), except that it involves y_f and u_f instead of y and u and is only valid for $t \geqslant T_f$. As before, we can re-write (4.6) compactly as

$$\frac{d^n y_f(t)}{dt^n} = \varphi_f(t)\cdot\theta \quad \forall t \geqslant T_f, \tag{4.7}$$

where the $(n+m+1)$-vector θ contains the coefficient of the transfer function and the regressor vector $\varphi_f(t)$ includes the derivatives of the inputs and outputs, i.e.,

$$\theta \coloneqq \begin{bmatrix}\alpha_m & \alpha_{m-1} & \cdots & \alpha_1 & \alpha_0 & \beta_{n-1} & \cdots & \beta_1 & \beta_0\end{bmatrix} \tag{4.8}$$

$$\varphi_f(t) \coloneqq \begin{bmatrix}\frac{d^m u_f(t)}{dt^m} & \frac{d^{m-1}u_f(t)}{dt^{m-1}} & \cdots & u_f(t) & -\frac{d^{n-1}y_f(t)}{dt^{n-1}} & -\frac{d^{n-2}y_f(t)}{dt^{n-2}} & \cdots & -y_f(t)\end{bmatrix}. \tag{4.9}$$

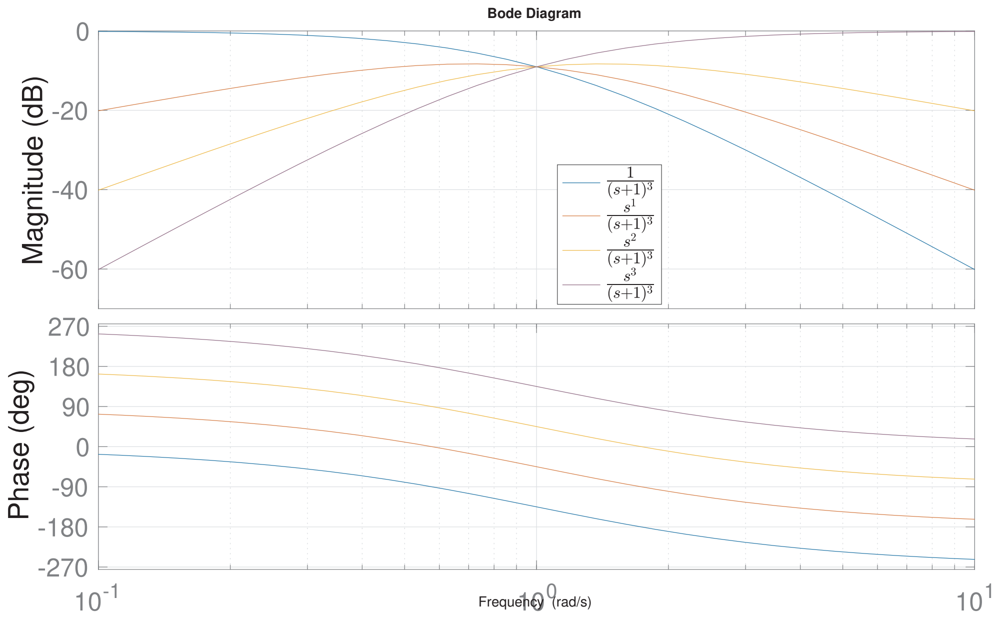


Figure 4.2. typical filter transfer functions used to generate $\frac{d^n y_f(t)}{dt^n}$ and the signals that appear in φ_f in the CARX Model with filtered signals in (4.7). In this figure we used $\omega(s) \coloneqq (s+1)^\ell$ and $\ell = n = 3$.

The key advantage of (4.7)–(4.6) over (4.4)–(4.3) and is that the latter does not require taking derivatives of y as it involves instead derivatives of y_f. In particular, computing the kth derivative of y_f, corresponds to filtering y with the following transfer function

$$\mathscr{L}\Big[\frac{d^k y_f(t)}{dt^k}\Big] = s^k Y_f(s) = \frac{s^k}{\omega(s)} Y(s)$$

which will not amplify high-frequency components of y as long as k is smaller than or equal to the order ℓ of ω, which explains why we selected $\ell \geqslant n$. Figure 4.2 shows typical transfer functions used to generate $\frac{d^n y_f(t)}{dt^n}$ and the signals that appear in φ_f in (4.7).

4.4 Identification of a CARX Model with Filtered Signals

We are now ready to propose a good solution for the system identification Problem 3.1 introduced in Lecture 3, by applying the least-squares method to the CARX model with filtered data:

Solution to Problem 3.1.

1. Apply a probe input signal $u(t)$, $t \in [0,T]$ to the system.
2. Measure the corresponding output $y(t)$ at a set of times $t_0, t_1, \dots, t_N \in [0,T]$.
3. Compute the first m derivatives of $u_f(t)$ and the first n derivatives of $y_f(t)$ at the times $t_0, t_1, \dots, t_N \in [0,T]$, using the fact that

$$\frac{d^k y_f(t)}{dt^k} = \mathscr{L}^{-1}\Big[\frac{s^k}{\omega(s)} Y(s)\Big], \qquad \frac{d^k u_f(t)}{dt^k} = \mathscr{L}^{-1}\Big[\frac{s^k}{\omega(s)} U(s)\Big].$$

4. Determine the values for the parameter θ that minimize the discrepancy between the left- and the right-hand-sides of (4.7) in a least-squares sense for those times $t_{k_o}, t_{k_0+1}, \dots, t_N$ that fall in $[T_f, T]$.

According to Section 3.3, the least-squares estimate of θ is given by

$$\hat{\theta} = (\Phi'\Phi)^{-1}\Phi' Y, \tag{4.10}$$

MATLAB® Hint 15. `PHI\Y` computes $\hat{\theta}$ directly, from the matrix `PHI`$= \Phi$ and the vector `Y`$= Y$.
Attention! Remember to through away the initial times that fall before T_f.

where

$$\Phi := \begin{bmatrix} \varphi_f(t_{k_0}) \\ \varphi_f(t_{k_0+1}) \\ \vdots \\ \varphi_f(t_N) \end{bmatrix} = \begin{bmatrix} \frac{d^m u_f(t_{k_0})}{dt^m} & \cdots & u_f(t_{k_0}) & -\frac{d^{n-1} y_f(t_{k_0})}{dt^{n-1}} & \cdots & -y_f(t_{k_0}) \\ \frac{d^m u_f(t_{k_0+1})}{dt^m} & \cdots & u_f(t_{k_0+1}) & -\frac{d^{n-1} y_f(t_{k_0+1})}{dt^{n-1}} & \cdots & -y_f(t_{k_0+1}) \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{d^m u_f(t_N)}{dt^m} & \cdots & u_f(t_N) & -\frac{d^{n-1} y_f(t_N)}{dt^{n-1}} & \cdots & -y_f(t_N) \end{bmatrix},$$

$$Y_F := \begin{bmatrix} \frac{d^n y_f(t_{k_0})}{dt^n} \\ \frac{d^n y_f(t_{k_0+1})}{dt^n} \\ \vdots \\ \frac{d^n y_f(t_N)}{dt^n} \end{bmatrix},$$

or equivalently

$$\hat{\theta} = R^{-1} f, \qquad R := \Phi'\Phi = \sum_{k=k_0}^{N} \varphi_f(t_k)' \varphi_f(t_k), \qquad f := \Phi' Y = \sum_{k=k_0}^{N} \varphi_f(t_k)' \frac{d^n y_f(t_k)}{dt^n}$$

MATLAB® Hint 16. `tfest` is convenient to perform CARX models identification because it does not require the construction of the matrix Φ and the vector Y. ► p. 46

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

$$\frac{MSE}{MSO} = \frac{\frac{1}{N}\|\Phi\hat{\theta} - Y\|^2}{\frac{1}{N}\|Y\|^2} \tag{4.11}$$

MATLAB® Hint 17. `compare` is useful to determine the quality of fit. ► p. 48

When this quantity is much smaller than one, the mismatch between the left- and right-hand-sides of (4.7) has been made significantly smaller than the output. The *covariance of the estimation error* can be computed using

$$E[(\theta - \hat{\theta})(\theta - \hat{\theta})'] \approx \frac{1}{(N - k_0 + 1) - (n + m + 1)} \|\Phi\hat{\theta} - Y\|^2 (\Phi'\Phi)^{-1}. \tag{4.12}$$

Note. Why? In (4.12) we simply used (3.9), taking into account that the number of rows of Φ is $N - k_0 + 1$ and the number of parameters to estimate in θ is $n + m + 1$. ► p. 37

When the square roots of the diagonal elements of this matrix are much smaller than the corresponding entries of $\hat{\theta}$, one can have confidence in the values $\hat{\theta}$.

4.5 Dealing with 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 $s = 0$; or
2. that the process has a differentiator, which corresponds to a continuous-time zero at $s = 0$.

In this case, it suffices to identify the remaining poles/zeros.

4.5.1 Known Pole

Suppose that it is known that the transfer function $H(s)$ has a pole at $s = \lambda$, i.e.,

$$H(s) = \frac{1}{s - \lambda} \bar{H}(s),$$

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

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

$$\frac{Y(s)}{U(s)} = \frac{1}{s-\lambda}\bar{H}(s) \quad \Leftrightarrow \quad \frac{(s-\lambda)Y(s)}{U(s)} = \bar{H}(s)$$

and therefore

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

where

$$\bar{Y}(s) \coloneqq (s-\lambda)Y(s) \quad \Rightarrow \quad \bar{y}(t) = \frac{dy(t)}{dt} - \lambda y(t). \tag{4.13}$$

Note. To compute the new output $\bar{y}(t)$ we need to differentiate $y(t)$, but in practice we do this in the filtered output y_f so we simply need to increase the order of $\omega(s)$ to make sure that we do not amplify high frequency noise.

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

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

4.5.2 Known Zero

MATLAB® Hint 18. One must be careful in constructing the vector $\bar{y}$ in (4.13) to be used by the function `tfest`. ► p. 48

Suppose now that it is known that the transfer function $H(s)$ has a zero at $s = \lambda$, i.e.,

$$H(s) = (s-\lambda)\bar{H}(s),$$

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

$$\frac{Y(s)}{U(s)} = (s-\lambda)\bar{H}(s) \quad \Leftrightarrow \quad \frac{Y(s)}{(s-\lambda)U(s)} = \bar{H}(s)$$

and therefore

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

where

$$\bar{U}(s) \coloneqq (s-\lambda)U(s) \quad \Rightarrow \quad \bar{u}(t) = \frac{du(t)}{dt} - \lambda u(t). \tag{4.14}$$

MATLAB® Hint 18. One must be careful in constructing the vector $\bar{u}$ in (4.14) to be used by the function `tfest`. ► p. 48

This means that we can directly estimate $\bar{H}(s)$ by computing $\bar{u}(t)$ prior to identification and then regarding this variable as the input, instead of $u(t)$.

Attention! To obtain the original transfer function $H(s)$, one needs to multiply the identified function $\bar{H}(s)$ by the term $(s-\lambda)$. □

4.6 MATLAB® Hints

MATLAB® Hint 16 (`tfest`)**.** The command `tfest` from the identification toolbox performs least-squares identification of CARX 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 `Ts` is the sampling interval.

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

```
data1=iddata(y1,u1,Ts);
data2=iddata(y2,u2,Ts);
data=merge(data1,data2);
```

2. Compute the estimated model using:

```
model=tfest(data,np,nz);
```

where `data` is the object with input/output data, `np` is the number of poles for the transfer function (i.e., the degree of the denominator), `nz` is the number of zeros (i.e., the degree of the numerator), and `model` a MATLAB® object with the result of the identification.

The estimated continuous-time transfer function can be recovered using the MATLAB® command

```
sysc=tf(model);
```

Additional information about the result of the system identification can be obtained in the structure `model.report`. Some of the most useful items in this structure include

- `model.report.Parameters.ParVector` is a vector $\hat{\theta}_{\text{all}}$ with all the parameters in the transfer function, starting with the numerator coefficients and followed by the denominator coefficients, as in (4.8). These coefficients also appear in the transfer function `tf(model)`.

Note. There is a mismatch between `ParVector` and `FreeParCovariance` in that the former includes *all* parameters of the transfer function, whereas the latter only includes parameters that need to be estimated. The mismatch arises because the leading coefficient of the denominator is always the number 1 and does not need to be estimated. So, e.g., in a SISO system, `ParVector` includes one parameter that is not included in `FreeParCovariance`.

- `model.report.Parameters.FreeParCovariance` is the error covariance matrix

$$E[(\theta_{\text{est}} - \hat{\theta}_{\text{est}})(\theta_{\text{est}} - \hat{\theta}_{\text{est}})']$$

for the estimated parameters. The diagonal elements of this matrix are the variances of the estimation errors of the parameter and their square roots are the corresponding standard deviations

$$\sqrt{E[(\theta_i - \hat{\theta}_i)^2]},$$

which can be obtained using

```
StdDev=sqrt(diag(model.report.parameters.FreeParCovariance));
```

One 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.

Note 17. Obtaining a standard deviation for one parameter that is comparable or larger than its estimated value typically arises in one of three situations. ► p. 48

- `model.report.Fit.MSE` is the mean-square estimation error (MSE), as in the numerator of (4.11), which measure of how well the response of the estimated model fits the estimation data. The MSE should be normalized by the Mean-Square Output

```
MSO=y'*y
```

when one want to compare the fit across different inputs/outputs.

The MATLAB® command `tfest` uses a more sophisticated algorithm than the one described in these notes so the results obtained using `tfest` may not match exactly the ones obtained using the formula (4.10). In particular, it automatically tries to adjust the polynomial $\omega(s)$ used to obtain the filtered input and output signals in (4.5). While this typically improves the quality of the estimate, it occasionally leads to poor results so one needs to exercise caution in evaluating the output of `tfest`.

□

MATLAB® Hint 17 (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® hints 9, 16, 23 for details on these objects. □

MATLAB® Hint 18 (Dealing with known parameters)**.** Assuming that the output $y(t)$ has been sampled with a sampling interval T_s, a first-order approximation to the vector $\bar{y}$ in (4.13) can be obtained with the following MATLAB® command

```
bary = (y(2:end)-y(1:end-1))/Ts-lambda*y(1:end-1),
```

where we used finite differences to approximate the derivative. However, this vector `bary` has one less element than the original `y`. Since the function `iddata` only takes input/output pairs of the same length, this means that we need to discard the last input data in `u`:

```
data=iddata(bary,u(1:end-1),Ts);
```

To construct the data object corresponding the case of a known zero in (4.14), we would use instead

```
baru = (u(2:end)-u(1:end-1))/Ts-lambda*u(1:end-1),
data=iddata(y(1:end-1),baru,Ts);
```

□

4.7 To Probe Further

Note 17 (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 5.1. 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 or the number of zeros. This issue is discussed in detail in Section 5.3. It can generally be resolved by decreasing the number of poles to be estimates (i.e., the order of the denominator polynomial) or the number of zeros (i.e., the order of the numerator polynomial).

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 is 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 Lectures 8–9. □

4.8 Exercises

4.1 (Known parameters)**.** The data provided was obtained from a system with one zero and two poles transfer function

$$H(s) = \frac{k(s+.5)}{(s+.3)(s-p)},$$

where the gain k and the pole p are unknown.

1. Estimate the system's transfer function *without making use of the fact that you know the location of the zero and one of the poles.*

2. Estimate the system's transfer function using the fact that you know that the zero is at $s = -.5$ and one of the poles at $s = -.3$.

   Hint: To solve this problem you will need to combine the ideas from Sections 4.5.1 and 4.5.2, by identifying the transfer function from an auxiliar input $\bar{u}$ to an auxiliar output $\bar{y}$. $\square$

If all went well, you should have gotten somewhat similar bode plots for the transfer functions obtained in 1 and 2, but you should have obtained a much better estimate of the poles and zeros in 2.

Lecture 5

Practical Considerations in Identification of Continuous-time CARX Models

This lecture discusses several practical considerations that are crucial for successful identifications.

Contents

5.1 Choice of Inputs

The quality of least-squares estimates depends significantly on the input $u(t)$ 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 ω will not allow us to distinguish between different transfer functions with exactly the same value at the point $s = j\omega$, as illustrated in Figure 5.1.

   (b) Good inputs include:

      i. square waves, which includes a base frequency and several high-order harmonics;

      ii. a combination of many sinusoids, either summed, one after another, or by merging multiple experiments;

      iii. a chirp signal (i.e., a signal with time-varying frequency).

      Combining multiple sinusoids is typically better than using a chirp signal since one has better control on the duration of the input signals at the different frequencies. Ideally one would like each frequency to have multiple periods, but a linear chirp signal provides equal *time* for each frequency, rather than the same number of periods.

Note. See Section 5.4 on combining data from multiple experiments. ► p. 60

Note. It is a good idea to log-space the frequencies of multiple sinusoids like in a Bode plot.

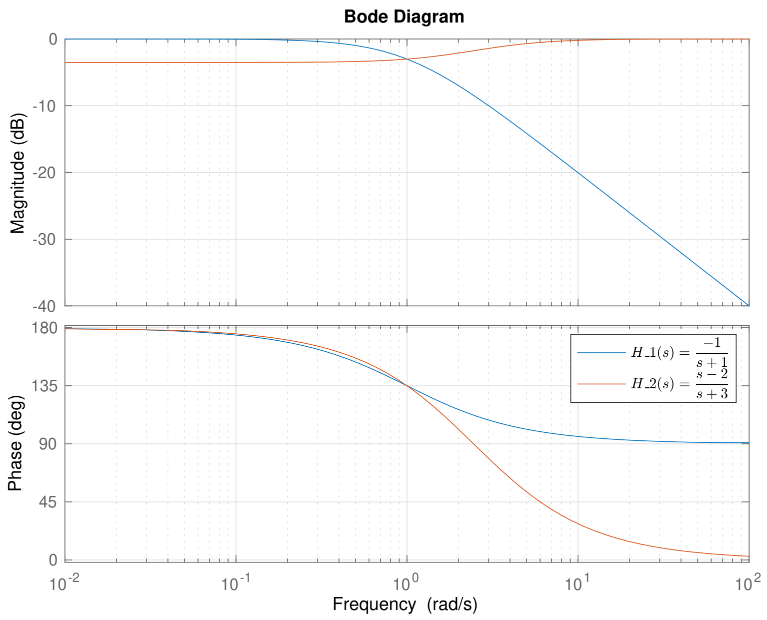


Figure 5.1. The transfer functions $H_1(s) = \frac{-1}{s+1}$ and $H_2(s) = \frac{s-2}{s+3}$ have the same value for $s = j$. Therefore they will lead to the same output $y(t)$ when $u(t)$ is a sinusoid with frequency $\omega = 1$ rad/sec. This input will not allow us to distinguish between H_1 and H_2.

2. The *amplitude of the resulting output* $y(t)$ 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 5.2, there is usually an "optimal" input level that needs to be determined by trial-and-error. □

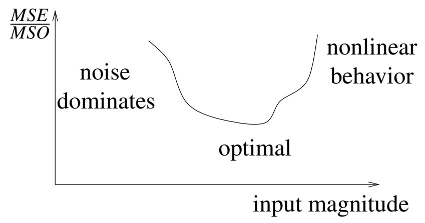


Figure 5.2. Optimal choice of input magnitude

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.

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. Repeat the identification experiment with the input $\alpha u(t)$ with $\alpha = 1$, $\alpha = -1$, and $\alpha = .5$. If the process is in the linear region, the measured outputs should roughly by equal to $\alpha y(t)$ and all 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.

2. Check if the matrix R is far from singular. If not a different "richer" input should be used.

3. Check the quality of fit by computing $\frac{MSE}{MSO}$. 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 5.3).

MATLAB® Hint 16. When using the `tfest` command, the singularity of the matrix R can be inferred from the standard deviations associated with the parameter estimates. ► p. 46

MATLAB® Hint 16. When using the `tfest` command, the quality of fit can be inferred from `model.report.Fit.MSE`, but it should generally be normalized by the Mean-Square Output $MSO = \frac{1}{N}\|Y\|^2$. ► p. 46

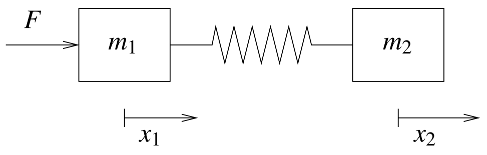


Figure 5.3. Two-mass with spring

Example 5.1 (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 5.3. From Newton's law one concludes that

$$m_1\ddot{x}_1 = k(x_2 - x_1) + f, \qquad m_2\ddot{x}_2 = k(x_1 - x_2), \tag{5.1}$$

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

$$f = \frac{K_m K_g}{R_m r}\Big(v - \frac{K_m K_g}{r}\dot{x}_1\Big), \tag{5.2}$$

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 function, we replace f from (5.2) into (5.1) and take Laplace transforms to conclude that

$$\begin{cases} m_1 s^2 X_1(s) = k\big(Y(s) - X_1(s)\big) + \frac{K_m K_g}{R_m r}\Big(U(s) - \frac{K_m K_g}{r} s X_1(s)\Big) \\ m_2 s^2 Y(s) = k\big(X_1(s) - Y(s)\big) \end{cases}$$

$$\Rightarrow \quad \begin{cases} \Big(m_1 s^2 + \frac{K_m^2 K_g^2}{R_m r^2} s + k\Big) X_1(s) = kY(s) + \frac{K_m K_g}{R_m r} U(s) \\ \big(m_2 s^2 + k\big) Y(s) = k X_1(s) \end{cases}$$

$$\Rightarrow \quad \big(m_2 s^2 + k\big)\Big(m_1 s^2 + \frac{K_m^2 K_g^2}{R_m r^2} s + k\Big) Y(s) = k^2 Y(s) + \frac{k K_m K_g}{R_m r} U(s)$$

$$\frac{Y(s)}{U(s)} = \frac{\frac{k K_m K_g}{R_m r}}{\big(m_2 s^2 + k\big)\Big(m_1 s^2 + \frac{K_m^2 K_g^2}{R_m r^2} s + k\Big) - k^2} \tag{5.3}$$

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 (5.3) 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 (5.3) 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

Note. Friction is especially hard to model from first principles. Often one models friction using viscous friction (i.e., proportional to the velocity), but this is just an empirical law so the friction coefficient needs to be determined experimentally.

2. the first-principles model (5.1)–(5.2) 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 5.4a shows four sets of input/output data and four continuous-time transfer functions identified from these data sets.

Key observations. A careful examination of the outputs of the `tfest` command and the plots in Figure 5.4 reveals the following:

1. While difficult to see in Figure 5.4a, 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 5.4a because of scaling may actually be an indication of trouble*.

2. The plots in Figure 5.4b show that the four sets of data input/output data resulted in *dramatically different identified transfer functions*. □

5.2 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

MATLAB® Hint 19. In MATLAB®, one could do:
```
data=iddata(a*y,b*u);
model=tfest(data,...);
H=b/a*tf(model);
```

$$\bar{u}(t) = bu(t), \qquad \bar{y}(t) = ay(t), \qquad \forall t.$$

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

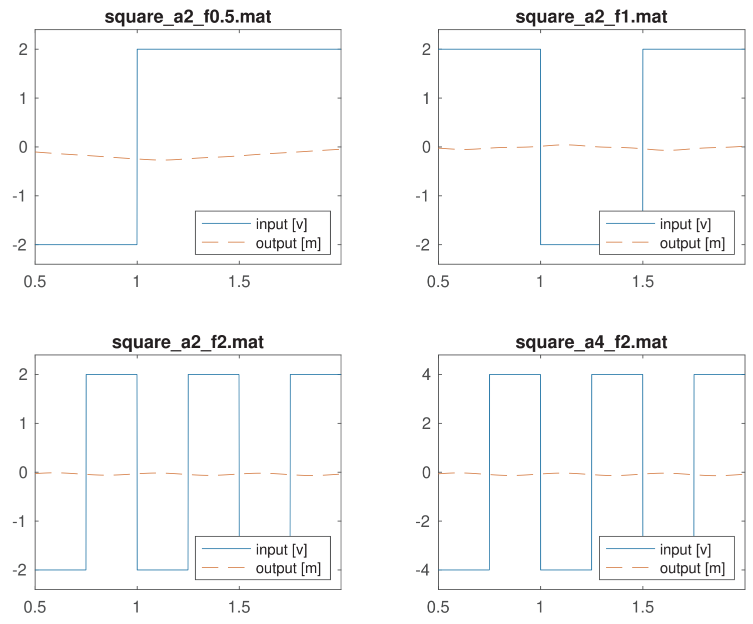
$$\bar{H}(s) = \frac{\bar{Y}(s)}{\bar{U}(s)} = \frac{a}{b}\frac{Y(s)}{U(s)},$$

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

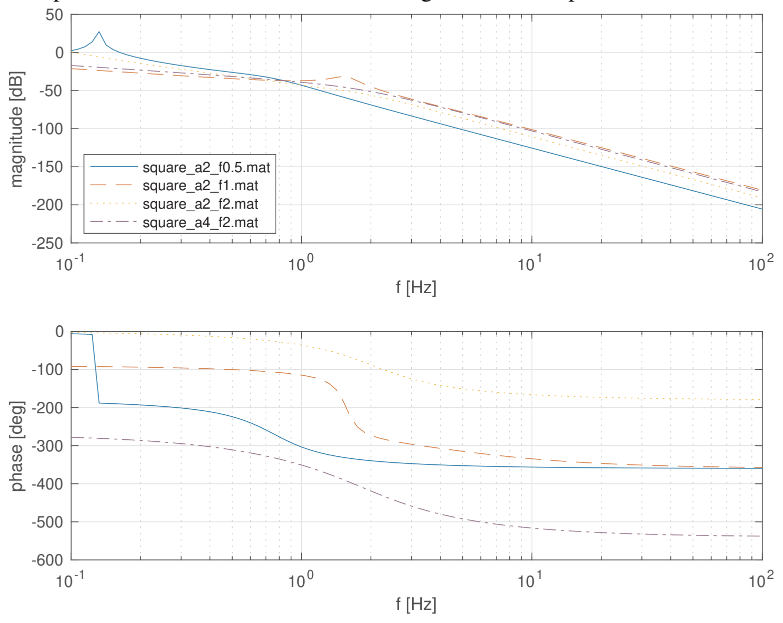
$$H(s) = \frac{Y(s)}{U(s)} = \frac{b}{a}\bar{H}(s). \qquad \square$$

Example 5.2 (Two-cart with spring (cont.)). We can see in Figure 5.4a, 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.

Figure 5.5a shows the same four sets of input/output data used to estimate the continuous-time transfer function, but the signal labeled "output" now corresponds to a scaled version of the output.

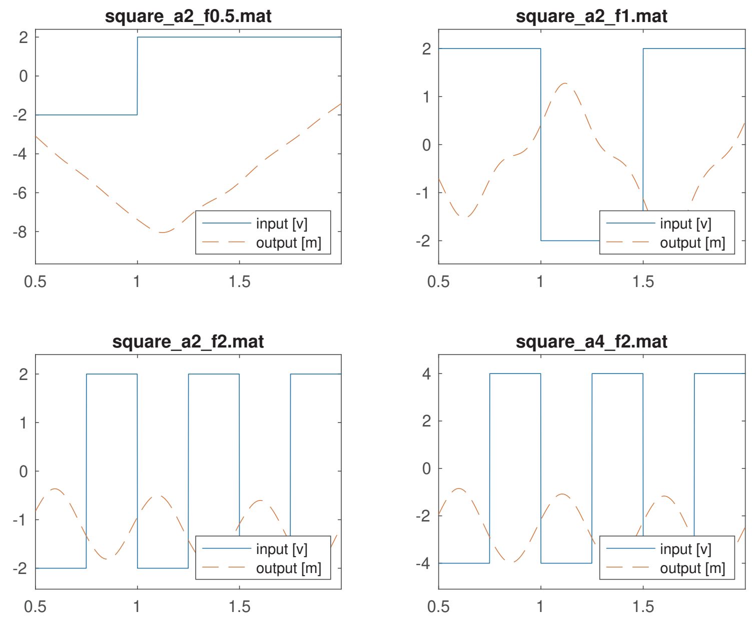


(a) Four sets of input/output data used to estimate the two-mass system in Figure 5.3. In all experiments the input signal u is a square wave with frequencies .5Hz, 1Hz, 2Hz, and 2Hz, respectively, from left to right and top to bottom. In the first three plots the square wave switches from -2v to +2v, whereas in the last plot it switches from -4v to +4v. All signals were sampled at 1KHz.

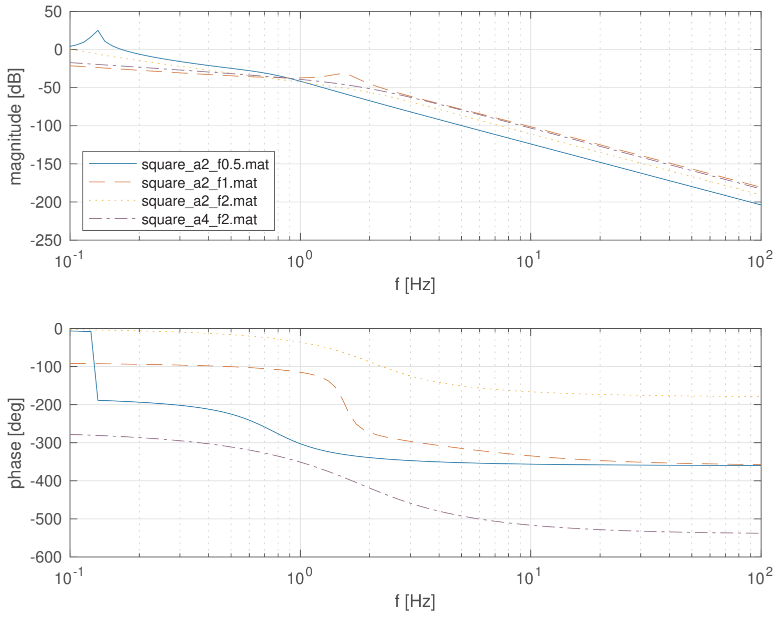


(b) Four continuous-time transfer functions estimated from the four sets of input/output data in (a), sampled at 1KHz. The transfer functions were obtained using the MATLAB® command `tfest` with np=4 and nz=0, which reflects an expectation of 4 poles and no zeros. The labels in the transfer functions refer to the titles of the four sets of input/output data in (a).

Figure 5.4. Initial attempt to estimate the continuous-time transfer function for the two-mass system in Figure 5.3.



(a) Four sets of input/output data used to estimate the two-mass system in Figure 5.3. 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 output y* so that it is comparable in magnitude to the input. All signals were sampled at 1KHz.



(b) Four continuous-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® command `tfest` with np=4 and nz=0, which reflects an expectation of 4 poles and no zeros. The labels in the transfer functions refer to the titles of the four sets of input/output data in (a). The transfer functions plotted are scaled so that they reflect an output in its original units.

Figure 5.5. Attempt to estimate the continuous-time transfer function for the two-mass system in Figure 5.3, *with a properly scaled output.*

Key observations. A careful examination of the outputs of the `tfest` command and the plots in Figure 5.5 reveals the following:

1. The quality of fit does not appear to be very good since for most data sets `tfest` reports a value for the `report.Fit.MSE` only 3-10 times smaller than the average value of y^2.
2. The standard deviations associated with several of the numerator and denominator parameters are very large, sometimes much larger than the parameters themselves, which *indicate problems in the estimated transfer functions.*
3. The integrator that we were expecting in the transfer functions is not always there.
4. The four sets of data input/output data continue to result in *dramatically different identified transfer functions*.

All these 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 5.5a), 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 in the sense that it is the largest at low frequencies, but not necessarily with the expected phase of $-\pi/2$.

Since we know that the system has a *structural pole* at $s = 0$, we should force it into the model using the technique seen in Section 4.5. Figure 5.6a shows the same four sets of input/output data used to estimate the continuous-time transfer function, but the signal labeled "output" now corresponds to the signal $\bar{y}$ that we encountered in Section 4.5. The new identified transfer functions that appear in Figure 5.6b are now much more consistent. In addition,

1. The quality of fit is very good, with `tfest` reporting values for `report.Fit.MSE` that are 100-1000 times smaller than the average value of y^2.
2. Almost all standard deviations associated with the numerator and denominator parameters are at least 10 times smaller than the parameters themselves, which gives additional confidence to the model. □

5.3 Choice of Model Order

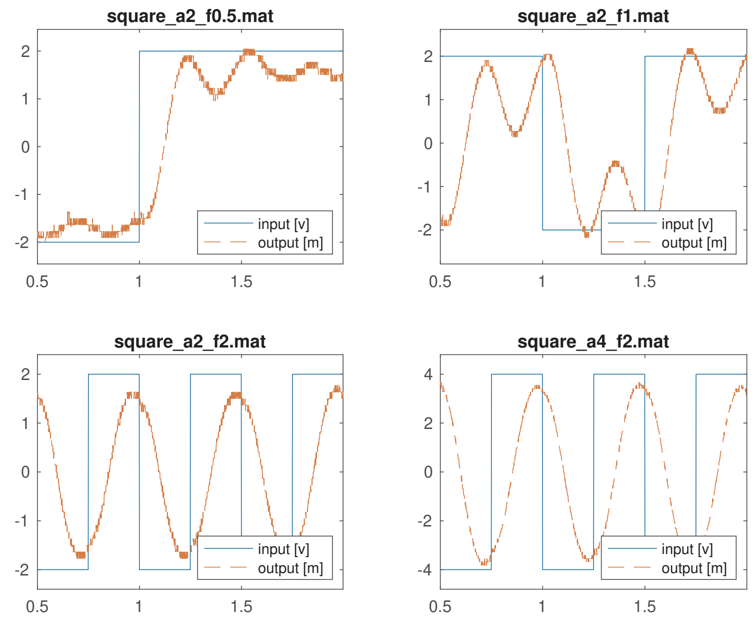
A significant difficulty in parametric identification of CARX models is that to construct the regression vector $\varphi_f(t)$ in (4.7), one needs to know the degrees m and n of the numerator and denominator. In fact, an incorrect choice for n will generally lead to difficulties.

1. Selecting a value for the number of poles n too small will lead to mismatch between the measured data and the model and the *MSE* 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

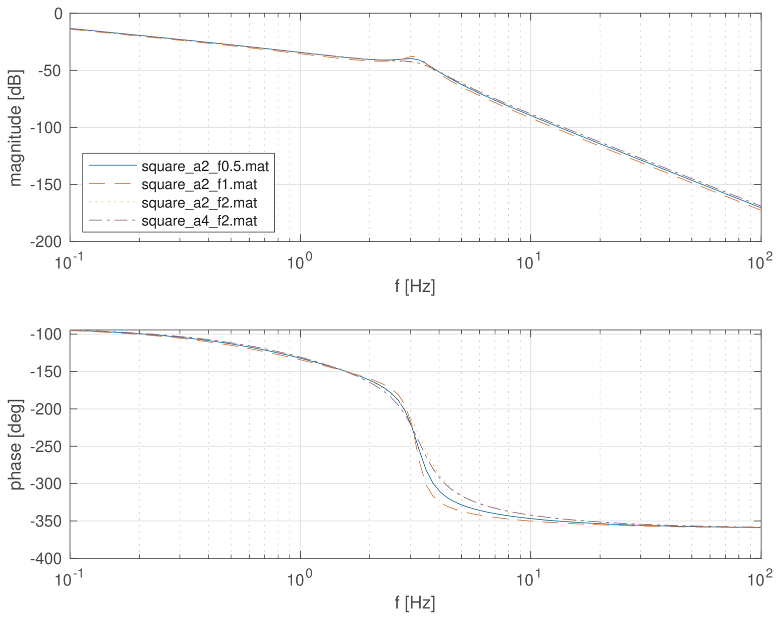
$$H(s) = \frac{1}{s+1},$$

but for estimation purposes we assumed that $m = n = 2$ and therefore attempted to determine constants α_i, β_i such that

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



(a) Four sets of input/output data used to estimate the two-mass system in Figure 5.3. 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 4.5, scaled so that its magnitude is comparable to that of the input u.* All signals were sampled at 1KHz.



(b) Four continuous-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® command `tfest` with np=3 and nz=0, which reflects an expectation of 3 poles in addition to the one at $s = 0$ and no zeros. A pole at $s = 0$ was inserted manually into the transfer function returned by `tfest`. The labels in the transfer functions refer to the titles of the four sets of input/output data in (a).

Figure 5.6. Attempt to estimate the continuous-time transfer function for the two-mass system in Figure 5.3, *forcing a pole at $s = 0$.*

If the model was perfect, it should be possible to match the data with any α_i, β_i such that

$$\frac{\alpha_2 s^2+\alpha_1 s+\alpha_0}{s^2+\beta_1 s+\beta_0}=\frac{s+p}{(s+1)(s+p)} \quad\Leftrightarrow\quad \begin{cases}\alpha_2=0,\ \alpha_1=1,\ \alpha_0=p,\\ \beta_1=p+1,\ \beta_0=p,\end{cases} \tag{5.4}$$

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.*

MATLAB® Hint 16. When using the `tfest` command, singularity of R can be inferred from standard deviations for the parameters that are large when compared with the estimated parameter values. ► p. 46

When there is noise, it will never be possible to perfectly explain the data and the smallest *MSE* will always be strictly positive (either with $n=1$ or $n=2$). However, in general, different values of p will result in different values for *MSE*. 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.

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

1. Perform system identification for a range of values for the numbers of poles n and the number of zeros m. For the different transfer functions identified,

   (a) compute the mean square error (MSE) normalized by the sum of squares of the output,

   (b) compute the largest parameter standard deviation, *normalized by the parameter values,*

   (c) plot the location of the transfer functions' poles and zeros in the complex plane.

   Note. Different parameters may have very different magnitudes, so it is important to normalize the values of the errors by the values of the parameters.

   These two values and plot can be obtained using

```
data=iddata(y,u);
% compute model estimate
model=tfest(data,npoles,nzeros);
% compute normalized mean-square error
normalizedMSE=report.Fit.MSE/((y'*y)/length(y));
% extract from report the numerator and denominator coefficients
num=report.parameters.ParVector(1:nz+1)';
den=report.parameters.ParVector(nz+2:end)';
% extract from report corresponding error standard deviations
std_num=sqrt(diag(...
   report.parameters.FreeParCovariance(1:nz+1,1:nz+1)))';
std_den=sqrt(diag(...
   report.parameters.FreeParCovariance(nz+2:nz+np+1,nz+np+1:end)))';
% compute normalized error standard deviations
maxStdDev=max([std_num,std_den]./[num,den]);
% plot root locus
rlocus(model);
```

2. Reject any choices of n and m for which any one of the following cases arise:

   (a) the normalized MSE is large, which means that the number of poles/zeros is not sufficiently large to match the data and likely m or n need to be increased; or

   (b) one or more of the parameter standard deviations are large (when compared to the corresponding parameter value), which means that the data is not sufficient to estimate all the parameters accurately and likely m and n need to be decreased; or

   (c) the identified transfer function has at least one pole almost as the same location as a zero and likely m and n need to be decreased; or

   (d) the leading coefficients of the numerator polynomial are very small (or equivalently the transfer function has very large zeros), which means that likely m should be decreased.

   One needs to exercise judgment in deciding when "the normalized MSE is large" or when "the parameter standard deviations are large." 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.

Note. A large parameter standard deviation may also mean that the input signal is not sufficiently rich to estimate the transfer function.

Note. Identifying a process transfer function with a pole-zero cancellation, like in (5.4), will make control extremely difficult since feedback will not be able to move that "phantom" zero/pole. This is especially problematic if the "phantom" pole is unstable or very slow.

Example 5.3 (Two-cart with spring (cont.))**.** Figures 5.7–5.8 show results obtained for the input/output data set shown in the bottom-right plot of Figure 5.6a (corresponding to the 2Hz square wave that switches between ±4v). The procedure to estimate the continuous-time transfer functions was similar to that used to obtain the those in Figure 5.6b, but we let the parameters np and nz that defines the number of poles and zeros vary.

Key observation. We can see from Figure 5.7 that 4-5 poles lead to a fairly small MSE. However, for every choice for the number of zeros/poles the worst standard deviation is still above 10% 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. □

5.4 Combination of Multiple Experiments

As discussed in Section 5.1, 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 $u(t)$ 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 $u_i(t)$. These experiments result in multiple sets of input/output data that can then be combined to identify a single CARX model:

MATLAB® Hint 16. `merge` allows one to combine data for this type of multi-input processing. ► p. 46

1. Apply L probe input signals $u_1(t)$, $u_2(t)$,..., $u_L(t)$ to the system.

2. Measure the corresponding outputs $y_1(t)$, $y_2(t)$, ..., $y_L(t)$.

3. Compute the first m derivatives of each filtered $u_i^f(t)$ and the first n derivatives of each filtered $y_i^f(t)$, using the fact that

$$\frac{d^k y_i^f(t)}{dt^k} = \mathscr{L}^{-1}\Big[\frac{s^k}{\omega(s)}Y_i(s)\Big], \qquad \frac{d^k u_i^f(t)}{dt^k} = \mathscr{L}^{-1}\Big[\frac{s^k}{\omega(s)}U_i(s)\Big].$$

4. Determine the values for the parameter θ that minimize the discrepancy between the left- and the right-hand-sides of (4.7) in a least-squares sense *for all signals*:

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

MATLAB® Hint 20. `PHI\Y` computes $\hat{\theta}$ directly, from the matrix `PHI`$=\Phi$ and the vector `Y`$=Y$.

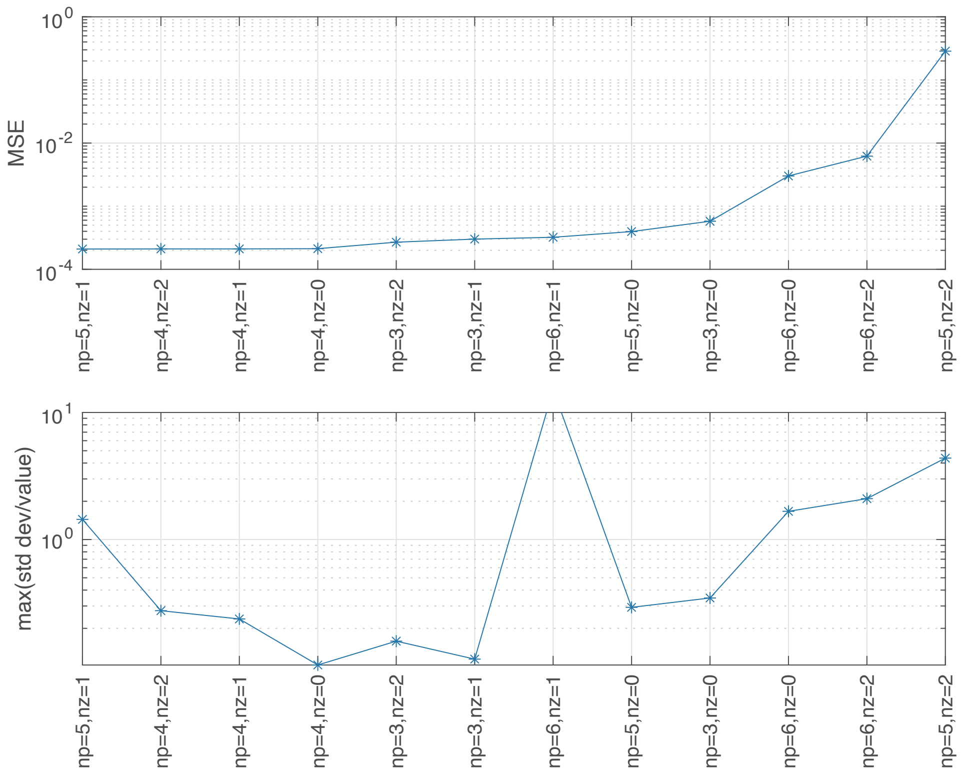


Figure 5.7. Choice of model order. All results in this figure refer to the estimation of the continuous-time transfer function for the two-mass system in Figure 5.3, using the set of input/output data shown in the bottom-right plot of Figure 5.6a, forcing a pole at $s = 0$ and with appropriate output scaling. The y-axis of the top plot shows the *MSE* and the y-axis of the bottom plot shows the highest (worst) ratio between a parameter standard deviation and its value, for different choices of the number of zeros (nz) and the number of poles (np, including the integrator at $s = 0$).

where

Attention! Remember to through away the initial times that fall before T_f.

$$\Phi := \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ \vdots \\ \Phi_L \end{bmatrix}, \quad \Phi_i := \begin{bmatrix} \varphi_i^f(t_{k_0}) \\ \varphi_i^f(t_{k_0+1}) \\ \vdots \\ \varphi_i^f(t_N) \end{bmatrix} = \begin{bmatrix} -\frac{d^{n-1}y_i^f(t_{k_0})}{dt^{n-1}} & \cdots & -y_i^f(t_{k_0}) & \frac{d^m u_i^f(t_{k_0})}{dt^m} & \cdots & u_i^f(t_{k_0}) \\ -\frac{d^{n-1}y_i^f(t_{k_0+1})}{dt^{n-1}} & \cdots & -y_i^f(t_{k_0+1}) & \frac{d^m u_i^f(t_{k_0+1})}{dt^m} & \cdots & u_i^f(t_{k_0+1}) \\ \vdots & & \vdots & \vdots & & \vdots \\ -\frac{d^{n-1}y_i^f(t_N)}{dt^{n-1}} & \cdots & -y_i^f(t_N) & \frac{d^m u_i^f(t_N)}{dt^m} & \cdots & u_i^f(t_N) \end{bmatrix},$$

$$Y := \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_L \end{bmatrix}, \quad Y_i := \begin{bmatrix} \frac{d^n y_i^f(t_{k_0})}{dt^n} \\ \frac{d^n y_i^f(t_{k_0+1})}{dt^n} \\ \vdots \\ \frac{d^n y_i^f(t_N)}{dt^n} \end{bmatrix}.$$

Example 5.4 (Two-cart with spring (cont.))**.** As noted before, we can see in Figure 5.7 that for every choice of the number of poles/zeros, at least one standard deviation is still above 10% 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 5.6a. 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.

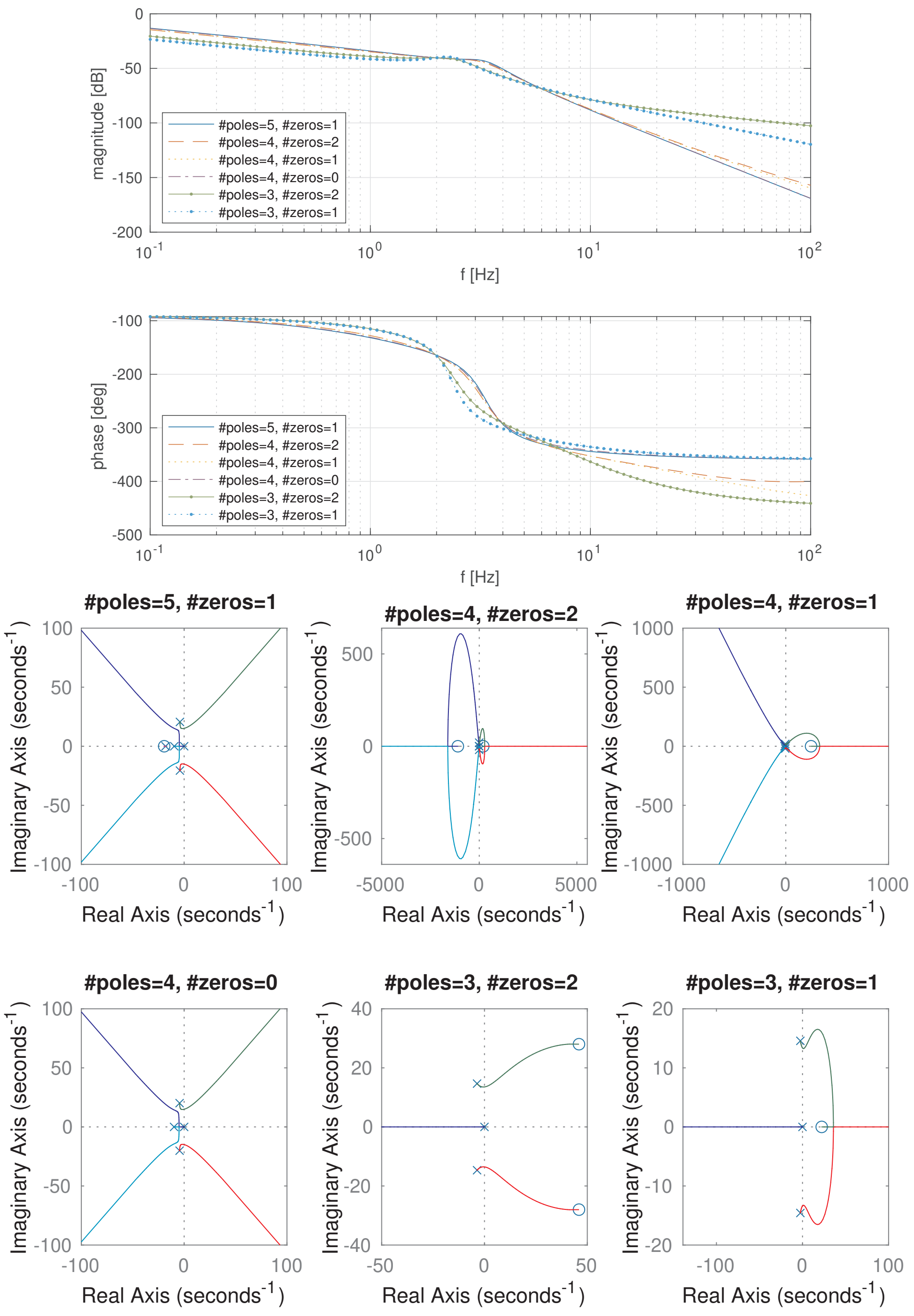


Figure 5.8. Bode plots and root locus for the transfer functions corresponding to the identification experiments in Figure 5.7 with smallest *MSE*. Each plot is labeled with the corresponding number of zeros and poles (including the integrator at $s = 0$).

Fix. By combining the data from the four sets of input/output data shown in Figure 5.6a, we should be able to decrease the uncertainty regarding the model parameters.

Figures 5.9–5.10 shows results obtained by combining all four sets of input/output data shown in Figure 5.6a. Aside from this change, the results shown follow from the same procedure used to construct the plots in Figures 5.7–5.8.

Key Observation. As expected, the *standard deviations for the parameter estimates decreased* and for several combinations of the number of poles/zeros the standard deviations are now well below 10% of the values of the corresponding parameter. However, one should still consider 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.

Regarding the choice of the system order, we are obtaining fairly consistent Bode plots for 4-6 poles (including the integrator at $s = 0$), at least up to frequencies around 10Hz. If the system is expected to operate below this frequency, then one should choose the simplest among the consistent models. This turns out to be 4 poles (excluding the integrator at $s = 0$) and no zeros. However, if one needs to operate the system at higher frequencies, a richer set of input signals is definitely needed and will hopefully shed further light into the choice of the number of poles. □

Note. 4 poles and no zeros is consistent with the physics-based model in (5.3). However, this model ignored the dynamics of the electrical motors and the spring mass, which may be important at higher frequencies.

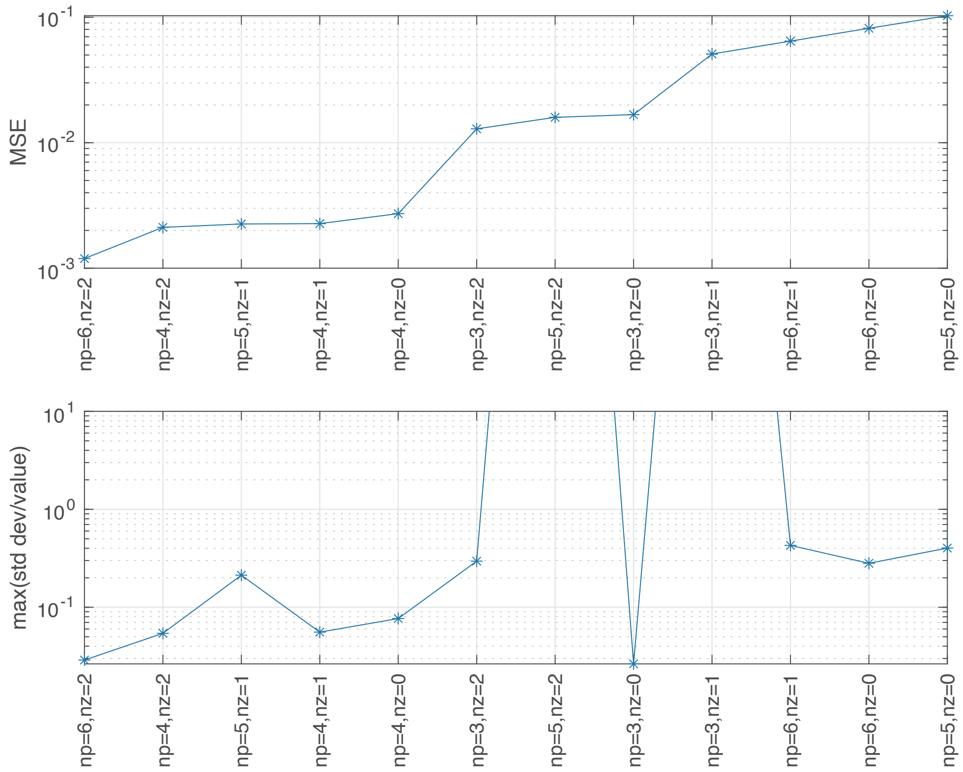


Figure 5.9. Choice of model order. All results in this figure refer to the estimation of the continuous-time transfer function for the two-mass system in Figure 5.3, *using all four sets of input/output data shown in Figure 5.6*, forcing a pole at $s = 0$ and with appropriate output scaling. The y-axis of the top plot shows the *MSE* and the y-axis of the bottom plot shows the highest (worst) ratio between a parameter standard deviation and its value, for different choices of the number of zeros (nz) and the number of poles (np, including the integrator at $s = 0$).

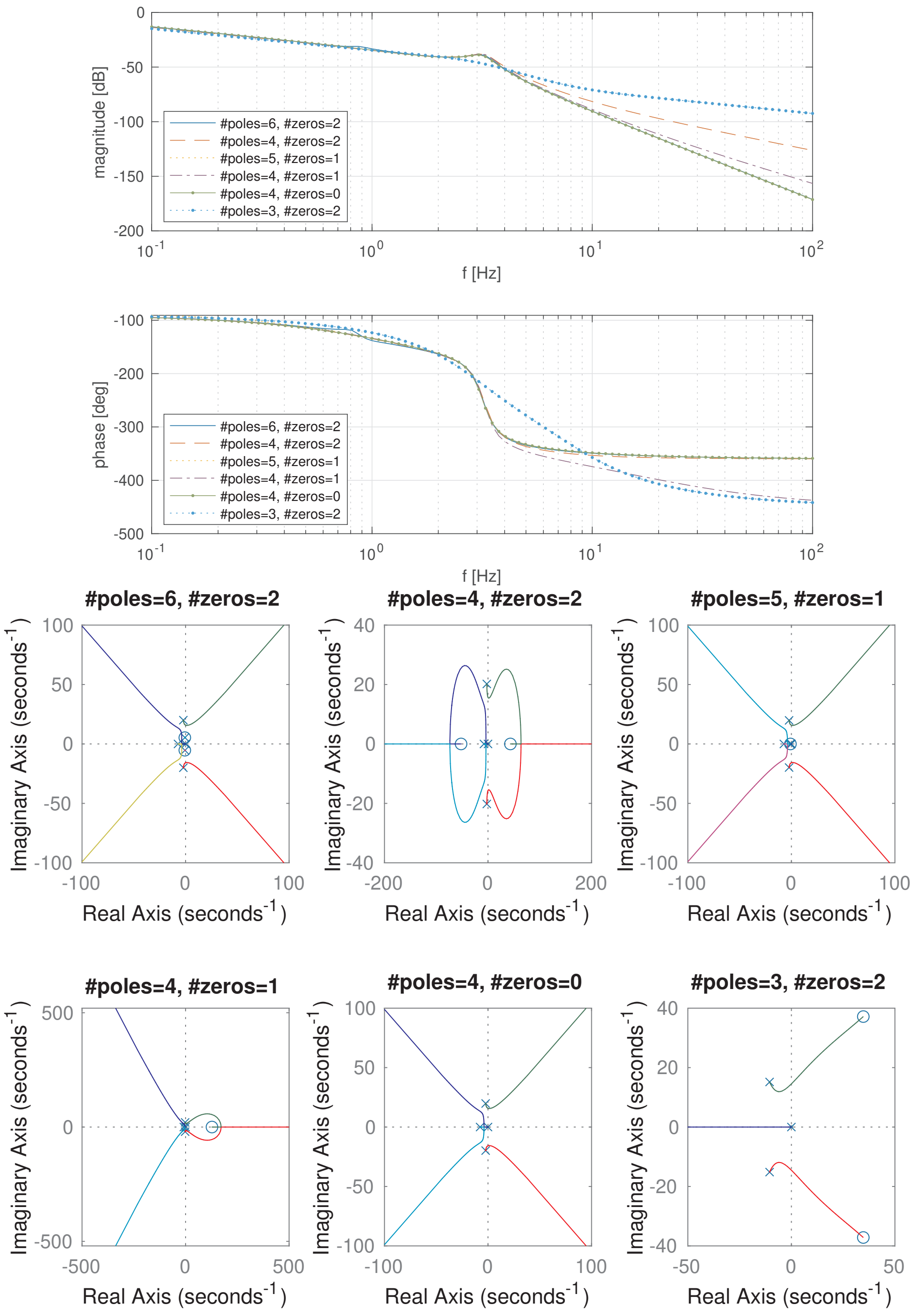


Figure 5.10. Bode plots and root locus of the transfer functions corresponding to the identification experiments in Figure 5.9 with smallest *MSE*. Each Bode plot is labeled with the corresponding number of zeros and poles (including the integrator at $s = 0$).

5.5 Closed-loop Identification

When the process is unstable, one cannot simply apply input probing signals to the open-loop systems. In this case, a stabilizing controller C must be available to collect the identification data. Typically, this is a low performance controller that was designed using only a coarse process model.

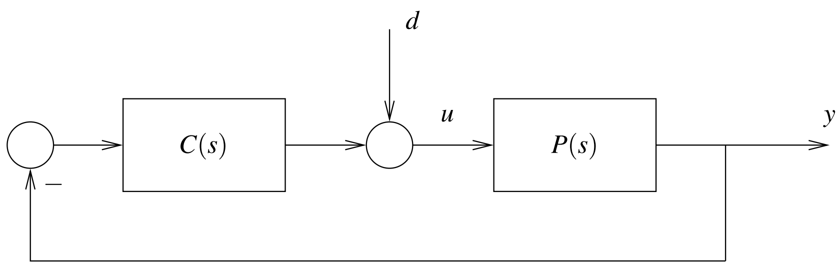


Figure 5.11. Closed-loop system

One effective method commonly used to identify processes that cannot be probed in open-loop consists of injecting an artificial disturbance d and estimating the closed-loop transfer function from this disturbance to the control input u and the process output y, as in Figure 5.11.

In this feedback configuration, the transfer functions from d to u and y are, respectively, given by

$$T_u(s) = \big(I + C(s)P(s)\big)^{-1}, \qquad T_y(s) = P(s)\big(I + C(s)P(s)\big)^{-1}, \tag{5.5}$$

where I denotes an identity matrix with size equal to the number of inputs. Therefore, we can recover $P(s)$ from these two closed-loop transfer functions, by computing

$$P(s) = T_y(s)T_u(s)^{-1}. \tag{5.6}$$

This formula can then be used to estimate the process transfer function from estimates of the two closed-loop transfer functions $T_y(s)$ and $T_u(s)$.

In closed-loop identification, if the controller is very good the disturbance will be mostly rejected from the output and $T_y(s)$ can become very small, leading to numerical errors and poor identification. For identification purposes a sluggish controller that does not do a good job at disturbance rejection is desirable.

Note. How did we get (5.5)? Denoting by U and D the Laplace transforms of u and d, respectively, we have that $U = D - CPU$, and therefore $(I + CP)U = D$, from which one concludes that $U = (I + CP)^{-1}D$. To obtain the transfer function to y, one simply needs to multiply this by P. *These formulas are valid, even if the signals are vectors and the transfer functions are matrices.*

Attention! There is a direct feedthrough term from d to u, which means that the transfer function $T_u(s)$ will have the same number of poles and zeros.

MATLAB® Hint 21. The system $P(s)$ in (5.6) can be computed using `Ty*inv(Tu)`.

5.6 Exercises

5.1 (Model order)**.** The data provided was obtained from a continuous-time linear system whose transfer functions has an unknown number of poles and zeros.

Use `tfest` to estimate the system's transfer function for different numbers of poles and zeros ranging from 1 through 5 poles. For the different transfer functions identified,

1. compute the mean square error (MSE) normalized by the sum of squares of the output,
2. compute the largest value of the parameter standard deviation normalized by the corresponding parameter value,
3. plot the location of the transfer functions' poles and zeros in the complex plane.

These two values and plot can be obtained using

```
model=tfest(data,npoles,nzeros);
% compute normalized mean-square error
normalizedMSE=sum(report.Fit.MSE)/(y'*y);
% extract from report the numerator and denominator coefficients
num=report.parameters.ParVector(1:nzeros+1)';
den=report.parameters.ParVector(nzeros+2:nzeros+npoles+1)';
% extract from report corresponding error standard deviations
std_num=sqrt(diag(...
   report.parameters.FreeParCovariance(1:nzeros+1,1:nzeros+1)))';
std_den=sqrt(diag(...
   report.parameters.FreeParCovariance(nzeros+2:nzeros+npoles+1,...
                                    nzeros+2:nzeros+npoles+1)))';
% compute normalized error standard deviations
maxStdDev=max([std_num,std_den]./[num,den]);
% plot root locus
rlocus(model);
```

Use this information to select the best values for the number of poles and zeros and provide the corresponding transfer function. Justify your choices.

Important: Write MATLAB® scripts to automate these procedures. You will need them for the lab. □

5.2 (Input magnitude)**.** A Simulink block that models a nonlinear spring-mass system is provided. This model expects the following variables to be defined:

```
Ts = 0.1;
tfinal = 100;
```

You will also need to define the magnitude of the step input and the measurement noise variance through the variables

```
step_mag
noise
```

Once all these variables have been set, you can run a simulation using

```
sim('spring',tfinal)
```

after which the variables `t`, `u`, and `y` are created with time, the control input, and the measured output.

1. Use the Simulink block to generate the system's response to step inputs with amplitude 0.1 and 2.0 and no measurement noise.

   For each of these inputs, use `tfest` to estimate the system's transfer function for different numbers of poles and zeros ranging from 1 through 3 poles.

   For the different transfer functions identified,

   (a) compute the mean square error (MSE) normalized by the sum of squares of the output,

   (b) compute the largest value of the parameter standard deviation normalized by the corresponding parameter value,

   (c) plot the transfer functions poles and zeros in the complex plane.

   These values can be obtained using

   ```
   model=tfest(data,npoles,nzeros);
   % compute normalized mean-square error
   normalizedMSE=sum(report.Fit.MSE)/(y'*y);
   % extract from report the numerator and denominator coefficients
   num=report.parameters.ParVector(1:nzeros+1)';
   den=report.parameters.ParVector(nzeros+2:nzeros+npoles+1)';
   ```

```
% extract from report corresponding error standard deviations
std_num=sqrt(diag(...
   report.parameters.FreeParCovariance(1:nzeros+1,1:nzeros+1)))';
std_den=sqrt(diag(...
   report.parameters.FreeParCovariance(nzeros+2:nzeros+npoles+1,...
                                       nzeros+2:nzeros+npoles+1)))';
% compute normalized error standard deviations
maxStdDev=max([std_num,std_den]./[num,den]);
% plot root locus
rlocus(model);
```

Use this information to select the best values for the number of poles and zeros and provide the corresponding transfer function. Justify your choices.

2. Use the Simulink block to generate the system's response to several step inputs with magnitudes in the range [0.1,2.0] and measurement noise with variance 10^{-3}.

   For the best values for the number of poles and zeros determined above, plot the normalized MSE as a function of the magnitude of the step input. Which magnitude leads to the best model?

Important: Write MATLAB® scripts to automate these procedures. You will need them for the lab.

□

Lecture 6

Parametric Identification of a Discrete-Time ARX Model

This lecture explains how the methods of least-squares can be used to identify ARX models.

Contents

6.1 ARX Model

Suppose that we want to determine the transfer function $H(z)$ of the SISO discrete-time system in Figure 6.1. In least-squares identification, one converts the problem of estimating $H(z)$ into the

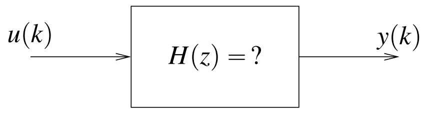


Figure 6.1. System identification from input/output experimental data

vector least-squares problem considered in Section 3.3. 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 6.1 are related by

$$\frac{Y(z)}{U(z)} = H(z) = \frac{\alpha_m z^m + \alpha_{m-1} z^{m-1} + \cdots + \alpha_1 z + \alpha_0}{z^n + \beta_{n-1} z^{n-1} + \cdots + \beta_1 z + \beta_0}, \tag{6.1}$$

where the α_i and the β_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:

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

and therefore

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

Note. We can view the difference $n-m$ between the degrees of the denominator and numerator of (6.1) as a delay, because the output $y(k)$ at time k in (6.2) only depends on the value of the input $u\big(k-(n-m)\big)$ at time $k-(n-m)$ and on previous values of the input.

Taking inverse z-transforms we obtain

$$y(k)+\beta_{n-1}y(k-1)+\cdots+\beta_1 y(k-n+1)+\beta_0 y(k-n)=\\ \alpha_m u(k-n+m)+\alpha_{m-1}u(k-n+m-1)+\cdots+\alpha_1 u(k-n+1)+\alpha_0 u(k-n). \tag{6.2}$$

This can be re-written compactly as

$$y(k)=\varphi(k)\cdot\theta \tag{6.3}$$

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

$$\theta := \begin{bmatrix}\alpha_m & \alpha_{m-1} & \cdots & \alpha_1 & \alpha_0 & \beta_{n-1} & \cdots & \beta_1 & \beta_0\end{bmatrix} \tag{6.4}$$

$$\varphi(k) := \begin{bmatrix}u(k-n+m) & \cdots & u(k-n) & -y(k-1) & \cdots & -y(k-n) & \end{bmatrix}. \tag{6.5}$$

The vector $\varphi(k)$ is called the *regression vector* and the equation (6.3) is called an *ARX model*, a short form of Auto-Regression model with eXogeneous inputs.

6.2 Identification of an ARX Model

We are now ready to solve the system identification Problem 3.2 introduced in Lecture 3, by applying the least-squares method to the ARX model:

Solution to Problem 3.2.

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 θ that minimize the discrepancy between the left- and the right-hand-sides of (6.3) in a least-squares sense.

MATLAB® Hint 22. `PHI\Y` computes $\hat\theta$ directly, from the matrix `PHI`$=\Phi$ and the vector `Y`$=Y$.

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

According to Section 3.3, the least-squares estimate of θ is given by

$$\hat\theta=(\Phi'\Phi)^{-1}\Phi'Y, \tag{6.6}$$

where

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

or equivalently

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

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

$$\frac{MSE}{MSO}=\frac{\frac{1}{N}\|\Phi\hat\theta-Y\|^2}{\frac{1}{N}\|Y\|^2} \tag{6.7}$$

When this quantity is much smaller than one, the mismatch between the left- and right-hand-sides of (6.3) has been made significantly smaller than the output. The *covariance of the estimation error* can be computed using

$$E[(\theta-\hat{\theta})(\theta-\hat{\theta})'] \approx \frac{1}{N-(n+m+1)}\|\Phi\hat{\theta}-Y\|^2(\Phi'\Phi)^{-1}. \tag{6.8}$$

Note. Why? In (6.8) we simply used (3.9), taking into account that the number of rows of Φ is N and the number of parameters to estimate in θ is $n+m+1$. ► p. 37

When the square roots of the diagonal elements of this matrix are much smaller than the corresponding entries of $\hat{\theta}$, one can have confidence in the values $\hat{\theta}$.

MATLAB® Hint 23. `arx` is convenient to perform ARX models identification because it does not require the construction of the matrix Φ and the vector Y. ► p. 72

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

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

MATLAB® Hint 24. `compare` is useful to determine the quality of fit. ► p. 73

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

6.3 Dealing with 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.,

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

where $\bar{H}(z)$ 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

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

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

and therefore

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

Note. The new output $\bar{y}(k)$ is not a causal function of the original output $y(k)$, but this is of no consequence since we have the whole $y(k)$ available when we carry out identification.

where

$$\bar{Y}(z) := (z-\lambda)Y(z) \quad\Rightarrow\quad \bar{y}(k)=y(k+1)-\lambda y(k). \tag{6.9}$$

MATLAB® Hint 25. One must be careful in constructing the vector $\bar{y}$ in (6.9) to be used by the function `arx`. ► p. 73

This means that we can directly estimate $\bar{H}(z)$ 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 $\bar{H}(z)$ by the term $\frac{1}{z-\lambda}$. □

6.4 MATLAB® Hints

MATLAB® Hint 23 (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 `Ts` is the sampling interval.

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

```
data1=iddata(y1,u1,Ts);
data2=iddata(y1,u1,Ts);
data=merge(data1,data2);
```

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, according to

Note 18. Typically, one chooses `na` equal to the (expected) number of poles, `nk` equal to the (expected) input delay, and nb=na−nk+1, which would be the corresponding number of zeros. In this case, $y(k)$ is a function of $y(k-1),\dots,y(k-\text{na})$ and of $u(k-\text{nk}),\dots,u(k-\text{nk}-\text{nb}+1)$. See equations (6.1)–(6.2). ► p. 69

$$\frac{Y(z)}{U(z)} = z^{-\text{nk}}\frac{b_1 + b_2 z^{-1} + \cdots + b_{\text{nk}+\text{nb}} z^{-\text{nb}+1}}{a_1 + a_2 z^{-1} + \cdots + a_{\text{na}+1} z^{-\text{na}}} \tag{6.10a}$$

$$= z^{\text{na}-\text{nk}-\text{nb}+1}\frac{b_1 z^{\text{nb}-1} + b_2 z^{\text{nb}-2} + \cdots + b_{\text{nk}+\text{nb}}}{a_1 z^{\text{na}} + a_2 z^{\text{na}-1} + \cdots + a_{\text{na}+1}} \tag{6.10b}$$

   and `model` is a MATLAB® object with the result of the identification.

   For processes with n_u inputs and n_y outputs, `na` should be an $n_y \times n_y$ square matrix and both `nb` and `nk` should be $n_y \times n_u$ rectangular matrices. In general, all entries of `na` should be equal to the number of poles of each transfer function, whereas the entries of `nk` and `nb` should reflect the (possibly different) delays and number of zeros, respectively, for each transfer function.

The estimated discrete-time transfer function can be recovered using the MATLAB® command

```
sysd=tf(model);
```

Additional information about the result of the system identification can be obtained in the structure `model.report`. Some of the most useful items in this structure include

- `model.report.Parameters.ParVector` is a vector with all the parameters that have been estimated, starting with the numerator coefficients and followed by the denominator coefficients, as in (6.4). These coefficients also appear in `tf(model)`.

- `model.report.Parameters.FreeParCovariance` is the error covariance matrix for the estimated parameters, as in (6.8). The diagonal elements of this matrix are the variances of the estimation errors of the parameter and their square roots are the corresponding standard deviations, which can be obtained using

```
StdDev=sqrt(diag(model.report.parameters.FreeParCovariance));
```

  One 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.

Note 19. Obtaining a standard deviation for one parameter that is comparable or larger than its estimated value typically arises in one of three situations. ► p. 73

- `model.report.Fit.MSE` is the mean-square estimation error (MSE), as in the numerator of (6.7), which measure of how well the response of the estimated model fits the estimation data. The MSE should be normalized by the Mean-Square Output

  ```
  MSO=y'*y
  ```

  when one want to compare the fit across different inputs/outputs.

The MATLAB® 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 (6.6). While this typically improves the quality of the estimate, it occasionally leads to poor results so one needs to exercise caution in evaluating the output of `arx`. □

MATLAB® Hint 25 (Dealing with known parameters)**.** One must be careful in constructing the vector $\bar{y}$ in (6.9) to be used by the function `arx`. In particular, its first element must be equal to

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

as suggested by (6.9). 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® commands could be used construct $\bar{y}$ from y and also to discard the last element of u:

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

□

MATLAB® Hint 24 (`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® hints 23, 9 for details on these objects. □

6.5 To Probe Further

Note 19 (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 7.1. 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, or the system delay. This issue is discussed in detail in Section 7.4. It can generally be resolved by one of three options:

   - When one encounters small estimated value for parameters corresponding to the terms in the *denominator* of (6.10) with the *most negative powers in z*, this may be resolved by selecting a smaller value for `nb`;
   - When one encounters small estimated value for parameters corresponding to the terms in the *numerator* of (6.10) with the *most negative powers in z*, this may be resolved by selecting a smaller value for `na`;

- When one encounters small estimated value for parameters corresponding to the terms in the *numerator* of (6.10) with the *least negative powers in z*, this may be resolved by selecting a large value for `nk`.

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 is 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 Lectures 8–9. □

6.6 Exercises

6.1 (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 known that the process has both a zero at γ and a pole at λ? □

6.2 (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. □

Lecture 7

Practical Considerations in Identification of Discrete-time ARX Models

This lecture discusses several practical considerations that are crucial for successful identifications.

Contents

7.1 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 Ω 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).

Note. *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.

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 7.1, there is usually an "optimal" input level that needs to be determined by trial-and-error. □

Note. Combining multiple experiments (see Section 7.5), each using a sinusoid of a different frequency is typically better than using a chirp signal since one has better control of the duration of the input signals at the different frequencies.

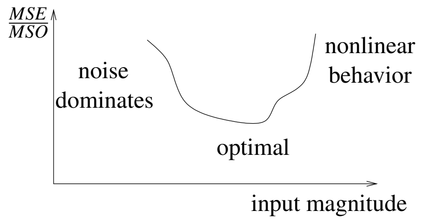


Figure 7.1. Optimal choice of input magnitude

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.

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. 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 by 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.

2. Check if the matrix R is far from singular. If not a different "richer" input should be used.

MATLAB® Hint 23. When using the `arx` command, the singularity of the matrix R can be inferred from the standard deviations associated with the parameter estimates. ► p. 72

3. Check the quality of fit by computing $\frac{MSE}{MSO}$. 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 7.4).

MATLAB® Hint 23. When using the `arx` command, the quality of fit can be inferred from the `noise` field in the estimated model. ► p. 72

Example 7.1 (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 7.2. From Newton's

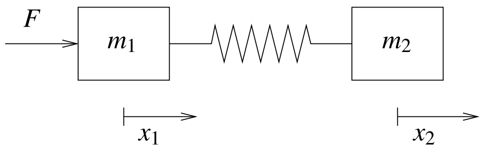


Figure 7.2. Two-mass with spring

law one concludes that

$$m_1\ddot{x}_1 = k(x_2 - x_1) + f, \qquad m_2\ddot{x}_2 = k(x_1 - x_2), \tag{7.1}$$

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

$$f = \frac{K_m K_g}{R_m r}\Big(v - \frac{K_m K_g}{r}\dot{x}_1\Big), \tag{7.2}$$

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 function, we replace f from (7.2) into (7.1) and take Laplace transforms to conclude that

$$\begin{cases} m_1 s^2 X_1(s) = k\big(Y(s) - X_1(s)\big) + \frac{K_m K_g}{R_m r}\Big(U(s) - \frac{K_m K_g}{r} s X_1(s)\Big) \\ m_2 s^2 Y(s) = k\big(X_1(s) - Y(s)\big) \end{cases}$$

$$\Rightarrow \quad \begin{cases} \Big(m_1 s^2 + \frac{K_m^2 K_g^2}{R_m r^2} s + k\Big) X_1(s) = kY(s) + \frac{K_m K_g}{R_m r} U(s) \\ \big(m_2 s^2 + k\big) Y(s) = k X_1(s) \end{cases}$$

$$\Rightarrow \quad \big(m_2 s^2 + k\big)\Big(m_1 s^2 + \frac{K_m^2 K_g^2}{R_m r^2} s + k\Big) Y(s) = k^2 Y(s) + \frac{k K_m K_g}{R_m r} U(s)$$

$$\frac{Y(s)}{U(s)} = \frac{\frac{k K_m K_g}{R_m r}}{\big(m_2 s^2 + k\big)\Big(m_1 s^2 + \frac{K_m^2 K_g^2}{R_m r^2} s + k\Big) - k^2} \tag{7.3}$$

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 (7.3) reveals that there is no term of order zero, since the denominator is equal to

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

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

Note. 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 7). ► p. 16

Our goal now is to determine the system's transfer function (7.3) 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 (7.1)–(7.2) 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.

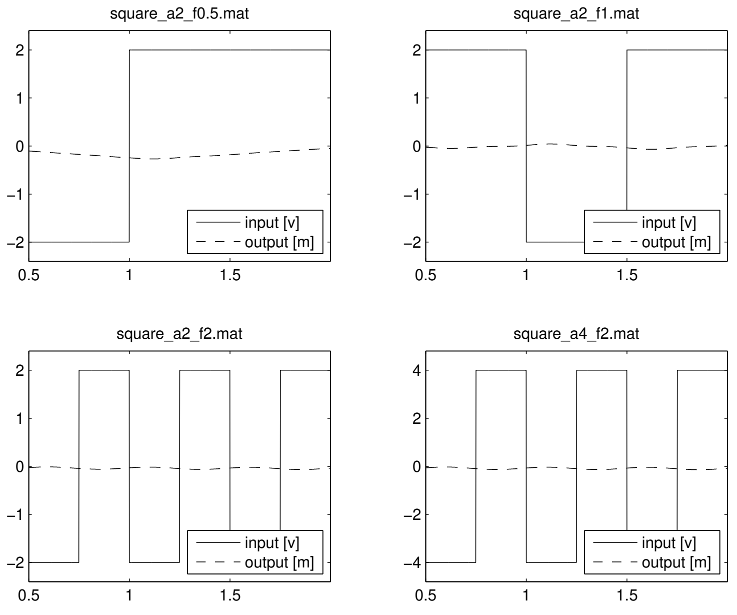
Figure 5.5a shows four sets of input/output data and four discrete-time transfer functions identified from these data sets.

Key observations. A careful examination of the outputs of the `arx` command and the plots in Figure 7.3 reveals the following:

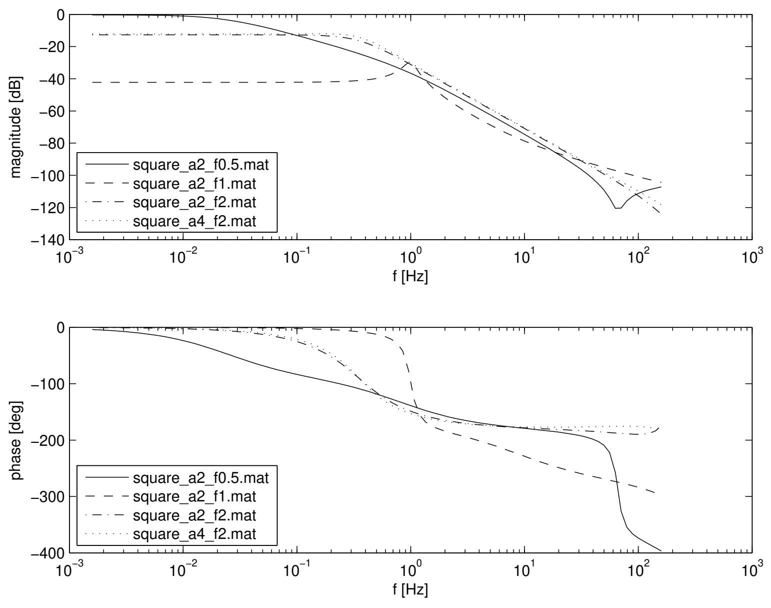
1. The quality of fit appears to be very good since `arx` reports a very small value for the `noise` parameter (around 10^{-10}). But this value is suspiciously small when compared to the average value of y^2 (around 10^{-2}).

2. While difficult to see in Figure 7.3a, 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 7.3a because of scaling is actually an indication of trouble*, as we shall discuss shortly.

3. 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).



(a) Four sets of input/output data used to estimate the two-mass system in Figure 7.2. 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® command `arx` with na=4, nb=4, and nk=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 18, p. 72). The labels in the transfer functions refer to the titles of the four sets of input/output data in (a).

Figure 7.3. Initial attempt to estimate the discrete-time transfer function for the two-mass system in Figure 7.2.

4. The standard deviations associated with the numerator parameters appear to be worse, often exceeding the estimate, which *indicate problems in the estimated transfer functions.*
5. The integrator that we were expecting in the transfer functions is not there.
6. The four sets of data input/output data resulted in *dramatically different identified transfer functions*.

The last two 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 7.3a), 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 6.3. Figure 7.4a 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 6.3. The new identified transfer functions that appear in Figure 7.4b 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. □

7.2 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

$$\bar{u}(k) = bu(k), \qquad \bar{y}(k) = ay(k), \qquad \forall k.$$

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

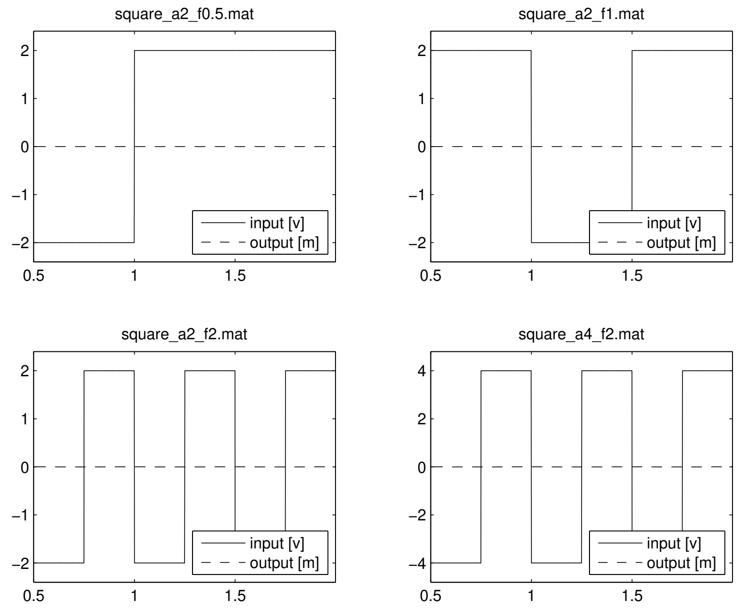
$$\bar{H}(z) = \frac{\bar{U}(z)}{\bar{Y}(z)} = \frac{b}{a}\frac{U(z)}{Y(z)},$$

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

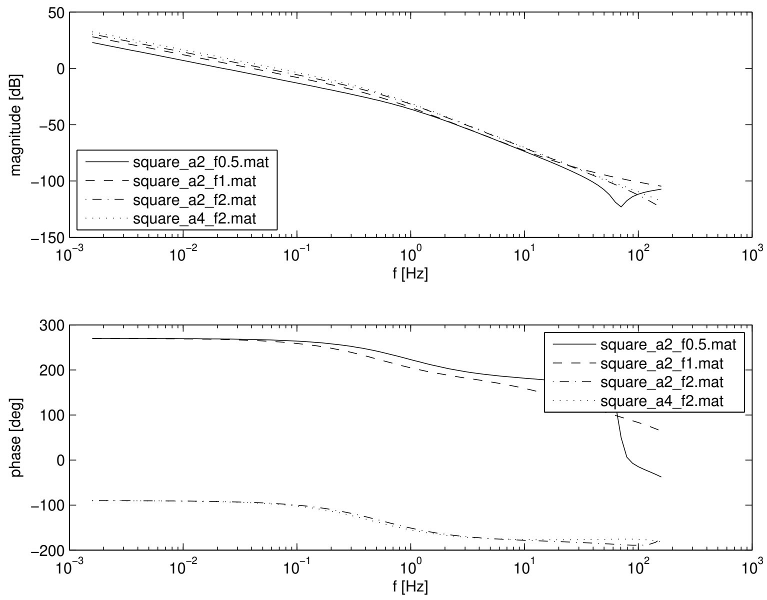
$$H(z) = \frac{U(z)}{Y(z)} = \frac{a}{b}\bar{H}(z). \qquad \square$$

Example 7.2 (Two-cart with spring (cont.)). We can see in Figure 7.4a, 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 7.5a 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 6.3.

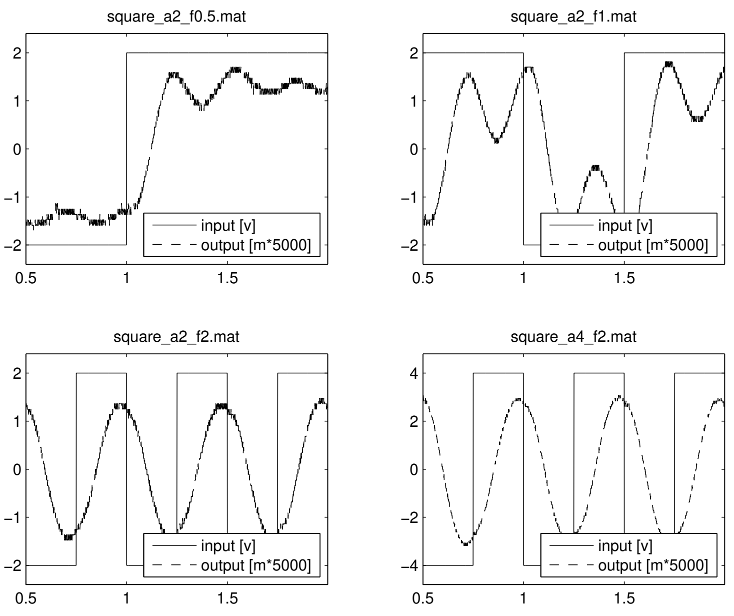


(a) Four sets of input/output data used to estimate the two-mass system in Figure 7.2. In all experiments the input signal u is a square wave with frequencies .5Hz, 1Hz, 2Hz, and 2Hz, respectively, from left to right and top to bottom. In the first three plots the square wave switches from -2v to +2v, 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 6.3.* All signals were sampled at 1KHz.

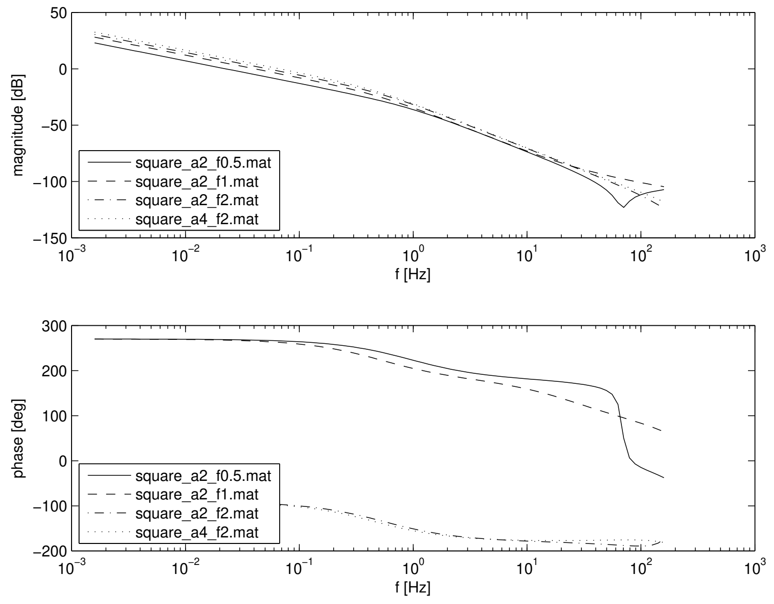


(b) Four discrete-time transfer functions estimated from the four sets of input/output data in (a), sampled at 1KHz. The transfer functions were obtained using the MATLAB® command `arx` with na=3, nb=4, and nk=0, which reflects an expectation of 3 poles in addition to the one at $z = 1$ and no delay 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 18, p. 72). 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 7.4. Attempt to estimate the discrete-time transfer function for the two-mass system in Figure 7.2, *forcing a pole at $z = 1$.*



(a) Four sets of input/output data used to estimate the two-mass system in Figure 7.2. 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 6.3. 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® command `arx` with na=3, nb=4, 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 7.5. Attempt to estimate the discrete-time transfer function for the two-mass system in Figure 7.2, 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 7.5a now shows a clue to further problems: the output signals exhibits significant quantization noise. □

7.3 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:

$$\begin{aligned} y(k) = &-\beta_{n-1}y(k-1) - \cdots - \beta_1 y(k-n+1) + \beta_0 y(k-n) + \\ &+ \alpha_m u(k-n+m) + \alpha_{m-1}u(k-n+m-1) + \cdots + \alpha_1 u(k-n+1) + \alpha_0 u(k-n). \end{aligned} \tag{7.4}$$

In particular, one is looking for values of the parameters that are optimal at predicting $y(k)$ based on inputs and outputs from time $k-1$ 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 (7.4) are very close to each other and often their difference is smaller than one quantization interval of the analogue-to-digital converter (ADC). As shown in Figure 7.6, in this case the pattern of output values that appear in (7.4) 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.

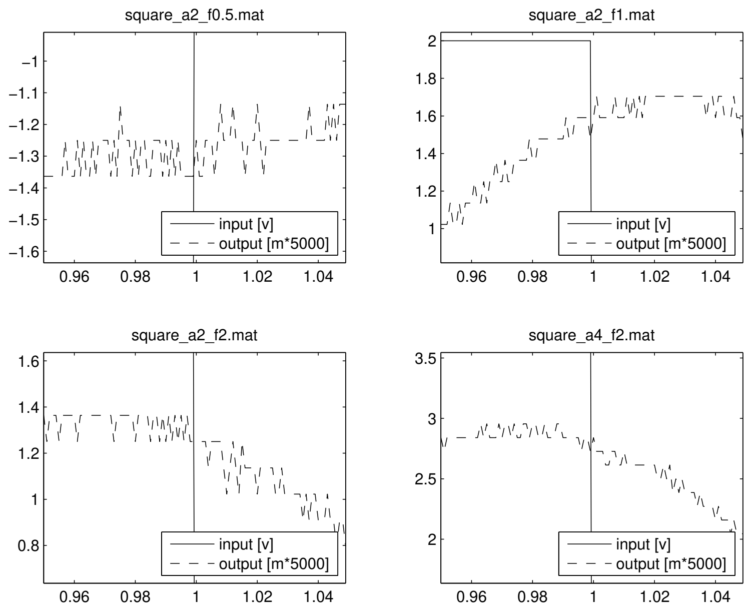


Figure 7.6. Magnified version of the four plots in Figure 7.5a showing the input and output signals around time $t = 1s$. The patterns seen in the output signals are mostly determined by the dynamics of the ADC converter.

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

MATLAB® Hint 26. A signal `y` can by down-sampled as in (7.5) using `bary=y(1:L:end)`. Down-sampling can also be achieved with the command `bary=downsample(y,L)`. This command requires MATLAB®'s signal processing toolbox.

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:

$$\bar{y}(k) = y(Lk), \qquad \bar{u}(k) = u(Lk). \tag{7.5}$$

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

MATLAB® Hint 27. A signal `y` can by down-sampled as in (7.6) using `bary=(y(1:L:end-2) + y(2:L:end-1)`

$$\bar{y}(k) = \frac{y(Lk-1)+y(Lk)+y(Lk+1)}{3}, \tag{7.6}$$

$$\bar{u}(k) = \frac{u(Lk-1)+u(Lk)+u(Lk+1)}{3}, \tag{7.7}$$

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."

MATLAB® Hint 28. 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®'s signal processing toolbox. ► p. 90

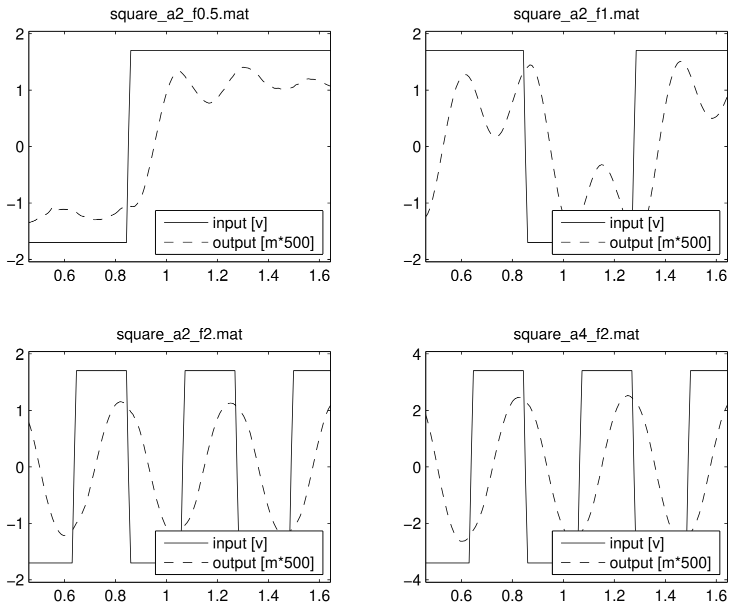
Example 7.3 (Two-cart with spring (cont.))**.** We can see, e.g., in Figure 7.4b that the system appears to have an "interesting" dynamical behavior in the range of frequencies from .1 to 10Hz. "Interesting," in the sense that the phase varies and the magnitude bode plot is not just a line. Since the 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 7.4a.

Fix. To remove some noise quantization noise, we down-sampled the input and output signals by a factor of 10 with the MATLAB® `resample` command. The new discrete-time frequency is therefore now sampled only at 100Hz, but this still appears to be sufficiently large. Figure 7.7a 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 7.5a, we can observe *a significant reduction in the output noise level.*

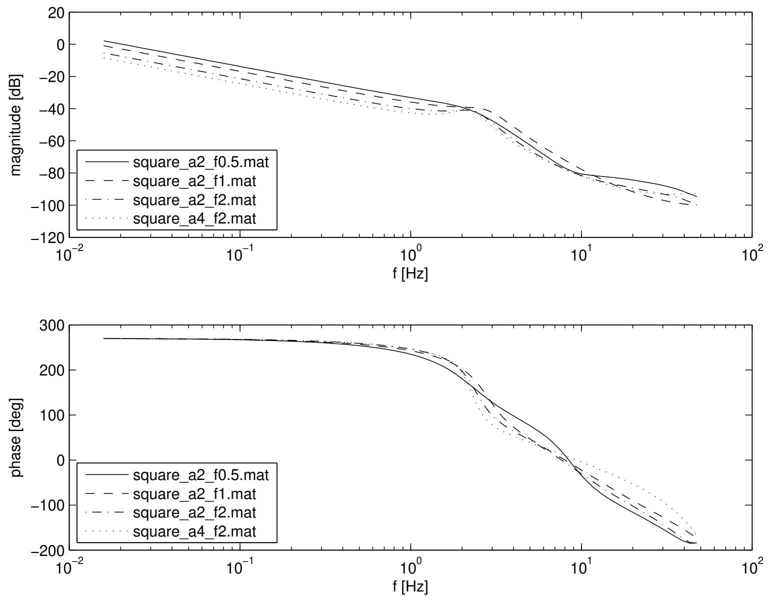
Key observation. The transfer functions identified show some improvement, especially because we are now starting to see some resonance, which would be expected in a system like this. More importantly, now *all the standard deviations associated with the numerator and denominator parameters are reasonably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

7.4 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.



(a) Four sets of input/output data used to estimate the two-mass system in Figure 7.2. 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 6.3. *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® command `arx` with na=3, nb=4, 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 7.7. Attempt to estimate the discrete-time transfer function for the two-mass system in Figure 7.2, forcing a pole at $z = 1$, with appropriate output scaling, and with the *signals down-sampled to 100Hz.*

1. Selecting a value for n too small will lead to mismatch between the measured data and the model and the *MSE* 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 α_i, β_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 α_i, β_i such that

$$\frac{\alpha_2 z^2 + \alpha_1 z + \alpha_0}{z^2 + \beta_1 z + \beta_0} = \frac{z-p}{(z-1)(z-p)} \quad \Leftrightarrow \quad \begin{cases} \alpha_2 = 0,\ \alpha_1 = 1,\ \alpha_0 = -p, \\ \beta_1 = -1-p,\ \beta_0 = p, \end{cases} \tag{7.8}$$

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 *MSE* will always be strictly positive (either with $n = 1$ or $n = 2$). However, in general, different values of p will result in different values for *MSE*. 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® Hint 29. When using the `arx` MATLAB® command, singularity of R can be inferred from standard deviations for the parameters that are large when compared with the estimated parameter values. ► p. 73

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

1. Perform system identification for a range of values for the numbers of poles n and the number of zeros m. For the different transfer functions identified,
   (a) compute the mean square error (MSE) normalized by the sum of squares of the output,
   (b) compute the largest parameter standard deviation,
   (c) plot the location of the transfer functions' poles and zeros in the complex plane.

Note. If it is known that there is a delay of at least d, one should make $m = n - d$ [cf. equation (6.2)]. ► p. 70

2. Reject any choices of n and m for which any one of the following cases arise:
   (a) the normalized MSE is large, which means that the number of poles/zeros is not sufficiently large to match the data and likely m or n need to be increased; or
   (b) one or more of the parameter standard deviations are large, which means that the data is not sufficient to estimate all the parameters accurately and likely m and n need to be decreased; or
   (c) the identified transfer function has at least one pole almost as the same location as a zero and likely m and n need to be decreased; or
   (d) the leading coefficients of the numerator polynomial are very small (or equivalently the transfer function has very large zeros), which means that likely m should be decreased.

Note. A large parameter standard deviation may also mean that the input signal is not sufficiently rich to estimate the transfer function.

Note. Identifying a process transfer function with a pole-zero cancellation, like in (5.4), will make control extremely difficult since feedback will not be able to move that "phantom" zero/pole. This is especially problematic if the "phantom" pole is unstable or very slow.

One needs to exercise judgment in deciding when "the normalized MSE is large" or when "the parameter standard deviations are large." 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 7.4 (Two-cart with spring (cont.))**.** Figure 7.8 shows results obtained for the input/output data set shown in the bottom-right plot of Figure 7.7a. The procedure to estimate the discrete-time transfer functions was similar to that used to obtain the those in Figure 7.7b, but we let the parameter na=nb−1 that defines the number of poles (excluding the integrator at $z = 1$) range from 1 to 12. The delay parameter `nk` was kept equal to 0.

As expected, the MSE error decreases as we increase `na`, but the standard deviation of the coefficients generally increases as we increase `na`, and rapidly becomes unacceptably large. The plot indicates that for na between 3 and 6 the parameter estimates exhibit relatively low variance. For larger values of na the decrease in MSE does not appear to be justify the increase in standard deviation.

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 7.8a that even for na=4, at least one standard deviation is still above 20% 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. □

7.5 Combination of Multiple Experiments

As discussed in Section 7.1, 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.

MATLAB® Hint 23. `merge` allows one to combine data for this type of multi-input processing. ► p. 72

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

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

of length N_1 and another input/output pair

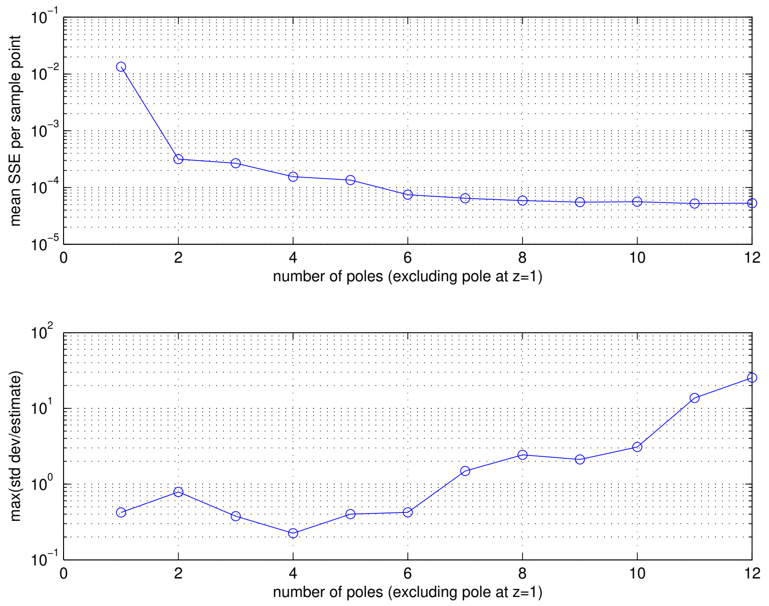
$$\big\{\big(u_2(k), y_2(k)\big) : k = 1, 2, \dots, N_2\big\}$$

of length N_2. One would construct

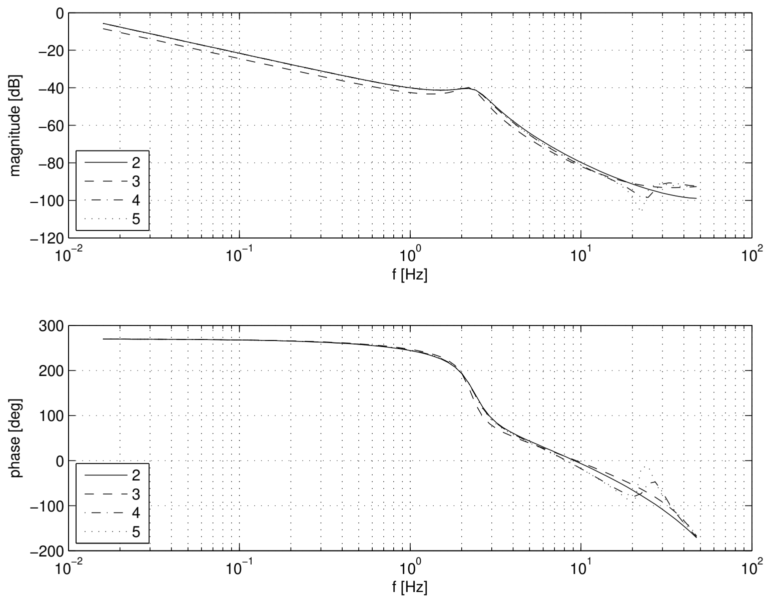
$$\Phi = \begin{bmatrix} y_1(2) & y_1(1) & u_1(3) & u_1(2) & u_1(1) \\ y_1(3) & y_1(2) & u_1(4) & u_1(3) & u_1(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_1(N_1-1) & y_1(N_1-2) & u_1(N_1) & u_1(N_1-1) & u_1(N_1-2) \\ y_2(2) & y_2(1) & u_2(3) & u_2(2) & u_2(1) \\ y_2(3) & y_2(2) & u_2(4) & u_2(3)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

and, according to Section 6.2, the least-squares estimate of

$$\theta := \begin{bmatrix} -\beta_1 & -\beta_0 & \alpha_m & \alpha_2 & \alpha_1 & \alpha_0 \end{bmatrix}$$



(a) The y-axis of the top plot shows the *MSE* 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 12 the parameter `na=nb` (shown in the x-axis) that defines the number of poles (excluding the integrator at $z = 1$). The delay parameter `nk` was kept the same equal to 0.



(b) Transfer functions corresponding to the identification experiments in (a). Each plot is labeled with the corresponding number of poles (excluding the integrator at $z = 1$).

Figure 7.8. 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 7.2, using the set of input/output data shown in the bottom-right plot of Figure 7.7a, forcing a pole at $z = 1$, with appropriate output scaling, and with the signals down-sampled to 100Hz.

MATLAB® Hint 30. `PHI\Y` computes $\hat{\theta}$ directly, from the matrix `PHI`$= \Phi$ and the vector `Y`$= Y$.

using *all the available data* is still given by

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

Example 7.5 (Two-cart with spring (cont.))**.** As noted before, we can see in Figure 7.8 that even for na=4, at least one standard deviation is still above 20% 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 7.7a. 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 7.7a, we should be able to decrease the uncertainty regarding the model parameters.

Figure 7.9 shows results obtained by combining all four sets of input/output data shown in Figure 7.7a. Aside from this change, the results shown follow from the same procedure used to construct the plots in Figure 7.8.

Key Observation. As expected, the *standard deviations for the parameter estimates decreased* and, for na $\leqslant 4$, all standard deviations are now below 15% 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.

Regarding the choice of the system order, we are obtaining fairly consistent Bode plots for 2-5 poles (excluding the integrator at $z = 1$), at least up to frequencies around 10Hz. If the system is expected to operate below this frequency, then one should choose the simplest model, which would correspond to 2 poles (excluding the integrator at $z = 1$). Otherwise richer input signals are definitely needed and will hopefully shed further light into the choice of the number of poles. □

7.6 Closed-loop Identification

When the process is unstable, one cannot simply apply input probing signals to the open-loop systems. In this case, a stabilizing controller C must be available to collect the identification data. Typically, this is a low performance controller that was designed using only a coarse process model.

One effective method commonly used to identify processes that cannot be probed in open-loop consists of injecting an artificial disturbance d and estimating the closed-loop transfer function from this disturbance to the control input u and the process output y, as in Figure 7.10.

In this feedback configuration, the transfer functions from d to u and y are, respectively, given by

$$T_u(s) = \big(I + C(s)P(s)\big)^{-1}, \qquad T_y(s) = P(s)\big(I + C(s)P(s)\big)^{-1}. \tag{7.9}$$

Note 20. The formulas in (7.9) are valid, even if the signals are vectors and the transfer functions are matrices. Why? ► p. 90

Therefore, we can recover $P(s)$ from these two closed-loop transfer functions, by computing

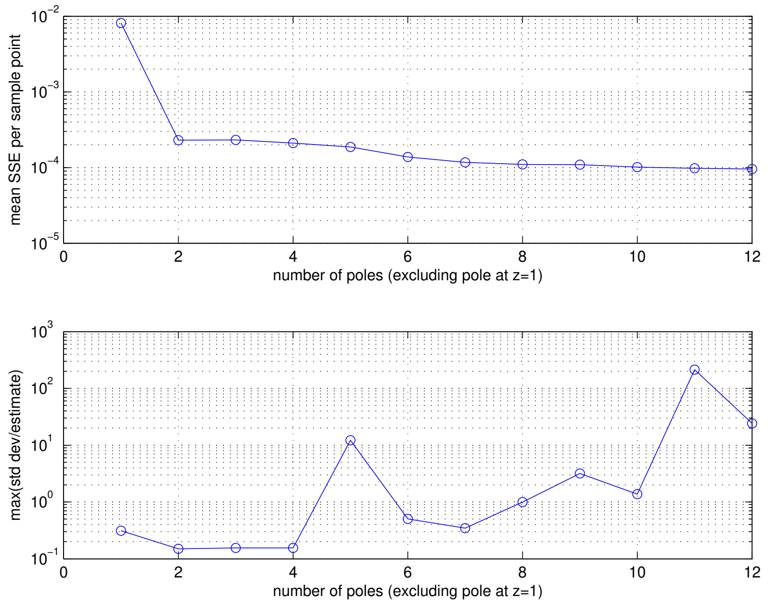
$$P(s) = T_y(s)T_u(s)^{-1}. \tag{7.10}$$

Attention! There is a direct feedthrough term from d to u, which means that the transfer function $T_u(s)$ will have no delay and therefore it will have the same number of poles and zeros.

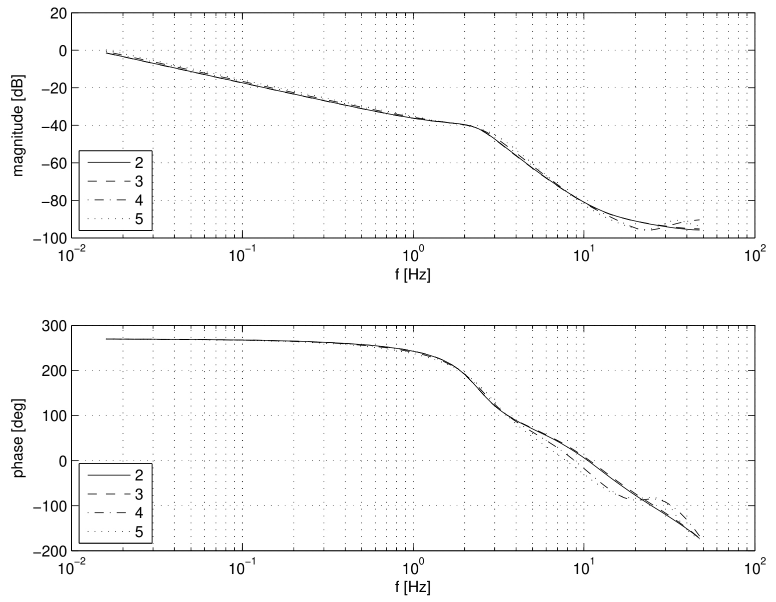
This formula can then be used to estimate the process transfer function from estimates of the two closed-loop transfer functions $T_y(s)$ and $T_u(s)$.

In closed-loop identification, if the controller is very good the disturbance will be mostly rejected from the output and $T_y(s)$ can become very small, leading to numerical errors and poor identification. For identification purposes a sluggish controller that does not do a good job at disturbance rejection is desirable.

MATLAB® Hint 31. The system $P(s)$ in (7.10) can be computed using `Ty*inv(Tu)`.



(a) The y-axis of the top plot shows the MSE 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 12 the parameter `na=nb` (shown in the x-axis) that defines the number of poles (excluding the integrator at $z = 1$). The delay parameter `nk` was kept the same equal to 0.



(b) Transfer functions corresponding to the identification experiments in (a). Each plot is labeled with the corresponding number of poles (excluding the integrator at $z = 1$).

Figure 7.9. 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 7.2, *using all four sets of input/output data shown in Figure 7.7*, forcing a pole at $z = 1$, with appropriate output scaling, and with the signals down-sampled to 100Hz.

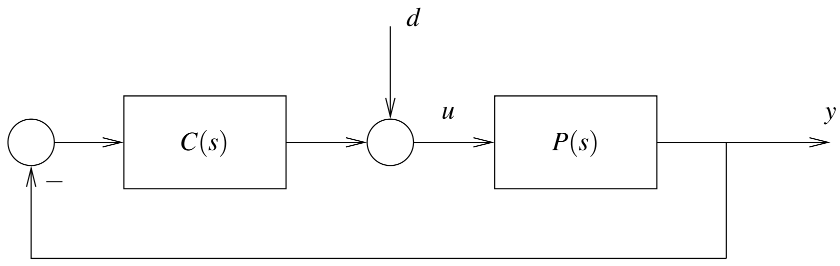


Figure 7.10. Closed-loop system

Note 20 (MIMO transfer functions)**.** How did we get (7.9)? Denoting by U and D the Laplace transforms of u and d, respectively, we have that $U = D - CPU$, and therefore $(I+CP)U = D$, from which one concludes that $U = (I+CP)^{-1}D$. To obtain the transfer function to y, one simply needs to multiply this by P. *These formulas are valid, even if the signals are vectors and the transfer functions are matrices.* □

7.7 MATLAB® Hints

MATLAB® Hint 28 (`resample`)**.** The command `resample` from the signal processing toolbox resamples a signal at a different sampling rate and simultaneously performs low-pass filtering to remove aliasing and noise. In particular,

bary=resample(y,M,L,F)

produces a signal `bary` with a sampling period that is `L/M` times that of `y`. Therefore, to reduce the sampling frequency one chooses `L>M` and the length of the signal `ybar` will be `L/M` times smaller than that of `y`.

In computing each entry of `bary`, this function averages the values of *F*L/M* entries of `y` forwards and backwards in time. In particular, selecting

bary=resample(y,1,L,1)

the signal `bary` will be a sub-sampled version of `y` with one sample of `ybar` for each `L` samples of `y` and each sample of `ybar` will be computed using a weighted average of the `2L` samples of `y` (`L` forwards in time and another `L` backwards in time). □

7.8 Exercises

7.1 (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.
3. 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? □

7.2 (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® scripts to automate this procedure. These scripts should take as inputs the simulation data $u(k)$, $y(k)$, and the integers m, n. □

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

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

1. 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 $T_s = .25$sec and $T_s = .0005$sec.
2. Identify the system's transfer function without down-sampling.
3. Identify the system's transfer function using the data collected with $T_s = .0005$sec but down-sampled.

   Important: write MATLAB® 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.
4. Briefly comment on the results. □

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 parameters that can only be determined up to some error and always involve simplifications.

   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.

Further reading A more extensive coverage of robust control can be found, e.g., in [4].

Lecture 8

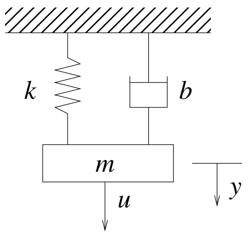
Robust stability

This lecture introduces the basic concepts of robust control.

Contents

8.1 Model Uncertainty

Suppose we want to control the spring-mass-damper system in Figure 8.1, which has the following



Measuring the mass' vertical position y with respect to the rest position of the spring, we obtain from Newton's law:

$$m\ddot{y} = -b\dot{y} - ky + u$$

Figure 8.1. Spring-mass-damper system.

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

$$P(s) = \frac{1}{ms^2 + bs + k}. \tag{8.1}$$

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:

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

The values with subscript $_0$ are called the *nominal values* for the parameters and the δ_i are called the *maximum deviations* from the nominal value.

MATLAB® Hint 32. `bode(sys)` draws the Bode plot of the system `sys`. ► p. 14

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 8.2 shows the Bode plot of (8.1) for different values of the parameters m, b, and k.

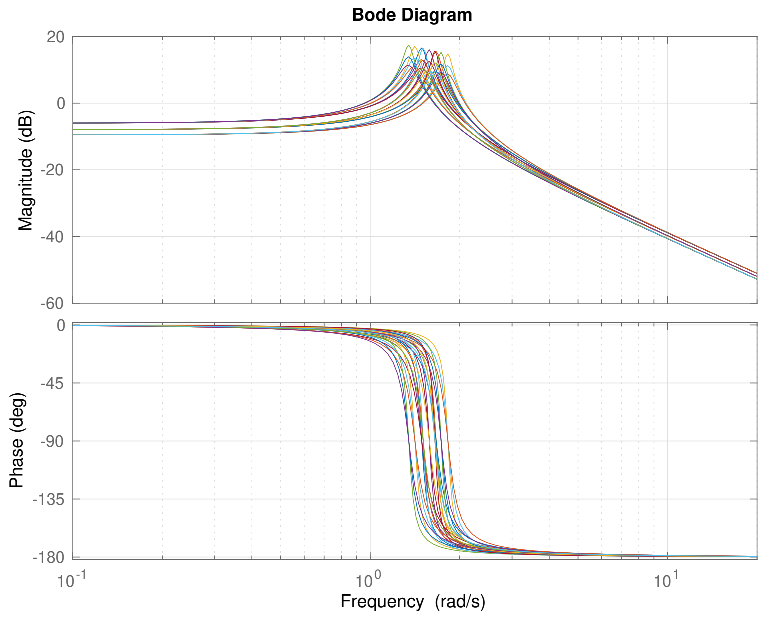


Figure 8.2. Bode plot of $P(s) = \frac{1}{ms^2+bs+k}$, for different values of $m \in [.9, 1.1]$, $b \in [.1, .2]$, and $k \in [2, 3]$. Note how the different values of the parameters lead to different values for the resonance frequency. One can see in the phase plot that it may be dangerous to have the cross-over frequency near the "uncertain" resonance frequency since the phase margin may very 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.

If we are given a specific controller, we can easily check if that controller is able to stabilize every one of the possible processes, e.g., by looking at the poles of each closed-loop transfer function. However, the *design problem* of constructing a controller that passes this test is more complex. 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.

8.1.1 Additive Uncertainty

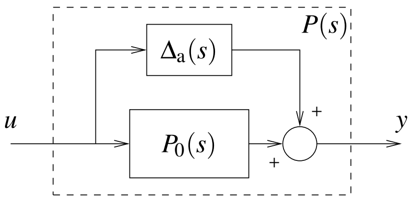
In robust control one starts by characterizing the uncertainty in terms of the process' frequency response in a way that make controller design "easy." To this effect one first selects a *nominal process transfer function* $P_0(s)$ and, then for each admissible process transfer function $P(s)$, one defines:

$$\Delta_a(s) \coloneqq P(s) - P_0(s), \tag{8.2}$$

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 8.3. Motivated by the diagram in Figure 8.3, $\Delta_a(s)$ is called an *additive uncertainty block*. $P_0(s)$ should correspond to the "most likely" transfer function, so that 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

$$P_0(s) = \frac{1}{m_0 s^2 + b_0 s + k_0},$$



$$\Delta_a(s) := P(s) - P_0(s)$$
$$\Updownarrow$$
$$P(s) = P_0(s) + \Delta_a(s)$$

Figure 8.3. Additive uncertainty

and, for each possible transfer function

$$P(s) = \frac{1}{ms^2 + bs + k}, \quad m \in [m_0 - \delta_1, m_0 + \delta_1], \quad b \in [b_0 - \delta_2, b_0 + \delta_2], \quad k \in [k_0 - \delta_3, k_0 + \delta_3], \tag{8.3}$$

one would then compute the corresponding additive uncertainty in (8.2). Figure 8.4 shows the magnitude Bode plots of $\Delta_a(s)$ for the process transfer functions in Figure 8.2.

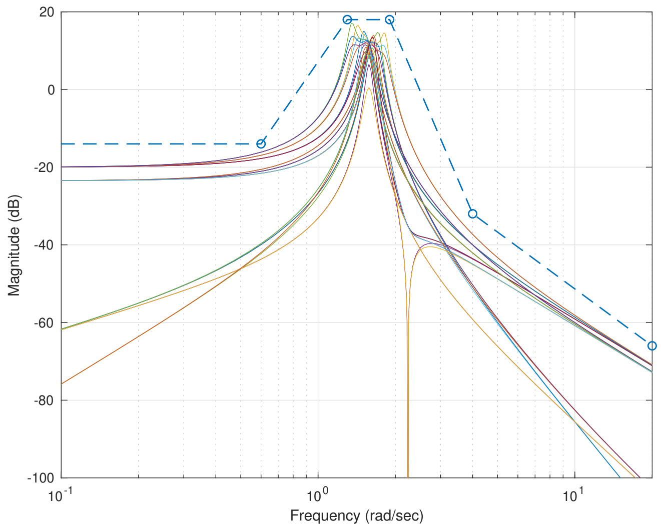


Figure 8.4. Additive uncertainty bounds for the process Bode plots in Figure 8.2 with $P_0(s) := \frac{1}{m_0 s^2 + b_0 s + k_0}$, $m_0 = 1$, $b_0 = 1.5$, $k_0 = 2.5$. The solid lines represent possible magnitude Bode plots for $|\Delta_a(j\omega)| = |P(j\omega) - P_0(j\omega)|$ and the dashed one represents 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.

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

$$|\Delta_a(j\omega)| = |P(j\omega) - P_0(j\omega)| \leqslant \ell_a(\omega), \quad \forall \omega. \tag{8.4}$$

In practice, the function $\ell_a(\omega)$ is simply an upper bound on the magnitude Bode plot of $\Delta_a(s)$.

Note. In what follows, we will attempt to stabilize *every process $P(s)$ that satisfies* (8.4). This generally asks for more than what is needed, since there will likely be transfer functions $\Delta_a(s)$ whose magnitude Bode plot lies below $\ell_a(\omega)$ but do not correspond to any process of the form (8.3). However, we will see that this description of uncertainty will simplify controller design.

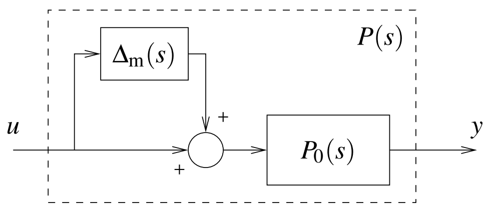
When one has available all the admissible $P(s)$ (or a representative set of them—such as in Figure 8.2), one can determine $\ell_a(\omega)$ by simply plotting $|P(j\omega) - P_0(j\omega)|$ vs. ω for all the $P(s)$ and choosing for $\ell_a(\omega)$ a function larger than all the plots. Since in general 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 8.4 shows a reasonable choice for $\ell_a(\omega)$ for the Bode plots in Figure 8.2.

8.1.2 Multiplicative Uncertainty

The additive uncertainty in (8.2) 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

$$\Delta_{\mathrm{m}}(s) := \frac{P(s) - P_0(s)}{P_0(s)}, \tag{8.5}$$

which measures how much $P(s)$ deviates from $P_0(s)$, relative to the size of $P_o(s)$. We can now express any admissible transfer function $P(s)$ as in Figure 8.5, which motivates calling $\Delta_{\mathrm{m}}(s)$ a *multiplicative uncertainty block*.



$$\Delta_{\mathrm{m}}(s) := \frac{P(s) - P_0(s)}{P_0(s)}$$
$$\Updownarrow$$
$$P(s) = P_0(s)\big(1 + \Delta_{\mathrm{m}}(s)\big)$$

Figure 8.5. Multiplicative uncertainty

Note. Also for multiplicative uncertainty, we will attempt to stabilize *every process $P(s)$ that satisfies* (8.6). This generally asks for more than what is needed, but we will see that this description of uncertainty will simplify controller design.

To obtain a characterization of the uncertainty *purely in the frequency domain* obtain a characterization of multiplicative uncertainty, one now determines a function $\ell_{\mathrm{m}}(\omega)$ sufficiently large so that for every admissible process transfer function $P(s)$ we have

$$|\Delta_{\mathrm{m}}(j\omega)| = \frac{|P(s) - P_0(s)|}{|P_0(s)|} \leqslant \ell_{\mathrm{m}}(\omega), \quad \forall \omega. \tag{8.6}$$

One can determine $\ell_{\mathrm{m}}(\omega)$ by plotting $\frac{|P(s)-P_0(s)|}{|P_0(s)|}$ vs. ω for all admissible $P(s)$ (or a representative set of them) and choosing $\ell_{\mathrm{m}}(\omega)$ to be larger than all the plots. Figure 8.6 shows $\ell_{\mathrm{m}}(\omega)$ for the Bode plots in Figure 8.2.

8.2 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 8.7 is stable *for every $\Delta_{\mathrm{m}}(j\omega)$ with norm smaller than $\ell_{\mathrm{m}}(\omega)$*. This can be done using the Nyquist stability criterion, which we review next.

Note. Figure 8.8 can represent the closed-loop system in Figure 8.7 if we choose the loop gain to be $L(s) := \big(1 + \Delta_{\mathrm{m}}(s)\big)P(s)C(s)$.

The Nyquist criterion is used to investigate the stability of the negative feedback connection in Figure 8.8. We briefly summarize it here. The textbook [5, Section 6.3] provides a more detailed description of it with several examples.

MATLAB® Hint 33. `nyquist(sys)` draws the Nyquist plot of the system `sys`. ► p. 105

Note 21. Why is the Nyquist plot a closed curve, symmetric with respect to the real axis? ► p. 101

The first step for the Nyquist Criterion consists of drawing the *Nyquist plot*, which is done by evaluating the loop gain $L(j\omega)$ 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 ω.

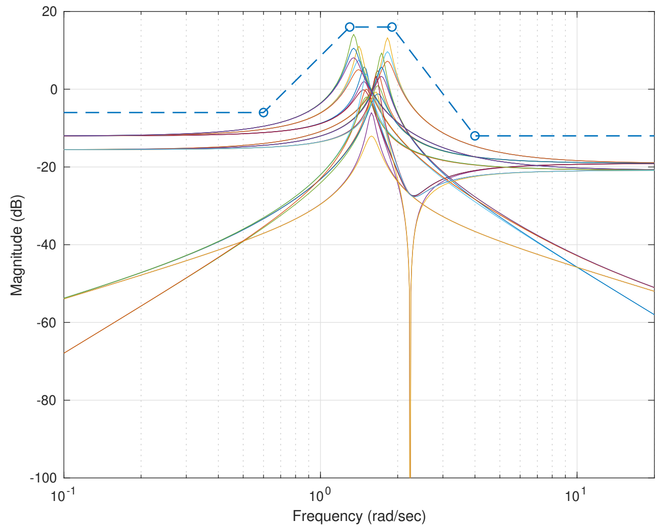


Figure 8.6. Multiplicative uncertainty bounds for the process Bode plots in Figure 8.2 with $P_0(s) := \frac{1}{m_0 s^2 + b_0 s + k_0}$, $m_0 = 1$, $b_0 = 1.5$, $k_0 = 2.5$. The solid lines represent possible magnitude Bode plots for $|\Delta_{\mathrm{m}}(j\omega)| = \frac{|P(j\omega) - P_0(j\omega)|}{|P_0(j\omega)|}$ and the dashed one represents the uncertainty bound $\ell_{\mathrm{m}}(\omega)$. 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.

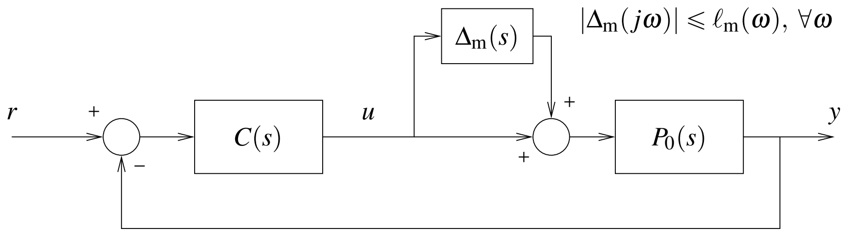

$|\Delta_{\mathrm{m}}(j\omega)| \leqslant \ell_{\mathrm{m}}(\omega), \ \forall \omega$
$\Delta_{\mathrm{m}}(s)$
r
$+$
$-$
$C(s)$
u
$+$
$+$
$P_0(s)$
y

Figure 8.7. Unity feedback configuration with multiplicative uncertainty

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

Note. Why are we "allowed" to move the poles on the imaginary axis? Because stability is a "robust property," in the sense that stability is preserved under small perturbations of the process (perturbations such as moving the poles by a small ε).

$$L(s) = \frac{s+1}{s(s-3)} \qquad \longrightarrow \qquad L_\varepsilon(s) \approx \frac{s+1}{(s+\varepsilon)(s-3)}$$
$$L(s) = \frac{s}{s^2+4} = \frac{s}{(s+2j)(s-2j)} \qquad \longrightarrow \qquad L_\varepsilon(s) \approx \frac{s}{(s+\varepsilon+2j)(s+\varepsilon-2j)} = \frac{s}{(s+\varepsilon)^2+4},$$

for a small $\varepsilon > 0$. The Nyquist criterion should then be applied to the "perturbed" transfer function $L_\varepsilon(s)$. If we conclude that the closed-loop is stable for $L_\varepsilon(s)$ with very small ε, then the closed-loop with $L(s)$ will also be stable and vice-versa. □

Note 21 (Nyquist plot)**.** Why is the Nyquist plot a closed curve? Because

$$\lim_{\omega\to\infty} L(j\omega) = \lim_{\omega\to-\infty} L(j\omega) = 0$$

if the number of poles is larger than the number of zeros or

$$\lim_{\omega\to\infty} L(j\omega) = \lim_{\omega\to-\infty} L(j\omega) = k \neq 0$$

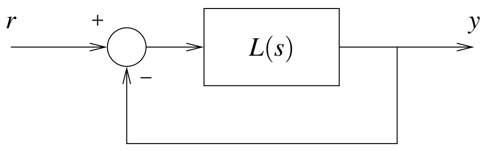


Figure 8.8. Negative feedback

if the number of poles is equal to the number of zeros, in which case k is the *high-frequency gain*.

Why is the Nyquist plot symmetric with respect to the real axis? Because $L(+j\omega)$ and $L(-j\omega)$ are complex conjugate numbers and therefore symmetric with respect to the real axis. Recall that addition, multiplication, and division of complex numbers "commute" with the complex-conjugate operation:

$$a^* + b^* = (a+b)^*, \qquad a^*b^* = (ab)^*, \qquad a^*/b^* = (a/b)^*. \qquad \square$$

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

$$\#\mathrm{CUP} = \#\mathrm{ENC} + \#\mathrm{OUP},$$

where #OUP *denotes the number of (open-loop) unstable poles of* $L(s)$ *and* #ENC *the number clockwise encirclements of the Nyquist plot around the point* -1. *To have a stable closed-loop one thus needs*

$$\#\mathrm{ENC} = -\#\mathrm{OUP}. \qquad \square$$

Note. To count the number of clockwise encirclements of the Nyquist plot around the point -1, we draw a ray from -1 to ∞ 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.

Example 8.1 (Spring-mass-damper system (cont.))**.** Figure 8.9 shows the Nyquist plot of the loop gain

$$L_0(s) = C(s)P_0(s), \tag{8.7}$$

for the nominal process model

$$P_0(s) := \frac{1}{m_0 s^2 + b_0 s + k_0}, \qquad m_0 = 1,\ b_0 = 1.5,\ k_0 = 2.5, \tag{8.8}$$

(used in Figures 8.4 and 8.6) and a PID controller

$$C(s) := \frac{10}{s} + 15 + 5s, \tag{8.9}$$

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. $\square$

8.3 Small Gain Condition

Consider the closed-loop system in Figure 8.7 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 $\Delta_{\mathrm{m}}(s) = 0$. Our goal is to find out if the closed-loop remains stable for every $\Delta_{\mathrm{m}}(j\omega)$ with norm smaller than $\ell_{\mathrm{m}}(\omega)$.

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

$$L_0(s) = C(s)P_0(s),$$

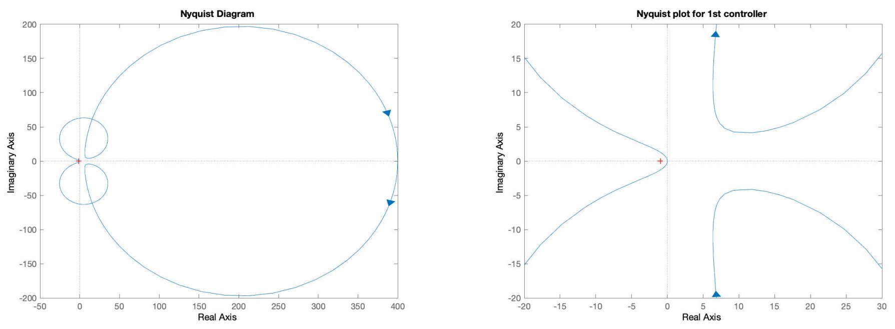


Figure 8.9. Nyquist plot for the (open-loop) transfer function in (8.7). The right figure shows a zoomed view of the origin. To avoid a pole over the imaginary axis, in these plots we moved the pole of the controller from 0 to $-.01$.

has the "right" number of encirclements (#ENC = $-$#OUP, where #OUP is the number of unstable poles of $L_0(s)$. To check is the closed-loop is stable from some admissible process transfer function

$$P(s) = P_0(s)\big(1+\Delta_{\rm m}(s)\big),$$

we need to draw the Nyquist plot of

$$L(s) = C(s)P(s) = C(s)P_0(s)\big(1+\Delta_{\rm m}(s)\big) = L_0(s) + L_0(s)\Delta_{\rm m}(s)$$

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

Note. We are assuming that $L(s)$ and $L_0(s)$ 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 open-loop pole.

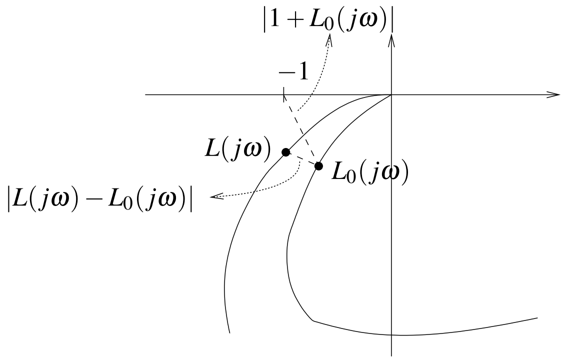


Figure 8.10. Nyquist plot derivation of the small-gain conditions

For a given frequency ω, the Nyquist plots of $L(s)$ and $L_0(s)$ differ by

$$|L(j\omega)-L_0(j\omega)| = |L_0(j\omega)\Delta_{\rm m}(j\omega)| \leqslant |L_0(j\omega)|\,\ell_{\rm m}(\omega),$$

A simple way to make sure that $L(j\omega)$ and $L_0(j\omega)$ have the same number of encirclements is to ask that the difference between the two always be smaller than the distance from $L_0(j\omega)$ to the point -1, i.e.,

$$|L_0(j\omega)|\,\ell_{\rm m}(\omega) < |1+L_0(j\omega)| \quad\Leftrightarrow\quad \frac{|L_0(j\omega)|}{|1+L_0(j\omega)|} < \frac{1}{\ell_{\rm m}(\omega)} \quad \forall\omega.$$

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

Note. A mnemonic to remember (8.10) is that the transfer function whose norm needs to be small is precisely the transfer function "seen" by the Δ_m block in Figure 8.7. This mnemonic also "works" for additive uncertainty.

► p. 106

MATLAB® Hint 34. To check if (8.10) 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.

Small-gain Condition. *The closed-loop system in Figure 8.7 is stable for every $\Delta_m(j\omega)$ with norm smaller than $\ell_m(\omega)$, provided that*

$$\left|\frac{C(j\omega)P_0(j\omega)}{1+C(j\omega)P_0(j\omega)}\right| < \frac{1}{\ell_m(\omega)}, \quad \forall\omega. \tag{8.10}$$

The transfer function on the left-hand-side of (8.10) is precisely the *complementary sensitivity function:*

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

for the nominal process. So (8.10) 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 (8.10) is called a *small-gain* condition.

Attention! The condition (8.10) only involves the controller $C(s)$ (to be designed) and the nominal process $P_0(s)$. Specifically, it *does not* involve testing some condition *for every* admissible process $P(s)$. This means that we can focus only on the nominal process $P_o(s)$ when we design the controller $C(s)$, and yet if we make sure that (8.10), we have the guarantee that $C(s)$ stabilizes every $P(s)$ in Figure 8.7. □

Example 8.2 (Spring-mass-damper system (cont.))**.** Figure 8.11 shows the Bode plot of the complementary sensitivity function for the nominal process in (8.8) and the PID controller in (8.9). In

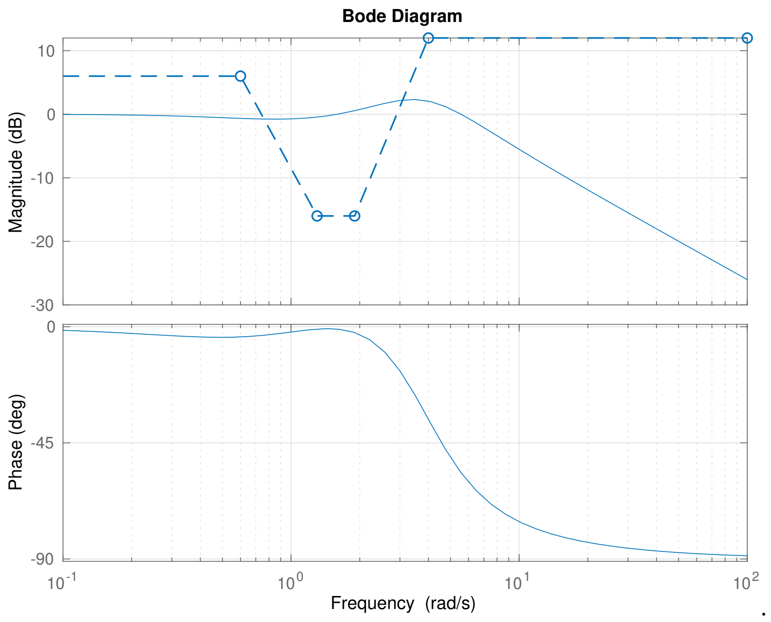


Figure 8.11. Verification of the small gain condition for the nominal process (8.8), the multiplicative uncertainty in Figure 8.6, and the PID controller (8.9). The solid line corresponds to the Bode plot of the complementary sensitivity function $T_0(s)$ and the dashed line to $\frac{1}{\ell_m(\omega)}$ (both in dB).

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 8.6. 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

$$C(s) = \frac{.005(s+5)^2}{s(s+.5)^2}, \tag{8.11}$$

the magnitude plot of $T_0(j\omega)$ is now always below that of $\frac{1}{\ell_{\mathrm{m}}(\omega)}$ and we can conclude from Figure 8.12 that we have stability for every admissible process. In this case, the price to pay was a low

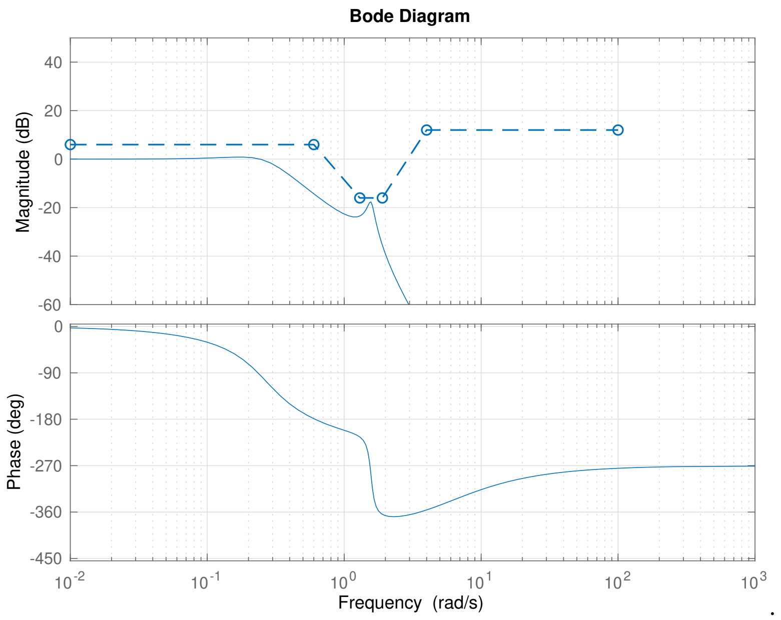


Figure 8.12. Verification of the small gain condition for the nominal process (8.8), the multiplicative uncertainty in Figure 8.6, and the integrator with 2 lags controller (8.11). The solid line corresponds to the Bode plot of the complementary sensitivity function $T_0(s)$ and the dashed line to $\frac{1}{\ell_{\mathrm{m}}(\omega)}$ (both in dBs).

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.* □

Note. Recall that the complementary sensitivity function is also the closed-loop transfer function for the reference r to the output y. 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.

8.4 MATLAB® Hints

MATLAB® Hint 33 (`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 32.

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.) □

8.5 Exercises

8.1 (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

$$.09 \leqslant k \leqslant .4 \qquad .006 \leqslant b \leqslant .03.$$

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

8.2 (Noisy identification in continuous time)**.** The data provided was obtained from 25 identification experiments for a continuous-time system with transfer function

$$H(s) = \frac{\alpha_1 s + \alpha_0}{s^3 + \beta_2 s^2 + \beta_1 s + \beta_0}.$$

The input and output corresponding to the `i`th experiment is stored in `u{i}` and `y{i}`, respectively, for `i`$\in \{1,2,\dots,25\}$. The sampling time is in the variable `Ts`.

MATLAB® Hint 16. `merge` allows one to combine data from multiple experiments... ► p. 46

1. Construct an estimate $\hat{H}_{\text{all}}(s)$ of the transfer function based on the data from *all* the 25 experiments.

   Hint: Use `tfest` and `merge`.

2. Construct 25 estimates $\hat{H}_i(s)$ of the transfer function, each based on the data from *a single* experiment. Use the estimate $\hat{H}_{\text{all}}(s)$ as a nominal model and compute additive and multiplicative uncertainty bounds that include all the $\hat{H}_i(s)$. Justify your answer with plots like those shown in Figures 8.4 and 8.6. □

8.3 (Noisy identification in discrete time)**.** 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. □

8.4 (Small gain)**.** For the nominal process model and multiplicative uncertainty that you obtained in Exercise 8.2, use the small gain condition in (8.10) to verify if the following controllers achieve stability for all admissible process models:

$$C_1(s) = 1000 \qquad C_2(s) = \frac{10}{s+.01} \qquad C_3(s) = \frac{10}{s-.01}$$
$$C_4(s) = \frac{100(s+1)}{(s+.01)(s+3)^2} \qquad C_5(s) = \frac{1000(s+1)}{(s+.01)(s^2+6s+18)} \qquad C_6(s) = \frac{1000}{(s+.01)(s^2+6s+25)}$$

Justify your answers with Bode plots like those in Figures 8.11 and 8.12.

Hint: Remember that checking the small gain condition in (8.10) does not suffice, you also need to check if the controller stabilizes the nominal process model. □

8.5 (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." □

8.6 (Additive uncertainty)**.** Derive a small-gain condition similar to (8.10) for additive uncertainty. Check if the mnemonic in Sidebar 8.3 still applies.

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

$$L(s) = C(s)P(s) = C(s)\big(P_0(s) + \Delta_{\text{a}}(s)\big) = L_0(s) + C(s)\Delta_{\text{a}}(s),$$

which differs from $L_0(s)$ by $C(s)\Delta_{\text{a}}(s)$. □

Lecture 9

Control design by loop shaping

The aim of this lecture is to show how to do control design taking into account unmodeled dynamics. The loop-shaping control design method will be used for this purpose.

Contents

9.1 The Loop-shaping Design Method

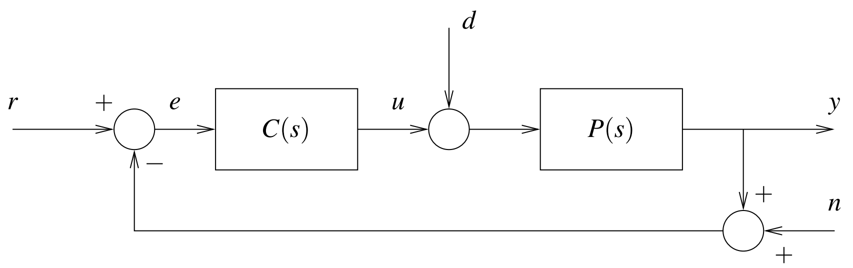


Figure 9.1. Closed-loop system

The goal of this lecture 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 system in Figure 9.1 into constraints on the open-loop gain

Note. The loop-shaping design method is covered extensively, e.g., in [5].

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

The controller $C(s)$ is then designed so that the open-loop gain $L(s)$ satisfies these constraints. The shaping of $L(s)$ can be done using the classical methods briefly mentioned in Section 9.3 and explained in much greater detail in [5, Chapter 6.7]. However, it can also be done using LQR state feedback, as discussed in Section 11.5, or using LQG/LQR output feedback controllers, as we shall see in Section 12.6.

9.2 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 $L(s)$.

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 $-180°$ at the cross-over frequency ω_c, i.e., at the frequency for which

Notation. The distance between the phase of $L(j\omega_c)$ and $-180°$ is called the *phase margin*. See Figure 9.2.

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

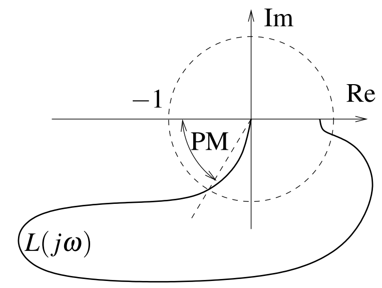


Figure 9.2. Phase margin (PM).

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 $L(s)$ has a pole at the origin, an additional real pole, and no zeros:

Note. 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.

$$L(s) = \frac{k}{s(s+p)}, \qquad p > 0.$$

They are useful to determine the value of the phase margin needed to obtain the desired overshoot:

Phase margin	overshoot
$45°$	$\leqslant 15\%$
$60°$	$\leqslant 10\%$
$65°$	$\leqslant 5\%$

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

Note. Typically one wants to track low frequency references, which justifies the requirement for equation (9.1) to hold in an interval of the form $[0, \omega_T]$.

$$\frac{|E(j\omega)|}{|R(j\omega)|} \leqslant k_T, \quad \forall \omega \in [0, \omega_T], \tag{9.1}$$

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 9.1,

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

Therefore (9.1) is equivalent to

$$\frac{1}{|1 + L(j\omega)|} \leqslant k_T, \quad \forall \omega \in [0, \omega_T] \quad \Leftrightarrow \quad |1 + L(j\omega)| \geqslant \frac{1}{k_T}, \quad \forall \omega \in [0, \omega_T]. \tag{9.2}$$

This condition is guaranteed to hold by requiring that

Note 22. Why? ► p. 109

$$\boxed{|L(j\omega)| \geqslant \frac{1}{k_T} + 1, \quad \forall \omega \in [0, \omega_T],} \tag{9.3}$$

which is a simple bound on the magnitude of the Bode plot of $L(s)$, as shown in Figure 9.3.

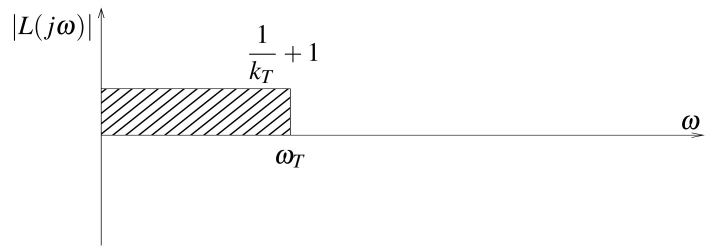


Figure 9.3. Reference tracking specifications for the loop shaping design method

Note 22 (Condition (9.3))**.** For (9.2) to hold, we need the distance from $L(j\omega)$ to the point -1 to always exceed $1/k_T$. As one can see in Figure 9.4, this always hold if $L(j\omega)$ remains outside a circle of radius $1+1/k_T$. □

Note. It is clear from Figure 9.4 that asking for $L(j\omega)$ to stay outside the larger circle of radius $1+1/k_T$ can be much more than what we need, which is just for $L(j\omega)$ to be remain outside the smaller circle with radius $1/k_T$ centered at the point -1. However, when $1/k_T$ is very large the two circles in Figure 9.4 are actually very close to each other and condition (9.3) is actually not very conservative.

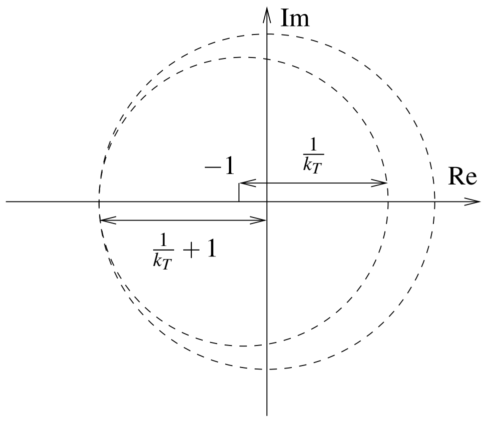


Figure 9.4. Justification for why (9.3) guarantees that (9.2) holds.

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

$$\frac{|Y(j\omega)|}{|D(j\omega)|} \leqslant k_D, \quad \forall \omega \in [\omega_{D_1}, \omega_{D_2}], \tag{9.4}$$

Note. Typically one wants to reject low-frequency disturbances and therefore ω_{D_1} and ω_{D_2} in (9.4) generally take low values. Occasionally, disturbances are known to have very narrow bandwidths, which case ω_{D_1} and ω_{D_2} are very close to each other, but may not necessarily be very small.

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 9.1,

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

and therefore (9.4) is equivalent to

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

This condition is guaranteed to hold by requiring that

$$\boxed{|L(j\omega)| \geqslant \frac{|P_0(j\omega)|}{k_D} + 1, \quad \forall \omega \in [\omega_{D_1}, \omega_{D_2}],} \tag{9.5}$$

Note. The reasoning needed to understand (9.5) is basically the same as for (9.3) but with $\frac{|P_0(j\omega)|}{k_D}$ instead of $\frac{1}{k_T}$.

which is again a simple bound on the magnitude of the Bode plot of $L(s)$, as shown in Figure 9.5.

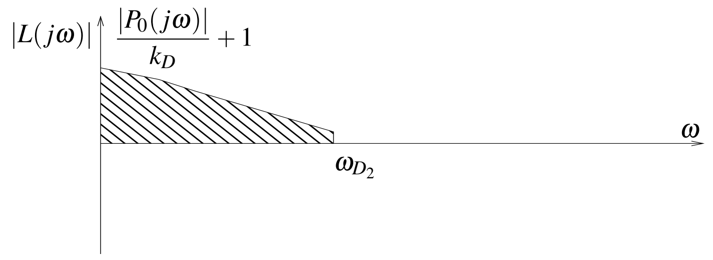


Figure 9.5. Disturbance rejection specifications for the loop shaping design method

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

Note. Typically one needs to reject high frequencies noise, which justifies the requirement for equation (9.6) to hold in an interval of the form $[\omega_N, \infty)$.

$$\frac{|Y(j\omega)|}{|N(j\omega)|} \leqslant k_N, \quad \forall \omega \in [\omega_N, \infty), \tag{9.6}$$

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 9.1,

$$Y(s) = -\frac{L(s)}{1+L(s)}N(s), \tag{9.7}$$

and therefore (9.6) is equivalent to

$$\frac{|L(j\omega)|}{|1+L(j\omega)|} \leqslant k_N, \quad \forall \omega \in [\omega_N, \infty) \quad \Leftrightarrow \quad \left|1+\frac{1}{L(j\omega)}\right| \geqslant \frac{1}{k_N}, \quad \forall \omega \in [\omega_N, \infty). \tag{9.8}$$

This condition is guaranteed to hold by requiring that

Note. The reasoning needed to understand (9.9) is basically the same as for (9.3) but with $\frac{1}{L(j\omega)}$ instead of $L(j\omega)$.

$$\left|\frac{1}{L(j\omega)}\right| \geqslant \frac{1}{k_N}+1, \quad \forall \omega \in [\omega_N, \infty) \quad \Leftrightarrow \quad \boxed{|L(j\omega)| \leqslant \frac{k_N}{1+k_N}, \quad \forall \omega \in [\omega_N, \infty),} \tag{9.9}$$

which is again a simple bound on the magnitude of the Bode plot of $L(s)$, as shown in Figure 9.6.

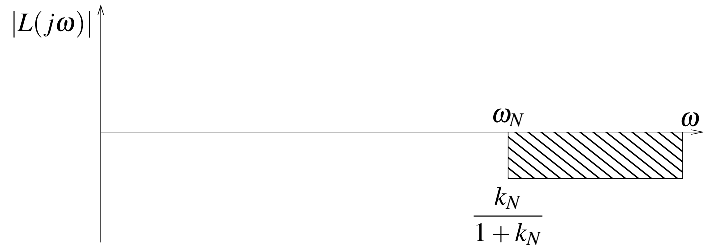


Figure 9.6. Noise rejection specifications for the loop shaping design method

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 $\ell_m(\omega)$. In view of the small gain condition in (8.10), this can be guaranteed by requiring that

$$\frac{|L(j\omega)|}{|1+L(j\omega)|} \leqslant \frac{1}{\ell_m(\omega)}, \quad \forall \omega, \tag{9.10}$$

which is equivalent to

$$\left|1+\frac{1}{L(j\omega)}\right| \geqslant \ell_m(\omega), \quad \forall\omega. \tag{9.11}$$

Note. While the condition in (9.11) is similar to the one in (9.8), we now need it to hold *for every frequency* and we no longer have the "luxury" of simply requiring $|L(j\omega)|$ to be small for every frequency.

We have *two options* to make sure that (9.11) hold:

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

$$\left|\frac{1}{L(j\omega)}\right| \geqslant \ell_m(\omega)+1, \quad \Leftrightarrow \quad \boxed{|L(j\omega)| \leqslant \frac{1}{1+\ell_m(\omega)} < 1.} \tag{9.12}$$

Note. To derive (9.12) simply re-write (9.8)–(9.9) with $1/k_N$ replaced by $\ell(\omega)$.

This condition will hold for frequencies for which we can make $|L(j\omega)|$ small; typically at high frequencies, as shown in Figure 9.7.

2. Alternatively, (9.11) may hold even when $|L(j\omega)|$ large, provided that $\ell_m(\omega)$ is small. In particular, since

$$\left|1+\frac{1}{L(j\omega)}\right| \geqslant 1-\left|\frac{1}{L(j\omega)}\right|, \tag{9.13}$$

the condition (9.11) holds, provided that we require that

$$1-\left|\frac{1}{L(j\omega)}\right| \geqslant \ell_m(\omega) \quad \Leftrightarrow \quad \boxed{|L(j\omega)| \geqslant \frac{1}{1-\ell_m(\omega)} > 1.} \tag{9.14}$$

This condition will hold for frequencies for which we can make $|L(j\omega)|$ large; typically at low frequencies, as shown in Figure 9.7.

Note. Why does (9.13) hold? Because of the triangular inequality: $1=\left|1+\frac{1}{L(j\omega)}-\frac{1}{L(j\omega)}\right| \leqslant \left|1+\frac{1}{L(j\omega)}\right|+\left|\frac{1}{L(j\omega)}\right|$ form which (9.13) follows.

From the two conditions (9.12)–(9.14), one generally needs to be mostly concerned about (9.12). This is because when $\ell_m(\omega)$ is small, (9.11) will generally hold. However, when $\ell_m(\omega)$ is large, for (9.11) to hold we need $L(j\omega)$ to be small, which corresponds precisely to the condition (9.12). Hopefully, $\ell_m(\omega)$ will only be large at high frequencies, for which we do not need (9.3) or (9.5) to hold. In view of this, it is common to design a controller ignoring (9.14) and, after the controller has been designed, verify that it satisfies (9.11) for every frequency.

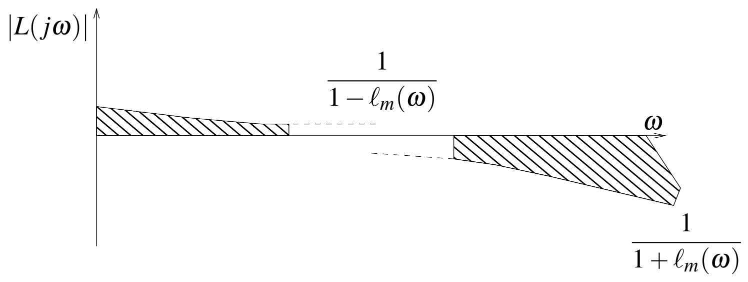


Figure 9.7. Robustness with respect to multiplicative uncertainty constraints for loop shaping design

Summary

Table 9.1 and Figure 9.8 summarize the constraints on the open-loop gain $L(j\omega)$ discussed above.

Attention! The conditions derived above for the open-loop gain $L(j\omega)$ 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 (9.12)–(9.14) that arise from robustness with respect to multiplicative uncertainty, because around the crossover frequency $|L(j\omega_c)| \approx 1$ will not satisfy either of these conditions, but it will generally satisfy the original condition (9.10) (as long as $\ell_m(\omega)$ is not too large). □

closed-loop specification	**open-loop constraint**
overshoot $\leqslant 10\%$ ($\leqslant 5\%$)	phase margin $\geqslant 60$ deg ($\geqslant 65$ deg)
$\dfrac{\lvert E(j\omega)\rvert}{\lvert R(j\omega)\rvert} \leqslant k_T, \quad \forall \omega \in [0, \omega_T]$	$\lvert L(j\omega)\rvert \geqslant \dfrac{1}{k_T} + 1, \quad \forall \omega \in [0, \omega_T]$
$\dfrac{\lvert Y(j\omega)\rvert}{\lvert D(j\omega)\rvert} \leqslant k_D, \quad \forall \omega \in [\omega_{D_1}, \omega_{D_2}]$	$\lvert L(j\omega)\rvert \geqslant \dfrac{\lvert P_0(j\omega)\rvert}{k_D} + 1, \quad \forall \omega \in [\omega_{D_1}, \omega_{D_2}]$
$\dfrac{\lvert Y(j\omega)\rvert}{\lvert N(j\omega)\rvert} \leqslant k_N, \quad \forall \omega \in [\omega_N, \infty)$	$\lvert L(j\omega)\rvert \leqslant \dfrac{k_N}{1+k_N}, \quad \forall \omega \in [\omega_N, \infty)$
$\dfrac{\lvert L(j\omega)\rvert}{\lvert 1+L(j\omega)\rvert} \leqslant \dfrac{1}{\ell_m(\omega)}, \quad \forall \omega$	$\lvert L(j\omega)\rvert \leqslant \dfrac{1}{1+\ell_m(\omega)} \left(\text{or } \lvert L(j\omega)\rvert \geqslant \dfrac{1}{1-\ell_m(\omega)} \right)$

Table 9.1. Summary of the relationship between closed-loop specifications and open-loop constraints for the loop shaping design method

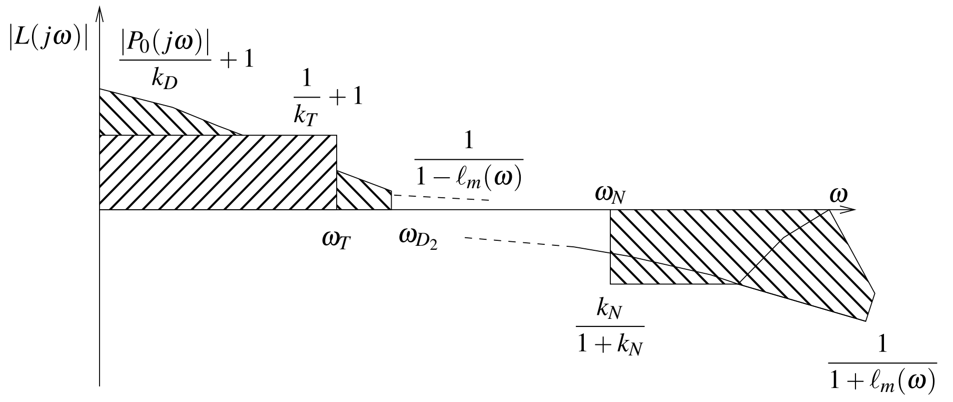


Figure 9.8. Typical open-loop specifications for the loop shaping design method

9.3 Open-loop Gain Shaping

Note. When the process does not have an integrator and we want zero steady-state error, it is common to include an integrator in the controller and thus start instead with $C(s) = 1/s$.

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

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

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.

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.

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.

2. *Lead compensation.* Multiplying the controller by a lead block with transfer function

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

increases the phase margin when placed at the cross-over frequency. Figure 9.9a shows the Bode plot of a lead compensator.

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

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

decreases the high-frequency gain. Figure 9.9b shows the Bode plot of a lag compensator.

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.

This design method has two important limitations:

1. When the cross-over region is very narrow, it may be hard to construct lead-lag compensators that achieve any reasonable phase margin, or even that are able to achieve closed-loop stability at all.

2. This procedure does not generalize easily to feedback loops with multiple sensors and/or multiple actuators.

9.4 Exercises

9.1 (Loop-shape 1)**.** Consider again the nominal process model and multiplicative uncertainty that you obtained in Exercise 8.2. 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 then 60degrees,

3. steady-state error for sinusoidal inputs with frequencies $\omega < 0.1$rad/sec smaller than 1/10 ($-20dB$).

4. Rise time faster than or equal to 1.5sec □

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

$$P_0(s) = \frac{1}{s(1+s/5)(1+s/20)}, \qquad \ell_{\text{m}}(\omega) = |L(j\omega)|,$$

where

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

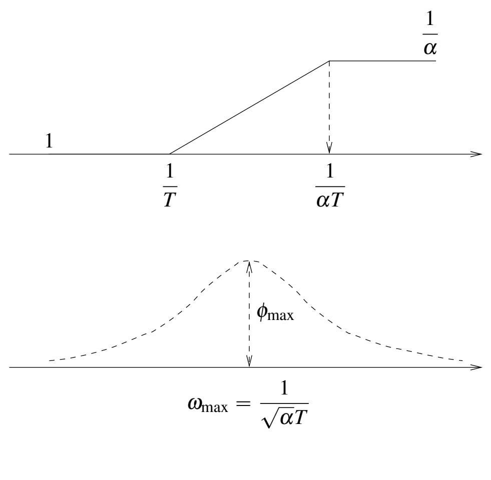
Use loop shaping to design a controller that achieves stability *for all admissible process models* and that exhibits:

1. steady-state error to a ramp input no larger than .01,

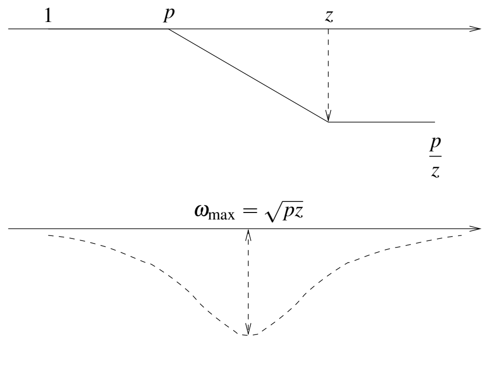
2. Phase Margin no smaller then 45degrees,

3. steady-state error for sinusoidal inputs with frequencies $\omega < 0.2$rad/sec smaller than 1/250 (-50dB), and

4. attenuation of measurement noise by at least a factor of 100 (-40dB) for frequencies greater than 200rad/sec. □



(a) Lead



(b) Lag

Figure 9.9. 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.

Further reading A more extensive coverage of LQG/LQR controller design can be found, e.g., in [8].

Lecture 10

Review of State-space models

Contents

10.1 State-space Models

Consider the system in Figure 10.1 with m inputs and k outputs. A *state-space* model for this system

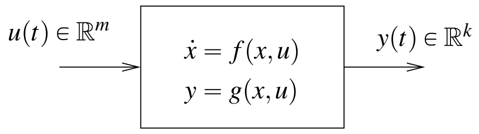


Figure 10.1. System with m inputs and k outputs

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

$$\dot{x} = f(x,u), \qquad y = g(x,u). \tag{10.1}$$

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, (10.1) has the special form

MATLAB® Hint 35. `ss(A,B,C,D)` creates a LTI state-space model with realization (10.2). ► p. 122

$$\dot{x} = Ax + Bu, \qquad y = Cx + Du, \tag{10.2}$$

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 10.1 (Aircraft roll-dynamics)**.** Figure 10.2 shows the roll-angle dynamics of an aircraft [12, p. 381]. Defining

$$x := \begin{bmatrix} \theta & \omega & \tau \end{bmatrix}'$$

we conclude that

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

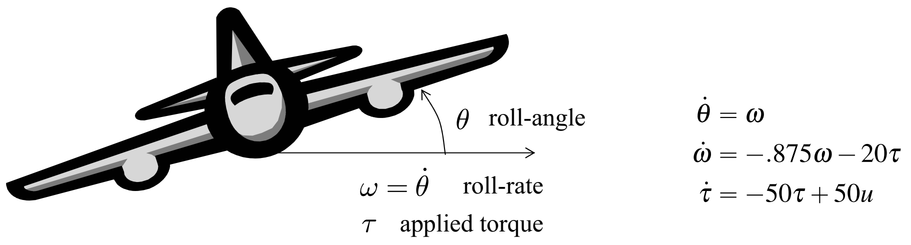

θ roll-angle
$\omega = \dot{\theta}$ roll-rate
τ applied torque
$\dot{\theta} = \omega$
$\dot{\omega} = -.875\omega - 20\tau$
$\dot{\tau} = -50\tau + 50u$

Figure 10.2. Aircraft roll-angle dynamics

with

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

If we have both θ and ω available for control, we can define

$$y := \begin{bmatrix} \theta & \omega \end{bmatrix}' = Cx + Du$$

with

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

10.2 Input-output Relations

Note 2. The *(unilateral) Laplace transform* of a signal $x(t)$ is given by

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

See [5, Appendix A] for a review of Laplace transforms.

The *transfer-function* of this system can be found by taking Laplace transforms of (10.2):

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

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

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

and therefore

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

Defining

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

we conclude that

$$Y(s) = T(s)U(s). \tag{10.3}$$

To emphasize the fact that $T(s)$ is a $k \times m$ matrix, we call it the *transfer-matrix* of the system (10.2).

The relation (10.3) 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 (10.2) in the *time domain* is given by

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-s)}Bu(s)ds, \tag{10.4}$$

$$y(t) = Ce^{At}x(0) + \int_0^t Ce^{A(t-s)}Bu(s)ds + Du(t), \qquad \forall t \geqslant 0. \tag{10.5}$$

MATLAB® Hint 1. `tf(num,den)` creates a transfer-function with numerator and denominator specified by `num`, `den`. ► p. 13

MATLAB® Hint 2. `zpk(z,p,k)` creates a transfer-function with zeros, poles, and gain specified by `z`, `p`, `k`. ► p. 13

MATLAB® Hint 36. `tf(sys_ss)` and `zpk(sys_ss)` compute the transfer-function of the state-space model `sys_ss`. ► p. 122

Equation (10.4) is called the *variation of constants formula.*

Example 10.2 (Aircraft roll-dynamics)**.** The transfer-function for the state-space model in Example 10.1 is given by:

$$T(s) = \begin{bmatrix} \frac{-1000}{s(s+.875)(s+50)} \\ \frac{-1000}{(s+.875)(s+50)} \end{bmatrix}$$

□

MATLAB® Hint 37. `expm` computes the exponential of a matrix. ► p. 122

10.3 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 (10.2) whose transfer-matrix is precisely $T(s)$. The model (10.2) is called a *realization* of $T(s)$.

MATLAB® Hint 38. `ss(sys_tf)` computes a realization of the transfer-function `sys_tf`. ► p. 122

Attention! Realizations are not unique, i.e., several state-space models may have the same transfer function.

10.4 Controllability and Observability

The system (10.2) 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 (10.2) from x_i to x_f in the interval of time $0 \leqslant t \leqslant T$, 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

$$\mathscr{C} := \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}.$$

The system is controllable if and only if this matrix has rank equal to the size n of the state vector.

MATLAB® Hint 39. `ctrb(sys)` computes the controllability matrix of the state-space system `sys`. Alternatively, one can use directly `ctrb(A,B)`. ► p. 123

MATLAB® Hint 40. `rank(M)` computes the rank of a matrix `M`.

The system (10.2) 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 $0 \leqslant t \leqslant T$, i.e., one can solve

$$y(t) = Ce^{At}x(0) + \int_0^t Ce^{A(t-s)}Bu(s)ds + Du(t), \qquad \forall 0 \leqslant t \leqslant T,$$

uniquely for the unknown $x(0)$. To determine if a system is observable, one can compute the *observability matrix*, which is defined by

$$\mathscr{O} := \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}.$$

The system is observable if and only if this matrix has rank equal to the size n of the state vector.

MATLAB® Hint 41. `obsv(sys)` computes the controllability matrix of the state-space system `sys`. Alternatively, one can use directly `obsv(A,C)`. ► p. 123

Example 10.3 (Aircraft roll-dynamics)**.** The controllability and observability matrices for the state-space model in Example 10.1 are given by:

$$\mathscr{C} = \begin{bmatrix} 0 & 0 & -1000 \\ 0 & -1000 & 50875 \\ 50 & -2500 & 125000 \end{bmatrix}, \qquad \mathscr{O} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & -.875 & -20 \\ 0 & -.875 & -20 \\ 0 & .7656 & 1017.5 \end{bmatrix}.$$

Both matrices have rank 3 so the system is both controllable and observable. □

10.5 Stability

MATLAB® Hint 42. `eig(A)` computes the eigenvalues of the matrix `A`. ► p. 123

The system (10.2) 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.,

$$\|u(t)\| \leqslant c_1,\ \forall t \geqslant 0 \quad \Rightarrow \quad \|y(t)\| \leqslant c_2,\ \|x(t)\| \leqslant c_3 \quad \forall t \geqslant 0.$$

Moreover, if $u(t)$ converges to zero as $t \to \infty$, then $x(t)$ and $y(t)$ also converge to zero as $t \to \infty$.

Example 10.4 (Aircraft roll-dynamics)**.** The eigenvalues of the matrix the A matrix for the state-space model in Example 10.1 are $\{0, -.875, -50\}$ so the system is not asymptotically stable. □

10.6 MATLAB® Hints

MATLAB® Hint 35 (`ss`)**.** The command `sys_ss=ss(A,B,C,D)` assigns to `sys_ss` a MATLAB LTI state-space model with realization

$$\dot{x} = \mathtt{A}x + \mathtt{B}u, \qquad y = \mathtt{C}x + \mathtt{D}u.$$

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. □

MATLAB® Hint 36 (`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 35.

`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. □

MATLAB® Hint 38 (`ss`)**.** The command `ss(sys_tf)` computes the state-space model of the transfer function `sys` specified as in Matlab Hints 1 or 2. □

MATLAB® Hint 37 (`expm`)**.** The command `expm(M)` computes the matrix exponential $e^{\mathtt{M}}$. With the symbolic toolbox, this command can be used to compute e^{At} symbolically as follows:

```
syms t
expm(A*t)
```

The first command defines `t` as a symbolic variable and the second computes $e^{\texttt{At}}$ (assuming that the matrix `A` has been previously defined). □

MATLAB® Hint 39 (`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 41 (`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 42 (`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. □

Lecture 11

Linear Quadratic Regulation (LQR)

This lecture introduces the Linear Quadratic Regulator (LQR) optimal control problem under state feedback. This control method leads to several desirable robustness properties and can be used for loop-shaping.

Contents

11.1 Feedback Configuration

Figure 11.1 shows the feedback configuration for the *Linear quadratic regulation (LQR) problem*.

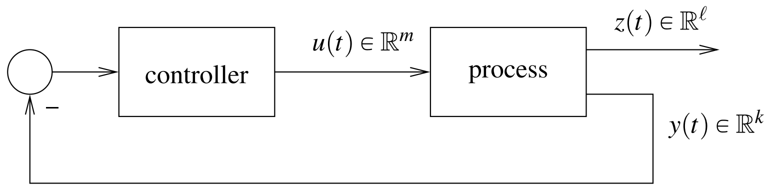


Figure 11.1. Linear quadratic regulation (LQR) feedback configuration. Note the *negative feedback* and the *absence of a reference signal*. Reference signals will be introduced in Lecture 13.

In this configuration, the state-space model of the process is of the form

$$\dot{x} = Ax + Bu, \qquad y = Cx, \qquad z = Gx + Hu. \tag{11.1}$$

Note. Recall from Section 10.2 that for this state-space model the transfer function from u to y is given by $C(sI - A)^{-1}B$ and the transfer function from u to z is given by $G(sI - A)^{-1}B + H$.
► p. 120

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 feedback. If the controller transfer-matrix is $C(s)$, we have

$$U(s) = -C(s)Y(s),$$

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.

   Sometimes $z(t) = y(t)$, which means that our control objective is to make the whole measured output very small. However, when the measured output $y(t)$ is a vector, often one simply needs to make one of its entries, say $y_1(t)$, 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 11.5 we will study the impact of this choice in the performance of the closed-loop.

Note 23. In the context of Example 10.1, one could imagine that, while both and θ and ω can be measured, one may mostly want to regulate the roll angle θ. ► p. 119

11.2 Optimal Regulation

The LQR problem is defined as follows:

Problem 11.1 (Optimal LQR)**.** Find the controller transfer-matrix $C(s)$ that makes the following criterion as small as possible

$$J_{\text{LQR}} := \int_0^\infty \|z(t)\|^2 + \rho \, \|u(t)\|^2 dt, \tag{11.2}$$

where ρ is a positive constant. □

Notation 5. Given an m-vector,

$$v = \begin{bmatrix} v_1 \; v_2 \; \cdots \; v_m \end{bmatrix},$$

$\|v\|$ denotes the Euclidean norm of v, i.e.,

$$\|v\| = v'v = \Big(\sum_{i=1}^m v_i^2\Big)^{\frac{1}{2}}.$$

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 input*. In LQR one seeks a controller that minimizes both energies. However, decreasing the energy of the controlled output will require a large control input, and a small control input will lead to large controlled outputs. The role of the constant ρ is to establish a trade-off between these conflicting goals.

1. When we chose ρ very large, the most effective way to decrease J_{LQR} is to employ a small control input, at the expense of a large controlled output.

2. When we chose ρ very small, the most effective way to decrease J_{LQR} is to obtain a very small controlled output, even if this is achieved at the expense of employing 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

$$J_{\text{LQR}} := \int_0^\infty z(t)'Qz(t) + \rho u'(t)Ru(t)dt, \tag{11.3}$$

Note 24. 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. 134

where Q is an $\ell \times \ell$ symmetric positive-definite matrix, R an $m \times m$ symmetric positive-definite matrix, and ρ a positive constant.

Note 25. Many 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. 135

Bryson's rule A first choice for the matrices Q and R in (11.3) is given by the *Bryson's rule* [5, p. 537]: select Q and R diagonal with

$$Q_{ii} = \frac{1}{\text{maximum acceptable value of } z_i^2}, \qquad i \in \{1, 2, \dots, \ell\}$$
$$R_{jj} = \frac{1}{\text{maximum acceptable value of } u_j^2}, \qquad j \in \{1, 2, \dots, m\},$$

Note. Bryson's rule was widely used when LQR first appeared, but became "obsolete" when robust control and loop shaping appeared. In Section 11.5 we will discuss systematic methods to chose the weights in the LQR criterion.

► p. 130

which corresponds to the following criteria

$$J_{\text{LQR}} := \int_0^\infty \Big(\sum_{i=1}^{\ell} Q_{ii} z_i(t)^2 + \rho \sum_{j=1}^{m} R_{jj} u_j(t)^2 \Big) dt.$$

In essence the Bryson's rule scales the variables that appear in J_{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.

11.3 State-Feedback LQR

In the *state-feedback* version of the LQR problem (Figure 11.2), we assume that the whole state x can be measured and therefore it is available for control.

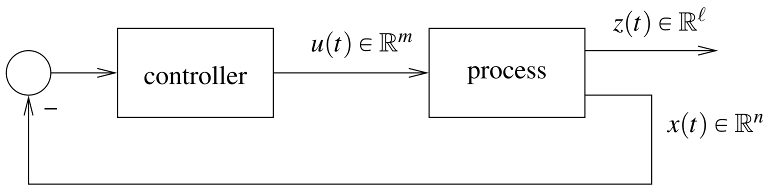


Figure 11.2. Linear quadratic regulation (LQR) with state feedback

Solution to the optimal state-feedback LQR Problem 11.1. The optimal state-feedback LQR controller for the criteria (11.3) is a simple matrix gain of the form

MATLAB® Hint 43. `lqr` computes the optimal state-feedback controller gain K.

► p. 133

$$u = -Kx \tag{11.4}$$

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).* □

11.4 Stability and Robustness

The state-feedback control law (11.4), results in a closed-loop system of the form

$$\dot{x} = (A - BK)x.$$

Note 26. A system $\dot{x} = Ax + Bu$ is *asymptotically stable* when all eigenvalues of A have negative real parts. See Section 10.5. ► p. 122

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:

Note 27. The definitions and tests for controllability and observability are reviewed in Section 10.4. ► p. 121

1. The system (11.1) is controllable.
2. The system (11.1) is observable when we ignore y and regard z as the sole output:

$$\dot{x} = Ax + Bu, \qquad z = Gx + Hu.$$

Attention! When selecting the measured output z, it is important to verify that the observability condition is satisfied.

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 11.3). The state-space model from u to $\bar{u}$ is given by

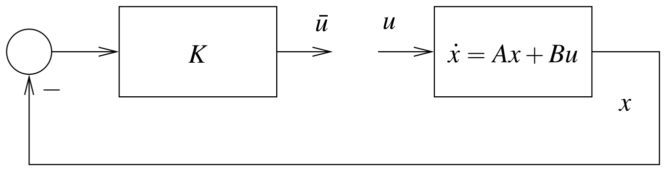


Figure 11.3. State-feedback open-loop gain

$$\dot{x} = Ax + Bu, \qquad \bar{u} = -Kx,$$

Note. In a *negative-feedback* configuration, the output of the process is multiplied by -1 before being connected to the input of the controller. Often a reference signal is also added to the input of the controller.

which corresponds to the following open-loop *negative* feedback $m \times m$ transfer-matrix

$$L(s) = K(sI - A)^{-1}B. \tag{11.5}$$

We focus our attention in single-input processes ($m = 1$), for which $L(s)$ is a scalar transfer-function and the following holds:

Note. LQR controllers also exhibit robustness properties for multiple-input processes. However, in this case $L(s)$ is a $m \times m$ transfer-matrix and one needs a *multi-variable Nyquist criterion.*

Kalman's Inequality. *When $H'G = 0$, the Nyquist plot of $L(j\omega)$ does not enter a circle of radius one around -1, i.e.,*

$$|1 + L(j\omega)| \geqslant 1, \qquad \forall \omega \in \mathbb{R}. \qquad \square$$

Kalman's Inequality is represented graphically in Figure 11.4 and has several significant implications, which are discussed next.

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* $20\log_{10}(.5) = -6dB$.

Phase margin If the process' phase increases by $\theta \in [-60, 60]$ degrees, its Nyquist plots rotates by θ but the number of encirclements still does not change. This corresponds to a *phase margin of* ± 60 *degrees.*

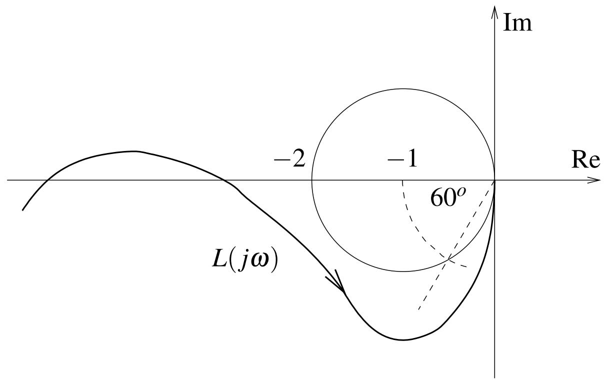


Figure 11.4. Nyquist plot for a LQR state-feedback controller

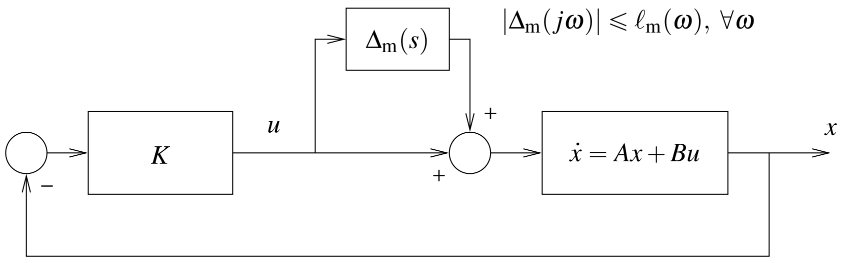

$|\Delta_m(j\omega)| \leqslant \ell_m(\omega),\ \forall\omega$
$\Delta_m(s)$
K
u
$+$
$+$
$-$
$\dot{x} = Ax + Bu$
x

Figure 11.5. Unity feedback configuration with multiplicative uncertainty

Multiplicative uncertainty Kalman's inequality guarantees that

Note 28. Why?... ► p. 135

$$\left|\frac{L(j\omega)}{1+L(j\omega)}\right| \leqslant 2. \tag{11.6}$$

Since, we known that the closed-loop system in Figure 11.5 remains stable for every multiplicative uncertainty block $\Delta_m(j\omega)$ with norm smaller than $\ell_m(\omega)$, as long as

$$\left|\frac{L(j\omega)}{1+L(j\omega)}\right| < \frac{1}{\ell_m(\omega)}, \tag{11.7}$$

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

$$\left|\frac{L(j\omega)}{1+L(j\omega)}\right| \leqslant 2 < \frac{1}{\ell_m(\omega)}.$$

However, much larger additive uncertainties may be admissible: e.g., when $L(j\omega) \gg 1$, (11.7) will hold for $\ell_m(\omega)$ almost equal to 1; and when $L(j\omega) \ll 1$, (11.7) will hold for $\ell_m(\omega)$ almost equal to $1/L(j\omega) \gg 1$.

Attention! Kalman's inequality is only valid when $H'G = 0$. When $H'G \neq 0$, LQR controllers can be significantly less robust. This limits to same extent the controlled outputs that can be placed in z. To explore this, consider the following examples:

Note 29. A simple condition to have $H'G = 0$ (that is sufficient but not necessary) is for the transfer function $T(s) = G(sI - A)^{-1}B + H$ from u to z to be strictly proper, as this means that $H = 0$. ► p. 130

1. Suppose that u is a scalar and $z = z_1$ is also a scalar, with the transfer function from u to $z = z_1$ given by
$$\frac{1}{s+1}.$$
Since this transfer function is strictly proper, we have $H = 0$ and therefore $H'G = 0$.

2. Suppose now that we had to z a second controlled output $z_2 = \dot z_1$. In this case, the transfer function from the scalar input u to the vector $z = \begin{bmatrix} z_1 & z_2 \end{bmatrix}'$ is given by
$$\begin{bmatrix} \frac{1}{s+1} \\ \frac{s}{s+1} \end{bmatrix}.$$
In this case, the H matrix is given by
$$H = \lim_{s\to\infty} \begin{bmatrix} \frac{1}{s+1} \\ \frac{s}{s+1} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$
and we no longer have $H'G = 0$. □

Note. To verify that we do have $H'G \neq 0$, we need to actually determine G: A realization for $\begin{bmatrix} \frac{1}{s+1} \\ \frac{s}{s+1} \end{bmatrix} = \begin{bmatrix} \frac{1}{s+1} \\ 1 - \frac{1}{s+1} \end{bmatrix}$ turns out to be one-dimensional with $A = -1$, $B = 1$, $G = \left[\begin{smallmatrix} 1 \\ -1 \end{smallmatrix}\right]$, $H = \left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$, for which we indeed have $H'G = -1 \neq 0$.

Note 29 (Strictly proper transfer functions)**.** Recall that the transfer function of the state-space model
$$\dot x = Ax + Bu, \qquad z = Gx + H$$
is given by
$$T(s) = G(sI - A)^{-1}B + H.$$
As we make $s \to \infty$, the term sI dominates over A in $(sI - A)$ and we get:
$$\begin{aligned} \lim_{s\to\infty} T(s) &= \lim_{s\to\infty} G(sI)^{-1}B + H \\ &= \lim_{s\to\infty} \frac{1}{s} GB + H \\ &= H. \end{aligned}$$
This shows that when the transfer function $T(s) = G(sI - A)^{-1}B + H$ is strictly proper, i.e., when it has more poles than zeros, we have $\lim_{s\to\infty} T(s) = 0$ and therefore $H = 0$. Conversely, if the transfer function is not strictly proper, we will necessarily have $H \neq 0$. However, this may not mean that $G'H \neq 0$. □

11.5 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

Note. Recall that the loop-gain (11.5) that appears in Kalman's inequality is from u to $\bar u$ in Figure 11.3.

$$J_{\text{LQR}} := \int_0^\infty \|z(t)\|^2 + \rho\, u(t)^2 dt$$
and a scalar loop gain $L(s) = K(sI - A)^{-1}B$.

Low-frequency open-loop gain For the range of frequencies for which $|L(j\omega)| \gg 1$ (typically low frequencies), we have that

$$|L(j\omega)| \approx \frac{\|P_z(j\omega)\|}{\sqrt{H'H+\rho}}$$

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 $z = z_1$, with $z_1 := C_1 x$ scalar, we have $G = C_1$ and $H = 0$, leading to

$$|L(j\omega)| \approx \frac{|P_1(j\omega)|}{\sqrt{\rho}}. \tag{11.8}$$

   where

$$P_1(s) := C_1(sI-A)^{-1}B$$

   is the transfer function from the control input u to the output z_1. In this case,

   (a) the "shape" of the magnitude of the open-loop gain $L(j\omega)$ is determined by the magnitude of the transfer function from the control input u to the output z_1;

   (b) the parameter ρ moves the magnitude Bode plot up and down (see Figure 11.6a).

2. When $z = \begin{bmatrix} z_1 & \gamma \dot{z}_1 \end{bmatrix}'$, we can show that

$$|L(j\omega)| \approx \frac{|1+j\gamma\omega|\,|P_1(j\omega)|}{\sqrt{H'H+\rho}}. \tag{11.9}$$

   In this case the low-frequency open-loop gain mimics the process transfer function from u to z_1, with an extra zero at $1/\gamma$ and scaled by $\frac{1}{\sqrt{H'H+\rho}}$. Thus

   (a) ρ moves the magnitude Bode plot up and down (more precisely $H'H+\rho$),

   (b) large values for γ 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 (see Figure 11.6b). However, this is often achieved at the expense of a slower response.

High-frequency open-loop gain For $\omega \gg 1$, we have that

$$|L(j\omega)| \approx \frac{c}{\omega\sqrt{\rho}},$$

for some constant c. We thus conclude the following:

1. LQR controllers always exhibit a high-frequency magnitude decay of -20dB/decade.

2. The cross-over frequency is approximately given by

$$\frac{c}{\omega_{\text{cross}}\sqrt{\rho}} \approx 1 \quad \Leftrightarrow \quad \omega_{\text{cross}} \approx \frac{c}{\sqrt{\rho}},$$

   which shows that the cross-over frequency is proportional to $1/\sqrt{\rho}$ and generally small values for ρ result in faster step responses.

MATLAB® Hint 44. `sigma(sys)` draws the norm-Bode plot of the system `sys`. ► p. 133

Note. Although the magnitude of $L(j\omega)$ mimics the magnitude of $P_1(j\omega)$, the phase of the open-loop gain $L(j\omega)$ always leads to a stable closed-loop with a phase margin of at least ± 60 degrees.

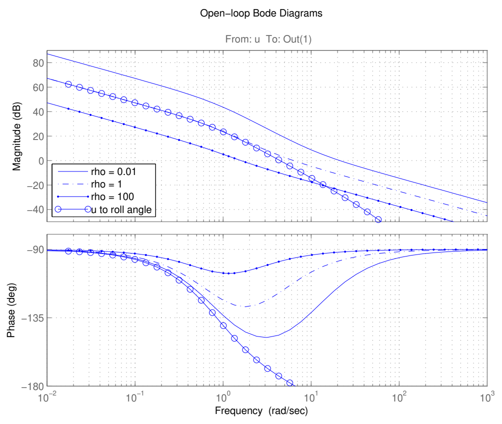
Note. The parameter ρ behaves like a proportional gain in classical loop shaping, but now we automatically get stability and phase margin without the need for any lead compensator. However, since ρ appears within a square root in (11.8), to move the magnitude plot down by say 20dBs we need to increase ρ by 100^2. In practice, large changes in ρ are needed to see significant effects in $|L(j\omega)|$.

Note 30. Why does $z = \begin{bmatrix} z_1 & \gamma \dot{z}_1 \end{bmatrix}'$ lead to (11.9)? ► p. 135

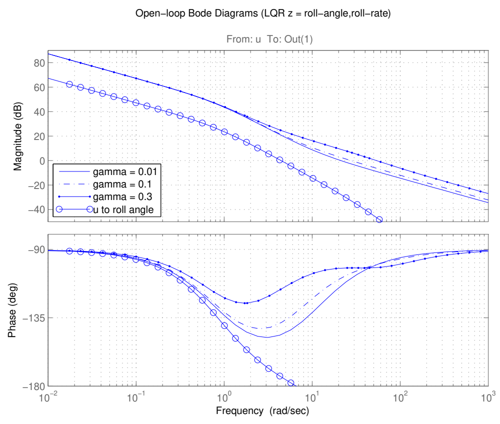
Note. The parameter ρ behaves like a proportional gain in classical loop shaping.

Note 31. The introduction of the output $\gamma \dot{z}_1$ has an effect similar to that of lead compensation in classical control (see Section 9.3). ► p. 112

Unfortunately, in LQR there is no equivalent to a lag compensator to decrease the high-frequency gain. We will see in Section 12.6 that this limitation of LQR can be overcome with output-feedback controllers.



(a) Open-loop gain for several values of ρ. This parameter allow us to move the whole magnitude Bode plot up and down.



(b) Open-loop gain for several values of γ. Larger values for this parameter result in a larger phase margin.

Figure 11.6. Bode plots for the open-loop gain of the LQR controllers in Example 11.1. As expected, for low frequencies the open-loop gain magnitude matches that of the process transfer function from u to θ (but with significantly lower/better phase) and at high-frequencies the gain's magnitude falls at -20dB/decade.

Attention! The (slow) -20dB/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 12.6 that this can actually be improved with output-feedback controllers. □

Note. This may appear counter intuitive as being able to measure the full state seems desirable. However, by introducing dynamics between measurements (including state measurements) and the control input u will enable us to filter-out measurement noise.

Example 11.1 (Aircraft roll-dynamics)**.** Figure 11.6 shows Bode plots of the open-loop gain $L(s) = K(sI-A)^{-1}B$ for several LQR controllers obtained for the aircraft roll-dynamics in Example 10.1. The controlled output was chosen to be $z := \begin{bmatrix}\theta & \gamma\dot\theta\end{bmatrix}'$, which corresponds to

$$G := \begin{bmatrix}1 & 0 & 0\\ 0 & \gamma & 0\end{bmatrix}, \qquad H := \begin{bmatrix}0\\0\end{bmatrix}.$$

The controllers were obtained with $R = 1$, $Q = I_{2\times 2}$, and several values for ρ and γ. Figure 11.6a shows the open-loop gain for several values of ρ and Figure 11.6b shows the open-loop gain for several values of γ.

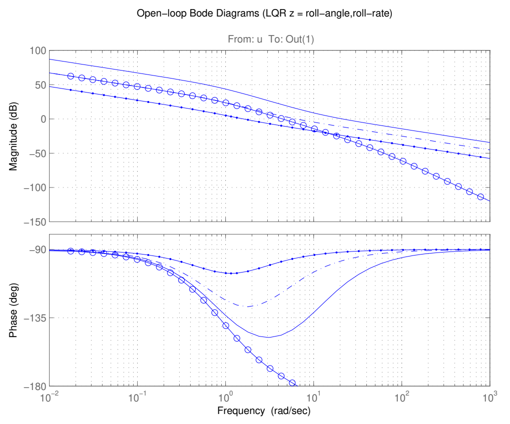
Figure 11.7 shows Nyquist plots of the open-loop gain $L(s) = K(sI-A)^{-1}B$ for different choices of the controlled output z. In Figure 11.7a $z := \begin{bmatrix}\theta & \dot\theta\end{bmatrix}'$, which corresponds to

$$G := \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}, \qquad H := \begin{bmatrix}0\\0\end{bmatrix}.$$

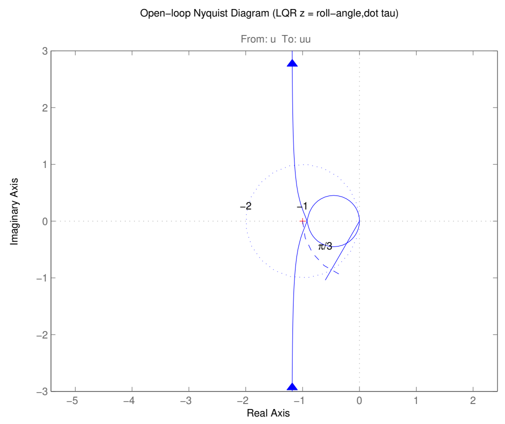
In this case, $H'G = \begin{bmatrix}0 & 0 & 0\end{bmatrix}$ and Kalman's inequality holds as can be seen in the Nyquist plot. In Figure 11.7b, the controlled output was chosen to be $z := \begin{bmatrix}\theta & \dot\tau\end{bmatrix}'$, which corresponds to

$$G := \begin{bmatrix}1 & 0 & 0\\ 0 & 0 & -50\end{bmatrix}, \qquad H := \begin{bmatrix}0\\50\end{bmatrix}.$$

In this case we have $H'G = \begin{bmatrix}0 & 0 & -2500\end{bmatrix}$ 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. □



(a) $z := \begin{bmatrix}\theta\\ \dot\theta\end{bmatrix}$, leads to $G = \left[\begin{smallmatrix}1&0&0\\0&1&0\end{smallmatrix}\right]$, $H = \begin{bmatrix}0\\0\end{bmatrix}$, and $H'G = 0$



(b) $z := \begin{bmatrix}\theta\\ \dot\tau\end{bmatrix}$, leads to $G = \left[\begin{smallmatrix}1&0&0\\0&0&-50\end{smallmatrix}\right]$, $H = \begin{bmatrix}0\\50\end{bmatrix}$, $H'G = \left[\begin{smallmatrix}0&0&-2500\end{smallmatrix}\right] \neq 0$

Figure 11.7. Bode plots for the open-loop gain of the LQR controllers in Example 11.1

11.6 MATLAB® Hints

MATLAB® Hint 43 (`lqr`)**.** The command `[K,S,E]=lqr(A,B,QQ,RR,NN)` computes the optimal state-feedback LQR controller for the process

$$\dot x = \mathtt{A}x + \mathtt{B}u$$

with criteria

$$J := \int_0^\infty x(t)'\mathtt{QQ}x(t) + u'(t)\mathtt{RR}u(t) + 2x'(t)\mathtt{NN}u(t)dt. \tag{11.10}$$

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.

The criteria in (11.10) is quite general and enable us to solve several versions of the LQR problem: With the criterion that we used in (11.3), we have

Note 32. How to get the matrices A, B, G, H for a model obtained using system identification? ► p. 134

$$\begin{aligned} J_{\text{LQR}} &:= \int_0^\infty z(t)'Qz(t) + \rho u'(t)Ru(t)dt \\ &:= \int_0^\infty \big(Gx(t)+Hu(t)\big)'Q\big(Gx(t)+Hu(t)\big) + \rho u'(t)Ru(t)dt \\ &= \int_0^\infty x(t)'G'QGx(t) + u(t)'\big(H'QH+\rho I\big)u(t) + 2x(t)'G'QHu(t)dt, \end{aligned}$$

which matches (11.10) by selecting

$$\mathtt{QQ} = G'QG, \qquad \mathtt{RR} = H'QH + \rho R, \qquad \mathtt{NN} = G'QH.$$

If instead we want to use the criterion in (11.2) one should select

$$\mathtt{QQ} = G'G, \qquad \mathtt{RR} = H'H + \rho I, \qquad \mathtt{NN} = G'H. \qquad \square$$

MATLAB® Hint 44 (`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. □

Note 32. For process models obtained from system identification using the procedures discussed in Lectures 2–7, the different components of the state may not have physical meaning. In this case, we need to obtain the different matrices directly from the identified model. To see how this case be done, suppose that we identified a SISO model with input u and measured output y and consider the following options for our desired controller output z:

1. The simplest option consists of using the measured output y as our only measured output z. In this case, we could get the matrices A, B, G, H as follows:

```
model=tfest(data)
T=tf(model)
[A,B,G,H]=ssdata(T)
```

2. Suppose now that the resulting controller exhibited too much overshoot and we want to use instead $z = \begin{bmatrix} y & \gamma\dot{y} \end{bmatrix}'$ for some $\gamma > 0$ (see Section 11.5). In this case, we could get the matrices A, B, G, H as follows:

Note sb:low-freq. Why? ► p. 135

```
model=tfest(data)
T=tf(model)
[A,B,C,D]=ssdata(T)
G=[C;gamma*C*A]
H=[0;gamma*C*B]
```

3. The previous controlled output might lead to $H'G \neq 0$ in case the transfer function from u to y is strictly proper, but the transfer function from u to $\dot{y}$ is not. In this case, we might want to consider instead $z = \begin{bmatrix} y & \gamma\bar{y} \end{bmatrix}'$ where $\bar{y}$ is obtained from y through a transfer function of the form $\frac{ps}{s+p}$, with $p > 0$, which essentially takes a derivative of y "up to the frequency of the pole p." To accomplish this, we could use the following code

```
model=tfest(data)
T=tf(model)
Tfilter=[1;tf([p,0],[1,p])]
T1=Tfilter*T
[A,B,G,H]=ssdata(T1)
```

where we use the "filter" `Tfilter` with transfer function $\begin{bmatrix} 1 \\ \frac{ps}{s+p} \end{bmatrix}$ to get y and $\bar{y}$ from the output y.

11.7 To Probe Further

Note 24 (General LQR)**.** The most general form for a quadratic criteria is

$$J := \int_0^\infty x(t)'\bar{Q}x(t) + u'(t)\bar{R}u(t) + 2x'(t)\bar{N}u(t)dt. \tag{11.11}$$

Since $z = Gx + Hu$, the criterion in (11.2) is a special form of (11.11) with

$$\bar{Q} = G'G, \qquad \bar{R} = H'H + \rho I, \qquad \bar{N} = G'H$$

and (11.3) is a special form of (11.11) with

$$\bar{Q} = G'QG, \qquad \bar{R} = H'QH + \rho R, \qquad \bar{N} = G'QH.$$

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)*

$$A'P + PA + \bar{Q} - (\bar{P}B + \bar{N})\bar{R}^{-1}(B'\bar{P} + \bar{N}') = 0. \qquad \square$$

Note 25. 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 define one can compute its eigenvalues. If they are all positive the matrix is positive-definite, otherwise it is not. $\square$

MATLAB® Hint 42. `eig(A)` computes the eigenvalues of the matrix `A`. ► p. 123

Note 28 (Multiplicative uncertainty)**.** Since the Nyquist plot of $L(j\omega)$ does not enter a circle of radius one around -1, we have that

$$|1 + L(j\omega)| \geqslant 1 \quad \Rightarrow \quad \left|\frac{1}{1+L(j\omega)}\right| = \left|1 - \frac{L(j\omega)}{1+L(j\omega)}\right| \leqslant 1 \quad \Rightarrow \quad \left|\frac{L(j\omega)}{1+L(j\omega)}\right| \leqslant 2. \qquad \square$$

Note 30 (Equation (11.9))**.** When $z = \begin{bmatrix} y & \gamma\dot{y} \end{bmatrix}'$, we have that

$$z = \begin{bmatrix} y \\ \gamma\dot{y} \end{bmatrix} = \begin{bmatrix} Cx \\ \gamma CAx + \gamma CBu \end{bmatrix} \quad \Rightarrow \quad G = \begin{bmatrix} C \\ \gamma CA \end{bmatrix}, \quad H = \begin{bmatrix} 0 \\ \gamma CB \end{bmatrix}.$$

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

$$|L(j\omega)| \approx \frac{\sqrt{1+\gamma^2\omega^2}\,|P_y(j\omega)|}{\sqrt{H'H + \rho}} = \frac{|1 + j\gamma\omega|\,|P_y(j\omega)|}{\sqrt{H'H + \rho}}. \qquad \square$$

11.8 Exercises

11.1. Verify using the diagram in Figure 13.1 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

$$T_u(s) = \frac{1}{1+L}(KF + N),$$

where $L(s) = K(sI - A)^{-1}B$, and the closed-loop transfer function T_z from the reference r to the controlled output z is given by

$$T_z(s) = \frac{1}{1+L}P_z(KF + N),$$

where $P_z(s) = G(sI - A)^{-1}B + H$. $\square$

11.2. Consider an inverted pendulum operating near the upright equilibrium position, with linearized model given by

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{g}{\ell} & -\frac{b}{m\ell^2} \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m\ell^2} \end{bmatrix} u, \qquad y = \theta,$$

where u denotes an applied torque, the output $y = \theta$ is the pendulum's angle with a vertical axis pointing up, and $\ell = 1$m, $m = 1$Kg, $b = .1$N/m/s, $g = 9.8$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 θ, $\dot{\theta}$, $z(t) = \int_0^t \theta(s)ds$.

For both controllers provide the Bode plot of the open-loop gain and the closed-loop step response.

□

Lecture 12

LQG/LQR Output Feedback

This lecture introduces full-state observers as a tool to complement LQR in the design of output-feedback controllers. Loop-gain recovery is need for loop shaping based on LQG/LQR controllers.

Contents

12.1 Output Feedback

The state-feedback LQR formulation considered in Chapter 11.3 suffered from the drawback that the optimal control law

$$u(t) = -Kx(t) \tag{12.1}$$

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 (12.1), where $\hat{x}(t)$ denotes an estimate of the process' state $x(t)$. In this chapter we consider the problem of constructing state estimates.

12.2 Full-order Observers

Consider a process with state-space model

$$\dot{x} = Ax + Bu, \qquad y = Cx, \tag{12.2}$$

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

$$\dot{\hat{x}} = A\hat{x} + Bu. \tag{12.3}$$

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 (12.2) and (12.3), 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 $t \to \infty$. However, when A is not stable e is unbounded and $\hat{x}$ grow further and further apart from x as $t \to \infty$. To avoid this, one includes a correction term in (12.3):

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - \hat{y}), \qquad \hat{y} = C\hat{x}, \tag{12.4}$$

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 (12.2) and (12.4):

$$\dot{e} = Ax - A\hat{x} - L(Cx - C\hat{x}) = (A - LC)e.$$

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 (12.4) can be re-write as

$$\dot{\hat{x}} = (A - LC)\hat{x} + Bu + Ly, \tag{12.5}$$

Note. "Full-order" refers to the fact that the size of its state $\hat{x}$ is equal to the size of the process' state x, even though the output y may directly provide the values of some components of x. In fact, we will find such observers useful *even when the whole state can be measured* to improve the high-frequency response of an LQR controller. See Section 11.5... ► p. 130

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 12.1 shows how a full-order observer is connected to the process.

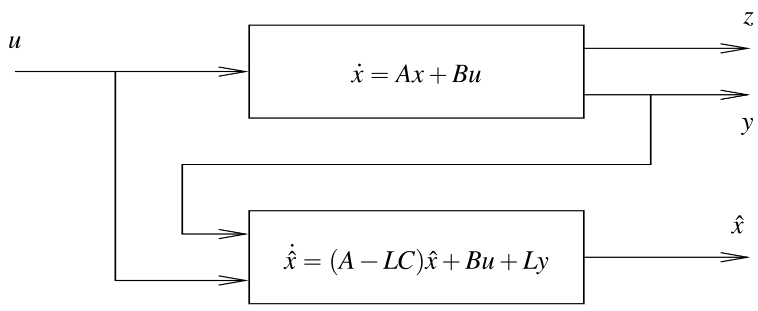


Figure 12.1. Full-order observer

12.3 LQG Estimation

Any choice of L in (12.4) for which $A - LC$ is asymptotically stable will make $\hat{x}$ converge to x, as long at the process dynamics is given by (12.2). 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

Note. We shall see later in Section 12.6 that we often use $\bar{B} = B$, which corresponds to an additive disturbance: $\dot{x} = Ax + B(u + d)$. ► p. 140

$$\dot{x} = Ax + Bu + \bar{B}d, \qquad y = Cx + n, \tag{12.6}$$

where d denotes a disturbance and n measurement noise. In this case, we need to re-write the estimation error dynamics for (12.6) and (12.4), 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 12.1 (Optimal LQG)**.** Find the matrix gain L that minimizes the asymptotic expected value of the estimation error:

$$J_{\mathrm{LQG}} := \lim_{t\to\infty} \mathrm{E}\left[\|e(t)\|^2\right],$$

where $d(t)$ and $n(t)$ are zero-mean Gaussian noise processes (uncorrelated from each other) with power spectrum

$$S_d(\omega) = Q_N, \qquad S_n(\omega) = R_N, \qquad \forall \omega. \qquad \Box$$

Solution to the optimal LQG Problem 12.1. 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 (12.5), 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 (12.6) is observable.
2. The system (12.6) 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 $A - LC$.

Note. A zero-mean white noise process n has an autocorrelation of the form

$$R_n(t_1,t_2) := \mathrm{E}\left[n(t_1)\,n'(t_2)\right] = Q_N\,\delta(t_1 - t_2).$$

Such process is *wide-sense stationary* in the sense that its mean is time-invariant and its autocorrelation $R(t_1,t_2)$ only depends on the difference $\tau := t_1 - t_2$. Its power spectrum is the same for every frequency ω and given by

$$S_n(\omega) := \int_{-\infty}^{\infty} R(\tau)e^{-j\omega\tau}d\tau = Q_N.$$

The precise mathematical meaning of this is not particularly important for us. The key point is that a large value for Q_N means that the disturbance is large, whereas a large S_N means that the noise is large.

MATLAB® Hint 45. `kalman` computes the optimal LQG estimator gain L. ► p. 142

Note 25. A symmetric $q \times q$ matrix M is *positive definite* if $x'Mx > 0$, for every nonzero vector $x \in \mathbb{R}^q$. ► p. 135

Note. We will return to the selection of Q_N and R_N in Section 12.6. ► p. 140

12.4 LQG/LQR Output Feedback

We now go back to the problem of designing an output-feedback controller for the process:

$$\dot{x} = Ax + Bu, \qquad y = Cx, \qquad z = Gx + Hu.$$

Suppose that we designed a state-feedback controller

$$u = -Kx \tag{12.7}$$

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 (12.7), instead of the true state x. This leads to the following output-feedback controller

$$\dot{\hat{x}} = (A - LC)\hat{x} + Bu + Ly = (A - LC - BK)\hat{x} + Ly, \qquad u = -K\hat{x},$$

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 12.2.

MATLAB® Hint 46. `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. 142

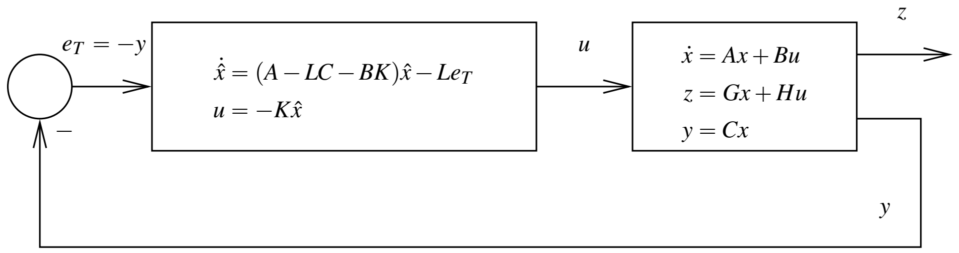


Figure 12.2. LQG/LQR output feedback

12.5 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:

$$\dot{x} = Ax + Bu, \qquad y = Cx, \tag{12.8}$$
$$\dot{\hat{x}} = (A - LC)\hat{x} + Bu + Ly, \qquad u = -K\hat{x}. \tag{12.9}$$

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:

$$\dot{x} = Ax + Bu = (A - BK)x + BKe, \qquad y = Cx,$$
$$\dot{e} = (A - LC)e, \qquad u = -K(x - e).$$

This can be written in matrix notation as

$$\begin{bmatrix} \dot{x} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} A - BK & BK \\ 0 & A - LC \end{bmatrix} \begin{bmatrix} x \\ e \end{bmatrix}, \qquad y = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x \\ e \end{bmatrix}.$$

Note. 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* (12.8)–(12.9) *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* (12.8)–(12.9).

12.6 Loop-gain Recovery

We saw in Sections 11.4 and 11.5 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 ρ 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.

Note. $\bar{B} = B$ corresponds to an additive *input disturbance* since the process becomes

$$\dot{x} = Ax + Bu + \bar{B}d$$
$$= Ax + B(u + d).$$

Loop-gain recovery. *Suppose that the process is single-input/single-output and has* no zeros in the right half-place. *Selecting*

$$\bar{B} := B, \qquad Q_N := 1, \qquad R_N := \sigma, \quad \sigma > 0,$$

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 $\sigma \to 0$, i.e.,

$$C(j\omega)P(j\omega) \xrightarrow{\sigma \to 0} K(j\omega I - A)^{-1}B, \quad \forall \omega \in [0, \omega_{\max}]$$

In general, the larger $\omega_{\max}$ is, the smaller σ needs to be for the gains to match.

Attention! 1. To achieve loop-gain recovery we need to chose $R_N := \sigma$, *even if this does not accurately describe the noise statistics.* This means that the estimator may not be optimal for the actual noise.

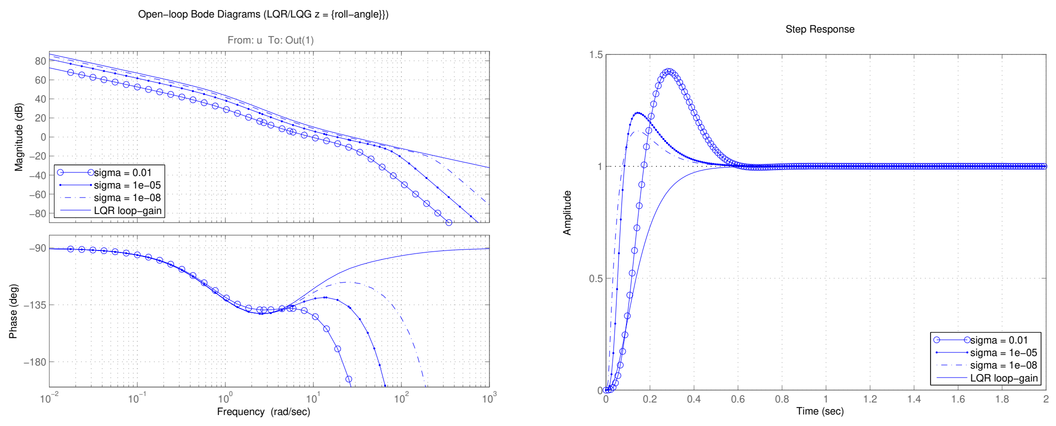
2. One should not make σ smaller than necessary because we do not want to recover the (slow) -20dB/decade magnitude decrease at high frequencies. In practice we should make σ *just small enough to get loop recovery until just above or at cross-over.* For larger values of ω, the output-feedback controller may actually behave much better than the state-feedback one.

Note. 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.

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. □

Example 12.1 (Aircraft roll-dynamics)**.** Figure 12.3a 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 10.1. The LQR controller



(a) Open-loop gain (b) Closed-loop step response

Figure 12.3. Bode plots and closed-loop step response for the open-loop gain of the LQR controllers in Examples 12.1, 13.2.

was designed using the controlled output $z := \begin{bmatrix} \theta & \gamma\dot{\theta} \end{bmatrix}'$, $\gamma = .1$ and $\rho = .01$. For the LQG state-estimators we used $\bar{B} = B$ and $R_N = \sigma$ for several values of σ. We can see that, as σ 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. □

12.7 Loop Shaping using LQR/LQG

We can now summarize the procedure to do open-loop gain shaping using LQG/LQR, to be contrasted with the classical lead-lag compensation briefly recalled in Section 9.3:

Note. A significant difference between this procedure and that discussed in Section 9.3, is that stability of the closed loop is now always automatically guaranteed.

1. Start by designing the LQR gain K assuming that the full state can be measured to obtain an *appropriate loop gain in the low-frequency range, up to the cross-over frequency*.

   This can be done using the procedure described in Section 11.5, where

   (a) the parameter ρ can be used to move the magnitude Bode plot up and down, and

   (b) the parameter γ can be used to improve the phase margin.

2. Use LQG to achieve loop-gain recovery up to the cross-over frequency.

   This can be done using the procedure described in Section 12.6, by decreasing σ just to the point where one gets loop recovery until just above or at the cross-over frequency. Hopefully, this still permits satisfying any high-frequency constraints for the loop gain.

12.8 MATLAB® Hints

MATLAB® Hint 45 (`kalman`)**.** The command `[est,L,P]=kalman(sys,QN,RN)` computes the optimal LQG estimator gain for the process

$$\dot{x} = \mathtt{A}x + \mathtt{B}u + \mathtt{BB}d, \qquad y = \mathtt{C}x + n, \tag{12.10}$$

where $d(t)$ and $n(t)$ are uncorrelated zero-mean Gaussian noise processes with covariance matrices

$$\mathrm{E}\big[d(t)\,d'(\tau)\big] = \delta(t-\tau)\mathtt{QN}, \qquad \mathrm{E}\big[n(t)\,n'(\tau)\big] = \delta(t-\tau)\mathtt{RN}.$$

Attention! The process model `sys` should have the D matrix equal to zero, even though (12.10) seems to imply otherwise.

The variable `sys` should be a state-space model created using `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 $[\hat{y};\, \hat{x}]$.

For loop transfer recovery (LTR), one should set

$$\mathtt{BB} = \mathtt{B}, \qquad \mathtt{QN} = I, \qquad \mathtt{RN} = \sigma I,\ \ \sigma \to 0.$$ □

Attention! Somewhat unexpectedly, the command `reg` produces a *positive* feedback controller.

MATLAB® Hint 46 (`reg`)**.** The function `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`. □

12.9 Exercises

12.1. Consider an inverted pendulum on a cart operating near the upright equilibrium position, with linearized model given by

$$\begin{bmatrix} p \\ \dot{p} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -2.94 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 11.76 & 0 \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ .325 \\ 0 \\ -.3 \end{bmatrix} F$$

where F denotes a force applied to the cart, p the cart's horizontal position, and θ the pendulum's angle with a vertical axis pointing up.

1. Design an LQG/LQR output-feedback controller that uses only the angle θ and the position of the cart p.

2. Design an LQG/LQR output-feedback controller that uses the angle θ, 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?

For both controllers provide the Bode plot of the open-loop gain of the LQR control $L(s) = k(sI - A)^{-1}B$, the open-loop gain of the LQR controller $C(s)P(s)$, and the closed-loop response to a step in the cart's horizontal position (with set-point for $\theta = 0$). □

Lecture 13

Set-Point Control

This lecture addresses the design of reference-tracking controllers based on LQG/LQR.

Contents

13.1 Nonzero Equilibrium State and Input

Consider again the system

$$\dot{x} = Ax + Bu, \qquad y = Cx, \qquad z = Gx + Hu, \qquad x \in \mathbb{R}^n,\ u \in \mathbb{R}^m,\ z \in \mathbb{R}^\ell.$$

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 (11.1) converge to values x^* and u^* for which

$$Ax^* + Bu^* = 0, \qquad r = Gx^* + Hu^*. \tag{13.1}$$

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 (13.1) is a system of linear equations and in general the solution to these equations is of the form

Note. When the process transfer-function has an integrator, one generally gets $u^* = 0$.

$$x^* = Fr, \qquad u^* = Nr. \tag{13.2}$$

Note. The matrix $\left[\begin{smallmatrix} A & B \\ G & H \end{smallmatrix}\right]$ 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.

For example, when the number of inputs to the process m is equal to the number of controlled outputs ℓ, we have

$$\begin{bmatrix} A & B \\ G & H \end{bmatrix}_{(n+\ell)\times(n+m)} \begin{bmatrix} x^* \\ u^* \end{bmatrix} = \begin{bmatrix} 0 \\ r \end{bmatrix} \quad \Leftrightarrow \quad \begin{bmatrix} x^* \\ u^* \end{bmatrix} = \begin{bmatrix} A & B \\ G & H \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ r \end{bmatrix} = \begin{bmatrix} \times & F_{n\times\ell} \\ \times & N_{m\times\ell} \end{bmatrix} \begin{bmatrix} 0 \\ r \end{bmatrix}, \tag{13.3}$$

where F is an $n\times\ell$ matrix given by the top n rows and right-most ℓ columns of $\left[\begin{smallmatrix} A & B \\ G & H \end{smallmatrix}\right]^{-1}$ and N is an $m\times\ell$ matrix given by the bottom m rows and right-most ℓ columns of $\left[\begin{smallmatrix} A & B \\ G & H \end{smallmatrix}\right]^{-1}$.

MATLAB® Hint 47. When the number of process inputs m is larger than or equal to the number of controlled outputs ℓ, the matrices F and N in (13.2) can be easily computed using MATLAB®. ► p. 146

Attention! When the number of process inputs m is larger than the number of controlled outputs ℓ we have an *over-actuated system* and the system of equations (13.1) will generally have multiple solutions. One of them is

$$\begin{bmatrix} x^* \\ u^* \end{bmatrix} = \underbrace{\begin{bmatrix} A & B \\ G & H \end{bmatrix}' \left(\begin{bmatrix} A & B \\ G & H \end{bmatrix} \begin{bmatrix} A & B \\ G & H \end{bmatrix}' \right)^{-1}}_{\text{pseudo inverse of } \left[\begin{smallmatrix} A & B \\ G & H \end{smallmatrix}\right]} \begin{bmatrix} 0 \\ r \end{bmatrix}. \tag{13.4}$$

In this case, we can still express x^* and u^* as in (13.2).

When the number of process inputs m is smaller than the number of controlled outputs ℓ we have an *under-actuated system* and the system of equations (13.1) may not have a solution. In fact, a solution will only exists for some specific references r. However, when it does exist, the solution can still express x^* and u^* as in (13.2). □

13.2 State feedback

When one wants z to converge to a given *set-point value* r, we define the following perturbations with respect to the desired equilibrium:

$$\Delta x := x - x^*, \qquad \Delta u := u - u^*,$$

where x^* and u^* are given by (13.2). To achieve set-point control one then sets

Note 33. Why? ► p. 144

$$\Delta u = -K\Delta x, \tag{13.5}$$

where K is the gain of the optimal regulation problem. This equation, is consistent with getting $u = u^*$ when $x = x^*$. Replacing Δu and Δx by their definitions, this leads to the following state-feedback controller

$$u = -K(x - x^*) + u^* = -Kx + (KF + N)r, \tag{13.6}$$

The corresponding control architecture is shown in Figure 13.1. The state-space model for the closed-loop system is given by

$$\dot{x} = Ax + Bu = (A - BK)x + B(KF + N)r$$
$$z = Gx + Hu = (G - HK)x + H(KF + N)r.$$

Note 33 (Set-point control with state-feedback)**.** To understand why (13.5) works, suppose we define

$$\Delta z = z - r, \qquad \Delta x = x - x^*, \qquad \Delta u = u - u^*.$$

Then

$$\dot{\Delta x} = Ax + Bu = A(x - x^*) + B(u - u^*) + Ax^* + Bu^*$$

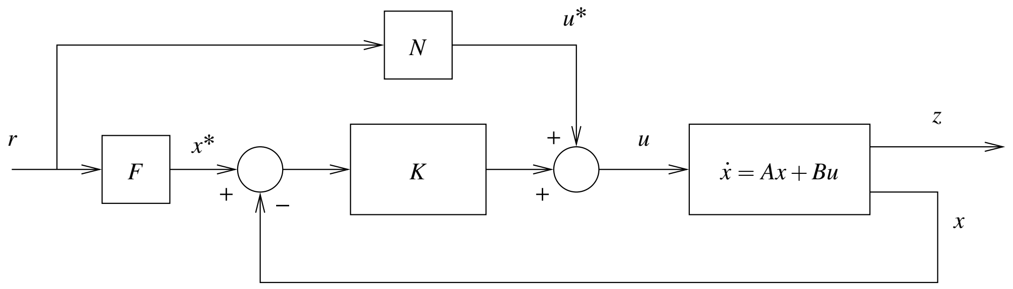


Figure 13.1. Linear quadratic set-point control with state feedback

$$\Delta z = Gx + Hu - r = G(x - x^*) + H(u - u^*) + Gx^* + Hu^* - r$$

and we conclude that

$$\dot{\Delta x} = A\Delta x + B\Delta u, \qquad \Delta z = G\Delta x + H\Delta u. \tag{13.7}$$

The control signal in (13.5), is therefore the optimal state-feedback LQR controller that minimizes

$$J_{\text{LQR}} := \int_0^\infty \Delta z(t)' Q \Delta z(t) + \rho \Delta u'(t) R \Delta u(t) dt,$$

This controller makes the system (13.7) asymptotically stable and therefore Δx, Δu, Δz all converge to zero as $t \to \infty$, which means that z converges to r. □

Example 13.1 (Aircraft roll-dynamics)**.** Figure 13.2 shows step responses for the state-feedback LQR controllers in Example 11.1, whose Bode plots for the open-loop gain are shown in Figure 11.6. Figure 13.2a shows that smaller values of ρ lead to faster responses and Figure 13.2b shows that

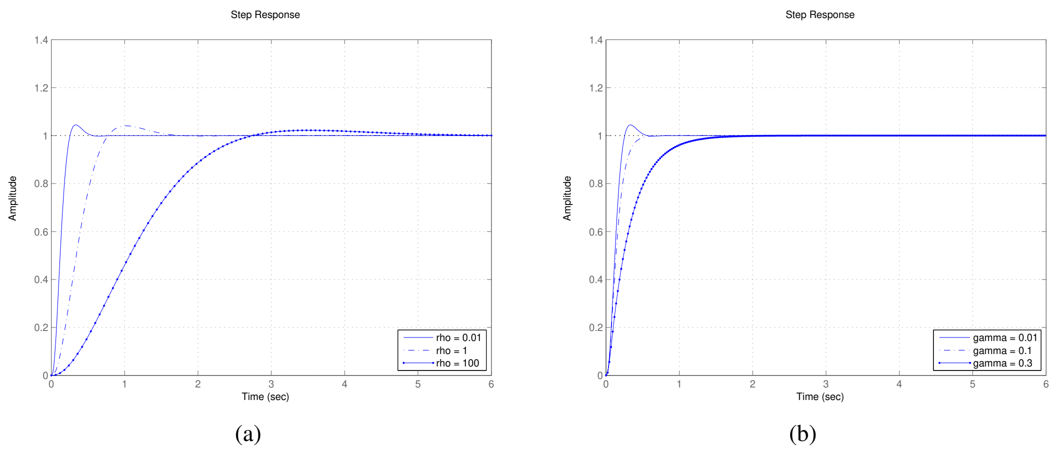


Figure 13.2. Step responses for the closed-loop LQR controllers in Example 13.1

larger values for γ lead to smaller overshoots (but slower responses). □

13.3 Output feedback

When one wants z to converge to a given *set-point value* r, the output-feedback LQG/LQR controller should be

$$\dot{\bar{x}} = (A - LC - BK)\bar{x} + L(Cx^* - y), \qquad u = K\bar{x} + u^*, \tag{13.8}$$

Note 34. Why? ► p. 146

Note. When $z = y$, we have $G = C$, $H = 0$ and in this case $Cx^* = r$. This corresponds to $CF = 1$ in Figure 13.3. When the process has an integrator we get $N = 0$ and obtain the usual unity-feedback configuration.

where x^* and u^* are given by (13.2). The controller (13.8) comes from setting $u - u^* = -K(\hat{x} - x^*)$ and defining $\bar{x} := x^* - \hat{x}$.

The corresponding control architecture is shown in Figure 13.3. The state-space model for the closed-loop system is given by

$$\begin{bmatrix} \dot{x} \\ \dot{\bar{x}} \end{bmatrix} = \begin{bmatrix} Ax + B(K\bar{x} + u^*) \\ (A - LC - BK)\bar{x} + L(Cx^* - Cx) \end{bmatrix} = \begin{bmatrix} A & BK \\ -LC & A - LC - BK \end{bmatrix} \begin{bmatrix} x \\ \bar{x} \end{bmatrix} + \begin{bmatrix} BN \\ LCF \end{bmatrix} r$$

$$z = Gx + H(K\bar{x} + u^*) = \begin{bmatrix} G & HK \end{bmatrix} \begin{bmatrix} x \\ \bar{x} \end{bmatrix} + HNr$$

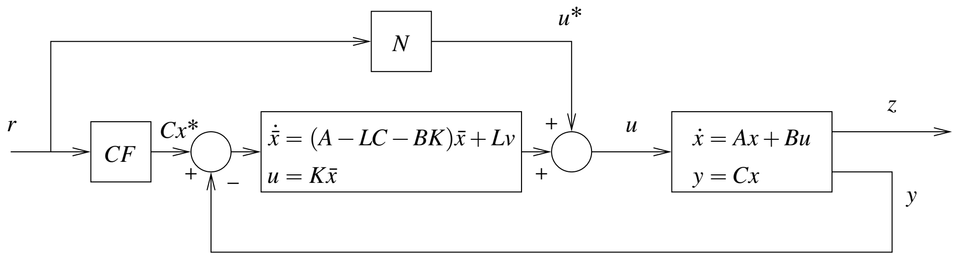


Figure 13.3. LQG/LQR set-point control

Note 34 (Set-point control with output feedback). To understand why (13.8) works suppose we define

$$\tilde{z} = z - r, \qquad \tilde{x} = x - x^* + \bar{x}.$$

Then

$$\dot{\tilde{x}} = (A - LC)\tilde{x} \tag{13.9}$$

$$\dot{\bar{x}} = (A - BK)\bar{x} - LC\tilde{x} \tag{13.10}$$

$$\tilde{z} = G(\tilde{x} - \bar{x}) + HK\bar{x}. \tag{13.11}$$

1. Since $A - LC$ is asymptotically stable, we conclude from (13.9) that $\tilde{x} \to 0$ as $t \to \infty$. 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 $\tilde{x} \to 0$ as $t \to \infty$, we conclude from (13.10) that $\bar{x} \to 0$ as $t \to \infty$.
3. Since $\tilde{x} \to 0$ and $\bar{x} \to 0$ as $t \to \infty$, we conclude from (13.11) that $z \to r$ as $t \to \infty$. □

Example 13.2 (Aircraft roll-dynamics). Figure 12.3b shows step responses for the output-feedback LQG/LQR controllers in Example 12.1, whose Bode plots for the open-loop gain are shown in Figure 12.3a. We can see that smaller values of σ lead to a smaller overshoot mostly due to a larger gain margin. □

13.4 MATLAB® Hints

MATLAB® Hint 47. When the number of process inputs m is larger than or equal to the number of controlled outputs ℓ, the matrices F and N in (13.2) that correspond to the equilibria in either (13.3) or (13.4) can be computed using the following MATLAB® commands:

```
M=pinv([A,B;G,H]);
F=M(1:n,end-l+1:end);
N=M(end-m+1:end,end-l+1:end);
```

where `n` denotes the size of the state, `l` the number of controlled outputs, and `m` the number of process inputs. □

13.5 Exercises

13.1. Verify equations (13.9), (13.10), and (13.11). □

Part IV

Nonlinear Control

Introduction to Nonlinear Control

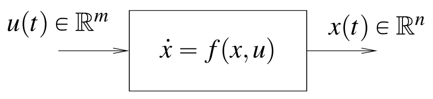


Figure 13.4. Nonlinear process with m inputs

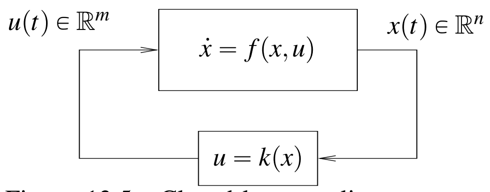


Figure 13.5. Closed-loop nonlinear system with state-feedback

In this section we consider the control of nonlinear systems such as the one shown in Figure 13.4. Our goal is to construct a state-feedback control law of the form

$$u = k(x)$$

Note. This control-law implicitly assumes that the whole state x can be measured.

that results in adequate performance for the closed-loop system

$$\dot{x} = f\big(x, k(x)\big).$$

Typically, at least we want x to be bounded and converge to some desired reference value r.

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.

Further reading A more extensive coverage of nonlinear control can be found, e.g., in [9].

Lecture 14

Feedback linearization controllers

This lecture introduces a control design method for nonlinear systems that uses state feedback to obtain linear closed-loop dynamics.

Contents

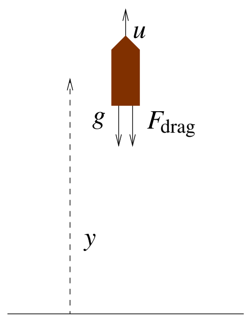
14.1 Feedback Linearization

In feedback linearization control design, we decompose the control signal u into two components with distinct roles:

$$u = u_{\text{nl}} + v,$$

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 14.1. Dynamics of a vehicle moving vertically in the atmosphere

Note. For large objects moving through air, the air resistance is approximately proportional to the square of the velocity, corresponding to a *drag force* that opposes motion, given by

$$F_{\text{drag}} = -\frac{1}{2} c\rho A \dot{y}|\dot{y}|$$

where ρ 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 [11]. F_{drag} always points in the opposite direction of $\dot{y}$.

To understand how this is done, consider the vehicle shown in Figure 14.1 moving vertically in the atmosphere. By choosing

$$u = u_{\text{nl}}(\dot{y}) + v, \qquad u_{\text{nl}}(\dot{y}) \coloneqq \frac{1}{2} c\rho A \dot{y}|\dot{y}| + mg,$$

we obtain

$$m\ddot{y} = -\frac{1}{2} c\rho A \dot{y}|\dot{y}| - mg + \big(u_{\text{nl}}(\dot{y}) + v\big) = v \quad \Rightarrow \quad \ddot{y} = \frac{1}{m} v.$$

In practice, we transformed the original nonlinear process into a (linear) double integrator with transfer function from v to y given by

$$T(s) \coloneqq \frac{1}{ms^2}.$$

Note. Any linear method can be used to design the controller for v, but we shall see shortly that robustness with respect to multiplicative uncertainty is important for the closed-loop.

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

$$v = K_P e + K_D \dot{e}, \qquad e \coloneqq r - y.$$

Figure 14.2 shows a diagram of the overall closed-loop system.

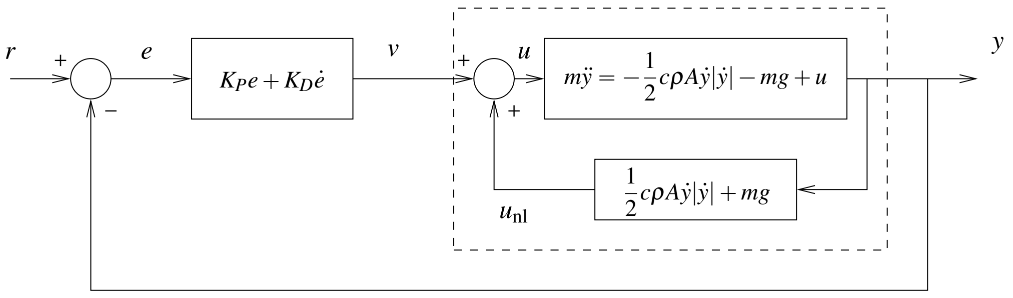

r
+
−
e
$K_P e + K_D \dot{e}$
v
+
u
$m\ddot{y} = -\frac{1}{2} c\rho A \dot{y}|\dot{y}| - mg + u$
+
$\frac{1}{2} c\rho A \dot{y}|\dot{y}| + mg$
u_{nl}
y

Figure 14.2. Feedback linearization controller. From an "input-output" perspective, the system in the dashed block behaves as a linear system with transfer function $T(s) \coloneqq \frac{1}{ms^2}$.

14.2 Generalized Model for Mechanical Systems

Note. These include robot-arms, mobile robots, airplanes, helicopters, underwater vehicles, hovercraft, etc.

The equations of motion of many mechanical systems can be written in the following general form

$$M(q)\ddot{q} + B(q,\dot{q})\dot{q} + G(q) = F, \tag{14.1}$$

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*;
- $G(q)$ is a k-vector sometimes called the *vector of conservative forces* (typically, gravity or forces generated by springs);
- $M(q)$ is a $k \times k$ non-singular symmetric positive-definite matrix called the *mass matrix*; and
- $B(q,\dot{q})$ is a $k \times k$ matrix sometimes called the *centrifugal/Coriolis/friction matrix*, for systems with no friction we generally have

$$\dot{q}' B(q,\dot{q})\dot{q} = 0, \qquad \forall \dot{q} \in \mathbb{R}^k$$

whereas for systems with friction

$$\dot{q}' B(q,\dot{q})\dot{q} \geqslant 0, \qquad \forall \dot{q} \in \mathbb{R}^k,$$

with equality only when $\dot{q} = 0$.

Note 25. A symmetric $k \times k$ matrix M is *positive definite* if $x'Mx > 0$, for every nonzero vector $x \in \mathbb{R}^k$. ► p. 135

Examples

Example 14.1 (Rocket)**.** The dynamics of the vehicle in Figure 14.1 that moves vertically in the atmosphere are given by

$$m\ddot{y} = -\frac{1}{2}c\rho A\dot{y}|\dot{y}| - mg + u,$$

where m is the vehicle's mass, g gravity's acceleration, ρ is the air density, A the cross-sectional area, c the drag coefficient, and u an applied force. This equation can be identified with (14.1), provided that we define

$$q := y, \qquad M(q) := m, \qquad B(q) := \frac{1}{2}c\rho A|\dot{y}|, \qquad G(q) := mg, \qquad F := u. \qquad \square$$

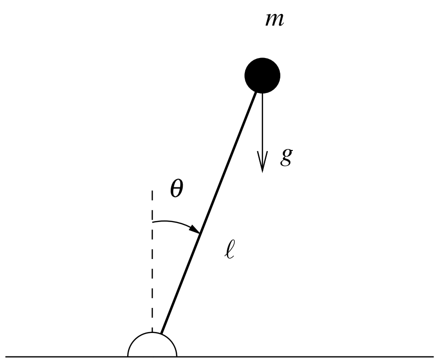


Figure 14.3. Inverted pendulum

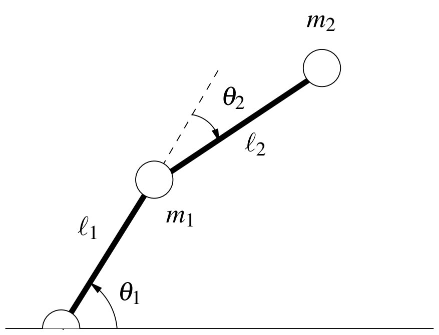


Figure 14.4. Two-link robot manipulator

Example 14.2 (Inverted pendulum)**.** The dynamics of the inverted pendulum shown in Figure 14.3 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 (14.1), provided that we define

$$q := \theta, \qquad M(q) := m\ell^2, \qquad B(q) := b, \qquad G(q) := -mg\ell\sin\theta, \qquad F := T. \qquad \square$$

Note. When the pendulum is attached to a cart and the input u is the cart's acceleration $\ddot{z}$, we have $T = -m\ell u\cos\theta$ but this makes the problem more difficult, especially around $\theta = \pm\pi/2$. Why?

Example 14.3 (Robot arm)**.** The dynamics of the robot-arm with two revolution joints shown in Figure 14.4 can be written as in (14.1), provided that we define

$$q := \begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}, \qquad F := \begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix}.$$

where τ_1 and τ_2 denote torques applied at the joints. For this system

$$\begin{aligned}
M(q) &:= \begin{bmatrix} m_2\ell_2^2 + 2m_2\ell_1\ell_2\cos\theta_2 + (m_1+m_2)\ell_1^2 & m_2\ell_2^2 + m_2\ell_1\ell_2\cos\theta_2 \\ m_2\ell_1\ell_2\cos\theta_2 + m_2\ell_2^2 & m_2\ell_2^2 \end{bmatrix}\\
B(q,\dot{q}) &:= \begin{bmatrix} -2m_2\ell_1\ell_2\dot{\theta}_2\sin\theta_2 & -m_2\ell_1\ell_2\dot{\theta}_2\sin\theta_2 \\ m_2\ell_1\ell_2\dot{\theta}_1\sin\theta_2 & 0 \end{bmatrix}\\
G(q) &:= g\begin{bmatrix} m_2\ell_2\cos(\theta_1+\theta_2) + (m_1+m_2)\ell_1\cos\theta_1 \\ m_2\ell_2\cos(\theta_1+\theta_2) \end{bmatrix}.
\end{aligned}$$

where g is the gravity's acceleration [2, p. 202–205]. $\square$

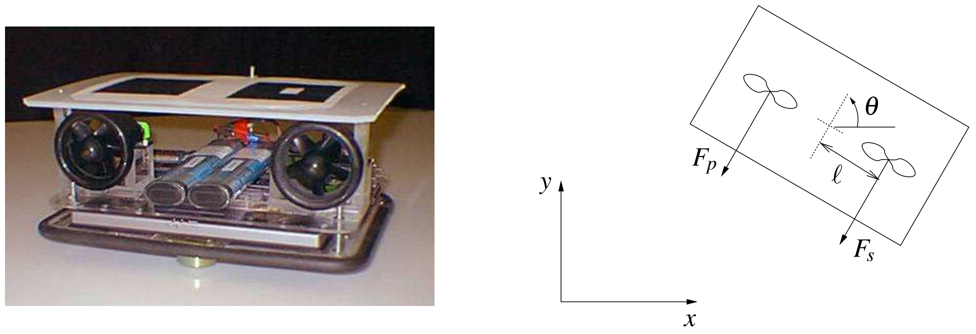


Figure 14.5. Hovercraft

Example 14.4 (Hovercraft)**.** The dynamics of the hovercraft shown in Figure 14.5 can be written as in (14.1), provided that we define

$$q := \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \qquad F := \begin{bmatrix} (F_s + F_p)\cos\theta - F_\ell \sin\theta \\ (F_s + F_p)\sin\theta + F_\ell \cos\theta \\ \ell(F_s - F_p) \end{bmatrix},$$

where F_s, F_p, and F_ℓ 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* because the number of controls (F_s and F_p) is smaller than the number of degrees of freedom (x, y, and θ). For this system

$$M(q) := \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix}, \qquad B(q) := \begin{bmatrix} d_v & 0 & 0 \\ 0 & d_v & 0 \\ 0 & 0 & d_\omega \end{bmatrix},$$

where $m = 5.5\,\mathrm{kg}$ is the mass of the vehicle, $J = 0.047\,\mathrm{kg\,m}^2$ its rotational inertia, $d_v = 4.5$ the coefficient of linear friction, $d_\omega = .41$ the coefficient of rotational friction, and $\ell = 0.123\,\mathrm{m}$ the moment arm of the forces. In these equations, the geometry and mass centers of the vehicle are assumed to coincide [3]. □

14.3 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

$$F = u = u_{\mathrm{nl}}(q,\dot q) + M(q)\,v, \qquad\qquad u_{\mathrm{nl}}(q,\dot q) := B(q,\dot q)\dot q + G(q),$$

we obtain

$$M(q)\ddot q + B(q,\dot q)\dot q + G(q) = B(q,\dot q)\dot q + G(q) + M(q)\,v \quad \Leftrightarrow \quad \ddot q = v.$$

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

$$M(q)\ddot{q} + B(q,\dot{q})\dot{q} + G(q) = B(q+n,\dot{q}+w)(\dot{q}+w) + G(q+n) + M(q+n)\,v \tag{14.2}$$

and (with some work) one can show that

$$\ddot{q} = \big(I+\Delta\big)\,v + d, \tag{14.3}$$

where

Note. See Exercise 14.1. ► p. 157

$$\begin{aligned}\Delta &:= M(q)^{-1}\big(M(q+n) - M(q)\big),\\ d &:= M(q)^{-1}\Big(\big(B(q+n,\dot{q}+w) - B(q,\dot{q})\big)\dot{q} + B(q+n,\dot{q}+w)w + G(q+n) - G(q)\Big).\end{aligned}$$

Since Δ 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.

14.4 Exercises

14.1. Verify that (14.3) is indeed equivalent to (14.2), by solving the latter equation with respect to $\ddot{q}$. □

14.2. Design a feedback linearization controllers to drive the inverted pendulum in Example 14.2 to the up-right position. Use the following values for the parameters: $\ell = 1\,\text{m}$, $m = 1\,\text{kg}$, $b = .1\,\text{N}\,\text{m}^{-1}\text{s}^{-1}$, and $g = 9.8\,\text{m}\,\text{s}^{-2}$. Verify the performance of your system in the presence of measurement noise using Simulink. □

Lecture 15

Lyapunov Stability

This lecture introduces a definition of stability for nonlinear systems and the basic tools used to check whether a nonlinear system is stable.

Contents

15.1 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 14.1 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.,

$$\dot{y}(t) = 1, \quad \forall t \geqslant 0,$$

is the only possible *constant trajectory* for the system. A question that may then arise is:

Will $\dot{y}(t)$ converge to 1 as $t \to \infty$, 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

$$\dot{x} = -(x+1)|x+1| + 1. \tag{15.1}$$

and the previous question could be reformulated as

Will $x(t)$ converge to 0 as $t \to \infty$, when $x(0) \neq 0$?

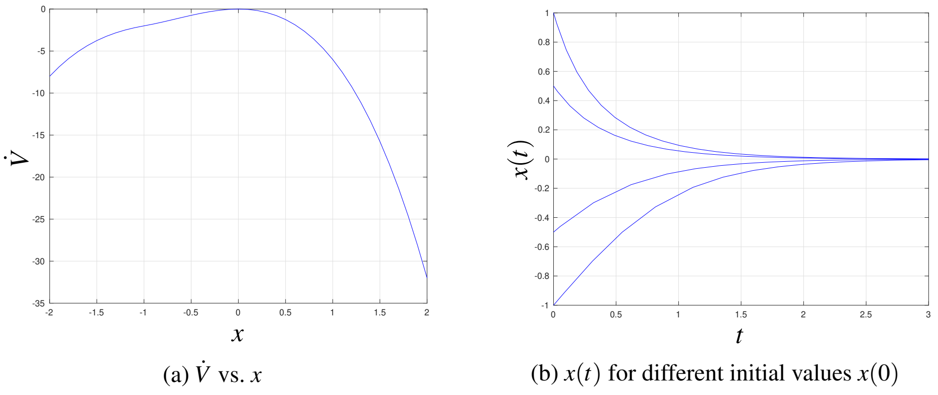
To address this question, suppose that we define

$$V(x) = x^2$$

and compute V's derivative with respect to time using the chain rule:

$$\dot{V}(t) = \frac{dV\big(x(t)\big)}{dt} = \frac{\partial V}{\partial x}\dot{x} = 2x\big(-(x+1)|x+1|+1\big) = \begin{cases} -2x^2(x+2) & x \geqslant -1 \\ 2x(x^2+2x+2) & x < -1. \end{cases} \tag{15.2}$$

Figure 15.1 shows the right-hand-side of (15.2) as a function of x and the solution of $x(t)$ of (15.1) for a couple of initial values $x(0)$. Three conclusions can be drawn from here:



(a) $\dot{V}$ vs. x (b) $x(t)$ for different initial values $x(0)$

Figure 15.1. Solutions to (15.1) and left-hand side of (15.2)

Boundedness For every initial condition $x(0)$, $|x(t)|$ will remain bounded for all $t \geqslant 0$, i.e., it will not blow up. This is because $V\big(x(t)\big) = x^2(t)$ cannot increase.

Stability If we start $x(0)$ very close to zero, then $x(t)$ will remain very close to zero for all times $t \geqslant 0$. This is because $V\big(x(t)\big) = x^2(t)$ will always be smaller or equal than $V\big(x(0)\big)$.

Convergence $x(t) \to 0$ as $t \to \infty$. This is because the right-hand side of (15.2) 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! $x \to 0$ as $t \to \infty$.*

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 15.1 (Lyapunov stability)**.** Given a system

$$\dot{x} = f(x), \qquad t \geqslant 0,\ x \in \mathbb{R}^n, \tag{15.3}$$

we say that:

(i) the *trajectories are globally bounded* if for every initial condition $x(0)$, there exists a scalar $\alpha\big(x(0)\big) \geqslant 0$ such that

$$\big\|x(t)\big\| \leqslant \alpha\big(x(0)\big), \quad \forall t \geqslant 0;$$

(ii) the *origin is globally stable* if the trajectories are globally bounded and $\alpha(x(0)) \to 0$ as $x(0) \to 0$;

(iii) the *origin is globally asymptotically stable* if it is globally stable and in addition $x(t) \to 0$ as $t \to \infty$.

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. □

15.2 Lyapunov Stability Theorem

The technique used before to conclude that the origin is globally asymptotically stable for the system (15.1) is a special case of a more general method proposed by Aleksandr M. Lyapunov. His main 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 $V : \mathbb{R}^n \to \mathbb{R}$ with the following properties:*

(i) *$V(x)$ is differentiable;*

(ii) *$V(x)$ is* positive definite*, which means that $V(0) = 0$ and*

$$V(x) > 0, \qquad \forall x \neq 0.$$

(iii) *$V(x)$ is* radially unbounded*, which means that $V(x) \to \infty$ whenever $x \to \infty$;*

(iv) *$\nabla_x V(x) \cdot f(x)$ is* negative definite*, which means that $\nabla_x V(0) \cdot f(0) = 0$ and*

$$\nabla_x V(x) \cdot f(x) < 0, \qquad \forall x \neq 0.$$

Then the origin is globally asymptotically stable for the system (15.3). The function $V(x)$ is called a Lyapunov function. □

Notation 6. Given a scalar function of n variables $f(x_1, \ldots, x_m)$, $\nabla_x f$ denotes the gradient of f, i.e.,

$$\nabla_x f(x_1, x_2, \ldots, x_m) = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_m} \end{bmatrix}.$$

The function $V(x) = x^2$ considered before satisfies all the conditions of the Lyapunov Stability Theorem for the system (15.1) so our conclusion that the origin was globally asymptotically stable is consistent with this theorem.

Note 35. What is the intuition behind Lyapunov's stability theorem? ► p. 161

Note 35 (Lyapunov Stability Theorem)**.** When the variable x is a scalar,

$$\dot{V}(t) = \frac{dV\big(x(t)\big)}{dt} = \frac{\partial V}{\partial x}\dot{x} = \frac{\partial V}{\partial x} f(x),$$

but when x and f are n-vectors, i.e.,

$$\dot{x} = f(x) \quad \Leftrightarrow \quad \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix},$$

we have

$$\dot{V}(t) = \frac{dV\big(x(t)\big)}{dt} = \frac{dV\big(x_1(t), x_2(t), \ldots, x_n(t)\big)}{dt} = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i}\dot{x}_i = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} f_i(x) = \nabla_x V(x) \cdot f(x).$$

Note 36. These derivatives exist because of condition (i) in Lyapunov Stability Theorem.

Attention! Computing the inner product $\frac{\partial V}{\partial x} f(x)$ actually gives us $\dot{V}(t)$.

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 and condition (iii) excludes the possibility that $V(x) \to 0$ as $x \to \infty$, this necessarily implies that $x \to 0$ as $t \to \infty$. □

Example 15.1. Consider the following system

$$\ddot{y} + (1 + |\dot{y}|)(y + \dot{y}) = 0.$$

Defining $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}' := \begin{bmatrix} y & \dot{y} \end{bmatrix}'$, this system can be written as $\dot{x} = f(x)$ as follows:

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -(1 + |x_2|)(x_1 + x_2). \end{aligned}$$

Note. Finding a Lyapunov function is more an art than a science. Fortunately several "artists" already discovered Lyapunov for many systems of interest. More on this shortly...

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,

Note. Even when $\nabla_x V(x) \cdot f(x)$ has a relatively simply form, it is often hard to verify that it is never positive. However, we will see in Lecture 16 that it is often easier to design a control that makes sure this is the case. ► p. 167

$$\begin{aligned} \nabla_x V(x) \cdot f(x) &= 2x_1\dot{x}_1 + 2(x_1 + x_2)(\dot{x}_1 + \dot{x}_2) \\ &= 2x_1x_2 + 2(x_1 + x_2)\big(x_2 - (1 + |x_2|)(x_1 + x_2)\big) \\ &= 2x_1(-x_1 + x_1 + x_2) + 2(x_1 + x_2)\big(x_2 - (1 + |x_2|)(x_1 + x_2)\big) \\ &= -2x_1^2 + 2(x_1 + x_2)\big(x_1 + x_2 - (1 + |x_2|)(x_1 + x_2)\big) \\ &= -2x_1^2 - 2|x_2|(x_1 + x_2)^2 \leqslant 0. \end{aligned}$$

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. □

15.3 Exercise

15.1. 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_{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_{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 $x := r - r_{\text{eq}}$. □

15.4 LaSalle's Invariance Principle

Consider now the system

$$\ddot{y} + \dot{y}|\dot{y}| + y^3 = 0$$

Defining $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}' := \begin{bmatrix} y & \dot{y} \end{bmatrix}'$, this system can be written as $\dot{x} = f(x)$ as follows:

$$\dot{x}_1 = x_2, \tag{15.4}$$
$$\dot{x}_2 = -x_2|x_2| - x_1^3. \tag{15.5}$$

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 (15.5) 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 $V : \mathbb{R}^n \to \mathbb{R}$ that satisfies the conditions (i)–(iii) of the Lyapunov Stability Theorem as well as*

Note 37. In general, LaSalle's Invariance Principle is stated in a more general form. ► p. 165

(iv)' $\nabla_x V(x) \cdot f(x)$ *is* negative semi-definite, *which means that*

$$\nabla_x V(x) \cdot f(x) \leqslant 0, \quad \forall x,$$

and, in addition, for every solution $x(t)$ for which

$$\nabla_x V\big(x(t)\big) \cdot f\big(x(t)\big) = 0,\ \ \forall t \geqslant 0 \tag{15.6}$$

we must necessarily have that

$$x(t) = 0,\ \ \forall t \geqslant 0. \tag{15.7}$$

Then the origin is globally asymptotically stable for the system (15.3). *In this case the function $V(x)$ is called a* weak Lyapunov function. □

15.5 Liénard Equation and Generalizations

LaSalle's Invariance principle is especially useful to prove stability for systems with dynamics described by the *Liénard equation*:

Note 38. The system considered in Section 15.4 was precisely of this form.

$$\ddot{y} + b(y,\dot{y})\dot{y} + \lambda(y) = 0,$$

where y is a scalar and $b(y,\dot{y})$, $\lambda(y)$ are functions such that

$$b(y,\dot{y}) > 0, \qquad \forall \dot{y} \neq 0 \tag{15.8}$$
$$\lambda(y) \neq 0, \qquad \forall y \neq 0 \tag{15.9}$$
$$\Lambda(y) := \int_0^y \lambda(z)dz > 0, \qquad \forall y \neq 0 \tag{15.10}$$

$$\lim_{y\to\infty} \Lambda(y) = \infty. \tag{15.11}$$

This type of equation arises in numerous mechanical systems from Newton's law.

Defining $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}' := \begin{bmatrix} y & \dot{y} \end{bmatrix}'$, the Liénard equation can be written as $\dot{x} = f(x)$ as follows:

$$\dot{x}_1 = x_2, \tag{15.12}$$
$$\dot{x}_2 = -b(x_1,x_2)x_2 - \lambda(x_1). \tag{15.13}$$

Note. This is how we got the Lyapunov function for the system in Section 15.4. Verify!

A candidate Lyapunov function for this system is

$$V(x) := \frac{x_2^2}{2} + \Lambda(x_1). \tag{15.14}$$

Note. Recall that (i) asked for $V(x)$ to be differentiable, (ii) asked for $V(x)$ to be positive definite, and (iii) asked for $V(x)$ to be radially unbounded.

This function satisfies the requirements (i)–(iii) of the Lyapunov Stability Theorem and

$$\nabla_x V(x) \cdot f(x) = x_2\dot{x}_2 + \lambda(x_1)\dot{x}_1 = -b(x_1,x_2)x_2^2 \leqslant 0,$$

since $b(x_1,x_2) > 0$ for every $x_2 \neq 0$ [cf. (15.8)]. Moreover, from (15.8) we conclude that

$$\nabla_x V(x) \cdot f(x) = 0 \quad \Rightarrow \quad x_2 = 0.$$

Because of (15.13), any trajectory with $x_2(t) = 0$, $\forall t$ necessarily must have $\lambda\big(x_1(t)\big) = 0$, $\forall t$, which in turn implies that $x_1(t) = 0$, $\forall t$ because of (15.9). 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

$$M(q)\ddot{q} + B(q,\dot{q})\dot{q} + Lq = 0, \tag{15.15}$$

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.

Note 25. A symmetric $k \times k$ matrix M is *positive definite* if $x'Mx > 0$, for every nonzero vector $x \in \mathbb{R}^k$. ► p. 135

Defining

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} := \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \in \mathbb{R}^{2k},$$

the system (15.15) can be written as $\dot{x} = f(x)$ as follows:

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = -M(x_1)^{-1}\Big(B(x_1,x_2)x_2 + Lx_1\Big). \tag{15.16}$$

A candidate Lyapunov function for this system is

$$V(x) := \frac{1}{2}x_2'M(x_1)x_2 + \frac{1}{2}x_1'Lx_1. \tag{15.17}$$

Note. 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 $\frac{d(x'Mx)}{dt} = 2x'M\frac{dx}{dt} + x'\frac{dM}{dt}x$.

This function satisfies the requirements (i)–(iii) and

$$\begin{aligned} \nabla_x V(x) \cdot f(x) &= x_2'M(x_1)\dot{x}_2 + \frac{1}{2}x_2'\Big(\frac{dM(x_1)}{dt}\Big)x_2 + \dot{x}_1'Lx_1 \\ &= -x_2'\Big(B(x_1,x_2) - \frac{1}{2}\frac{dM(x_1)}{dt}\Big)x_2. \end{aligned} \tag{15.18}$$

Using LaSalle's Invariance Principle, we conclude that the origin is globally asymptotically stable as long as

$$B(x_1,x_2) - \frac{1}{2}\frac{dM(x_1)}{dt} \tag{15.19}$$

is positive definite. This will be used shortly to design controllers for mechanical systems.

Note. Why do we need (15.19) to be positive definite? Because in this case, $\dot{V} = 0$ in (15.18) necessarily implies that $x_2 = 0$. Replacing $x_2 = 0$ in (15.16) leads to $M(x_1)^{-1}Lx_1 = 0$, which means that $x_1 = 0$ because both $M(x_1)$ and L are nonsingular matrices.

15.6 To Probe Further

Note 37 (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 $V : \mathbb{R}^n \to \mathbb{R}$ 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*

$$\nabla_x V(x) \cdot f(x) \leqslant 0, \quad \forall x.$$

Then the origin is globally stable for the system (15.3)*. 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 (15.3) *that is fully contained in E. Then every solution to* (15.3) *approaches M as $t \to \infty$. In case M only contains the origin, the origin is globally asymptotically stable for the system* (15.3).

Note. A set M is said to be invariant for the system (15.3) if every trajectory that starts inside M will remain inside this set forever.

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 (15.6)]. 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. □

15.7 Exercises

15.2. For the following systems, show that the origin is asymptotically stable using the Lyapunov function provided.

$$\begin{cases} \dot{x} = -x + y - xy^2 \\ \dot{y} = -x - y - x^2 y \end{cases} \qquad V(x,y) = x^2 + y^2 \tag{15.20}$$

$$\begin{cases} \dot{x} = -x^3 + y^4 \\ \dot{y} = -y^3(x+1) \end{cases} \qquad V(x,y) = x^2 + y^2 \tag{15.21}$$

$$\begin{cases} \dot{x} = -x + 4y \\ \dot{y} = -x - y^3 \end{cases} \qquad V(x,y) = x^2 + 4y^2 \tag{15.22}$$

$$\begin{cases} \dot{x} = x + 4y \\ \dot{y} = -2x - 5y \end{cases} \qquad V(x,y) = x^2 + bxy + ay^2 \tag{15.23}$$

For the last system, determine possible values for the constants $a, b \in \mathbb{R}$ (recall that V must be positive definite).

Hint for (15.23)*: it may be useful to know that a quadratic function of the form* $f(x,y) = \alpha x^2 + \beta xy + \gamma y^2$ *is*

1. *positive definite (and radially unbounded) if and only if*

$$\alpha > 0, \quad \gamma > 0, \quad \alpha\gamma > \frac{\beta^2}{4};$$

2. *and it is negative definite (and radially unbounded) if and only if*

$$\alpha < 0, \quad \gamma < 0, \quad \alpha\gamma > \frac{\beta^2}{4}.$$

You should simplify the problem by trying to pick values for a and b that lead to a simple expression for $\dot{V} := \nabla_x V(x) \cdot f(x)$. This can make it easy to checked whether or not $\dot{V}$ is negative definite and thus simplify the choice of a, b. □

15.3. For the following systems, show that the origin is asymptotically stable using the Lyapunov function provided.

$$\begin{cases} \dot{x} = y \\ \dot{y} = -x - y \end{cases} \qquad V(x,y) = x^2 + y^2 \tag{15.24}$$

$$\begin{cases} \dot{x} = y \\ \dot{y} = -y|y| - 3x \end{cases} \qquad V(x,y) = ax^2 + by^2 \tag{15.25}$$

For the last system, determine possible values for the constants a and b (recall that V must be positive definite).

Hint for (15.25): You can simplify the problem by picking values for a and b that lead to a simple expression for $\dot{V} := \nabla_x V(x) \cdot f(x)$. This can make it easy to checked whether or not $\dot{V}$ is negative definite. □

15.4. Consider the hovercraft model in Example 14.4. Show that if the generalized force vector is set to $F = -q$, then the origin is globally asymptotically stable. □

Lecture 16

Lyapunov-based Designs

This lecture introduces a method to design controllers for nonlinear systems based directly on satisfying the conditions of Lyapunov Stability theorem.

Contents

16.1 Lyapunov-based Controllers

In Lyapunov-based designs one starts by selecting a candidate Lyapunov function and then chooses the control signal for which the chosen Lyapunov function does indeed decrease along the trajectories of the closed-loop system.

To understand how this can be done, consider again the vehicle shown in Figure 14.1 with units chosen so that all constant coefficient are numerically equal to 1

$$\ddot{y} = -\dot{y}|\dot{y}| - 1 + u$$

and suppose we want y to converge to some constant reference value r. Defining the tracking error $e \coloneqq y - r$, we conclude that

$$\ddot{e} = -\dot{e}|\dot{e}| - 1 + u.$$

Setting the state

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \coloneqq \begin{bmatrix} e \\ \dot{e} \end{bmatrix}$$

the dynamics of the system can be written as $\dot{x} = f(x,u)$ as follows:

$$\dot{x}_1 = x_2 \tag{16.1}$$
$$\dot{x}_2 = -x_2|x_2| - 1 + u, \tag{16.2}$$

and we would like to make the origin globally asymptotically stable.

Note. It is useful to recall here the Liénard equation $\ddot{y} + b(y,\dot{y})\dot{y} + \lambda(y) = 0$ and its state-space form $\dot{x}_1 = x_2$, $\dot{x}_2 = -b(x_1,x_2)x_2 - \lambda(x_1)$, for which we used the Lyapunov function $V(x) \coloneqq \frac{x_2^2}{2} + \Lambda(x_1)$, $\Lambda(y) \coloneqq \int_0^y \lambda(z)dz > 0$.

In view of what we saw for the Liénard equation, we will try to make

$$V(x) := \frac{x_2^2}{2} + \rho \frac{x_1^2}{2}, \qquad \rho > 0$$

a Lyapunov function for the system by appropriate choice of u. This function satisfies the requirements (i)–(iii) and

$$\nabla_x V(x) \cdot f(x,u) = x_2\dot{x}_2 + \rho x_1 \dot{x}_1 = -x_2^2|x_2| + x_2(-1 + u + \rho x_1).$$

Note. Compare with

$$V(x) := \frac{x_2^2}{2} + \Lambda(x_1)$$

in (15.14). Essentially, we are aiming for

$$\Lambda(x_1) = \int_0^{x_1} \lambda(z)dz = \rho\frac{x_1^2}{2},$$

which would correspond to $\lambda(x_1) = \rho x_1$ [Why? Take derivatives of the equalities above]. We will find other interesting options in Exercise 16.1. ► p. 170

A simple way to make the system Lyapunov stable is then to select

$$-1 + u + \rho x_1 = 0 \quad \Leftrightarrow \quad u = 1 - \rho x_1 = 1 - \rho(y - r),$$

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:

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -x_2|x_2| - 1 + u(x_1) = -x_2|x_2| - \rho x_1. \end{aligned}$$

Attention! A feedback linearization controller for (16.2) 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.

Note. This controller does not attempt to cancel the term $x_2|x_2|$ that actually helps in making $\dot{V} < 0$.

Consider now the Lyapunov-based controller

$$u(x_1) = 1 - \rho x_1$$

also in the presence of noise:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -x_2|x_2| - 1 + u(x_1 + n_1) = -x_2|x_2| - \rho x_1 + d, \qquad d = -\rho n_1. \end{aligned}$$

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. □

16.2 Application to Mechanical Systems

Consider again a fully actuated mechanical system of the following form

$$M(q)\ddot{q} + B(q,\dot{q})\dot{q} + G(q) = u \tag{16.3}$$

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 (16.3) can be written as $\dot{x} = f(x,u)$ as follows:

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -M(x_1+r)^{-1}\Big(B(x_1+r,x_2)x_2 + G(x_1+r) - u\Big). \end{aligned}$$

Based on what we saw in Section 15.5, we will try to make

Note. Compare this $V(x)$ with equation (15.17). ► p. 164

$$V(x) := \frac{1}{2}x_2' M(x_1+r)x_2 + \frac{\gamma_1}{2}x_1' x_1, \qquad \gamma_1 > 0$$

a Lyapunov function for the system by appropriate choice of u. This function satisfies the requirements (i)–(iii) and

$$\begin{aligned} \nabla_x V(x)\cdot f(x) &= x_2' M(x_1+r)\dot{x}_2 + \frac{1}{2}x_2'\Big(\frac{dM(x_1+r)}{dt}\Big)x_2 + \gamma_1 \dot{x}_1' x_1 \\ &= -x_2'\Big(B(x_1+r,x_2)x_2 - \frac{1}{2}\frac{dM(x_1+r)}{dt}x_2 + G(x_1+r) - u - \gamma_1 x_1\Big). \end{aligned}$$

Since in general $x_2' B(x_1+r,x_2)x_2 \geqslant 0$, a simple way to make the system Lyapunov stable is to select

$$\begin{aligned} -\frac{1}{2}\frac{dM(x_1+r)}{dt}x_2 + G(x_1+r) - u - \gamma_1 x_1 &= \gamma_2 x_2 \\ \Leftrightarrow \quad u = -\frac{1}{2}\frac{dM(x_1+r)}{dt}x_2 + G(x_1+r) - \gamma_1 x_1 - \gamma_2 x_2 & \\ = -\frac{1}{2}\frac{dM(q)}{dt}\dot{q} + G(q) - \gamma_1(q-r) - \gamma_2\dot{q}, & \end{aligned}$$

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

$$x_2'\Big(B(x_1+r,x_2)x_2 - \frac{1}{2}\frac{dM(x_1+r)}{dt}\Big)x_2 \geqslant 0.$$

For those systems it suffices to select

$$\begin{aligned} G(x_1+r) - u - \gamma_1 x_1 = \gamma_2 x_2 \quad \Leftrightarrow \quad u &= G(x_1+r) - \gamma_1 x_1 - \gamma_2 x_2 \\ &= G(q) - \gamma_1(q-r) - \gamma_2\dot{q}, \end{aligned} \tag{16.4}$$

which leads to

$$\begin{aligned} \nabla_x V(x)\cdot f(x) &= -x_2'\Big(B(x_1+r,x_2)x_2 - \frac{1}{2}\frac{dM(x_1+r)}{dt}\Big)x_2 - \gamma_2 x_2' x_2 \\ &\leqslant -\gamma_2 x_2' x_2. \end{aligned}$$

□

The controller (16.4) corresponds to the control architecture shown in Figure 16.1. This controller resembles the feedback linearization controller in Figure 14.2, with the key difference that now the gravity term is canceled, but not the friction term. □

Note. This type of control is very common in robotic manipulators.

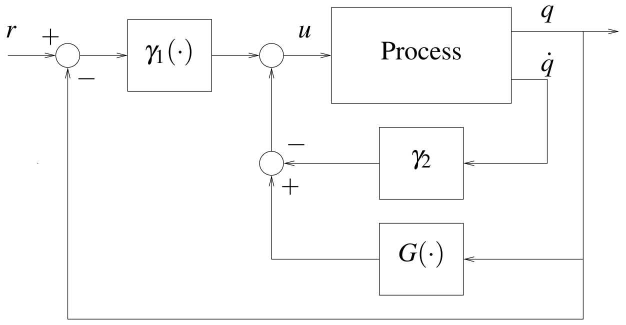


Figure 16.1. Control architecture corresponding to the Lyapunov-based controller in (16.4).

16.3 Exercises

Note. Compare with

$$V(x) \coloneqq \frac{x_2^2}{2} + \Lambda(x_1)$$

in (15.14). Essentially, we are aiming for

$$\Lambda(x_1) = \int_0^{x_1} \lambda(z)dz = \rho x_1 \arctan(x_1),$$

which would correspond to $\lambda(x_1) = \rho\left(\frac{x_1}{1+x_1^2} + \arctan(x_1)\right)$ [Why? Take derivatives of the equalities above].

16.1. Design a controller for the system (16.2) using the following candidate Lyapunov function

$$V(x) \coloneqq \frac{x_2^2}{2} + \rho x_1 \arctan(x_1), \qquad \rho > 0.$$

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? □

16.2. Consider again system (16.2) and the following candidate Lyapunov function

$$V(x) \coloneqq \frac{x_2^2}{2} + \rho x_1 \arctan(x_1), \qquad \rho > 0.$$

Find a Lyapunov-based control law $u(x_1, x_2)$ that keeps

$$\nabla_x V(x) \cdot f(x,u) \leqslant -x_2^2 |x_2|,$$

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. □

16.3. Design a Lyapunov based controller for the inverted pendulum considered in Exercise 14.2 and compare its noise rejection properties with the feedback linearization controller previously designed. □

16.4. Re-design the controller in Exercise 16.3 to make sure that the control signal always remains bounded. Investigate its noise rejection properties.

Hint: Draw inspiration from Exercises 16.1 and 16.2. □

Bibliography

[1] R. Adrain. Research concerning the probabilities of the errors which happen in making observations. *The Analyst*, I:93–109, 1808. Article XIV. (cited in p. 33)

[2] J. J. Craig. *Introduction to Robotics Mechanics and Control*. Addison Wesley, Reading, MA, 2nd edition, 1986. (cited in p. 155)

[3] L. Cremean, W. Dunbar, D. van Gogh, J. Hickey, E. Klavins, J. Meltzer, and R. Murray. The Caltech multi-vehicle wireless testbed. In *Proc. of the 41st IEEE Conf. on Decision and Contr.*, Dec. 2002. (cited in p. 156)

[4] G. E. Dullerud and F. Paganini. *A Course in Robust Control Theory*. Number 36 in Texts in Applied Mathematics. Springer, New York, 1999. (cited in p. 95)

[5] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Prentice Hall, Upper Saddle River, NJ, 4th edition, 2002. (cited in p. 7, 9, 18, 100, 107, 120, 127)

[6] 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`. (cited in p. 33)

[7] B. Hayes. Science on the farther shore. *American Scientist*, 90(6):499–502, Nov. 2002. (cited in p. 33)

[8] J. P. Hespanha. *Linear Systems Theory*. Princeton Press, Princeton, New Jersey, 2nd edition, Feb. 2018. (cited in p. 117)

[9] H. K. Khalil. *Nonlinear Systems*. Prentice Hall, Englewood Cliffs, NJ, 3rd edition, 2002. (cited in p. 151)

[10] L. Ljung. *System Identification: Theory for the user*. Information and System Sciences Series. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 1999. (cited in p. 3, 18, 33)

[11] R. A. Serway and R. J. Beichner. *Physics for Scientists and Engineers*. Brooks Cole, 5th edition, 1999. (cited in p. 154)

[12] J. V. Vegte. *Feedback Control Systems*. Prentice Hall, Englewood Cliffs, NJ, 3rd edition, 1994. (cited in p. 119)