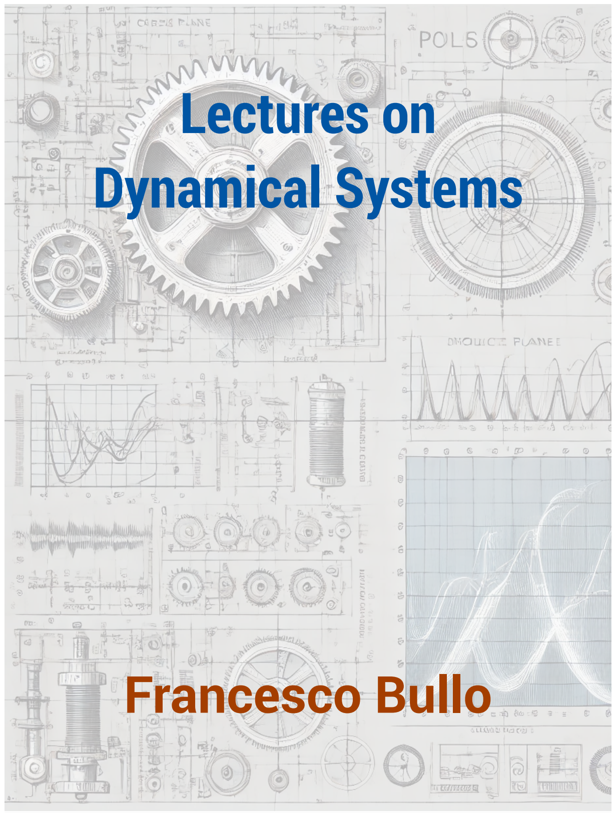
UC Santa Barbara, Department of Mechanical Engineering
ME103 Dynamical Systems. Chapter Slides.

Francesco Bullo
http://motion.me.ucsb.edu/ME103-Fall2025/syllabus.html



Contents

Chapter 8

Control Systems Design

In this chapter, we study some foundational aspects and strategies of feedback control systems, emphasizing the utility of the Laplace domain. We use transfer functions and block diagrams in the Laplace domain to reason about interconnections of dynamical systems and control systems.

A central focus is on *PID controllers*, which integrate proportional, integral, and derivative actions and are the most widely used control strategy in engineering. We study PID control strategies for both first and second-order systems, using *pole placement* to design controller gains that achieve closed-loop stability and reference tracking.

The chapter also introduces the *root locus* method, a graphical and intuitive technique that aids in visualizing how closed-loop poles shift in the complex plane as a scalar gain varies. This method is helpful to design controllers in low-order, single-input single-output systems.

Finally, we examine potential *instability* mechanisms in control systems, such as unmodeled dynamics and delays, which can undermine controller effectiveness despite seemingly robust designs. These scenarios underscore the importance of comprehensive system analysis to prevent local stability issues.

8.1 PID controllers, transfer functions, and modifications

8.1.1 PID controllers and their transfer functions

In Chapter 7, we introduced proportional (P) and integral (I) feedback. It is natural to extend these ideas with *derivative (D)* feedback. Together, these three actions form the foundation of the widely used *PID controller*.

Table 8.1 summarizes the three basic actions in both the time and Laplace domains.

Control action	*Time domain*	*Laplace domain*
Proportional (P)	$u(t) = k_{\mathrm{P}} e(t)$	$\frac{U(s)}{E(s)} = k_{\mathrm{P}}$
Integral (I)	$u(t) = k_{\mathrm{I}} \int_0^t e(\sigma)\, d\sigma$	$\frac{U(s)}{E(s)} = \frac{k_{\mathrm{I}}}{s}$
Derivative (D)	$u(t) = k_{\mathrm{D}} \dot{e}(t)$	$\frac{U(s)}{E(s)} = k_{\mathrm{D}} s$

Table 8.1: Proportional, integral, and derivative control. The proportional, integral and derivative gains are denoted by k_{P}, k_{I}, and k_{D}, respectively.

The full PID controller combines these three actions:

$$u(t) = k_{\mathsf{P}} e(t) + k_{\mathsf{I}} \int_0^t e(\sigma)\, d\sigma + k_{\mathsf{D}} \dot{e}(t), \tag{8.1}$$

with transfer function

$$\frac{U(s)}{E(s)} = k_{\mathsf{P}} + \frac{k_{\mathsf{I}}}{s} + k_{\mathsf{D}} s. \tag{8.2}$$

We illustrate the PID controller in Figure 8.1.

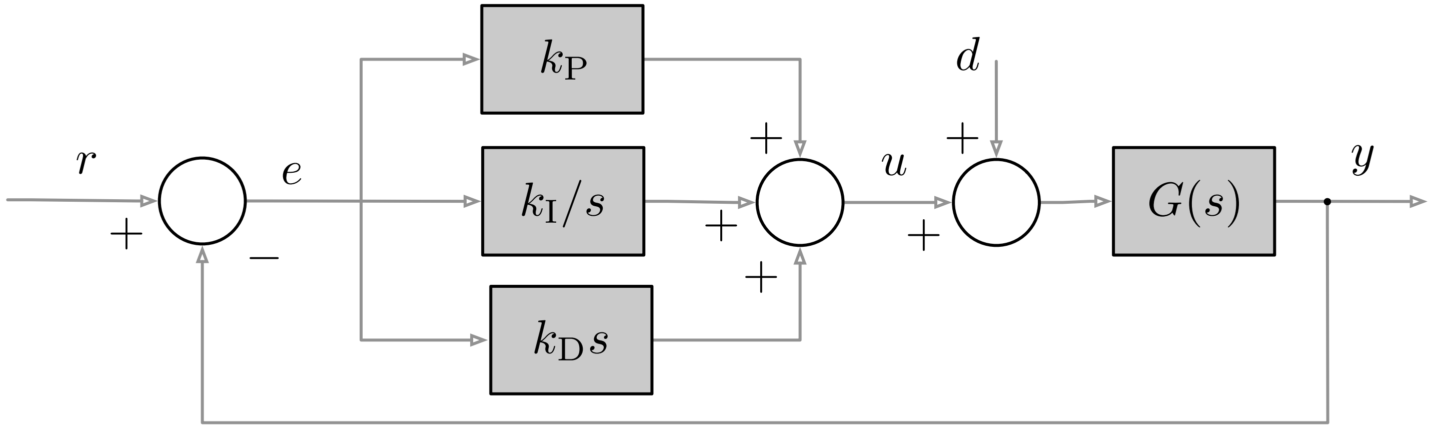


Figure 8.1: Feedback diagram with a PID controller. The controller transfer function is $k_{\mathsf{P}} + \frac{k_{\mathsf{I}}}{s} + k_{\mathsf{D}} s$.

It is often convenient to parameterize the I and D actions in terms of time constants:

$$\tau_{\mathsf{I}} = \frac{k_{\mathsf{P}}}{k_{\mathsf{I}}}, \qquad \tau_{\mathsf{D}} = \frac{k_{\mathsf{D}}}{k_{\mathsf{P}}}, \tag{8.3}$$

known as the *integral time constant* and the *derivative time constant*. In this notation, the PID transfer function becomes

$$\frac{U(s)}{E(s)} = k_{\mathsf{P}} \left(1 + \tau_{\mathsf{D}} s + \frac{1}{\tau_{\mathsf{I}} s} \right). \tag{8.4}$$

Special cases arise by setting one or more gains to zero: P control ($k_{\mathsf{P}} > 0$, $k_{\mathsf{I}} = k_{\mathsf{D}} = 0$), PI control ($k_{\mathsf{P}}, k_{\mathsf{I}} > 0$, $k_{\mathsf{D}} = 0$), PD control ($k_{\mathsf{P}}, k_{\mathsf{D}} > 0$, $k_{\mathsf{I}} = 0$), and so on.

Figure 8.2 offers an intuitive interpretation of the three terms in PID control:

- proportional action reacts to the *present* error,
- integral action accounts for the *past* accumulated error,
- derivative action anticipates the *future* trend of the error.

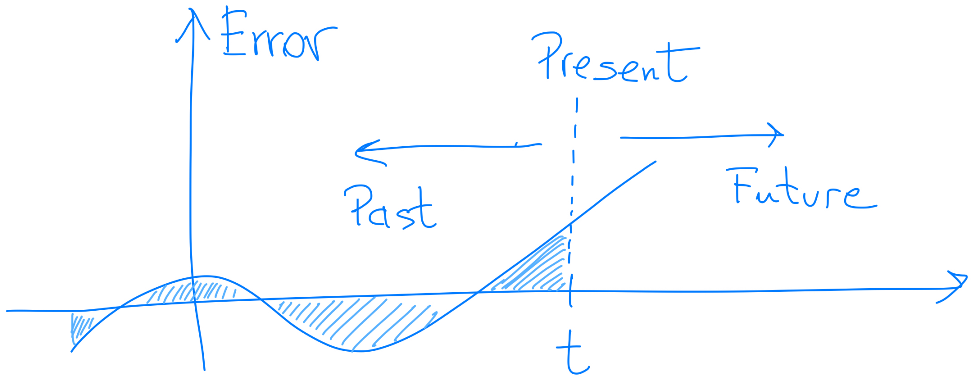


Figure 8.2: Proportional, integral, and derivative actions use information about present, past, and future error, respectively.

8.1.2 Modification #1: Realizability of the derivative action and lead compensators

The transfer function of an ideal derivative is $k_{\mathrm{D}}s$, a rational function whose numerator has higher degree than its denominator. Such functions are called *improper* and, as discussed in Appendix 6.3, cannot be realized in physical systems. A common workaround is the approximation

$$\frac{\tau_{\mathrm{D}}s}{1+\gamma\tau_{\mathrm{D}}s}, \qquad \text{with } \gamma \approx 0.1, \tag{8.5}$$

which regularizes the high-frequency behavior while retaining the desired differentiating action. This modification replaces the ideal PD transfer function $k_{\mathrm{P}} + k_{\mathrm{D}}s$ with the proper form

$$k_{\mathrm{P}}\,\frac{1+\tau_{\mathrm{D}}s}{1+\gamma\tau_{\mathrm{D}}s}. \tag{8.6}$$

The modified controller introduces a zero at $-1/\tau_{\mathrm{D}}$ and a pole at $-1/(\gamma\tau_{\mathrm{D}})$. Since the pole is approximately ten times farther from the origin than the zero, the zero dominates the behavior (recall the discussion in Section 5.4). A controller with transfer function of the form (8.6) is called a *lead compensator*.

8.1.3 Modification #2: PI-D and I-PD architectures

A practical issue with PID control (reported in Figure 8.3 for convenience), is the so-called *setpoint kick*: a sudden, undesirable jump in the control output when the reference $r(t)$ changes abruptly. This undesirable jump is due to the derivative term acting directly on the reference signal. (A version of this problem is relevant also when we approximate the derivative control action by a lead compensator.)

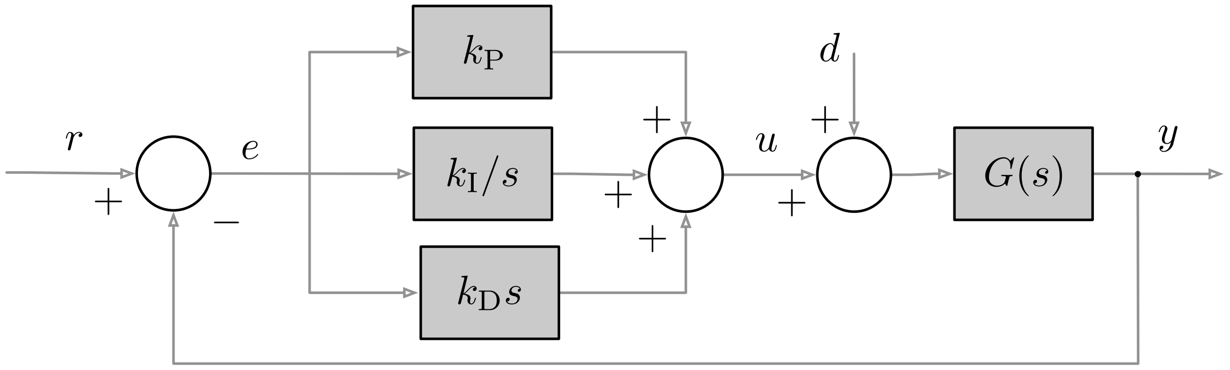


Figure 8.3: A feedback control diagram with a PID controller.

To mitigate the setpoint kick effect, alternative architectures are used:

- *The PI-D architecture* (center image of Figure 8.4): the derivative action is applied only to the output, not to the reference. A step in $r(t)$ then causes only a step (not an impulse) in $u(t)$.
- *The I-PD architecture* (right image of Figure 8.4): the reference enters only the integral channel, further smoothing the control response when $r(t)$ changes.

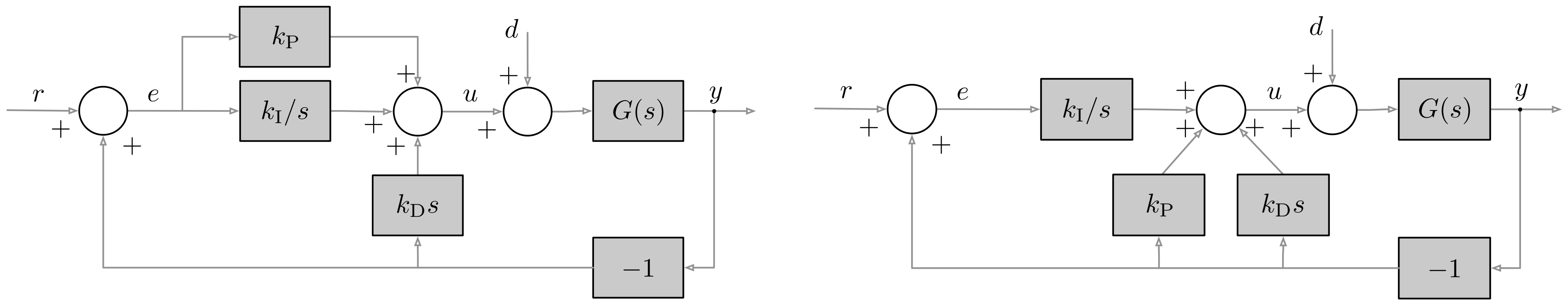


Figure 8.4: Alternative architectures to the standard PID architecture in Figure 8.1: PI-D and I-PD control, respectively.

To compute and compare the behavior of the three architectures, it is possible to (see E8.1):

(i) show that the disturbance-to-output transfer function $Y(s)/D(s)$ is identical for PID, PI-D, and I-PD architectures (disturbance rejection is unaffected by the architecture),

(ii) derive the reference-to-output transfer functions $Y(s)/R(s)$ for each architecture, and

(iii) compare the step responses for PI-D and I-PD in response to (i) a disturbance step and (ii) a reference step.

8.2 PID design for first and second-order systems

In this section we consider various control designs for first and second order systems. We do not necessarily assume that the open-loop systems are stable. We consider PID controllers and their control gains k_P, k_I, and k_D. We consider also simplified versions of PID controllers, including P control ($k_\mathsf{I} = k_\mathsf{D} = 0$), PI ($k_\mathsf{D} = 0$), PD ($k_\mathsf{I} = 0$).

Pole placement approach For each system (i.e., first order or second order) and controller architectures of interest (P, PI, PD, or PID), we will design controller gains via *pole placement*, i.e., by making the denominator of the closed-loop transfer function match a desirable given polynomial.

Pole placement is a fundamental control strategy that leverages the fact that the poles of a linear system determine key aspects of its dynamic behavior, such as stability, responsiveness, and damping. In short, pole placement means selecting a set of desired poles and then calculating the controller gains that place the system's closed-loop poles at those locations. By appropriately choosing the poles of the closed-loop system, we can shape the transient response to meet desired performance specifications — for example, faster settling time, reduced overshoot, or improved disturbance rejection.

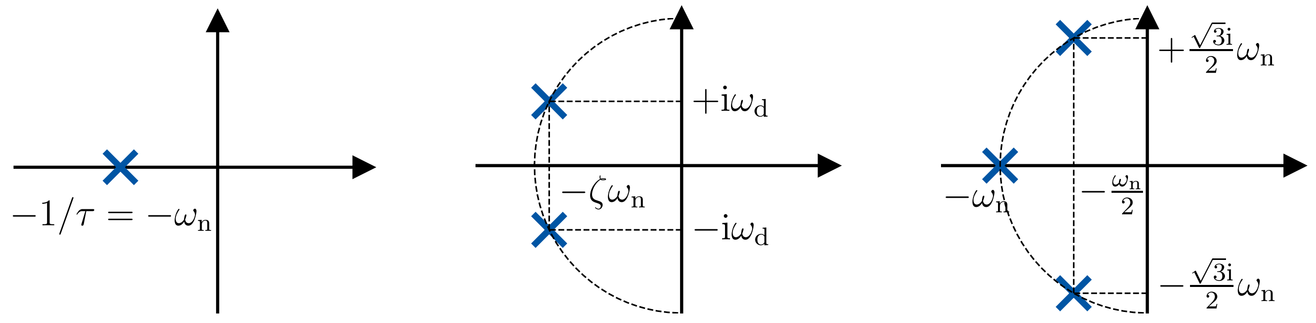


Figure 8.5: The poles of stable polynomials. Left image: equation (8.7) for 1st order. Center image: equation (8.8) for 2nd order. Right image: equation (8.10) for 3rd order.

(i) The canonical form of a *stable polynomial of 1st order* is

$$s + \omega_n = s + 1/\tau \qquad \text{for arbitrary time constant } \tau > 0. \tag{8.7}$$

(ii) The canonical form of a *stable polynomial of 2nd order* is

$$s^2 + 2\zeta\omega_n s + \omega_n^2 \qquad \text{for arbitrary natural frequency } \omega_n > 0 \text{ and damping ratio } \zeta > 0. \tag{8.8}$$

(iii) The canonical form of a *stable polynomial of 3rd order* is

$(s^2 + 2\zeta\omega_n s + \omega_n^2)(s + \alpha\omega_n)$ for arbitrary natural frequency $\omega_n > 0$, damping ratio $\zeta > 0$, and scaling parameter $\alpha > 0$,

with special cases:

- $\zeta = \alpha = 1$, the three roots are at $-\omega_n$, and the polynomial is

$$(s^2 + 2\omega_n s + \omega_n^2)(s + \omega_n) = s^3 + 3\omega_n s^2 + 3\omega_n^2 s + \omega_n^3 = (s + \omega_n)^3, \tag{8.9}$$

- $\alpha = 1$ and $\zeta = \frac{1}{2}$, the roots are at $-\omega_n$ and $\left(-\frac{1}{2} \pm \frac{\sqrt{3}i}{2}\right)\omega_n$, and the polynomial is

$$(s^2 + \omega_n s + \omega_n^2)(s + \omega_n) = s^3 + 2\omega_n s^2 + 2\omega_n^2 s + \omega_n^3. \tag{8.10}$$

8.2.1 Specifications for transient performance for complex conjugate poles

We report Figure 5.10 in Section 5.3.6 for convenience:

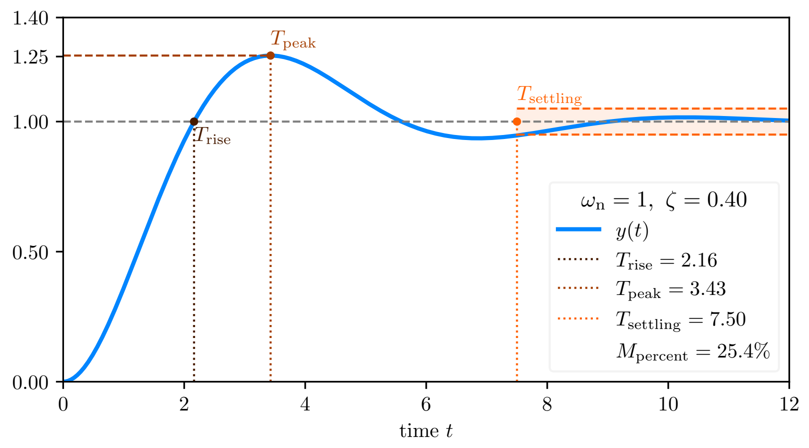


Figure 8.6: Transient performance of an *underdamped* second order system with natural frequency ω_n and damping ratio $\zeta < 1$. It is useful to recall the formulas: $T_{\text{rise},10\%\text{-}90\%} \approx \frac{1.8}{\omega_n}$, $T_{\text{settling }1\%} = \frac{5}{\zeta\omega_n}$, and $M_{\text{percent}} = \mathrm{e}^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}}$.

Based on this figure and the formulas in its caption, we can establish the following specifications for an underdamped system ($\zeta < 1$):

(i) if we aim to have a small rise time $T_{\text{rise},10\%\text{-}90\%} \approx \frac{1.8}{\omega_n}$, then we need a large ω_n,

(ii) if we aim to have a small settling time $T_{\text{settling }1\%} = \frac{5}{\zeta\omega_n}$, then we need to large $\zeta\omega_n$ (while keeping $\zeta < 1$), and

(iii) if we wish to have a small percent overshoot $M_{\text{percent}} = \mathrm{e}^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}}$, then we need a large ζ (while keeping $\zeta < 1$).

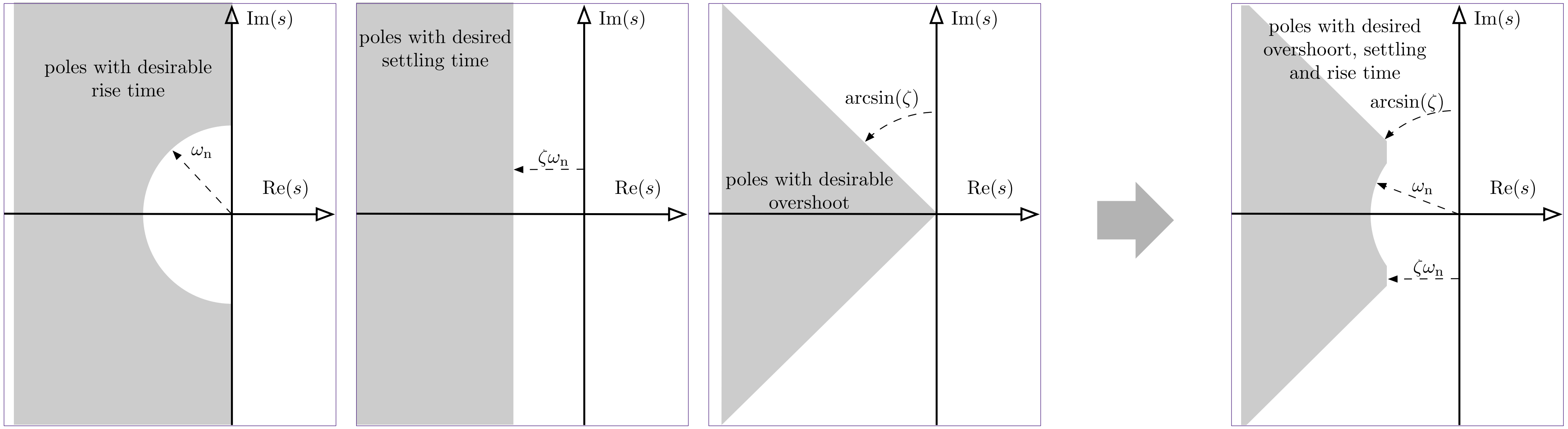

Im(s)
Re(s)
poles with desirable rise time
ω_n
poles with desired settling time
$\zeta\omega_n$
poles with desirable overshoot
arcsin(ζ)
poles with desired overshoort, settling and rise time

Figure 8.7: From time specifications to desired pole placement regions (for complex conjugate poles). Left panels: Regions where the poles satisfy the following requirements: percent overshoot, rise time, and settling time, respectively. Right panel: Regions where the poles satisfy simultaneously all three requirements.

8.2.2 P control of first-order systems

We consider

$$\text{first-order system} \qquad \boxed{G(s) = \frac{Y(s)}{U(s)} = \frac{b}{s+a}} \tag{8.11}$$

$$\text{P controller} \qquad \boxed{C(s) = k_{\mathrm{P}}} \tag{8.12}$$

From equation (7.9), the resulting closed-loop system[1] is

$$\boxed{G_{\text{closed-loop}}(s) = \frac{Y(s)}{R(s)} = \frac{bk_{\mathrm{P}}}{s + (a + bk_{\mathrm{P}})}} \tag{8.13}$$

We now match the denominator to the 1st-order polynomial (8.7). Given a desired time constant τ, the control gain is:

$$\text{control gain selection} \qquad \boxed{k_{\mathrm{P}} = \frac{1/\tau - a}{b}} \tag{8.14}$$

Therefore, the P controller achieves closed-loop stability and arbitrary pole placement, but not exact reference tracking since $G_{\text{closed-loop}}(0) = \frac{bk_{\mathrm{P}}}{a+bk_{\mathrm{P}}} \neq 1$. Note: no assumptions are made on the sign of the system coefficient a. Hence, the P controller with gain (8.14) can not only improve the time constant of a stable system, but also stabilize unstable first-order systems.

[1]This closed-loop transfer function gives the identical input/output relationship as we obtained in the time-domain analysis in Section ??.

8.2.3 PI control of first-order systems

We consider

first-order system
$$G(s) = \frac{Y(s)}{U(s)} = \frac{b}{s+a} \tag{8.15}$$

PI controller
$$C(s) = k_{\mathrm{P}} + \frac{k_{\mathrm{I}}}{s} \tag{8.16}$$

From equation (7.9), the resulting closed-loop system[2] is

$$G_{\text{closed-loop}}(s) = \frac{Y(s)}{R(s)} = \frac{bk_{\mathrm{P}}s + bk_{\mathrm{I}}}{s^2 + (a + bk_{\mathrm{P}})s + bk_{\mathrm{I}}} \tag{8.17}$$

We now match the denominator to the 2nd-order polynomial (8.8). Given a desired natural frequency ω_{n} and damping ratio ζ, the control gains are:

control gain selection
$$k_{\mathrm{P}} = \frac{2\zeta\omega_{\mathrm{n}} - a}{b} \quad \text{and} \quad k_{\mathrm{I}} = \frac{\omega_{\mathrm{n}}^2}{b} \tag{8.18}$$

Therefore, the PI controller achieves closed-loop stability, arbitrary pole placement, and exact reference tracking since $G_{\text{closed-loop}}(0) = \frac{bk_{\mathrm{I}}}{bk_{\mathrm{I}}} = 1$. Note: no assumptions are made on the sign of the system coefficient a. Hence, the PI controller with gains (8.18) stabilizes unstable first-order systems.

[2]This closed-loop transfer function gives the identical input/output relationship as we obtained in the time-domain analysis in equation (7.8) in Section 7.3 (where $d = 0$ and r is constant).

8.2.4 PD control of second-order systems

We consider

$$\text{second-order system} \qquad G(s) = \frac{Y(s)}{U(s)} = \frac{b}{s^2 + a_1 s + a_2} \tag{8.19}$$

$$\text{PD controller} \qquad C(s) = k_{\mathsf{P}} + k_{\mathsf{D}} s \tag{8.20}$$

From equation (7.9), the resulting closed-loop system is

$$G_{\text{closed-loop}}(s) = \frac{Y(s)}{R(s)} = \frac{b k_{\mathsf{D}} s + b k_{\mathsf{P}}}{s^2 + (a_1 + b k_{\mathsf{D}})s + (a_2 + b k_{\mathsf{P}})} \tag{8.21}$$

We now match the denominator to the 2nd-order polynomial (8.8). Given a desired natural frequency ω_{n} and damping ratio ζ, the control gains are:

$$\text{control gain selection} \qquad k_{\mathsf{P}} = \frac{\omega_{\mathsf{n}}^2 - a_2}{b} \qquad \text{and} \qquad k_{\mathsf{D}} = \frac{2\zeta\omega_{\mathsf{n}} - a_1}{b} \tag{8.22}$$

Therefore, the PD controller achieves closed-loop stability and arbitrary pole placement. Note: no assumptions are made on the sign of the system coefficients a_1 and a_2. Hence, the PD controller with gains (8.22) stabilizes unstable second-order systems.

However, the PD controller does not achieve exact reference tracking since:

$$G_{\text{closed-loop}}(0) = \frac{b k_{\mathsf{P}}}{a_2 + b k_{\mathsf{P}}} = 1 - \frac{a_2}{\omega_{\mathsf{n}}^2} \tag{8.23}$$

8.2.5 PID control of second-order systems

We consider

$$\text{second-order system} \qquad G(s) = \frac{Y(s)}{U(s)} = \frac{b}{s^2 + a_1 s + a_2} \tag{8.24}$$

$$\text{PID controller} \qquad C(s) = k_\mathsf{P} + \frac{k_\mathsf{I}}{s} + k_\mathsf{D} s \tag{8.25}$$

From equation (7.9), the resulting closed-loop system is

$$G_{\text{closed-loop}}(s) = \frac{Y(s)}{R(s)} = \frac{b k_\mathsf{D} s^2 + b k_\mathsf{P} s + b k_\mathsf{I}}{s^3 + (a_1 + b k_\mathsf{D}) s^2 + (a_2 + bk) s + b k_\mathsf{I}} \tag{8.26}$$

We now match the denominator to the simplified 3nd-order polynomial (8.9) (with three roots at $-\omega_\mathsf{n}$). Given a desired natural frequency ω_n, the control gains are:

$$\text{control gain selection} \qquad k_\mathsf{P} = \frac{3\omega_\mathsf{n}^2 - a_2}{b} \qquad k_\mathsf{I} = \frac{\omega_\mathsf{n}^3}{b} \qquad \text{and} \qquad k_\mathsf{D} = \frac{2\omega_\mathsf{n} - a_1}{b}. \tag{8.27}$$

The PID controller achieves closed-loop stability, arbitrary pole placement, and exact reference tracking since $G_{\text{closed-loop}}(0) = 1$.

8.3 Root locus analysis and design methods

The *root locus* is a classical control analysis technique that visualizes how the closed-loop poles of a system move in the complex plane as a scalar gain k varies from 0 to $+\infty$.

For a plant with open-loop transfer function:

$$G(s) = \frac{\mathrm{Num}(s)}{\mathrm{Den}(s)} \tag{8.28}$$

where $\mathrm{Num}(s)$ and $\mathrm{Den}(s)$ are polynomials in s, root locus analysis helps design controllers by choosing appropriate gains and compensators to achieve desired closed-loop performance.

The basic idea

The root locus is used to design controllers by choosing a gain k (and sometimes shaping the plant with added poles/zeros) such that the closed-loop poles lie in desired locations to meet performance specs (stability, damping, overshoot, etc).

For a feedback system with proportional gain k, the closed-loop transfer function is

$$G_{\text{closed-loop}}(s) = \frac{kG(s)}{1 + kG(s)}. \tag{8.29}$$

The closed-loop poles are the roots of the characteristic equation

$$1 + kG(s) = 0 \qquad \Longleftrightarrow \qquad \mathrm{Den}(s) + k\,\mathrm{Num}(s) = 0. \tag{8.30}$$

As k varies from 0 to $+\infty$, the roots of this equation trace out the *root locus* (locus means location in Latin).

8.3.1 Basic examples

We now show examples of root locus plots for first, second and third order systems, for a system with a zero. We show open-loop poles with an $\times$ symbol and closed-loop poles (the root locus) with blue lines. For the second-order system with a zero, and for the third-order system, we obtain a root locus with multiple *branches*.

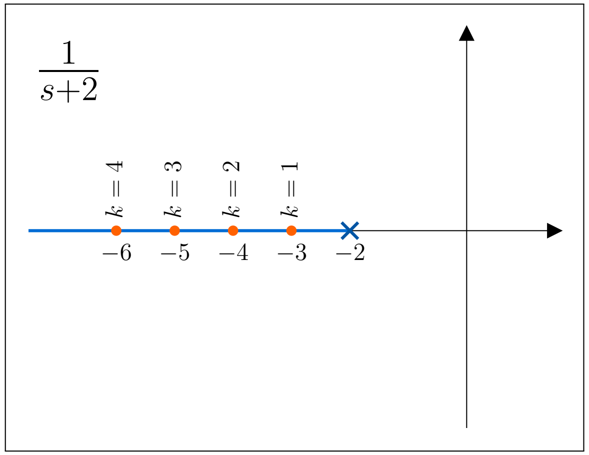


Figure 8.8: Root locus of the first-order system $G(s) = \frac{1}{s+2}$. The system has a single open-loop pole at $s = -2$. The closed-loop characteristic equation is $1 + kG(s) = 0$, i.e.,

$$1 + \frac{k}{s+2} = 0 \qquad \iff \qquad s = -2 - k.$$

As the gain k increases from 0 to $+\infty$, the closed-loop pole moves along the real axis to the left, approaching $-\infty$.
This example illustrates the simplest possible root locus: a single branch that remains entirely on the real line and demonstrates how proportional gain alone shifts the pole location without introducing oscillatory behavior.

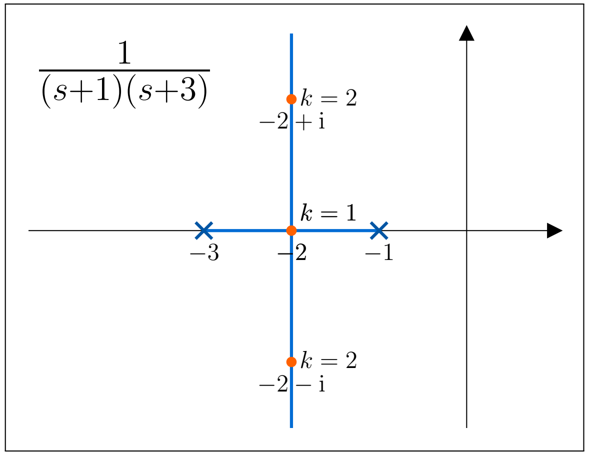


Figure 8.9: Root locus of the second-order system $G(s) = \frac{1}{(s+1)(s+3)}$. The open-loop poles are at $s = -1$ and $s = -3$. The closed-loop characteristic equation is $1 + kG(s) = 0$, i.e.,

$$1 + \frac{k}{(s+1)(s+3)} = 0 \qquad \iff \qquad (s+1)(s+3) + k = 0.$$

Expanding yields $s^2 + 4s + 3 + k = 0$, so the closed-loop poles are
$s = -2 \pm \sqrt{1-k}$.
For $0 < k < 1$, the poles are real and distinct; for $k = 1$, they coincide at $s = -2$; and for $k > 1$, they form a complex-conjugate pair moving into the left half-plane. Thus, the root locus begins on the real axis at $s = -1$ and $s = -3$, then branches off into the complex plane as k grows, illustrating how higher gain can introduce oscillatory dynamics.

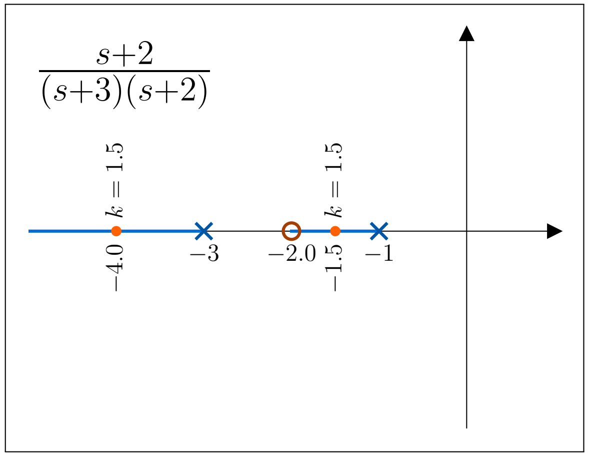


Figure 8.10: Root locus of the second-order system with a zero, $G(s) = \dfrac{s+1}{s(s+2)}$. The open-loop poles are at $s = 0$ and $s = -2$, and there is a finite zero at $s = -1$. The closed-loop characteristic equation is $1 + kG(s) = 0$, i.e.,

$$1 + \frac{k(s+1)}{s(s+2)} = 0 \quad \Longleftrightarrow \quad s(s+2) + k(s+1) = 0.$$

Expanding yields $s^2 + (2+k)s + k = 0$. The closed-loop poles are

$$s = -\frac{2+k}{2} \pm \tfrac{1}{2}\sqrt{(2+k)^2 - 4k}.$$

At $k = 0$, the poles are at $s = 0$ and $s = -2$. As k increases, one pole moves leftward along the real axis while the other approaches the zero at $s = -1$, so the root locus begins at the open-loop poles and terminates at the finite zero and at $-\infty$.

This is an example of a root locus with two *branches*.

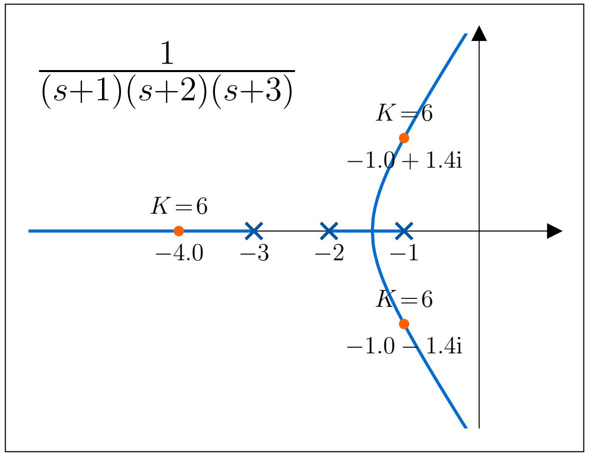


Figure 8.11: Root locus of the third-order system $G(s) = \dfrac{1}{s(s+3)(s+6)}$. The open-loop poles are at $s = 0$, $s = -3$, and $s = -6$. The closed-loop characteristic equation is $1 + kG(s) = 0$, i.e.,

$$1 + \frac{k}{s(s+3)(s+6)} = 0 \quad \Longleftrightarrow \quad s(s+3)(s+6) + k = 0.$$

Expanding yields $s^3 + 9s^2 + 18s + k = 0$. The closed-loop poles are the roots of this cubic, which depend on k. At $k = 0$ the poles coincide with the open-loop poles. As k increases, one branch of the root locus proceeds toward $-\infty$ along the real axis, while the other two form a complex-conjugate pair that moves into the left half-plane. Thus, the root locus illustrates how increasing gain shifts pole locations and can induce oscillatory dynamics in a third-order system.

One can use mathematical software to plot the root locus, e.g., the `ctrl.root_locus` routing inside the Python Control Systems Library.

```python
# Python code to plot the root locus

import control as ctrl
import matplotlib.pyplot as plt

# Define a transfer function G(s) = 1 / (s^2 + 4s + 3)
num = [1]
den = [1, 4, 3]
sys = ctrl.tf(num, den)

# Plot the root locus
ctrl.root_locus(sys)

# Show the figure
plt.show()
```

8.3.2 Short incomplete description of root locus construction rules

For an open-loop system $G(s) = \frac{\mathrm{Num}(s)}{\mathrm{Den}(s)}$, let

- $p_1, \dots, p_n$ be the n open-loop poles (i.e., the roots of $\mathrm{Den}(s)$), and
- $z_1, \dots, z_m$ be the m open-loop zeros (i.e., the roots of $\mathrm{Num}(s)$).

Then the root locus can be (partially) constructed using these rules:

(R1) **Symmetry:** The root locus is symmetric about the real axis.

(R2) **Odd real-axis rule:** A point on the real axis belongs to the root locus if the number of real poles and zeros to its right is odd.

(R3) **Start and end points:** The root locus starts at the open-loop poles $p_1, \dots, p_n$ and ends at the open-loop zeros $z_1, \dots, z_m$. If the number of poles exceeds the number of zeros ($n > m$), then $n - m$ branches of the locus do not terminate at finite zeros but instead go to infinity.

(R4) **Equispaced asymptotes:** The $n - m$ infinite branches described in rule (R3) follow straight-line asymptotes. Their angles θ_h depend on $n - m$:

$$\begin{aligned} n-m=1: &\quad \theta_0 = \pi, \\ n-m=2: &\quad \theta_0 = \tfrac{\pi}{2}, \quad \theta_1 = \tfrac{3\pi}{2}, \\ n-m=3: &\quad \theta_0 = \tfrac{\pi}{3}, \quad \theta_1 = \pi, \quad \theta_2 = \tfrac{5\pi}{3}, \\ &\vdots \end{aligned}$$

In general,

$$\theta_h = \frac{(2h+1)\pi}{n-m}, \qquad h \in \{0, \dots, n-m-1\}.$$

Additionally, these asymptotes intersect the real axis at the point

$$\frac{\sum_{i=1}^{n} p_i - \sum_{j=1}^{m} z_j}{n-m}.$$

In the Figures 8.12 and 8.13 we verify visually that all four rules are satisfied.

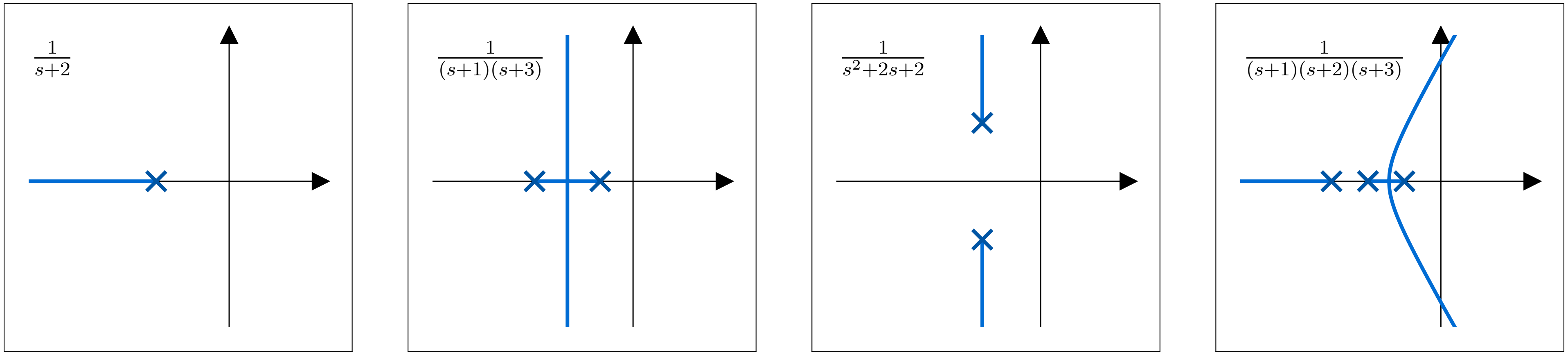


Figure 8.12: Basic root locus: Root locus of transfer functions with one, two real, two complex conjugate, or three real poles. Each locus is symmetric (R1). Segments of the locus belonging to the horizontal axis are to the left of an odd number of poles and zeros (R2). Branches go from poles to zeros or escape to infinity along asymptotes (R3) that are equispaces in an angular sense (R4).

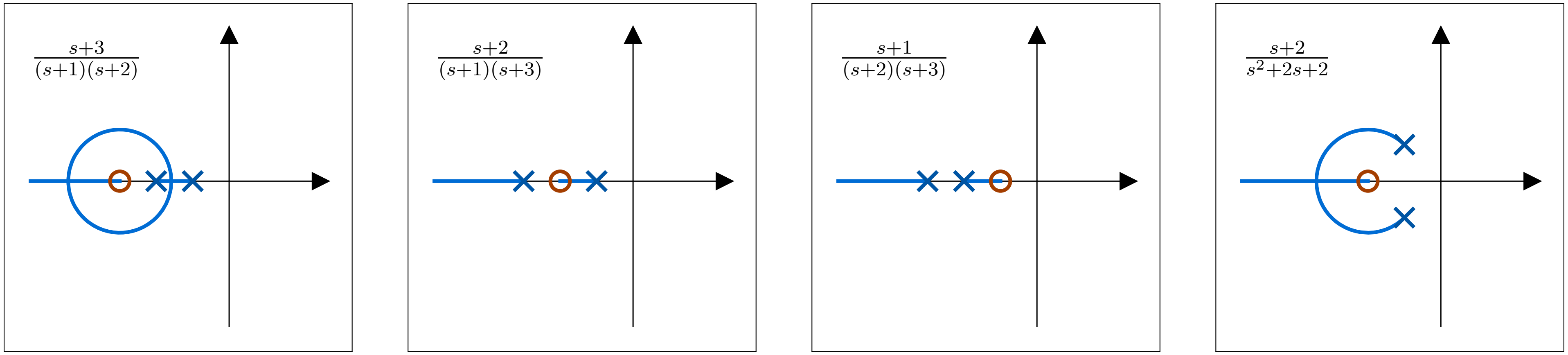


Figure 8.13: The effect of adding zeros: Root locus of a transfer function with two real poles and a zero. We illustrate all four possible relative placements of poles and zero. Note: there always is a branch going to $-\infty$. We leave it to the reader to visually verify that all four root locus rules are satisfied.

8.3.3 Design via root locus

We begin with some *illustrative consequences*:

(i) If the relative degree $n - m$ is greater than or equal to 3 or if the open-loop system has a right-half-plane zero, then a large gain k inevitably leads to instability.

(ii) For a first-order system without zeros, the root locus (Figure 8.8) allows direct selection of a gain k to achieve the desired closed-loop time constant (cf. Section 8.2.2).

(iii) For a second-order overdamped system without zeros, the root locus (Figure 8.9) allows direct selection of a gain k to achieve a desired closed-loop damping ratio.

These examples motivate a more systematic *design procedure*:

(Step 1) Plot the root locus of $G(s)$ using standard rules or software.

(Step 2) Identify desired closed-loop pole locations (e.g., from settling time and overshoot specifications).

(Step 3) Select the gain k that places poles closest to these target locations.

(Step 4) If necessary, reshape the root locus by adding poles or zeros (e.g., through PD, PI, lead, or lag compensators).

Finally, we summarize some practical *controller design insights*:

Proportional (P): $C(s) = k_\mathsf{P}$. Simply scales the open-loop gain. This moves the closed-loop poles along the existing root locus but does not change its shape. Useful for quick tuning, but offers limited design flexibility.

Proportional-Derivative (PD): $C(s) = k_\mathsf{P} + k_\mathsf{D}s = k_\mathsf{P}(1 + s\tau_\mathsf{D})$, where $\tau_\mathsf{D} = k_\mathsf{D}/k_\mathsf{P}$. Introduces a zero at $s = -1/\tau_\mathsf{D}$ in the left half-plane, bending the root locus leftward and improving transient response and stability margins. (Recall that the D controller is an improper transfer function, as discussed in Appendix 6.3.)

Proportional–Integral (PI): $C(s) = k_\mathsf{P} + \frac{k_\mathsf{I}}{s} = k_\mathsf{P}\big(1 + \frac{1}{\tau_\mathsf{I} s}\big)$, where $\tau_\mathsf{I} = k_\mathsf{P}/k_\mathsf{I}$. Introduces a pole at the origin and a zero at $s = -1/\tau_\mathsf{I}$ in the left half plane. This ensures zero steady-state error for step inputs, but the added pole can slow down the transient response.

Lead compensator: $C(s) = k\frac{s+z_c}{s+p_c}$ with $z_c < p_c$. Attracts the root locus leftward, improving transient response and stability margins.

Lag compensator: $C(s) = k\frac{s+z_c}{s+p_c}$ with $z_c > p_c$. Increases low-frequency gain (better steady-state accuracy) with little effect on transient response.

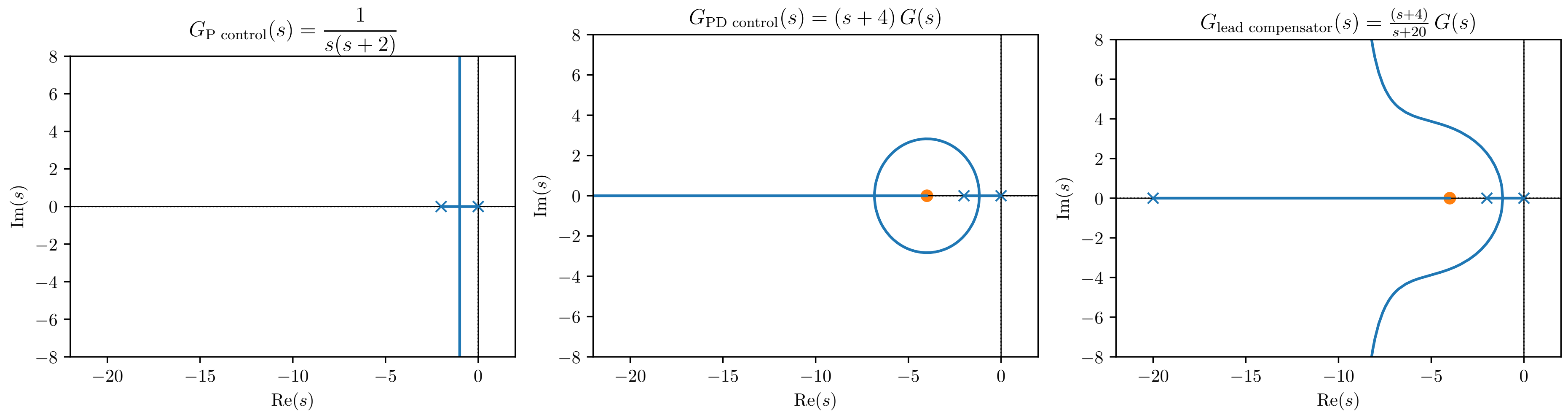


Figure 8.14: Root locus design for the system $G(s) = 1/(s(s+2))$.
Left image: With proportional control, both poles remain close to the imaginary axis and performance is poor.
Center image: Adding derivative action, $C(s) = s + 4$, introduces a zero that bends the locus leftward and improves stability and transient response, but at the cost of an improper controller.
Right image: A proper lead compensator, $C(s) = (s+4)/(s+20)$, preserves the benefits of derivative action while ensuring physical realizability. Note that the pole at -20 is 5x slower than the zero at -4, this is a useful property as discussed in Section 5.4.
Lesson: Lead compensation offers a practical way to shift poles leftward, combining stability improvement with realizable controller structure.

8.3.4 Final observations about the root locus

Root locus analysis offers several distinct advantages for control system design. The method is graphical and intuitive, providing direct insight into how system poles move as gain varies. This makes it particularly useful for understanding stability boundaries and transient response characteristics. It works especially well for low-order, single-input single-output systems, where the geometric relationships are clear and easily interpreted:

- Graphical and intuitive.
- Works well for low-order SISO systems.
- Provides direct insight into pole location and transient response.

At the same time, the technique has important limitations that must be recognized. For high-order systems the locus patterns become complex and less intuitive, and the method does not directly extend to multiple-input multiple-output systems. Moreover, root locus analysis only shows the effect of varying a single gain parameter and does not readily address design problems with multiple constraints or optimization goals:

- Hard to use for high-order systems.
- Not suited for constraints or optimization.
- Often requires trial-and-error or frequency-domain design.

In summary, root locus is best regarded as a valuable conceptual and graphical tool, well suited for developing intuition and preliminary designs, but not as a comprehensive method for modern control-system design.

8.4 Analysis of instability mechanisms

Everything should be made as simple as possible, but not simpler. Albert Einstein

In this section we discuss multiple cases where you think you designed a fully satisfactory controller and yet, instability lurks beneath an overly-simplified mathematical analysis. We consider:

(i) unmodeled nonlinear dynamics (due for example to linearization),

(ii) bounded control in Section 8.4.2,

(iii) bounded control rate in Section 8.4.3,

(iv) unmodeled linear dynamics in Section 8.4.4,

(v) delays in Section 8.4.5.

8.4.1 Nonlinearities might render a linear system stable only locally

Consider

$$\dot{x} = -x + x^3 = (x^2 - 1)x. \tag{8.31}$$

Its linearization at the equilibrium $x^* = 0$ is $\dot{\delta x} = -\delta x$ and it appears to be stable. But it is easy to see, for example from its phase portrait, that the origin is only locally stable and that all trajectories with $x(0) > 1$ (respectively $x(0) < -1$) diverge to $+\infty$ (respectively $-\infty$). In short, the system is only locally stable.

8.4.2 Bounded control can stabilize an unstable system only locally

When the uncontrolled system is naturally unstable, control strategies can provide only local stability because of an inevitable realistic constraint: a maximum value on control actuation.

We consider the simplest example:

$$\dot{x} = rx + u \tag{8.32}$$

where the growth rate is positive $r > 0$ so that the system at $u = 0$ is unstable. We then assume that $u(t)$ cannot take values larger than u_{max} or smaller than $-u_{\mathsf{max}}$. For simplicity of notation, we set $r = u_{\mathsf{max}} = 1$ (the analysis of the general case is identical, but more cumbersome to follow).

As control design, we would like to design $u = -2x$, but we cannot apply a control signal larger than 1 and so the closed loop system is:

$$\dot{x} = x + \operatorname{sat}(-2x). \tag{8.33}$$

where we define the saturation nonlinearity as in Figure 8.15.

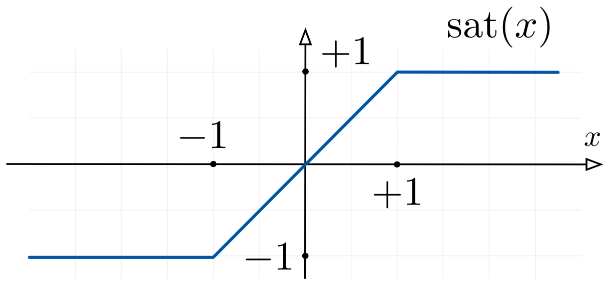
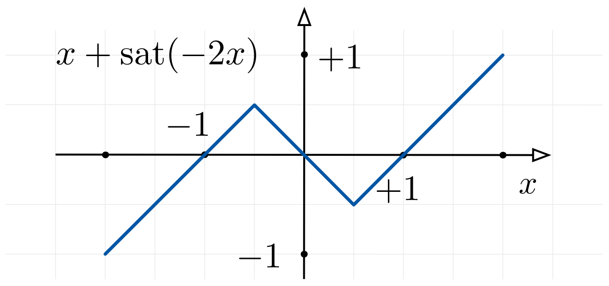


Figure 8.15: Consider the *saturation nonlinearity* $\operatorname{sat}(x) = \begin{cases} +1 & x > 1 \\ x & -1 \leq x \leq 1 \\ -1 & x < -1 \end{cases}$.

We now plot the right-hand side as a function of x and the phase portrait in Figure 8.16.



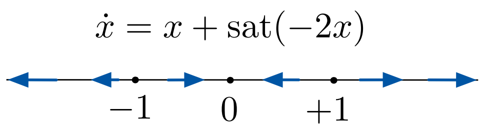


Figure 8.16: Saturation-induced instability. Left image: the right-hand side. Right image: the phase portrait.

This plot shows that the dynamical system $\dot{x} = x + \mathrm{sat}(-2x)$

(i) has three equilibrium points: -1, 0, and $+1$,

(ii) the equilibrium points 0 is stable,

(iii) the equilibrium points -1 and $+1$ are unstable,

(iv) each trajectory starting from $x(0) > 1$ goes to $+\infty$ and each trajectory starting from $x(0) < -1$ goes to $-\infty$.

Hence, the equilibrium point 0 is only *locally stable*. From this simple example we learn these lessons:

> *Unstable systems can never be operated without control systems regulating their behavior.*
>
> *However, unstable systems are fundamentally more difficult to control. Specifically, because of inevitable magnitude and rate constraints on the control signal, closed-loop systems with unstable processes are only locally stable.*

8.4.3 Control with bounded rate of change can stabilize an unstable system only locally

We now consider the same dynamical system as before

$$\dot{x} = x + u \tag{8.34}$$

but now assume that $\dot{u}(t)$ cannot take values larger than a constant $\dot{u}_{\text{max}}$, which again for simplicity take equal to 1.

To design a controller with limited derivative with respect to time, we taking derivatives with respect to time, we obtain

$$\ddot{x} = \dot{x} + \dot{u}. \tag{8.35}$$

We now have an unstable second-order system with a bounded control. We would like to design $\dot{u} = -2\dot{x} - x$ because, without saturation, the resulting closed loop system would be: $\ddot{x} + \dot{x} + x = 0$, a stable underdamped second-order system. Because of the rate-saturation we obtain instead the closed-loop system:[3]

$$\ddot{x} = \dot{x} + \text{sat}(-2\dot{x} - x). \tag{8.36}$$

We plot the phase portrait of the dynamics (8.36) in Figure 8.17.

[3] It is also important to take into account the equality: $\dot{x}(0) = x(0) + u(0)$. In what follows we will assume that $u(0) = 0$ and focus on non-zero $x(0) = \dot{x}(0)$.

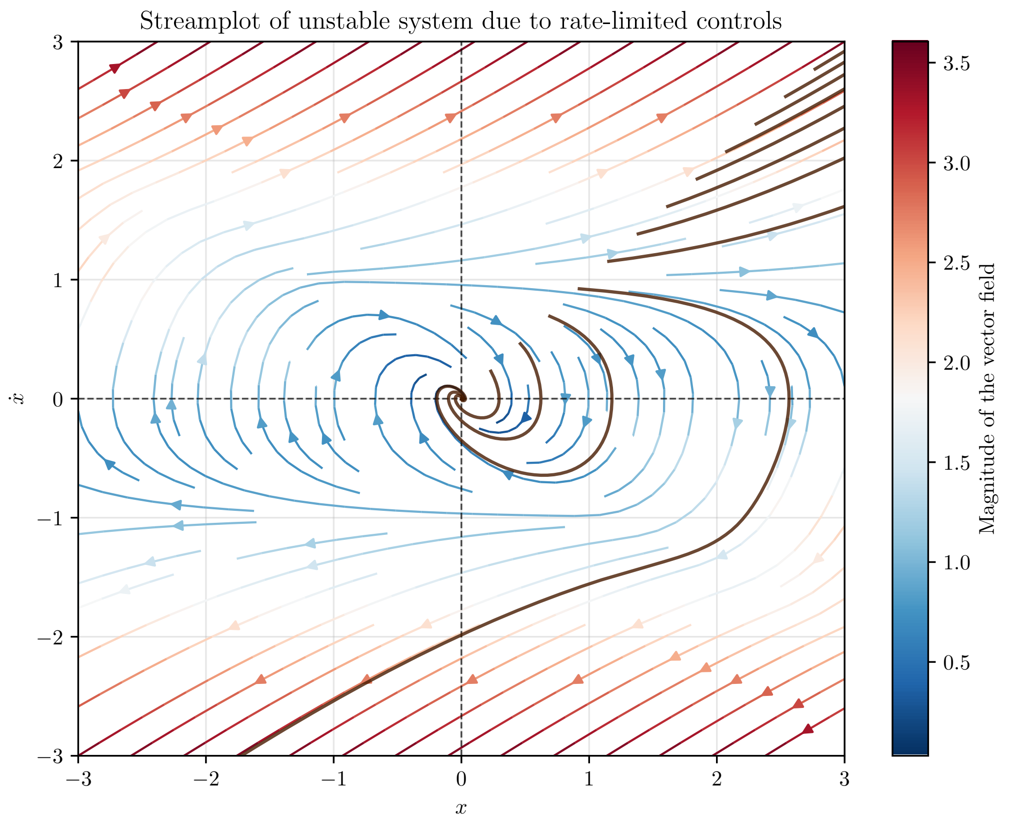


Figure 8.17: Saturation-induced instability: the rate limitation case. Note that the initial conditions satify $x(0) = \dot{x}(0)$ because we assume $u(0) = 0$.

We note that:

(i) the equilibrium $(x, \dot{x}) = (0, 0)$ is locally stable,

(ii) each trajectory from initial condition $x(0) > 1$ diverges to $+\infty$, and

(iii) each trajectory from initial condition $x(0) < -1$ diverges to $-\infty$.

(iv) (The precise calculation of which initial conditions converge to $(0, 0)$ and which diverge is beyond this simplified discussion.)

The overall lessons here are the same as in the previous example.

8.4.4 Systems with unmodeled dynamics and destabilizing control design

Inspired by (Åström and Murray, 2021, Section 2.3), we now study (i) systems whose dynamics is not fully modeled or fully known, and (ii) the undesirable consequences of using large control gains on such systems.

(i) We assume that the system to be controlled (called *complete system*) is described by the transfer function

$$G_{\text{complete}}(s) = \frac{k_{\text{sys}}}{(s\tau + 1)(Ts + 1)} \tag{8.37}$$

with system gain k_{sys} and two time constants τ and T satisfying $T \ll \tau$ (therefore, T describes a much faster dynamics than τ). The complete system has two poles at $-1/\tau$ and $-1/T$. We design a PI controller $C(s) = k_{\text{P}} + \dfrac{k_{\text{I}}}{s}$ and compute the closed-loop transfer function for the complete system:

$$\frac{C(s)G(s)}{1 + C(s)G(s)} = \frac{k_{\text{sys}}\,(k_{\text{I}} + k_{\text{P}}s)}{k_{\text{sys}}\,(k_{\text{I}} + k_{\text{P}}s) + s\,(Ts + 1)\,(s\tau + 1)} = \frac{k_{\text{sys}}k_{\text{P}}s + k_{\text{sys}}k_{\text{I}}}{\tau T s^3 + (T + \tau)\,s^2 + \big(k_{\text{P}}k_{\text{sys}} + 1\big)\,s + k_{\text{I}}k_{\text{sys}}} \tag{8.38}$$

Hint: This calculation is tedious, but not difficulr. There are five parameters: τ, T, k_{sys}, k_{P} and k_{I}.

(ii) We now consider the situation where we do not know the precise value of T, but we do know that $T \ll \tau$ (so that the pole at $-1/T$ is much faster than the one at $-1/\tau$). According to the approximation strategy given in Section 5.4, we decide to neglect the pole at $-1/T$ and consider the *simplified system*:

$$G_{\text{simplified}}(s) = \frac{k_{\text{sys}}}{s\tau + 1} \tag{8.39}$$

In other words, there is some system dynamics that is *unmodeled*. This situation is entirely typical, one cannot expect to model exactly every single phenomenon in the system dynamics.

We now design control gains using the definitions (7.31)-(7.32) given in Section **??** for the first-order simplified system $G_{\text{simplified}}(s) = \frac{k_{\text{sys}}}{s\tau+1}$. Given desirable natural frequency ω_{n} and damping ratio ζ, we design

$$k_{\text{I}} = \frac{\omega_{\text{n}}^2 \tau}{k_{\text{sys}}}, \qquad k_{\text{P}} = \frac{2\omega_{\text{n}}\tau\zeta - 1}{k_{\text{sys}}}. \tag{8.40}$$

Substituting these values into equation (8.38), we compute the closed-loop transfer function for the complete system:

$$\frac{C(s)G(s)}{1 + C(s)G(s)} = \frac{(2\omega_{\text{n}}\tau\zeta - 1)\,s + \omega_{\text{n}}^2\tau}{\tau T s^3 + (T + \tau)\,s^2 + 2\omega_{\text{n}}\tau\zeta s + \omega_{\text{n}}^2\tau} \tag{8.41}$$

Note: this closed-loop transfer function includes the neglected pole at $-1/T$.

(iii) We now study when the transfer function (8.41) is stable. The denominator is a third-order polynomial with positive coefficients. To determine if all poles of the transfer functions are stable, we apply the condition (5.45) from the Appendix 5.8. The closed-loop system is stable when

$$(T+\tau)\,2\omega_{\mathsf{n}}\tau\zeta > \tau T\omega_{\mathsf{n}}^2\tau \tag{8.42}$$

$$\iff \qquad \omega_{\mathsf{n}} < 2\zeta\frac{1+T/\tau}{T}. \tag{8.43}$$

This is a remarkable result: The closed-loop system is not always stable! Given the two time constants and the damping ratio, a sufficiently large value of ω_{n} leads to a closed-loop system that is unstable.

The open-loop complete system is stable, the open-loop simplified system is stable, but the closed-loop system (for the complete system, designed using the simplified system) may now be unstable.

In other words, using the definitions (7.31)-(7.32) in Section **??** assuming that the plan is $G(s) = \frac{k_{\mathsf{sys}}}{s\tau+1}$ must be done with care when selecting control gains.

(iv) We now provide a numerical example of this instability. We consider

$$\text{system:}\quad \tau = 1, \quad T = .1, \quad k_{\mathsf{sys}} = 1, \tag{8.44}$$

This means that $T = \tau/10$, which is consistent with the assumption $T \ll \tau$. Next, we select $\zeta = 0.4$ (as we discussed in Section 5.3.6, such a value of ζ leads to a fast response with 25% overshoot). Then, the closed-loop system is stable for

$$\omega_{\mathsf{n}} < 2\zeta \frac{1 + T/\tau}{T} = 0.88 \frac{1}{T} = 8.8 \tag{8.45}$$

We design our controller with a natural frequency larger than this threshold:

$$\text{desired controller:}\quad \zeta = 0.4, \quad \omega_{\mathsf{n}} = 10, \tag{8.46}$$

$$\text{PI gains :}\quad k_{\mathsf{I}} = 100, \quad k_{\mathsf{P}} = 7. \tag{8.47}$$

These parameters lead to

$$\frac{C(s)G(s)}{1 + C(s)G(s)} = \frac{7s + 100}{0.1s^3 + 1.1s^2 + 8s + 100} \tag{8.48}$$

which has the three poles (computed numerically) at:

$$s_1 = -11.6$$
$$s_2 = 0.28 \pm \mathrm{i}9.3$$

As predicted, the closed-loop system is unstable!

From this simple example we learn these lessons:

The choice of natural frequency ω_n for the closed-loop system need to be informed by an understanding of what are the time constant of unknown or unmodeled dynamics.

8.4.5 Systems with delays

Consider the familiar situation of a *microphone placed in front of a loudspeaker*. The microphone and the speaker are individually stable devices, but the amplifier drives the speaker using the microphone signal, and the speaker output is in turn picked up by the microphone. The result is an *acoustic feedback loop*, in which the propagation delay of the sound wave from the speaker to the microphone plays a key role. This interconnection can become unstable and produces the high-pitched squeal often heard in public address systems, known as the *Larsen effect*.

We model the microphone–amplifier–speaker system as a *first-order stable system with delay*:

$$G(s) = \frac{k}{\tau s + 1} \, \mathrm{e}^{-Ts} \, . \tag{8.49}$$

The parameters are:

- $G(s)$ effective microphone–amplifier–speaker transfer function,
- $\tau > 0$ time constant of the combined speaker and microphone dynamics,
- $T > 0$ propagation and electronic delay, and
- k loop gain of the amplifier.

The closed-loop transfer function with unity feedback is

$$G_{\text{closed-loop}}(s) = \frac{G(s)}{1 + G(s)} = \frac{k \, \mathrm{e}^{-Ts}}{\tau s + 1 + k \, \mathrm{e}^{-Ts}}. \tag{8.50}$$

Note that, when there is no delay $T = 0$, the closed-loop system is stable for all $k > 0$ (with a single pole at $-(1+k)/\tau$). We will see, in two distinct ways, that for small k, the closed-loop system is stable and that, as k increases, the closed-loop system becomes unstable.

First, we examine the root locus of $G_{\text{closed-loop}}(s)$ in equation (8.50). Because of the time delay e^{-Ts}, the closed-loop transfer function is not rational (i.e., it is not a ratio of polynomials). To obtain a rational approximation, we use *Padé approximants*, which are rational functions chosen so that their Taylor series matches that of e^{-Ts} up to a desired order (see also Exercise E4.12). The *first-order Padé approximation* of e^{-Ts} is

$$\mathrm{e}^{-sT} \approx \frac{1-\frac{sT}{2}}{1+\frac{sT}{2}} = \frac{2-sT}{2+sT}, \tag{8.51}$$

and the *second-order Padé approximant* is:

$$\mathrm{e}^{-sT} \approx \frac{1-\frac{sT}{2}+\frac{(sT)^2}{12}}{1+\frac{sT}{2}+\frac{(sT)^2}{12}} = \frac{(sT)^2-6(sT)+12}{(sT)^2+6(sT)+12}. \tag{8.52}$$

We do not report the higher-order approximant for simplicity; Padé approximants of arbitrary order can be computed directly in the Python Control Systems Library with the command `control.pade`.

Note that each of the Padé approximant has one or more *right-half plane zeros*. Because of this property, the root locus plots in Figure 8.18 show that a time delay always renders the closed-loop system unstable when the loop gain k is sufficiently large.

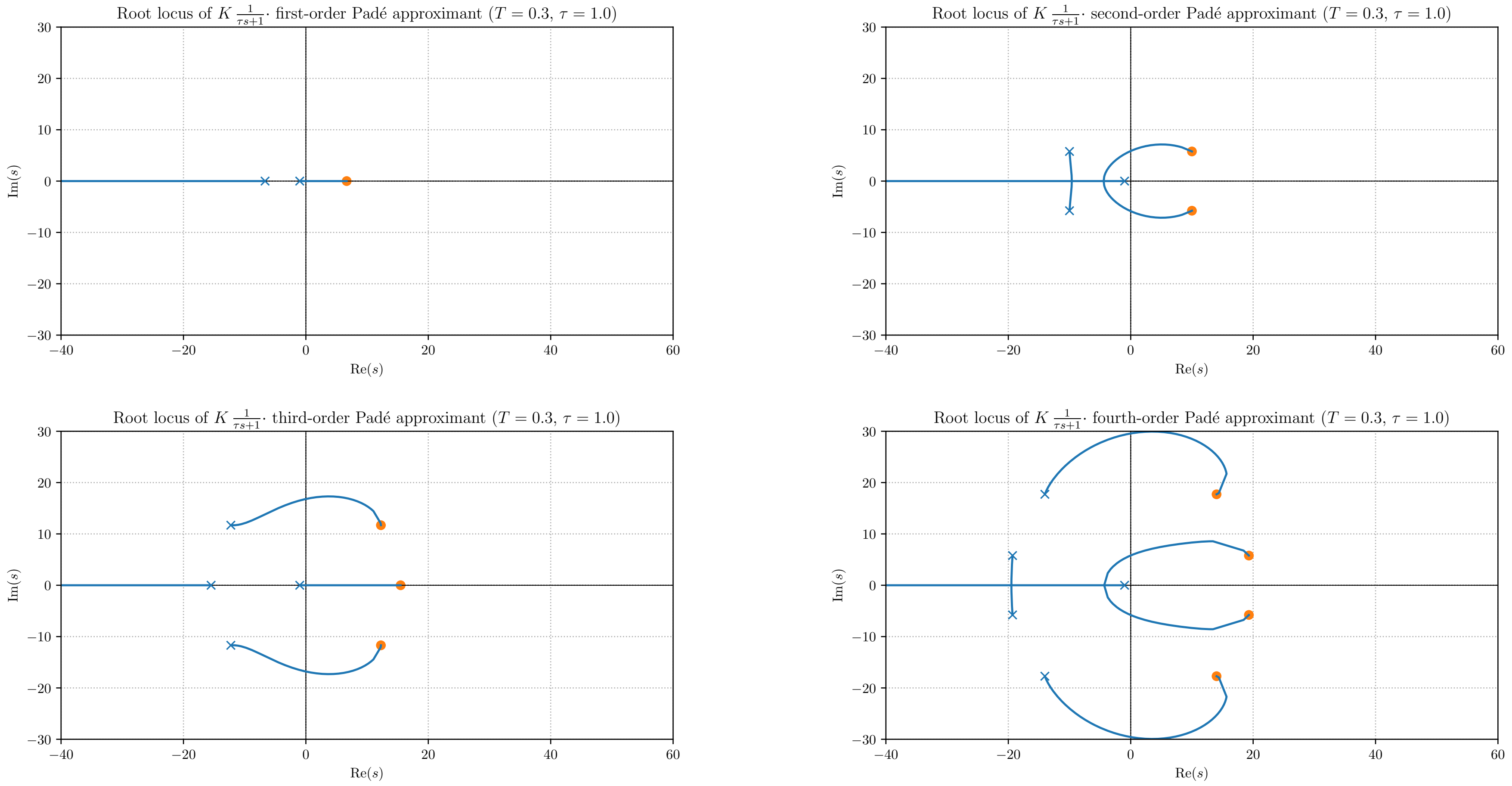


Figure 8.18: Root locus of the closed-loop transfer function $G_{\text{closed-loop}}(s)$ in equation (8.50) with Padé approximants of orders 1–4. Parameters: $\tau = 0.3$, $T = 1$. All approximations capture the onset of instability at large loop gain k, with the higher-order approximations giving a better estimate of the critical gain at which poles cross the imaginary axis.
Image generated by `delay-rootlocus.py` .

As a second approach to demonstrate instability as the loop gain k increases, we can directly simulate the closed-loop system with delays (we do not use Padé approximants now). For the unity-feedback loop with transfer function

$$G(s) = \frac{k}{\tau s + 1}\,\mathrm{e}^{-Ts}, \tag{8.53}$$

the time-domain closed-loop equation for a unit-step reference $r(t) = \mathbf{1}(t) = \mathbf{1}(t)$ is

$$\tau \dot{y}(t) + y(t) = k\big(r(t-T) - y(t-T)\big). \tag{8.54}$$

Figure 8.19 illustrates the resulting step responses for varying k. As predicted, for large k the trajectories grow without bound, showing instability.

Remark 8.1 (How to simulate delay differential equations). *A simple and effective approach to simulate the delay differential equation (8.54) is the method of steps with a history buffer. Using a forward Euler with step size h and a delay index $m = \mathrm{round}(T/h)$, the discrete-time update reads*

$$y_{n+1} = y_n + \frac{h}{\tau}\big(-y_n + k(r_{n-m} - y_{n-m})\big), \tag{8.55}$$

with $y_n = 0$ for $n \leq 0$ and $r_n = 1$ for $n \geq 0$. The step size h should be chosen to satisfy $h \ll \min\{\tau, T\}$ (e.g., $h = 10^{-3}$ when $T = 1$ and $\tau = 0.3$) in order to keep numerical error and spurious oscillations in check. A smaller h yields more accuracy. ●

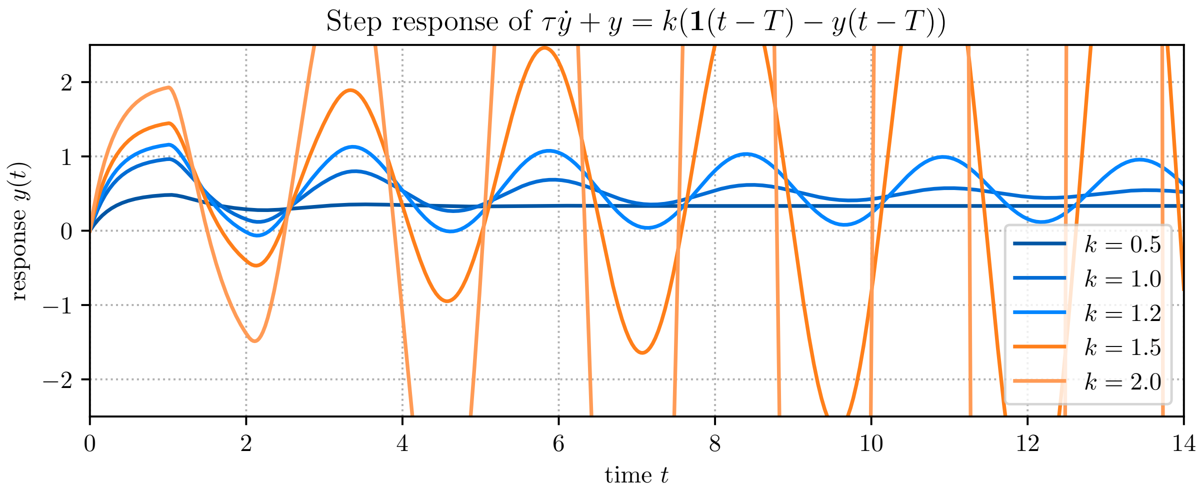


Figure 8.19: Step response of the delayed closed-loop system $\tau\dot{y} + y = k\,(1(t-T) - y(t-T))$ simulated by the method of steps with a history buffer. Parameters: $\tau = 0.3, T = 1.0$. The system is stable for small k (see $k \in \{0.5, 1, 1.2\}$), while for large k the response diverges (see $k \in \{1.5, 2\}$), demonstrating instability.
Image generated by `delay-stepresponse.py` .

8.5 Historical notes and further resources

PID controllers are routinely and commonly implemented on PLCs, or Programmable Logic Controllers. Modern PLCs include built-in PID function blocks that make it straightforward to configure and use the control algorithm for various industrial automation processes. For a useful discussion we refer to https://en.wikipedia.org/wiki/Programmable_logic_controller.

8.6 Appendix: The canonical control system and its transfer functions

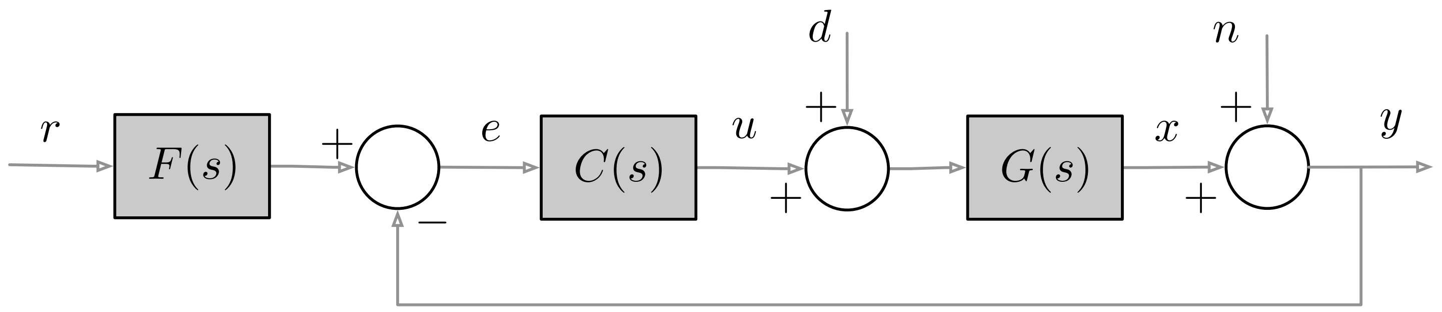


Figure 8.20: The canonical block diagram of a feedback control system, with reference input r, error e, control action u, disturbance d, state x, measurement noise n, and output y. The three main blocks are the process $G(s)$, the feedback controller $C(s)$, and an optional feedforward controller $F(s)$.

Signals:

• r	is the reference input,
• $e = r - y$	is the error signal,
• u	is the control action (the output of the controller),
• d	is the load disturbance signal (pushes state away from desired state),
• x	is the system state,
• n	is the measurement noise (corrupts information about the state x), and
• y	is the system output.

Blocks (described by transfer functions):

• $G(s)$	is the system or process,
• $C(s)$	is the feedback controller
• $F(s)$	is the feedforward controller

The inclusion of a feedforward controller $F(s)$ allows us to treat reference tracking separately from disturbance/noise rejection, leading to the so-called two-degree-of-freedom control design (i.e., the design of F and C).

Control objectives: The design task is to select $C(s)$ and $F(s)$ so that the closed-loop system achieves the following objectives:

(Objective 1) *reference tracking*: make the state x follow the reference signal r,

(Objective 2) *load disturbance rejection*: reduce the effect of the load disturbance d (on all signals),

(Objective 3) *measurement noise rejection*: reduce the effect of the measurement noise n (on all signals), and

(Objective 4) *sensitivity reduction*: make the closed-loop system insensitive to parameter variations.

Typically, *2 degree-of-freedom (dof) control design*:

- design C to achieve disturbance rejection (Objective 2), noise rejection (Objective 3), and sensitivity reduction (Objective 4).
- design F to achieve reference tracking (Objective 1).

There are many equivalent block diagrams that implement 2 dof control design.

8.6.1 The multiple transfer functions relevant in control system

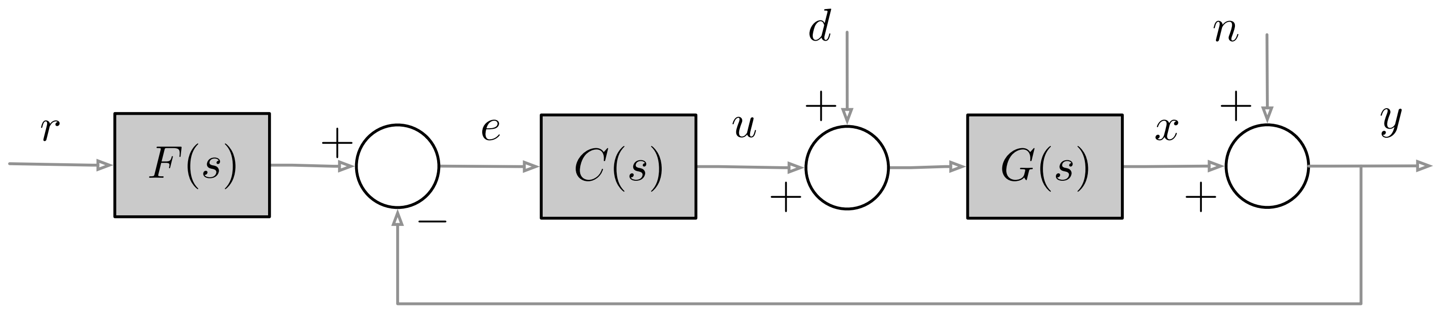


Figure 8.21: Canonical control system

From Figure 8.20, it is possible (and easy) to compute all relations between the inputs signals r, d, n and the signals of interest x, y, u. For example, considering one input at a time (set $r = 0$ and $n = 0$), one can see

$$X(s) = \frac{G(s)}{1 + G(s)C(s)} D(s) \tag{8.56}$$

Note that

- we call $L(s) = G(s)C(s)$ the *loop transfer function* (or open-loop transfer function),
- for $X(s)/D(s)$, but also for all other possible pairs:

$$\text{closed-loop transfer function from } D \text{ to } X = \frac{\text{open-loop transfer function from } D \text{ to } X}{1 + L(s)} = \frac{G(s)}{1 + G(s)C(s)}. \tag{8.57}$$

Keeping $r = 0$ and dropping the s variable, we can compute all transfer functions relevant for the design of C:

$$X = \frac{G}{1+GC} D - \frac{GC}{1+GC} N \qquad \text{(relevant for disturbance rejection (Objective 2))}$$

$$Y = \frac{G}{1+GC} D + \frac{1}{1+GC} N \qquad \text{(relevant for noise rejection (Objective 3))}$$

$$U = -\frac{GC}{1+GC} D - \frac{C}{1+GC} N \qquad \text{(relevant for sensitivity reduction (Objective 4))}$$

Note that there are only four distinct transfer functions, since two are repeated. Adopting nomenclature from (Åström and Murray, 2021), the *Gang of Four* transfer functions are:

sensitivity: $$S(s) := \frac{Y(s)}{N(s)} = \frac{1}{1+G(s)C(s)}$$

complementary sensitivity: $$T(s) := -\frac{X(s)}{N(s)} = -\frac{U(s)}{D(s)} = \frac{G(s)C(s)}{1+G(s)C(s)}$$

load→state sensitivity: $$\frac{X(s)}{D(s)} = \frac{Y(s)}{D(s)} = \frac{G(s)}{1+G(s)C(s)}$$

noise→control sensitivity: $$-\frac{U(s)}{N(s)} = \frac{C(s)}{1+G(s)C(s)}.$$

For the design of the feedforward controller F and the system response to a reference input r, we are interested also in the transfer functions from r to x, y, u. We can compute:

$$\begin{aligned} X &= \frac{G}{1+GC}D - \frac{GC}{1+GC}N + \frac{GCF}{1+GC}R && (8.58)\\ Y &= \frac{G}{1+GC}D + \frac{1}{1+GC}N + \frac{GCF}{1+GC}R && (8.59)\\ U &= -\frac{GC}{1+GC}D - \frac{C}{1+GC}N + \frac{CF}{1+GC}R && (8.60)\end{aligned}$$

Note that there are two additional transfer functions:

reference→control sensitivity: $\dfrac{U(s)}{R(s)} = \dfrac{C(s)F(s)}{1+G(s)C(s)}$

reference→state sensitivity: $\dfrac{X(s)}{R(s)} = \dfrac{G(s)C(s)F(s)}{1+G(s)C(s)}$

Remarks 8.2. *Some remarks are in order.*

(i) *A control engineer needs to analyze all 4 transfer functions to fully understand a closed-loop system*

(ii) *Additionally, if a feedforward controller is also included, then the transfer functions are 6.*

(iii) *Bottom line: look at step transient response and frequency response of all functions.*

8.6.2 Control design examples: PI control of first-order systems

A first-order process with time constant τ

$$G(s) = \frac{1}{\tau s + 1} \tag{8.61}$$

A PI feedback controller with gains k_{P} and k_{I}

$$C(s) = k_{\mathsf{P}} + \frac{k_{\mathsf{I}}}{s} = \frac{k_{\mathsf{P}} s + k_{\mathsf{I}}}{s} \tag{8.62}$$

Note the loop transfer function is $L(s) = \dfrac{k_{\mathsf{P}} s + k_{\mathsf{I}}}{s(\tau s + 1)}$. We compute the gang of four:

Sensitivity Y/N: $$\frac{1}{1+G(s)C(s)} = \frac{1}{1+\frac{k_{\mathsf{P}}s+k_{\mathsf{I}}}{s(\tau s+1)}} = \frac{s(\tau s+1)}{\tau s^2 + (1+k_{\mathsf{P}})s + k_{\mathsf{I}}}$$

Complementary Sensitivity $-X/N$: $$\frac{G(s)C(s)}{1+G(s)C(s)} = \frac{\frac{k_{\mathsf{P}}s+k_{\mathsf{I}}}{s(\tau s+1)}}{1+\frac{k_{\mathsf{P}}s+k_{\mathsf{I}}}{s(\tau s+1)}} = \frac{k_{\mathsf{P}}s+k_{\mathsf{I}}}{\tau s^2 + (1+k_{\mathsf{P}})s + k_{\mathsf{I}}}$$

Load$\to$State X/D Sensitivity: $$\frac{G(s)}{1+G(s)C(s)} = \frac{\frac{1}{\tau s+1}}{1+\frac{k_{\mathsf{P}}s+k_{\mathsf{I}}}{s(\tau s+1)}} = \frac{s}{\tau s^2 + (1+k_{\mathsf{P}})s + k_{\mathsf{I}}}$$

Noise$\to$Control U/N Sensitivity: $$\frac{C(s)}{1+G(s)C(s)} = \frac{\frac{k_{\mathsf{P}}s+k_{\mathsf{I}}}{s}}{1+\frac{k_{\mathsf{P}}s+k_{\mathsf{I}}}{s(\tau s+1)}} = \frac{(\tau s+1)(k_{\mathsf{P}}s+k_{\mathsf{I}})}{\tau s^2 + (1+k_{\mathsf{P}})s + k_{\mathsf{I}}}$$

After simplifications, it can be seen that the denominators are identical and of second order.

In the plots, we select $\tau = 2.0$, $k_{\mathsf{I}} = 0.5$, and $k_{\mathsf{P}} = \{0.01, 0.1, 1, 10, 100\}$.

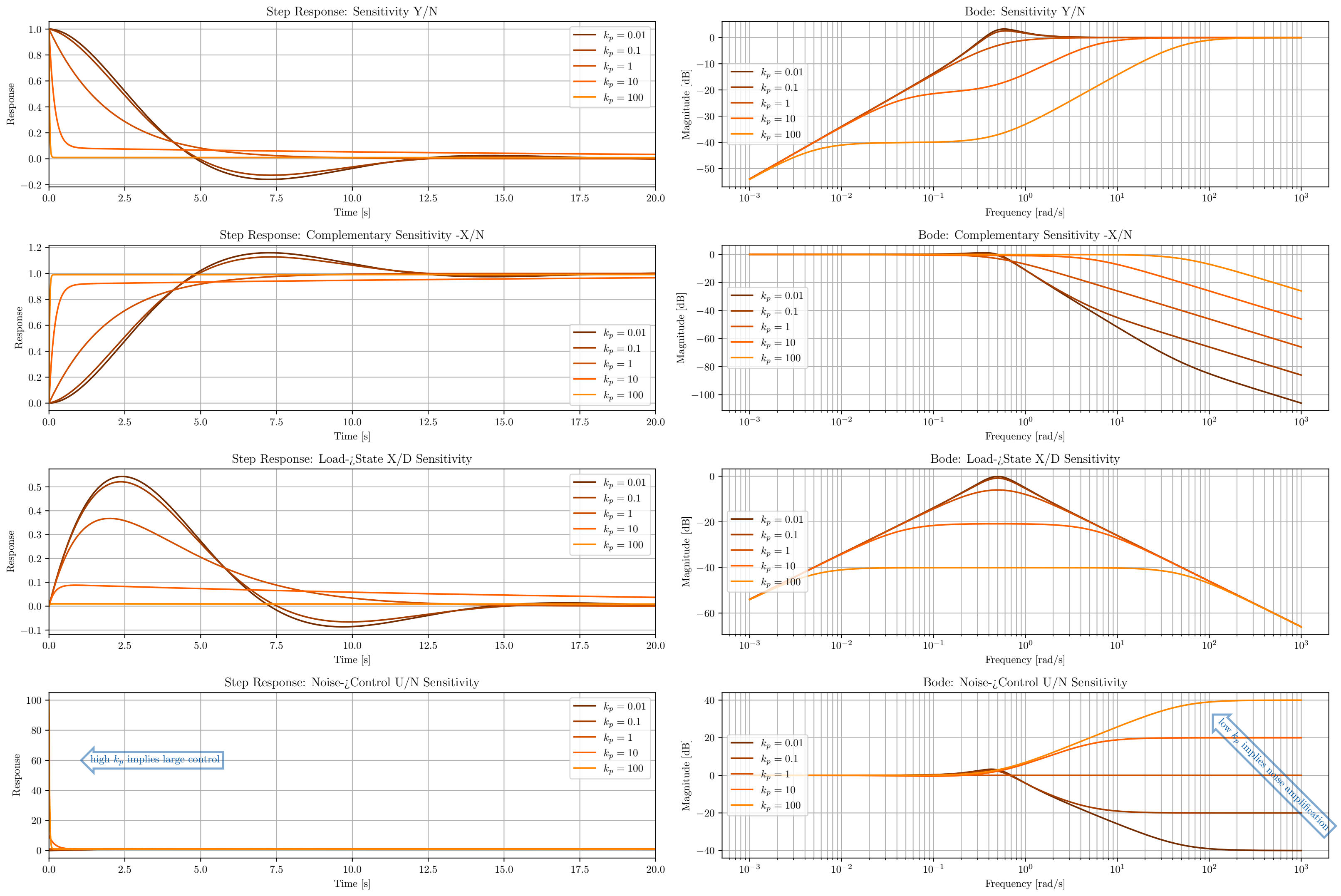


Figure 8.22: A first order system controlled by a PI controller: the step response (left four panels) and the Bode magnitude frequency (right four panels) for each transfer function in the gang of four. For the selected parameter values,
too low proportional gain k_P leads to large oscillations in the top three step responses, and
too high k_P leads to high-frequency noise amplification in the control action.

8.6.3 Control design examples: Right-halfplane zero-pole cancellation (some more lurking danger)

Consider an unstable first-order process:

$$G(s) = \frac{1}{s-1} \tag{8.63}$$

feedback controller

$$C(s) = 1 - \frac{1}{s} = \frac{s-1}{s} \tag{8.64}$$

With such a design, a zero-pole cancellation occurs in the loop function:

$$L(s) = G(s)C(s) = \frac{1}{s} \tag{8.65}$$

Therefore, it appears that we get very nice behavior. Take for example $F = 1$ and calculate

$$\frac{Y(s)}{R(s)} = \frac{G(s)C(s)F(s)}{1+G(s)C(s)} = \frac{1/s}{1+1/s} = \frac{1}{1+s} \tag{8.66}$$

Specifically, since $\left.\frac{1}{1+s}\right|_{s=0} = 1$, a reference signal $r(t) = \mathbf{1}(t)$, causes a steady state response $y_{\text{steady-state}}(t) = \mathbf{1}(t)$.

However, a more careful analysis is well warranted and revealing. Consider the load→state sensitivity:

$$\frac{X(s)}{D(s)} = \frac{G(s)}{1+G(s)C(s)} = \frac{\frac{1}{s-1}}{1+\frac{1}{s}} = \frac{s}{(s+1)(s-1)} \tag{8.67}$$

This means that this transfer function is unstable!

To explain a second potentially-surprising result in the next slide, we compute also the noise→control sensitivity:

$$\frac{C(s)}{1+G(s)C(s)} = \frac{\frac{s-1}{s}}{1+\frac{1}{s}} = \frac{s-1}{s+1} \quad \Longrightarrow \quad \left|\frac{C(\mathrm{i}\omega)}{1+G(\mathrm{i}\omega)C(\mathrm{i}\omega)}\right| = 1. \tag{8.68}$$

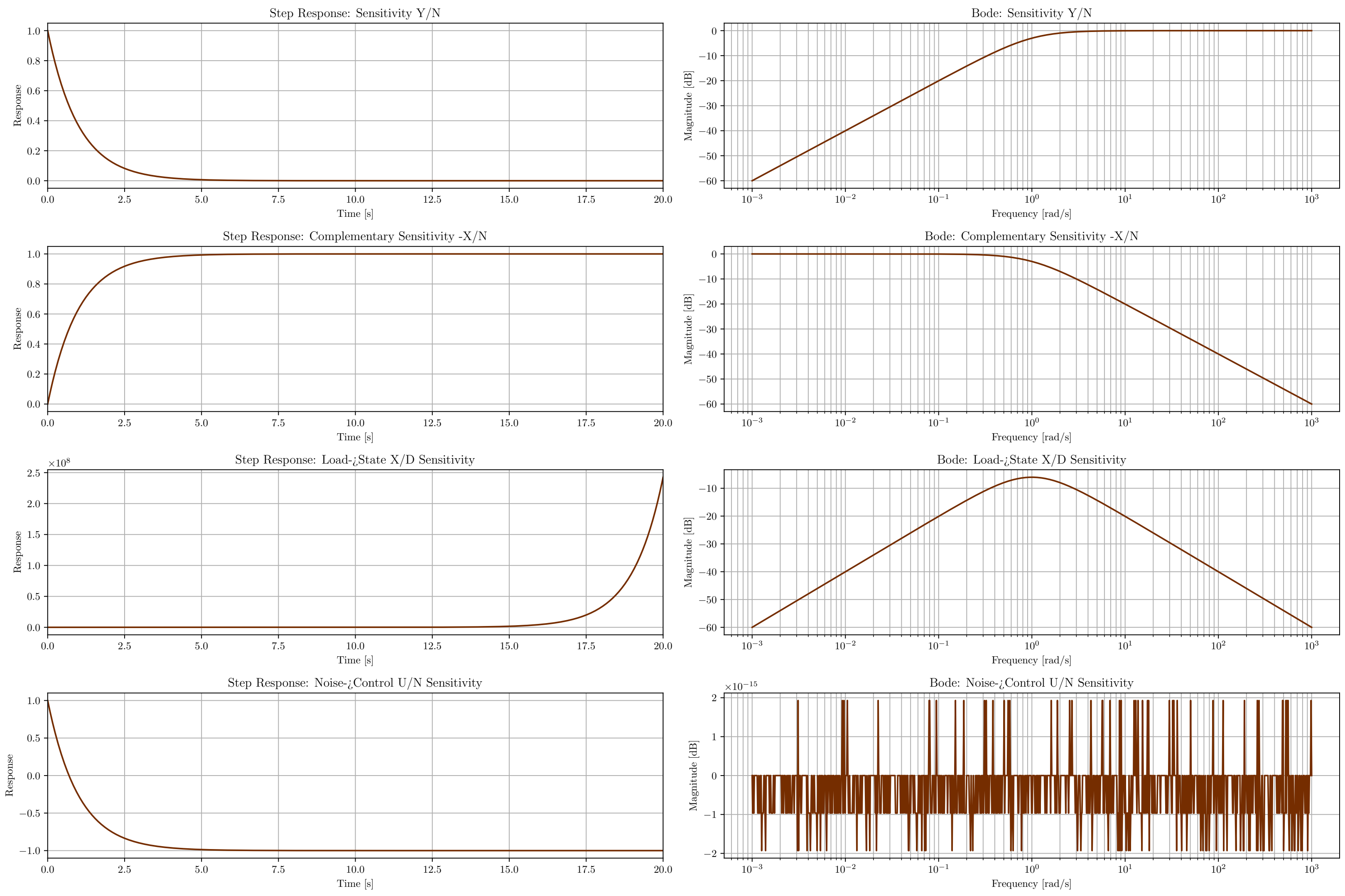


Figure 8.23: An unstable first order system controlled by a PI controller (with negative k_I gain) with parameters achieving an unstable zero-pole cancellation: the Bode magnitude frequency response (right four panels) and the step response (left four panels) for each transfer function in the gang of four.
Note that the state $x(t)$ response to a step input in the load disturbance (plotted in the third panel on the right) is unbounded!

8.7 Exercises

Section 8.1: PID controllers, transfer functions and modifications

E8.1 **PI-D and I-PD architectures.** Consider the PI-D and I-PD architectures in Figure 8.4.

(i) Show that the disturbance-to-output transfer function $Y(s)/D(s)$ is identical for PID, PI-D, and I-PD architectures, i.e., disturbance rejection is unaffected by the architecture;

(ii) derive the reference-to-output transfer functions $Y(s)/R(s)$ for each architecture, and

(iii) compare the step responses for PI-D and I-PD in response to (i) a disturbance step and (ii) a reference step.

Section 8.2: PID control strategies for first and second-order systems

E8.2 **PI control of second-order systems.** Consider a second-order system endowed with a PI controller. Determine to what extent one can execute pole placement and achieve exact reference tracking.

E8.3 **Internal model principle: Tracking ramps and sinusoids.** A basic result in feedback control is that a unity-feedback system needs an integrator (a $1/s$ factor in the open-loop transfer function) in order to track constant inputs (steps) with zero steady-state error. In this exercise we extend that reasoning to ramps and sinusoids, and we arrive at the general *internal model principle*.

Consider a unity-feedback control system with plant $P(s)$, controller $C(s)$, and reference signal $r(t)$. Let $L(s) = C(s)P(s)$ be the open-loop transfer function, and denote the error in the Laplace domain by

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

(i) **Tracking a ramp signal.** Suppose the reference is a ramp $r(t) = \nu t\, \mathbf{1}(t)$, where ν is the slope. Its Laplace transform is $R(s) = \nu/s^2$. Show that the steady-state error is

$$e_{\text{ss}} = \lim_{t\to\infty} e(t) = \lim_{s\to 0} sE(s) = \nu \lim_{s\to 0} \frac{1}{s(1+L(s))}.$$

Conclude that, in order for $e_{\text{ss}} = 0$ for any ν, one must have

$$\lim_{s\to 0} s\, L(s) = \infty,$$

which requires *at least two integrators* in the open loop $L(s)$, i.e., a factor $1/s^2$.

(ii) **Tracking a sinusoidal signal.** Suppose the reference is $r(t) = \sin(\omega_0 t)\, \mathbf{1}(t)$ with Laplace transform

$$R(s) = \frac{\omega_0}{s^2+\omega_0^2}.$$

Show that, in order to have zero steady-state error, the open-loop transfer function must satisfy

$$\lim_{s\to j\omega_0} (s - j\omega_0)\, L(s) = \infty,$$

which means $L(s)$ must have a *pole* at $s = \pm j\omega_0$.

***Note:** Internal Model Principle: From these two cases we can state a general rule: to track a given type of input (or reject a disturbance of that type) without steady-state error, the open-loop transfer function must contain the dynamics that generate that input.*
1. For step inputs, this means a pole at $s = 0$ (an integrator).
2. For ramps, this means a double pole at $s = 0$ (a double integrator).
3. For sinusoids at frequency ω_0, this means poles at $s = \pm j\omega_0$ (an oscillator).
This requirement generalizes to arbitrary signals: if the Laplace transform of the reference has poles at certain locations p_k on the imaginary axis, then the open loop $L(s)$ must include poles at the same locations. This requirement is called the internal model principle.

Section 8.3: The root locus method

E8.4 **Second-order system with increasing proportional feedback.** Consider $G(s) = \dfrac{1}{s^2 + 2s + 2}$ and use a proportional controller with gain k.

(i) Derive the closed-loop characteristic polynomial of the closed-loop transfer function.

(ii) Show that the closed-loop poles are complex conjugates for $k > 0$.

(iii) Express the damping ratio ζ and natural frequency ω_n of the closed-loop system as functions of k.

(iv) Using your results for the pole locations, $\zeta(k)$, and $\omega_\mathrm{n}(k)$, predict what happens to the rise time, settling time, and percentage overshoot as $k \to \infty$. Explain briefly the behavior of the solutions.

(v) Sketch the resulting root locus: Indicate the starting points, the asymptote angles and centroid, the direction of motion as k increases, and whether any segment lies on the real axis.

Section 8.4: Analysis of instability mechanisms

E8.5 **A root locus approach to unmodeled dynamics and destabilizing control design.** Consider the discussion in Section 8.4.4 on unmodeled dynamics and destabilizing control design. Based upon the root locus analysis method, with the help of root locus plotting routine, explain why large gains (due to the selection of a large closed-loop natural frequency ω_{n}) lead to unstable designs.

Bibliography

K. J. Åström and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2 edition, 2021. URL http://www.cds.caltech.edu/~murray/books/AM08/pdf/fbs-public_24Jul2020.pdf.