ECE171A: Linear Control System Theory
Lecture 11: Control Design

Nikolay Atanasov

natanasov@ucsd.edu

UC San Diego

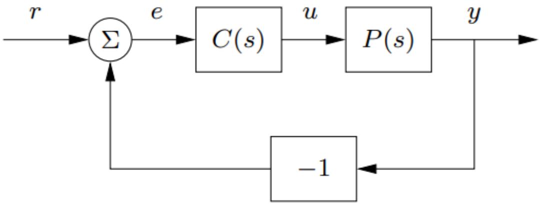
JACOBS SCHOOL OF ENGINEERING

Electrical and Computer Engineering

Outline

Outline

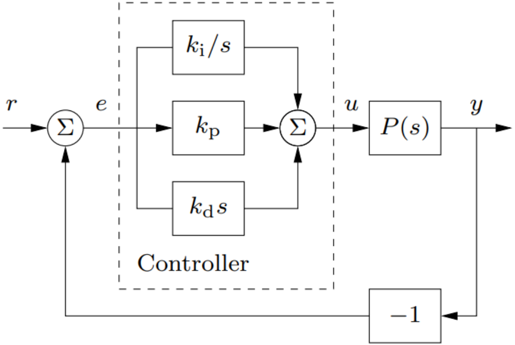
Feedback Control System



Signals	t domain	s domain
Input	$u(t)$	$U(s)$
Output	$y(t)$	$Y(s)$
Reference	$r(t)$	$R(s)$
Error	$e(t) = r(t) - y(t)$	$E(s) = R(s) - Y(s)$

Components	Transfer function
Plant	$P(s) = \frac{Y(s)}{U(s)}$
Controller	$C(s) = \frac{U(s)}{E(s)}$

Proportional Integral Derivative Control



Proportional Integral Derivative (PID) Controller

Uses proportional gain k_p, integral gain k_i, derivative gain k_d:

t domain

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de(t)}{dt}$$

s domain

$$\frac{U(s)}{E(s)} = C(s) = k_p + \frac{k_i}{s} + k_d s$$

PID Control

- PID control is the most common approach for utilizing feedback in engineering systems:
  - Survey of 100+ boiler-turbine controllers: 94.4% PI, 3.7% PID, 1.9% other
- PID control appears in both simple and complex systems: as a stand-alone controller, as an element of hierarchical or distributed systems, etc.
- PID control appears in biological systems, where proportional, integral, and derivative action is generated by subsystems with dynamic behavior
  - Example: Eye pupil opening regulates the amount of light entering the eye

Roles of PID Terms

- PID control terms:
  - **Proportional (P) term**: responds to present error
  - **Integral (I) term**: accumulates past error
  - **Derivative (D) term**: anticipates future error

- PID time constants:

$$u(t) = k_{\mathrm{p}} \left(e(t) + \frac{1}{T_{\mathrm{i}}} \int_0^t e(\tau) d\tau + T_{\mathrm{d}} \frac{de(t)}{dt} \right)$$

  - **Integral time constant**: $T_{\mathrm{i}} = k_{\mathrm{p}} / k_{\mathrm{i}}$
  - **Derivative time constant**: $T_{\mathrm{d}} = k_{\mathrm{d}} / k_{\mathrm{p}}$

Role of P Term

- **Proportional term**: $u(t) = k_\mathrm{p} e(t)$
- Transfer function: $T(s) = \dfrac{Y(s)}{R(s)} = \dfrac{C(s)P(s)}{1 + C(s)P(s)} = \dfrac{k_\mathrm{p} P(s)}{1 + k_\mathrm{p} P(s)}$
- Error: $E(s) = R(s) - Y(s) = (1 - T(s))R(s)$
- Steady-state error of stable system for step reference $R(s) = 1/s$:

$$\lim_{t \to \infty} e(t) = \lim_{s \to 0} sE(s) = \frac{1}{1 + k_\mathrm{p} P(0)}$$

- **Increasing k_p decreases steady-state error but also stability margins**
- **Feedforward term**: used to reduce steady-state error in early controllers:

$$u(t) = k_\mathrm{p} e(t) + u_\mathrm{ff}$$

- For step reference, if the DC gain is known, choose $u_\mathrm{ff} = 1/P(0)$:

$$\lim_{s \to 0} sE(s) = \lim_{s \to \infty} s \left(\frac{1}{1 + k_\mathrm{p} P(s)} R(s) - \frac{P(s)}{1 + k_\mathrm{p} P(s)} \frac{u_\mathrm{ff}}{s} \right) = \frac{1 - u_\mathrm{ff} P(0)}{1 + k_\mathrm{p} P(0)}$$

Role of I Term

- **Integral term**: feedforward term that **guarantees zero steady-state error**:

$$u(t) = k_{\mathrm{p}} e(t) + k_{\mathrm{i}} \int_0^t e(\tau) d\tau \qquad U(s) = \left(k_{\mathrm{p}} + \frac{k_{\mathrm{i}}}{s} \right) E(s)$$

- Transfer function: $T(s) = \dfrac{Y(s)}{R(s)} = \dfrac{C(s)P(s)}{1 + C(s)P(s)}$

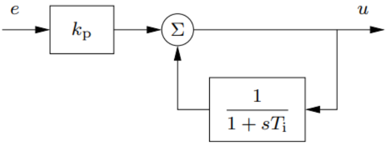
- Steady-state error of stable system for step reference $R(s) = 1/s$:

$$\lim_{s \to 0} sE(s) = \lim_{s \to 0} s \left(1 - T(s)\right) R(s) = \lim_{s \to 0} \frac{1}{1 + C(s)P(s)} \underbrace{=}_{C(s) \to \infty} 0$$

- **Magic of integral action**: if a steady state exists, the error will be zero

- The PI term is implemented using a low-pass filter $H_{\mathrm{pi}}(s) = \frac{1}{1+sT_{\mathrm{i}}}$:

$$\frac{U(s)}{E(s)} = k_{\mathrm{p}} \frac{1 + sT_{\mathrm{i}}}{sT_{\mathrm{i}}} = k_{\mathrm{p}} + \frac{k_{\mathrm{p}}}{sT_{\mathrm{i}}}$$



(a) Integral action (automatic reset)

Role of D Term

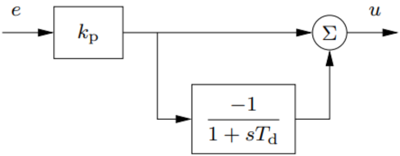
- **Derivative term**: provides predictive action:

$$u(t) = k_\mathrm{p} e(t) + k_\mathrm{d} \frac{de(t)}{dt} = k_\mathrm{p} \left(e(t) + T_\mathrm{d} \frac{de(t)}{dt} \right) =: k_\mathrm{p} e_\mathrm{p}(t)$$

- **Prediction error** e_p: linear extrapolation of the error to time $t + T_\mathrm{d}$
- In practice the error signal $e(t)$ is measured and contains high-frequency noise which should not be differentiated
- The D term is implemented using a low-pass filter $H_\mathrm{d}(s) = \frac{1}{1+sT_\mathrm{d}}$
- **Filtered derivative**: difference between a signal and its low-pass filtered version:

$$\frac{U_\mathrm{d}(s)}{E(s)} = k_\mathrm{p} \left(1 - \frac{1}{1 + sT_\mathrm{d}} \right) = \frac{k_\mathrm{d} s}{1 + sT_\mathrm{d}}$$

- Acts as **differentiator** for low-frequency signals and as **constant gain** k_p for high-frequency signals



(b) Derivative action

Numerical Experiments

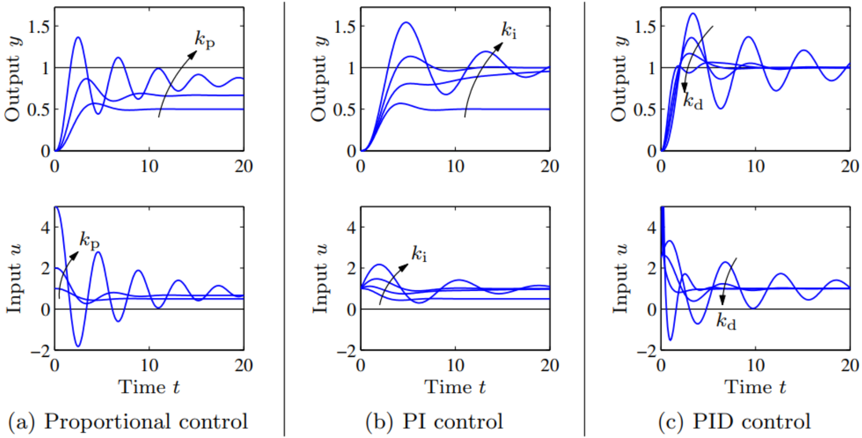
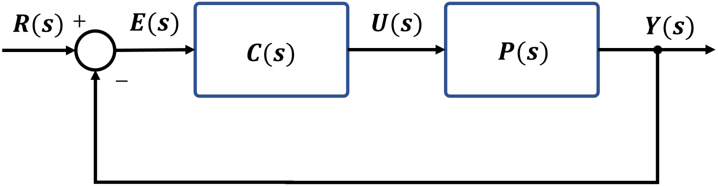


Figure 11.2: Responses to step changes in the reference value for a system with a proportional controller (a), PI controller (b), and PID controller (c). The process has the transfer function $P(s) = 1/(s+1)^3$, the proportional controller has parameters $k_p = 1$, 2, and 5, the PI controller has parameters $k_p = 1$, $k_i = 0$, 0.2, 0.5, and 1, and the PID controller has parameters $k_p = 2.5$, $k_i = 1.5$, and $k_d = 0$, 1, 2, and 4.

Model Reduction

- Practical systems are complex
- While a high-order model may describe the system behavior accurately, a low-order model may simplify the system analysis and control design
- **Model reduction**: simplification of a system model that captures the essential properties needed for control design
- Various model reduction techniques are available:
  - **Dominant pole-zero approximation**: cancel pole-zero pairs or eliminate states that have little effect on the model response
  - **Mode selection**: eliminate poles and zeros that fall outside a specific frequency range of interest
- Low-order models can be obtained from first principles:
  - A system can be modeled as zeroth-order if its inputs are sufficiently slow
  - A system can be modeled as first-order if the change of its mass, momentum, or energy can be captured by a single variable (e.g., velocity)
  - A system can be modeled as second-order if the change of its mass, momentum, or energy can be captured by two variables (e.g., position and velocity)

Second-Order System Control Design



- Consider a feedback control system with a second-order plant:

$$P(s) = \frac{b_0}{s^2 + a_1 s + a_0}$$

- How should the controller $C(s)$ be designed to ensure that the closed-loop system is **stable** and its **step response has zero steady-state error**?

P Control for Second-Order System

- **P controller**:

$$u(t) = k_\mathrm{p} e(t) \qquad \Leftrightarrow \qquad \frac{U(s)}{E(s)} = C(s) = k_\mathrm{p}$$

- Closed-loop transfer function:

$$T(s) = \frac{Y(s)}{R(s)} = \frac{C(s)P(s)}{1 + C(s)P(s)} = \frac{k_\mathrm{p} b_0}{s^2 + a_1 s + (a_0 + k_\mathrm{p} b_0)}$$

- P control can accelerate the response of a second-order system by changing the natural frequency $\omega_n^2 = (a_0 + k_\mathrm{p} b_0)$
- To ensure stability, we need $a_1 > 0$ and $a_0 + K_p b_0 > 0$
- P control can stabilize only some systems because it adjusts one coefficient of the characteristic equation

For $a_0 \neq 0$, $C(s)P(s)$ has 0 poles at the origin (type 0 system) and the closed-loop step response has a **constant finite steady-state error**:

$$\lim_{t \to \infty} e(t) = \lim_{s \to 0} (1 - T(s)) = \frac{a_0}{a_0 + k_\mathrm{p} b_0}.$$

PI Control for Second-Order System

- To achieve zero steady-state step error, we need to add a pole at the origin in $C(s)P(s)$ to obtain a type 1 system
- **PI controller**:

$$u(t) = k_{\mathrm{p}} e(t) + k_{\mathrm{i}} \int_0^t e(\tau) d\tau \qquad \Leftrightarrow \qquad \frac{U(s)}{E(s)} = C(s) = k_{\mathrm{p}} + \frac{k_{\mathrm{i}}}{s}$$

- Closed-loop transfer function:

$$T(s) = \frac{Y(s)}{R(s)} = \frac{C(s)P(s)}{1 + C(s)P(s)} = \frac{b_0(k_{\mathrm{p}} s + k_{\mathrm{i}})}{s^3 + a_1 s^2 + (a_0 + k_{\mathrm{p}} b_0)s + k_{\mathrm{i}} b_0}$$

PI control achieves **zero steady-state error**:

$$\lim_{t \to \infty} e(t) = \lim_{s \to 0} (1 - T(s)) = 1 - T(0) = 0$$

but the closed-loop system may be unstable if $a_1 < 0$.

PID Control for Second-Order System

- **PID controller**:

$$u(t) = k_\mathrm{p} e(t) + k_\mathrm{i} \int_0^t e(\tau) d\tau + k_\mathrm{d} \frac{de(t)}{dt} \qquad \Leftrightarrow \qquad C(s) = k_\mathrm{p} + \frac{k_\mathrm{i}}{s} + k_\mathrm{d} s$$

- Closed-loop transfer function:

$$T(s) = \frac{Y(s)}{R(s)} = \frac{C(s)P(s)}{1 + C(s)P(s)} = \frac{b_0(k_\mathrm{p} s + k_\mathrm{i} + k_\mathrm{d} s^2)}{s^3 + (a_1 + k_\mathrm{d} b_0)s^2 + (a_0 + k_\mathrm{p} b_0)s + k_\mathrm{i} b_0}$$

- The coefficients of the characteristic polynomial can be set **arbitrarily** via an appropriate choice of k_p, k_i, k_d

For a second-order plant, PID control can guarantee **stability**, **good transient behavior**, and **zero steady-state step error**.

PID Control Example

- Consider the plant $P(s) = \frac{1}{s^2-3s-1}$
- Design a PID controller $C(s)$ to achieve step response with zero steady-state error and place the closed-loop system poles at -5, -6, -7
- PID controller: $C(s) = k_\mathrm{p} + \frac{k_\mathrm{i}}{s} + k_\mathrm{d} s$
- Closed-loop transfer function:

$$T(s) = \frac{Y(s)}{R(s)} = \frac{C(s)P(s)}{1 + C(s)P(s)} = \frac{k_\mathrm{d} s^2 + k_\mathrm{p} s + k_\mathrm{i}}{s^3 + (k_\mathrm{d} - 3)s^2 + (k_\mathrm{p} - 1)s + k_\mathrm{i}}$$

- Match coefficients with:

$$\Delta(s) = (s+5)(s+6)(s+7) = s^3 + 18s^2 + 107s + 210$$

- PID control gains:

$$k_\mathrm{d} = 21 \qquad k_\mathrm{p} = 108 \qquad k_\mathrm{i} = 210$$

Outline

PID Control

PID Tuning and Implementation

Inverted Pendulum Example

Lead-Lag Compensation

PID Control Gain Tuning

- **PID control gain tuning**: the process of determining satisfactory PID control gains
  - Manual tuning
  - Ziegler-Nichols method
  - First-order and time-delay (FOTD) method
  - Automatic tuning via relay feedback

Manual PID Control Gain Tuning

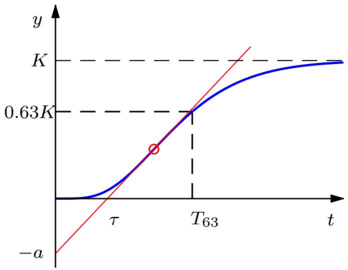
- Set $k_{\mathrm{i}} = k_{\mathrm{d}} = 0$
- Increase k_{p} slowly until the output of the closed-loop system oscillates on the verge of instability
- Reduce k_{p} to achieve **quarter amplitude decay** of the closed-loop response, i.e., the amplitude should be one-fourth of the maximum value during the oscillatory period
- Increase k_{i} and k_{d} to achieve the desired response

Table 7.4 Effect of Increasing the PID Gains K_p, K_D, and K_I on the Step Response

PID Gain	Percent Overshoot	Settling Time	Steady-State Error
Increasing K_P	Increases	Minimal impact	Decreases
Increasing K_I	Increases	Increases	Zero steady-state error
Increasing K_D	Decreases	Decreases	No impact

Ziegler-Nichols Method

- Developed by John Ziegler and Nathaniel Nichols in the 1940s
- Perform a simple experiment on the system to extract features from its time domain or frequency domain response
- **Time-domain method**
  - Apply a unit step input to the **open-loop** system
  - Record the x-intercept τ and y-intercept $-a$ with the coordinate axes of the steepest tangent to the step response
  - Use τ and a to choose the PID control gains



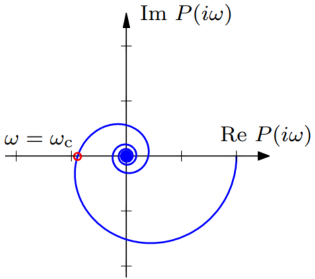
(a) Step response method

Type	k_p	T_i	T_d
P	$1/a$		
PI	$0.9/a$	$\tau/0.3$	
PID	$1.2/a$	$\tau/0.5$	0.5τ

(a) Step response method

Ziegler-Nichols Method

- **Frequency-domain method**
  - Connect a PID controller to the plant with $k_{\mathrm{i}} = k_{\mathrm{d}} = 0$
  - Increase k_{p} until the closed-loop response oscillates on the verge of instability
  - Record the critical proportional gain k_{c} and the period of oscillation T_c
  - Nyquist contour of $k_{\mathrm{c}}P(s)$ passes through -1 at frequency $\omega_c = 2\pi/T_c$
  - Use k_{c} and T_c to choose the PID control gains



(b) Frequency response method

Type	k_p	T_i	T_d
P	$0.5k_c$		
PI	$0.45k_c$	$T_c/1.2$	
PID	$0.6k_c$	$T_c/2$	$T_c/8$

(b) Frequency response method

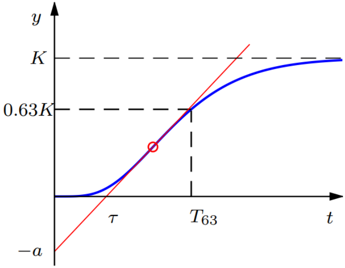
FOTD method

- Ziegler–Nichols methods use 2 parameters to determine the PID control gains
- **First-order and time-delay (FOTD) method**: uses plant model with more parameters:

$$P(s) = \frac{K}{1 + sT} e^{-\tau s}$$

- Apply unit-step input to **open-loop** system
- Record **time delay** τ (x-intercept of steepest tangent), **steady-state value** K, and $T = T_{63} - \tau$, where T_{63} is the time when the output reaches 63% of K
- Use τ, K, and T to choose the PI gains:

$$k_{\mathrm{p}} = \frac{0.15\tau + 0.35T}{K\tau} \qquad k_{\mathrm{i}} = \frac{0.46\tau + 0.02T}{K\tau^2}$$



(a) Step response method

Integral Windup

- **Integral windup**: accumulation of integral error due to input saturation
- Physical actuators have limits, e.g., a motor has maximum speed, a valve cannot be more than fully opened
- When actuator limits are reached, the input remains at its limit (**input saturation**) and the system runs in open-loop
- The integral error $\int_0^t e(\tau)d\tau$ accumulates while the input is saturated
- Once the input leaves the saturation range the accumulated integral error induces **large transient response**

Example: Cruise control

- When a car encounters a steep hill (e.g., 6°), the throttle saturates
- The resulting integral windup leads to velocity overshoot

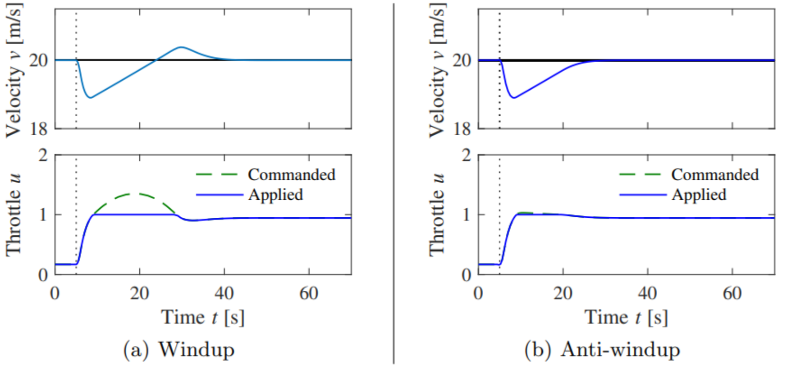


Figure 11.10: Simulation of PI cruise control with windup (a) and anti-windup (b). The figure shows the speed v and the throttle u for a car that encounters a slope that is so steep that the throttle saturates. The controller output is a dashed line. The controller parameters are $k_\mathrm{p} = 0.5$, $k_\mathrm{i} = 0.1$ and $k_\mathrm{aw} = 2.0$. The anti-windup compensator eliminates the overshoot by preventing the error from building up in the integral term of the controller.

Avoiding Integral Windup

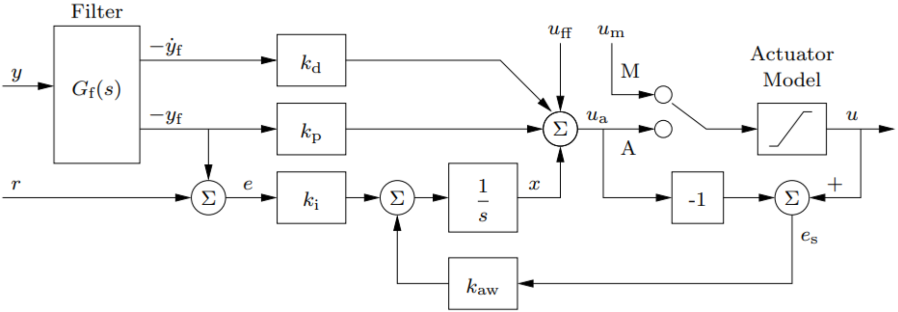


Figure: Anti-windup PID controller with output filtering, feedforward input u_{ff}, and input saturation error e_s

- The controller has an extra feedback path from the saturating actuator to measure saturation error $e_s = u - u_a$
- When the actuator saturates, the saturation error e_s if fed back to the integrator to reduce the integral error

Avoiding Derivative Noise

- Derivative control requires differentiation of the error signal:

$$\dot{e}(t) \approx \frac{e(t) - e(t - \tau)}{\tau}$$

- In practice, the error signal is measured and contains high-frequency noise, which should not be differentiated
- The derivative term $k_\mathrm{d} s$ is implemented using a low-pass filter $H_\mathrm{d}(s) = \frac{1}{\tau_f s + 1}$ with a small filter time constant τ_f
- PID control with high-frequency noise attenuation:

$$u(t) = k_\mathrm{p} e(t) + k_\mathrm{i} \int_0^t e(\tau) d\tau + k_\mathrm{d} \dot{e}_f(t) \qquad C(s) = k_\mathrm{p} + \frac{k_\mathrm{i}}{s} + \frac{k_\mathrm{d} s}{\tau_f s + 1}$$

$$\tau_f \dot{e}_f(t) = -e_f(t) + e(t)$$

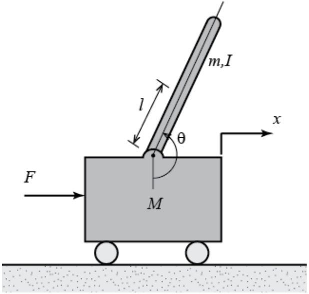
Discrete-time PID Control Implementation

- sampling interval: τ_s
- filter time constant: τ_f
- sampled error: $e[k] = e(k\tau_s)$
- filtered error: $e_f[k] = \frac{\tau_s}{\tau_f} e[k] + \left(1 - \frac{\tau_s}{\tau_f}\right) e_f[k-1]$
- derivative error: $e_d[k] = \frac{e_f[k] - e_f[k-1]}{\tau_s}$
- integral error: $e_i[k] = e_i[k-1] + \tau_s e[k-1]$
- control: $u[k] = k_\mathrm{p} e[k] + k_\mathrm{i} e_i[k] + k_\mathrm{d} e_d[k]$

Outline

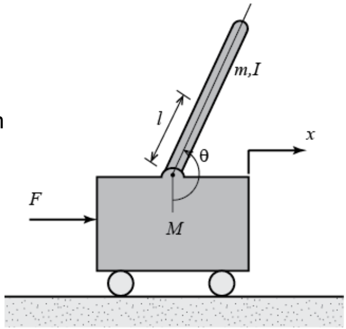
Inverted Pendulum Example

- Consider an inverted pendulum mounted on a motorized cart
- **Objective**: control the cart force to balance the inverted pendulum in an upright position
- Popular example in control theory and reinforcement learning
- Nonlinear system that is unstable without control



Inverted Pendulum: Parameters

- Cart mass: $M = 0.5$ kg
- Pendulum mass: $m = 0.2$ kg
- Cart friction coefficient: $b = 0.1$ N/m/sec
- Length to pendulum center of mass: $\ell = 0.3$ m
- Pendulum moment of inertia:
  $I = 0.006$ kg m^2
- Cart input force: F
- Cart position: x
- Pendulum angle: θ



Inverted Pendulum: System Model

- Horizontal direction force balance for the cart:

$$M\ddot{x} + b\dot{x} + N = F$$

- Horizontal direction force balance for the pendulum:

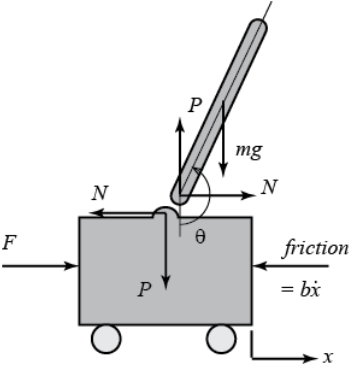
$$N = m\ddot{x} + m\ell\ddot{\theta}\cos\theta - m\ell\dot{\theta}^2\sin\theta$$

- Force balance perpendicular to the pendulum:

$$P\sin\theta + N\cos\theta - mg\sin\theta = m\ell\ddot{\theta} + m\ddot{x}\cos\theta$$

- Torque balance about the pendulum centroid:

$$-P\ell\sin\theta - N\ell\cos\theta = I\ddot{\theta}$$



Inverted Pendulum: System Model

- Eliminating reaction force N and normal force P and denoting the input force F by u, we get the cart-pole equations of motion:

$$(M+m)\ddot{x} + b\dot{x} + m\ell\ddot{\theta}\cos\theta - m\ell\dot{\theta}^2\sin\theta = u$$
$$(I+m\ell^2)\ddot{\theta} + mg\ell\sin\theta = -m\ell\ddot{x}\cos\theta$$

- Since our control techniques apply to linear time-invariant systems only, we need to linearize the equations of motion

- Linearize about the upright pendulum position $\theta_{\mathrm{e}} = \pi$ and assume that the pendulum remains within a small neighborhood: $\phi = \theta - \pi$

- Small angle approximation:

$$\cos\theta = \cos(\pi+\phi) \approx -1 \qquad \sin\theta = \sin(\pi+\phi) \approx -\phi \qquad \dot{\theta}^2 = \dot{\phi}^2 \approx 0$$

- Linearized equations of motion:

$$(M+m)\ddot{x} + b\dot{x} - m\ell\ddot{\phi} = u$$
$$(I+m\ell^2)\ddot{\phi} - mg\ell\phi = m\ell\ddot{x}$$

Inverted Pendulum: Transfer Function

- Laplace transform of the equations of motion with zero initial conditions:

$$(M+m)s^2X(s)+bsX(s)-m\ell s^2\Phi(s)=U(s)$$
$$(I+m\ell^2)s^2\Phi(s)-mg\ell\Phi(s)=m\ell s^2X(s)$$

- Eliminating $X(s)$ leads to:

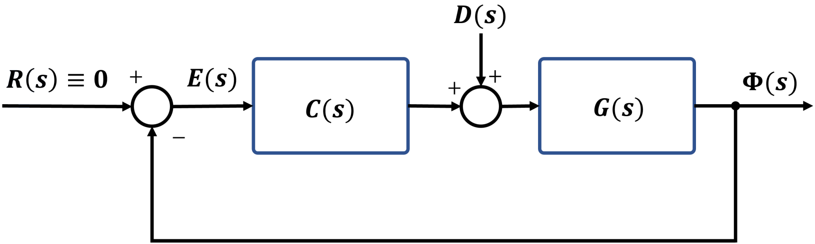
$$(M+m)\left(\frac{I+m\ell^2}{m\ell}-\frac{g}{s^2}\right)s^2\Phi(s)+b\left(\frac{I+m\ell^2}{m\ell}-\frac{g}{s^2}\right)s\Phi(s)-m\ell s^2\Phi(s)=U(s)$$

- Pendulum transfer function with $q=(M+m)(I+m\ell^2)-(m\ell)^2$:

$$G(s)=\frac{\Phi(s)}{U(s)}=\frac{m\ell s^2}{qs^4+b(I+m\ell^2)s^3-(M+m)mg\ell s^2-bmgls}$$

Inverted Pendulum: PID Control

- Design a controller $C(s)$ to maintain the pendulum vertically upward when the cart input F is subjected to a 1-Nsec impulse disturbance $D(s)$
- Design specifications:
  - Settling time of less than 5 seconds
  - Maximum pendulum deviation from the vertical position of 0.05 rad



Inverted Pendulum: PID Control

- Pendulum transfer function with $q = (M+m)(I+m\ell^2) - (m\ell)^2$:

$$G(s) = \frac{\Phi(s)}{U(s)} = \frac{m\ell s^2}{qs^4 + b(I+m\ell^2)s^3 - (M+m)mg\ell s^2 - bmgls}$$

```
M = 0.5; m = 0.2; b = 0.1; I = 0.006;
g = 9.8; l = 0.3; q = (M+m)*(I+m*l^2)-(m*l)^2;
s = tf('s');
G = (m*l*s^2)/(q*s^4 + b*(I + m*l^2)*s^3 -(M + m)*m*g*l*s^2 -b*m*g*l*s);
```

- PID control design: $C(s) = k_{\mathrm{p}} + k_{\mathrm{i}}\frac{1}{s} + k_{\mathrm{d}}s$

```
Kp = 100; Ki = 1; Kd = 1;
C = pid(Kp,Ki,Kd);
```

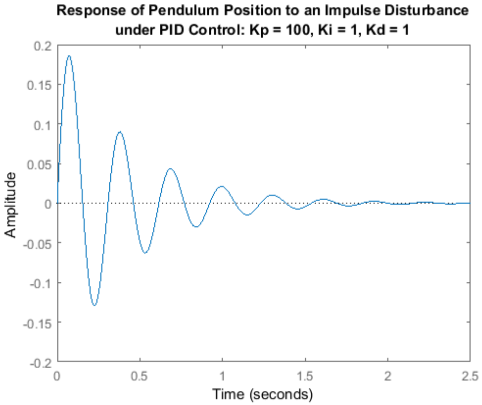
- Closed-loop transfer function from $D(s)$ to $\Phi(s)$:

$$T(s) = \frac{\Phi(s)}{D(s)} = \frac{G(s)}{1 + C(s)G(s)}$$

```
T = feedback(G,C);
```

Inverted Pendulum: PID Control

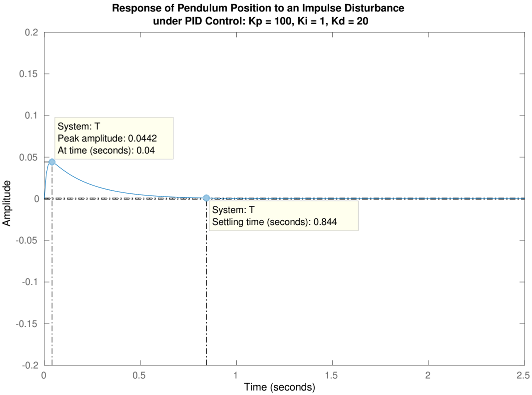
```
t=0:0.01:10;
impulse(T,t)
axis([0, 2.5, -0.2, 0.2]);
title({'Response of Pendulum Position to an Impulse Disturbance';'under PID Control: Kp = 100, Ki = 1, Kd = 1'});
```



- **Settling time**: 1.64 sec meets the specifications (no additional integral control is needed)
- **Peak response**: 0.2 rad exceeds the requirement of 0.05 rad (the overshoot can be reduced by increasing the derivative control gain)

Inverted Pendulum: PID Control

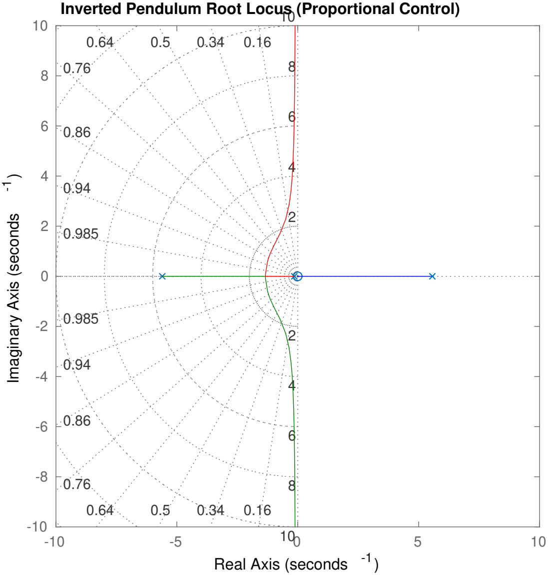
```
t=0:0.01:10;
impulse(T,t)
axis([0, 2.5, -0.2, 0.2]);
title({'Response of Pendulum Position to an Impulse Disturbance';'under PID
    Control: Kp = 100, Ki = 1, Kd = 20'});
```



- **Settling time**: 0.844 sec meets the specifications
- **Peak response**: 0.044 rad meets the specifications

Inverted Pendulum: Root Locus with Proportional Control

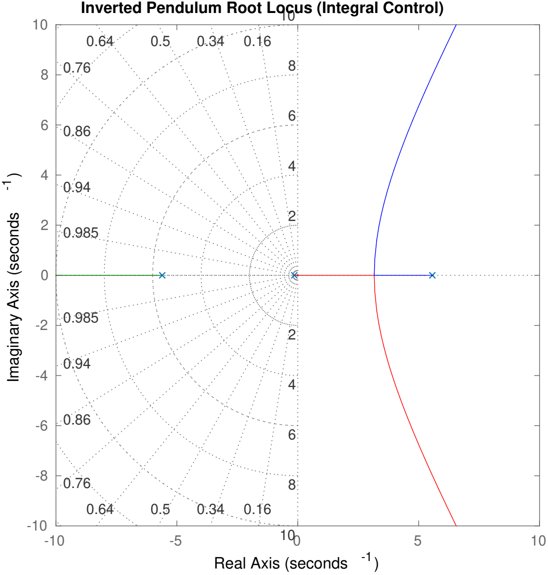
- Positive root locus for the inverted pendulum plant $G(s)$



- One branch entirely in the right half-plane
- Need to add a pole at the origin to cancel the plant zero at the origin
- This will produce two closed-loop poles in the right half-plane that we can then draw to the left-half plane to stabilize the closed-loop system

Inverted Pendulum: Root Locus with Integral Control

- Positive root locus for integral control of the inverted pendulum $\frac{1}{s}G(s)$



- We need to draw the two branches to the left-half plane to stabilize the closed-loop system
- Adding a zeros to the controller will pull the branches to the left

Inverted Pendulum: Root Locus Manipulation

- Poles and zeros of $\frac{1}{s}G(s) = \frac{m\ell s^2}{qs^5 + b(I+m\ell^2)s^4 - (M+m)mg\ell s^3 - bmgls^2}$:

$$z_1 = z_2 = 0$$

$$p_1 = p_2 = 0, \quad p_3 = -0.143, \quad p_4 = -5.604 \quad p_5 = 5.565$$

- Suppose we introduce a zero to the controller: $\frac{(s-z_3)}{s}G(s)$
- There will be $5 - 3 = 2$ asymptotes with angles $\frac{\pi}{2}$, $\frac{3\pi}{2}$ and centroid:

$$\alpha = \frac{1}{2}\left(-5.604 + 5.565 - 0.143 - z_3\right) = -\frac{0.182 + z_3}{2}$$

- We cannot have z_3 in the right half-plane so the best we can do to pull the root locus branches is to have $z_3 \approx 0$ so that $\alpha \approx -0.1$.
- The real parts of the two poles $-\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}$ will approach $\alpha \approx -0.1$ as $K \to \infty$
- This design is insufficient to meet the settling time specification:

$$t_s \approx \frac{4}{\zeta\omega_n} \approx \frac{4}{0.1} = 40\ s$$

Inverted Pendulum: Root Locus Manipulation

- Adding a single zero to the controller is not sufficient to pull the root locus branches far enough to the left
- Add two zeros between $p_3 = -0.143$ and $p_4 = -5.604$ to pull the root locus branches towards them, leaving a single asymptote at $-\pi$
- Let $z_3 = -3$ and $z_4 = -4$ and consider the controller:

$$C(s) = \frac{(s+3)(s+4)}{s} = 7 + 12\frac{1}{s} + s$$

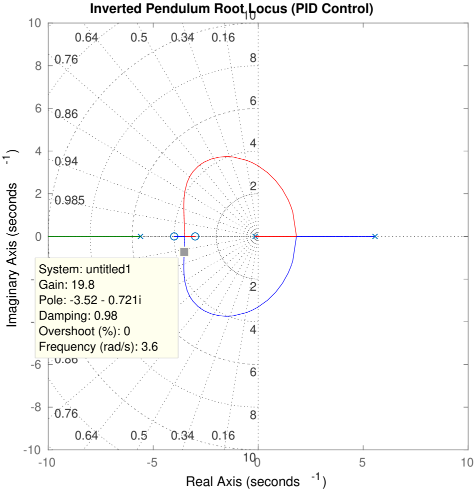
- Note that $kC(s)$ is a PID controller:

$$k_{\mathrm{p}} = 7k \qquad k_{\mathrm{i}} = 12k \qquad k_{\mathrm{d}} = k$$

Inverted Pendulum: Root Locus with PID Control

- Positive root locus for PID control of the inverted pendulum:

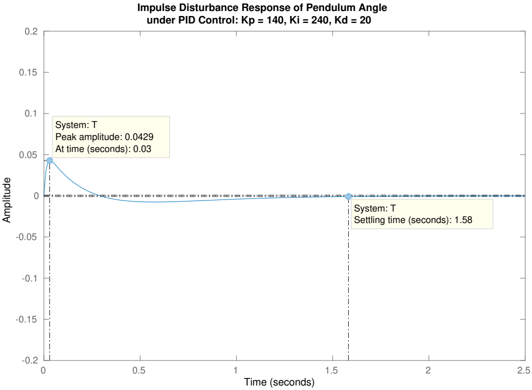
$$\frac{(s+3)(s+4)}{s}G(s)$$



- To achieve $t_s \leq 5$ sec, we need the real parts of the dominant closed-loop poles to be less than $-4/5 = -0.8$
- To ensure that p.o. $\leq 5\%$, we also need sufficient damping for the dominant closed-loop poles
- Placing the dominant poles near the real axis increases the damping ratio ζ
- Choose $k \approx 20$

Inverted Pendulum: PID Control

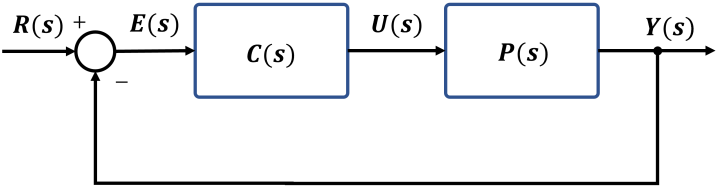
```
T = feedback(G,20*(s+3)*(s+4)/s);
t=0:0.01:10;
impulse(T,t);
title({'Impulse Disturbance Response of Pendulum Angle'; 'under PID Control: Kp
    = 140, Ki = 240, Kd = 20'});
```



- **Settling time**: 1.580 sec meets the specifications
- **Peak response**: 0.043 rad meets the specifications

Outline

Loop Shaping

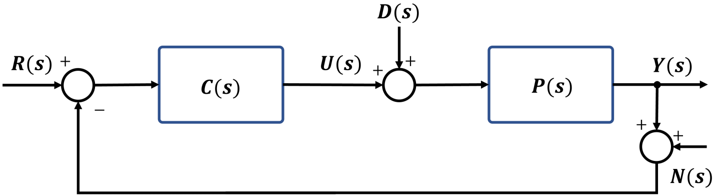


- **Loop shaping**: a trial and error procedure to choose a controller $C(s)$ that gives a loop transfer function $L(s) = C(s)P(s)$ with a desired shape

- **Backward method**:
  - Determine a desired loop transfer function $L(s)$
  - Compute the controller as $C(s) = L(s)/P(s)$

- **Forward method**:
  - Adjust proportional gain $C(s) = k_{\mathrm{p}}$ to obtain desired closed-loop bandwidth
  - Add stable poles and zeros to $C(s)$ until a desired shape of $L(s)$ is obtained

Design Considerations



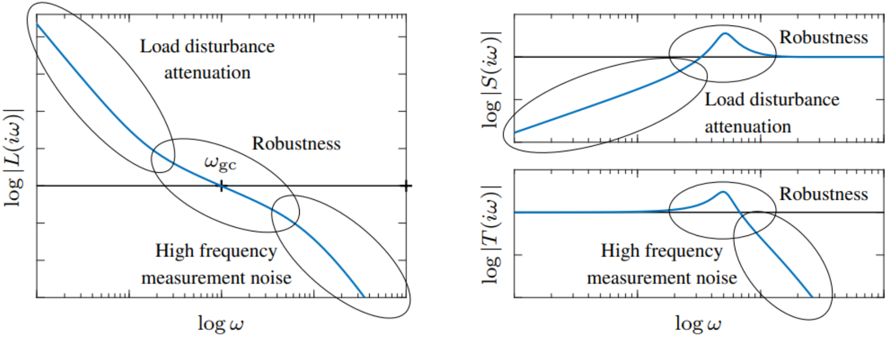
- Tracking error with input disturbance and measurement noise:

$$E(s) = \underbrace{\frac{1}{1+L(s)}}_{\text{Sensitivity } S(s)} R(s) - \frac{P(s)}{1+L(s)} D(s) + \underbrace{\frac{L(s)}{1+L(s)}}_{\text{Complementary Sensitivity } T(s)} N(s)$$

- We need a loop transfer function $L(s) = C(s)P(s)$ that leads to good **closed-loop performance** and good **stability margins**
  - $|L(s)|$ should be large at low frequencies $s = j\omega$ to ensure good reference tracking and low sensitivity to input disturbances (associated with low ω)
  - $|L(s)|$ should be small at high frequencies $s = j\omega$ to ensure low sensitivity to measurement noise (associated with high ω)

Design Considerations

- An ideal loop transfer function $L(j\omega)$ should have the shape below:
  - Unit gain at gain crossover: $|L(j\omega_g)| = 1$
  - Large gain at $\omega < \omega_g$
  - Small gain at $\omega > \omega_g$



(a) Gain plot of loop transfer function (b) Gain plot of sensitivity functions

- The **phase margin is inversely proportional to the slope of** $L(j\omega)$ **around gain crossover frequency** ω_g (transition from high gain at low ω to low gain at high ω cannot be too fast)

Loop Shaping via Lead and Lag Compensation

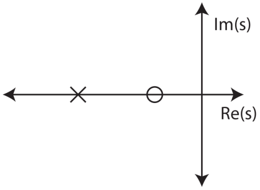
- Loop shaping is a trial-and-error procedure
- Start with a Bode plot of the plant transfer function $P(s)$
- Adjust the **proportional gain** to choose the gain crossover frequency ω_g (compromise between disturbance attenuation and measurement noise)
- Add left-half-plane poles and zeros to $C(s)$ to shape $L(s)$
- The behavior around ω_g can be changed by **lead compensation**
- The loop gain at low frequencies can be increased by **lag compensation**

Lead and Lag Compensation

- Consider a controller with transfer function:

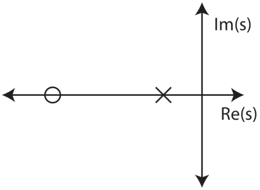
$$C(s) = k\frac{s+z}{s+p} \qquad z > 0,\ p > 0$$

- **Lead compensator**: $z < p$
  - Adds **phase lead** in the frequency range $\omega \in [z, p]$
  - Provides **additional phase margin** at ω_g
  - Equivalent to PD control with filtering
  - Root locus branches move left



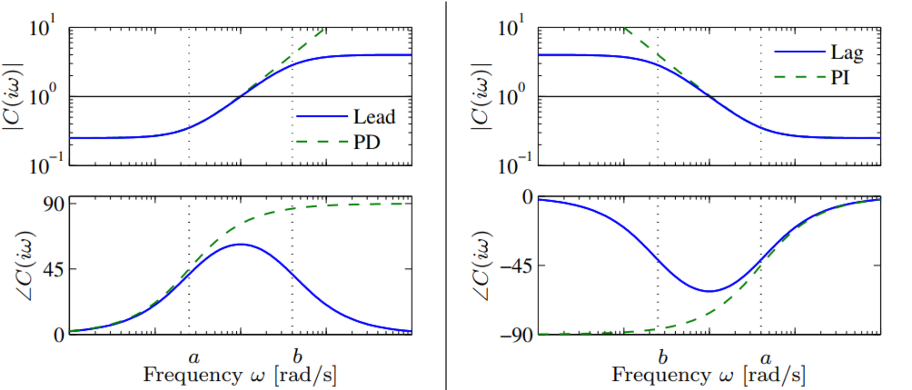
(when we want a zero)

- **Lag compensator**: $z > p$
  - **Increases the gain at low frequencies** leading to improved tracking and disturbance attenuation
  - PI control is a special case with $p = 0$
  - Root locus branches move right



(when we want a pole)

Lead and Lag Compensation

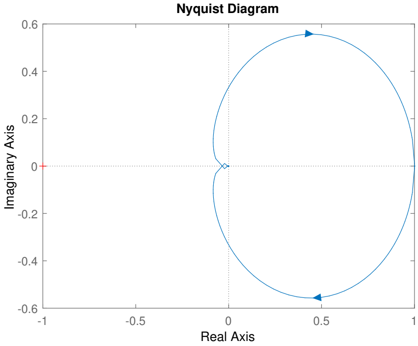


(a) Lead compensation, $a < b$

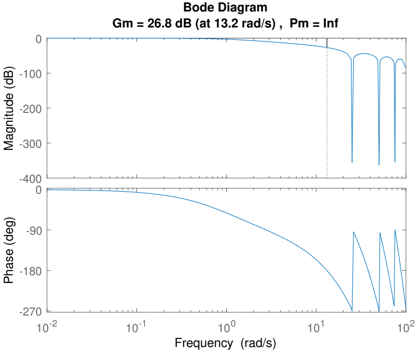
(b) Lag compensation, $b < a$

Example 1

- Plant: $P(s) = \dfrac{4(1 - e^{-s/4})}{s(s+1)}$



(a) Nyquist plot



(b) Bode plot

Example 1: Tracking Performance

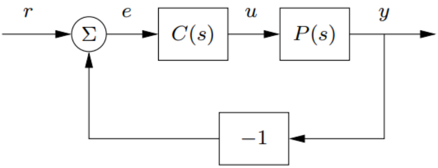
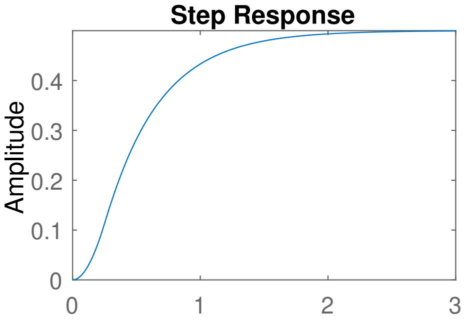
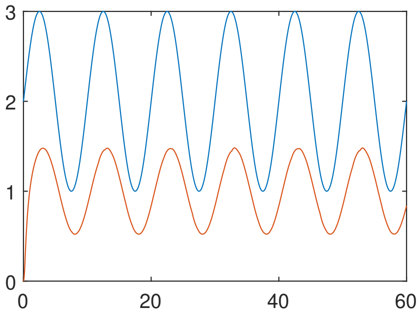


Figure: Proportional control: $C(s) = 1$



(a) Step response



(b) Frequency response at $\omega = \pi/5$

Example 1: Lag Compensation

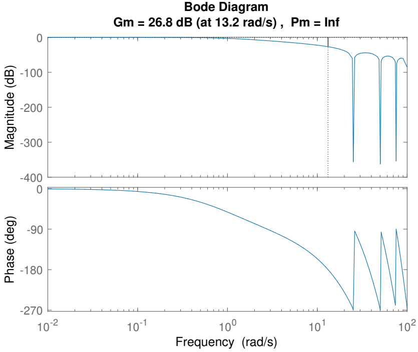


Figure: $P(s)$

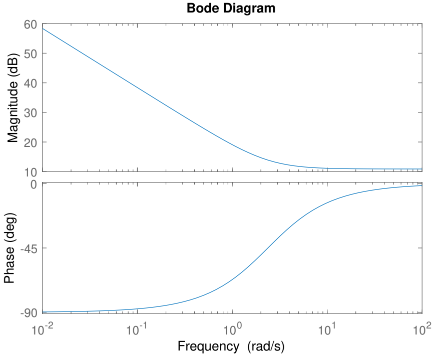


Figure: Lag compensator $C(s) = 3.5 + \frac{8.3}{s}$ (PI)

Example 1: Lag Compensation

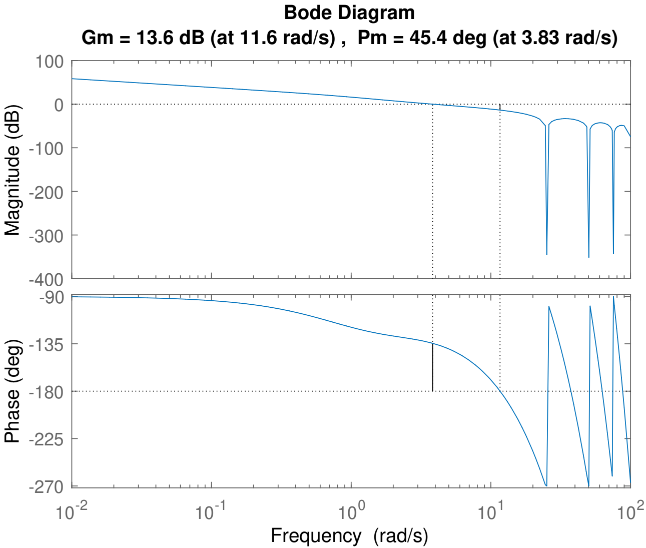


Figure: Margins for $L(s) = C(s)P(s)$

Example 1: Lag Compensation

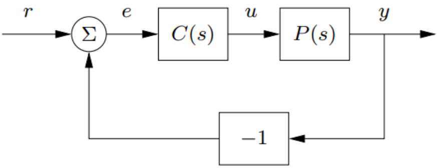
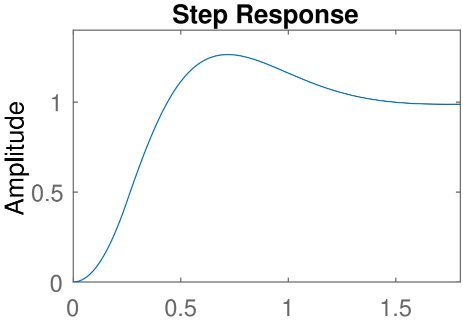
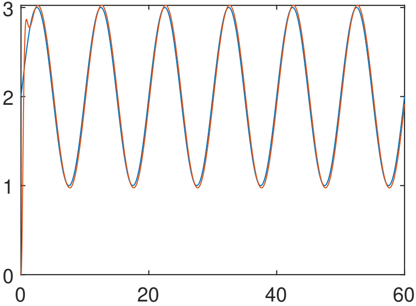


Figure: Lag compensator $C(s) = k_p + \frac{k_i}{s}$



(a) Step response



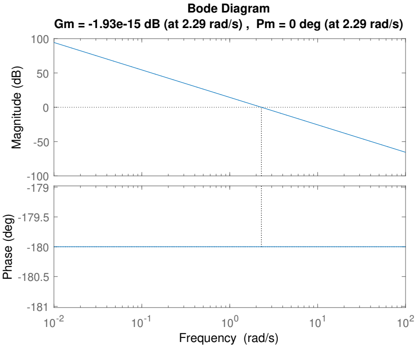
(b) Frequency response at $\omega = \pi/5$

Example 2

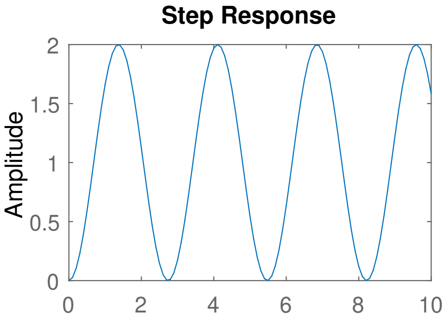
- Plant:

$$P(s) = \frac{r}{Js^2}, \qquad r = 0.25, \quad J = 0.0475$$

- Objectives:
  - Steady-state step error at most 1%
  - Tracking error with $\omega \leq 10$ rad/s at most 10%



(a) Bode plot



(b) Step response for unit negative feedback

Example 2: Lead Compensation

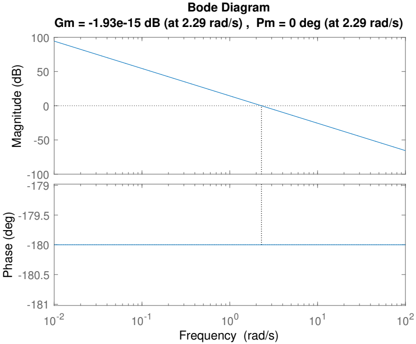


Figure: $P(s)$

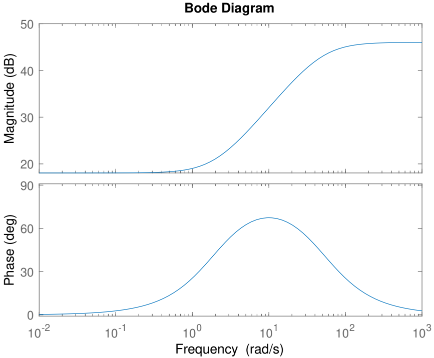


Figure: Lead compensator $C(s) = 200\dfrac{s+2}{s+50}$

Example 2: Lead Compensation

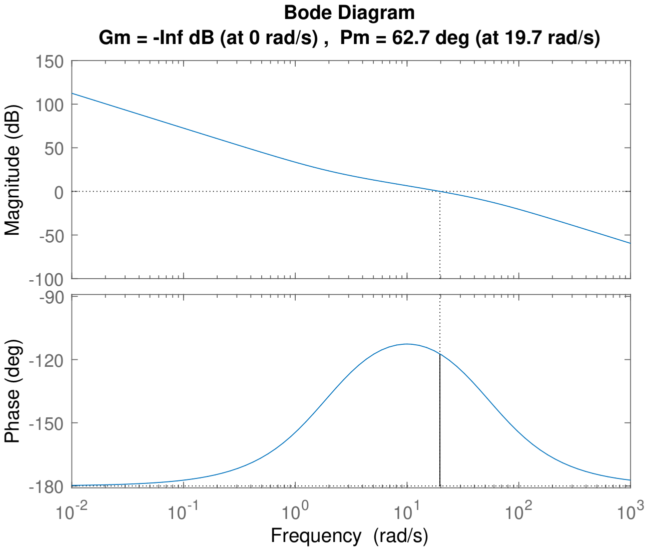
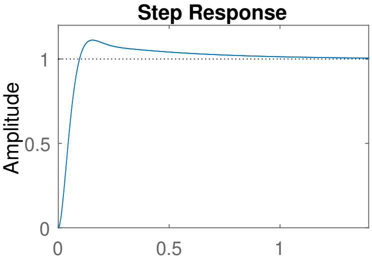
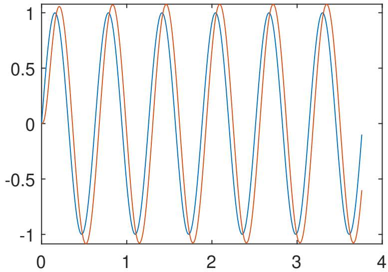


Figure: Margins for $L(s) = C(s)P(s)$

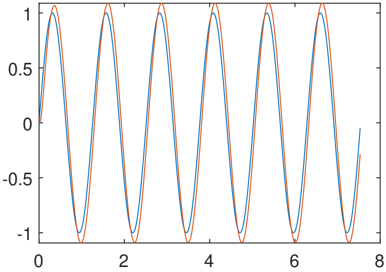
Example 2: Lead Compensation



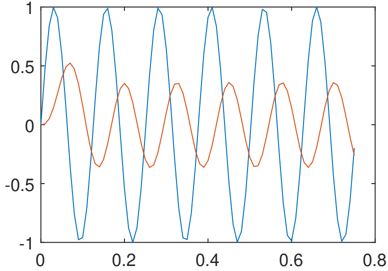
(a) Step response



(b) Frequency response $\omega = 10$



(c) Frequency response $\omega = 1$



(d) Frequency response $\omega = 50$

Example 3

- Plant:

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

- Objectives:
  - Percent overshoot of at most 20% $\Rightarrow$ $\zeta \geq 0.5$
  - Settling time of at most 4 sec $\Rightarrow$ $\zeta\omega_n \geq 1$

- Desired closed-loop poles: $s_{1,2} = -1 \pm j\sqrt{3}$

- Can we place $s_{1,2}$ on the root locus using lead-lag compensation?

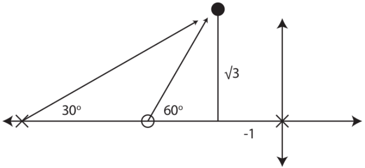
Example 3

- Is $s_1 = -1 + j\sqrt{3}$ already on the Root Locus?
- Check via the **phase condition**:

$$\angle G(s_1) = -\angle s_1 - \angle s_1 + 1 = -120^\circ - 90^\circ = -210^\circ$$

- s_1 is not on the Root Locus and lacks 30° of phase
- Need to add 30° at s_1
- Add a zero at 60° and a pole at 30°:

$$\tan 60^\circ = \frac{\sqrt{3}}{z-1} \qquad \tan 30^\circ = \frac{\sqrt{3}}{p-1}$$

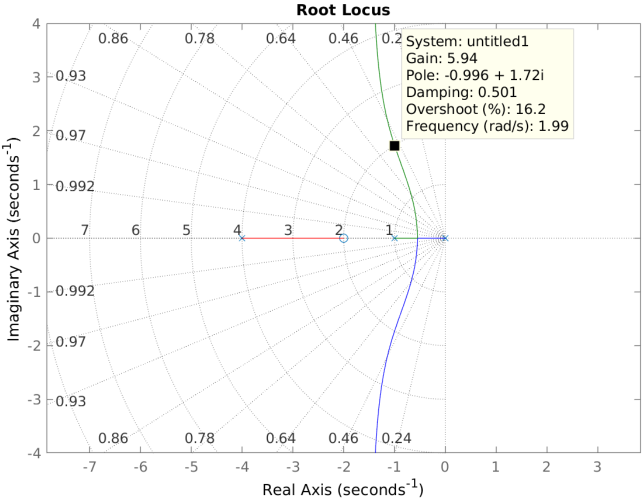


- **Lead compensator**:

$$C(s) = \frac{s+2}{s+4}$$

Example 3

- Root locus of $L(s) = C(s)P(s) = \dfrac{s+2}{s(s+1)(s+4)}$



- Final control design: $C(s) = 6\dfrac{s+2}{s+4}$