ROB 201 | 14 April 2025

Summary: $G(s) = \dfrac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}$ Standard 2nd-order

ω_n = undamped natural frequency

ξ = damping ratio

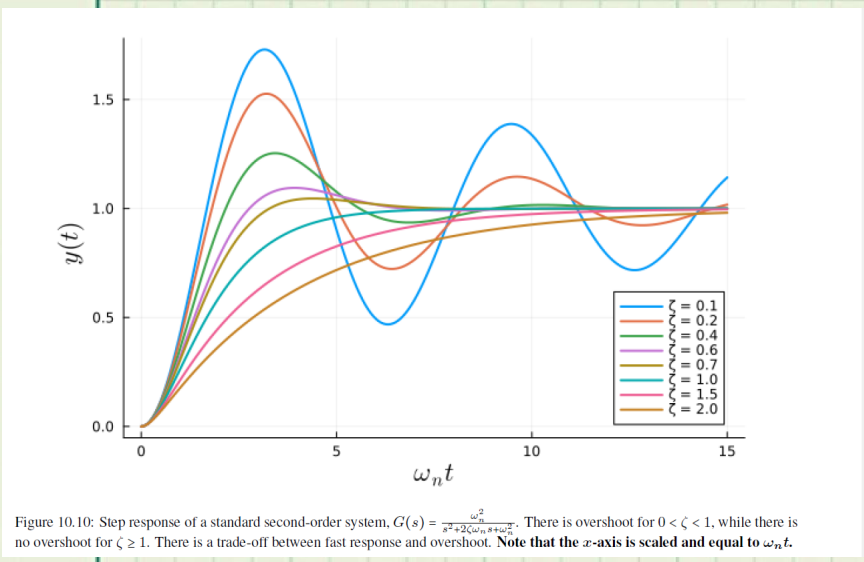
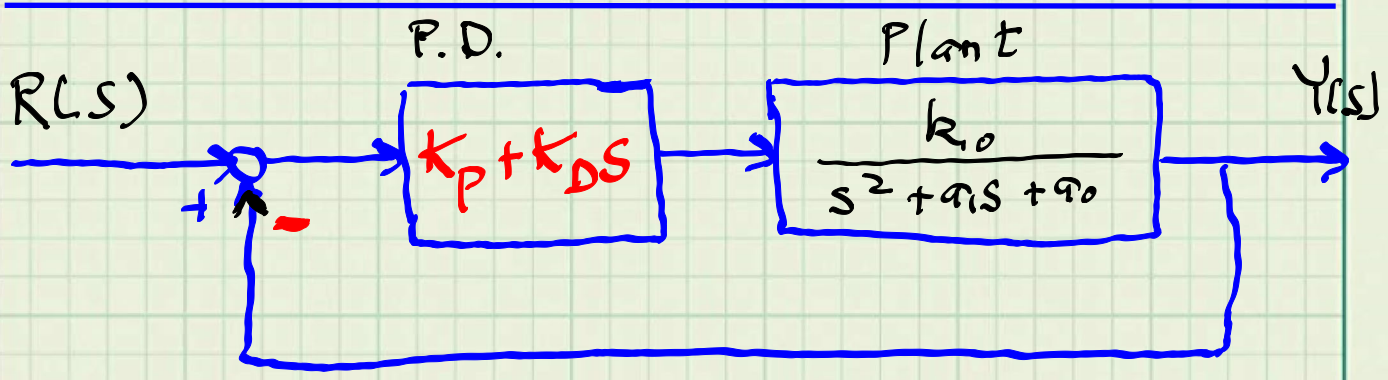


Figure 10.10: Step response of a standard second-order system, $G(s) = \frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}$. There is overshoot for $0 < \zeta < 1$, while there is no overshoot for $\zeta \geq 1$. There is a trade-off between fast response and overshoot. **Note that the x-axis is scaled equal to $\omega_n t$.**

ζ	$\leftrightarrow$	percent overshoot	$= 100e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}}$ %
ω_n	$\leftrightarrow$	5% settling time	$\approx \frac{3}{\zeta\omega_n}$.

$\zeta \to 1$ yields no overshoot. Common numbers to keep in
ntrol system designers use the exact formula for the damp
g will be required in the end no matter what because the m

ζ	≈ Overshoot	True Overshoot
1.0	0 %	0.00 %
0.9	0 %	0.15 %
0.8	2 %	1.5 %
0.7	5 %	4.6 %
0.6	10 %	9.5 %
0.5	15 %	16.3 %



$$\frac{Y(s)}{R(s)} = \frac{k_0(k_P + k_D s)}{s^2 + (a_1 + k_0 k_D)s + (a_0 + k_0 k_P)}$$

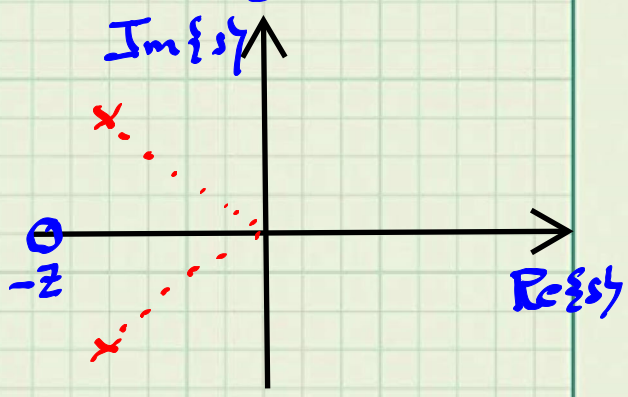
$a_1 + k_0 k_D = 2\xi\omega_n$; $a_0 + k_0 k_P = \omega_n^2$

Overshoot $\longleftrightarrow$ ξ 5% $T_s = \frac{3}{\xi\omega_n} \Leftrightarrow \omega_n = \frac{3}{\xi T_s}$

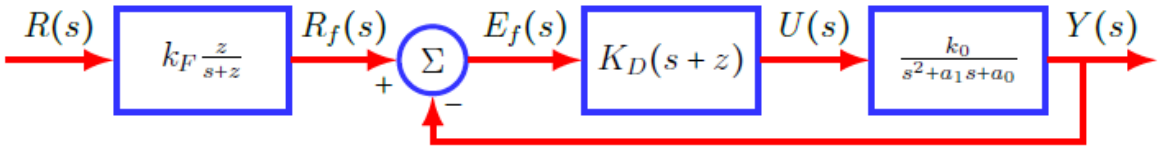
"Small issue": The zero introduced by the PD controller adds EXTRA OVERSHOOT

Write $K_P + K_D s = K_D\left(s + \frac{K_P}{K_D}\right) = K_D(s+z)$

If the zero is in the left-half plane, we can cancel it with a Pre-Compensator



Pre-Compensator **PD $K_D s + K_P$** **Plant**

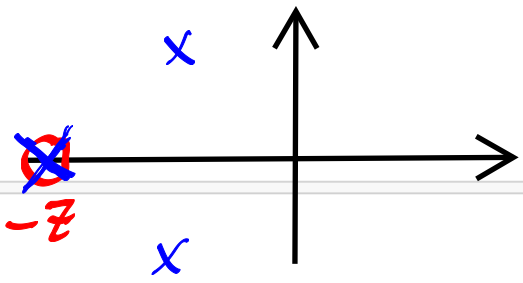


$$\frac{Y(s)}{R(s)} = \left(k_F \frac{z}{s+z}\right)\left(k_0 \frac{K_D(s+z)}{s^2 + (a_1 + K_D k_0)s + (a_0 + K_P k_0)}\right)$$

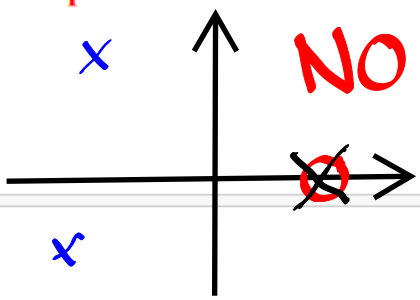
$$= (k_F\ z\ k_0\ K_D)\left(\frac{1}{s^2 + (a_1 + K_D k_0)s + (a_0 + K_P k_0)}\right)$$

Canceling the zero is only allowed when it is in the left-half plane. It returns us to a

YES



NO!



standard second-order system where the overshoot and settling time are easily related to ζ and ω_n.

Fact: "DC Gain" equals one if, and only if, $\frac{k_F\ z\ k_0\ K_D}{a_0 + K_P k_0} = 1$. Solving for k_F yields

$$k_F = \frac{a_0 + k_0\ K_P}{z\ k_0\ K_D} = \frac{a_0 + k_0\ K_P}{k_0\ K_P} = \frac{1 + \frac{k_0}{a_0} K_P}{\frac{k_0}{a_0} K_P} = \frac{1 + C(0)\ P(0)}{C(0)\ P(0)}$$

Today:

Steady-state Step Response of a BIBO Stable Transfer Function

Prop. If $G(s)$ is a rational proper transfer function, then its steady state response to a unit step input is

$$y(t) \longrightarrow G(0)u_s(t),$$

where $G(0)$ is called the DC-gain of the transfer function. ■

Why? From the Laplace transform, we have that $u_s(t) \overset{\mathcal{L}\{\bullet\}}{\longleftrightarrow} \frac{1}{s}$. Let p_j be the poles of $G(s)$, assumed to be distinct. Then, by partial fraction expansion (PFE),

$$Y(s) = \frac{k_0}{s} + \sum_{j=1}^{n} \frac{k_j}{s - p_j},$$

and, by the limit method,

$$k_0 = \lim_{s\to 0} sY(s) = \lim_{s\to 0} sG(s)\frac{1}{s} = G(0).$$

By BIBO stability,

$$\lim_{t\to\infty} \sum_{j=1}^{n} k_j e^{p_j\, t} = 0,$$

which establishes the result. ■

Intuition on PD Controllers

From our understanding of linearization about a point

$$e(t) \approx \underbrace{e(t_0)}_{\substack{\text{current} \\ \text{error}}} + \underbrace{\dot{e}(t)(t-t_0)}_{\substack{\text{predicted error, or} \\ \text{anticipated error}}}$$

$$e(t+\Delta t) \approx e(t_0) + \dot{e}(t_0) \cdot \Delta t$$

where Δt is the look-ahead time. Imagine driving a car through a hole in the car floor, which is equivalent to using proportional control. Then show the videos below.

1. Controlling Self Driving Cars
2. How to Turn - Vision (the most important thing) serves as a metaphor for the look-ahead (prediction into the future) provided by a PD controller, as shown below.

The derivative term allows prediction, as shown below

$$\begin{aligned} u(t) &= K_p e(t) + K_d \dot{e}(t) \\ &= K_p \left[e(t) + \frac{K_d}{K_p} \dot{e}(t) \right] \quad \frac{K_d}{K_p} \text{ is like } \Delta t \\ &= K_p \left[e(t) + \dot{e}(t) \frac{K_d}{K_p} \right] \quad \text{the look-ahead time} \end{aligned}$$

Linearization (aka Linear Approximation) of Nonlinear ODEs

Equilibrium Points

Definition A vector $x_e \in \mathbb{R}^n$ is an **equilibrium point** of the differential equation $\dot{x} = f(x)$ if $f(x_e) = 0_{n\times 1}$. Equivalently, x_e is an equilibrium point if the constant function $\varphi(t, t_0, x_e) := x_e$ is a solution of the ODE.

Note: In simple terms, equilibrium points are initial conditions from which the solution of the ODE does not change; it remains constant and equal to the initial condition. For example, if a pendulum is initialized at rest in the downward hanging position, it stays there!

Examples Find equilibrium points for the following ordinary differential equations,

(a) $\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{g}{L} \cdot \sin(x_1) \end{bmatrix}$ (pendulum model)

(b) $\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -D^{-1}(x_1) \cdot [C(x_1, x_2) \cdot x_2 + G(x_1)] \end{bmatrix}$ (the robot equations).

Solutions:

(a) $\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{g}{L} \cdot \sin(x_1) \end{bmatrix}$ **Ans.** $x_e = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $x_e = \begin{bmatrix} \pi \\ 0 \end{bmatrix}$.

We set the right side to zero and solve for x. This gives $x_2 = 0$, the angular velocity of the pendulum, and $\sin(x_1) = 0$, which implies $x_1 = k\pi$, $k \in \mathbb{Z}$. $k = 0$ corresponds to the downward equilibrium and $k = 1$ gives the upward equilibrium.

(b) $\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -D^{-1}(x_1) \cdot [C(x_1, x_2) \cdot x_2 + G(x_1)] \end{bmatrix}$ **Ans.** $x_e = \left\{ \begin{bmatrix} q_e \\ 0 \end{bmatrix} \in \mathbb{R}^{2n} \mid G(q_e) = 0_{n\times 1} \right\}$

We set the right side to zero and solve for x. The top row gives $x_2 = 0$, the generalized velocity. Substituting this into the bottom row gives $-D^{-1}(x_1) \cdot G(x_1) = 0_{n\times 1} \iff G(x_1) = 0_{n\times 1}$. Because x_1 is the generalized position of the system, we'll use q_e to denote the roots of the potential energy gradient vector, $G(q_e) = 0_{n\times 1}$. There could be no roots, one root, or multiple roots, as with any vector function. ■

Linearization of $\dot{x} = f(x)$

Definition For a nonlinear vector ODE $\dot{x} = f(x)$, the **linearization about an equilibrium point** x_e is the linear vector ODE

$$\delta\dot{x} = \underbrace{\frac{\partial f(x_e)}{\partial x}}_{A} \cdot \delta x,$$

where $\delta x := x - x_e$.

Example Linearize the pendulum model about its equilibrium points.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{g}{L} \cdot \sin(x_1) \end{bmatrix} \text{ with } x_e = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \text{ and } x_e = \begin{bmatrix} \pi \\ 0 \end{bmatrix}$$

Solution:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{g}{L} \cdot \sin(x_1) \end{bmatrix} \quad \textbf{Ans.} \quad \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{L} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \text{ and } \begin{bmatrix} \delta\dot{x}_1 \\ \delta\dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{g}{L} & 0 \end{bmatrix} \begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix}.$$

We identify $f(x) = f(x_1, x_2) = \begin{bmatrix} x_2 \\ -\frac{g}{L} \cdot \sin(x_1) \end{bmatrix}$ and compute $\frac{\partial f(x_1,x_2)}{\partial x} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{L} \cdot \cos(x_1) & 0 \end{bmatrix}$.
Evaluating the Jacobian at the two equilibria gives

$$\frac{\partial f(0,0)}{\partial x} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{L} & 0 \end{bmatrix} \quad \text{and} \quad \frac{\partial f(\pi,0)}{\partial x} = \begin{bmatrix} 0 & 1 \\ \frac{g}{L} & 0 \end{bmatrix}.$$

Hence, the two linear models are

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{L} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \delta\dot{x}_1 \\ \delta\dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{g}{L} & 0 \end{bmatrix} \begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix},$$

where in the first model, because the equilibrium is at the origin, we kept x as the state variable, and in the second model, we are using $\begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix} := \begin{bmatrix} x_1 - \pi \\ x_2 \end{bmatrix}$. There are no hard and fast rules governing these choices. You have a lot of freedom to do what you prefer.

Linearization of the Robot Equations

$$D(q) \cdot \ddot{q} + C(q, \dot{q}) \cdot \dot{q} + G(q) = \Gamma u,$$

where u represents motor torques. When we set $u = 0_{m\times 1}$, the equilibrium points correspond to $G(q_e) = 0_{n\times 1}$ and $\dot{q}_e = 0_{n\times 1}$.

Their linearization about an equilibrium can be expressed in both second-order and first-order forms, namely,

$$D(q_e) \cdot \delta\ddot{q} + \frac{\partial G(q_e)}{\partial q} \cdot \delta q = \Gamma u, \tag{4}$$

where $\delta q = q - q_e$, and

$$\begin{bmatrix} \delta\dot{x}_1 \\ \delta\dot{x}_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 0_n & I_n \\ A_{21} & 0_n \end{bmatrix}}_{A} \begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix} + \underbrace{\begin{bmatrix} 0_{n\times 1} \\ D(q_e)\backslash\Gamma \end{bmatrix}}_{B} u, \tag{5}$$

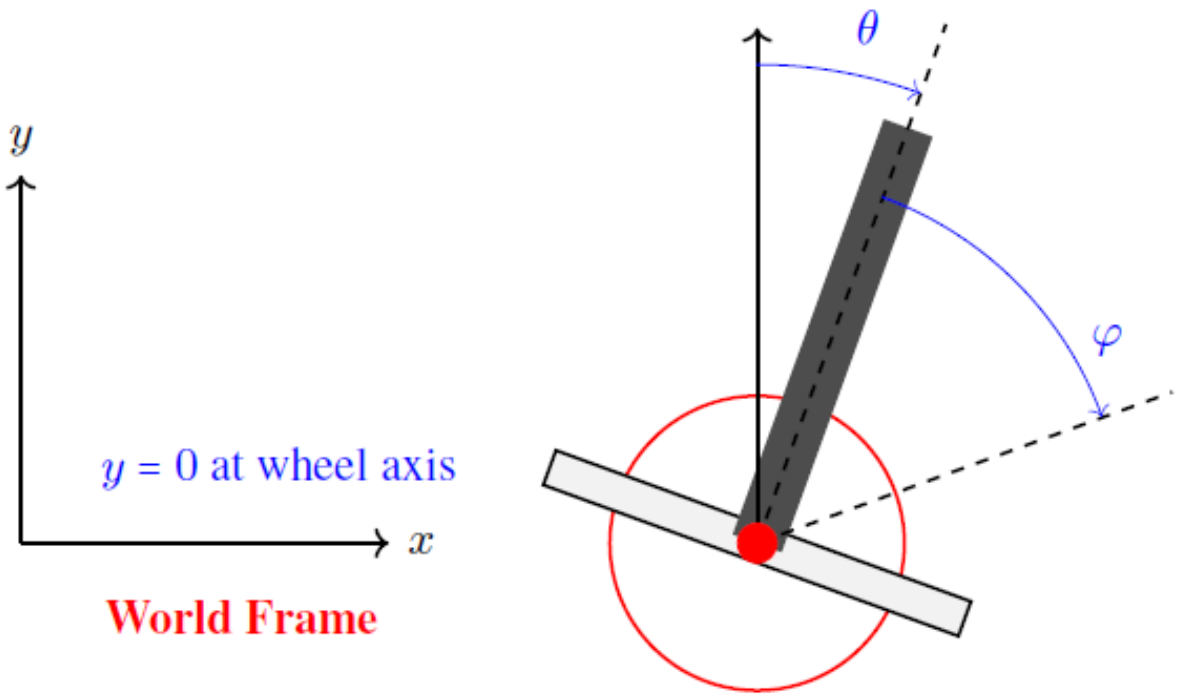
where $A_{21} = -D(q_e)\backslash\frac{\partial G(q_e)}{\partial q}$ because we know not to invert large matrices, I_n is the $n \times n$ identity matrix, $\delta x_1 = q - q_e$, and $\delta x_2 = \dot{q} - 0_{n\times 1}$.

Note: The term $C(q, \dot{q}) \cdot \dot{q}$ does not show up in the linearization because it consists of quadratic terms of the form $c_{ij}(q) \cdot \dot{q}_i \cdot \dot{q}_j$, which when linearized about $\dot{q}_e = 0_{n\times 1}$, $1 \le i \le n$, all vanish.

Feedback Design for a Segway Transporter

(a) (b) (c)

Figure 10.2: Segways take many forms. The one we are modeling is closer to image (a), which shows a classic Segway transporter. Do you think the designer of the balance algorithm for the devices knew ahead of time the exact mass, moments of inertia, etc, for the riders? Or did they have to make the balance algorithm robust to a wide range of human and mechanical riders? To accomplish the latter takes more than our brief introduction to feedback control. You can read about the algorithm used by Cassie to achieve the very meta stunt of a robot riding another robot.



$$x = r_{wh} \cdot (\theta + \varphi) \implies v = \dot{x} = r_{wh} \cdot (\dot{\theta} + \dot{\varphi})$$

$$G_\theta(s) = \frac{-21.468}{s^2 - 50.937} \qquad G_v(s) = \frac{-20.057(s^2 - 7.404)}{s\,(s^2 - 50.937)}$$

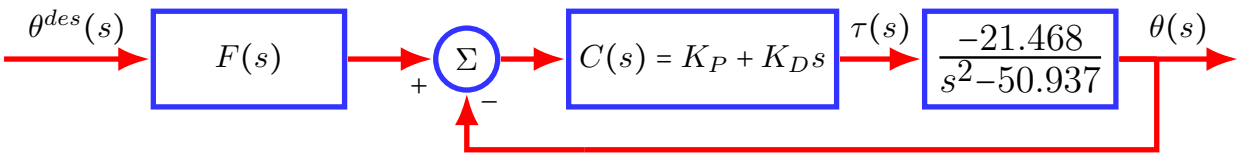
$$u(t) \xrightarrow{\tau(s)} \boxed{\frac{Y_\theta(s)}{U(s)} = \frac{-21.468}{s^2 - 50.937}} \xrightarrow{\theta(s)} y_\theta(t)$$

$$u(t) \xrightarrow{\tau(s)} \boxed{\frac{Y_v(s)}{U(s)} = \frac{-20.057(s^2 - 7.404)}{s(s^2 - 50.937)}} \xrightarrow{V(s)} y_v(t)$$

We start with balance control: maintaining the lean angle, θ, near zero

63 milliseconds for a Face Plant. Terrifying!

- Roots of the transfer function's denominator: $s^2 - 50.937 \implies s_{1,2} = +/- 7.137$
- $\exp(7.137) = 1,257.65$ radians $= 72,058$ degrees = 200 revolutions
- Hence, without feedback control, at T = 0.063 seconds, the Segway has rotated $\exp(7.137 \cdot T) = \frac{\pi}{2}$ radians = 1/4 revolution = **FACE PLANT**.

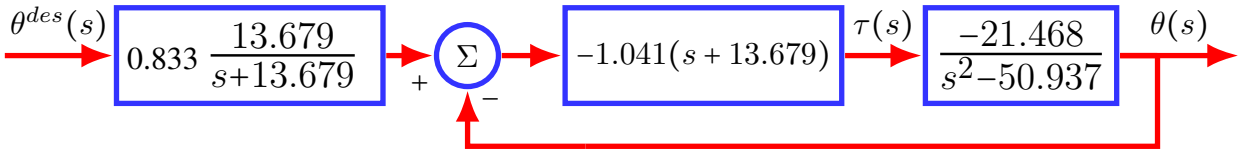


We need to do the following:

- Identify the coefficients $k_0 = -21.468, a_1 = 0, a_0 = -50.937$ in $P(s) = \frac{k_0}{s^2+a_1 s+a_0}$
- Select percent overshoot and settling time. Solve for ζ and ω_n
- Solve for K_P and K_D
- Identify the location of the zero in the PD controller $K_D(s + z)$, where $z = \frac{K_P}{K_D}$
- Design $F(s) = k_F \frac{z}{s+z}$, perhaps setting the DC gain from $\theta^{des}(s)$ to $\theta(s)$ to be one $\left(k_F \cdot \frac{K_P \cdot k_0 \cdot a_0}{a_0 + K_P \cdot k_0} = 1\right)$.

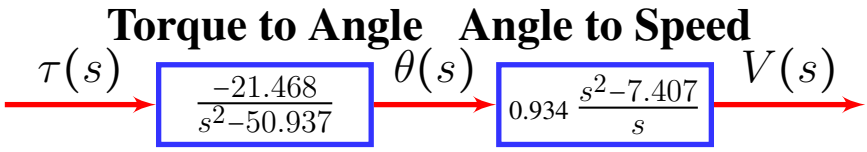
What do you get for K_P and K_D for a percent overshoot of 5% and a settling time of 0.27 seconds? And for k_F?

Go to Demo 5 in Chapter 10: Approx Cell 26 for this balance controller



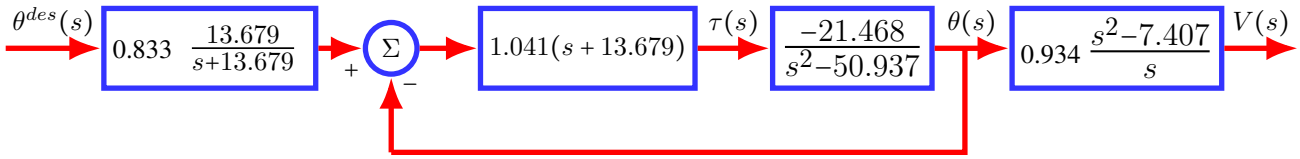
Return here to discuss speed control:

Overall Transfer Function from Torque to Speed



Speed Control:

Have we Pre-Pended a Controller for Learn Angle or Post-Pended the TF for Speed?



While the task may look complicated, from our work on regulating the body orientation, we have that

$$\frac{\theta(s)}{\theta^{des}(s)} = \frac{254.685}{s^2 + 22.342\,s + 254.685},$$

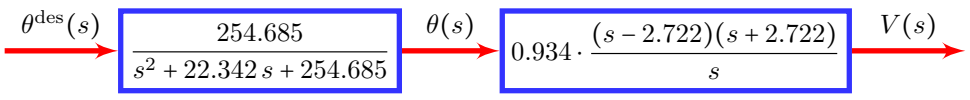
which allows us to replace the first three blocks of the feedback diagram with a standard second-order transfer function, yielding

$$\frac{V(s)}{\theta^{des}(s)} = \left(\frac{254.685}{s^2 + 22.342\,s + 254.685}\right)\left(0.934\,\frac{s^2 - 7.407}{s}\right),$$

or, equivalently,

$$\frac{V(s)}{\theta^{des}(s)} = \left(\frac{254.685}{s^2 + 22.342\,s + 254.685}\right)\left(0.934\,\frac{(s-2.722)(s+2.722)}{s}\right)$$

Hence, we can now focus on the transfer function,



We need to stabilize the pole at the origin and mitigate the effects of the zeros. We cannot do

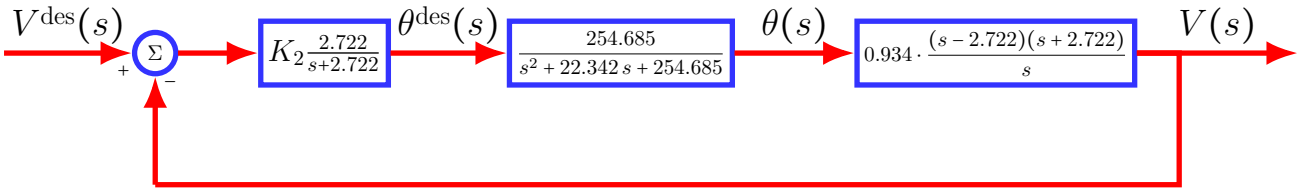
anything about the unstable zero, but we can cancel the stable zero with the transfer function

$$\frac{2.722}{s+2.722}.$$

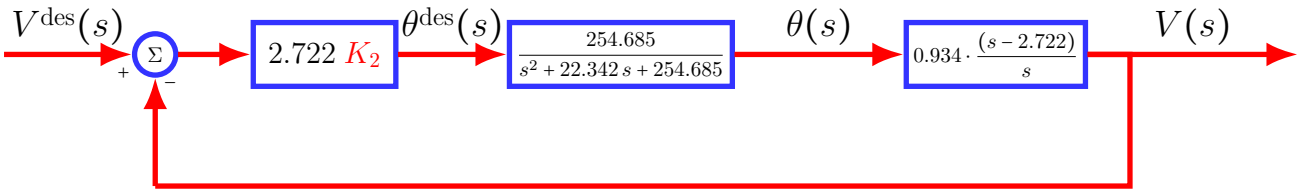
Because the poles of $\frac{254.685}{s^2+22.342\,s+254.685}$ are $-11.171 \pm 11.397\,\mathrm{i}$, which are already nicely into the left half plane, it turns out that, similar to a pole of a first-order transfer function, the pole at the origin can be stabilized with proportional feedback. Hence, for our second compensator, we propose

$$C_2(s) = K_2 \frac{2.722}{s+2.722},$$

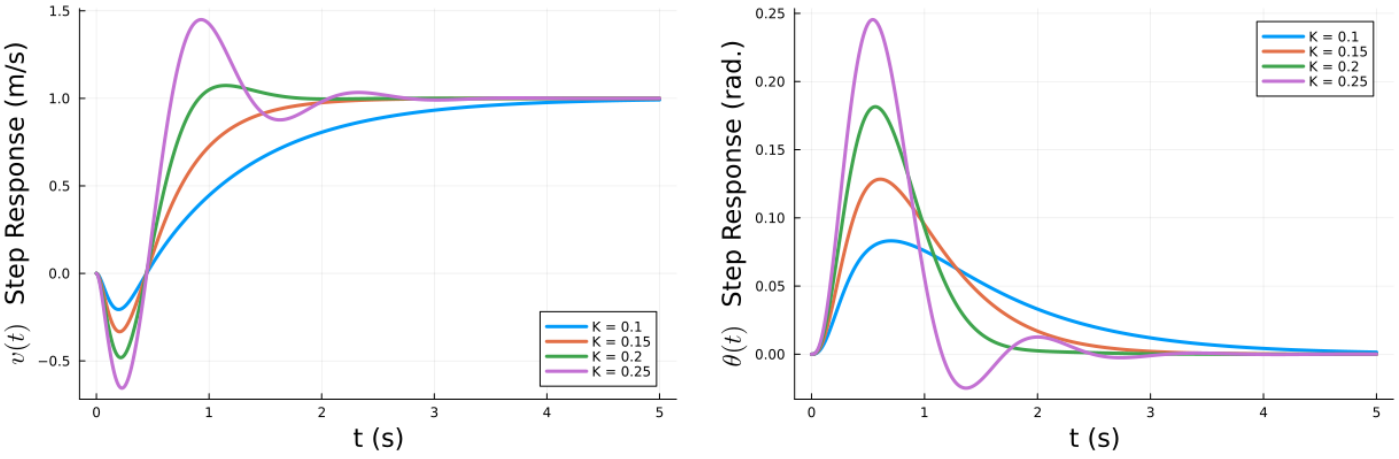
where the gain K_2 will be tuned to achieve a "nice step response".



After performing the **stable pole-zero cancellation**, we are left with hand-tuning a single gain, K_2. The textbook provides the analytical range for BIBO stability, $-0.351 < K_2 < 0$.



Go to Demo 6 in Chapter 10: Approx Cell 30



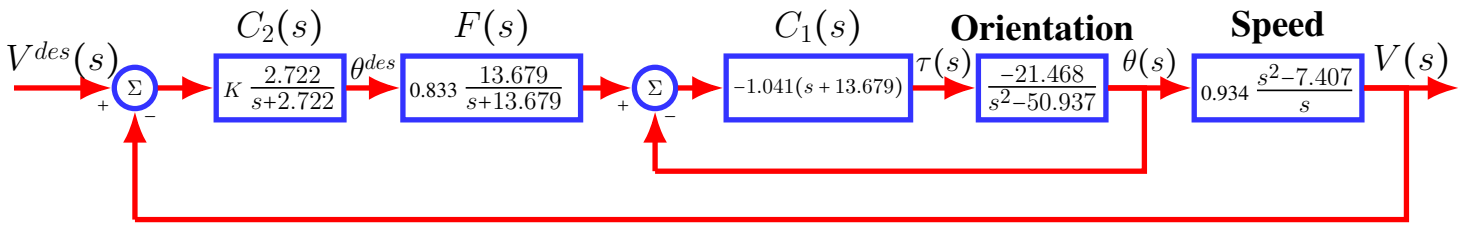
(a) (b)

Step responses from Demo 6: (a) Shows the step response in speed for a range of gains K in $C_2(s)$, while (b) shows the corresponding lean angle trajectories. It is worth looking at both plots. Here, a unit step in speed means 1 m/s, a comfortable walking speed, hence somewhat slow in Segway terms. For $K = 0.2$, there is some overshoot in the speed response, but perhaps not too much. However, in the lean angle, we see approximately 0.17 radians, or 19.5^o of undershoot, which is a lot. The safer bet is likely $K = 0.15$ or even 0.1.

(Optional Read:) Analysis Required to Stabilize the Speed

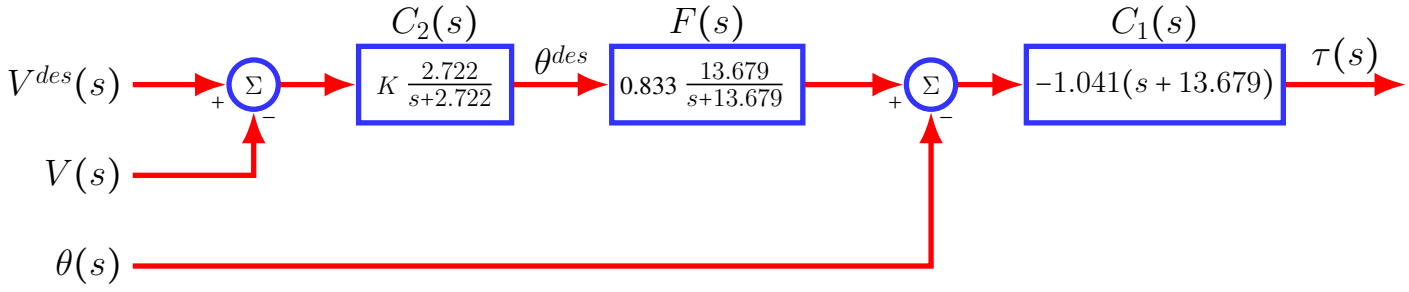
To analyze how to regulate the Segway's forward speed using feedback control, we begin by constructing the full open-loop transfer function of the speed control system. The proposed control architecture consists of a compensator $C_2(s)$ in series with the system's dynamics, all placed in a unity feedback loop.

Come back here for how to implement on a set of nonlinear ODEs



(Inner-Outer Feedback Loops for the Segway) There are two unity feedback loops. The inner loop is from our controller for stabilizing the orientation of the Segway. It includes everything from $\theta^{des}(s)$ (the input to the pre-compensator, $F(s)$) all the way to $\theta(s)$, the output of the orientation block. The other blocks and the left-most summing junction form the outer unity feedback loop for stabilizing the Segway's speed.

Isolating the Controller



(Just the Controller for the Segway) Once the transfer function blocks for the Segway's model are removed from the inner-loop outer-loop block diagram, we are left with just the controller blocks. The controller has three input signals: the desired speed, v^{des}, the measured speed, v, and the measured orientation or lean angle, θ. It has one output signal, the motor torque, τ.

$$\tau(\mathbf{s}) = C_2(s)\,F(s)\,C_1(s)\left(\mathbf{V^{des}(s) - V(s)}\right) - C_1(s)\,\theta(\mathbf{s})$$

$$= \begin{bmatrix} C_2(s)\,F(s)\,C_1(s) & -C_1(s) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{V^{des}(s) - V(s)} \\ \\ \theta(\mathbf{s}) \end{bmatrix},$$

We need to connect this to the Lagrangian model of the Segway, which is expressed in the **time domain** by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -D^{-1}(x_1) \cdot [C(x_1, x_2) \cdot x_2 + G(x_1)] \end{bmatrix} + \begin{bmatrix} 0_{2\times 1} \\ D^{-1}(x_1) \cdot \Gamma \end{bmatrix}$$

We use the Julia Package, **ControlSystems**, to take the controller's transfer function to an equivalent linear ODE model

$$\dot{x}_c = A_c + B_c \begin{bmatrix} v^{des}(t) - v(t) \\ \theta(t) \end{bmatrix}$$

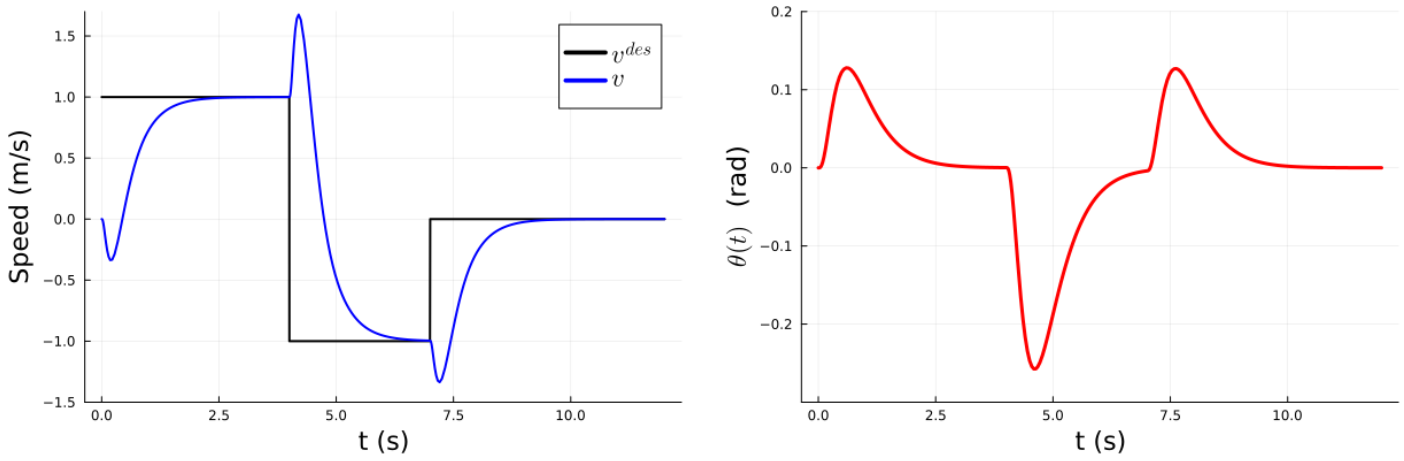
$$\tau(t) = C_c x_c(t)$$

This process is similar to $y(t) \overset{\mathcal{L}\{\bullet\}}{\longleftrightarrow} Y(s)$; you can learn it in EECS 560 Linear Systems Theory, and maybe in ROB 315, eventually!

The ODE form of the controller can be implemented on the Lagrangian model.

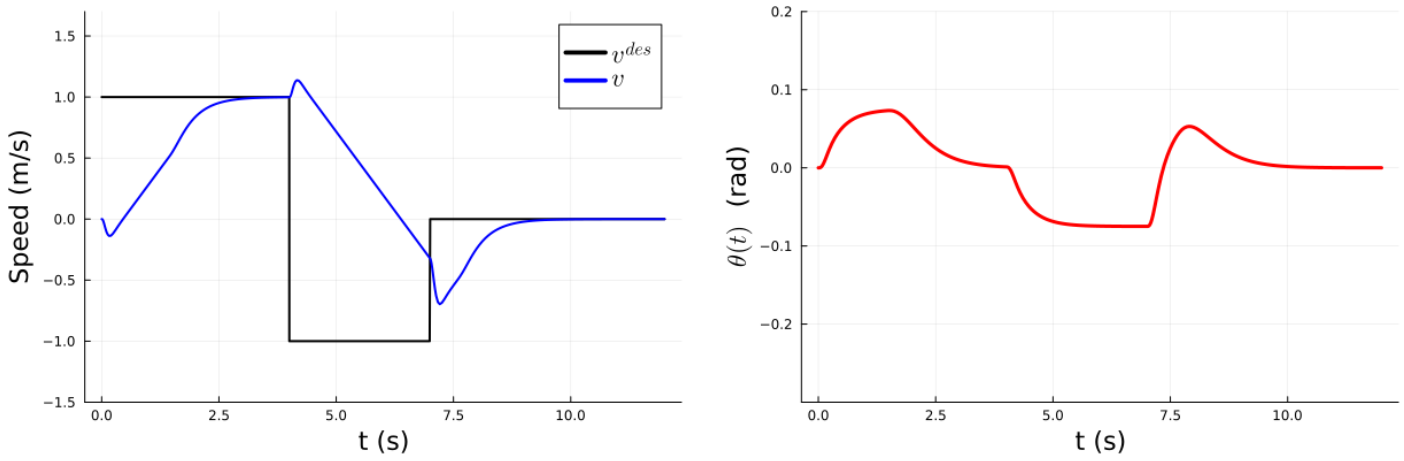
Go to Demo 7 in Chapter 10: The Real Deal: Implementing the Controller on the Nonlinear Model: Approx Cell 33

BallBot: The MBot falls off the basketball when the lean angle exceeds 4^o. Chapter 10 shows how to account for this!



(a) (b)

Tracking a Speed Profile with the Nonlinear Segway Model: (a) Shows a time-varying speed profile as the reference command and the closed-loop linear system's response, while (b) shows the corresponding lean angle trajectory. The results are indistinguishable from the linear case. Why is that? Because the controller maintains a small lean angle, the linear model is valid.



(a) (b)

Tracking a Speed Profile with the Nonlinear Segway Model after a Saturation Element was added to the Feedback Loop: (a) Shows a time-varying speed profile as the reference command and the closed-loop nonlinear system's response, while (b) shows the corresponding lean angle trajectory. The saturation element limits the controller's response to errors in tracking the reference speed to ± 0.5 m/s. This makes the system less responsive on the one hand, while drastically reducing lean angle on the other. **Which is better? Depends on the planned use of the Segway.**