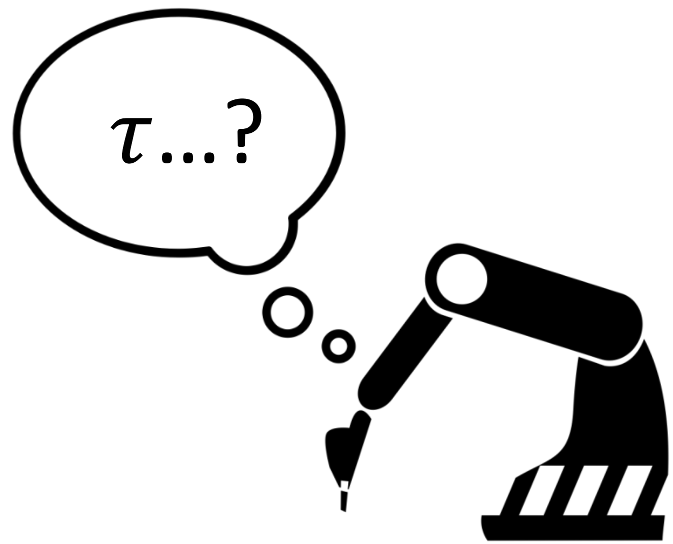
Linear Control for Robotic Manipulators

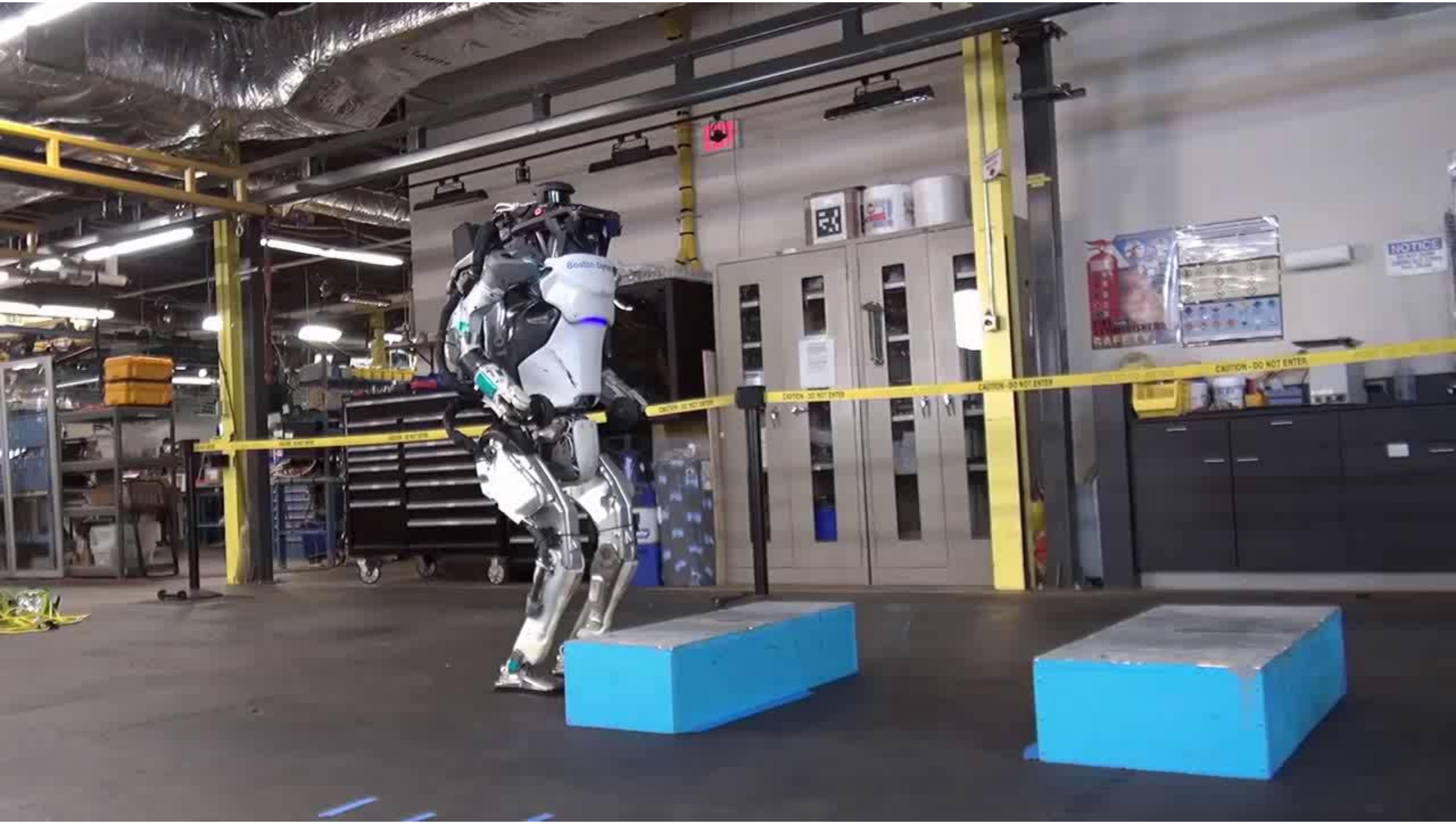


Dylan Losey

MECH 498

Overview

- So far, we have covered:
  - Forward and inverse kinematics
  - Jacobian
  - Dynamics
  - Actuators and sensors
- What’s missing?


CAUTION - DO NOT ENTER
NOTICE

Control

Definition: The *control problem* for robotic manipulators is to determine the sequence of actions (or joint inputs τ) required to cause the robot to execute a desired motion while satisfying certain performance criteria.

- Recall that we found the equation of motion for a robot manipulator:

$$M(q)\ddot{q} + V(q, \dot{q}) + G(q) = \tau$$

- Joint position: $q \in \mathbb{R}^n$
- Mass matrix: $M(q) \in \mathbb{R}^{n \times n}$
- Coriolis terms: $V(q, \dot{q}) \in \mathbb{R}^n$
- Gravity terms: $G(q) \in \mathbb{R}^n$
- Joint inputs: $\tau \in \mathbb{R}^n$

- We want to find the sequence: $\tau(t) \text{ or } \{\tau^0, \tau^1, \tau^2, \ldots\}$

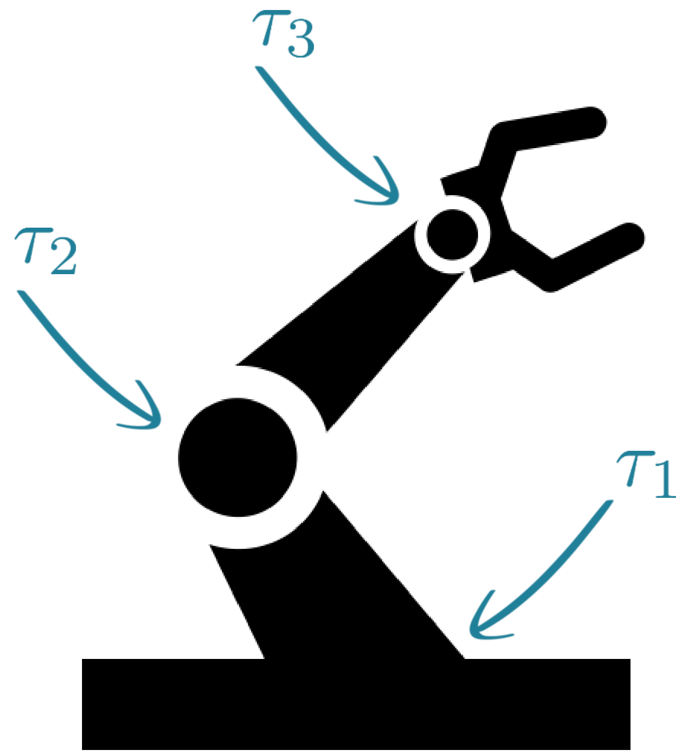
Control

- For multiple DoF robot, need to simultaneously control all joints, so multiple-input, multiple-output (MIMO) control system
- But there is a simpler control strategy: individually controlling each joint, or *independent joint control*

Assumption: the equations of motion for each joint can be separated, and any coupling effects due to the motion of other joints can be treated as a disturbance.

- We now can focus on single-input, single-output (SISO) control systems

Remark: Independent joint control is often used in industry. However, the performance decreases as coupling terms increase!



$$\tau = [\tau_1, \tau_2, \tau_3]^T$$

Linear Control

- Independent joint control is an instance of linear control, since the joints are linear time-invariant (LTI) systems:

$$I_i \ddot{q}_i + b_i \dot{q}_i = \tau_i$$

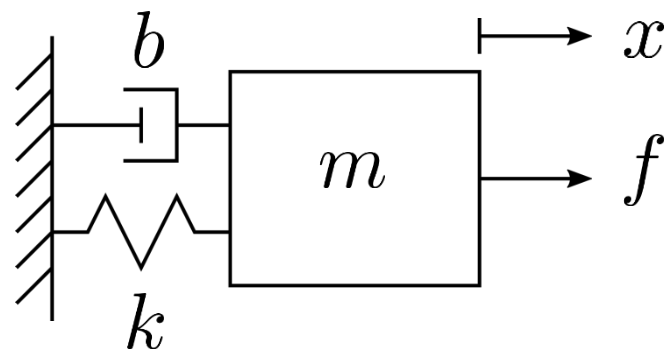
Definition: A *linear time-invariant system* can be modeled with a linear differential equation, and has parameters (i.e., inertia, damping) that do not change over time.

- Main points of linear control for robots we will cover:
  - Second-order linear systems
  - Open-loop control and closed-loop control
  - Stability and passivity
  - Modern control theory
- Reading: **Chapter 9** in Craig, or **Chapter 7** in Spong

Second-Order Linear Systems

- To review basic concepts of linear control, we start by considering a second-order linear system
- A good mechanical analogy is the linear mass-spring-damper:

$$m\ddot{x} + b\dot{x} + kx = f$$



- Moving from the time domain to the Laplace domain, we get the characteristic equation:

$$ms^2 + bs + k = 0$$

Q: *How does the system behave (move) with different mass, damping, and spring constants?*

Second-Order Linear Systems

- There are two common ways to think about the characteristic equation:

$$ms^2 + bs + k = 0$$

$$s^2 + \frac{b}{m}s + \frac{k}{m} = 0$$

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0$$

$$\omega_n = \sqrt{k/m} \qquad \zeta = b/(2\sqrt{km})$$

- Note that either way the characteristic equation is a quadratic equation, and we can solve for the roots of the system:

$$s_{1,2} = -\frac{b}{2m} \pm \frac{\sqrt{b^2-4mk}}{2m}$$

$$s_{1,2} = -\zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1}$$

- These roots tell us how the system will behave when it is perturbed

Second-Order Linear Systems

Definition: if the roots are (negative) real and unequal, such that

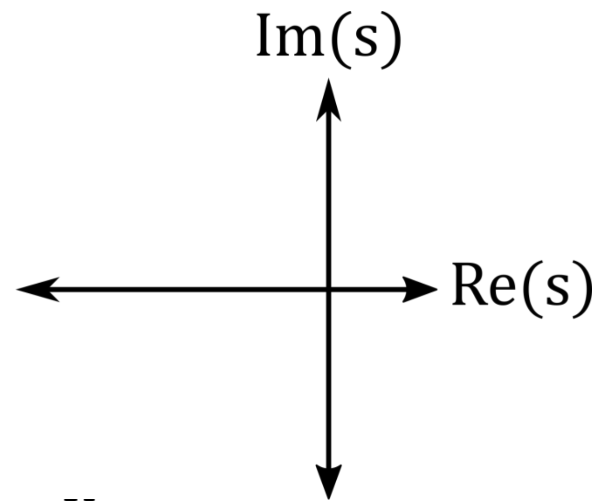
$$b^2 > 4mk \qquad \zeta > 1$$

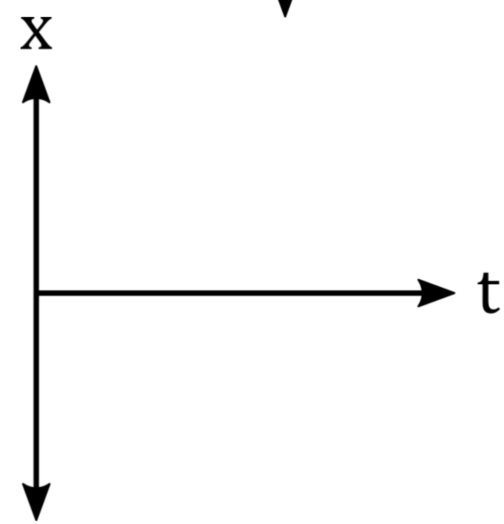
the response is *overdamped*.

- Going back to the time domain, the mass moves according to:

$$x(t) = c_1 \exp s_1 t + c_2 \exp s_2 t$$

- Practically: the response is dominated by the dominant root
- The concept of dominance extends to higher order systems!





Second-Order Linear Systems

Definition: if the roots are complex conjugates, such that

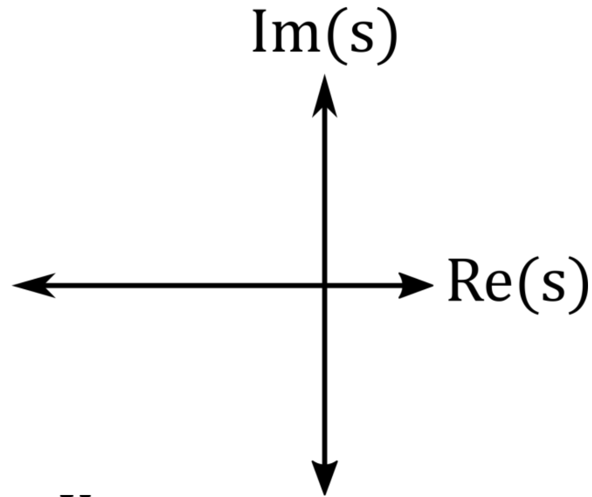
$$b^2 < 4mk \qquad \zeta < 1$$

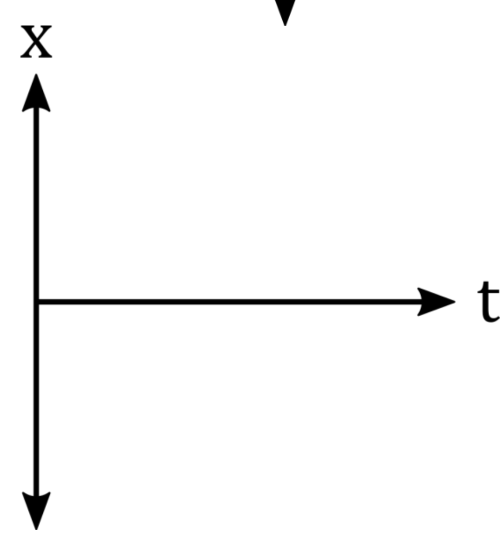
the response is *underdamped*.

- Introduce the damped natural frequency: $\omega_d = \omega_n \sqrt{1 - \zeta^2}$
- Going back to the time domain, the mass moves according to:

$$x(t) = (c_1 \cos \omega_d t + c_2 \sin \omega_d t) e^{-\zeta \omega_n t}$$

- When damping ratio is zero, the system oscillates continually



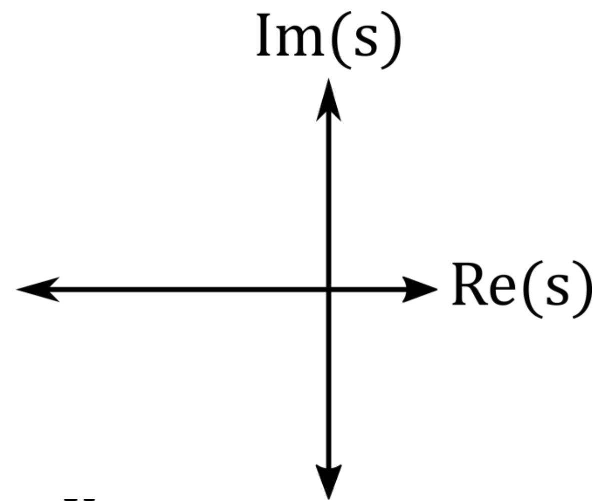


Second-Order Linear Systems

Definition: if the roots are (negative) real and equal, such that

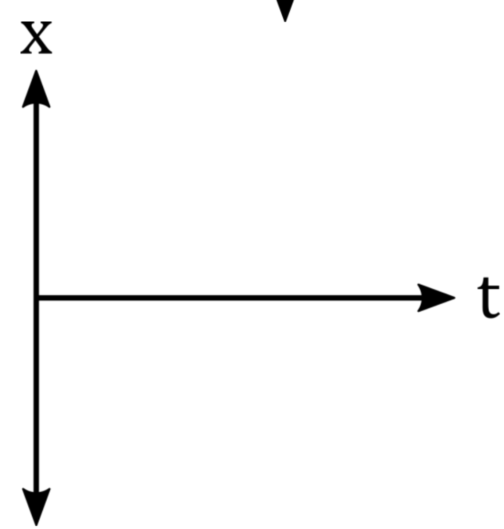
$$b^2 = 4mk \qquad \zeta = 1$$

the response is *critically damped*.



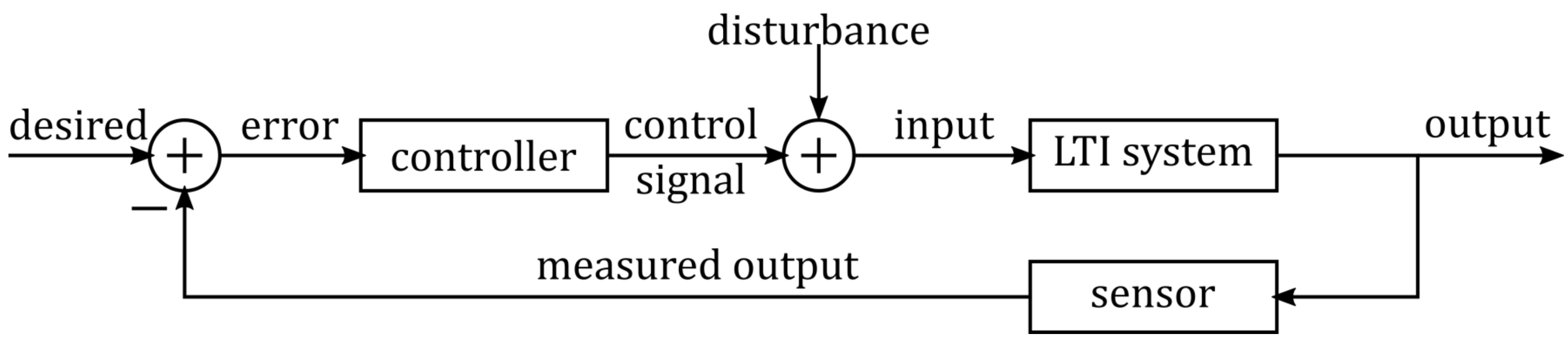
- Going back to the time domain, the mass moves according to:

$$x(t) = (c_1 + c_2 t)\exp\{-\omega_n t\}$$



- We typically want the system to be critically damped
- Q: *But what if we don't have a critically damped system (or the correct response behavior)?*

Linear Control Block Diagram

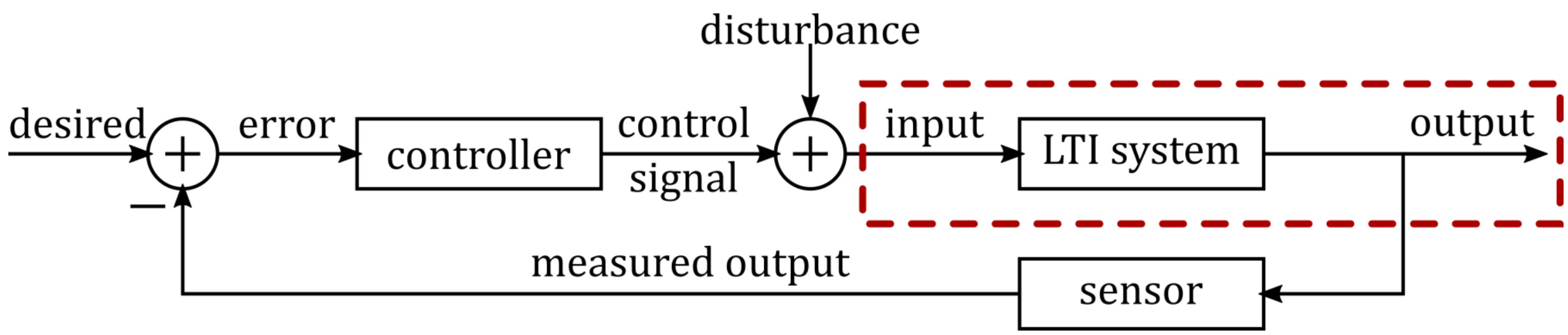


- We can use control to change the behavior of our LTI system
- LTI system may include: amplifier, actuator, transmission
- Sensor is typically: encoder, tachometer, force-torque sensor

Remark: the *control design objective* is to choose the controller so that the output (the actual behavior) always tracks the desired behavior

Remark: the LTI system and sensor are usually given, and *cannot be changed*; the controller is an algorithm, and *can be changed* by the designer

Transfer Function

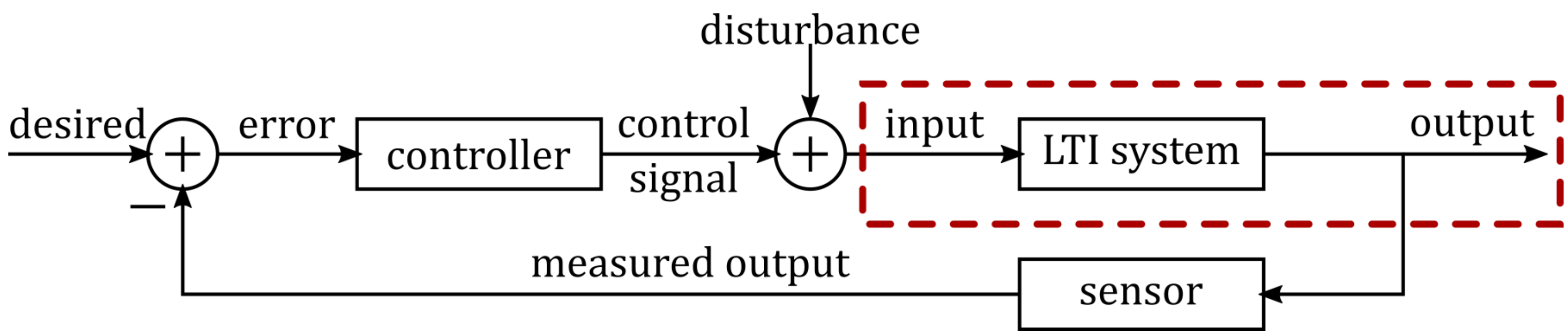


- Recall that an LTI system is a linear ODE relating input to output
- For example, the mass-spring-damper with input $f(t)$ and output $x(t)$:
$$m\ddot{x} + b\dot{x} + kx = f$$
- More generally, a LTI system with input $u(t)$ and output $y(t)$ is modeled:
$$a_n y^{(n)} + \ldots + a_2\ddot{y} + a_1\dot{y} + a_0 y = b_m u^{(m)} + \ldots + b_0 u$$

Theorem: every LTI system is characterized by its *impulse response function*

Definition: the *transfer function* (TF) of a LTI system is the Laplace transform of the impulse response function

Transfer Function



- In the time domain, the output $y(t)$ of the LTI system is the convolution of the input $u(t)$ with the impulse response function $h(t)$:

$$y(t) = u(t) * h(t)$$

- Taking the Laplace transform to get the TF, where convolution is multiplication in the Laplace domain:

$$Y(s) = U(s)H(s)$$

- $H(s)$ is the TF, and can be thought of as the output over input:

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

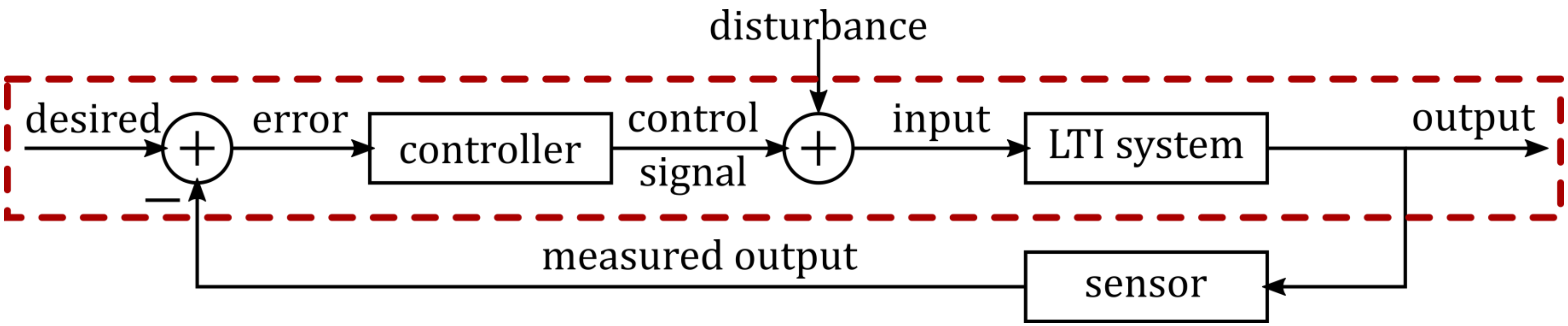
Assumption: Let all *initial conditions be zero* when finding the TF of LTI syst.

Rule of Thumb: TF of LTI syst. is Laplace transform of the equations of motion.

$$a_n y^{(n)} + \ldots + a_2 \ddot{y} + a_1 \dot{y} + a_0 y = b_m u^{(m)} + \ldots + b_0 u$$

$$m\ddot{x} + b\dot{x} + kx = f$$

Open-Loop Control



- Now that we have the TF for the LTI system, we can choose a controller to change the system dynamics (achieve good tracking)

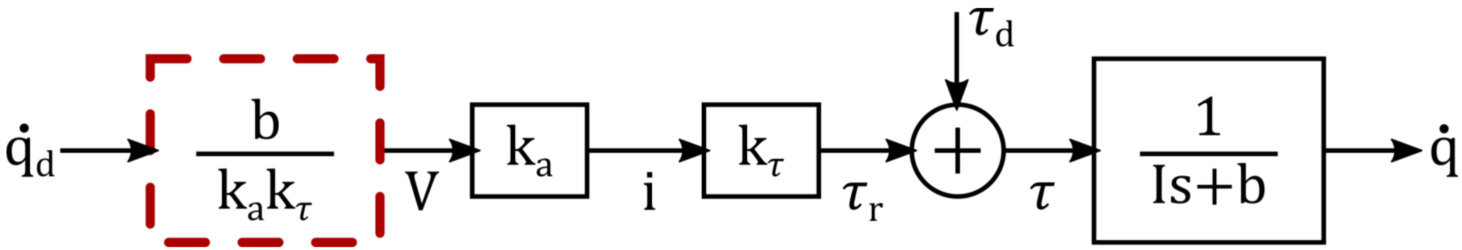
Definition: In *open-loop control*, the system output has no effect on the control signal. This can also be called *feedforward* control.

- Recall our model of a (1-DoF) revolute robotic joint:

$$I\ddot{q} + b\dot{q} = \tau$$

- Let's say that it has a DC motor (torque constant k_τ) and servo amplifier (current gain k_a), and we want to control speed using open-loop control

Open-Loop Control



How did I get the TF of the plant?

$$I\ddot{q} + b\dot{q} = \tau$$

$$Is\dot{q}(s) + b\dot{q}(s) = \tau(s)$$

$$\frac{\dot{q}(s)}{\tau(s)} = \frac{1}{Is+b}$$

Note that the controller commands voltage, and there is an external disturbance

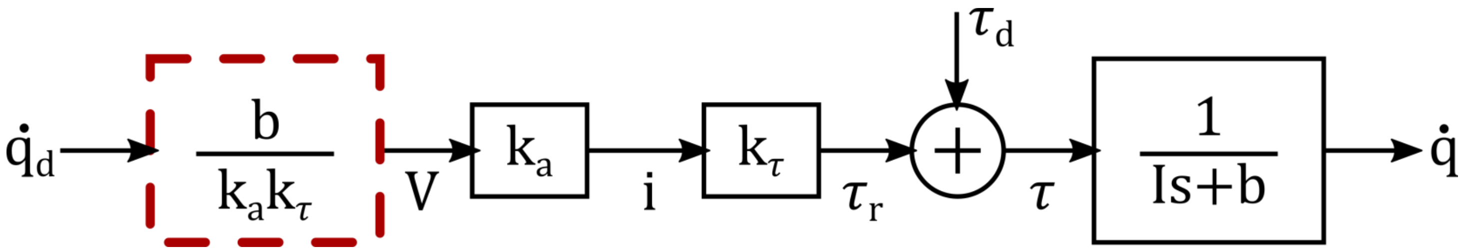
How did I choose the controller?

Remark: to get the *open-loop transfer function* (OLTF), set the disturbance to zero, and then reduce the block diagram

$$\frac{\dot{q}(s)}{V(s)} = \frac{k_a k_\tau}{Is+b}$$

$$\dot{q}(s) = \frac{k_a k_\tau}{Is+b} V(s)$$

Open-Loop Control



How did I choose the controller? (continued)

Theorem (Final Value): If $H(s)$ is the TF, $U(s)$ is the input, $Y(s) = U(s)H(s)$ is the output, the steady state value y_{ss} is given by:

$$y_{ss} = \lim_{t\to\infty} y(t) = \lim_{s\to 0} sY(s) = \lim_{s\to 0} sU(s)H(s)$$

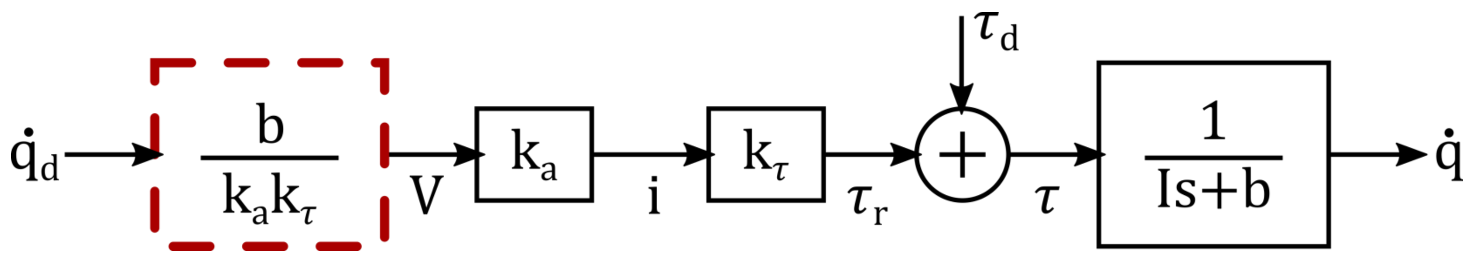
when the poles of $sU(s)H(s)$ are in the left-half plane.

- Applying the FVT to our example, where we have a step input of magnitude v:

$$V(s) = \frac{v}{s} \qquad H(s) = \frac{k_a k_\tau}{Is+b}$$

- So we chose the controller because…

Open-Loop Control



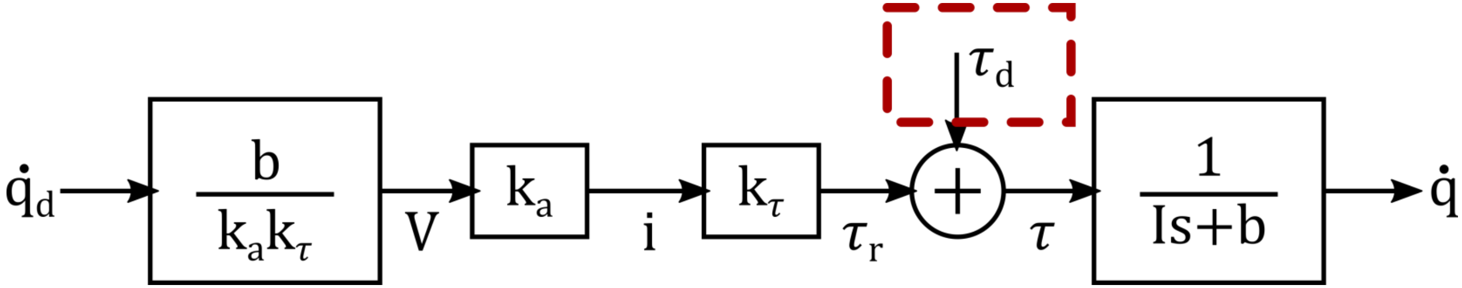
- The OLTF from desired to output is:

$$\frac{\dot{q}(s)}{\dot{q}_d(s)} = \frac{b}{Is+b}$$

- Tracks a step input with no steady-state error
- Easy to implement; no need for sensors or feedback analysis
- What could possibly go wrong?


Boston Dynamics

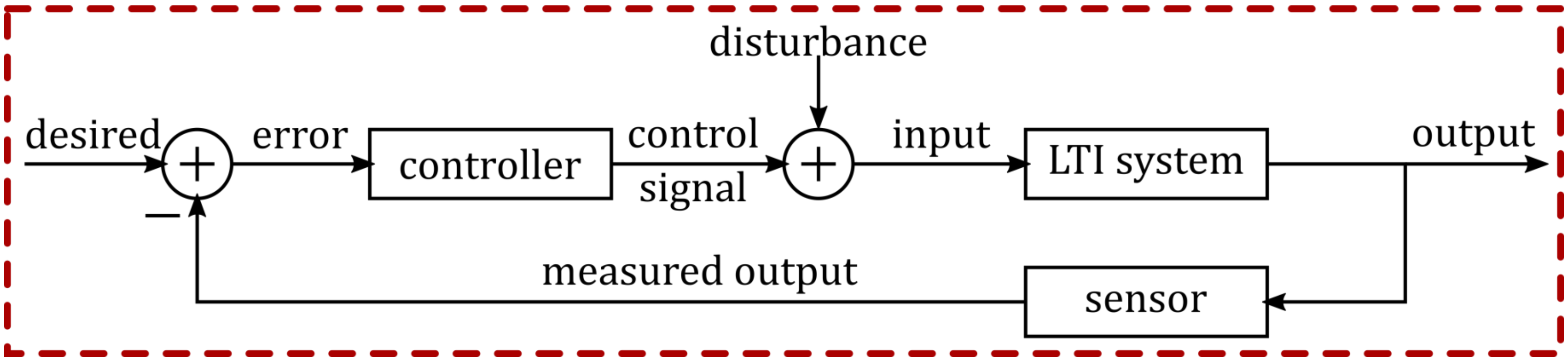
Open-Loop Control



Remark: To get the *disturbance transfer function* (DTF), set the desired to zero, and then reduce the block diagram

- Here we get the DTF: $$\frac{\dot{q}(s)}{\tau_d(s)} = \frac{1}{Is+b}$$
- Applying the FVT, we see that for a unit step disturbance of magnitude τ_d leads to the steady-state error: τ_d/b
- Problems with open-loop control
  - Poor disturbance rejection
  - No compensation for an imperfect model of the LTI system
  - Errors accumulate over time

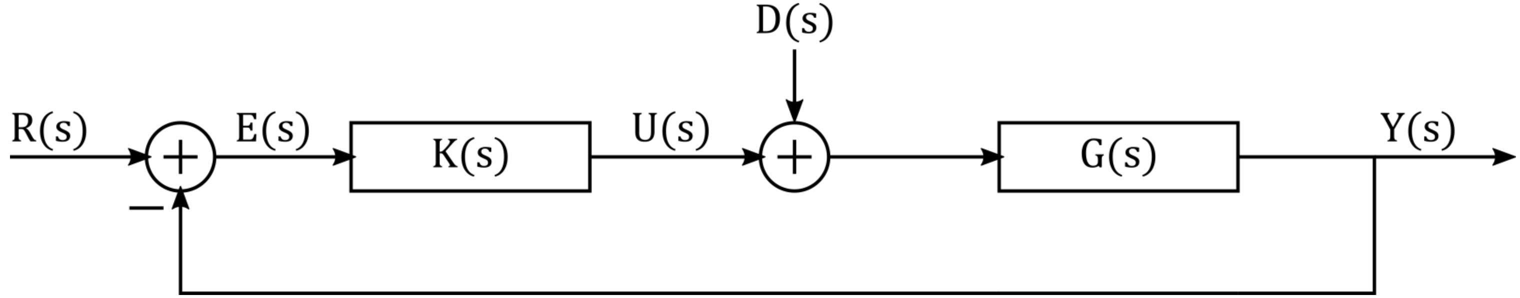
Closed-Loop Control



Definition: In *closed-loop control*, the control signal depends on the measured output. This can also be called *feedback* control.

- Advantages:
  - Superior disturbance rejection
  - Reduces sensitivity to modeling errors
  - Can cause larger changes in the system dynamics
- Disadvantages:
  - Can make system unstable (we will focus on this)
  - Requires sensors for feedback, expensive
  - Sensitive to errors or changes in sensors

Closed-Loop Control



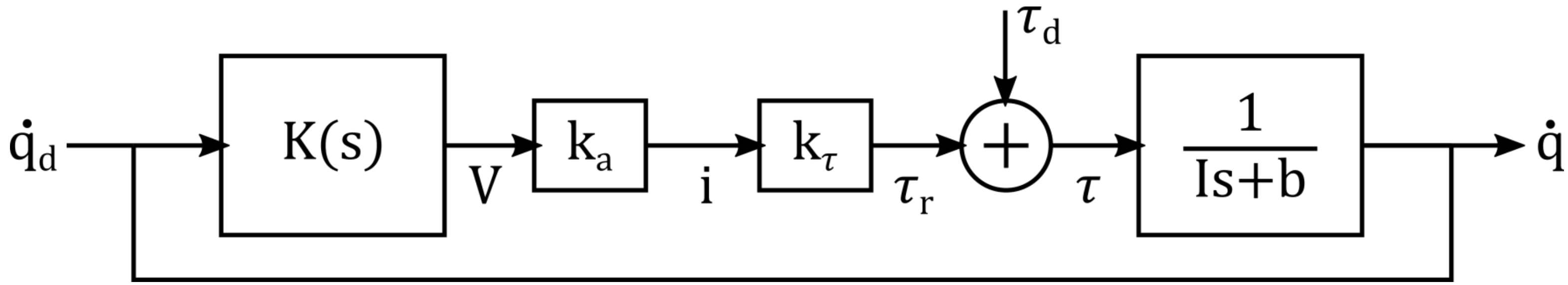
- Here is a *general* block diagram for most linear closed-loop systems:
  - $K(s)$ is the TF of the controller
  - $G(s)$ is the TF of the LTI system, also called the "plant"
- Important transfer functions that will come up: CLTF, DTF, Total Response

Remark: to get the *closed-loop transfer function* (CLTF), set the disturbance to zero, and then reduce the block diagram using Black's Law

$$\frac{Y(s)}{R(s)} = \frac{K(s)G(s)}{1+K(s)G(s)} \qquad \frac{Y(s)}{D(s)} = \frac{G(s)}{1+K(s)G(s)}$$

$$Y(s) = \frac{K(s)G(s)}{1+K(s)G(s)}R(s) + \frac{G(s)}{1+K(s)G(s)}D(s)$$

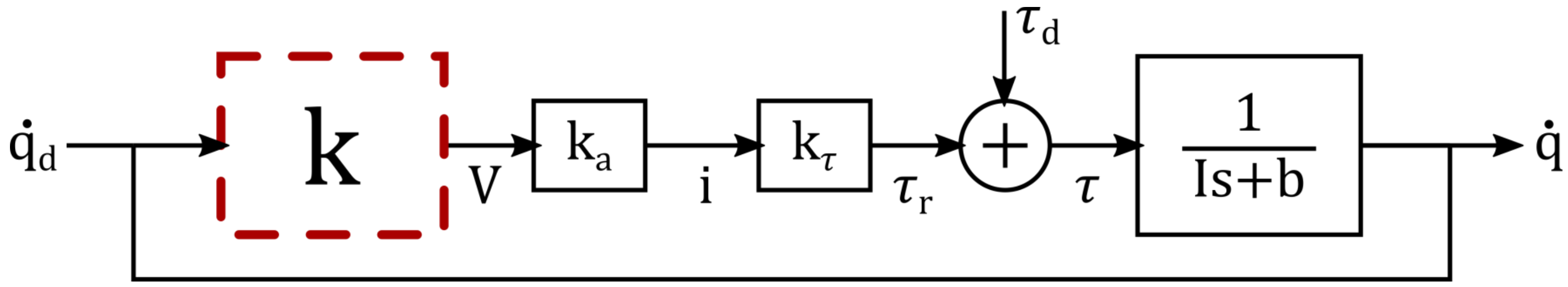
Closed-Loop Control



- Let's repeat our example of controlling the speed of a revolute joint
- What are the CLTF and the DTF? Recall that we know:

$$\frac{Y(s)}{R(s)} = \frac{K(s)G(s)}{1+K(s)G(s)} \qquad \frac{Y(s)}{D(s)} = \frac{G(s)}{1+K(s)G(s)}$$

P Controller



Definition: In a *proportional controller*, $K(s) = k$, where k is a positive constant.

- When we use a proportional controller, the CLTF and DTF are:

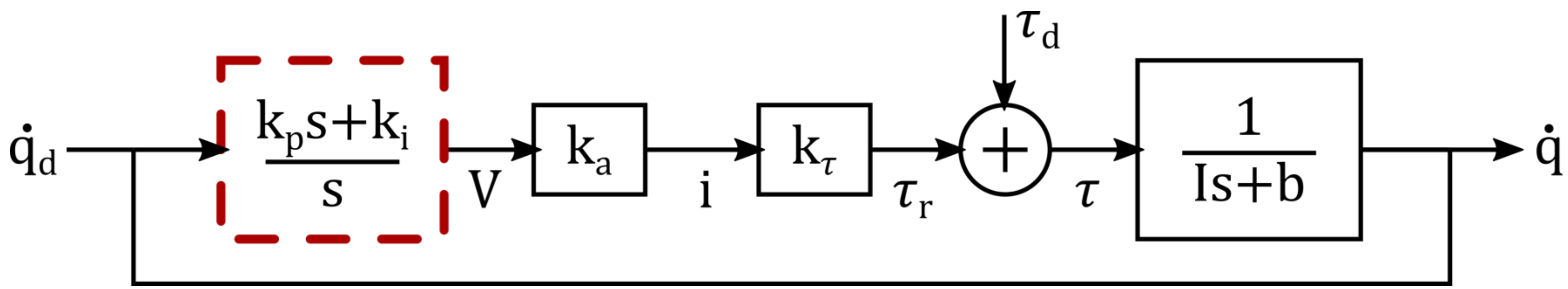
$$\frac{\dot{q}(s)}{\dot{q}_d(s)} = \frac{k k_a k_\tau}{Is+b+k k_a k_\tau} \qquad \frac{\dot{q}(s)}{\tau_d(s)} = \frac{1}{Is+b+k k_a k_\tau}$$

- Apply the FVT (check poles!) to determine the step response for both CLTF and DTF:

$$\dot{q}_{ss} = \frac{k k_a k_\tau}{b+k k_a k_\tau}\dot{q}_d \qquad \dot{q}_{ss} = \frac{1}{b+k k_a k_\tau}\tau_d$$

- So, for proportional control, we can reduce the steady-state error and improve disturbance rejection by increasing k

PI Controller



Definition: In a *proportional-integral controller*, with positive constants k_p and k_i:

$$K(s) = \frac{k_p s + k_i}{s}$$

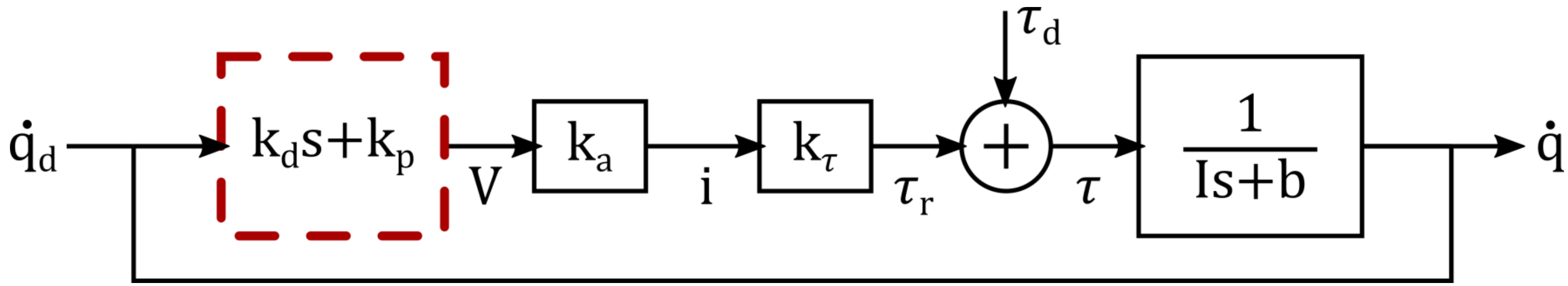
- Repeating the steps from before, when we use a PI controller:

$$\frac{\dot{q}(s)}{\dot{q}_d(s)} = \frac{k_p k_a k_\tau s + k_i k_a k_\tau}{Is^2 + (b + k_p k_a k_\tau)s + k_i k_a k_\tau} \qquad \frac{\dot{q}(s)}{\tau_d(s)} = \frac{s}{Is^2 + (b + k_p k_a k_\tau)s + k_i k_a k_\tau}$$

$$\dot{q}_{ss} = \frac{k_i k_a k_\tau}{k_i k_a k_\tau} \dot{q}_d \qquad \dot{q}_{ss} = \frac{0}{k_i k_a k_\tau} \tau_d$$

- Here PI control eliminates the steady-state tracking error, and the system is robust to step disturbances!

PD Controller



Definition: In a *proportional-derivative controller*, with positive constants k_p and k_d:

$$K(s) = k_d s + k_p$$

- Repeating the steps from before, when we use a PI controller:

$$\frac{\dot{q}(s)}{\dot{q}_d(s)} = \frac{(k_d s + k_p)k_a k_\tau}{(I + k_d k_a k_\tau)s + b + k_p k_a k_\tau} \qquad \frac{\dot{q}(s)}{\dot{\tau}_d(s)} = \frac{1}{(I + k_d k_a k_\tau)s + b + k_p k_a k_\tau}$$

$$\dot{q}_{ss} = \frac{k_p k_a k_\tau}{b + k_p k_a k_\tau}\dot{q}_d \qquad \dot{q}_{ss} = \frac{1}{b + k_p k_a k_\tau}\tau_d$$

- Here PD control allows us to move the pole to make a slower (or faster) CL response!
- Steady-state response here similar to P control, but not always the case

Stability

Definition: if a dynamical system at equilibrium is perturbed, the system is *stable* if it returns to that equilibrium

- Stability is a property of the system, not the input

Theorem: a LTI system is (exponentially) stable *iff all TF poles are in the left-half plane* (LHP), i.e., have strictly negative real parts

Definition: *poles* are the roots of the denominator of a TF

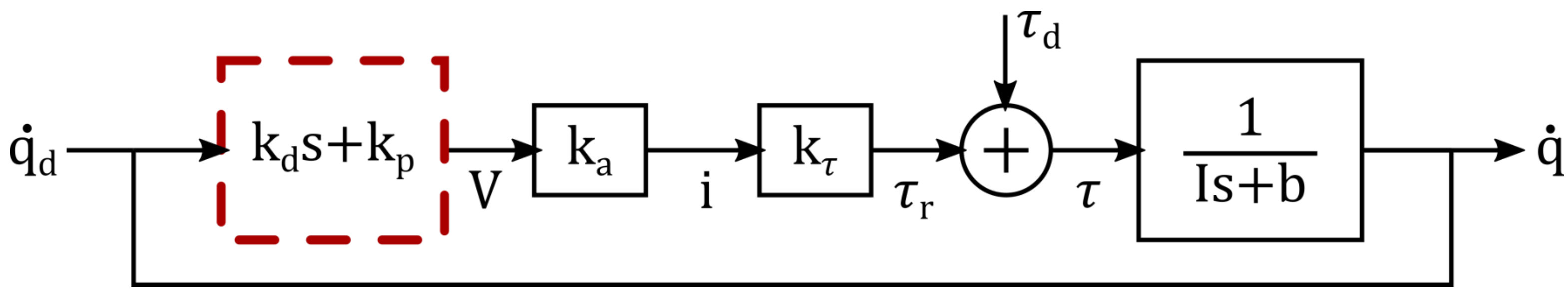
Definition: *zeros* are the roots of the numerator of a TF

- LTI systems are either asymptotically stable or unstable

Aside: Exponential stability is a type of asymptotic stability

- Stability issues usually come up for CL systems, since most physical OL plants are already stable

Stability



- To see an example of CL stability, let's look back at the PD example
- Here are the original plant $G(s)$ and the CLTF with PD control:

$$G(s) = \frac{\dot{q}(s)}{\tau(s)} = \frac{1}{Is+b} \qquad \frac{\dot{q}(s)}{\dot{q}_d(s)} = \frac{(k_d s + k_p) k_a k_\tau}{(I + k_d k_a k_\tau)s + b + k_p k_a k_\tau}$$

- So the pole of the system was originally at: $s = -b/I$ (stable)
- And the pole of the CLTF is now at: $s = \frac{-b - k_p k_a k_\tau}{I + k_d k_a k_\tau}$
- *Q: Is this CL system stable?*
- What if we were to (inexplicably) choose: $k_p < -\frac{b}{k_a k_\tau}, \quad k_d > 0$

Stability

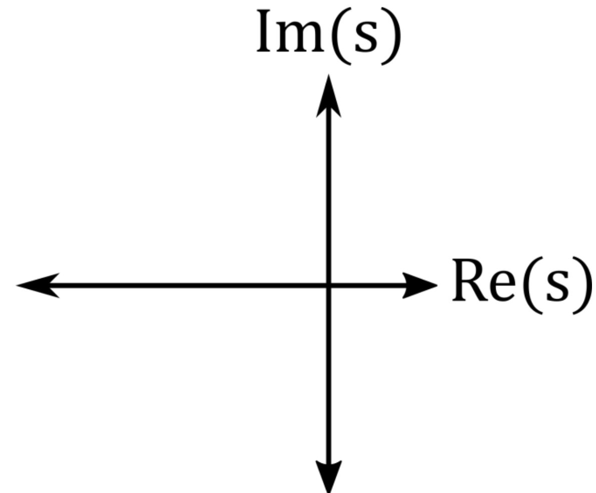
- We can make a stable OL system CL unstable by choosing
  - "wrong" type of controller
  - "wrong" controller gains

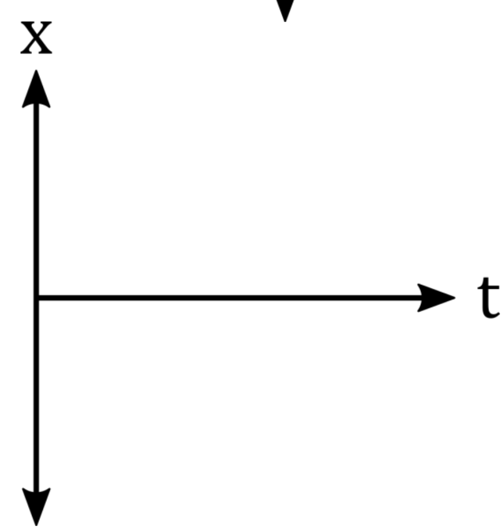
Remark: we typically want to design $K(s)$ such that the roots of the *characteristic equation:*

$$1 + K(s)G(s) = 0$$

are as far into the LHP as possible

- Stability has to be proven before we consider performance
- Stability is particularly important for robots working next to humans





So how do I pick a controller to make sure my robot is stable?

Stability Analysis

- The name of the game in classical LTI controls is to predict the behavior of the CL system by looking at OL systems
  - Root locus (visualize CL poles)
  - Bode plot (frequency response)
  - Nyquist plot (relative stability)
- We can use these techniques to assess the stability of different controllers
- Here we will briefly review root locus plots

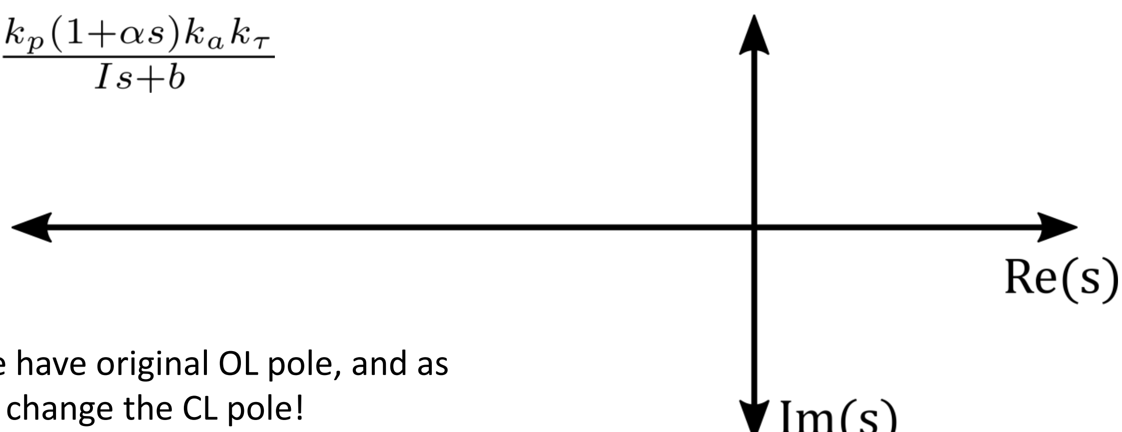
Root Locus

Definition: Root locus is a method of visualizing the locations of the CL poles as the controller gain changes

- Can check root locus to see if controller leads to stable system
- Can also see how different controllers affect performance
- Helps you design a controller (where to put poles and zeros)

Example: PD controller for robot joint

$$OLTF = \frac{k_p(1+\alpha s)k_a k_\tau}{Is+b}$$
$$\alpha = \frac{k_d}{kp}$$



Note: if k=0, we have original OL pole, and as k increases, we change the CL pole!

Root Locus

There are eight rules to making a RL plot:

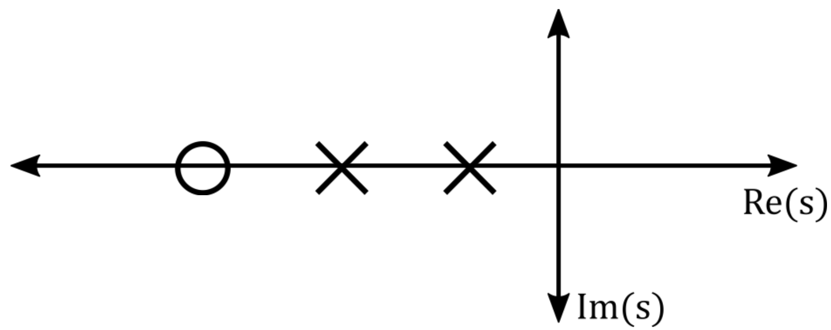
1. Write characteristic equation in form $1 + kG^*(s) = 0$
2. Plot poles "x" and zeros "o" of $G^*(s)$ on the s-plane
   – Aside: $G^*(s)$ is the scaled OLTF, contains the (scaled) controller and plant
3. Calculate $r = n - m$, where n number of poles, m number of zeros
4. RL starts at poles and ends at zeros, leaves along r asymptotes as $k \to \infty$
5. Draw r asymptotes
   – Angle is given by $180/r$
   – Intercept is given by: $\sigma = \frac{\sum \Re e[poles] - \sum \Re e[zeros]}{r}$
6. Draw the locus to the left of an odd number of poles + zeros
7. Break-out points occur half-way between two poles, and break in points occur half-way between two zeros (on the real axis)
8. Zeros attract the locus, poles repel the locus

Root Locus

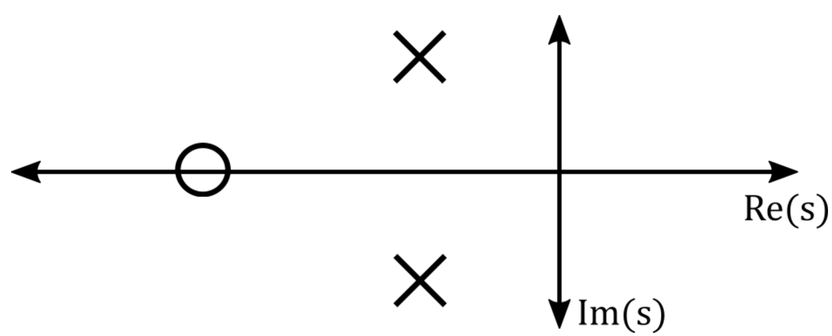
- **Example**: second-order linear system and PD controller
- Recall that we want to make the system critically damped

$$1 + kG^*(s) = 1 + k\frac{1+\alpha s}{ms^2+bs+k} = 0$$

Case 1: Overdamped
$b^2 > 4mk$



Case 2: Underdamped
$b^2 < 4mk$



Root Locus (review)

Remark: a root locus plot shows us the closed loop poles for different overall controller gains given the (scaled) OLTF

Remark: we can use root locus to quickly check stability, and to reverse engineer stable controllers

- Great summary of root locus:
  - http://lpsa.swarthmore.edu/Root_Locus/DeriveRootLocusRules.html
- For our purposes, the main rules are:
  - The root locus goes from OL poles to zeros as k increases
  - The locus exists on real axis to the left of an odd # of poles + zeros
  - The root locus is symmetric about the real axis
  - The root locus has $r = \#poles - \#zeros$ asymptotes
- Can always use rlocus in MATLAB to check

Root Locus Examples

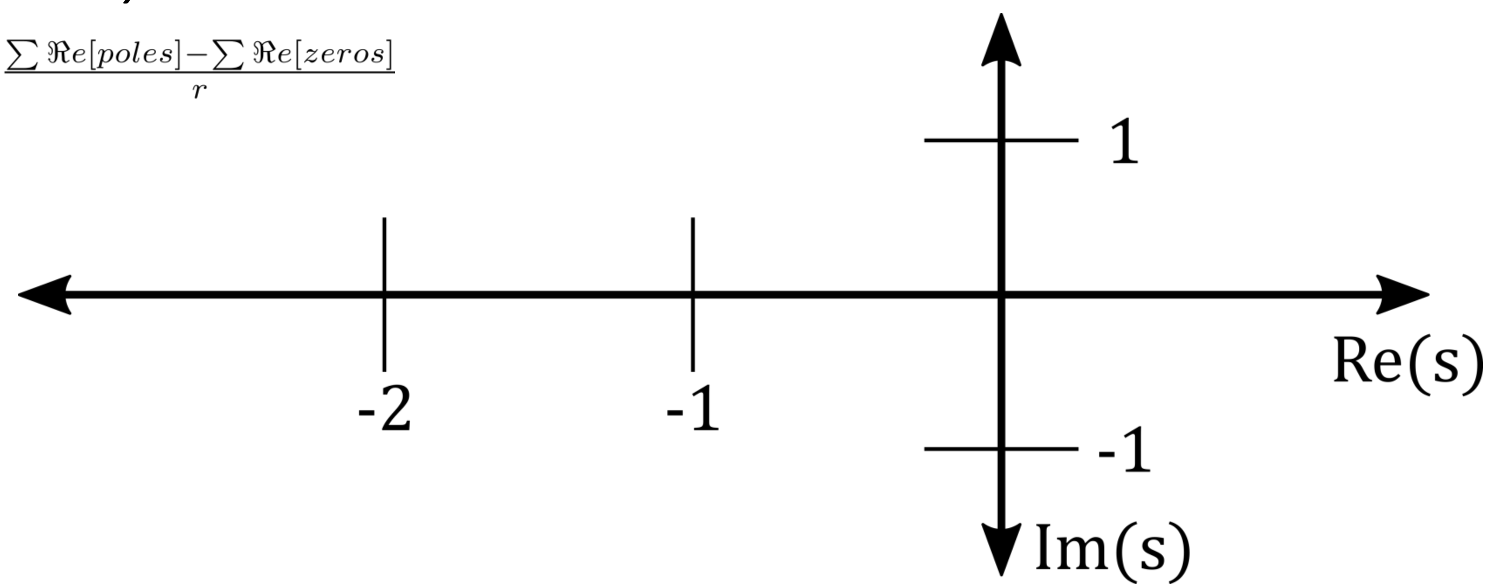
Example: Consider a OLTF with the following poles and zeros:

Poles: $s = -2, s = -1 \pm i$

Zeros: $s = -1$

Q: is this system stable?

$$\sigma = \frac{\sum \Re e[poles] - \sum \Re e[zeros]}{r}$$



Root Locus Examples

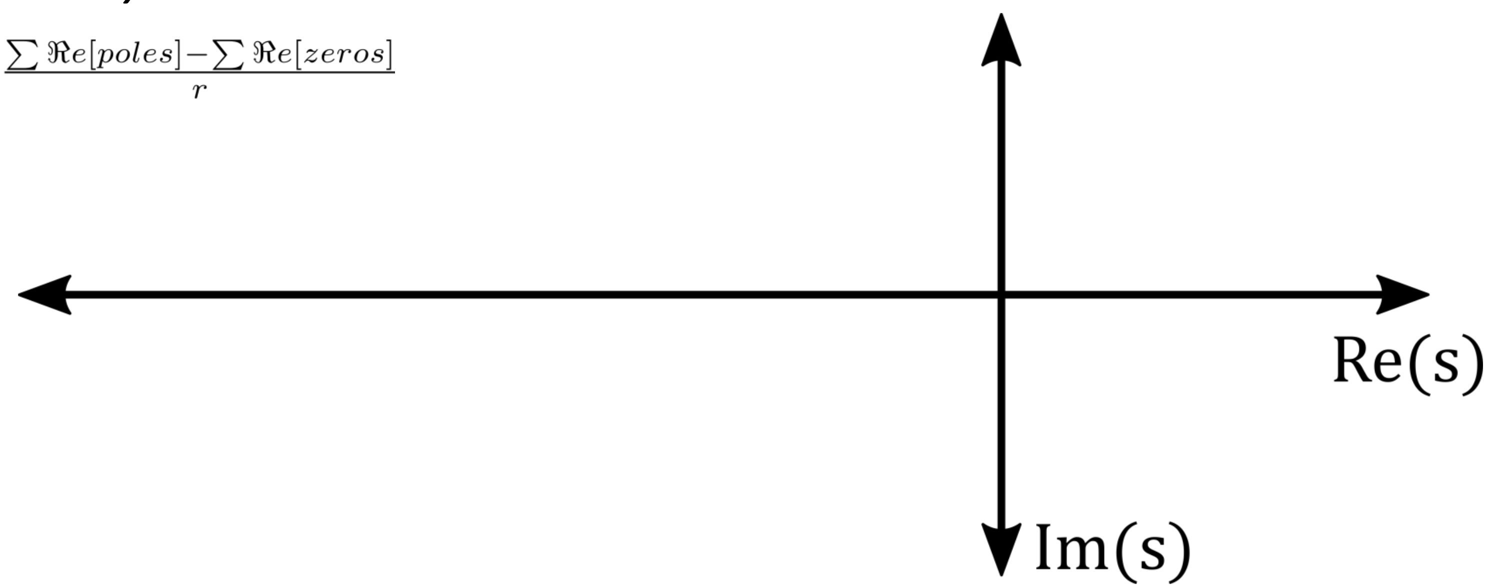
Example: Consider a OLTF with the following poles and zeros:

Poles: $s = -1, s = -2, s = -3$

Zeros: $none$

Q: is this system stable?

$$\sigma = \frac{\sum \Re e[poles] - \sum \Re e[zeros]}{r}$$



Root Locus Examples

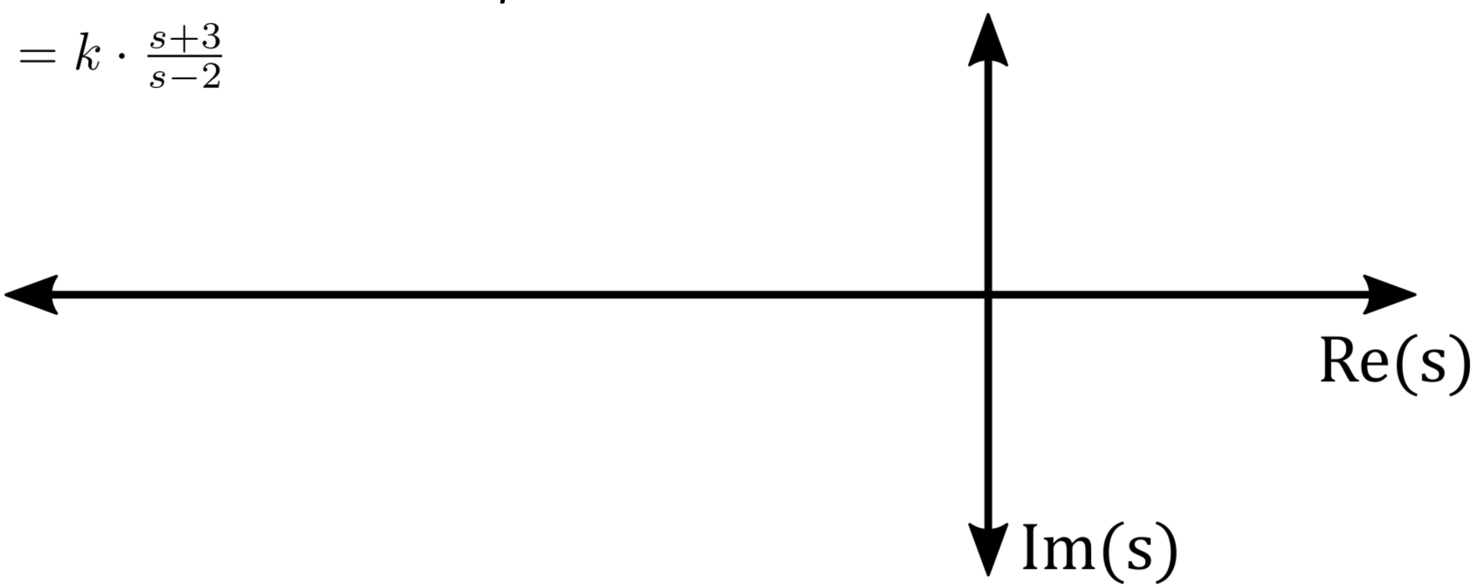
Example: Consider a OLTF with the following poles and zeros:

Poles: $s = 3, s = -4$

Zeros: $s = 1$

Q: what controller could we pick to stabilize?

$K(s) = k \cdot \frac{s+3}{s-2}$



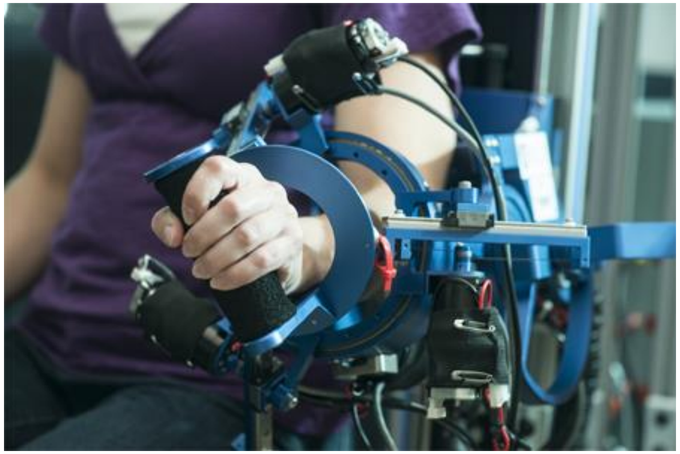
Physical Human-Robot Interaction

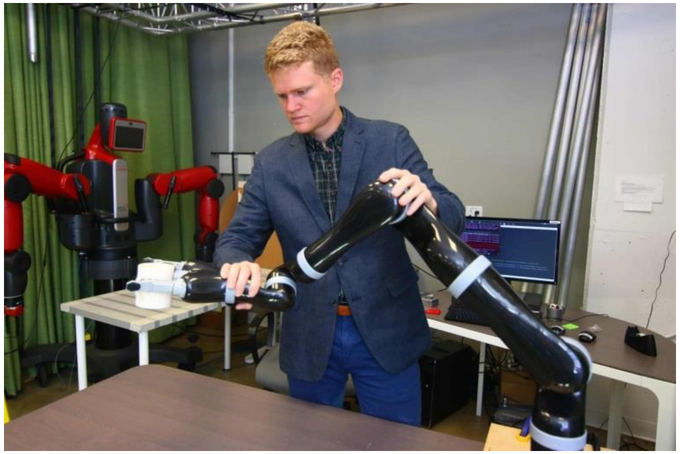
- We have thought about stability for the robot joints in isolation
- But what about the stability of connected (coupled) systems?

Definition: *Physical human-robot interaction (pHRI)* occurs when both the human and robot are in direct physical contact

- Applications: rehabilitation, prosthetics, surgery, exoskeletons, wheelchairs, cars, co-manipulation

Remark: When a human and robot are physically interacting, we *cannot claim that the coupled system is stable* given that the robot and human are stable in isolation.





Passivity

Theorem: A system formed out of two connected systems in a feedback loop is *stable* if both of the connected systems are *passive*

Remark: It is standard to model the human as passive. Hence, *pHRI is stable if the robotic system is passive*!

Definition: A system is passive if it dissipates or conserves energy. A system with input "force" $u(t)$ and output "velocity" $y(t)$ is *passive* if:

$$\int_0^T u(t) \cdot y(t) \ dt \geq 0$$

where this gives the total energy dissipated by the system over time T.

Remark: A passive system is also a stable system; a stable system is not necessarily passive

- When designing controllers for pHRI applications, passivity is a desirable quality. However, it leads to conservative controllers.

Passivity

- There is a nice test for passivity in linear time-invariant (LTI) systems

Theorem: an LTI system is passive if its transfer function $H(s)$ is *positive real*

- Necessary and Sufficient conditions for Positive Realness:

1. TF $H(s)$ has no poles in the right half-plane (stability)
2. The real part of $H(s)$ is nonnegative along the $i\omega$ axis:

$$H(s) = \frac{B(s)}{A(s)}$$

$$Re\{B(i\omega)A(-i\omega)\} \geq 0 \quad \forall \omega \geq 0$$

Example: is a mass-damper with P control (on velocity error) passive?

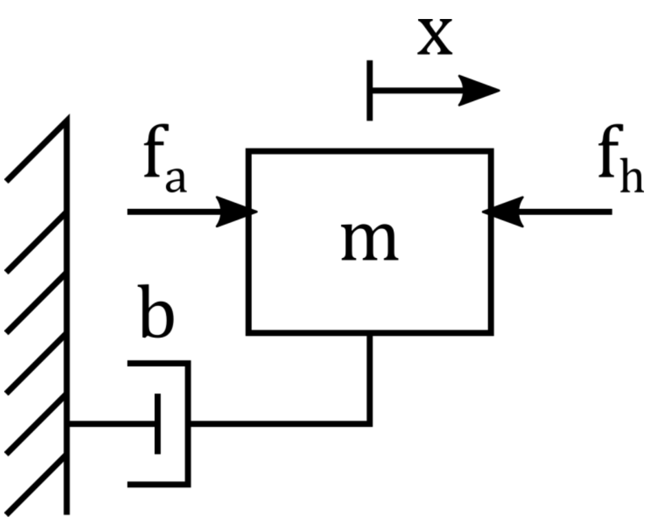
$$H(s) = \frac{k_p}{ms^2+bs+k_p}$$

$$Re\{k_p \cdot [m(-i\omega)^2 + b(-i\omega) + k_p]\}$$

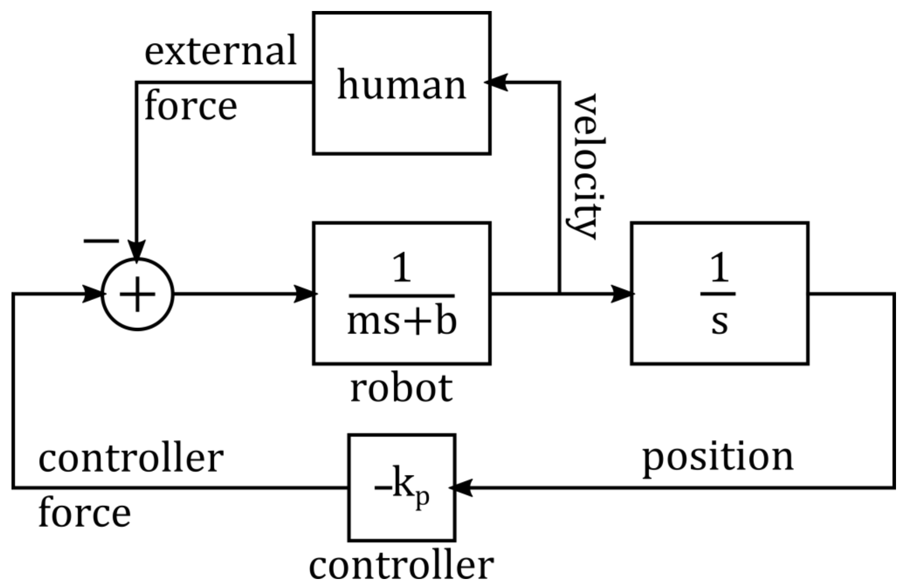
$$Re\{mk_p\omega^2 - bk_p\omega i + k_p^2\}$$

Passivity

- Let’s consider the simplest pHRI system using passivity
- Single (linear) joint trying to maintain $x = 0$ using P control
- Like rendering a “virtual wall” or “virtual stiffness” to the human



$$f_a - f_h = (ms + b)\dot{x}$$
$$f_a = -k_p x$$



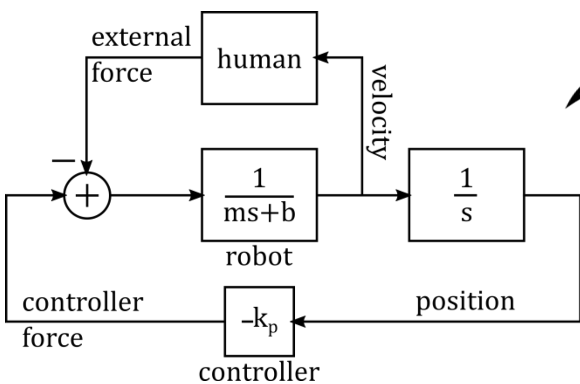
Passivity

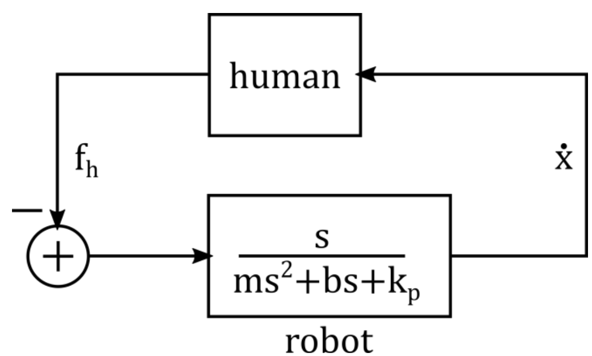
- We can reduce the block diagram to a coupled system
- The human is in *impedance*, and the robot is in *admittance*

Definition: (Mechanical) *impedance* relates input velocity to output force, and *admittance* relates input force to output velocity

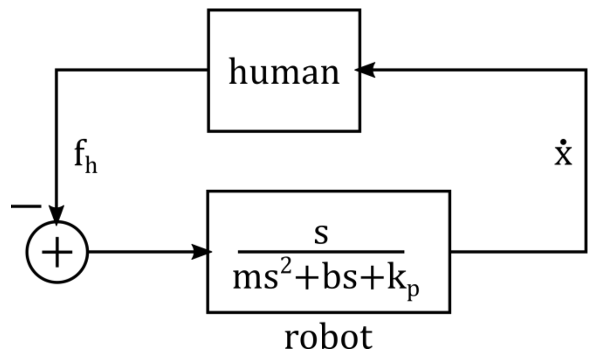
$$f_a - f_h = (ms + b)\dot{x} \qquad -f_h = (ms^2 + bs + k_p)x$$

$$f_a = -k_p x \qquad \frac{-\dot{x}}{f_h} = \frac{s}{ms^2+bs+k_p}$$





Passivity



- Recall our original definition of passivity and the positive realness test
- To ensure that the robot is passive, here we need (in the *time domain*):

$$\int_0^T -f_h(t) \cdot \dot{x}(t)\ dt \geq 0$$

- Since the LTI system is stable, we can check (in the *frequency domain*):

$$Re\{B(i\omega)A(-i\omega)\} \geq 0 \quad \forall \omega \geq 0$$

$$Re\{i\omega \cdot [m(-i\omega)^2 + b(-i\omega) + k_p]\} \geq 0 \quad \forall \omega \geq 0$$

$$b\omega^2 \geq 0 \quad \forall \omega$$

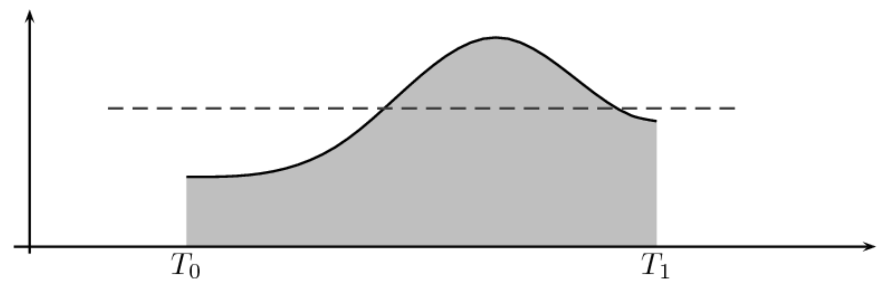
Q: So what's the big deal?

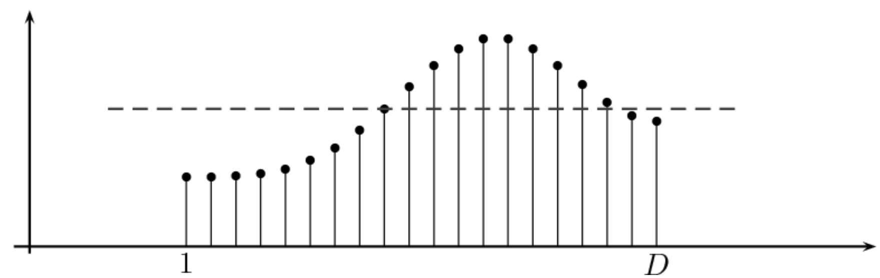
Discretization

- So far we have assumed that the controller (computer interface) operates in continuous time
- But that's not actually true!
- Controller samples sensor and determines input torques with a discrete sampling period T

Assumption: the computer sampling rate T is small enough ($T \rightarrow 0$) that we can approximate the controller as operating in *continuous time*

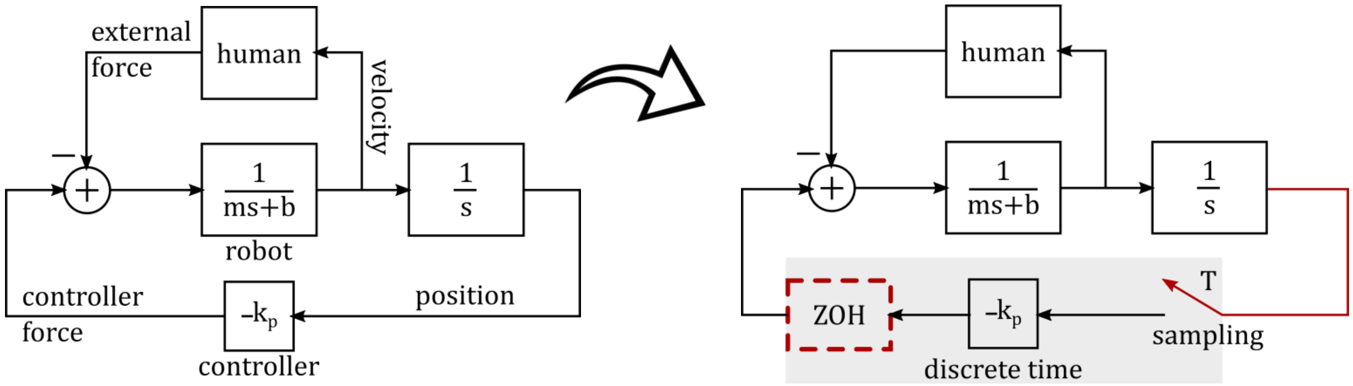
- Let's see what is different when we remove this assumption





Remark: there are other practical issues during controller implementation, such as *quantization, time delays, and amplifier dynamics*

Discretization

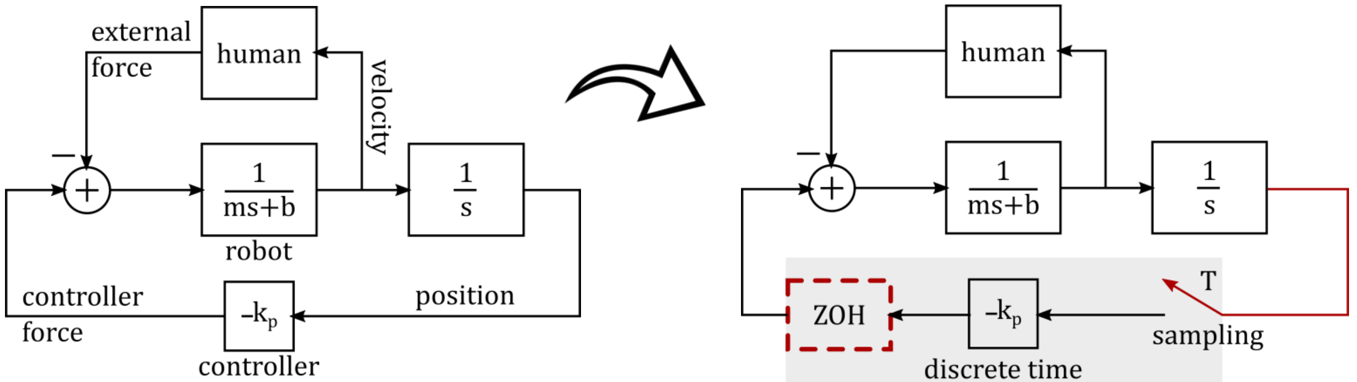


- The computer interface:
  - Samples the sensor to measure position
  - Determines control signal from sampled position
  - Applies that input for the current timestep

Remark: while the computer operates in discrete time, the physical robot moves in continuous time; robots are therefore *hybrid systems.*

Remark: discretization can have an impact on controller stability, and in particular on controller passivity.

Discretization



Theorem: If we approximate the controller as continuous time, the robot is passive for any b such that:

$$b \geq 0$$

- Recall that we have proven this
- Intuitive; the system can only dissipate energy with damping

Theorem: If we account for discretization, such that the robot has sampling period T, the robot is passive if and only if:

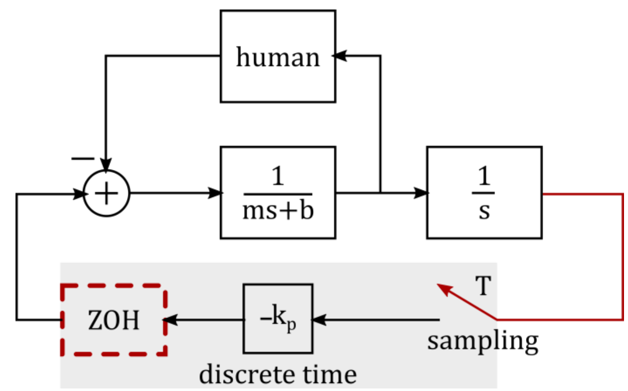
$$b \geq \frac{k_p T}{2}$$

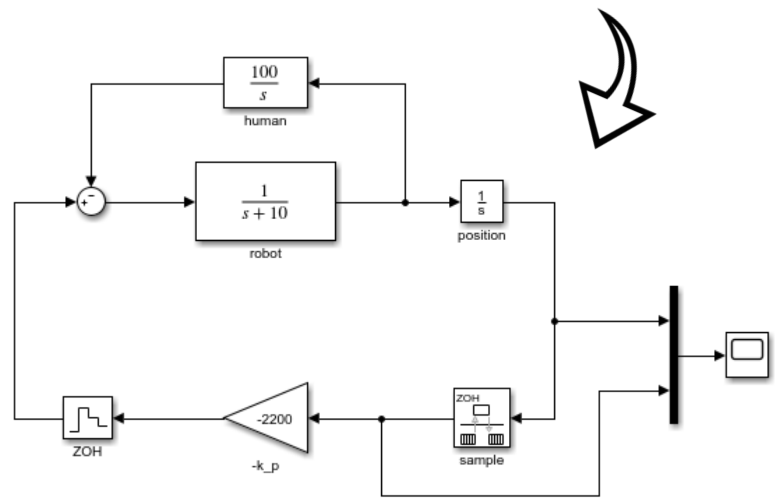
- Passivity is affected by wall stiffness
- What happens as $T \rightarrow 0$?

Discretization

Example: let's test the effects of discretization in simulation

- Render a virtual wall k_p
- Robot sampling at $T = 0.01$s
- Human modeled as a spring with stiffness $k_h = 100$ N/m
  - Note that a spring is passive, and so **the human is a passive system**
- Robot is a mass-damper (1-DoF prismatic robotic joint)
- Can use simulink (MATLAB)
  - "rate transition" to sample
  - "ZOH" to go back to continuous time



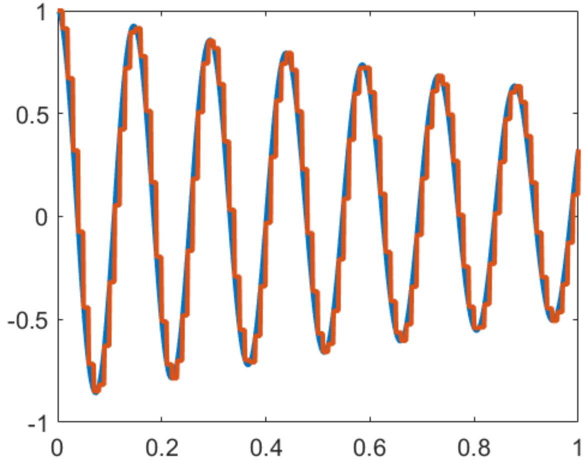


Discretization

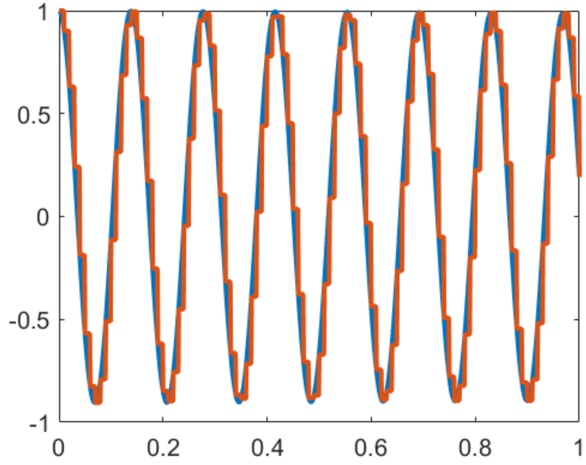
Results: the system is *stable* for $k_p \leq 2030$ N/m, but the human and robot become *unstable* when $k_p > 2030$ N/m

- This is because the robot subsystem is not passive for $b > 2000$ N/m
- Note that we have not chosen the most “destabilizing operator”

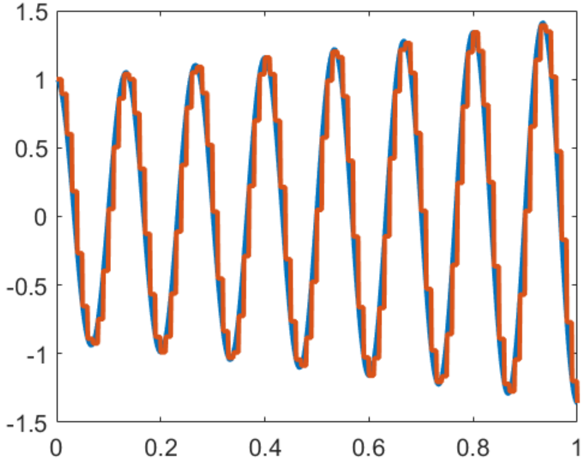
$k_p = 1800$



$k_p = 2030$



$k_p = 2200$



Passivity and Discretization

- When making robots for pHRI, we need *safe* controllers
- A stable controller may become unstable when coupled to a human operator, even a passive human operator
- We can design safe controllers using *passivity*
  - A great way to test passivity for LTI systems is positive realness
  - We can also use time-domain tests to check passivity
- During implementation, our controller operates in *discrete time* (as opposed to continuous time)
- Often we can ignore discretization if we sample fast enough
- In practice, however, discretization can make a normally passive system non-passive, and result in *unsafe pHRI*

Passivity and Discretization

Modern (Linear) Control Theory

Classical Control

1. Joint space: q
2. Laplace domain
3. Transfer function:

$$Y(s) = H(s)U(s)$$

4. Provides more general insight
5. MIMO design is difficult

Modern Control

1. State space: $x = (q, \dot{q})$
2. Time domain
3. State and output equation:

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

4. Provides more specific insight
5. MIMO is straightforward!

Example: Let's write the state equation and output equation for a mass-spring damper with input $f(t)$, and we observe position

Transfer Function vs. State Space

Theorem: the mapping from state space (SS) to a transfer function (TF) is *unique* (injective), and is given by the following equation:

$$\frac{Y(s)}{U(s)} = C(sI - A)^{-1}B + D$$

- Returning to our mass-spring-damper example:

$$A = \begin{bmatrix} 0 & 1 \\ -k/m & -b/m \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 1/m \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad D = 0$$

$$(sI - A)^{-1} = \frac{1}{s^2 + bs/m + k/m} \begin{bmatrix} s + b/m & 1 \\ -k/m & s \end{bmatrix}$$

Remark: the mapping from TF to state space is *non-unique* (non-injective), since the SS representation includes more information than a TF

Stability

- From the last example, we see that the denominator of the TF is:

$$\det\left(sI - A\right)$$

- And so the characteristic equation becomes:

$$\det\left(sI - A\right) = 0$$

- But, this is the same equation that gives us the *eigenvalues* of matrix A

Theorem: the *poles* of an LTI system represented using state space are the *eigenvalues* of the passive dynamics matrix A.

- Recall that a LTI system is exponentially stable if all the poles are of the transfer function are in the left-half plane

Remark: A system represented using state-space is stable if all the eigenvalues of A are negative

- Note that techniques like root locus, bode, nyquist are no longer available!

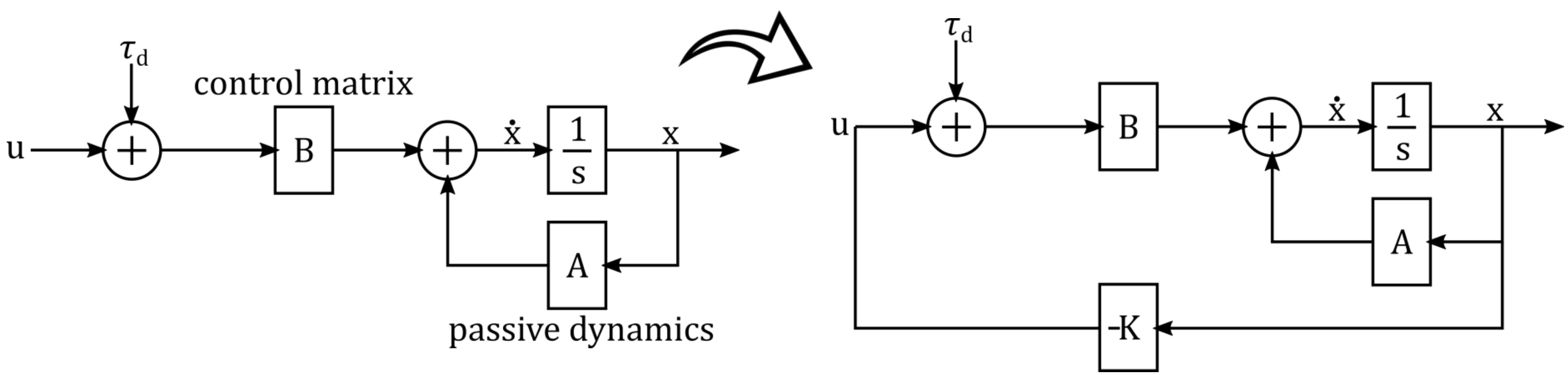
Stability

- We can place the CL poles using full-state feedback

Definition: in *full state feedback*, we measure all of the states so that the control input is a weighted sum of the states:

$$u = -Kx$$

Remark: full state feedback ($y = x$) is a very common type of state space controller in robotics, and generalizes PD control



Stability

Q: If we use full state feedback, where are the poles of the CL system?

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = x(t)$$
$$u(t) = -Kx(t)$$

Remark: the CL poles are the same for both tracking a reference and *regulation*, where the reference is zero.

Controllability

Definition: a system is *controllable* if any initial state $x(0)$ can be transferred to any final state $x(t_f)$ within some input $u(t)$

Theorem: a LTI system is (full state) *controllable* if and only if:

$$R = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}$$

is full rank, such that $rank(R) = n$. Note that n is the number of states.

Remark: this test also holds for MIMO systems!

- Controllability tells us whether we can move the robot anywhere in its state space, or if we cannot completely manipulate the state
- Controllability is a property of the physical system, not the controller
- We usually check for controllability before thinking about stability
- The dual concept to controllability is observability

Definition: a system is *observable* if every state $x(0)$ can be determined from observing $y(t)$ and our applied input $u(t)$ over a finite time interval

Stability

Theorem: if the system is (full-state) controllable, and we use full-state feedback, then *we can place the CL poles anywhere*

Example: Given the following state equation, design a full-state feedback controller so that the CL poles are $s = -2 \pm 4i$, $s = -10$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -5 & -6 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

1. Check for controllability --- “ctrb” in matlab
2. Find control gains to place CL poles:

$$\det(sI - A + BK) = (s+10)(s+2+4i)(s+2-4i)$$

Can solve for K using “place” in matlab

Optimal LTI Control

- Consider the system with full-state feedback:

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = x(t)$$

Q: What sensors do we need to implement this system on a serial robot?

- We can think of the robot as having an objective, or cost function, which it seeks to minimize during the task
- An optimal robot should choose actions u to minimize this cost function
- As a special (but common) case of this cost function, let the robot have a cost based on error and effort.

Definition: the robot is attempting to complete some task while minimizing the following *quadratic cost function*:

$$J = \int_0^\infty (x - x_d)^T Q (x - x_d) + u^T R u \; dt$$

Optimal LTI Control

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = x(t)$$
$$J = \int_0^\infty (x - x_d)^T Q (x - x_d) + u^T R u \; dt$$

What are these terms? Q is an $n \times n$ matrix that expresses the cost of trajectory errors, and R is a $m \times m$ matrix that expresses the control cost

Remark: intuitively, the robot is attempting to follow the desired trajectory as *closely as possible*, while applying the *smallest* amount of control inputs

Remark: this is a common problem (applications in pHRI), which leads to robots that move efficiently and safely

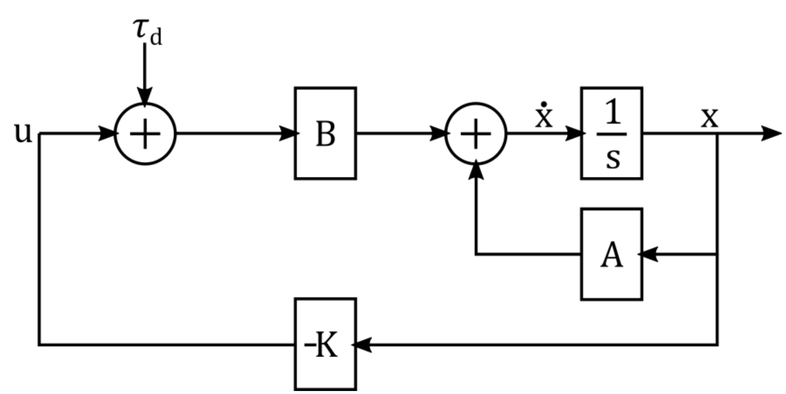
Q: What is the control law $u(t)$ that will minimize the cost function J given the LTI state and output equations?

Optimal LTI Control

Theorem: the robot's optimal control input is a given by a *Linear-Quadratic Regulator*, which is a feedback controller

- The robot's control law is: $u = -Kx$
- Here K is given by: $K = R^{-1}B^{T}P(t)$
- And P is the solution to the continuous-time Riccati equation:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0$$



Optimal LTI Control

- Most robots we work with are controllable, and we use full-state feedback
- For these robot, we can (theoretically) place the CL poles anywhere
- So where should we place the poles?
  - Can't just put them infinitely far into the LHP because of torque limits (and other practical complications, remember discretization)
  - But we also want good performance…
- Using a LQR is a good way to choose where to place the poles so that we achieve tracking and minimal controller effort
- An LQR here is really a PD controller where we used an optimization procedure to select the gains

Remark: using optimal control theory to choose the "best" gains is only available when we use a state-space representation

Linear Control for Robots: Summary

- Control joints independently
- Can use either transfer function or state equation to design controller
- We design controllers in order to obtain the desired behavior
  - Control design entails placing poles and zeros, and choosing overall gain
- Closed-loop control is better for unknown environments…
- …but closed-loop control can introduce instability
- There are also practical considerations (like pHRI, discretization)
- Without choosing a controller, we can't make our robot move!

Homework

Two options:

1. **Discretization and passivity**
2. **Optimal LTI control**

Both will involve implementation in matlab/simulink