a place of mind

ELEC 341: Systems and Control

Lecture 15

Root locus: PID controller design

Course roadmap

a place of mind
UBC

Modeling	*Analysis*	*Design*
✓ Laplace transform ✓ Transfer function Models for systems ✓ • Electrical ✓ • Electromechanical ✓ • Mechanical ✓ Linearization, delay	✓ Stability ✓ • Routh-Hurwitz • Nyquist ✓ Time response ✓ • Transient ✓ • Steady state Frequency response • Bode plot	Design specs ✓ Root locus Frequency domain ✓ PID & Lead-lag Design examples

Matlab simulations

Important remarks on using MATLAB

- It is convenient to use ***MATLAB Control System Toolbox*** in practical controller design problems.
  - ❑ Hand-calculations are often impractical for real-life engineering problems.
- Theory that you have learned so far is still very important!
  - ❑ These theories include topics such as, Routh-Hurwitz, how to sketch RL, how to use RL for controller design, etc.
    - ➢ You have to detect errors, if any, in the results that MATLAB returns.
    - ➢ You have to interpret what MATLAB returns.
    - ➢ If we know theory, we can take full advantage of MATLAB to design controllers effectively.

SISO Design Tool in Matlab

- Graphical-user interface (GUI) that allows you to design compensators.
- Type “sisotool” in Matlab prompt.

```
>> sisotool
```

- Select “Root locus” from *View* → *Design Plots configuration*.

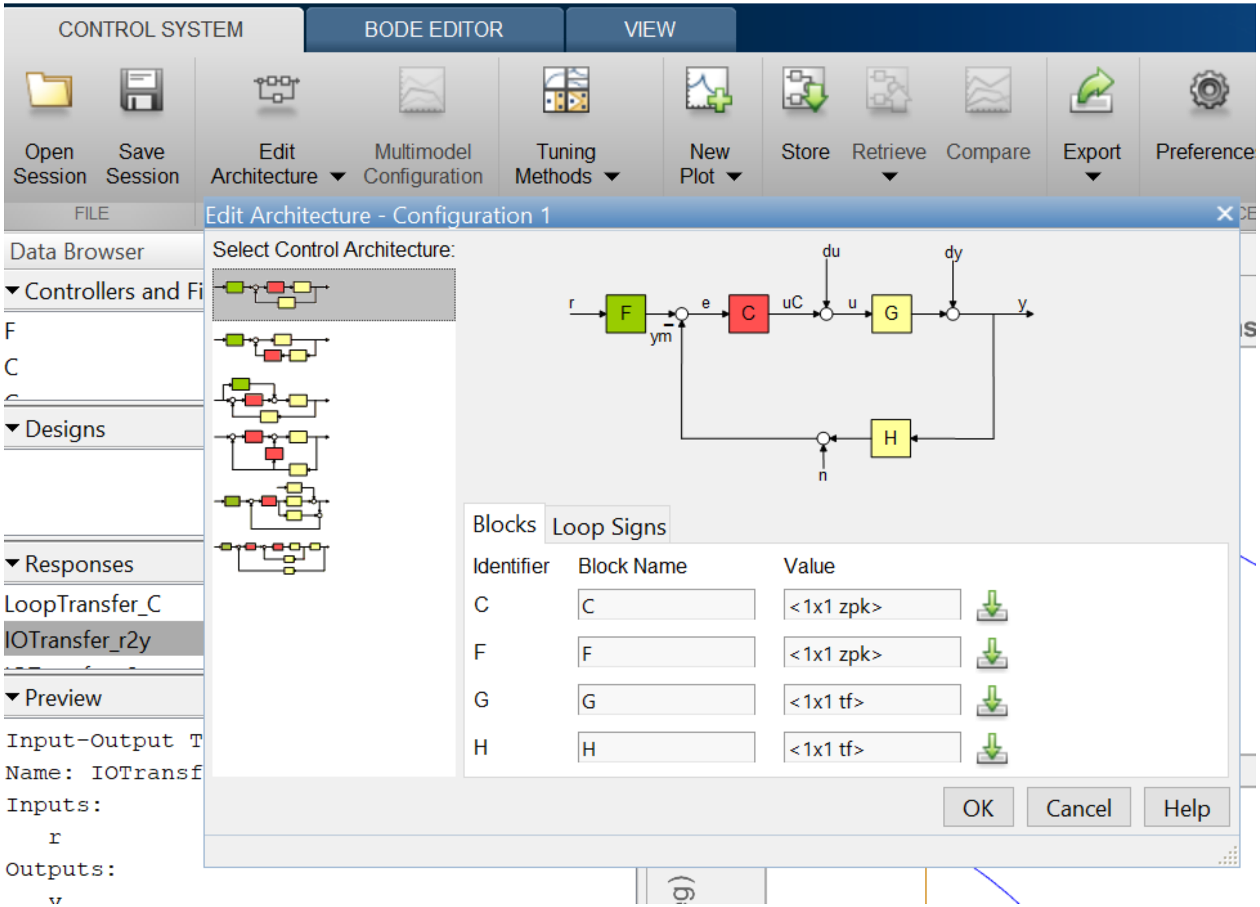
SISO Design Tool (Gain Compensator)

- Input the plant

```
>> sysG = tf(1,[1 2 0]);
```

or

```
>> s = tf('s');
>> sysG = 1/(s^2+2*s);
>> sisotool(sysG)
```

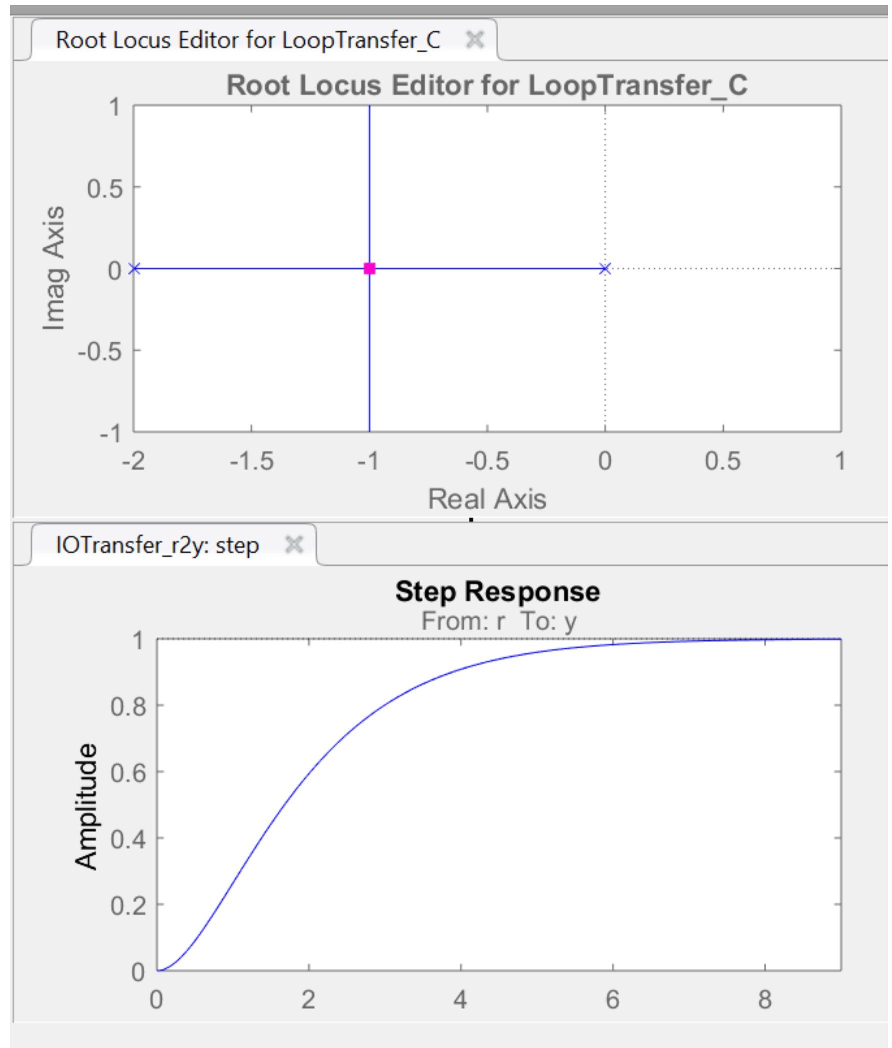


SISO Design Tool (Gain Compensator), (cont'd)

- You will see the root locus plot for

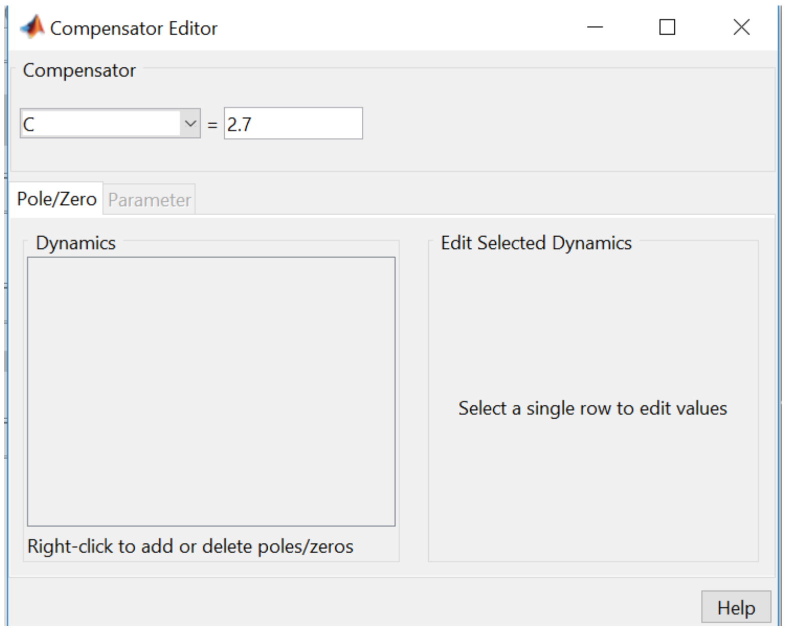
$$G(s) = \frac{1}{s(s+2)}$$

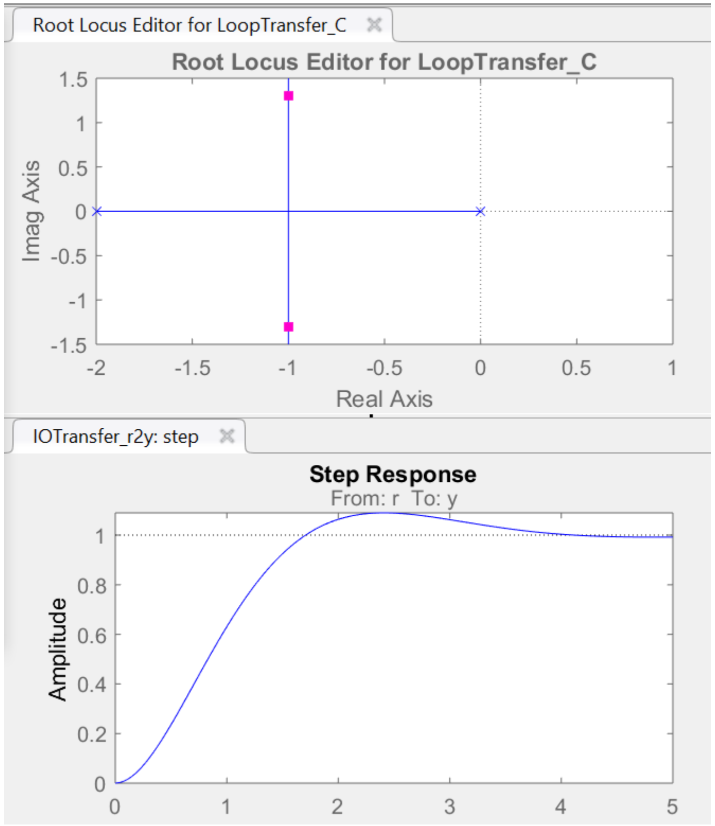
- The default setting is $C(s) = 1$.
- You can right-click on the plots to change the gain (among other things).



SISO Design Tool (Gain Compensator), (cont'd)

- Modify the gain for "good" step response.





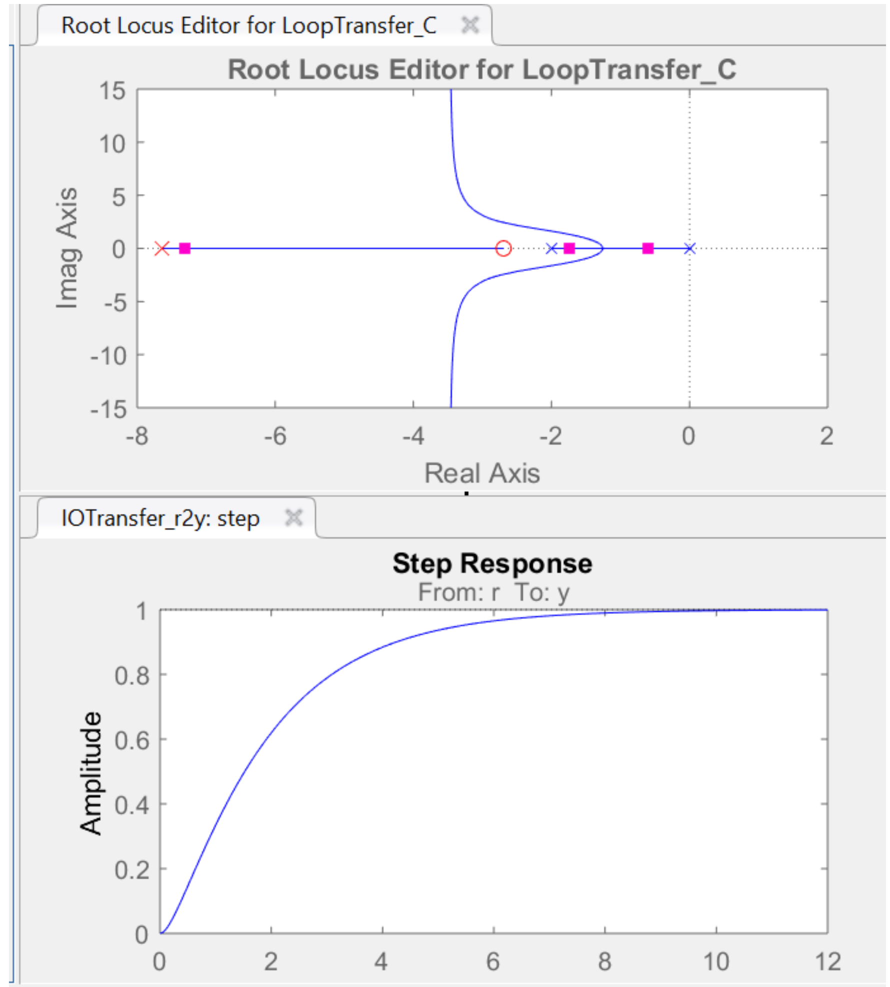
- By right-clicking on these figures, you can add various specs on the figures.

SISO Design Tool (Lead Compensator)

- Add a pole & a zero of a compensator, and adjust its gain:

$$C_{Lead}(s) = \frac{15.84(s+2.5)}{s+6.86}$$

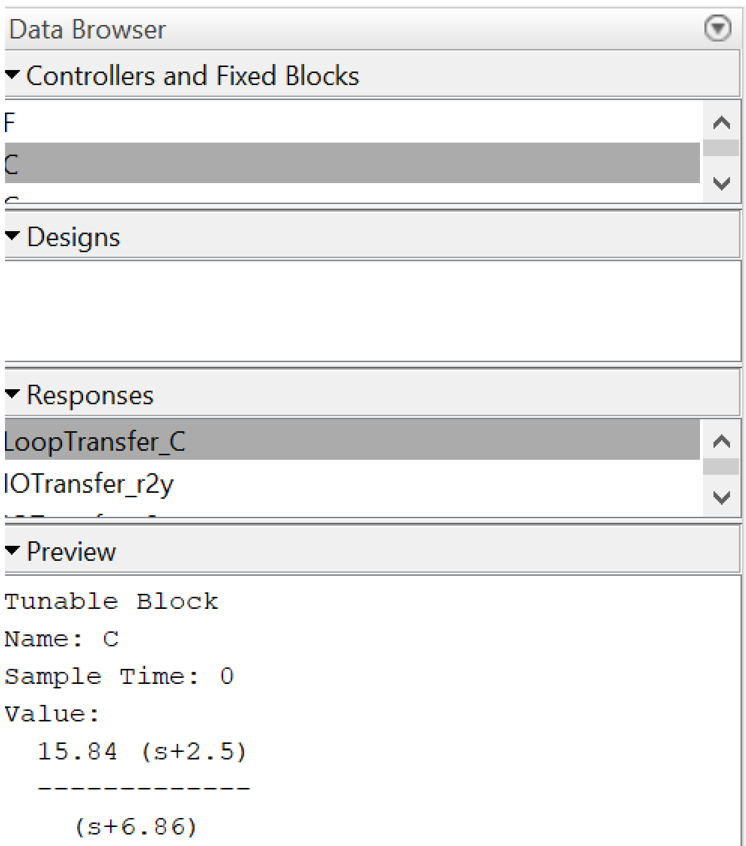
- If necessary, move the pole and zero
  - by click-and-drag, or
  - *Design → Edit Compensator…*

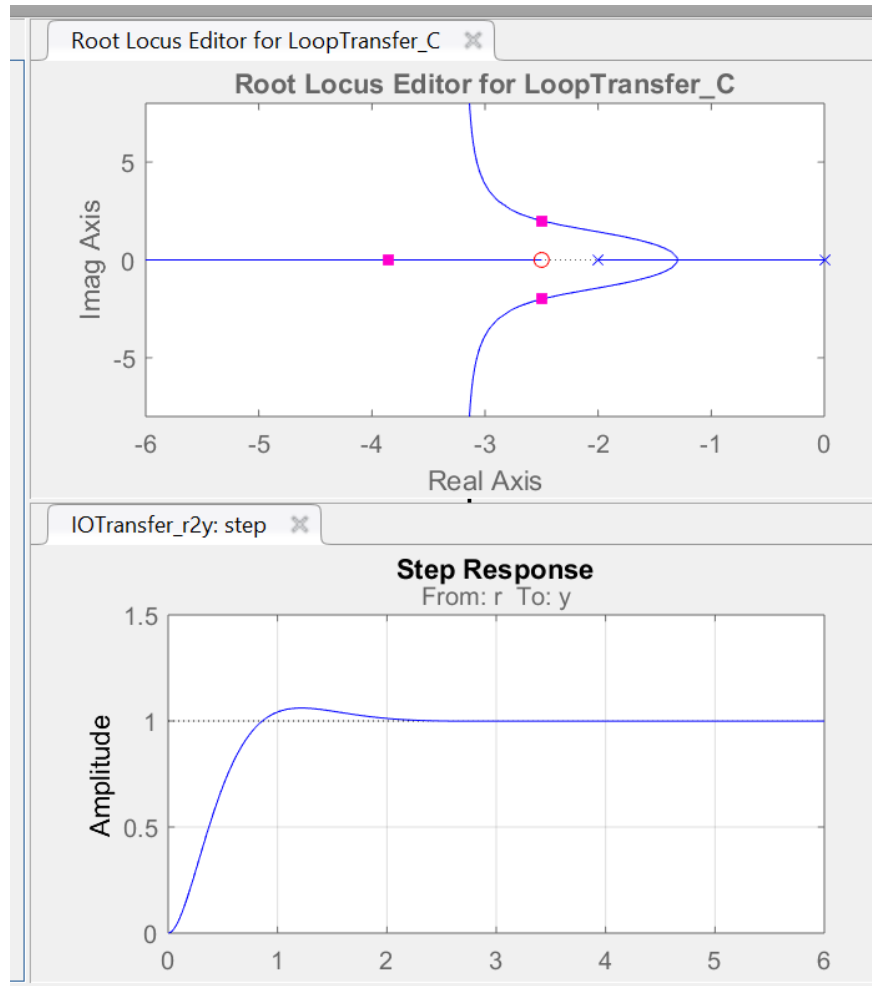


This is not with the gain we wanted (i.e., 15.84).

SISO Design Tool (Lead Compensator), (cont'd)

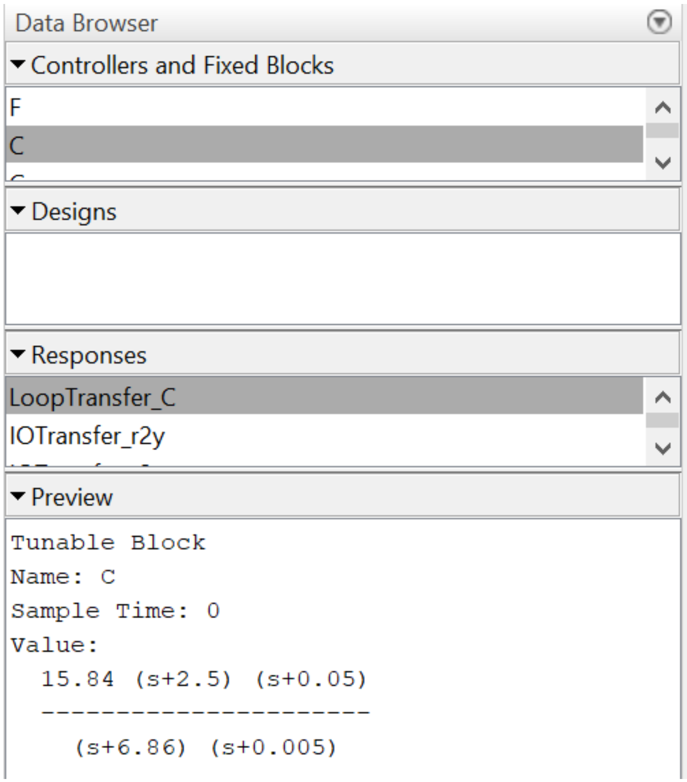
- Adjust the gain to 15.84 in order to obtain the final step response.

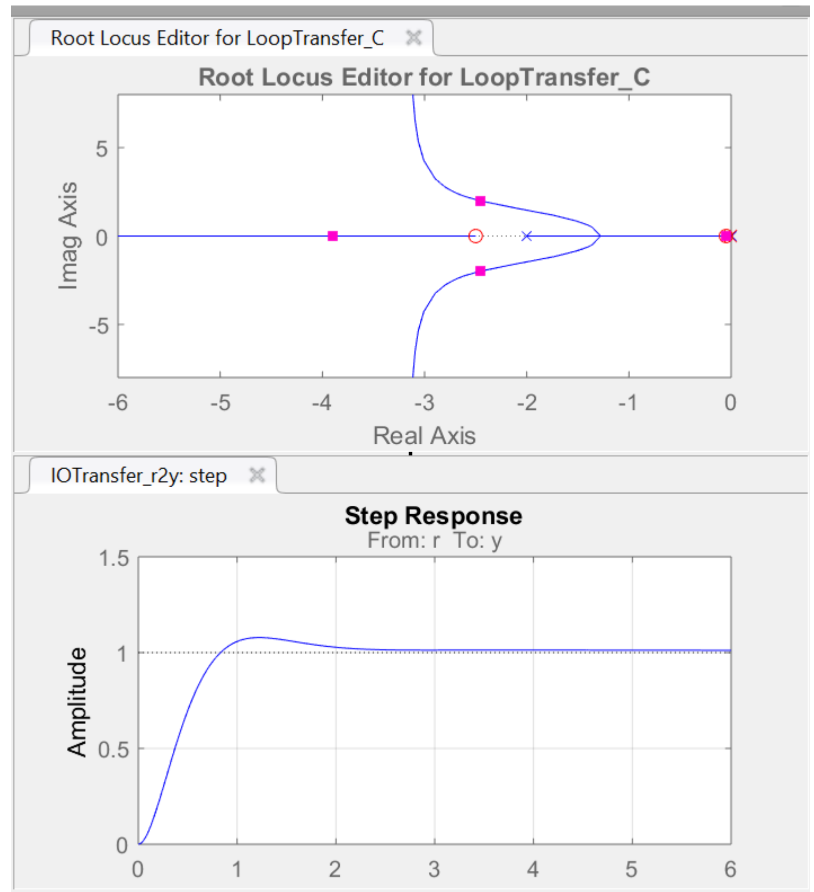




SISO Design Tool (Lead-Lag Compensator)

- ***Add*** a pole & a zero of $C_{Lag}(s) = \dfrac{s+0.05}{s+0.005}$
- Adjust controller gain to 15.84.





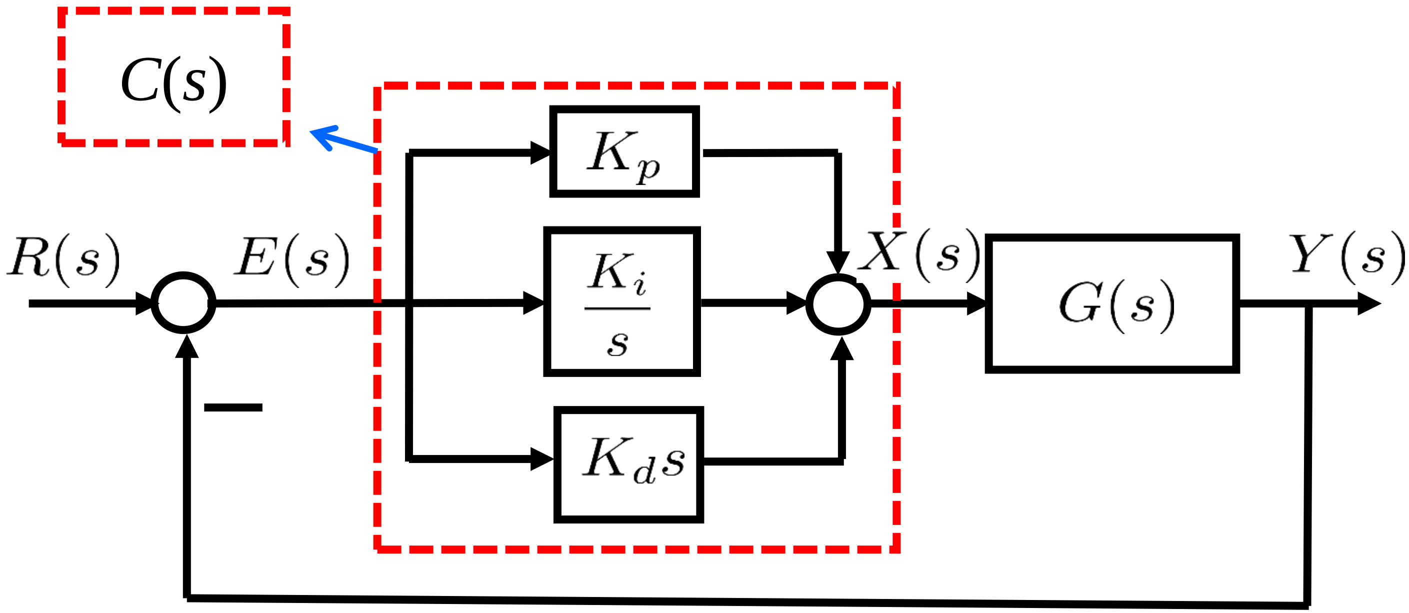
$$C_{LL}(s) = \underbrace{\frac{15.84(s+2.5)}{s+6.86}}_{C_{Lead}(s)} \times \underbrace{\frac{s+0.05}{s+0.005}}_{C_{Lag}(s)}$$

Useful MATLAB commands in Control System Toolbox

- sys=tf(num,den): Transfer function generation with numerator and denominator coefficient vector (num and den)
- pole(sys), zero(sys): Pole/zero calculations
- step(sys), impulse(sys): Step and impulse responses
- feedback(sys1,sys2): Calculation of closed-loop transfer function (Black's formula)
- rlocus(sys): Root locus drawing
- sisotool(sys): GUI for controller design

See the manual or help-command (e.g., >> help tf)

PID controller



t-domain: $$x(t) = \underbrace{K_p e(t)}_{\text{Proportional}} + \underbrace{K_i \int_0^t e(\tau)d\tau}_{\text{Integral}} + \underbrace{K_d \frac{de(t)}{dt}}_{\text{Derivative}}$$

s-domain: $$\underbrace{C(s) = K_p + \frac{K_i}{s} + K_d s}_{\text{Parallel form}} = \underbrace{K_p\left(1 + \frac{1}{T_I s} + T_D s\right)}_{\text{Standard form}}$$

$$T_D = \frac{K_d}{K_p};\quad T_I = \frac{K_p}{K_i}$$

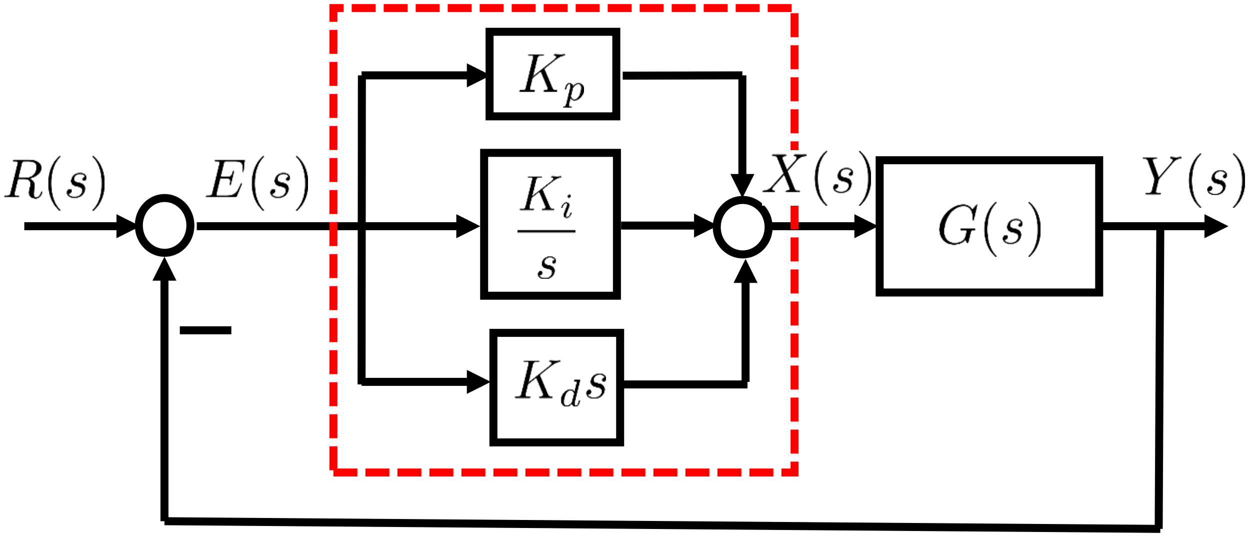
Notes on PID controller

- Most popular in process and robotics industries
  - Good performance
  - Functional simplicity (operators can easily tune.)
- To avoid high frequency noise amplification, derivative term is often implemented as

$$K_d s \approx \frac{K_d s}{\tau_d s + 1}$$

  with τ_d much smaller than plant time constant.
- PI controller $C(s) = K_p + K_i/s$
- PD controller $C(s) = K_p + K_d s$

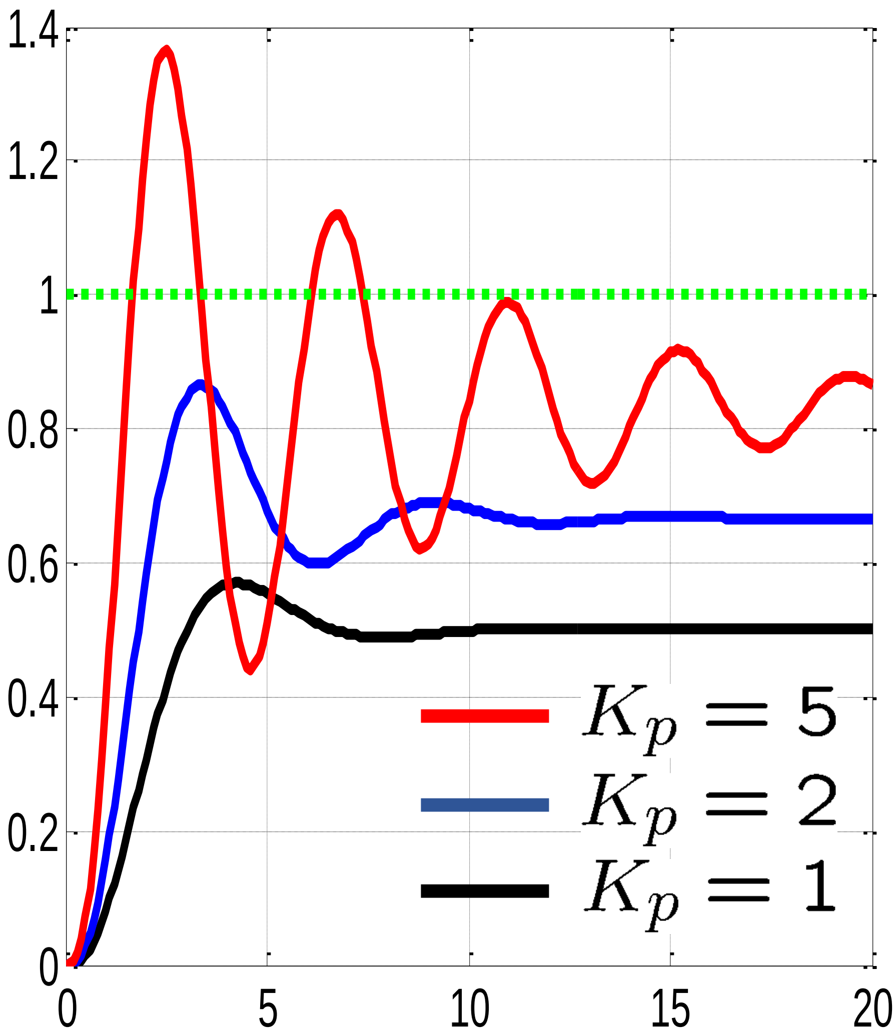
Example 1



$$G(s) = \frac{1}{(s+1)^3}$$

- Plot $y(t)$ for unit step input $r(t)$ with
  - P controller
  - PI controller
  - PID controller

Example 1 (cont'd)
P controller

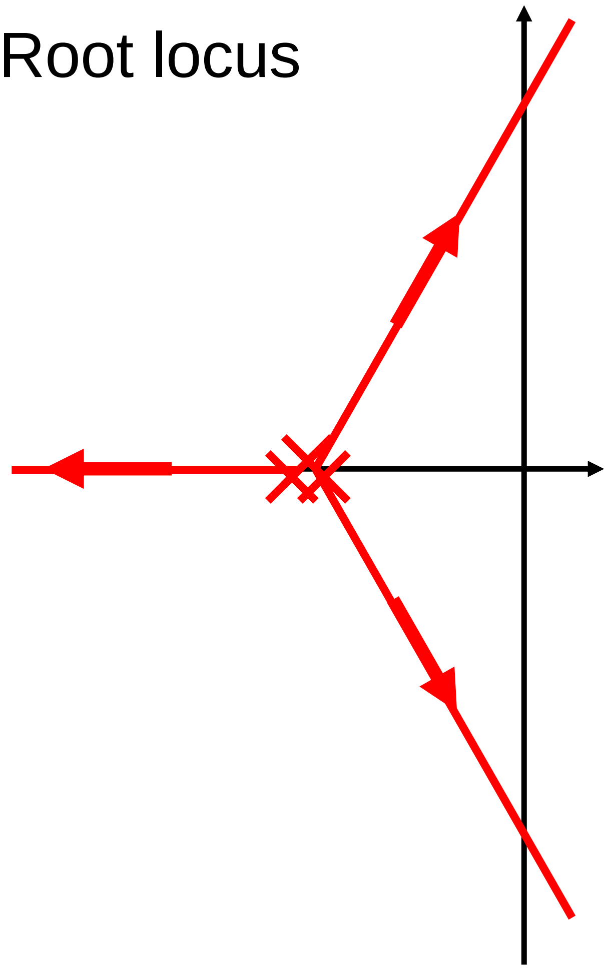


$$C(s) = K_p$$

- Simple
- Steady state error
  - Higher gain gives smaller SS error
- Stability
  - Higher gain gives faster (shorter rise time) and more oscillatory response

Example 1 (cont'd)
Interpretation: P controller

Root locus

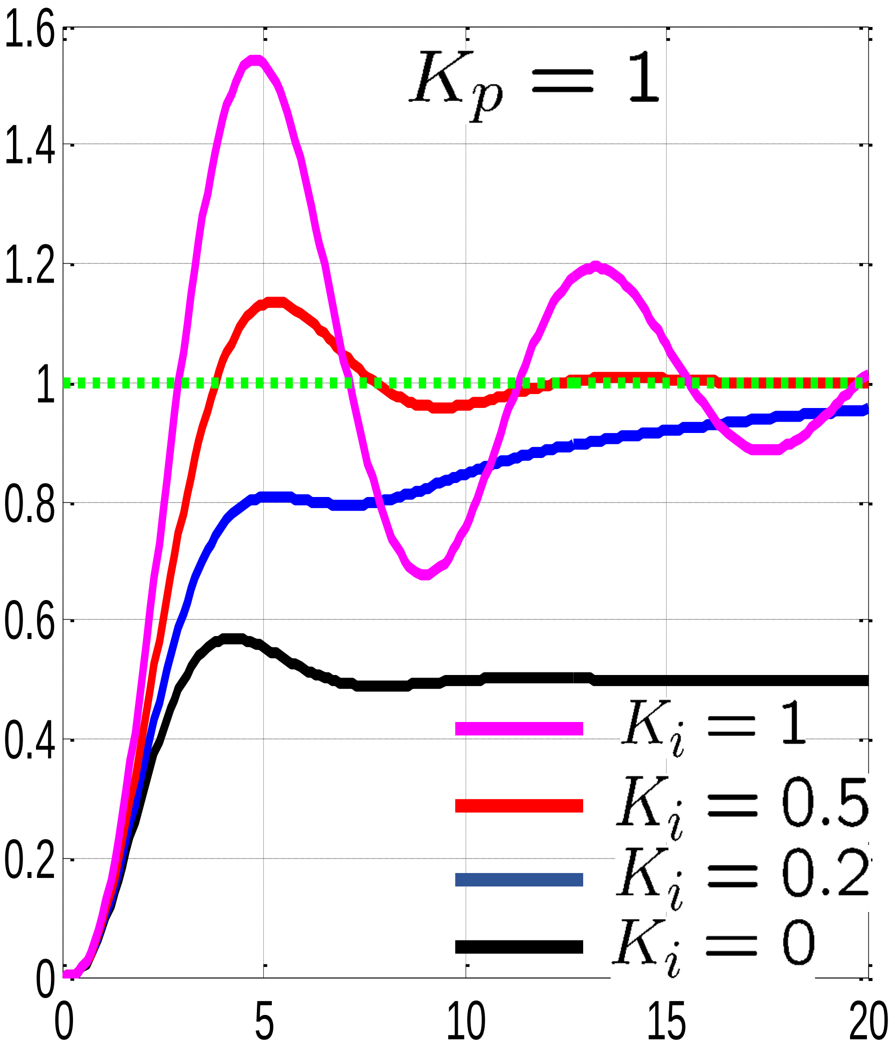


Steady state error

$$e_{ss} = \frac{1}{1+G(0)C(0)} = \frac{1}{1+K_p}$$

(This K_p is the P controller gain.)

Example 1 (cont'd)
PI controller

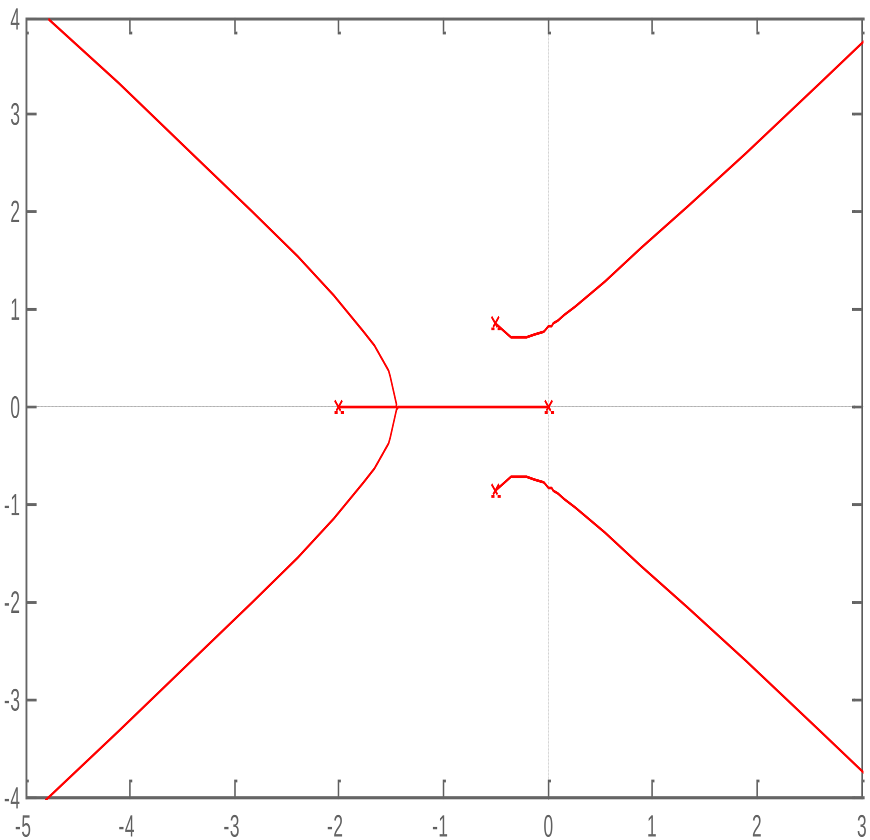


$$C(s) = K_p + \frac{K_i}{s}$$

- Zero steady state error (provided that CL is stable.)
- Stability
  - Higher gain (K_i) gives faster (shorter rise time) and more oscillatory response

Example 1 (cont'd)
Interpretation: PI controller

Root locus by increasing K_i



$$1 + \frac{1}{(s+1)^3} \cdot \frac{s+K_i}{s} = 0$$

$$\Rightarrow \quad s\left\{1+(s+1)^3\right\} + K_i = 0$$

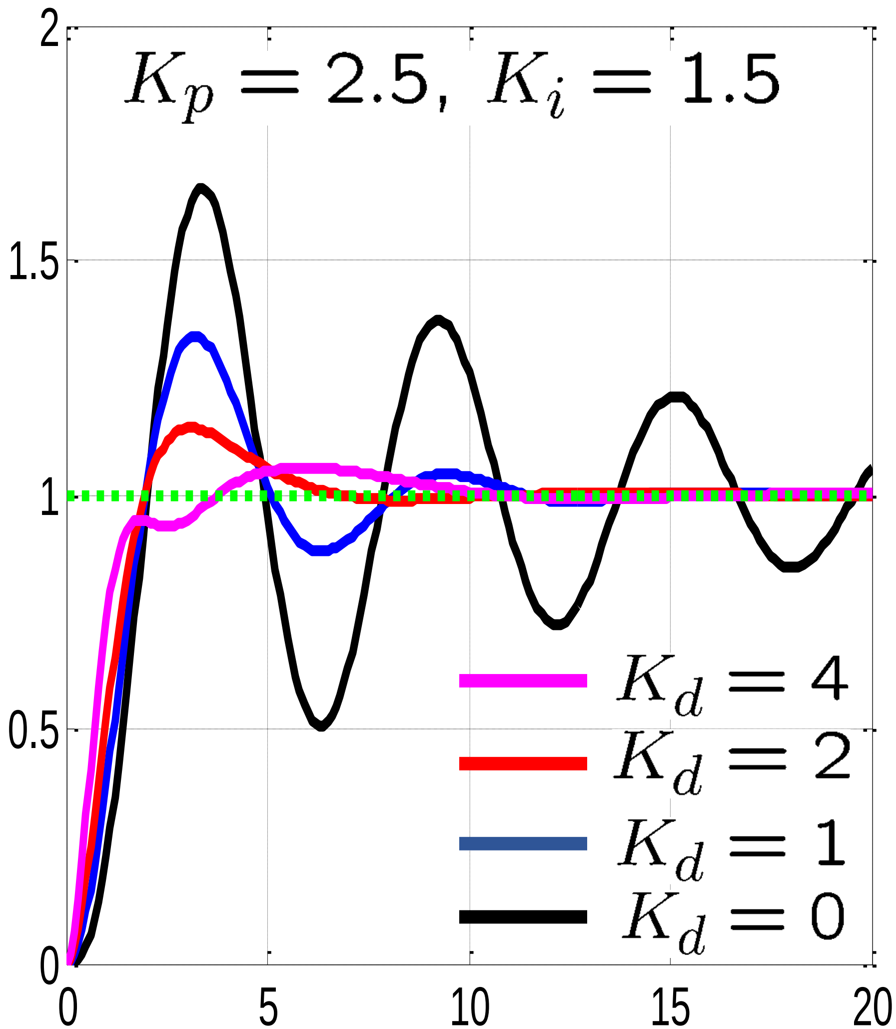
$$\Rightarrow \quad 1 + K_i \frac{1}{s(s+2)(s^2+s+1)} = 0$$

Steady state error

$$e_{ss} = \frac{1}{1+G(0)C(0)} = 0$$

(due to an integrator in $C(s)$)

Example 1 (cont'd)
PID controller

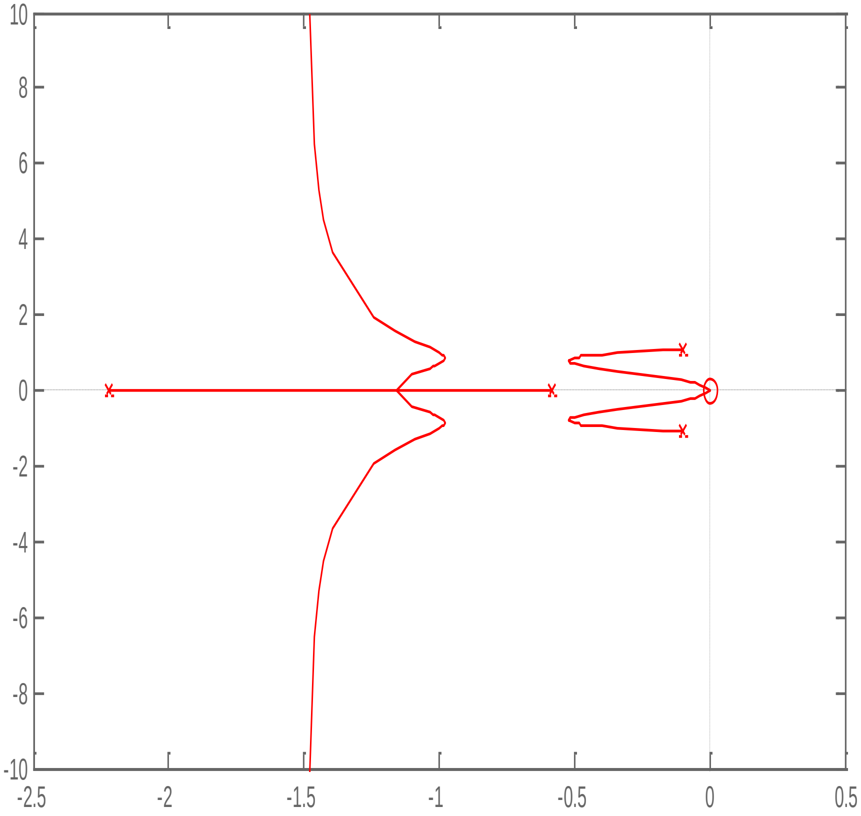


$$C(s) = K_p + \frac{K_i}{s} + K_d s$$

- Zero steady state error (due to integral control)
- Stability
  - Higher gain (K_d) gives more damped response.
- Too high gain (K_d) worsens performance.

Example 1 (cont'd)
Interpretation: PID controller

Root locus



$$1 + \frac{1}{(s+1)^3} \cdot \frac{K_d s^2 + 2.5s + 1.5}{s} = 0$$

$$\Rightarrow \quad 1 + K_d \frac{s^2}{s(s+1)^3 + 2.5s + 1.5} = 0$$

Steady state error

$$e_{ss} = \frac{1}{1 + G(0)C(0)} = 0$$

(due to an integrator in *C*(*s*))

a place of mind
UBC

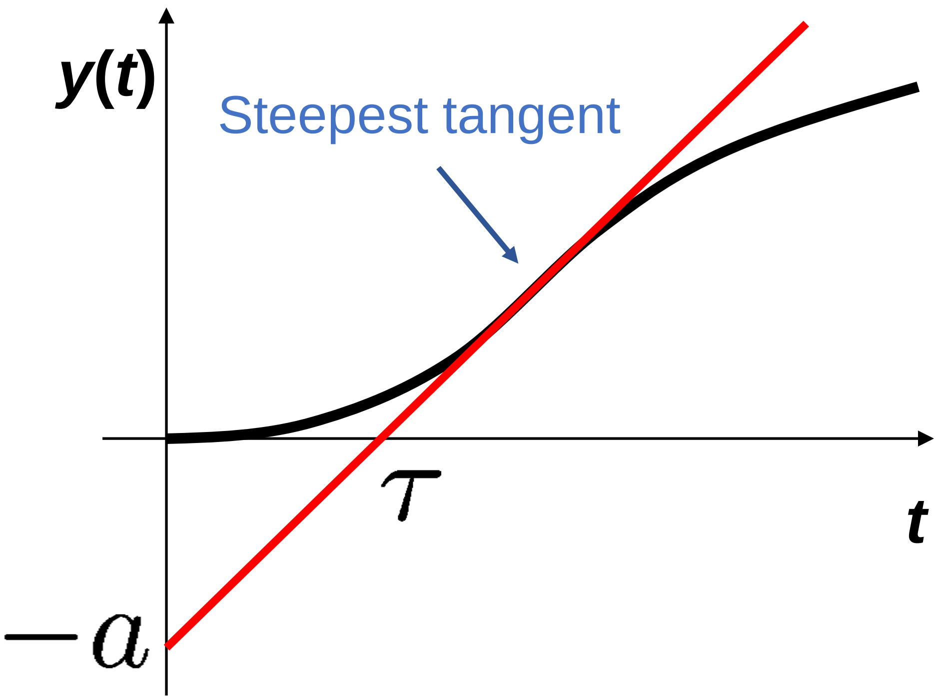
How to tune PID parameters

- Empirical (Model-free)
  - ❖ Trial and error
    - ➢ Useful only when trial-and-error tuning is allowed.
  - ❖ Ziegler-Nichols tuning rule
    - ➢ Useful even if a system is too complex to model.
- Model-based
  - ❖ Root locus (RL)
  - ❖ Frequency response (FR) approach
    - ➢ Both RL and FR are useful only when a model is available.
    - ➢ Both RL and FR become necessary if a system has to work at the first trial.

Ziegler-Nichols PID tuning rules

- **Step response method** (only for stable systems)

Open-loop step response → **PID parameters**



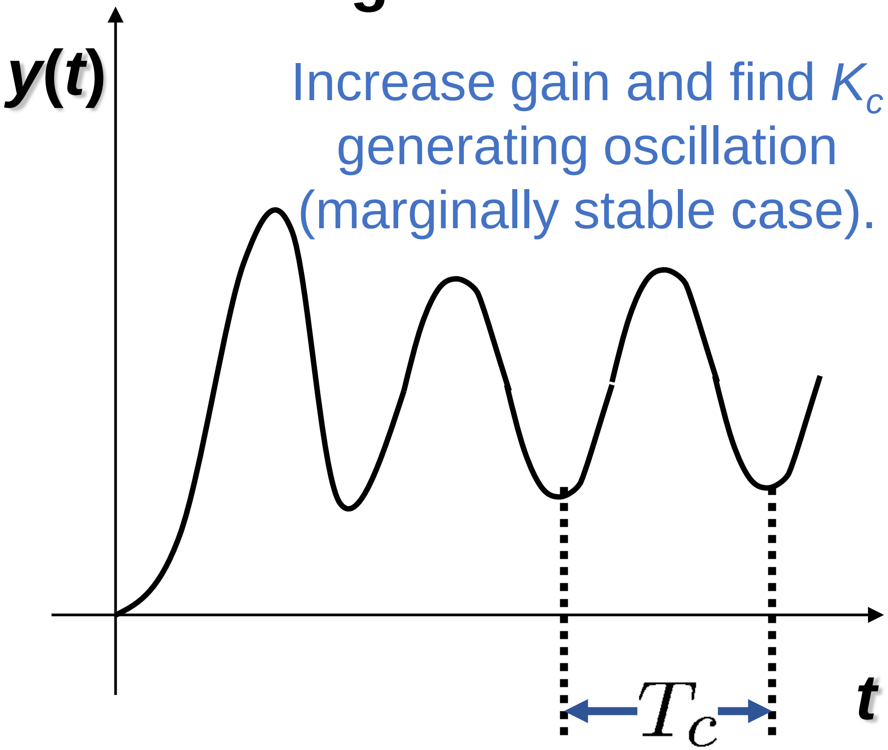
$$C(s) = K_p\left(1 + \frac{1}{T_I s} + T_D s\right)$$

Type	K_p	T_I	T_D
P	$1/a$		
PI	$0.9/a$	3τ	
PID	$1.2/a$	2τ	0.5τ

Ziegler-Nichols PID tuning rules

- **Ultimate sensitivity method**

Closed-loop step response with a gain controller → **PID parameters**



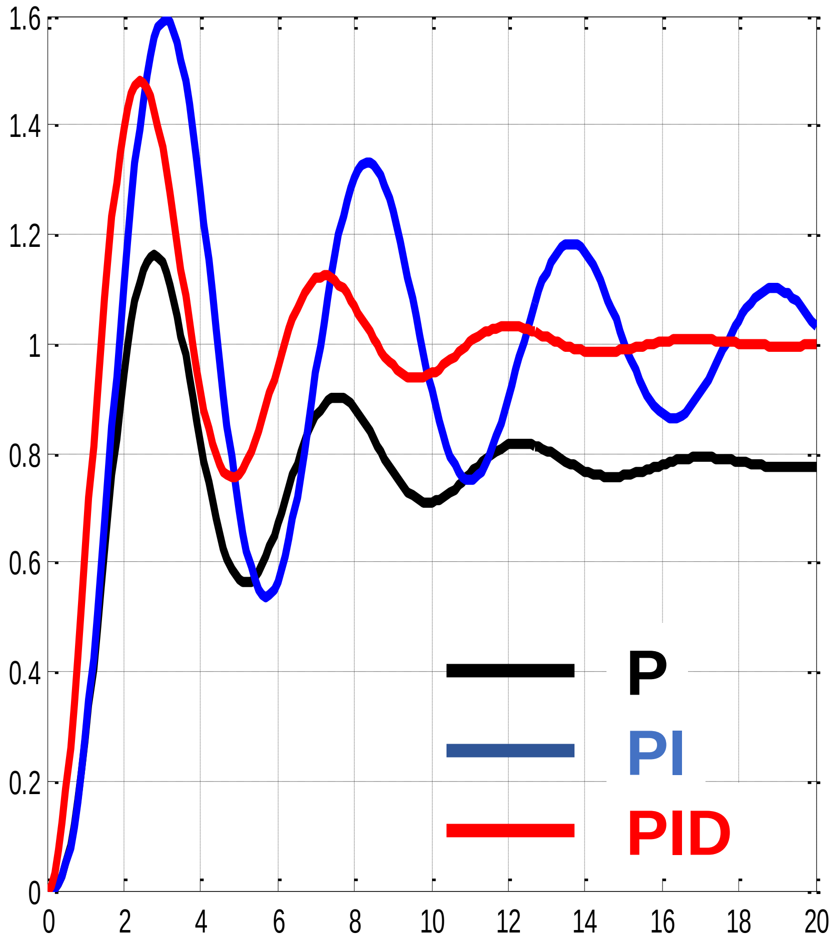
$$C(s) = K_p\left(1 + \frac{1}{T_I s} + T_D s\right)$$

Type	K_p	T_I	T_D
P	$0.5K_c$		
PI	$0.4K_c$	$0.8T_c$	
PID	$0.6K_c$	$0.5T_c$	$0.125T_c$

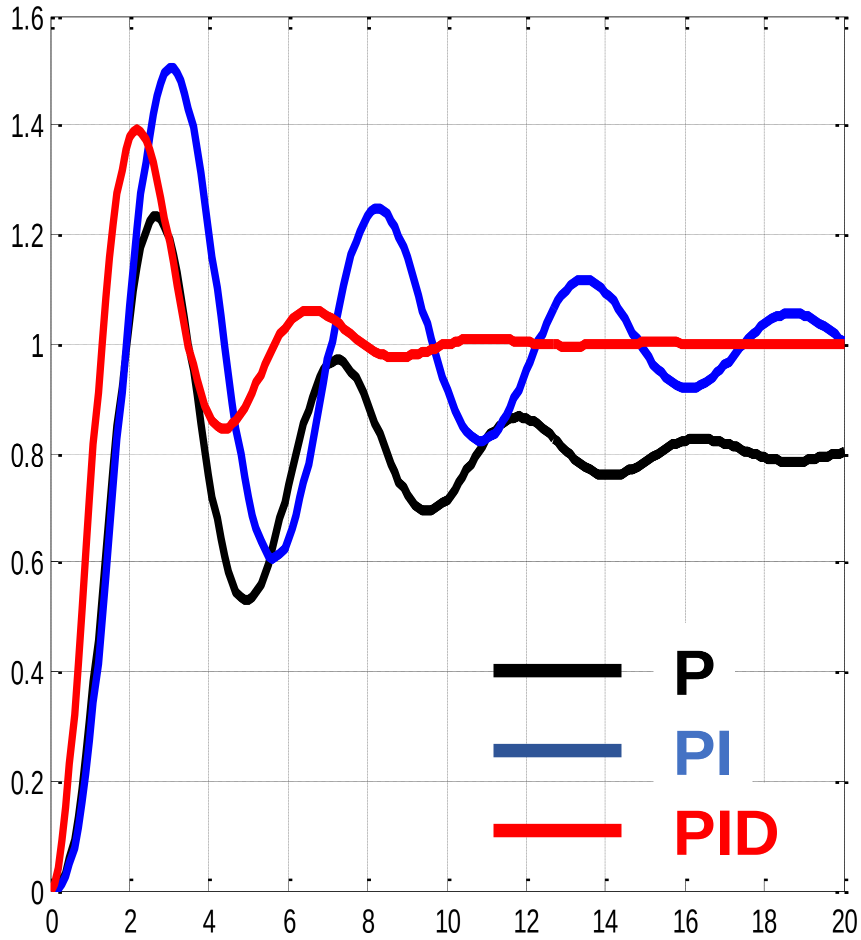
Example 1 (revisited) $G(s) = \dfrac{1}{(s+1)^3}$

a place of mind

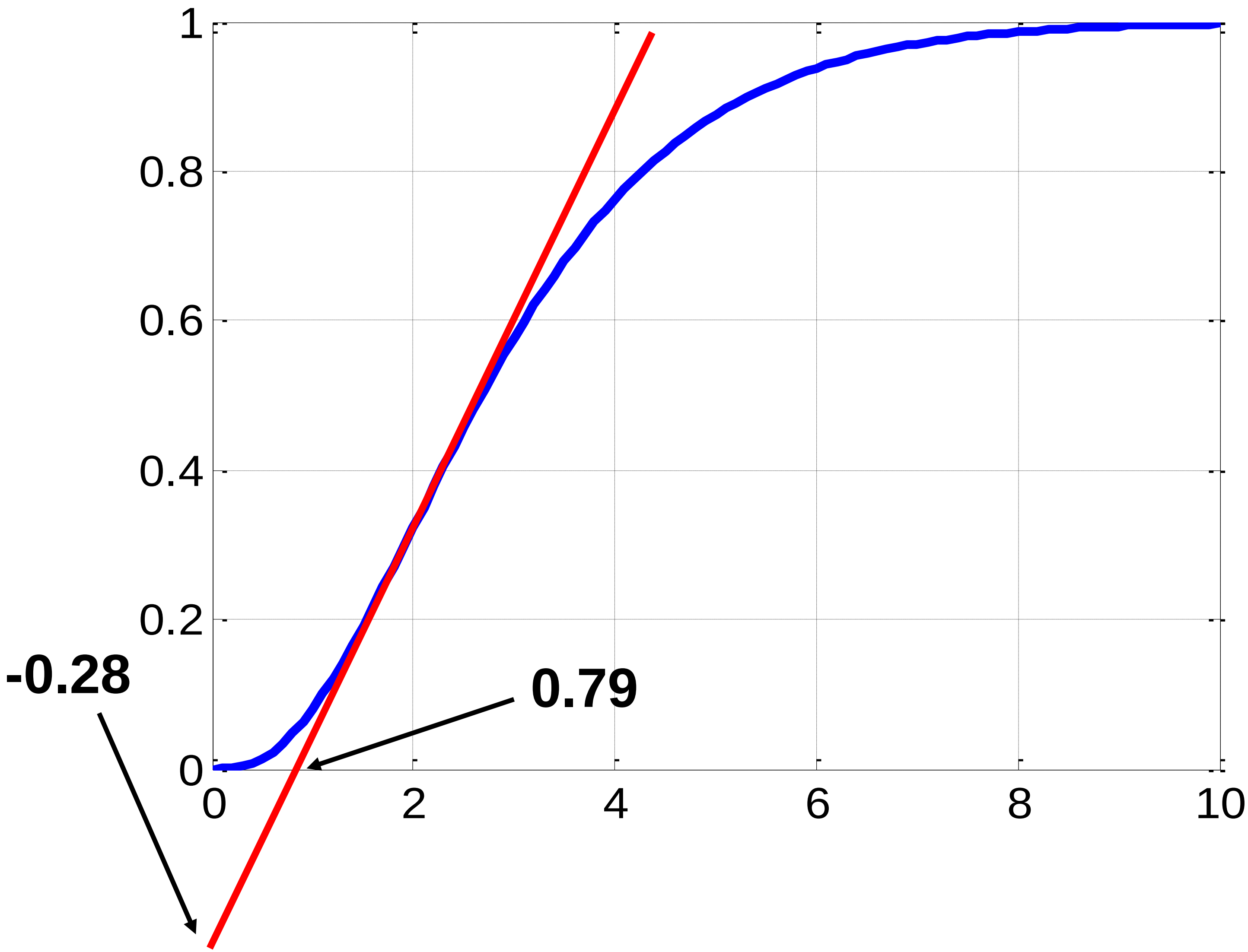
Step response method



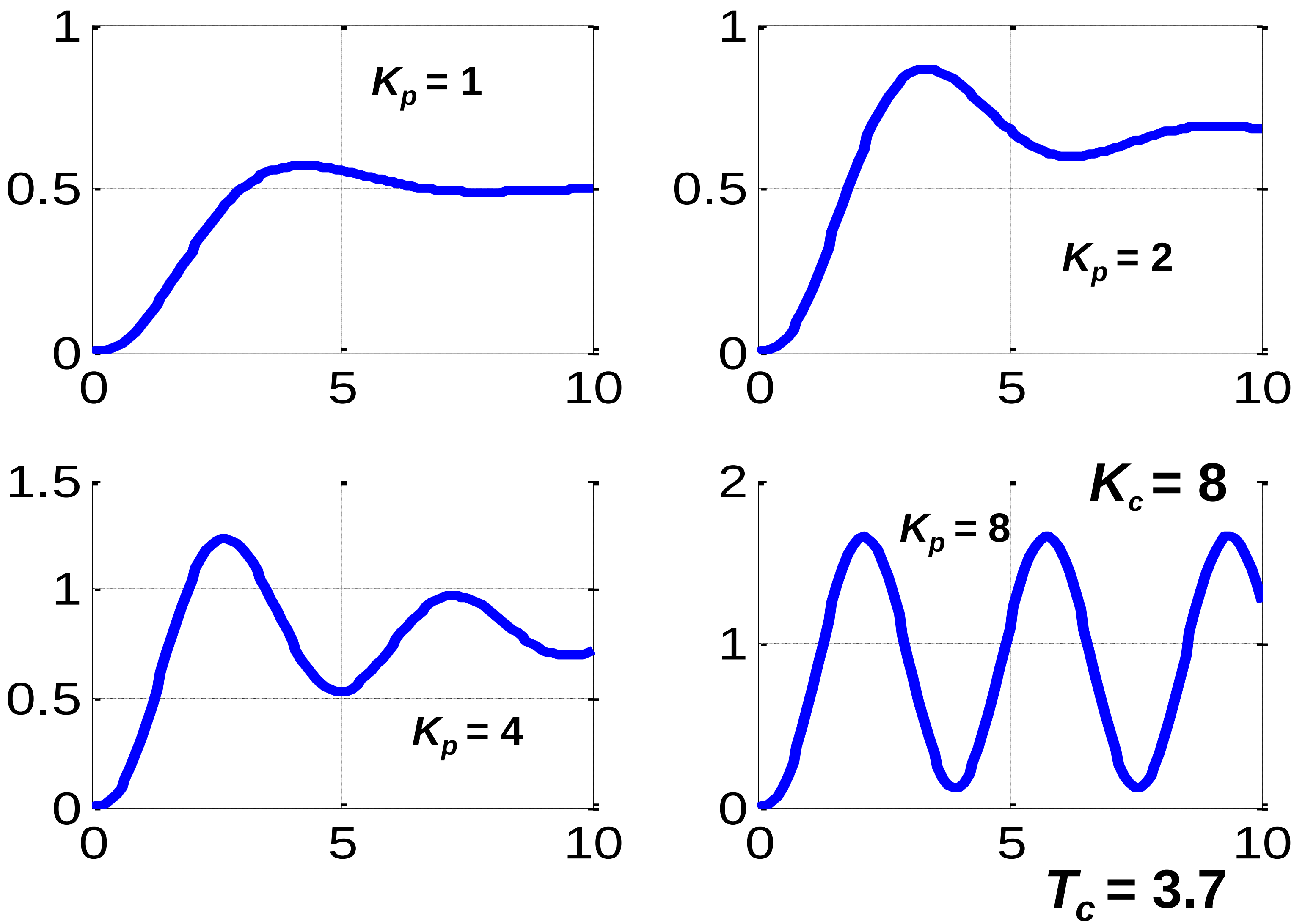
Ultimate sensitivity method



Example 1: Open-loop step response for "step response method"



Example 1 (cont'd): Closed-loop step responses for "Ultimate sensitivity method"



Summary

- Controller design based on root locus in MATLAB
- PID control
  - Most popular controller in industry
  - Simple controller structure & controller tuning
- Next
  - Frequency response