a place of mind

ELEC 341: Systems and Control

Lecture 15

Root locus:
PID controller design

Course roadmap

Modeling

- Laplace transform
- Transfer function
- Models for systems
  - Electrical
  - Electromechanical
  - Mechanical
- Linearization, delay

Analysis

- Stability
  - Routh-Hurwitz
  - Nyquist
- Time response
  - Transient
  - Steady state
- Frequency response
  - Bode plot

Design

- 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.
- The topics in Control Theory that you have learned so far are still very important!
  - ❑ These 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.
    - ➢ Using the control theory covered so far, we can take full advantage of MATLAB to design controllers more effectively.

SISO Design Tool in Matlab

- SISO (Single Input Single Output) design tool is the Graphical-User Interface (GUI) that allows you to design controllers/compensators.
- Type “sisotool (G)” in Matlab prompt:

$$\gg \mathbf{sisotool\ (G)}$$

- In the above command, “G” is the name of the open-loop transfer function that you have selected.

SISO Design Tool (Gain Compensator)

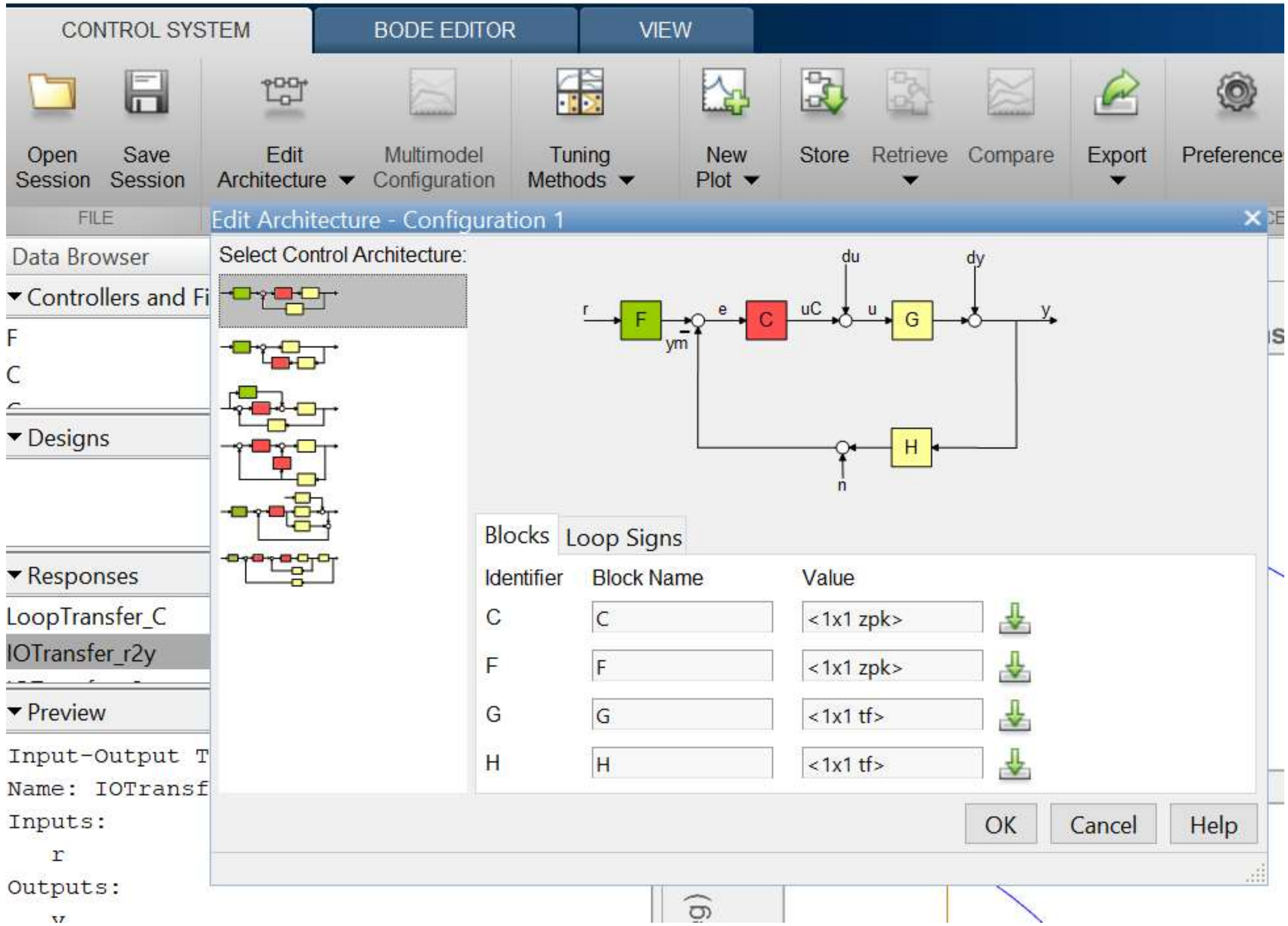
- Input the plant

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

or

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

```
>> sisotool(sysG)
```

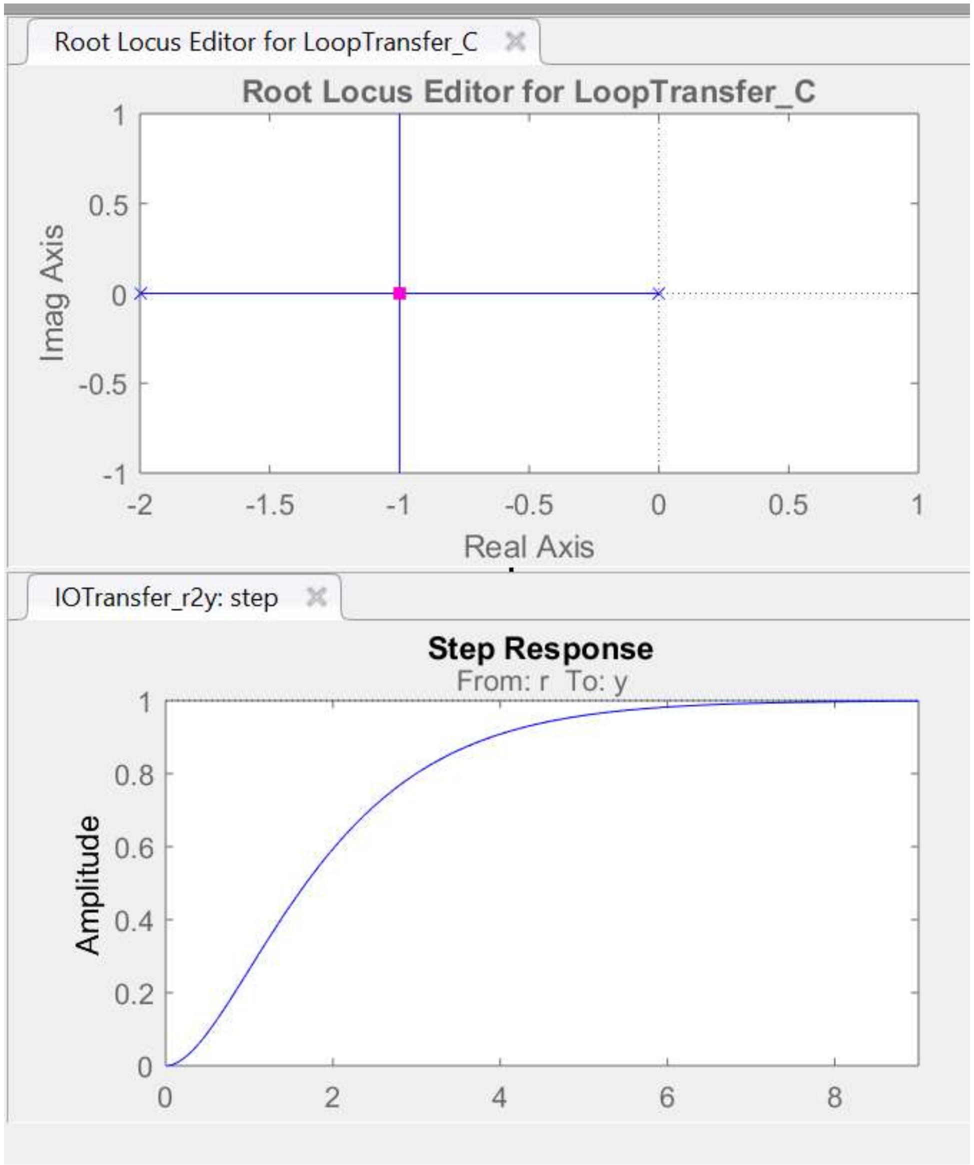


SISO Design Tool (Gain Compensator), (cont'd)

- You will see the root locus plot for

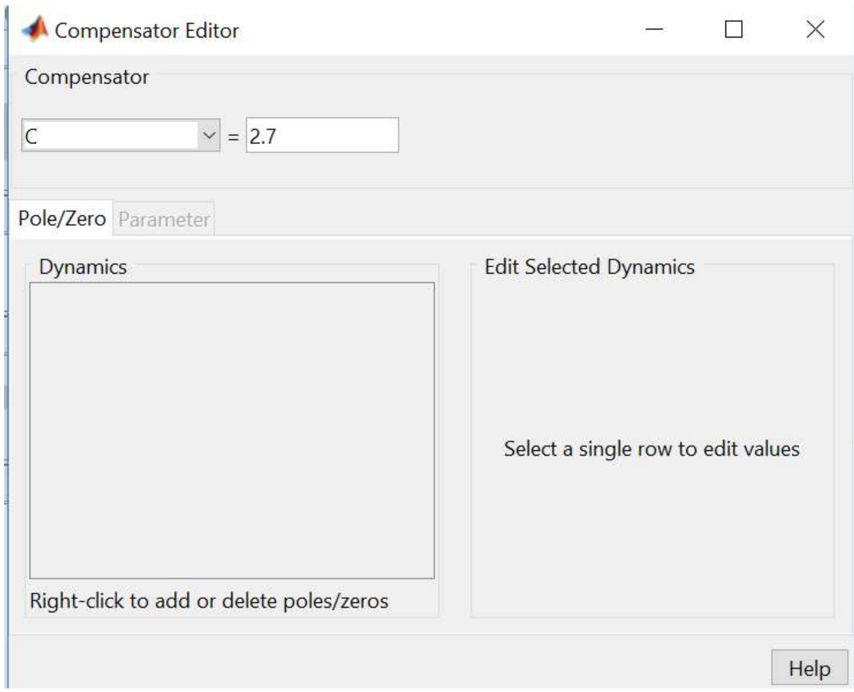
$$G(s) = \frac{1}{s(s+2)}$$

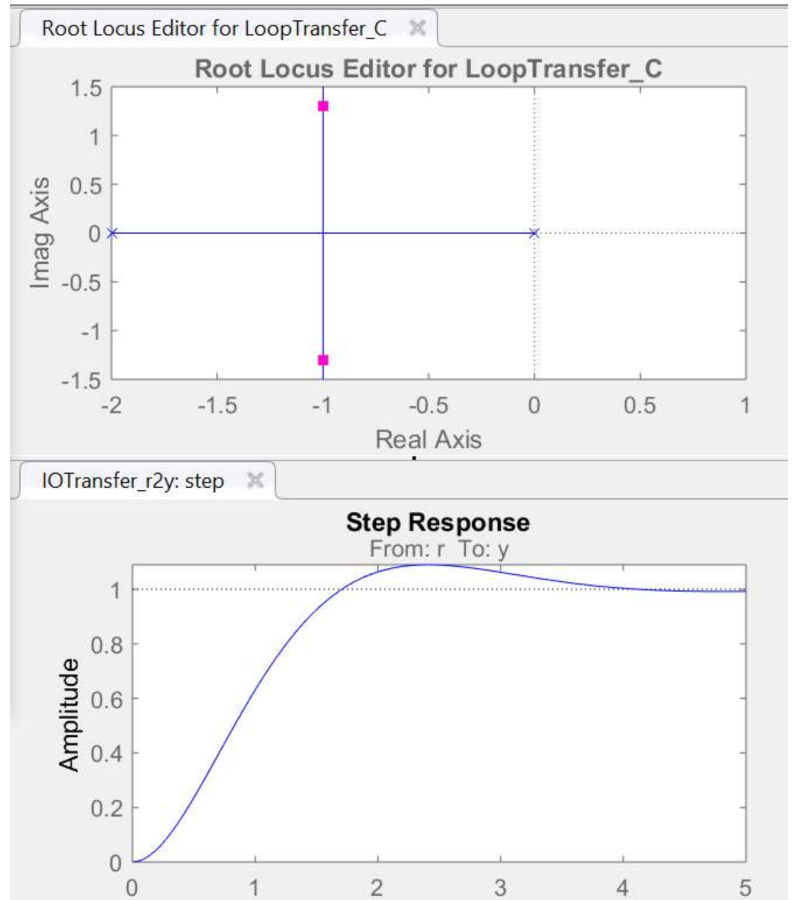
- The default setting is $C(s) = 1$.
- You can right-click on the plots to change the gain (among other things).



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

- Modify the gain for "**satisfactory**" step response.





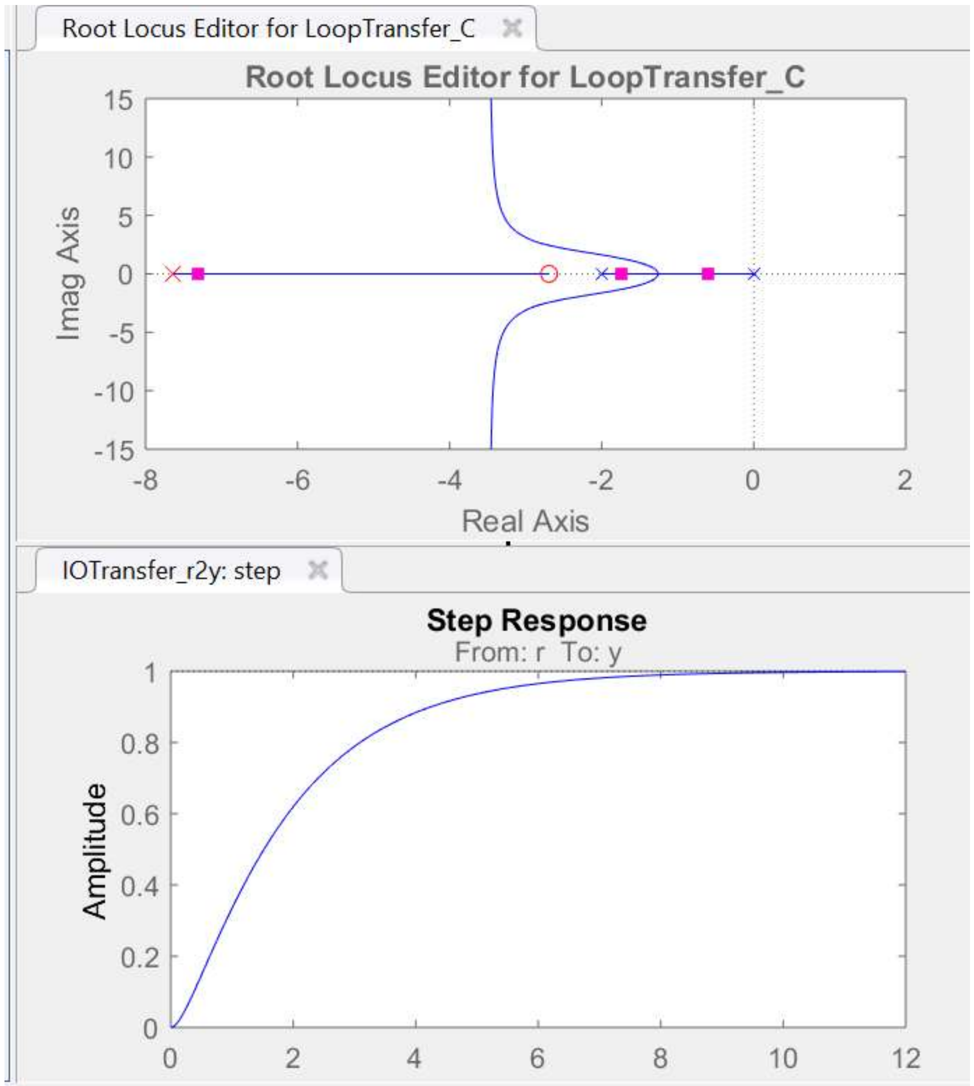
- By right-clicking on these figures, you can add various specs on the figures.

SISO Design Tool (Lead Compensator)

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

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

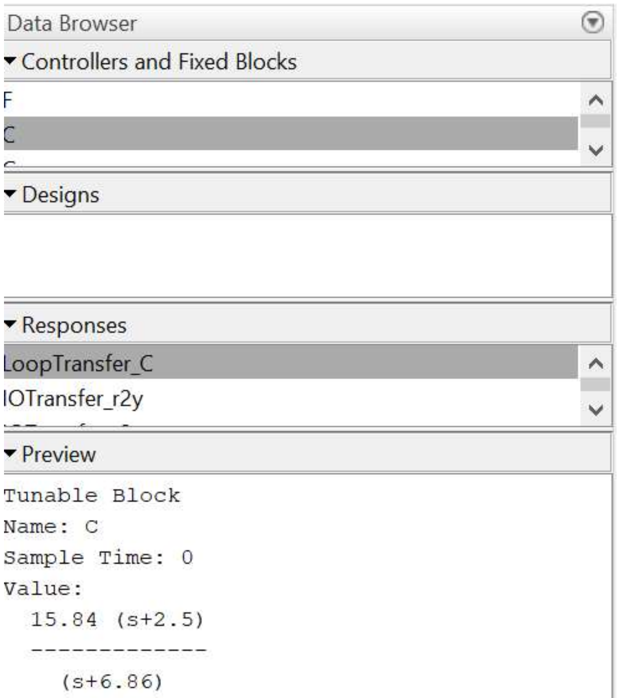
- If necessary, move the pole and zero
    - by click-and-drag, or
    - *Design → Edit Compensator…*

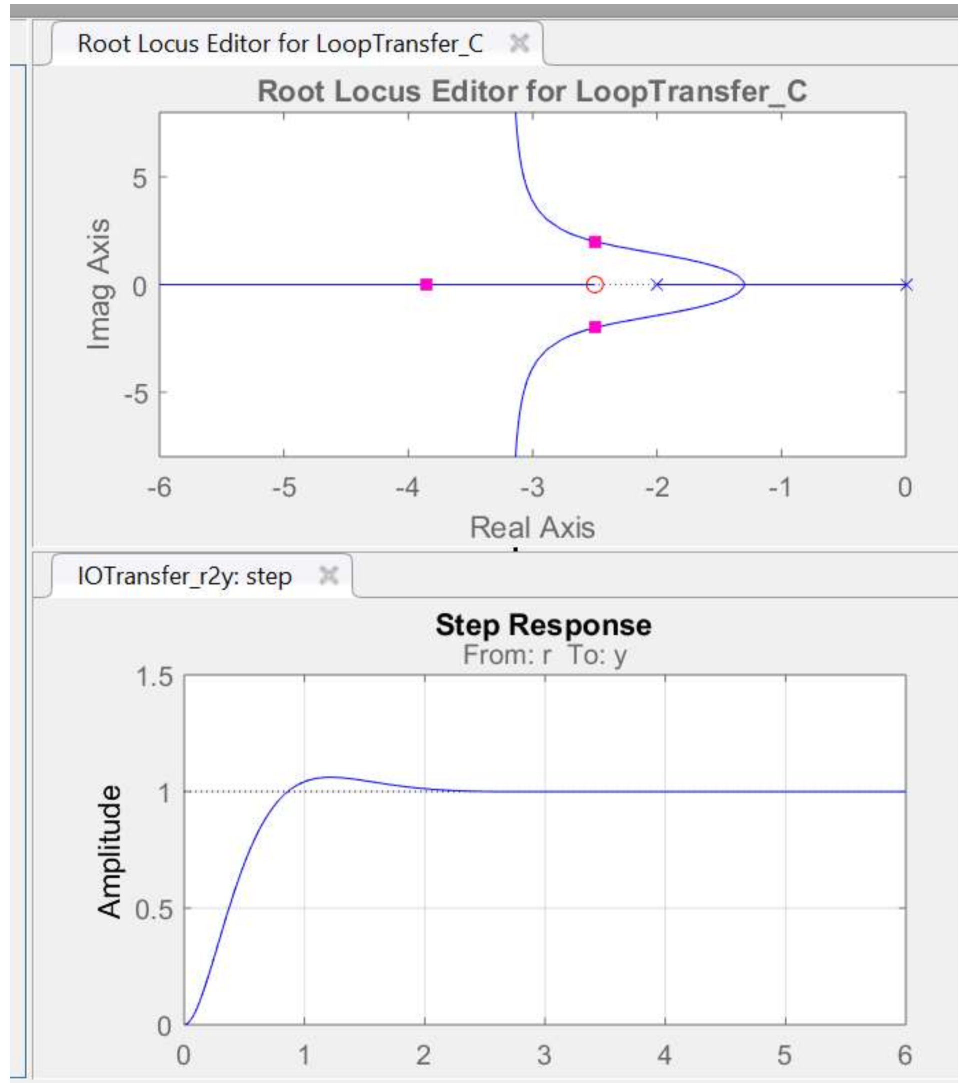


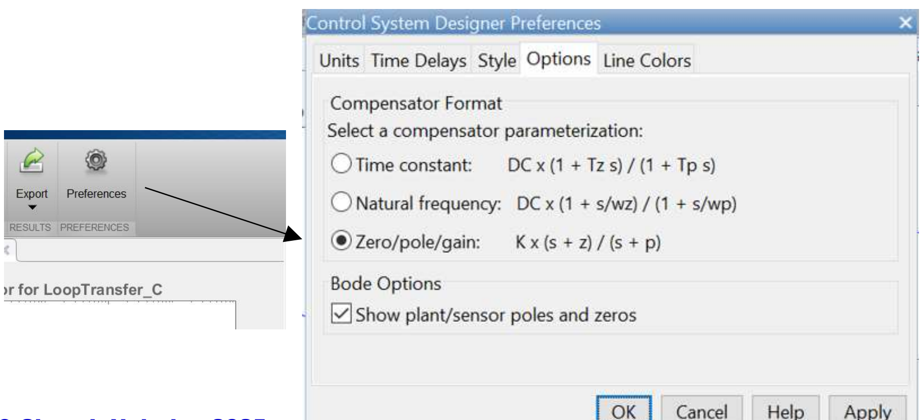
This is not with the gain we wanted (i.e., 15.84).

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

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

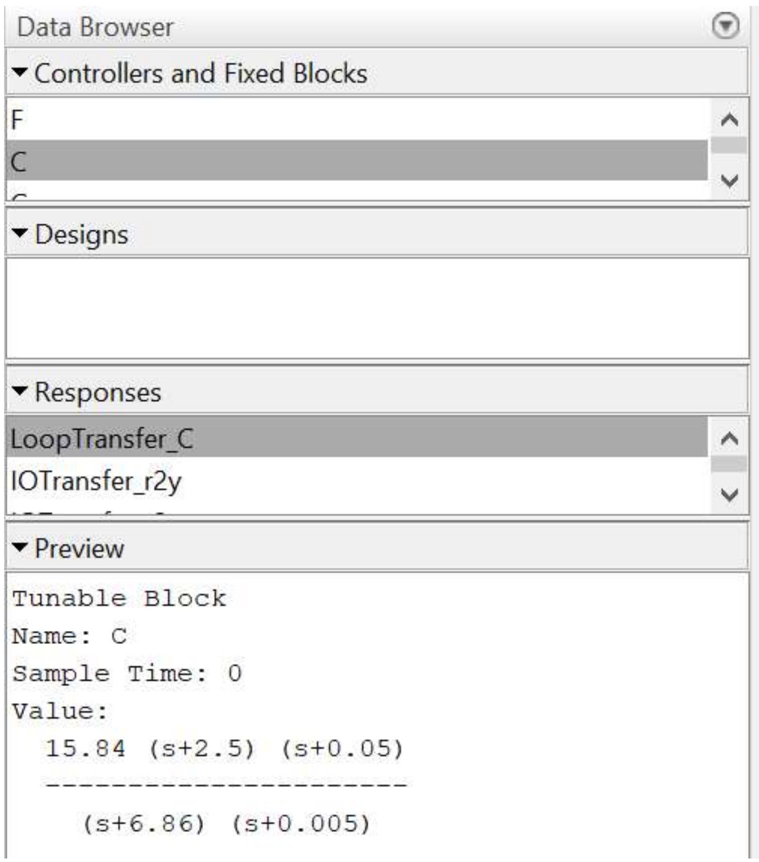


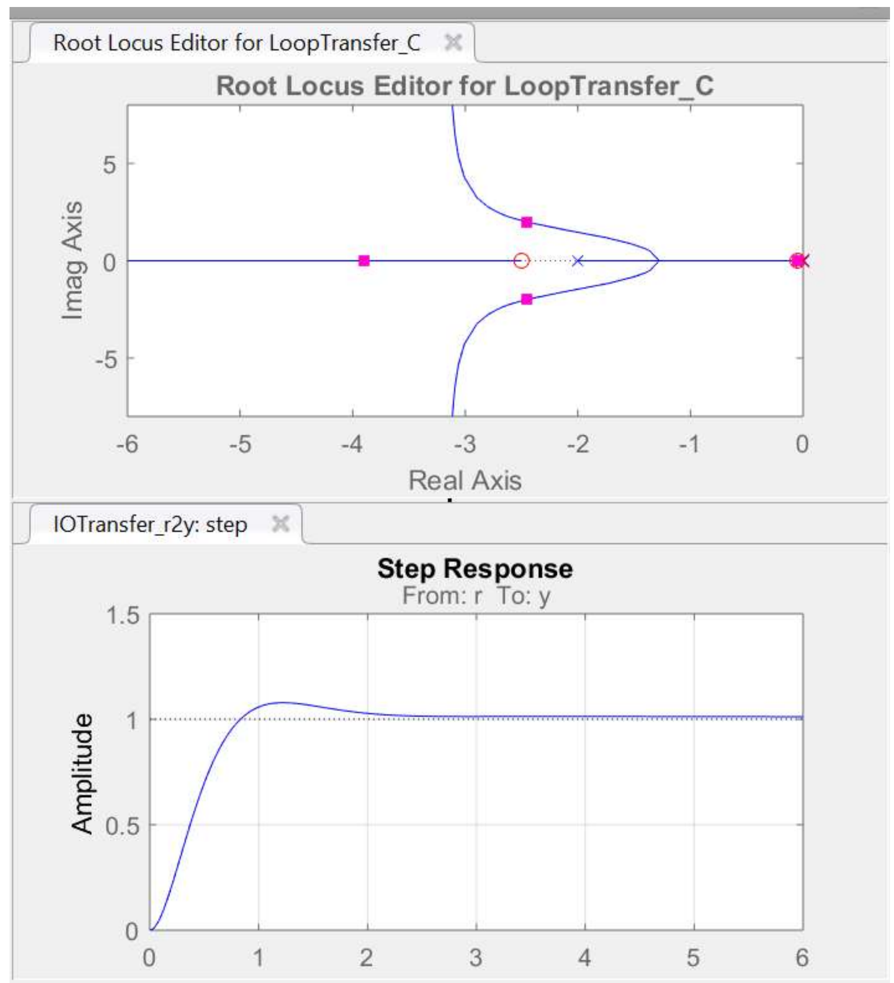




SISO Design Tool (Lead-Lag Compensator)

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





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

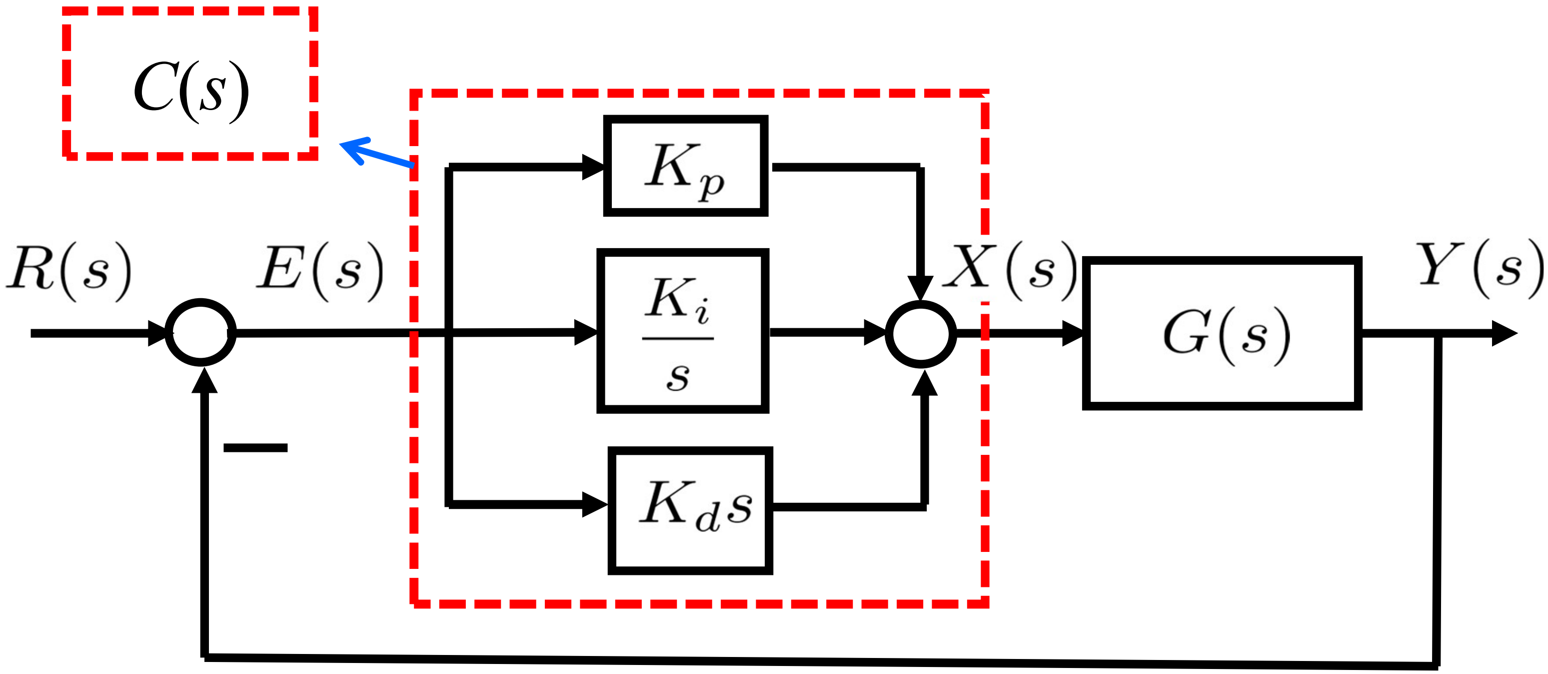
a place of mind

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 Design



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

K_p = proportional gain
K_i = integral gain
K_d = derivative gain

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

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

Notes on PID Controller Design

- Most popular in various industries (such as, process and robotics industries)
  - Good performance
  - Functional simplicity (operators can easily tune it)
- 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 $C(s) = K_p + K_d s$

a place of mind

Notes on PID Controller Design

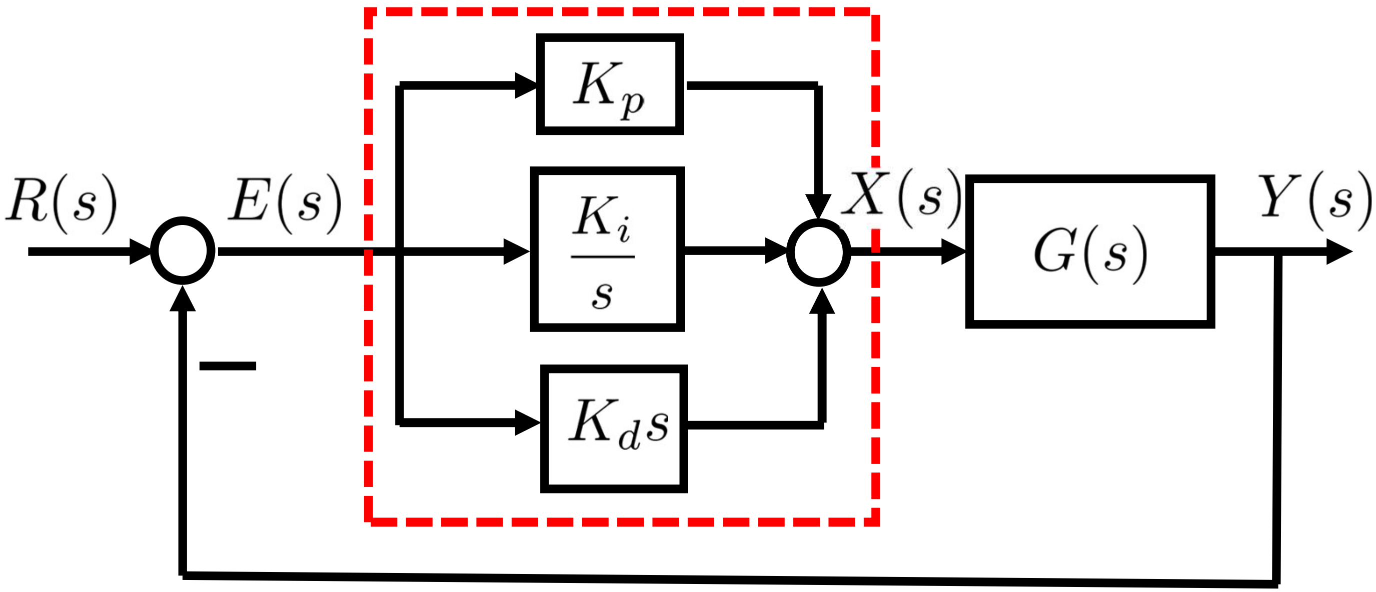
Applications of PID Controllers in Various Industries:

- **Automotive Industry**
  - **Cruise control**: Maintains vehicle speed under varying load and terrain.
  - **Engine control**: Regulates air-fuel mixture, ignition timing, and idle speed.
  - **Automatic transmission**: Enables smooth gear shifting.
  - **Active suspension**: Enhances ride comfort and vehicle stability.
- **Biomedical Engineering**
  - **Infusion pumps**: Precisely administer medication doses over time.
  - **Ventilators**: Controls airflow, pressure, and respiration timing.
  - **Artificial pancreas**: Adjusts insulin delivery in real time for diabetic patients.
  - **Neonatal incubators**: Maintains controlled temperature and humidity.
- **Wind Turbine Systems**
  - **Blade pitch control**: Improves efficiency and protects against high winds.
  - **Generator torque control**: Regulates output power based on wind speed.
  - **Yaw control**: Keeps turbine facing optimal wind direction.
- **Nuclear Reactor Control**
  - **Core temperature regulation**: Ensures safe and stable reactor operation.
  - **Steam generator control**: Maintains pressure and heat exchange efficiency.
  - **Neutron flux control**: Adjusts fission rate via control rod mechanisms.

Notes on PID Controller Design

- **Solar Power Plants**
  - **Solar tracking systems**: Adjust panel orientation for maximum sunlight exposure.
  - **Inverter control**: Stabilizes AC output despite DC input variation.
  - **Thermal regulation**: Prevents overheating of photovoltaic modules.
- **Robotics**
  - **Robotic surgery**: Enhances precision and stability in minimally invasive procedures through real-time force and motion control.
  - **Tactile sensing**: Enables robots to react to touch and grip objects delicately using pressure feedback.
  - **Haptics**: Provides realistic force feedback in teleoperation and prosthetics.
  - **Joint and limb control**: Ensures accurate, smooth movement of robotic arms and manipulators.
- **Manufacturing and Industrial Automation**
  - **Conveyor systems**: Controls belt speed and product spacing.
  - **CNC machines**: Maintains exact tool path and cutting depth.
  - **Chemical process control**: Adjusts flow, temperature, and pressure in real-time.
- **Aerospace and Aviation**
  - **Autopilot systems**: Maintains stable heading, altitude, and pitch.
  - **Flight control surfaces**: Adjusts ailerons, elevators, and rudders accurately.
  - **Environmental systems**: Regulates cabin pressure and temperature.

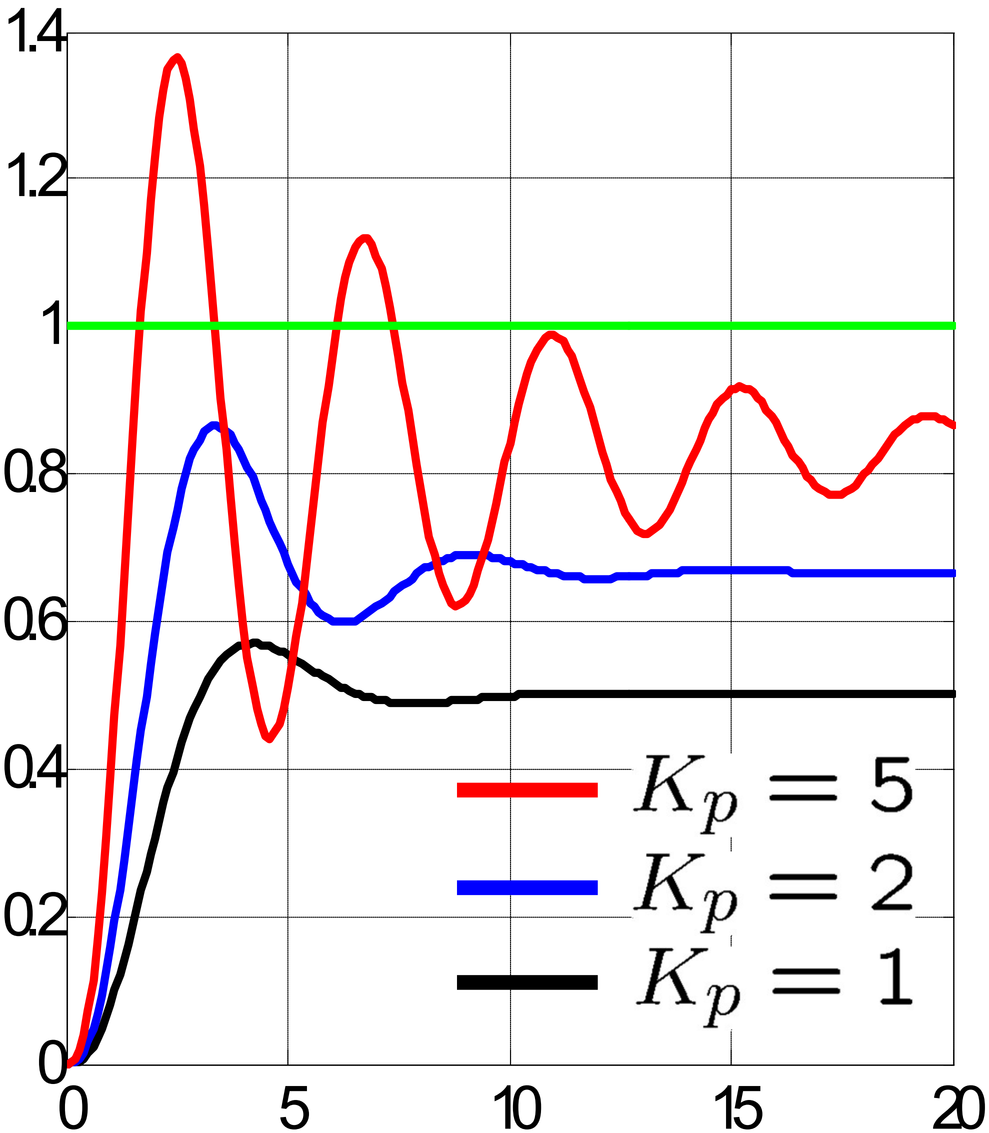
Example 1



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

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

Example 1 (cont'd): P controller

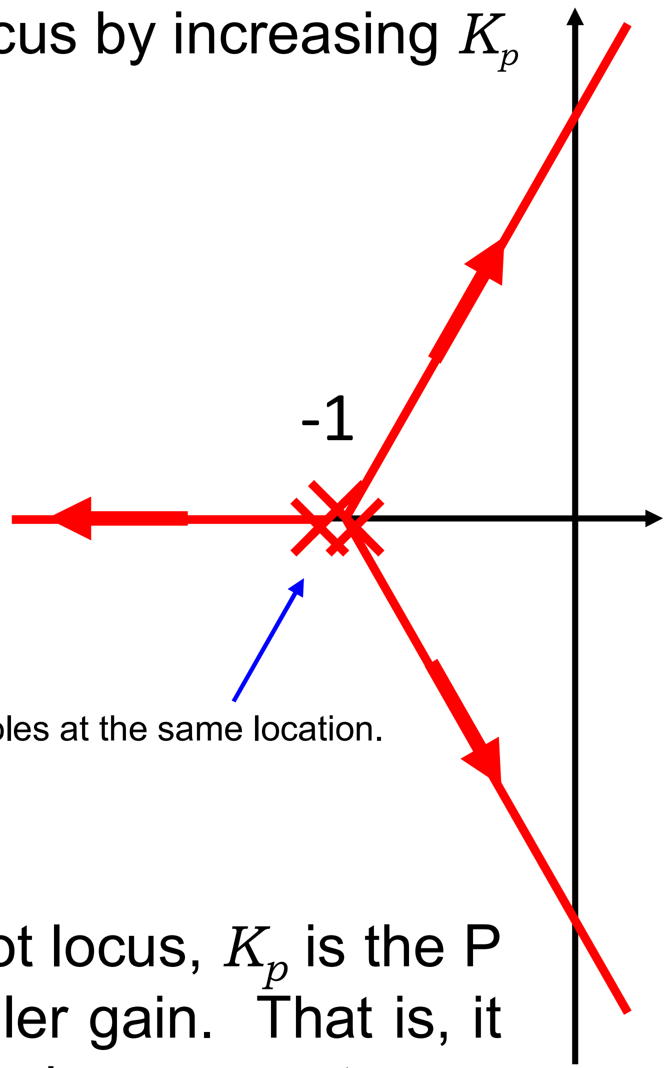


$$C(s) = K_p$$

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

Example 1 (cont'd) Interpretation: P controller

Root locus by increasing K_p



We have 3 poles at the same location.

For root locus, K_p is the P controller gain. That is, it is a varying parameter.

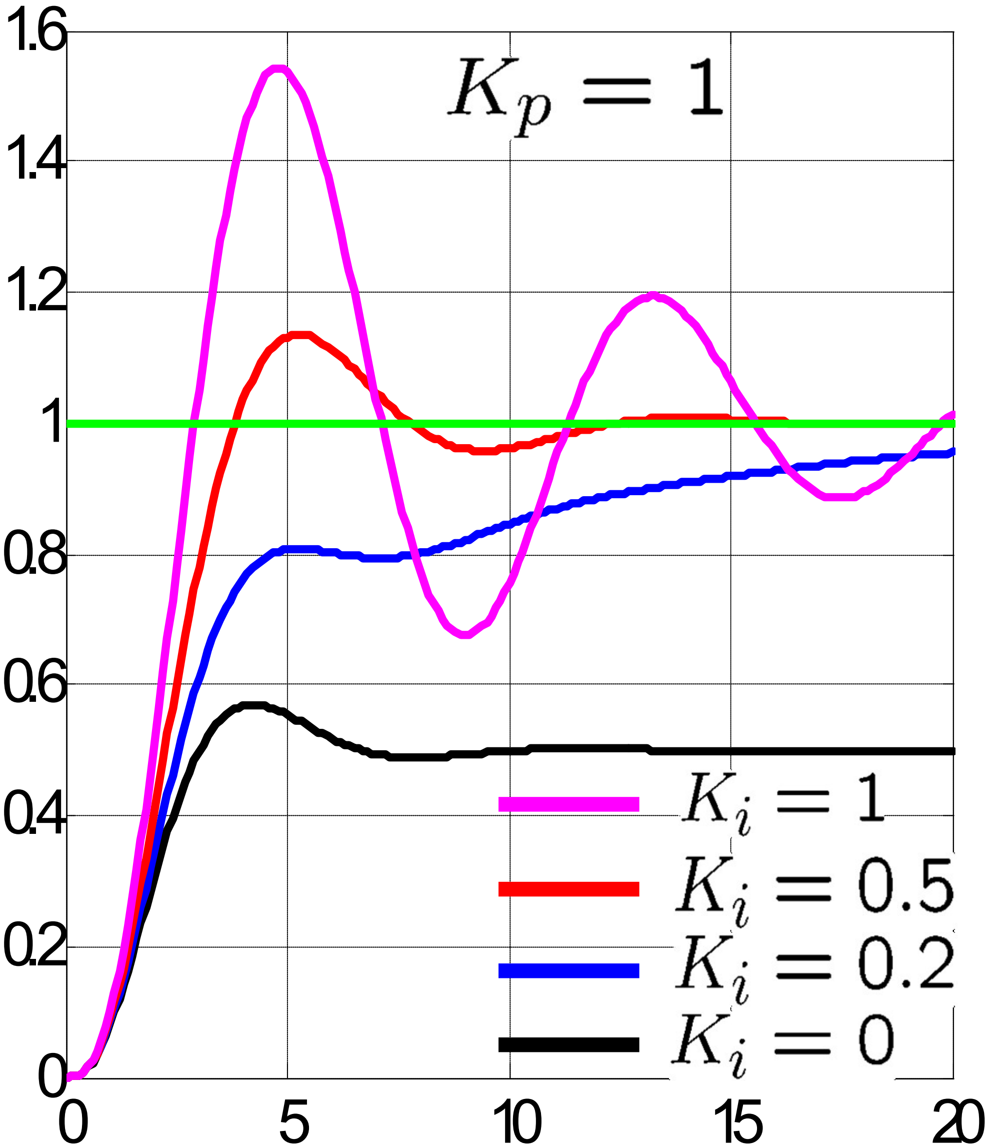
Steady state error for K_p = 1, the **black** curve in the previous slide:

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

$$e_{ss} = 0.5$$

a place of mind UBC

Example 1 (cont'd): PI controller

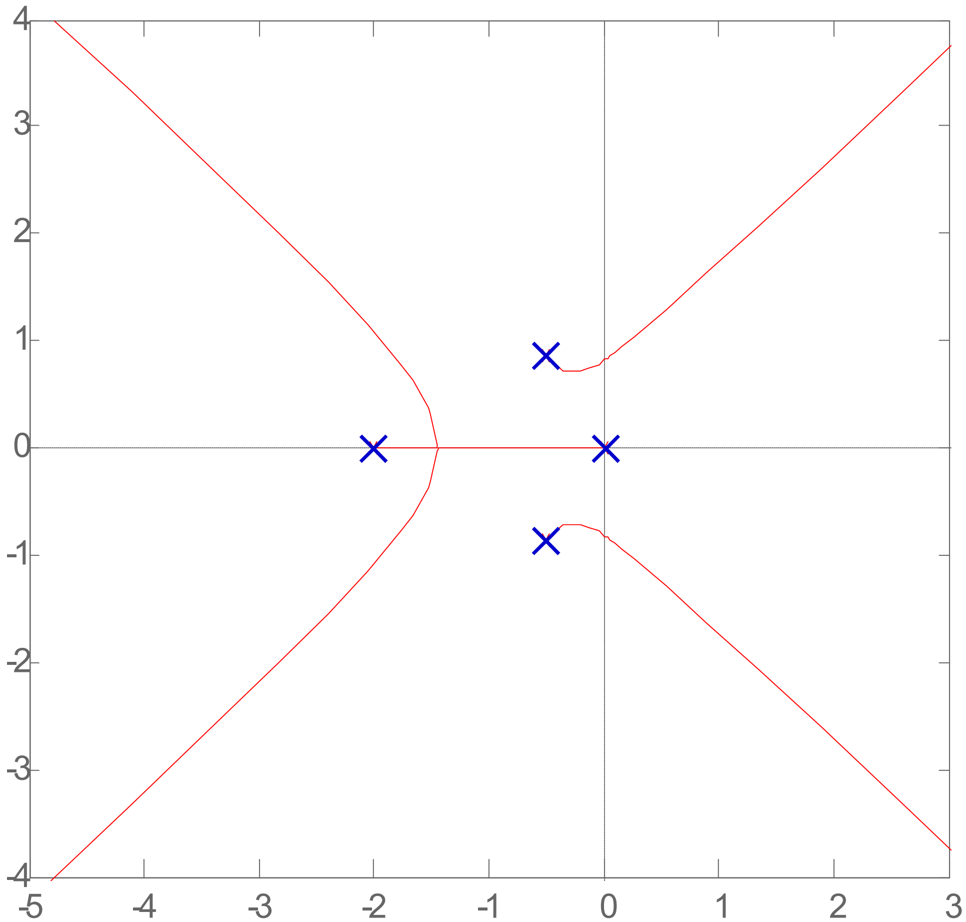


$$C(s) = K_p + \frac{K_i}{s}$$

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

Example 1 (cont'd)
Interpretation: PI controller (for $K_p = 1$)

Root locus by increasing K_i



$$C(s) = K_p + \frac{K_i}{s} = 1 + \frac{K_i}{s} = \frac{s + K_i}{s}$$

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

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

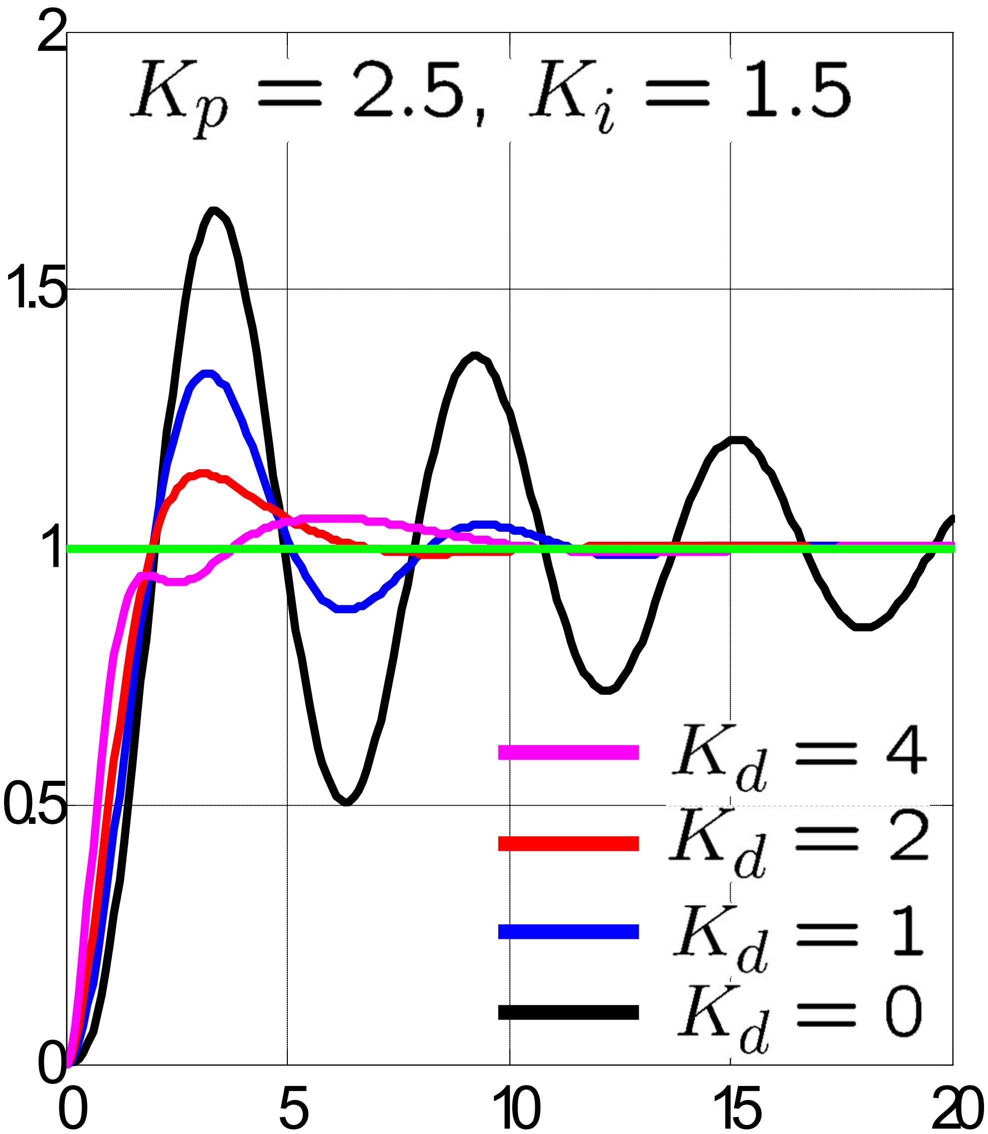
$$1 + K_i \frac{1}{s(s+2)(s^2+s+1)} = 0$$

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 (K_p = 2.5, K_i = 1.5)

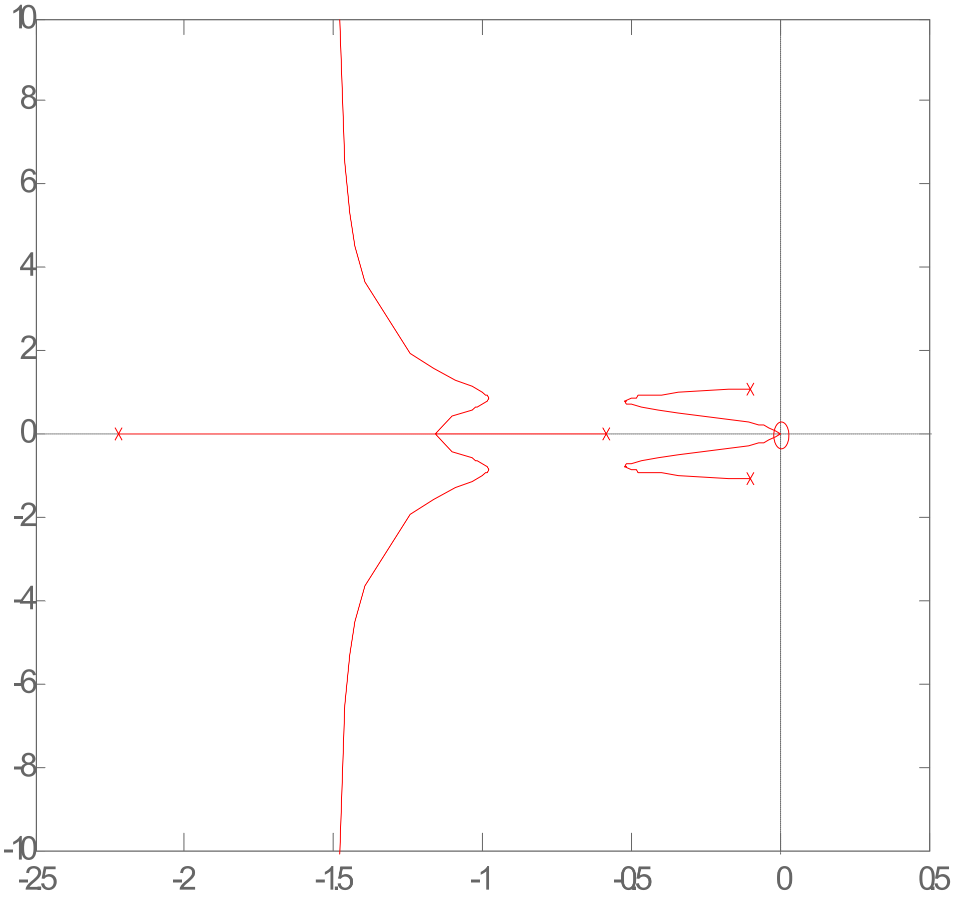


$$C(s) = K_p + \frac{K_i}{s} + K_d s$$

- 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 by increasing K_d



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

$$C(s) = 2.5 + \frac{1.5}{s} + K_d s = \frac{K_d s^2 + 2.5s + 1.5}{s}$$

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

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

Steady state error:

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

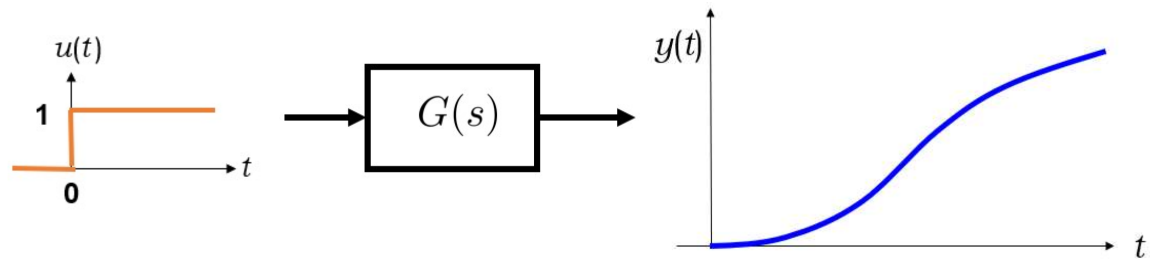
(due to an integrator in $C(s)$)

How to tune PID parameters

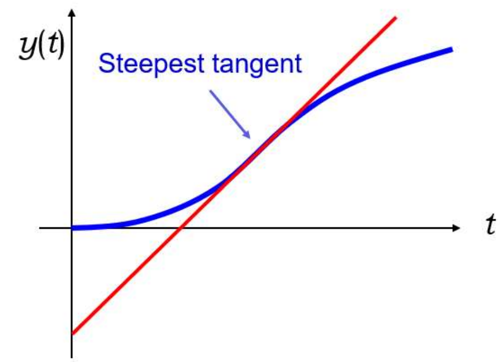
- Model-free (Empirical)
  - ❖ Trial and error
    - ➢ Useful only when trial-and-error tuning is allowed.
- Model-based (Analytical)
  - ❖ Root locus (RL)
  - ❖ Frequency response (FR) approach
  - ❖ Both RL and FR are useful only when a model is available.
- For both Model-free and Model-based systems, we can use a method called **"Ziegler-Nichols tuning rule"**
  - ❖ Useful even if a system is too complex to model.

Ziegler-Nichols PID tuning rules

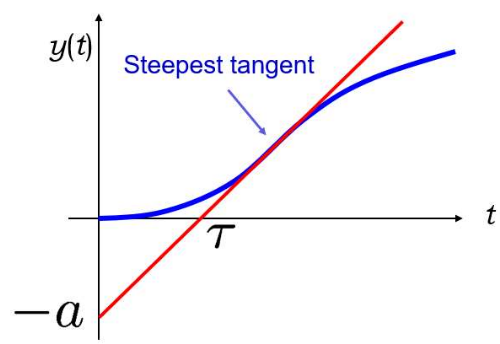
- **Open-loop Step response method** (only for stable systems)
- Follow the below steps:
- **Step 1:** Apply a unit step input to the below arrangement:



- **Step 2:** Find the steepest tangent for the output curve.



- **Step 3:** Find the values of τ and a.



To be continued

Ziegler-Nichols PID tuning rules

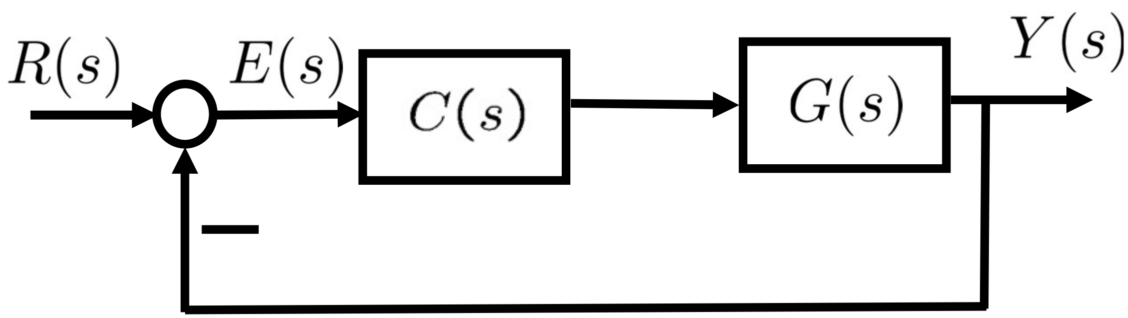
- **Step 4:** Use the following table to find the PID parameters:

PID parameters:

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

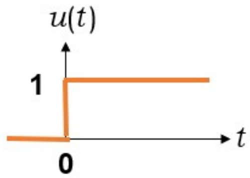
Standard form:

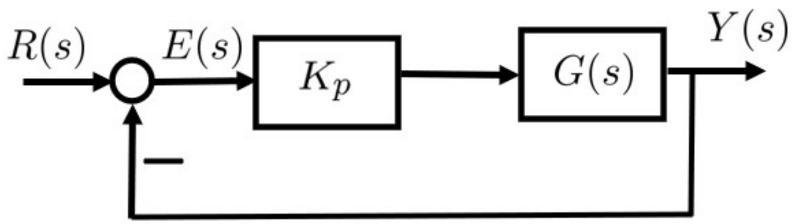
$$C(s) = K_p\left(1 + \frac{1}{T_I s} + T_D s\right)$$



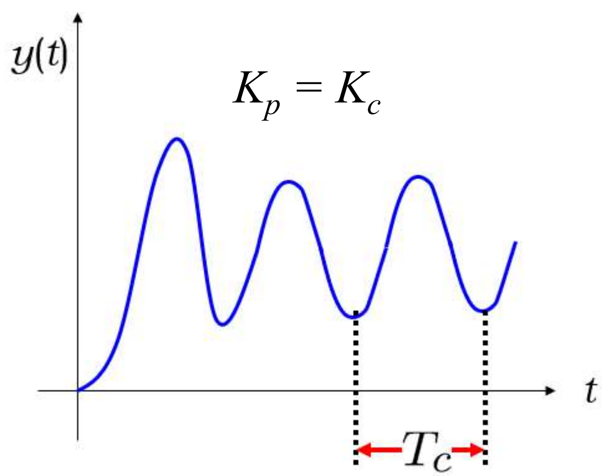
Ziegler-Nichols PID tuning rules

- **Ultimate sensitivity method** (closed-loop step response with a gain controller)
- Follow the below steps:
- **Step 1:** Apply a unit step input to the below arrangement:





- **Step 2:** Increase gain (K_p) until the output becomes oscillatory. Call this specific K_p, the K_c.



- **Step 3:** Find the value of T_c, which is basically the period of oscillation.

To be continued

Ziegler-Nichols PID tuning rules

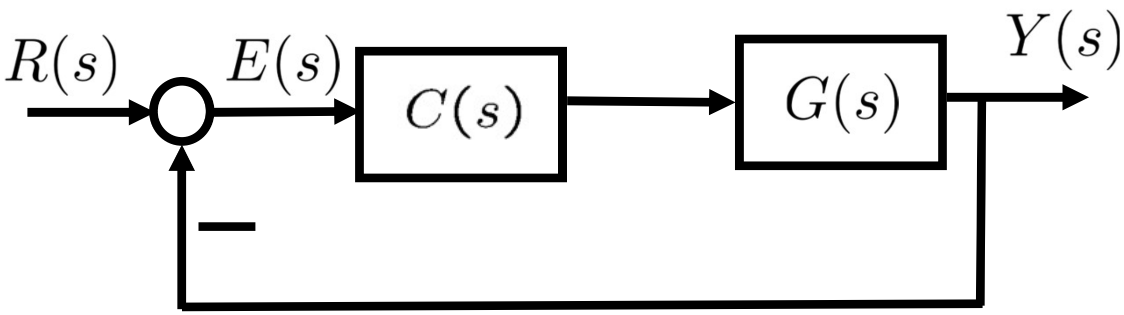
- **Step 4:** Use the following table to find the PID parameters:

PID parameters

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

Standard form:

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



Example 1: Open-loop Step Response Method

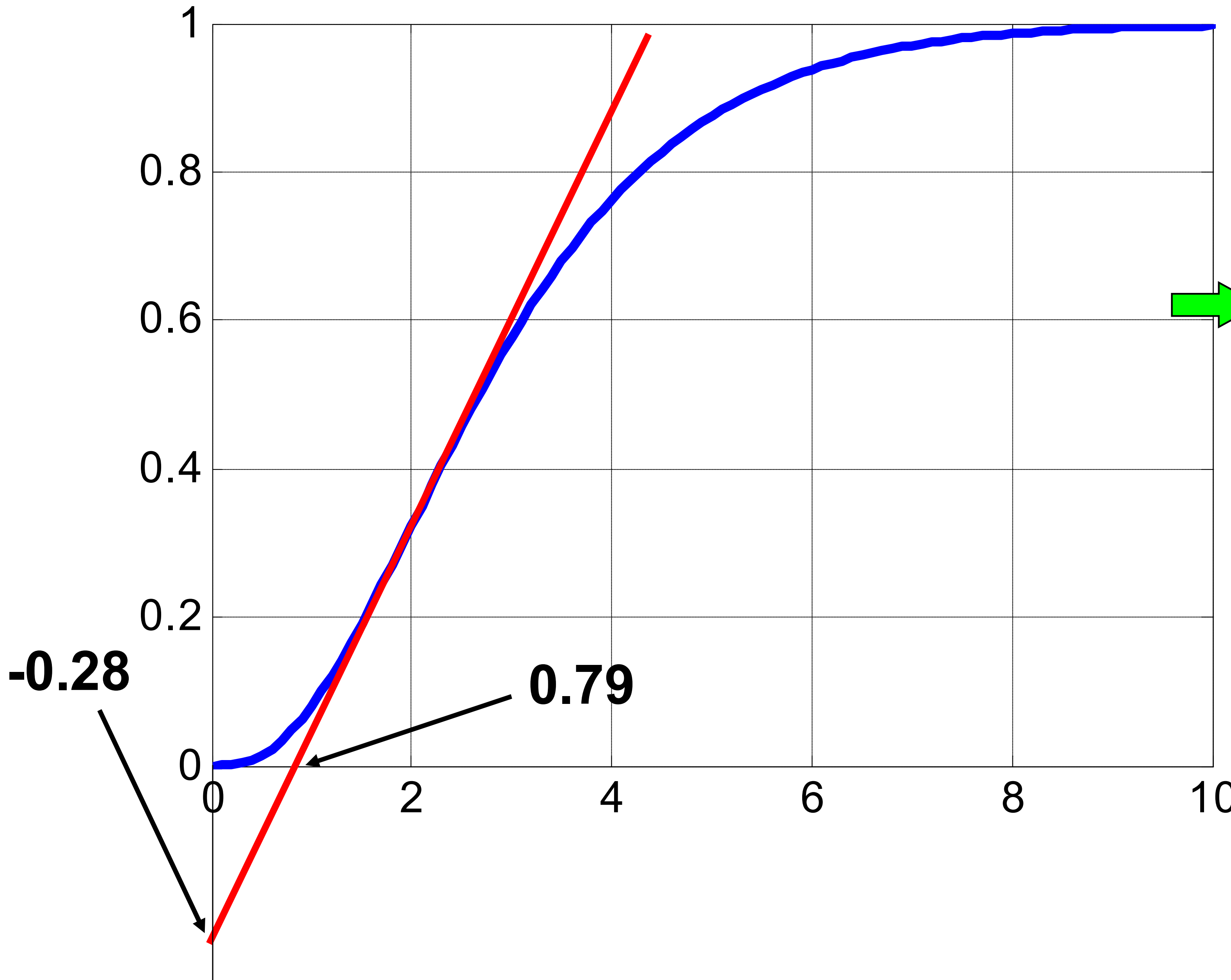
Design a suitable PID controller using **Open-loop Step Response Method** for the experimentally obtained response to a unit step input. The result of this experiment is presented in the next slide.

Note:

Although not necessary for the computation process of this design problem, the actual $G(s)$ is:

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

Example 1 (cont'd): Open-loop Step Response Method



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

$$\boxed{\begin{aligned} K_P &= 4.285 \\ T_I &= 1.580 \\ T_D &= 0.395 \end{aligned}}$$

Example 1 (cont'd): Open-loop Step Response Method

Details of Solution:

Step response method (PID):

$$\tau = 0.79$$
$$a = 0.28$$

PID parameters:

$$K_p = \frac{1.2}{a} = \frac{1.2}{0.28} = 4.285$$
$$T_I = 2\tau = 2 \times 0.79 = 1.58$$
$$T_D = 0.5\tau = 0.5 \times 0.79 = 0.395$$

The PID controller transfer function $C(s)$ is:

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

Substituting the values:

$$\boxed{C(s) = 4.285 \left(1 + \frac{1}{1.58 s} + 0.395 s\right)}$$

Example 2: Ultimate Sensitivity Method

(a) Find the numerical values of the three parameters in a PID controller that is designed using **Ultimate Sensitivity Method**. Use analytical method.

(b) Explain the experimental protocol for how to design a suitable PID controller using Ultimate Sensitivity Method for the given $G(s)$.

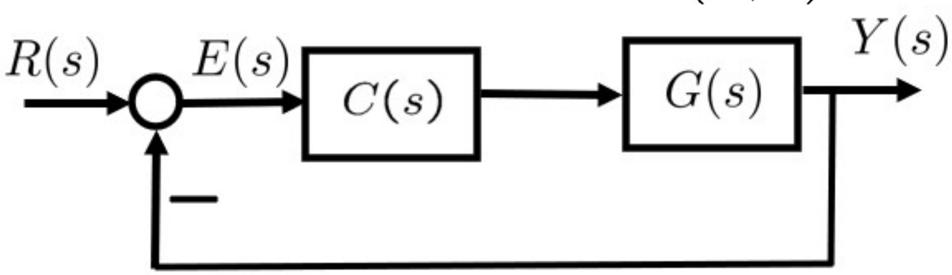
$$G(s) = \frac{1}{(s+1)^3}$$

Example 2 (cont'd): Ultimate Sensitivity Method

(a)

For the following closed-loop system, calculate the CLTF and then the characteristic equation:

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



$\Rightarrow$ characteristic equation:

$$s^3 + 3s^2 + 3s + 1 + K_c = 0$$

s^3	1	3
s^2	3	$1 + K_c$
s^1	$\frac{8 - K_c}{3}$	
s^0	$1 + K_c$	

ROZ

$\Rightarrow \frac{8 - K_c}{3} = 0 \Rightarrow K_c = 8$

$3s^2 + (1+8)s^0 = 0 \Rightarrow 3s^2 + 9 = 0 \Rightarrow s = \pm 1.732i \Rightarrow$

$\omega = \frac{2\pi}{T_c}$

$\omega = 1.732 \Rightarrow T_c = 3.7 \Rightarrow$

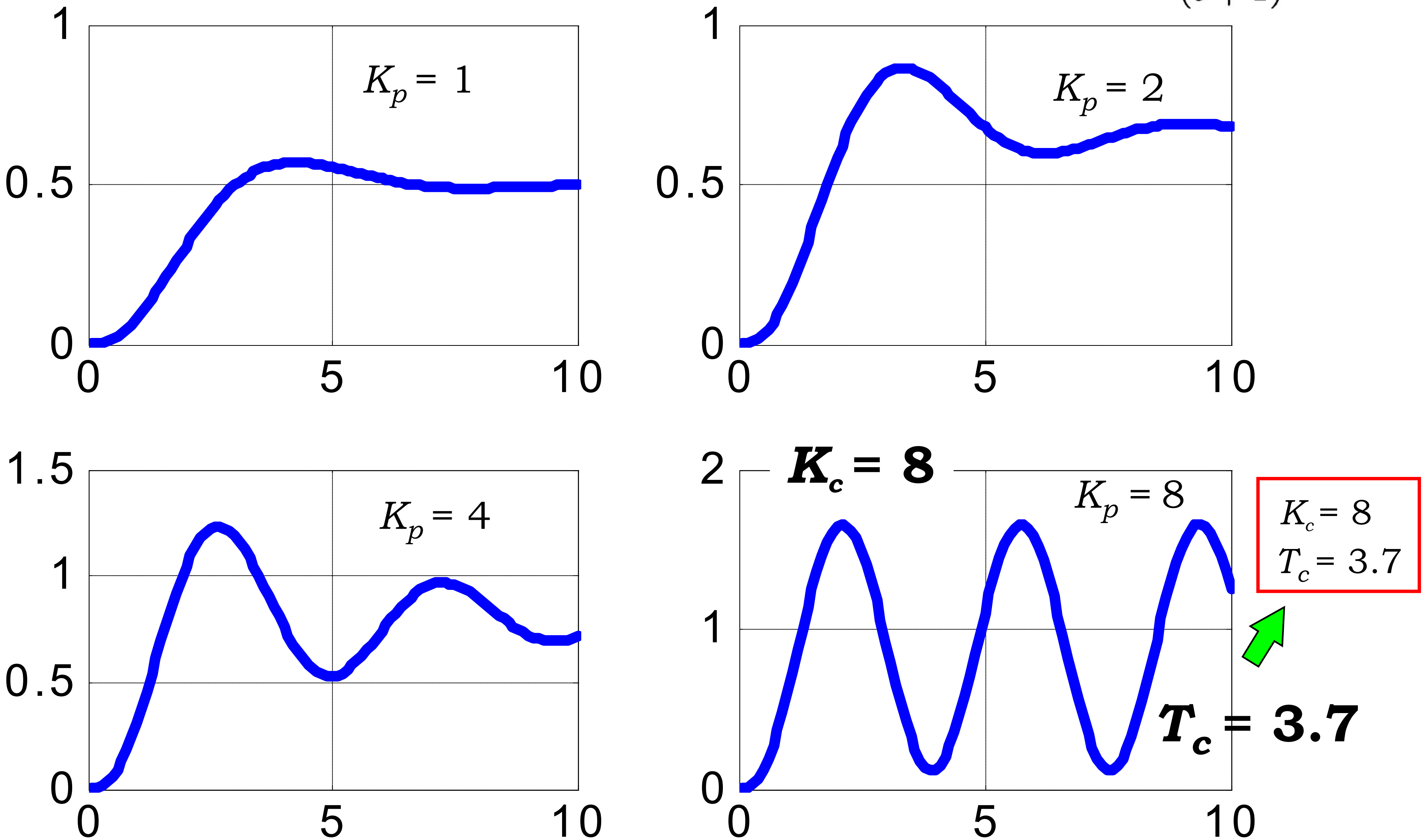
$K_P = 4.800$
$T_I = 1.813$
$T_D = 0.453$ $\Rightarrow$

$$\boxed{C(s) = 4.800\left(1 + \frac{1}{1.813s} + 0.453s\right)}$$

Example 2 (cont'd): Ultimate Sensitivity Method

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

(b)

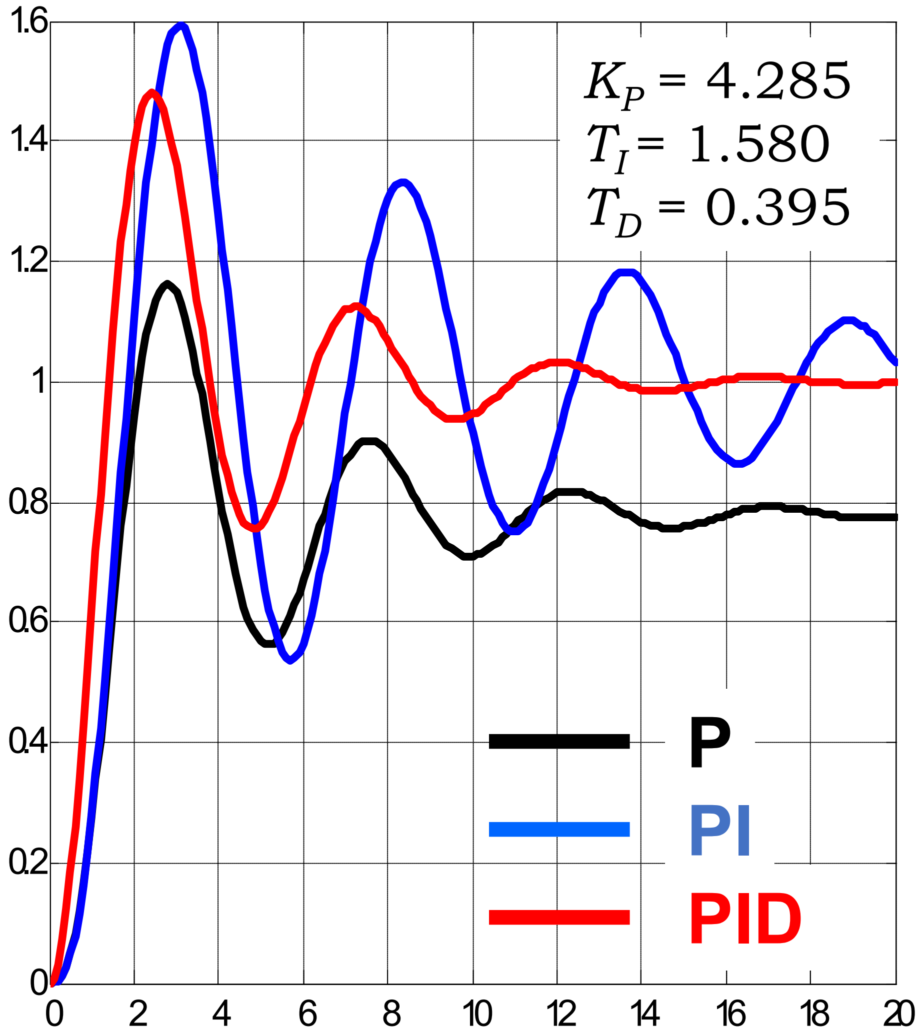


Note: In our experiment, we keep changing the value of gain (K_p) until we see continuous oscillations at the output (i.e., an oscillatory output). We show this particular value of gain by K_c. So, K_c is K_p when the output is oscillatory.

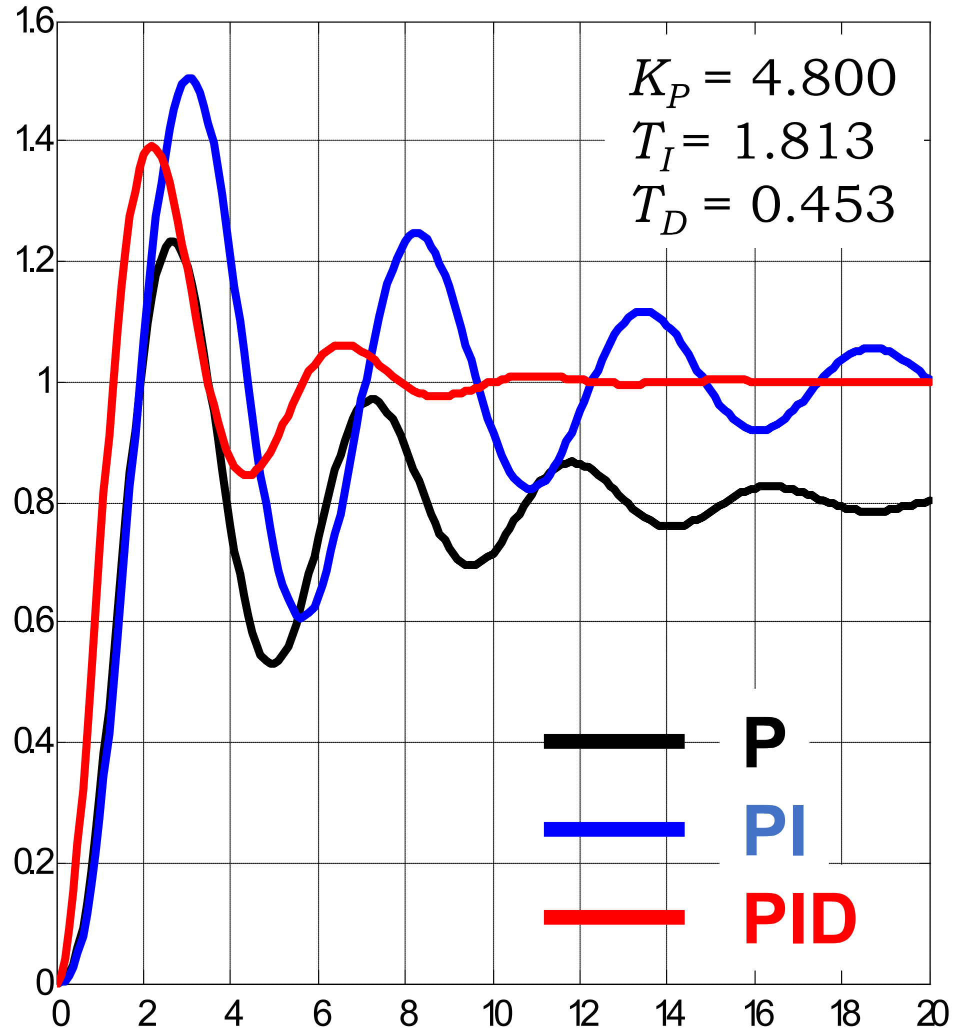
Example 3 (revisited): $G(s) = \dfrac{1}{(s+1)^3}$

Make a comparison between Method 1 and Method 2.

Open-loop step response method



Ultimate sensitivity method



Summary

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