Basic Controllers for LTI Systems

Lecture 5

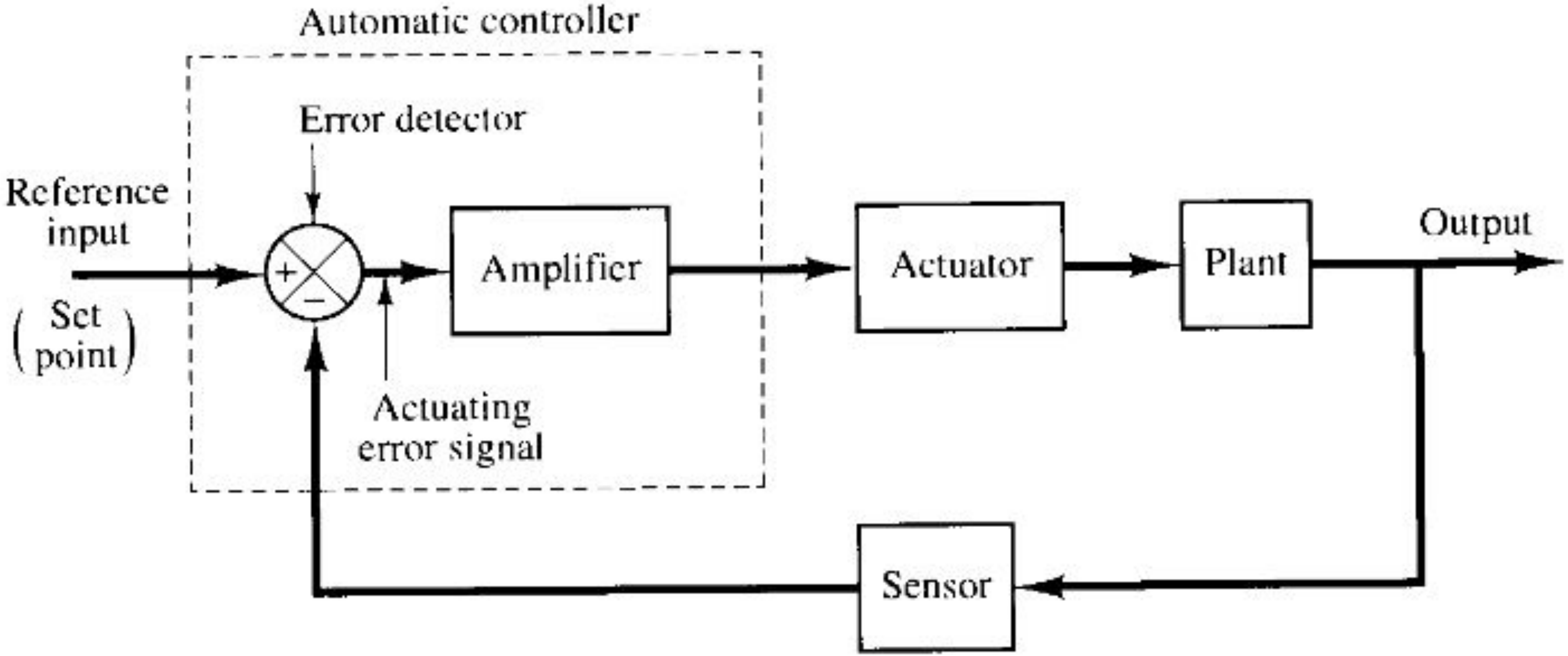
Outline

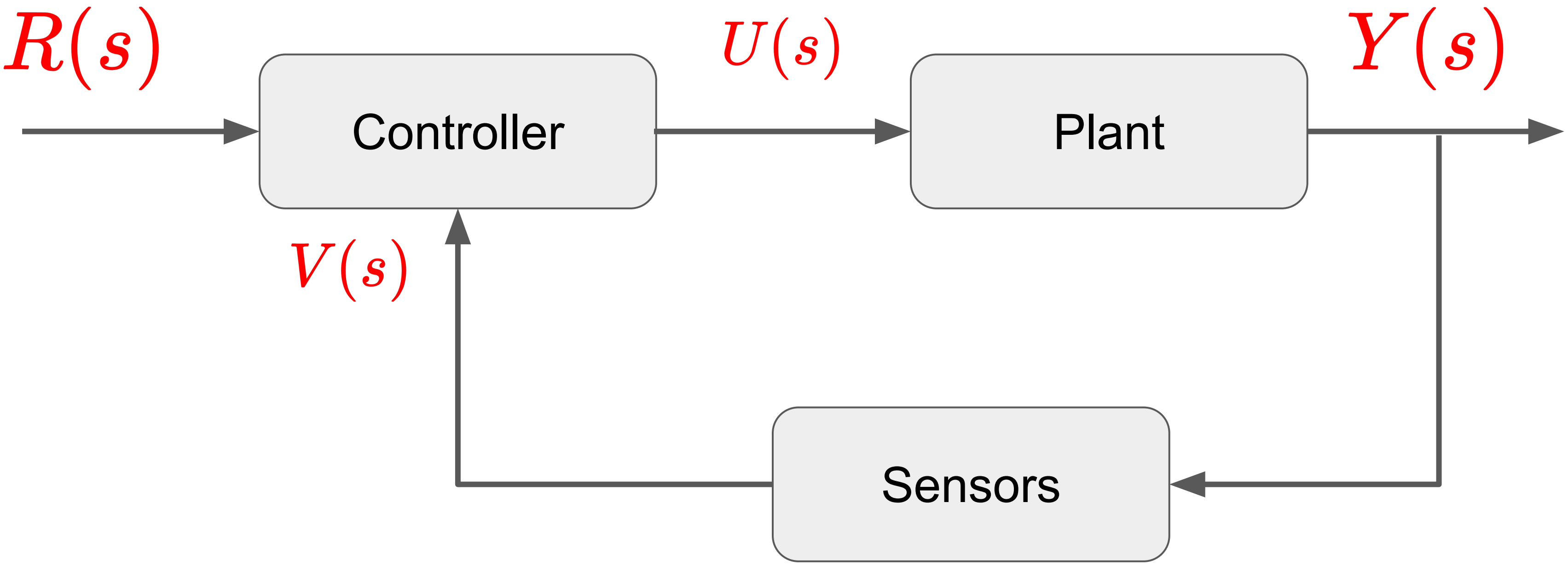
- Some of the Basic controllers
  - P, PD, PI and PID controller
- Analyse stability of Higher-order Systems
- Steady-State Response of LTI systems

Basic Controllers for Linear Systems

- On-off Controllers
- Proportional Controllers
- Integral Controllers
- Proportional-plus-integral (PI) Controllers
- Proportional-plus-derivative (PD) Controllers
- Proportional-plus-integral-plus-derivative (PID) Controllers

Block Diagram of a Typical Industrial Control System





$R(s)$
Controller
$U(s)$
Plant
$Y(s)$
$V(s)$
Sensors

Effect of Feedback

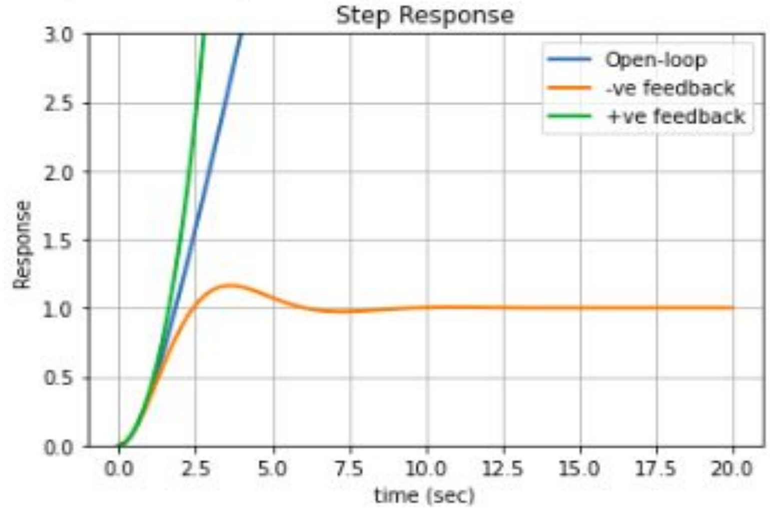
- Positive Feedback Destabilizes the system.

- Negative Feedback tends to stabilizes the system.

- The objective of controller design is to obtain desirable closed-loop performance which includes the following:
    - Stability (better relative stability)
    - Better Transient performance (low rise time, low peak time, low peak overshoot, lower settling time and preferably critical damping)
    - Better steady-state response (or zero steady state error)

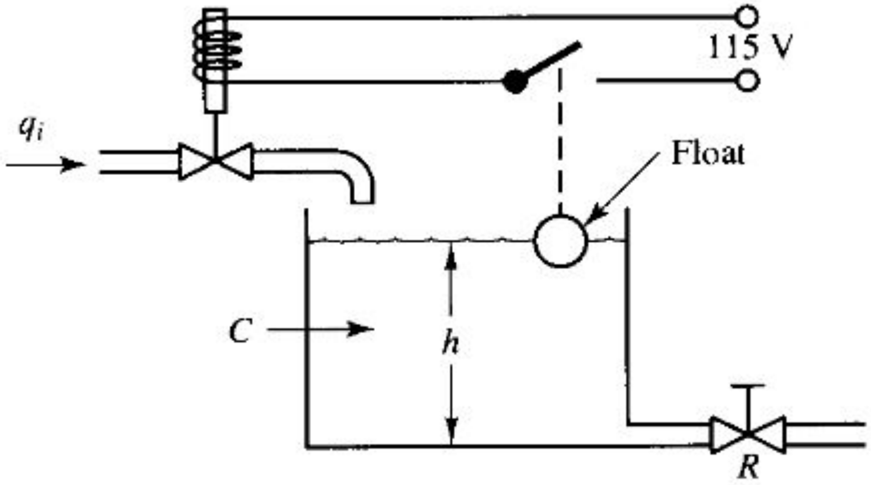
```python
from control import *
import matplotlib.pyplot as plt
import numpy as np

# open-loop system
g1 = tf(1, [1, 1, 0])
print('Open-loop system:', g1)

# closed-loop system
gc1 = feedback(g1, 1, -1) # negative feedback
print('Closed-loop system with negative feedback:', gc1)
gc2 = feedback(g1, 1, 1) # positive feedback
print('Closed-loop system with positive feedback:', gc2)

t = np.linspace(0,20,100)
t,y = step_response(g1, t)
t,y1 = step_response(gc1, t)
t,y2 = step_response(gc2, t)

plt.plot(t, y, lw = 2, label='Open-loop')
plt.plot(t, y1, lw = 2, label='-ve feedback')
plt.plot(t, y2, lw = 2, label='+ve feedback')
plt.ylim((0,3))
plt.xlabel('time (sec)')
plt.ylabel('Response')
plt.title('Step Response')
plt.grid()
plt.legend(loc='best')
```

```
Open-loop system:
   1
-------
s^2 + s

Closed-loop system with negative feedback:
     1
-----------
s^2 + s + 1

Closed-loop system with positive feedback:
     1
-----------
s^2 + s - 1

<matplotlib.legend.Legend at 0x7eff6f3fe5c0>
```



On-Off Controllers

- The controller has only two-values or two positions: On-off.
- Very simple and commonly used controller in industries:
  - Liquid-level controllers
  - Temperature controllers

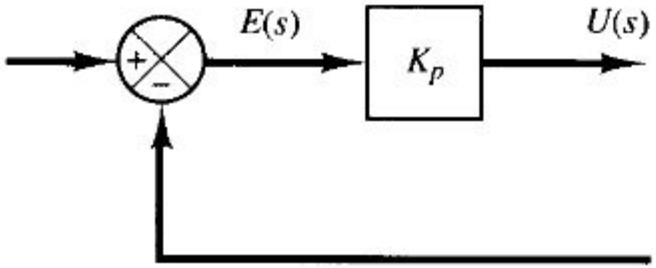


Liquid-level Control System

Proportional Controller

$$u(t) = K_p e(t)$$

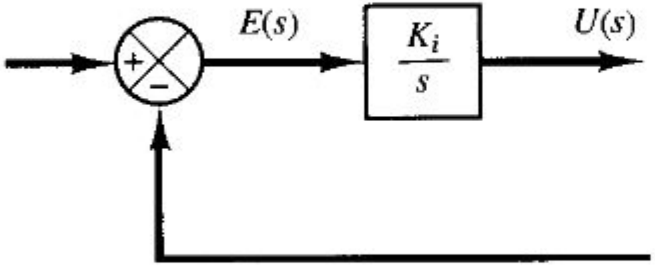
$$\frac{U(s)}{E(s)} = K_p$$



Integral Control Action

$$u(t) = K_i \int_0^t e(t)\, dt$$

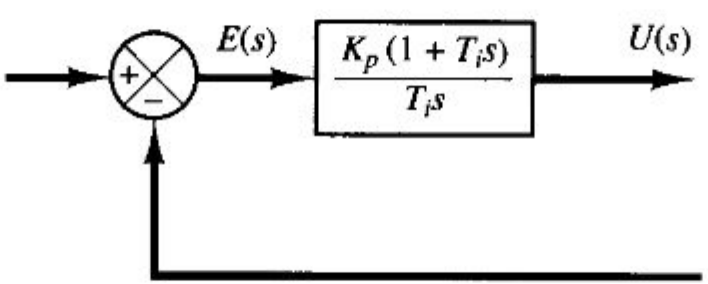
$$\frac{U(s)}{E(s)} = \frac{K_i}{s}$$

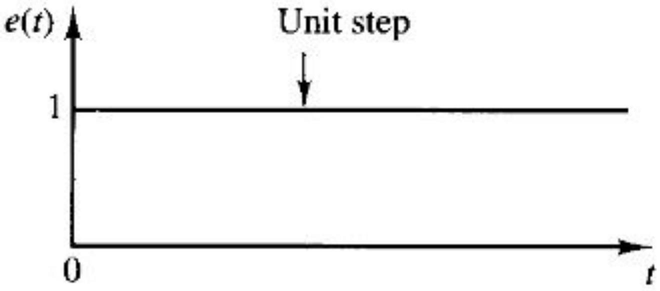


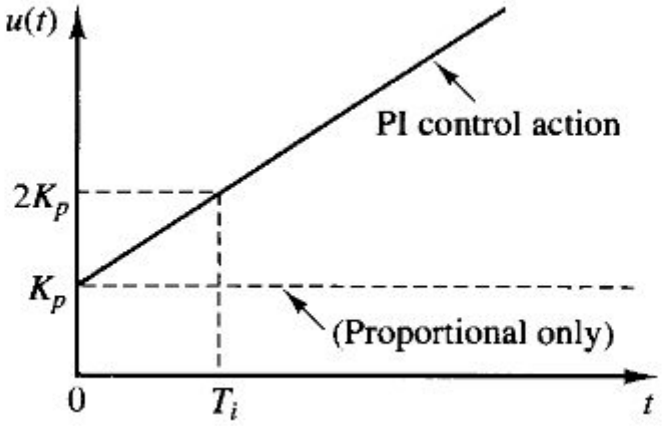
Proportional-plus-integral Control Action

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)\,dt$$

$$\frac{U(s)}{E(s)} = K_p\left(1 + \frac{1}{T_i s}\right)$$



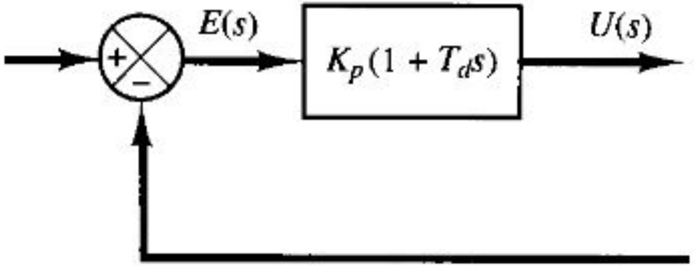


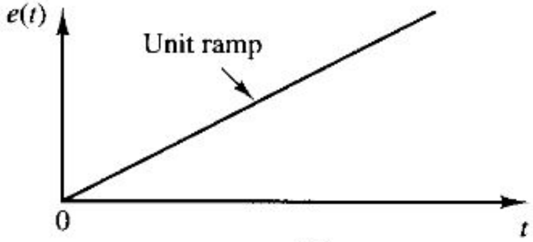


Proportional-plus-derivative control action

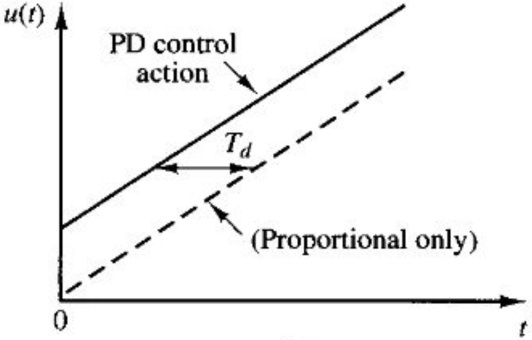
$$u(t) = K_p e(t) + K_p T_d \frac{de(t)}{dt}$$

$$\frac{U(s)}{E(s)} = K_p(1 + T_d s)$$





(b)

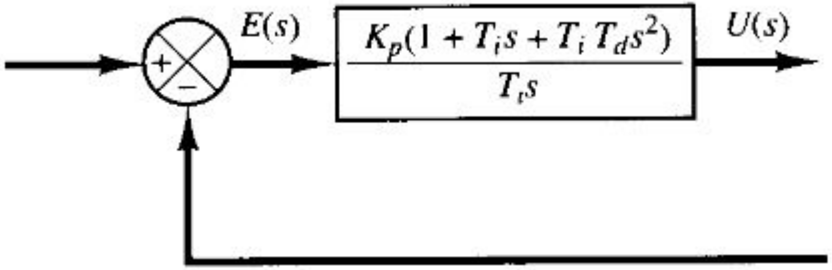


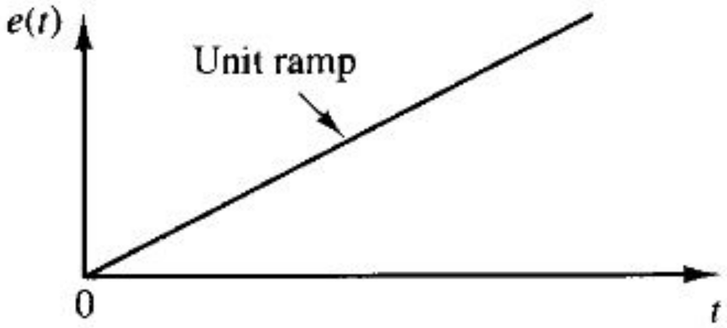
(c)

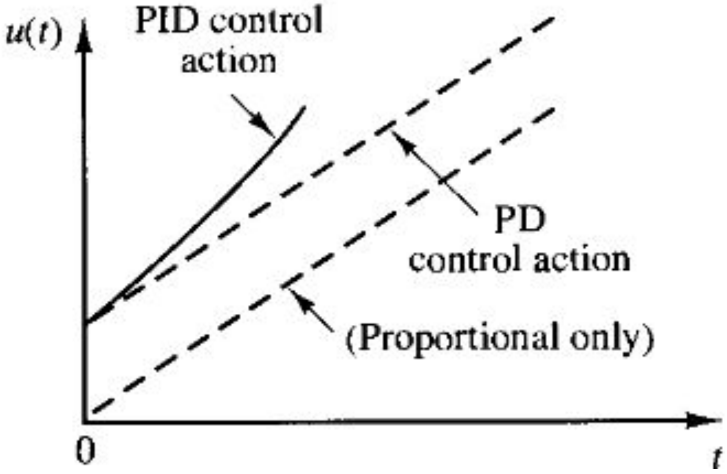
Proportional-plus-integral-plus-derivative control action

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)\,dt + K_p T_d \frac{de(t)}{dt}$$

$$\frac{U(s)}{E(s)} = K_p\left(1 + \frac{1}{T_i s} + T_d s\right)$$





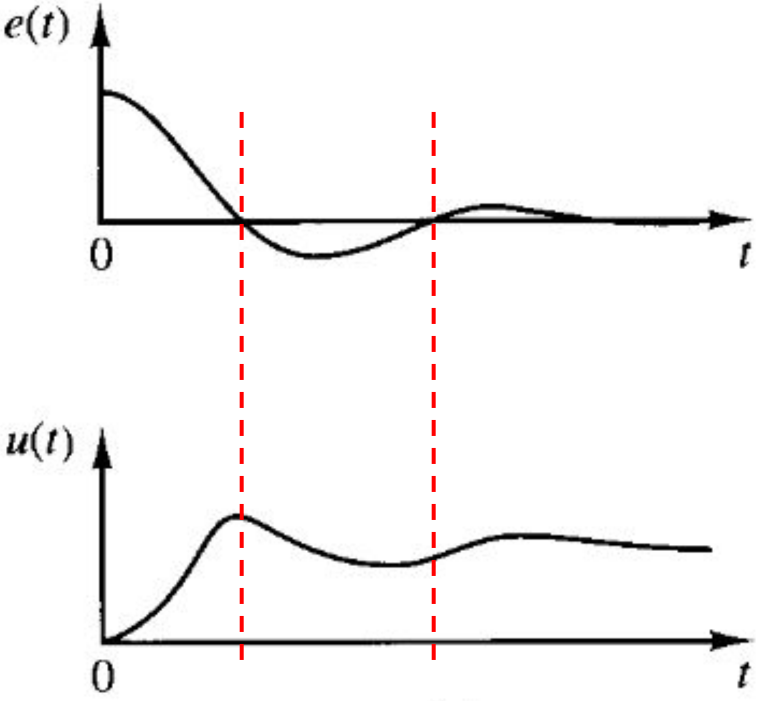


Effect of Integral Control Action on System Performance

- In the proportional control of a plant whose transfer function does not possess an integrator 1/s, there is a steady state error or offset in the response to a step input.

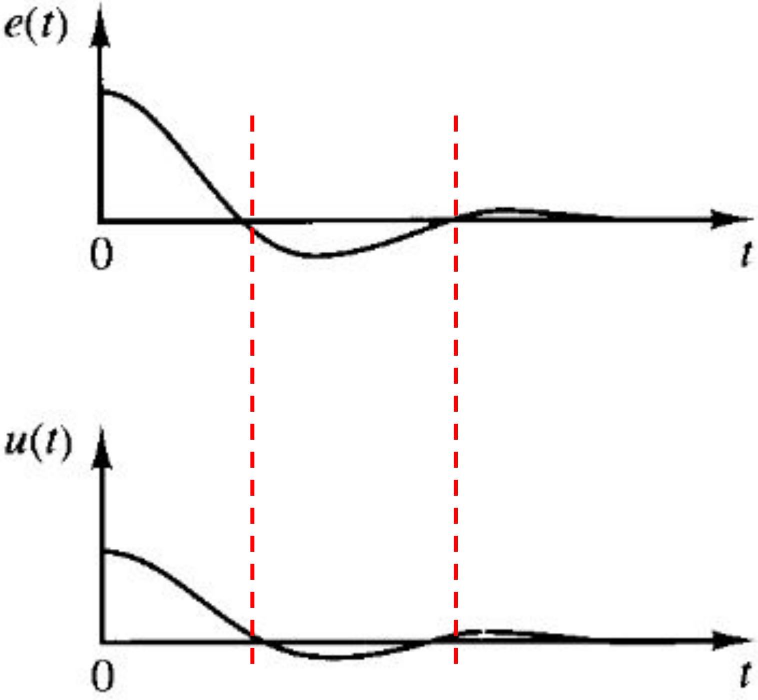
- In the integral control of a plant, the control signal, at any instant, is the area under the error signal curve upto that instant. So u(t) can have a non-zero value when error signal e(t) is zero.

- So, PI control eliminates steady-state error for step-input in this case, but may lead to oscillatory response.



(a)

Integral Control Action

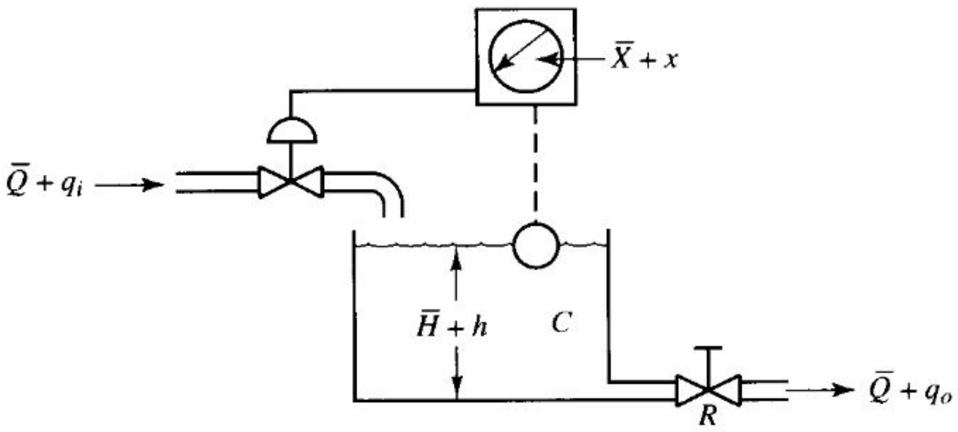


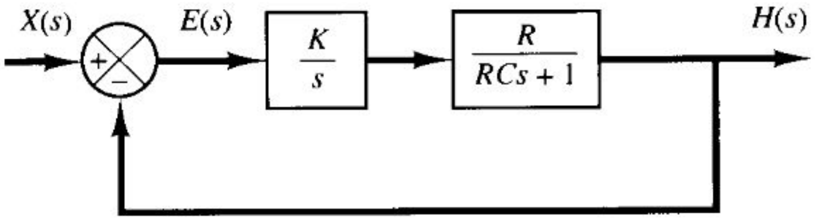
(b)

Proportional Control Action

Example: Liquid-level control System

$$\frac{H(s)}{X(s)} = \frac{KR}{RCs^2 + s + KR}$$





$$\frac{E(s)}{X(s)} = \frac{X(s) - H(s)}{X(s)}$$

$$= \frac{RCs^2 + s}{RCs^2 + s + KR}$$

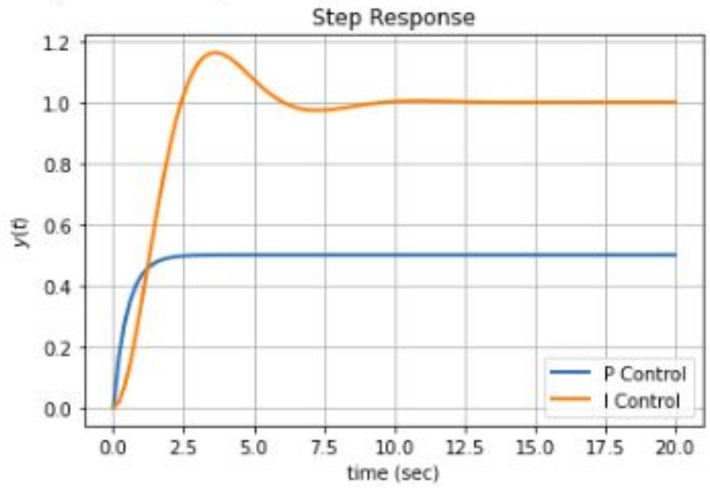
$$e_{ss} = \lim_{s \to 0} sE(s)$$

$$= \lim_{s \to 0} \frac{s(RCs^2 + s)}{RCs^2 + s + KR} \frac{1}{s}$$

$$= 0$$

```python
from control import *
import matplotlib.pyplot as plt
import numpy as np

# constants
K, R, C = 1, 1, 1

# plant
g = tf(1, [1, 1])
print('Open loop plant: ', g)

# controller
c = tf(1,[1, 0])

# closed-loop system
gc1 = feedback(g, 1, -1) # without integrator
print('Closed-loop system with proportional control:', gc1)

gc2 = feedback(series(g,c), 1, -1) # with integrator
print('Closed-loop system with Integral Control:', gc2)

t = np.linspace(0,20,100)
t,y1 = step_response(gc1, t)
t,y2 = step_response(gc2, t)

plt.plot(t, y1, lw = 2, label='P Control')
plt.plot(t, y2, lw = 2, label='I Control')
#plt.ylim((0,3))
plt.xlabel('time (sec)')
plt.ylabel('$y(t)$')
plt.title('Step Response')
plt.grid()
plt.legend(loc='best')
```

```
Open loop plant:
  1
-----
s + 1

Closed-loop system with proportional control:
  1
-----
s + 2

Closed-loop system with Integral Control:
     1
-----------
s^2 + s + 1

<matplotlib.legend.Legend at 0x7fe814305908>
```



Effect of Derivative Control Action on System Performance

- Derivative control action, when added to a proportional controller, provides a means of obtaining a controller with high sensitivity - faster response.

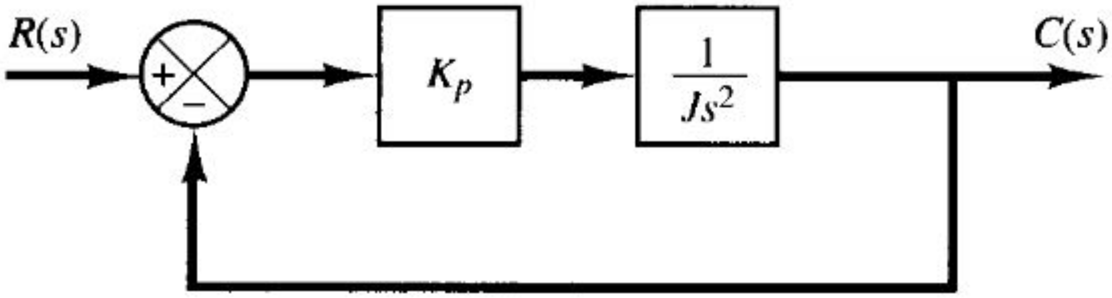
- Derivative control action responds to rate of change of actuating error and hence can produce significant correction before the actuating error becomes too large.

- Derivative control action is anticipatory in nature, initiates an early corrective action, tends to increase the stability of the system

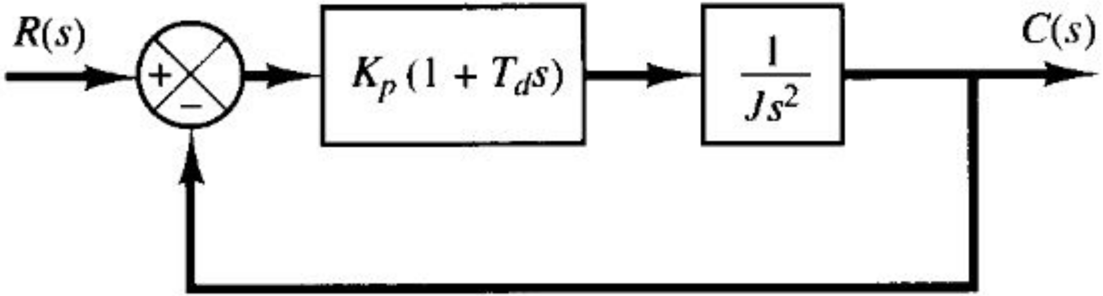
- Although derivative control action does not affect steady-state error directly, it adds damping to the system and thus permits the use of larger value of proportional gain which improves the steady-state accuracy.

- Derivative control action is never used alone. It is used in combination with Proportional or Proportional-plus-integral control action.

Example: Inertial load system



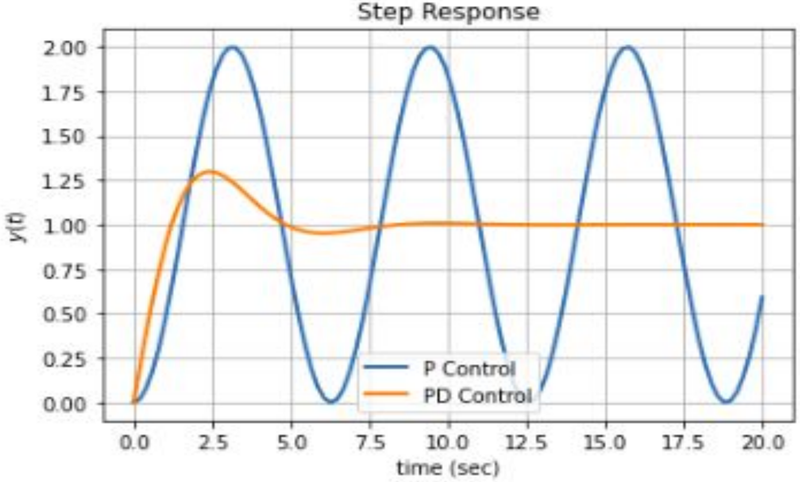
$$\frac{C(s)}{R(s)} = \frac{K_p}{Js^2 + K_p}$$



$$\frac{C(s)}{R(s)} = \frac{K_p(1 + T_d s)}{Js^2 + K_p T_d s + K_p}$$

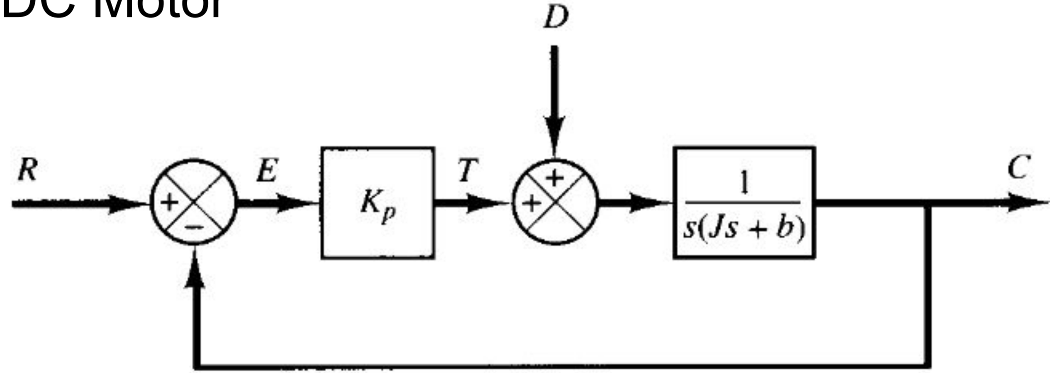
```python
from control import *
import matplotlib.pyplot as plt
import numpy as np

# constants
J, Kp, Kd = 1, 1, 1

# plant g = 1/Js^2
g = tf(1, [J, 0, 0])
print('Open loop plant: ', g)

# Prop controller
c1 = tf(Kp, [1])
# PD controller
c2 = tf([Kd, Kp],[1])

# closed-loop system
gc1 = feedback(series(c1, g), 1, -1)
print('Closed-loop system with proportional control:', gc1)

gc2 = feedback(series(c2,g), 1, -1)
print('Closed-loop system with PD Control:', gc2)

t = np.linspace(0,20,100)
t,y1 = step_response(gc1, t)
t,y2 = step_response(gc2, t)

plt.plot(t, y1, lw = 2, label='P Control')
plt.plot(t, y2, lw = 2, label='PD Control')
#plt.ylim((0,3))
plt.xlabel('time (sec)')
plt.ylabel('$y(t)$')
plt.title('Step Response')
plt.grid()
plt.legend(loc='best')
```

```
Open loop plant:
 1
---
s^2

Closed-loop system with proportional control:
    1
-------
s^2 + 1

Closed-loop system with PD Control:
    s + 1
-----------
s^2 + s + 1

<matplotlib.legend.Legend at 0x7f67cac64518>
```

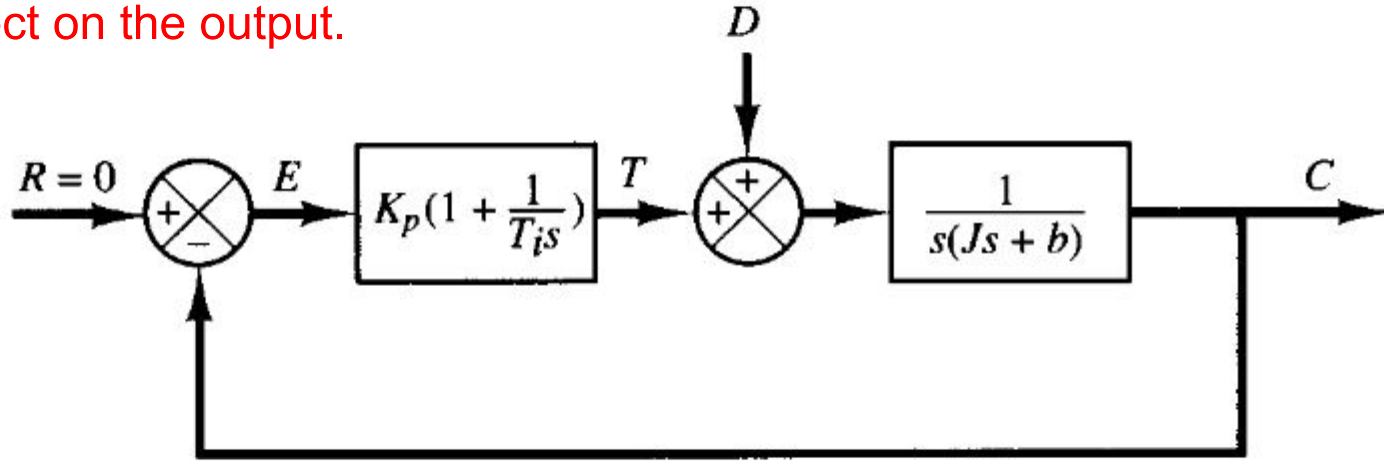


Disturbance Torque Rejection in DC Motor

$$\frac{C(s)}{D(s)} = \frac{1}{Js^2 + bs + K_p}$$



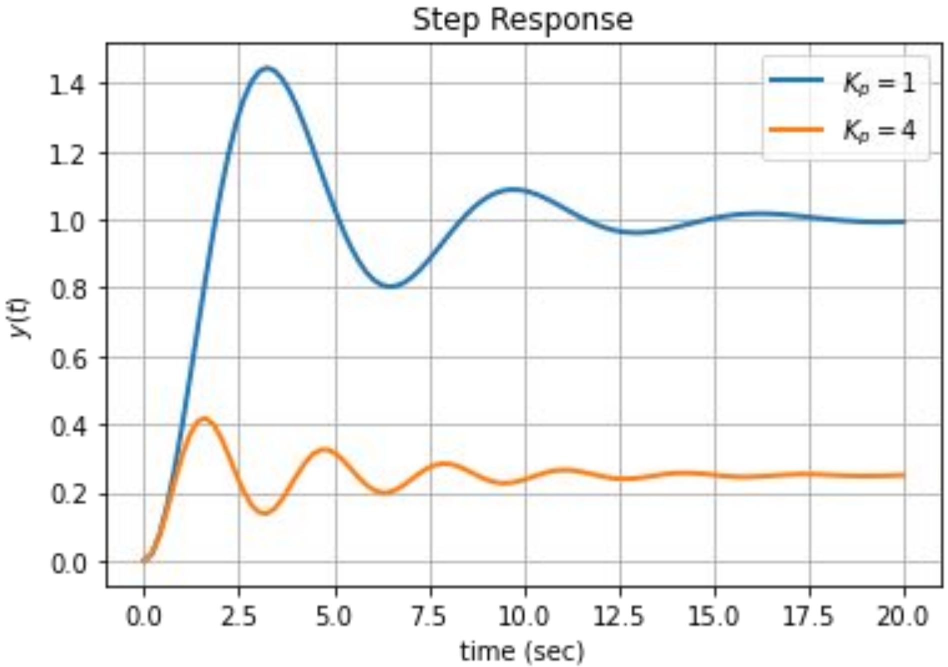
We apply a Step Input as a Disturbance Signal and the goal is to eliminate its effect on the output.



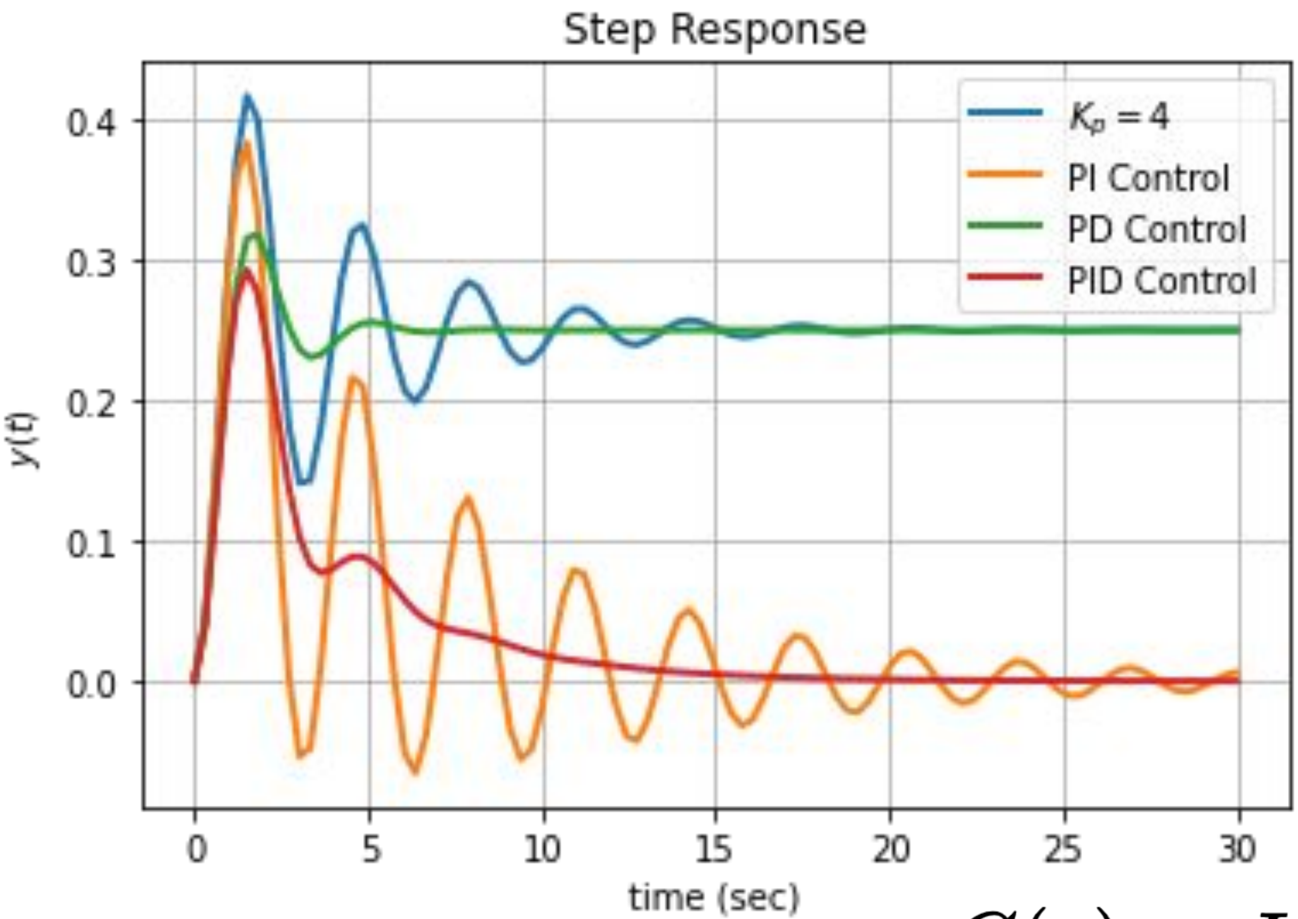
J = 1.0
B = 0.5
Kp = 1, 4
Ki = 0.8
Kd = 0.2

```python
from control import *
import matplotlib.pyplot as plt
import numpy as np

# constants
J, b = 1, 0.5

Kp1 = 1
Kp2 = 4

Ki = 0.8
Kd = 0.2

# plant
g = tf(1, [J, b, 0])
print('Open loop plant: ', g)

# PI controller
c1 = tf([Kp2, Ki],[1, 0])

# PD controller
c2 = tf([Kd, Kp2],[1])

# PID Controller
c3 = tf([Kd, Kp2, Ki],[1, 0])

# closed-loop system
gc1 = feedback(g, Kp1, -1) # prop cont
print('Closed-loop system with P control:', gc1)
gc2 = feedback(g, Kp2, -1)
print('Closed-loop system with P control:', gc2)

gc3 = feedback(g, c1, -1) # PI control
print('Closed-loop system with PI Controller:', gc3)

gc4 = feedback(g, c2, -1) # PD controller
print('Closed-loop system with PD Controller:', gc4)

gc5 = feedback(g, c3, -1) # PID controller
print('Closed-loop system with PID Controller:', gc5)

t = np.linspace(0,30,100)
t,y1 = step_response(gc1, t)
t,y2 = step_response(gc2, t)
t,y3 = step_response(gc3, t)
t,y4 = step_response(gc4, t)
t,y5 = step_response(gc5, t)

#plt.plot(t, y1, lw = 2, label='$K_p=1$')
plt.plot(t, y2, lw = 2, label='$K_p=4$')
plt.plot(t, y3, lw = 2, label='PI Control')
plt.plot(t, y4, lw = 2, label='PD Control')
plt.plot(t, y5, lw = 2, label='PID Control')
plt.xlabel('time (sec)')
plt.ylabel('$y(t)$')
plt.title('Step Response')
plt.grid()
plt.legend(loc='best')
```

```
Open loop plant:
     1
-----------
s^2 + 0.5 s

Closed-loop system with P control with Kp = 1:
       1
---------------
s^2 + 0.5 s + 1

Closed-loop system with P control with Kp = 4:
       1
---------------
s^2 + 0.5 s + 4

Closed-loop system with PI Controller:
             s
-------------------------
s^3 + 0.5 s^2 + 4 s + 0.8

Closed-loop system with PD Controller:
       1
---------------
s^2 + 0.7 s + 4

Closed-loop system with PID Controller:
             s
-------------------------
s^3 + 0.7 s^2 + 4 s + 0.8
```



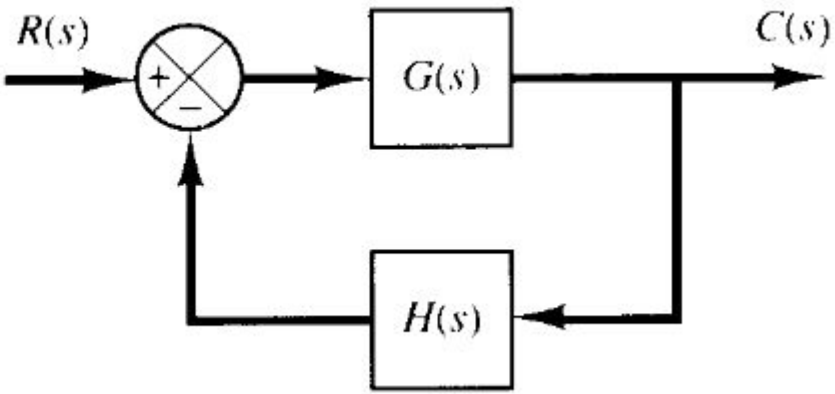
- In steady state, Output should go to zero at steady-state for step disturbance signal.
- Steady-State error reduces with increasing value of Kp.



Kp = 4
Ki = 1.0
Kd = 1.0
J = 1.0
B = 0.5

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

Higher-Order Systems



$$\frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)}$$

$$\frac{C(s)}{R(s)} = \frac{b_0 s^m + b_1 s^{m-2} + \ldots + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n}; m \leq n$$

$$\frac{C(s)}{R(s)} = \frac{K(s + z_1)(s + z_2) \cdots (s + z_m)}{(s + p_1)(s + p_2) \cdots (s + p_n)}$$

Case 1: Real and Distinct Poles

- For a step-input, C(s) can be written as

$$C(s) = \frac{a}{s} + \sum_{i=1}^{n} \frac{a_i}{s + p_i}$$

  Where a_i is the residue of the pole at s = -p_i.
- If all closed-loop poles lie in the left-half s-plane, then the relative magnitude of residues determine relative importance of the poles in the output response c(t).

- A pair of closely located poles and zeros cancel each other.

- If a pole is located far from the origin, its residue at this pole may be small and it will have little effect on the output transient response. Such terms may be neglected and a higher order system can be approximated by a lower order system.

Case II: A pair of Complex-Conjugate Poles

- A pair of complex-conjugate poles yields a second-order term.

- For a step input, the output may be written as

$$C(s) = \frac{K \prod_{i=1}^{m} (s + z_i)}{s \prod_{j=1}^{q} (s + p_j) \prod_{k=1}^{r} (s^2 + 2\zeta_k \omega_k s + \omega_k^2)}$$

- If the closed-loop poles are distinct, the output can be written as:

$$C(s) = \frac{a}{s} + \sum_{j=1}^{q} \frac{a_j}{s + p_j} + \sum_{k=1}^{r} \frac{b_k(s + \zeta_k \omega_k) + c_k \omega_k \sqrt{1 - \zeta_k^2}}{s^2 + 2\zeta_k \omega_k s + \omega_k^2}$$

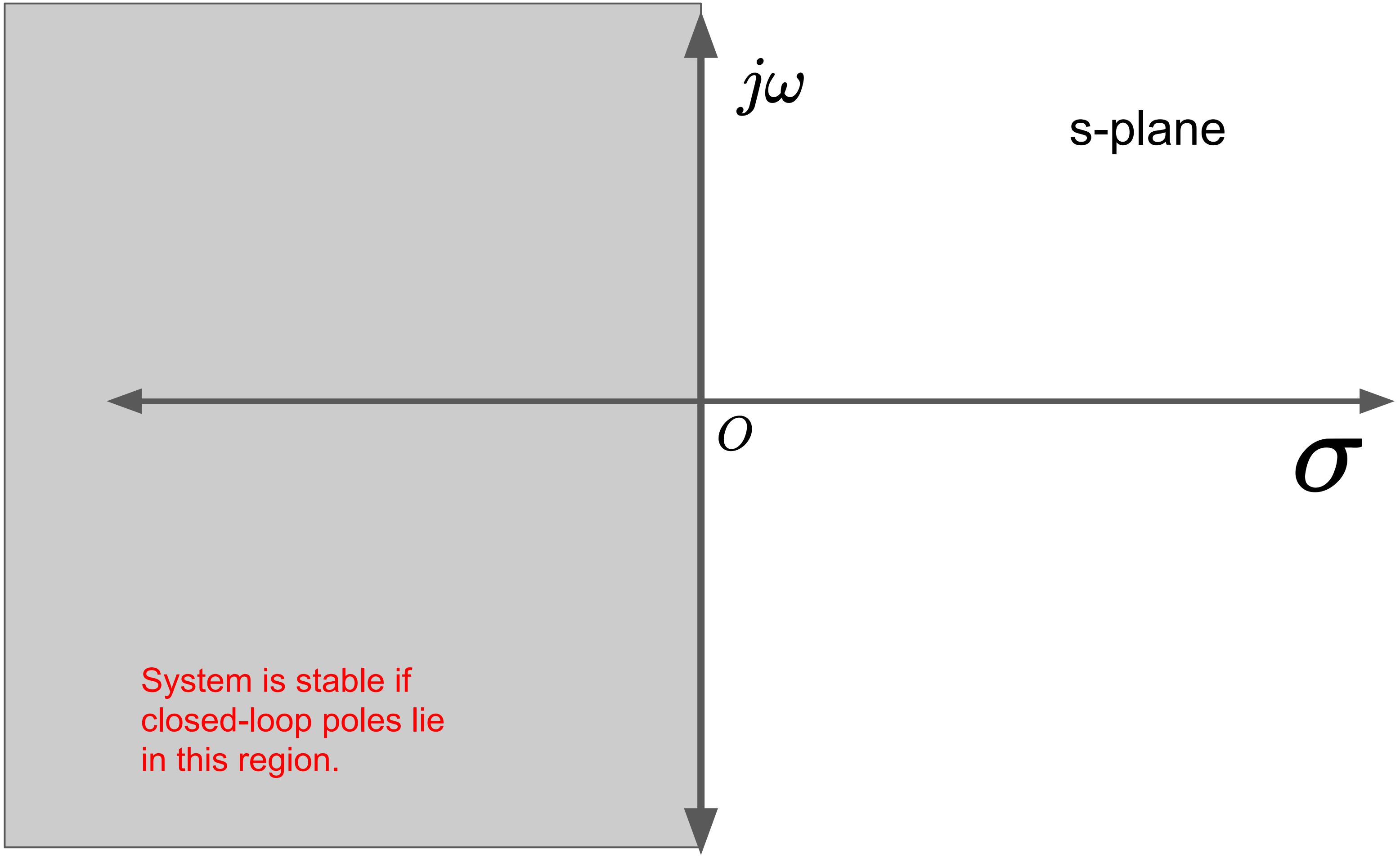
- The corresponding time response

$$c(t) = a + \sum_{j=1}^{q} a_j e^{-p_j t} + \sum_{k=1}^{r} b_k e^{-\zeta_k \omega_k t} \cos \omega_k \sqrt{1 - \zeta_k^2}\, t + \sum_{k=1}^{r} c_k e^{-\zeta_k \omega_k t} \sin \omega_k \sqrt{1 - \zeta_k^2}\, t, \qquad \text{for } t \geq 0$$

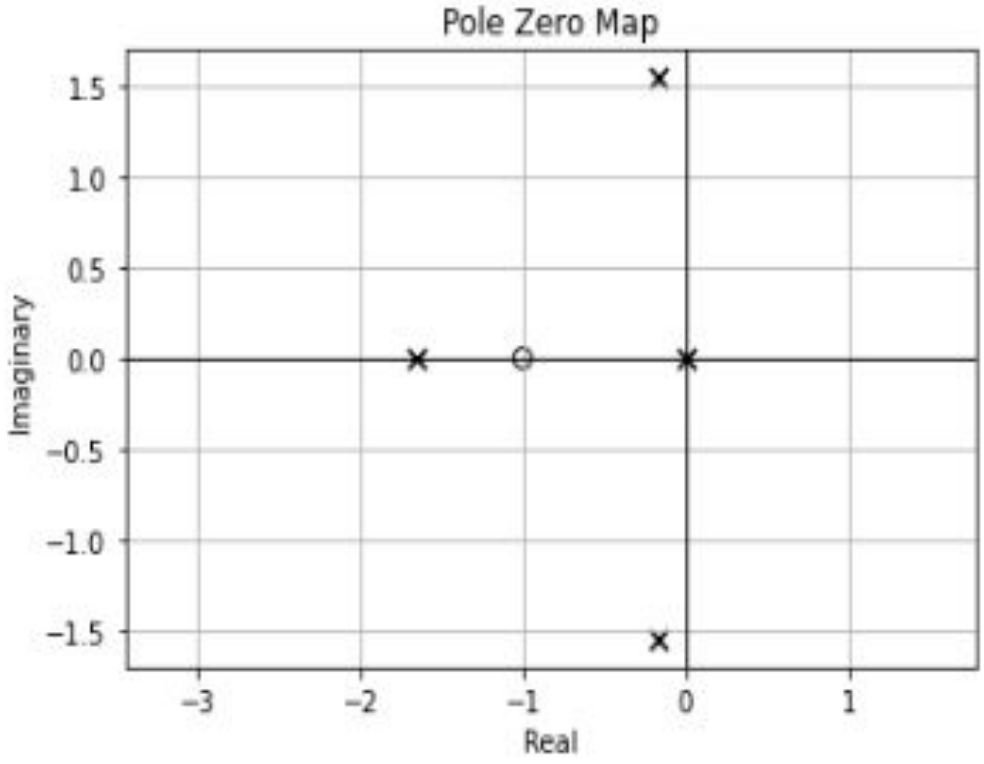
As time increases, $c(\infty) = a$

- The closed loop poles that are located far away from jw axis decay very rapidly to zero. The horizontal distance of a pole from the jw-axis determines the settling time of transients due to that pole.

- A HO system can be represented by a second order system comprising of a pair of complex-conjugate poles.

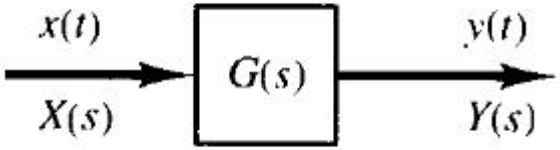

$j\omega$
s-plane
O
σ
System is stable if
closed-loop poles lie
in this region.

```
Poles: [-1.65062919+0.j         -0.1746854 +1.54686889j -0.1746854 -1.54686889j
  0.        +0.j        ]
Zeros: [-1.]
```




```python
from control import *
import matplotlib.pyplot as plt


g = tf([1,1],[1,2,3,4, 0])

print('Poles:',g.pole())

print('Zeros:', g.zero())

# Pole-Zero map
poles,zeros = pzmap(g)
ax = plt.gca()
ax.grid()
```

Phase Lead and Lag in Sinusoidal Response



$$x(t) = X \sin \omega t$$

$$Y(s) = G(s)X(s) = G(s)\frac{\omega X}{s^2 + \omega^2}$$

$$= \frac{a}{s + j\omega} + \frac{\bar{a}}{s - j\omega} + \frac{b_1}{s + s_1} + \frac{b_2}{s + s_2} + \cdots + \frac{b_n}{s + s_n}$$

$$y(t) = ae^{-j\omega t} + \bar{a}e^{j\omega t} + b_1 e^{-s_1 t} + b_2 e^{-s_2 t} + \cdots + b_n e^{-s_n t} \qquad (t \geq 0)$$

In the steady-state when $t \longrightarrow \infty$ $y_{ss}(t) = ae^{-j\omega t} + \bar{a}e^{j\omega t}$

Where the constants are computed as follows:

$$a = G(s)\frac{\omega X}{s^2 + \omega^2}(s + j\omega)\bigg|_{s=-j\omega} = -\frac{XG(-j\omega)}{2j}$$

$$\bar{a} = G(s)\frac{\omega X}{s^2 + \omega^2}(s - j\omega)\bigg|_{s=j\omega} = \frac{XG(j\omega)}{2j}$$

$$G(j\omega) = Me^{j\phi} = M\underline{/\phi}$$

$$G(j\omega) = |G(j\omega)|e^{j\phi}$$

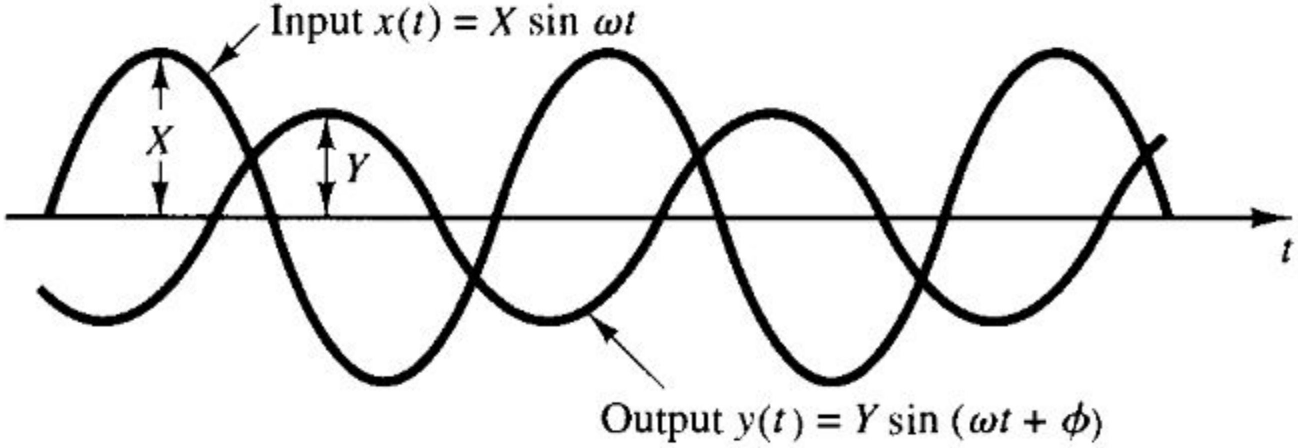
$$G(-j\omega) = |G(-j\omega)|e^{-j\phi} = |G(j\omega)|e^{-j\phi}$$

This gives,

$$a = -\frac{X|G(j\omega)|e^{-j\phi}}{2j}, \qquad \bar{a} = \frac{X|G(j\omega)|e^{j\phi}}{2j}$$

$$\begin{aligned} y_{ss}(t) &= X|G(j\omega)|\frac{e^{j(\omega t+\phi)} - e^{-j(\omega t+\phi)}}{2j} \\ &= X|G(j\omega)|\sin(\omega t + \phi) \\ &= Y\sin(\omega t + \phi) \end{aligned}$$

Where $Y = X|G(j\omega)|$.



For Sinusoidal inputs,

$$|G(j\omega)| = \left|\frac{Y(j\omega)}{X(j\omega)}\right| = \text{amplitude ratio of the output sinuisoid to the input sinusoid}$$

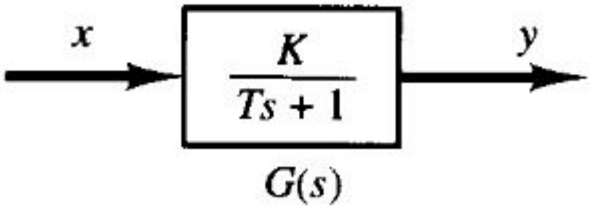
$$\angle G(j\omega) = \angle \frac{Y(j\omega)}{X(j\omega)} = \text{phase shift of the output sinusoid with respect to the input sinusoid}$$

- The response characteristics of a system to a sinusoidal input can be obtained directly from:

$$\frac{Y(j\omega)}{X(j\omega)} = G(j\omega)$$

And $G(j\omega)$ is called **sinusoidal transfer function**. It is a complex quantity represented by a magnitude and a phase. If the phase angle is positive, the system is a ***lead network*** and if the phase angle is negative, it is called a ***lag network***.

Example of Phase Lag Network



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

$$|G(j\omega)| = \frac{K}{\sqrt{1 + T^2\omega^2}}$$

$$G(j\omega) = \frac{K}{jT\omega + 1}$$

$$\phi = \underline{/G(j\omega)} = -\tan^{-1} T\omega$$

For sinusoidal input, the steady-state output is given by

$$y_{ss}(t) = \frac{XK}{\sqrt{1 + T^2\omega^2}} \sin(\omega t - \tan^{-1} T\omega)$$

$$\text{As } \omega \to \infty, \phi \to -90^\circ$$

Example of Phase Lead Network

$$G(j\omega) = \frac{j\omega + \frac{1}{T_1}}{j\omega + \frac{1}{T_2}} = \frac{T_2(1 + T_1 j\omega)}{T_1(1 + T_2 j\omega)}$$

$$G(s) = \frac{s + \frac{1}{T_1}}{s + \frac{1}{T_2}}$$

$$|G(j\omega)| = \frac{T_2\sqrt{1 + T_1^2\omega^2}}{T_1\sqrt{1 + T_2^2\omega^2}}$$

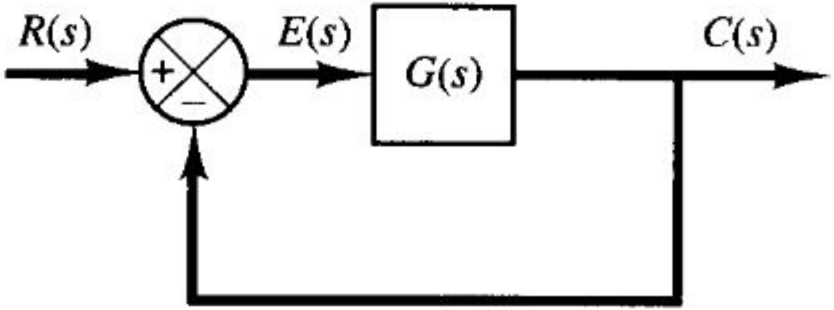
$$\phi = \angle G(j\omega) = \tan^{-1} T_1\omega - \tan^{-1} T_2\omega$$

$$y_{ss}(t) = \frac{XT_2\sqrt{1 + T_1^2\omega^2}}{T_1\sqrt{1 + T_2^2\omega^2}} \sin(\omega t + \tan^{-1} T_1\omega - \tan^{-1} T_2\omega)$$

Its a lead network if $T_1 > T_2$

Steady-State Error in Unity-Feedback Systems

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



$$\frac{E(s)}{R(s)} = 1 - \frac{C(s)}{R(s)} = \frac{1}{1 + G(s)}$$

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

Steady-state Error:

$$e_{ss} = \lim_{t \to \infty} e(t) = \lim_{s \to 0} sE(s) = \lim_{s \to 0} \frac{sR(s)}{1 + G(s)}$$

$$G(s) = \frac{K(T_a s + 1)(T_b s + 1)\cdots(T_m s + 1)}{s^N(T_1 s + 1)(T_2 s + 1)\cdots(T_p s + 1)}$$

A system is called type-0, 1, 2 …. If N = 0, 1, 2, ….

For a step-input:

$$e_{ss} = \lim_{s\to 0} \frac{s}{1 + G(s)} \frac{1}{s}$$

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

$$e_{ss} = \frac{1}{1 + K_p}$$

Define Position-Error Constant:

$$K_p = \lim_{s\to 0} G(s) = G(0)$$

For a type 0 system,

$$K_p = \lim_{s \to 0} \frac{K(T_a s + 1)(T_b s + 1) \cdots}{(T_1 s + 1)(T_2 s + 1) \cdots} = K$$

For a type 1 or higher system,

$$K_p = \lim_{s \to 0} \frac{K(T_a s + 1)(T_b s + 1) \cdots}{s^N (T_1 s + 1)(T_2 s + 1) \cdots} = \infty, \qquad \text{for } N \geq 1$$

For a step-input:

$$e_{ss} = \frac{1}{1 + K}, \qquad \text{for type 0 systems}$$

$$e_{ss} = 0, \qquad \text{for type 1 or higher systems}$$

For a Ramp Input:

$$e_{ss} = \lim_{s \to 0} \frac{s}{1 + G(s)} \frac{1}{s^2}$$

$$= \lim_{s \to 0} \frac{1}{sG(s)}$$

$$e_{ss} = \frac{1}{K_v}$$

Static Velocity error constant: $K_v = \lim_{s \to 0} sG(s)$

$$e_{ss} = \frac{1}{K_v} = \infty, \qquad \text{for type 0 systems}$$

$$e_{ss} = \frac{1}{K_v} = \frac{1}{K}, \qquad \text{for type 1 systems}$$

$$e_{ss} = \frac{1}{K_v} = 0, \qquad \text{for type 2 or higher systems}$$

For an acceleration input (unit-parabolic) input:

$$r(t) = \frac{t^2}{2}, \qquad \text{for } t \geq 0$$
$$= 0, \qquad \text{for } t < 0$$

Acceleration Error Constant:

$$K_a = \lim_{s \to 0} s^2 G(s)$$

$$e_{ss} = \lim_{s \to 0} \frac{s}{1 + G(s)} \frac{1}{s^3}$$
$$= \frac{1}{\lim_{s \to 0} s^2 G(s)}$$
$$e_{ss} = \frac{1}{K_a}$$

$$e_{ss} = \infty, \qquad \text{for type 0 and type 1 systems}$$
$$e_{ss} = \frac{1}{K}, \qquad \text{for type 2 systems}$$
$$e_{ss} = 0, \qquad \text{for type 3 or higher systems}$$

	Step Input $r(t) = 1$	Ramp Input $r(t) = t$	Acceleration Input $r(t) = \frac{1}{2}t^2$
Type 0 system	$\frac{1}{1 + K}$	∞	∞
Type 1 system	0	$\frac{1}{K}$	∞
Type 2 system	0	0	$\frac{1}{K}$

Steady State Error in terms of Gain K

Summary

We studied the following in this lecture:

- How various control actions influence the system behaviour?
- In particular, we see the effect of P, PI, PD and PID controllers.
- Analyze the response of higher-order systems
- Steady-state response of LTI systems - Phase lead / lag system
- Compute Steady-state error for various inputs: step, ramp and acceleration.
- In the lab session, we will write a few python programs to better understand the concepts presented in this lecture.