Transient Analysis of Linear Systems

Lecture 4

Overview

- Specifying Parameters for Comparing Control Systems.
- Transient Performance Parameters for
    - First Order Systems
    - Second Order Systems

Introduction

- In analyzing and designing control systems, we must have a basis for comparison of performance of various control systems.

- The basis is set by specifying particular test signals and by comparing the responses of various systems to these input signals.

- Typical test signals: Step function, ramp functions, impulse functions, sinusoidal functions etc.

- The time response of a control system consists of two parts: the transient and the steady-state response.
  - Transient response: how system goes from an initial state to the final state.
  - Steady-state response: how system output behaves as t approaches infinity.

- A control system is ***in equilibrium*** if, in the absence of any disturbance or input, the output stays in the same state.

- A LTI system is ***stable*** if the output eventually comes back to its equilibrium state when the system is subjected to an initial condition.

- A LTI system is ***critically stable*** if oscillations of the output continue forever.

- A LTI system is ***unstable*** if the output diverges without bound from its equilibrium state when the system is subjected to an initial condition.

- It is also important to analyze ***relative stability*** and ***steady-state error*** for analyzing system performance.

First-Order System

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

T is called the time-constant

Unit-step response of first-order systems:

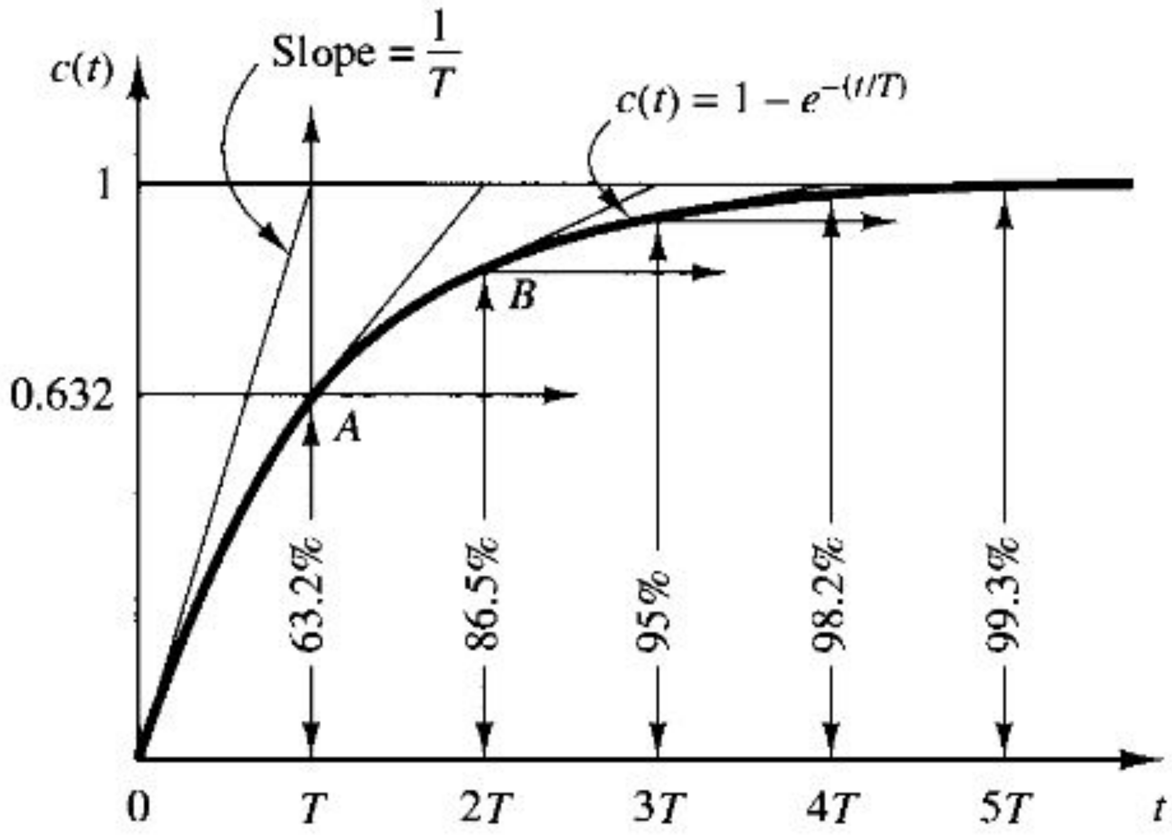
$$C(s) = \frac{1}{Ts + 1}\frac{1}{s}$$

$$C(s) = \frac{1}{s} - \frac{T}{Ts + 1} = \frac{1}{s} - \frac{1}{s + (1/T)}$$

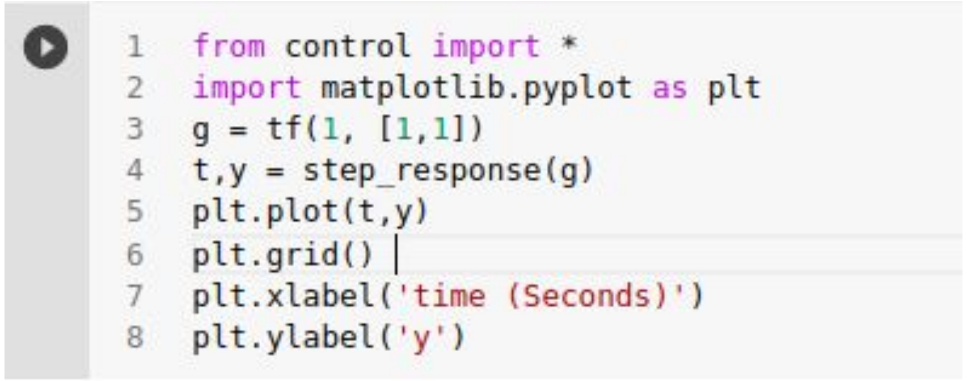
$$c(t) = 1 - e^{-t/T}, \qquad \text{for } t \geq 0$$

$$c(T) = 1 - e^{-1} = 0.632$$

$$c(\infty) = \lim_{s=0} sC(s) = 1$$

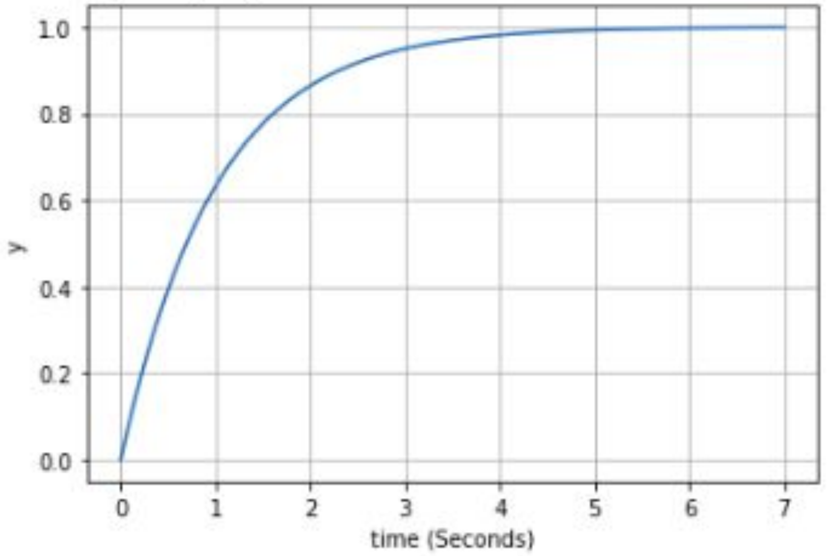


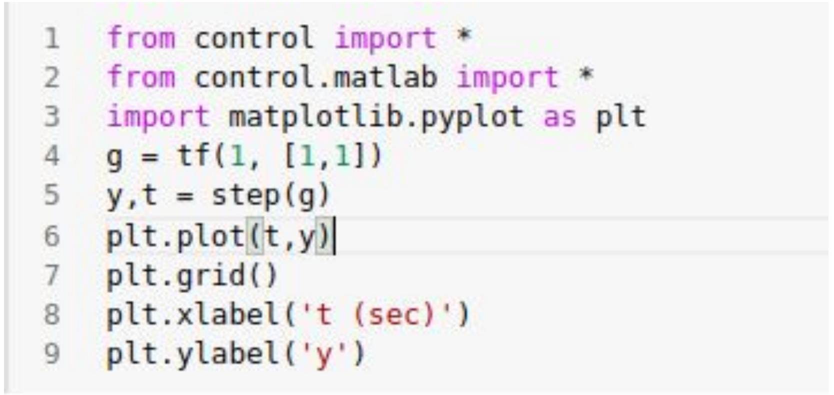
Smaller time-constants will have faster response.



```python
from control import *
import matplotlib.pyplot as plt
g = tf(1, [1,1])
t,y = step_response(g)
plt.plot(t,y)
plt.grid()
plt.xlabel('time (Seconds)')
plt.ylabel('y')
```

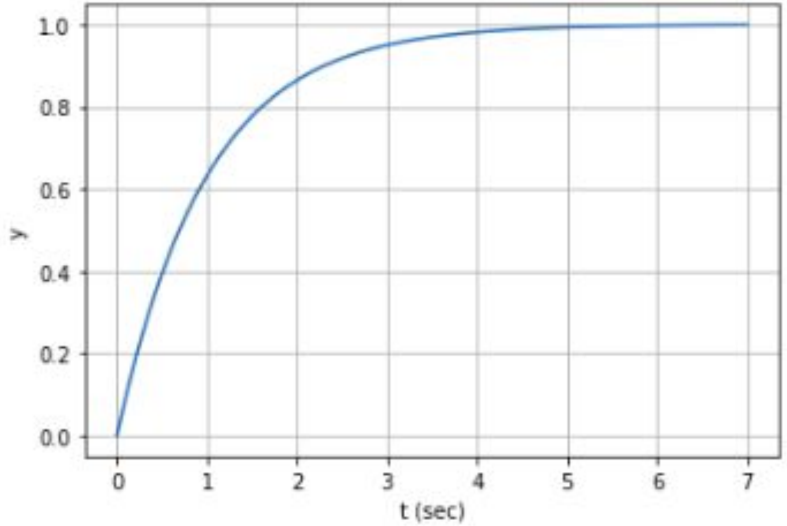
```
Text(0, 0.5, 'y')
```





```python
from control import *
from control.matlab import *
import matplotlib.pyplot as plt
g = tf(1, [1,1])
y,t = step(g)
plt.plot(t,y)
plt.grid()
plt.xlabel('t (sec)')
plt.ylabel('y')
```

```
Text(0, 0.5, 'y')
```



- Unit ramp response

$$C(s) = \frac{1}{Ts+1}\frac{1}{s^2}$$

By Partial Fraction Expansion:

$$C(s) = \frac{1}{s^2} - \frac{T}{s} + \frac{T^2}{Ts+1}$$

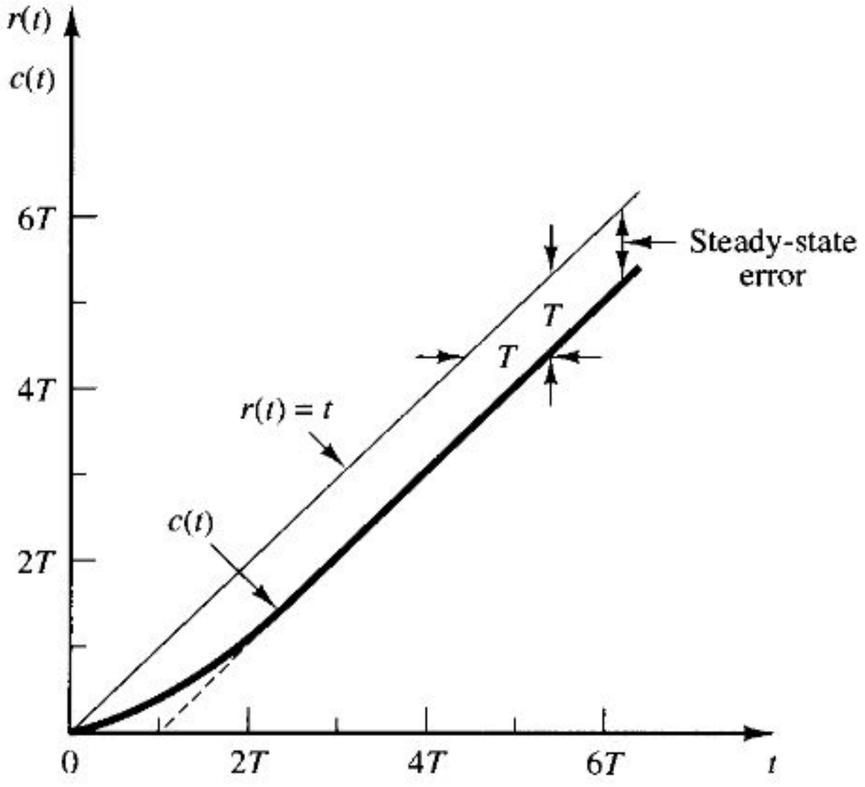
$$c(t) = t - T + Te^{-t/T}, \qquad \text{for } t \geq 0$$

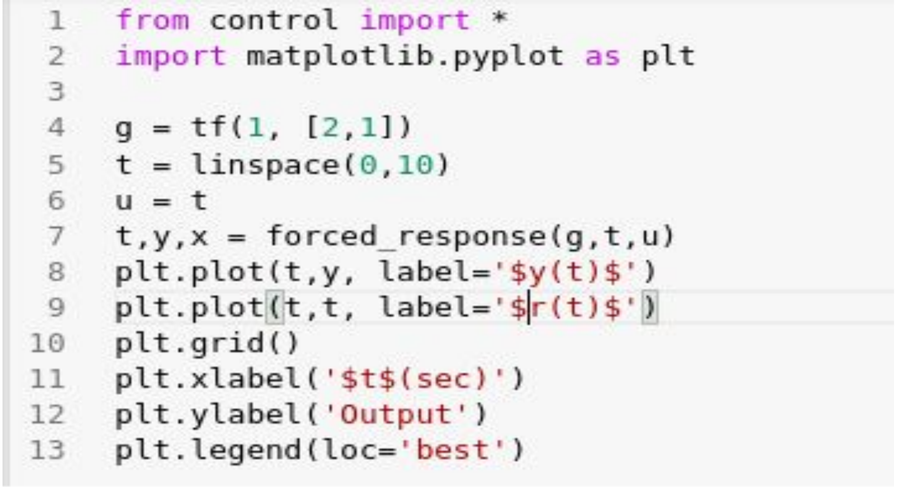
Error Signal:

$$\begin{aligned} e(t) &= r(t) - c(t) \\ &= T(1 - e^{-t/T}) \end{aligned}$$

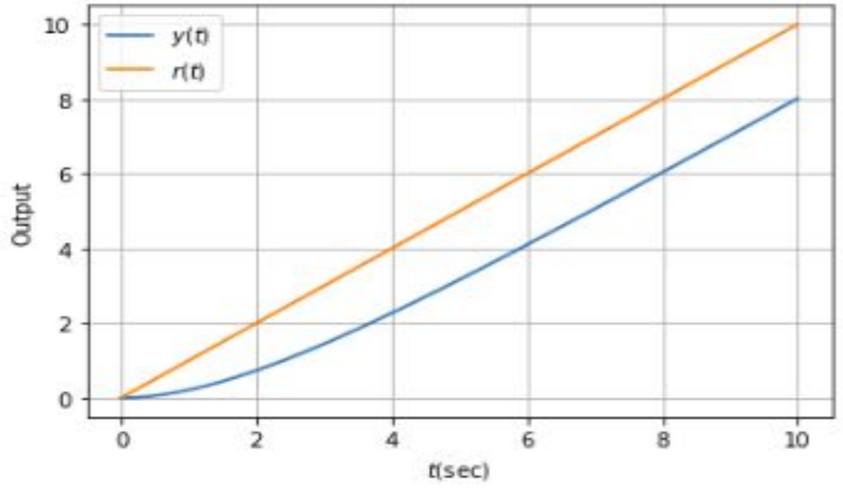
Steady-state error:

$$e(\infty) = T$$




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

g = tf(1, [2,1])
t = linspace(0,10)
u = t
t,y,x = forced_response(g,t,u)
plt.plot(t,y, label='$y(t)$')
plt.plot(t,t, label='$r(t)$')
plt.grid()
plt.xlabel('$t$(sec)')
plt.ylabel('Output')
plt.legend(loc='best')
```

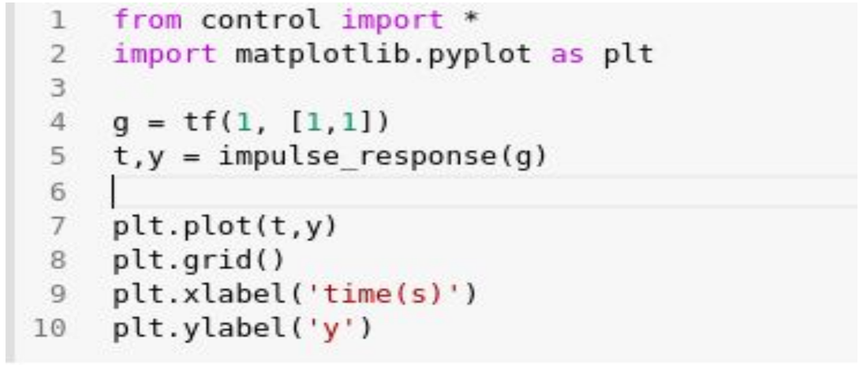
```
<matplotlib.legend.Legend at 0x7f84f4785d68>
```



- Unit Impulse Response of First Order Systems

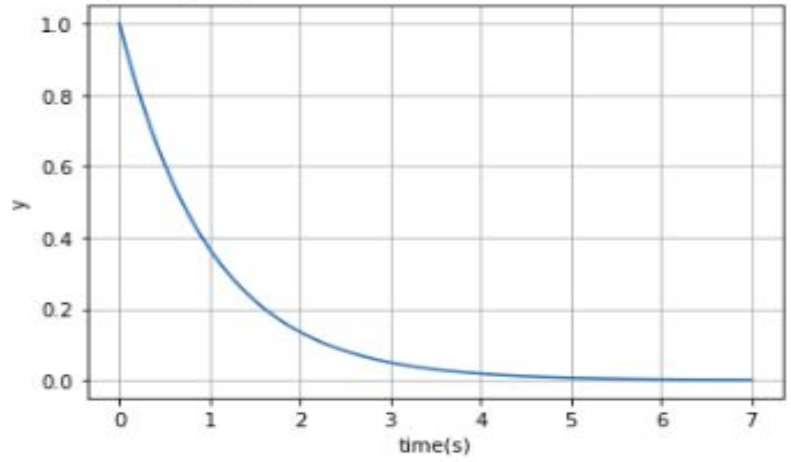
$$C(s) = \frac{1}{Ts + 1} \qquad \text{as } R(s) = 1$$

$$c(t) = \frac{1}{T} e^{-t/T}, \qquad \text{for } t \geq 0$$



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

g = tf(1, [1,1])
t,y = impulse_response(g)

plt.plot(t,y)
plt.grid()
plt.xlabel('time(s)')
plt.ylabel('y')
```

```
Text(0, 0.5, 'y')
```



Important Property of Linear Time Invariant (LTI) Systems

- Response to the derivative of an input signal can be obtained by differentiating the response of the system to the original signal.

- Response to the integral of an input signal can be obtained by integrating the response of the system to the original signal and determining the integration constants from zero output initial conditions.

- These properties are not valid for LTV or nonlinear systems.

Second-Order Systems

- Transfer function for a typical second order system is given by

$$\frac{C(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

Where ω_n is the undamped natural frequency and ζ is the damping ratio.

- For an unit-step input, the output becomes:

$$C(s) = \frac{\omega_n^2}{(s^2 + 2\zeta\omega_n s + \omega_n^2)s}$$

- Under-damped case: $0 < \zeta < 1$

  Damping frequency:

$$\omega_d = \omega_n \sqrt{1 - \zeta^2}$$

The output oscillates around its final (steady-state) value and oscillations damp out over time.

$$\begin{aligned}
\mathcal{L}^{-1}[C(s)] &= c(t) \\
&= 1 - e^{-\zeta\omega_n t}\left(\cos \omega_d t + \frac{\zeta}{\sqrt{1-\zeta^2}} \sin \omega_d t\right) \\
&= 1 - \frac{e^{-\zeta\omega_n t}}{\sqrt{1-\zeta^2}} \sin\left(\omega_d t + \tan^{-1} \frac{\sqrt{1-\zeta^2}}{\zeta}\right), \qquad \text{for } t \geq 0
\end{aligned}$$

- Undamped Case: $\zeta = 0$

$$c(t) = 1 - \cos \omega_n t, \qquad \text{for } t \geq 0$$

Undamped oscillations continue indefinitely.

- Critical Damped Case: $\zeta = 1$

$$C(s) = \frac{\omega_n^2}{(s + \omega_n)^2 s} \qquad c(t) = 1 - e^{-\omega_n t}(1 + \omega_n t), \qquad \text{for } t \geq 0$$

Output is an exponential function without any oscillations.

- Overdamped Case: $\zeta > 1$

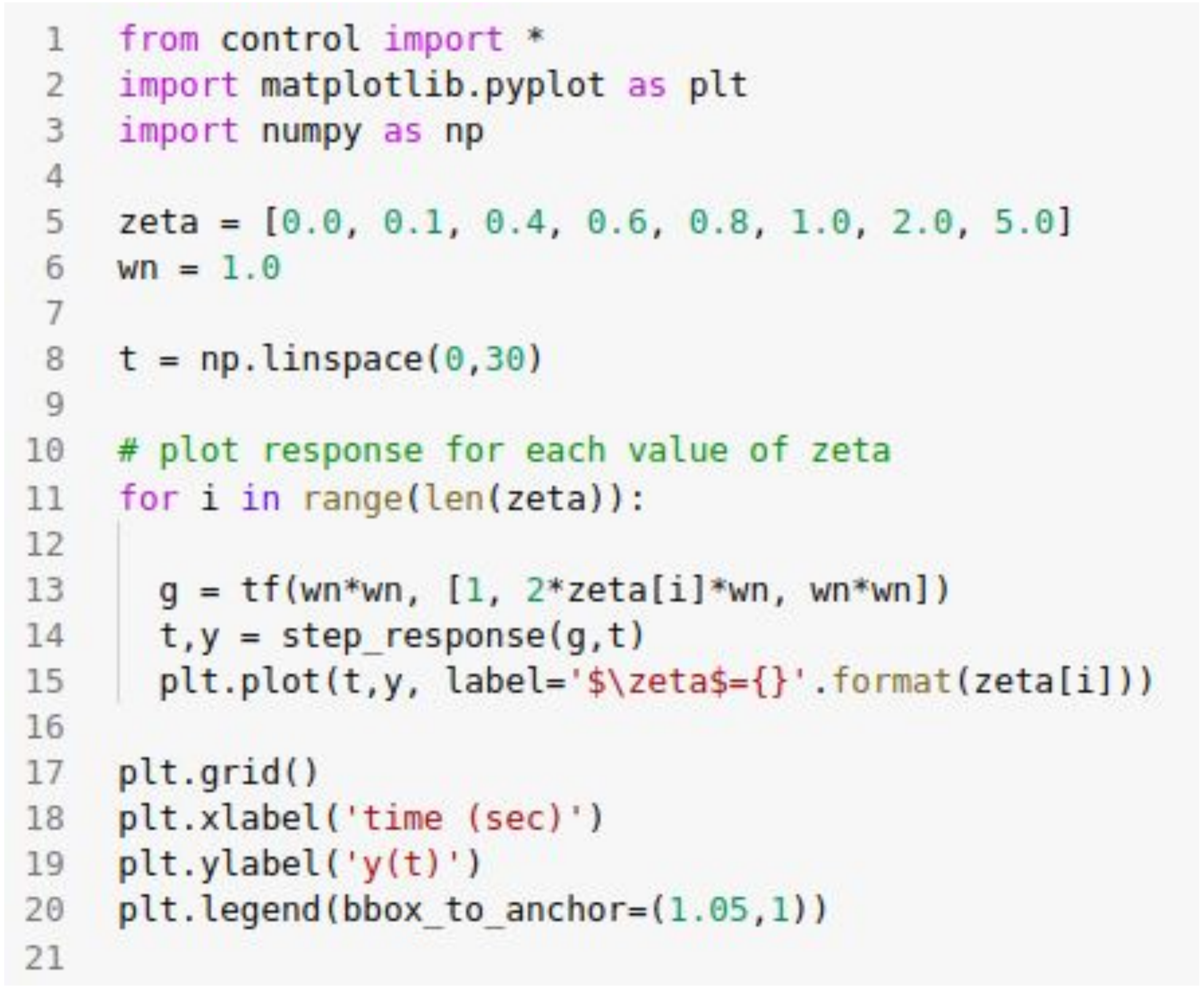
$$C(s) = \frac{\omega_n^2}{(s + \zeta\omega_n + \omega_n\sqrt{\zeta^2 - 1})(s + \zeta\omega_n - \omega_n\sqrt{\zeta^2 - 1})s}$$

$$c(t) = 1 + \frac{1}{2\sqrt{\zeta^2 - 1}(\zeta + \sqrt{\zeta^2 - 1})} e^{-(\zeta + \sqrt{\zeta^2 - 1})\omega_n t}$$

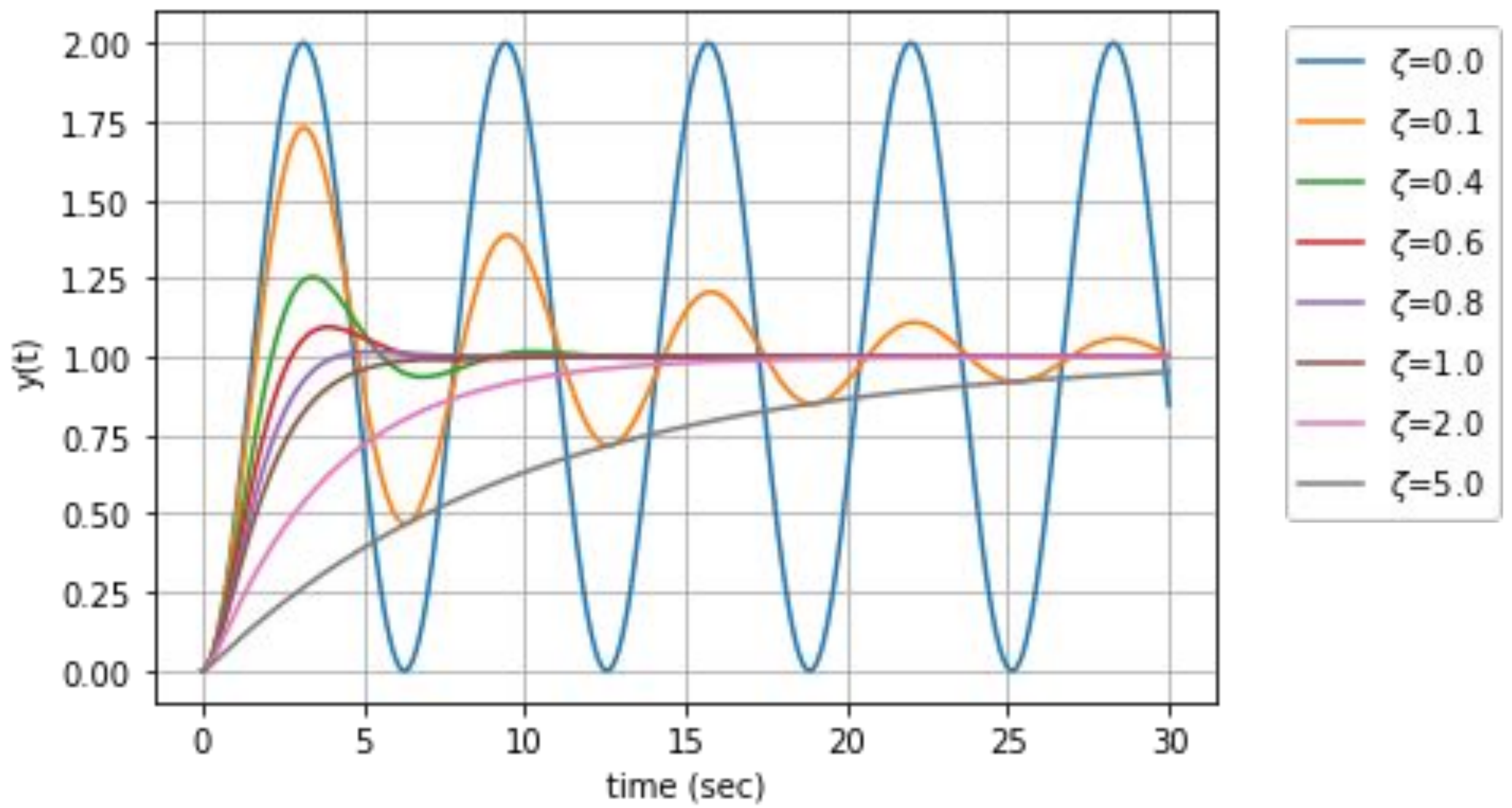
$$- \frac{1}{2\sqrt{\zeta^2 - 1}(\zeta - \sqrt{\zeta^2 - 1})} e^{-(\zeta - \sqrt{\zeta^2 - 1})\omega_n t}$$

$$= 1 + \frac{\omega_n}{2\sqrt{\zeta^2 - 1}}\left(\frac{e^{-s_1 t}}{s_1} - \frac{e^{-s_2 t}}{s_2}\right), \qquad \text{for } t \geq 0$$

where $s_1 = (\zeta + \sqrt{\zeta^2 - 1})\omega_n$ and $s_2 = (\zeta - \sqrt{\zeta^2 - 1})\omega_n$.



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

zeta = [0.0, 0.1, 0.4, 0.6, 0.8, 1.0, 2.0, 5.0]
wn = 1.0

t = np.linspace(0,30)

# plot response for each value of zeta
for i in range(len(zeta)):

    g = tf(wn*wn, [1, 2*zeta[i]*wn, wn*wn])
    t,y = step_response(g,t)
    plt.plot(t,y, label='$\zeta$={}'.format(zeta[i]))

plt.grid()
plt.xlabel('time (sec)')
plt.ylabel('y(t)')
plt.legend(bbox_to_anchor=(1.05,1))

```



2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
y(t)
0
5
10
15
20
25
30
time (sec)
ζ=0.0
ζ=0.1
ζ=0.4
ζ=0.6
ζ=0.8
ζ=1.0
ζ=2.0
ζ=5.0

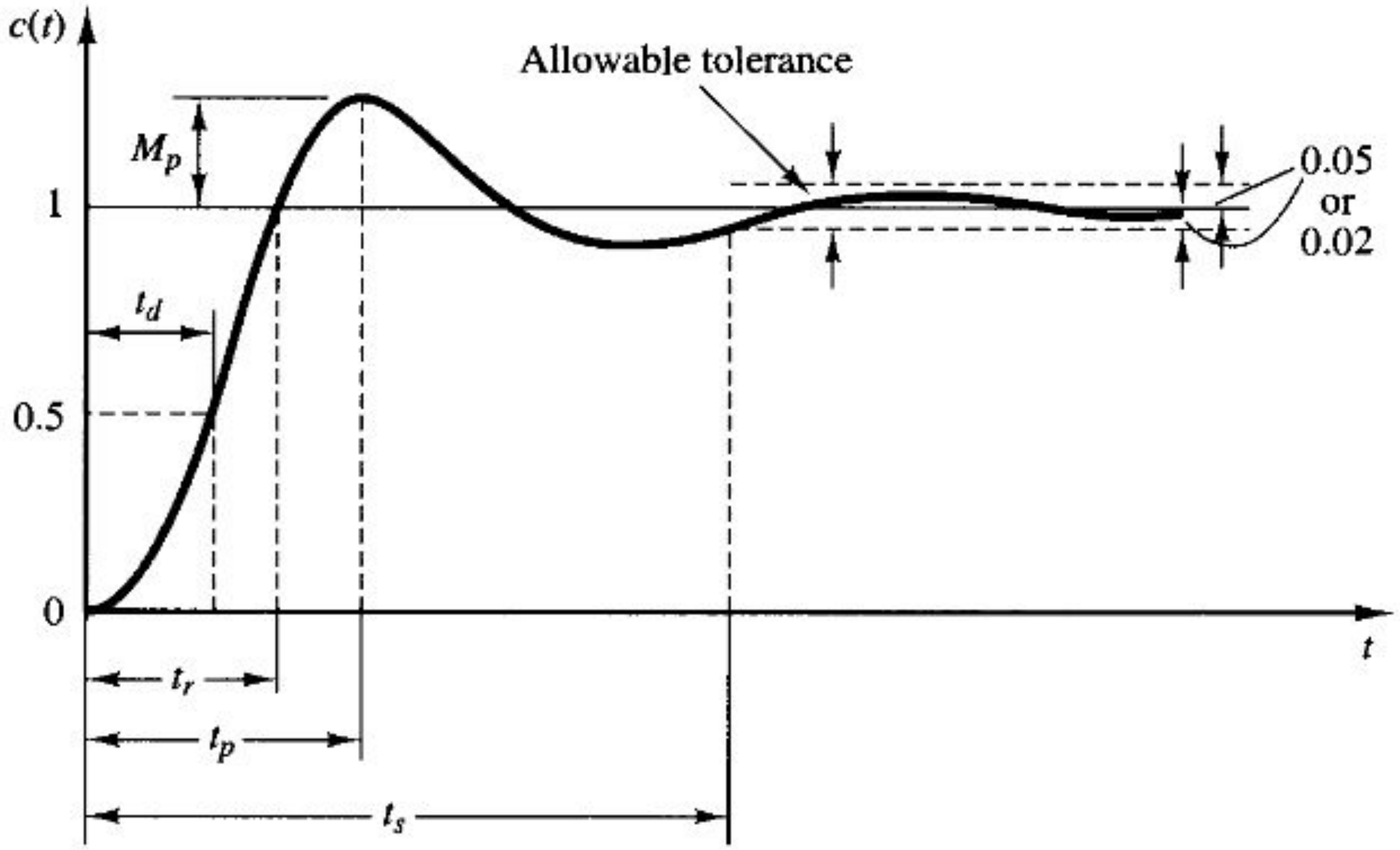
Transient-Response Specifications

- Most of the practical systems with energy storage devices can not respond instantaneously and will exhibit transient responses when subjected to inputs or disturbances.

- Performance characteristics of a control system are specified in terms of transient response to a ***unit-step function***.

- For convenience it is assumed that the initial conditions are at rest (zero).

- Transient response characteristics to a unit-step input is specified in terms of the following quantities:
  - (1) Delay time, (2) Rise time, (3) Peak time, (4) Maximum Overshoot and (5) Settling time


$c(t)$
Allowable tolerance
M_p
1
0.5
0
t_d
0.05
or
0.02
t
t_r
t_p
t_s

t_d

- **Delay time**: Time required for the response to reach half the final value the very first time.
- **Rise time**: Time required for the response to rise from 0% to 100% of the final value.
- **Peak time**: Time required for the response to reach the first peak of the overshoot.
- **Maximum (percent) overshoot**: Maximum peak value of the response curve measured from unity.

$$\text{Maximum percent overshoot} = \frac{c(t_p) - c(\infty)}{c(\infty)} \times 100\%$$

- **Settling time**: Time required for the response curve to reach or stay within a range of 2% or 5% of the final value.

Rise time: $t_r = \dfrac{\pi - \beta}{\omega_d}$

$$\omega_d = \omega_n\sqrt{1 - \zeta^2}$$

$$\sigma = \zeta\omega_n$$

$$\beta = \tan^{-1}\frac{\omega_d}{\sigma}$$

Peak time: $t_p = \dfrac{\pi}{\omega_d}$

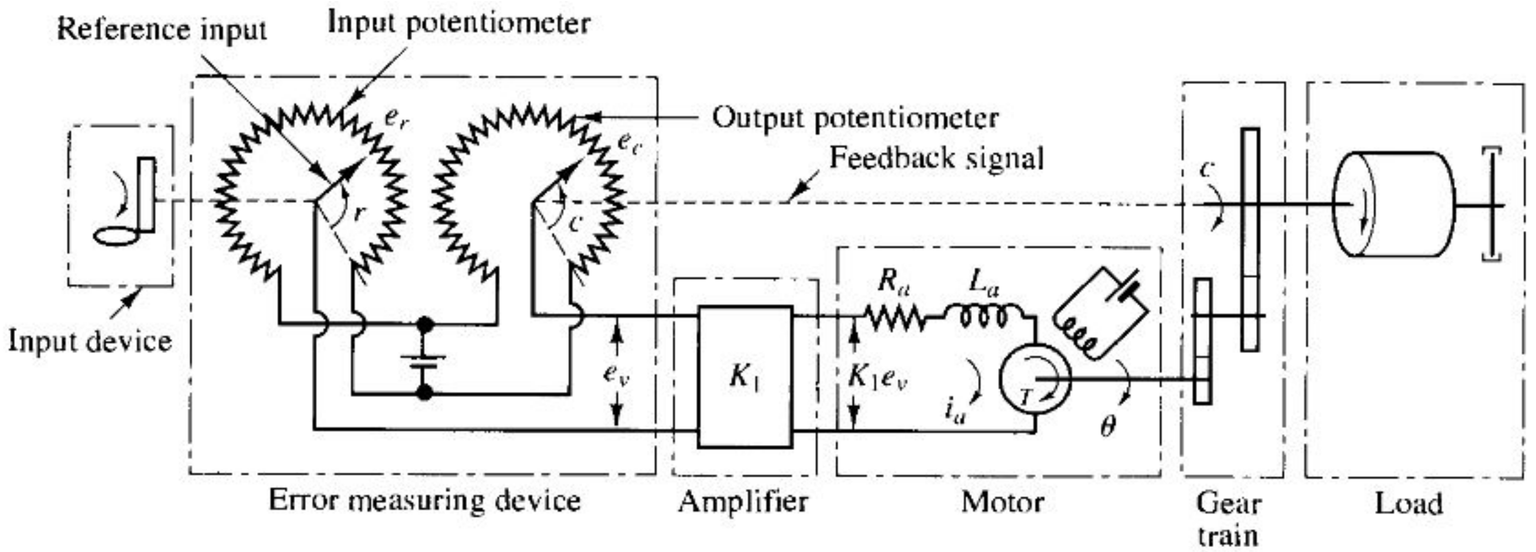
Maximum Overshoot: $M_p = e^{-(\zeta/\sqrt{1-\zeta^2})\pi}$ $\qquad M_p = e^{-(\sigma/\omega_d)\pi}$

$$t_s = 4T = \frac{4}{\sigma} = \frac{4}{\zeta\omega_n} \qquad \text{(2\% criterion)}$$

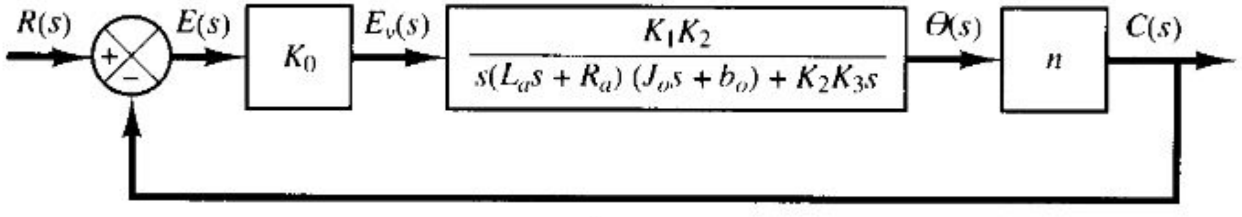
Settling time:

$$t_s = 3T = \frac{3}{\sigma} = \frac{3}{\zeta\omega_n} \qquad \text{(5\% criterion)}$$

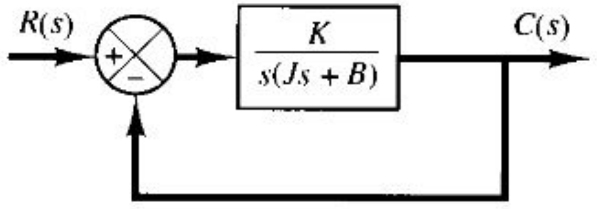
DC Motor Control (Servo Control)



(a)



(b)



(c)

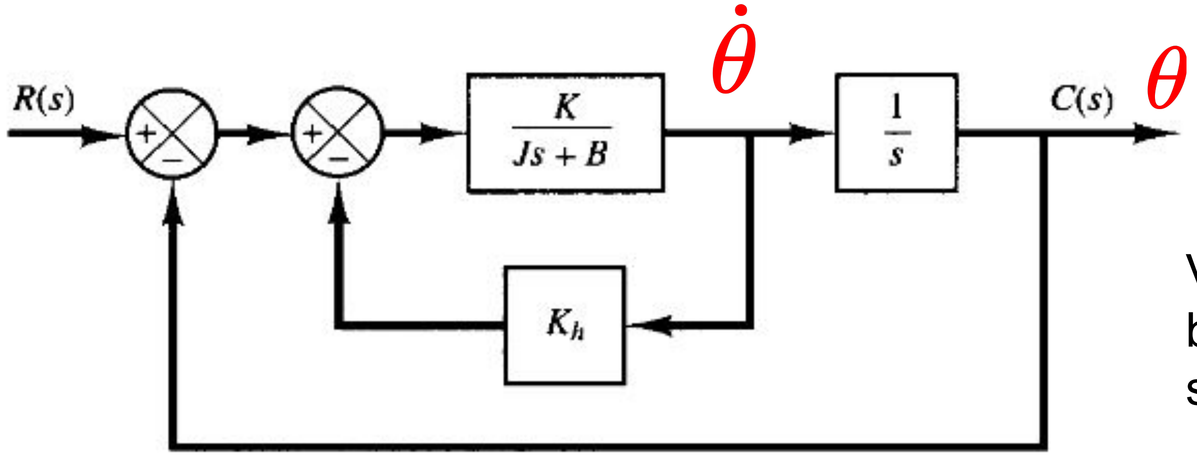
- The purpose of a servo system is to control the position of a mechanical load in accordance with the reference position.

- The error between the reference position and the current position generates a voltage that drives the motor which in turn moves the load through a gear train.

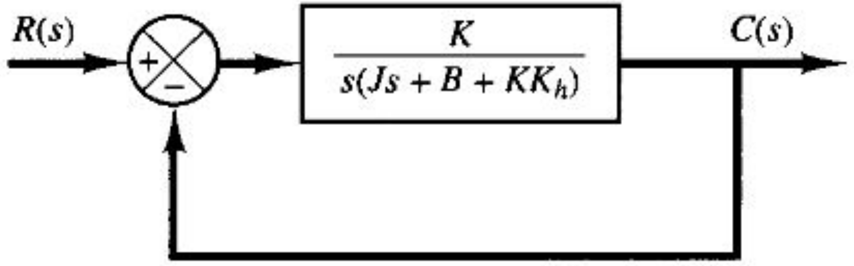
- The task of feedback control is to minimize the error between the reference position and the current position.

- Where is it used? - Electric Vehicles

Servo Control with Velocity Feedback

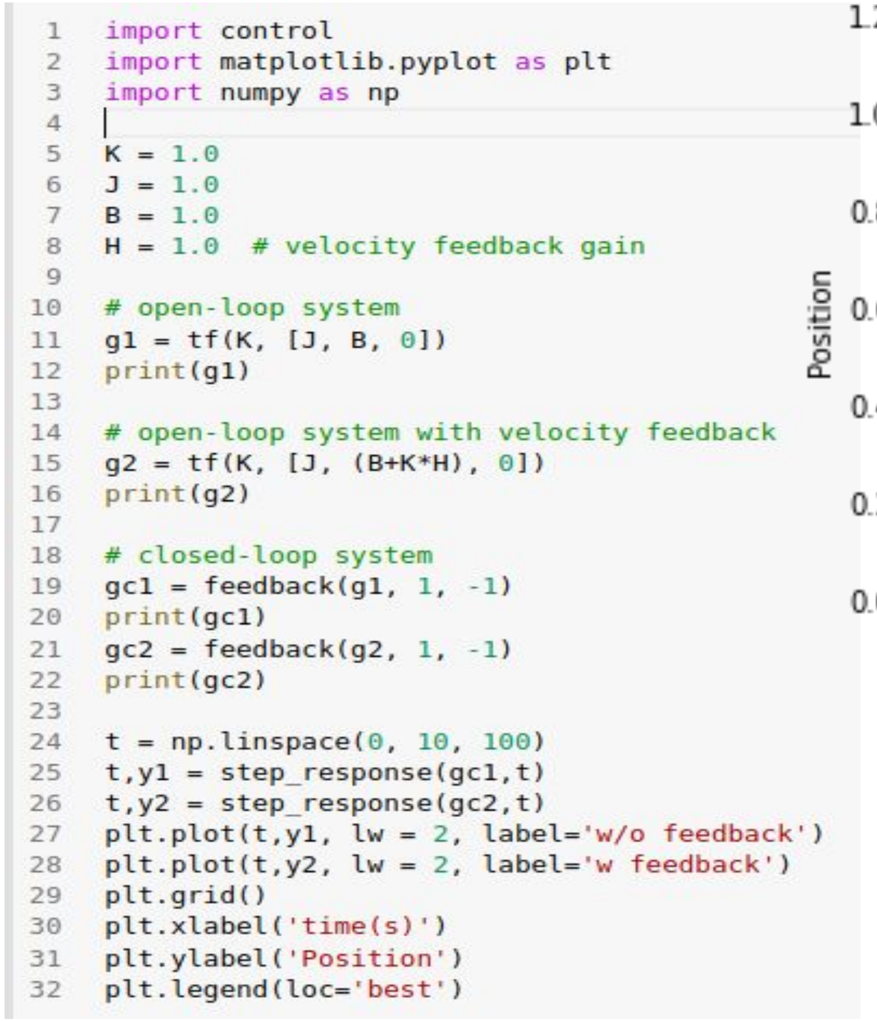


(a)

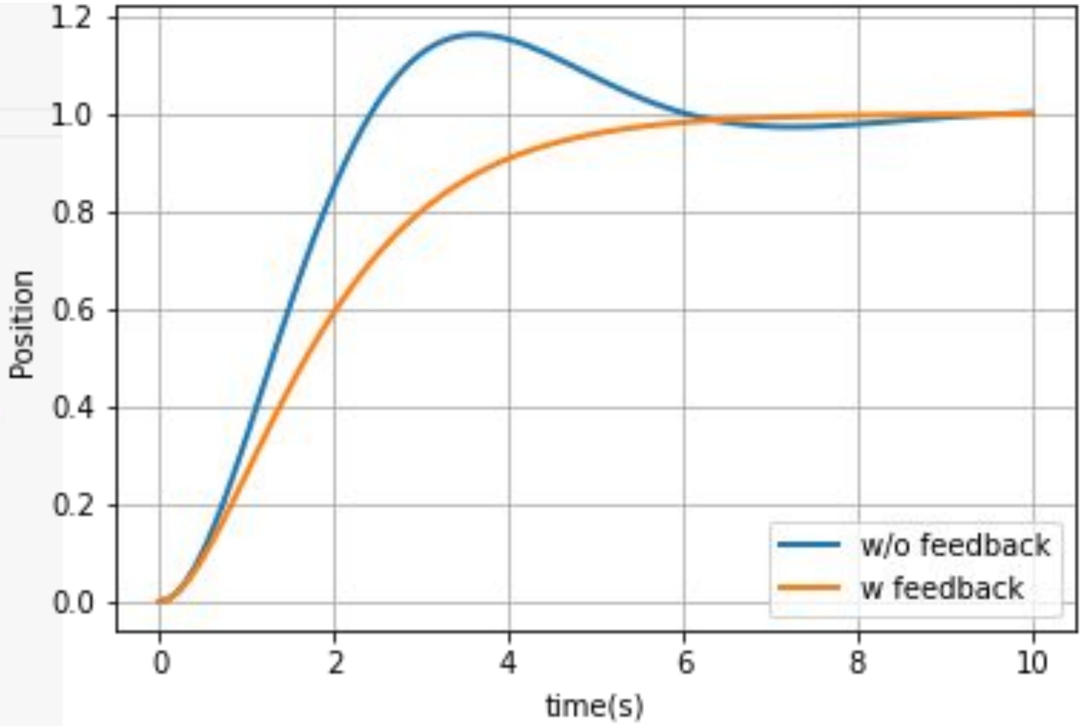


(b)

Velocity feedback can be used to improve system performance.

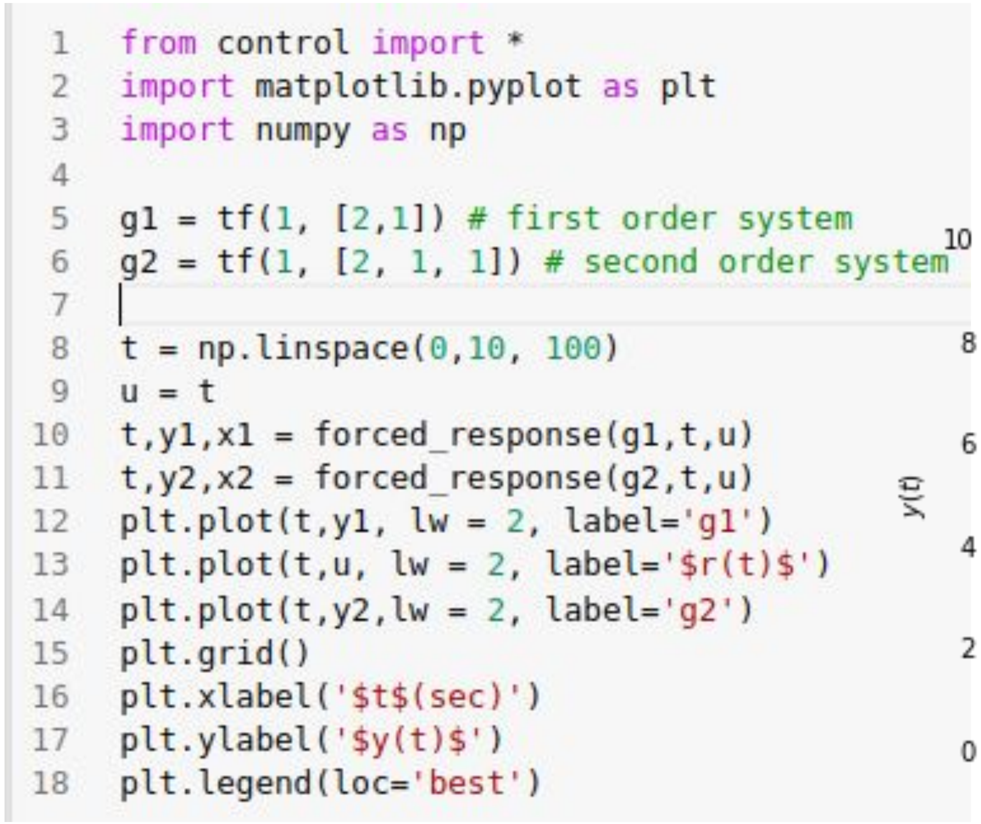


```python
import control
import matplotlib.pyplot as plt
import numpy as np

K = 1.0
J = 1.0
B = 1.0
H = 1.0  # velocity feedback gain

# open-loop system
g1 = tf(K, [J, B, 0])
print(g1)

# open-loop system with velocity feedback
g2 = tf(K, [J, (B+K*H), 0])
print(g2)

# closed-loop system
gc1 = feedback(g1, 1, -1)
print(gc1)
gc2 = feedback(g2, 1, -1)
print(gc2)

t = np.linspace(0, 10, 100)
t,y1 = step_response(gc1,t)
t,y2 = step_response(gc2,t)
plt.plot(t,y1, lw = 2, label='w/o feedback')
plt.plot(t,y2, lw = 2, label='w feedback')
plt.grid()
plt.xlabel('time(s)')
plt.ylabel('Position')
plt.legend(loc='best')
```



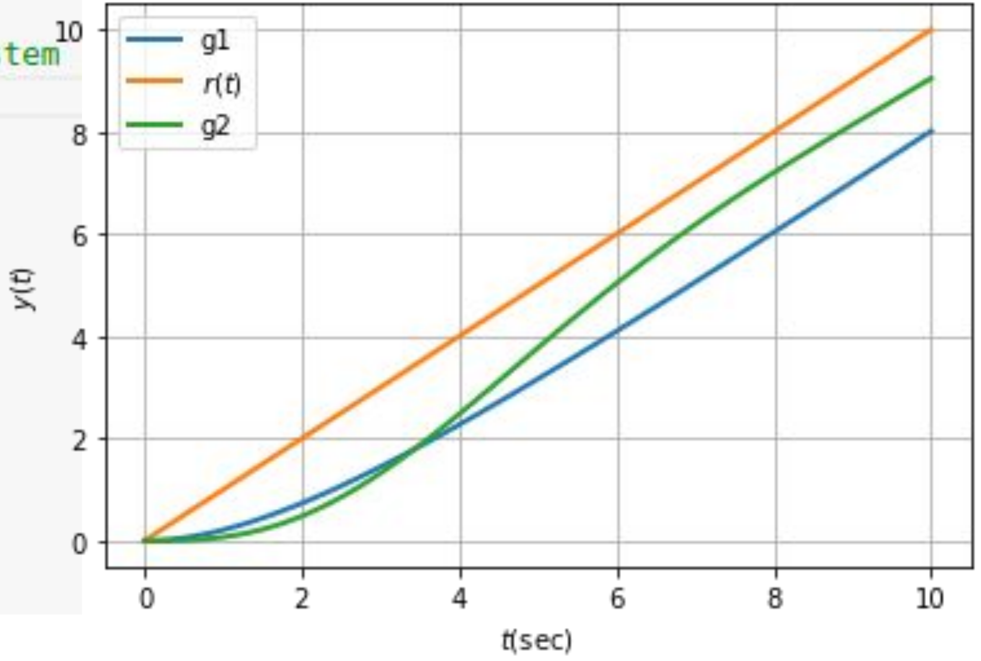
Velocity feedback adds damping to the system response.

Response of SO systems to Ramp Input



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

g1 = tf(1, [2,1]) # first order system
g2 = tf(1, [2, 1, 1]) # second order system

t = np.linspace(0,10, 100)
u = t
t,y1,x1 = forced_response(g1,t,u)
t,y2,x2 = forced_response(g2,t,u)
plt.plot(t,y1, lw = 2, label='g1')
plt.plot(t,u, lw = 2, label='$r(t)$')
plt.plot(t,y2,lw = 2, label='g2')
plt.grid()
plt.xlabel('$t$(sec)')
plt.ylabel('$y(t)$')
plt.legend(loc='best')
```

Second-order system tends to reduce the steady-state error for a ramp input.



Summary

We studied the following in this lecture:

- Defining specifications for comparing control system performances.
- Behaviour of First-order and Second-Order Systems
    - Mostly transient responses.
- Modeling a DC Motor system
- In the lab session, we will write some python programs to better understand the concepts presented in this lecture.