CONTENTS

***Abstract*—This manual is an introduction to control systems based on GATE problems.Links to sample Python codes are available in the text.**

Download python codes using

```
svn co https://github.com/gadepall/school/trunk/
    control/codes
```

1 STABILITY

1.1 Second order System

1.1. Consider the following second order system with the transfer function

$$G(s) = \frac{1}{1+2s+s^2} \tag{1.1.1}$$

Is the system stable?
Solution:

$$G(s) = \frac{1}{1+2s+s^2} \tag{1.1.2}$$

From given expression of G(s),both poles of G(s) are at (-1,0) which is on the left half of s-plane, therefore we can conclude that the system is stable.

1.2. Find and sketch the step response $c(t)$ of the system.
Solution: For step-response, we take input as unit-step function u(t)

$$C(s) = R(s).G(s) = \left[\frac{1}{s}\right]\left[\frac{1}{1+2s+s^2}\right] \tag{1.2.1}$$

$$C(s) = \frac{1}{s(1+s)^2} \tag{1.2.2}$$

Using Partial Fractions,

$$C(s) = \frac{1}{s} - \frac{1}{(1+s)} - \frac{1}{(1+s)^2} \tag{1.2.3}$$

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

x = np.linspace(0,6,1000)
y = 1−(np.exp(−x))−(x∗np.exp(−x))
plt.plot(4.52218,0.94,'o')
plt.text(3.4,0.95,'(4.52,0.94)')
plt.grid()
plt.plot(x,y)
plt.show()
```

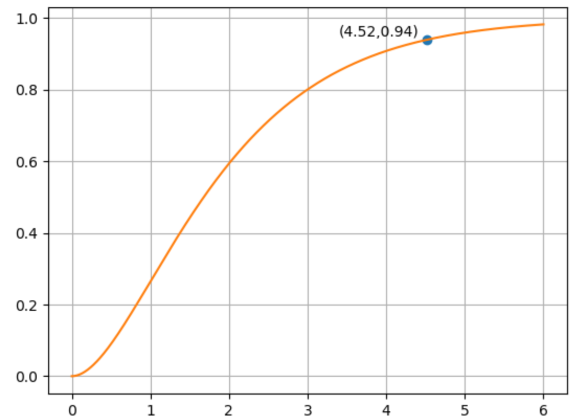

Fig. 1.2

Therefore;

$$c(t) = L^{-1}\left[\frac{1}{s}\right] - L^{-1}\left[\frac{1}{1+s}\right] - L^{-1}\left[\frac{1}{(1+s)^2}\right] \tag{1.2.4}$$

Using the Known inverse transforms:

$$c(t) = (1 - e^{-t} - te^{-t}).u(t) \tag{1.2.5}$$

1.3. Find the steady state response of the system.
Solution: To know the steady response value of c(t), we calculate

$$\lim_{t\to\infty} c(t) = (1+0+0).(1) = 1 \tag{1.3.1}$$

1.4. Find the time system output c(t) to reach 94% of its steady state value.
Solution: Now, 94% of 1 is 0.94, so we should now solve for a positive t such that

$$(1 - e^{-t} - te^{-t}) = 0.94 \tag{1.4.1}$$

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

t = 0
c = 0
y = 0

# defining a function whose solution of f=0
    will give us the value of t for 94% of
    output
def s(a):
        v = 0.06 − np.exp(−a) − a∗np.exp(−
```

```
            a)
        return v

#loop starts at t = 0, checks wether function
    is +ve, if not, then increases t by 0.001
while c == 0:
        y = s(t)
        if y < 0:
                c = 0
                t = t+0.001

        else:
                c = 1


#approx. the curve as a straight line, we find
    the crossing point by weighted mean of
    the two succesive values
t1 = ((t-0.001)*(-s(t-0.001)) + t*s(t))/(-s(t
    -0.001) + s(t))

print(t1)
```

$$t = 4.5228 \tag{1.4.2}$$

2 Routh Hurwitz Criterion

3 Compensators

4 Nyquist Plot