# EE2703 Assignment 6

Gali Ushasri Roll Number : EE20B033

March 2022

## 1 Introduction

In this assignment, we will look at how to analyze “Linear Time-invariant Systems” using the scipy.signal library in Python .We limit our analysis to systems with rational polynomial transfer functions. More specifically we consider 3 systems: A forced oscillatory system, A coupled system and RLC low pass filter

### 1.1 Assignment Questions

#### 1.1.1 Question 1

We first consider the forced oscillatory system(with 0 initial conditions):

$$\ddot{x} + 2.25x = f(t) \tag{1}$$

We solve for $X(s)$ using the following equation, derived from the above equation.

$$X(s) = \frac{F(s)}{s^2 + 2.25} \tag{2}$$

We then use the impulse response of $X(s)$ to get its inverse Laplace transform.

```
import scipy.signal as sp
#this function returns X(s) from equation(2)
def transfer_function(freq,decay):
    k= np.polymul([1.0,0,2.25],[1,-2*decay,freq*freq + decay*decay])
    H= sp.lti([1,-1*decay],k)
    return H
#Taking impulse response of X(s) and plotting
t,x = sp.impulse(transfer_function(1.5,-0.5),None,np.linspace(0,50,5001))
plot_function(t,x,"t","x","Forced Damping Oscillator with decay = 0.5")
```

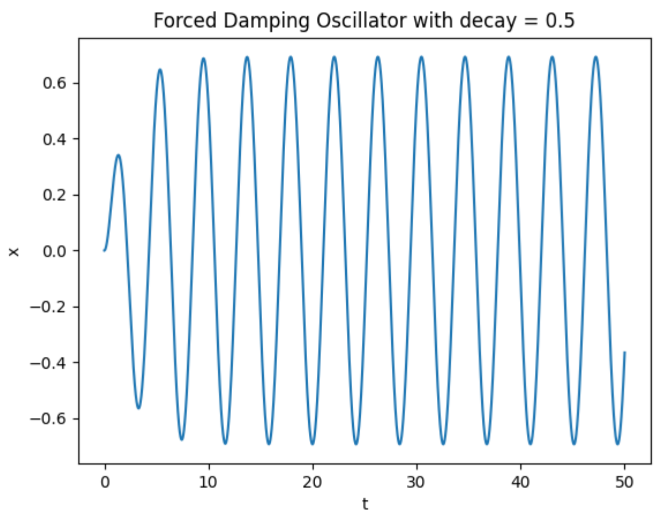


Figure 1: System Response with Decay = 0.5

### 1.1.2 Question 2

We now see what happens with Decay Constant=0.05.

```
t,x = sp.impulse(transfer_function(1.5,-0.05),None,np.linspace(0,50,5001))
plot_function(t,x,"t","x","Forced Damping Oscillator with decay = 0.05")
```

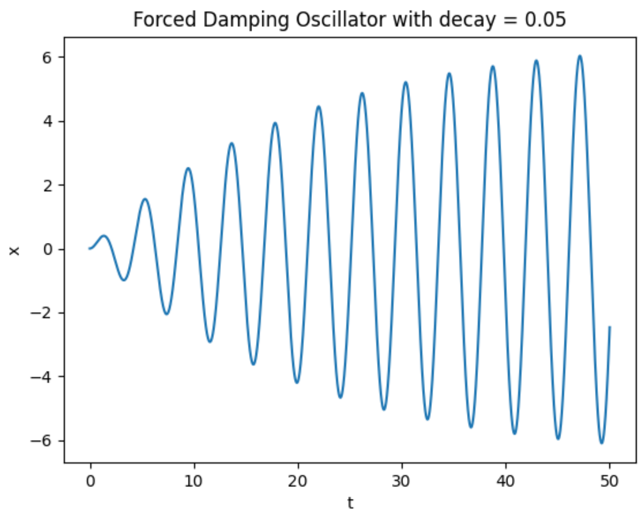


Figure 2: System Response with Decay = 0.05

I noticed that the result is very similar to that of question 1, except with a different amplitude. This is because the system takes longer to reach a steady state.

### 1.1.3 Question 3

We now see what happens when we vary the frequency from 1.4 to 1.6. We note the the amplitude is maximum at frequency = 1.5, which is the natural frequency of the given system

```
frequency = np.linspace(1.4,1.6,5)
for freq in frequency:
    H = sp.lti([1],[1,0,2.25])
    t = np.linspace(0,150,5001)
    F = np.cos(freq*t)*np.exp(-0.05*t)*(t>0)
    t,y,svec = sp.lsim(H,F,t)
    plt.title("Forced Damping Oscillator with %.2f frequency" % freq)
    plt.xlabel("t")
    plt.ylabel("y")
    plt.plot(t,y)
```

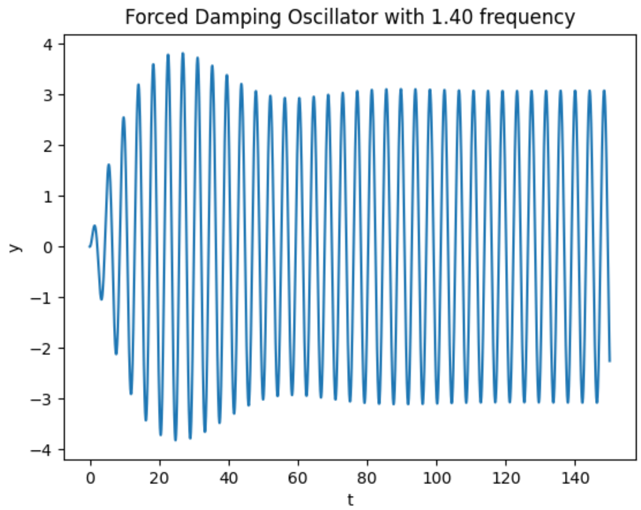


Figure 3: System Response with frequency = 1.40

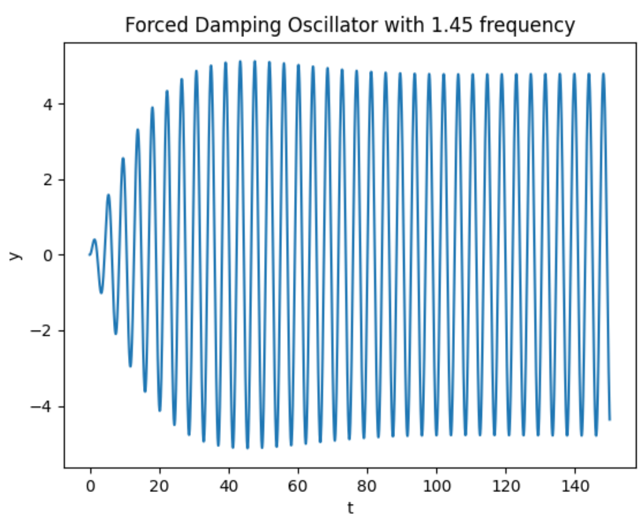


Figure 4: System Response with frequency = 1.45

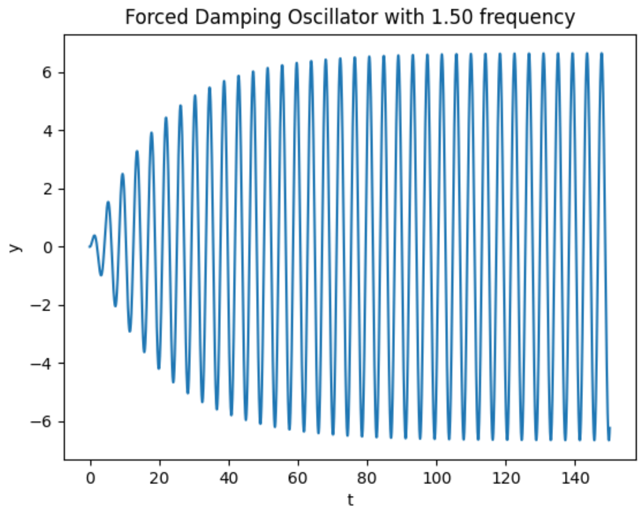


Figure 5: System Response with frequency = 1.50

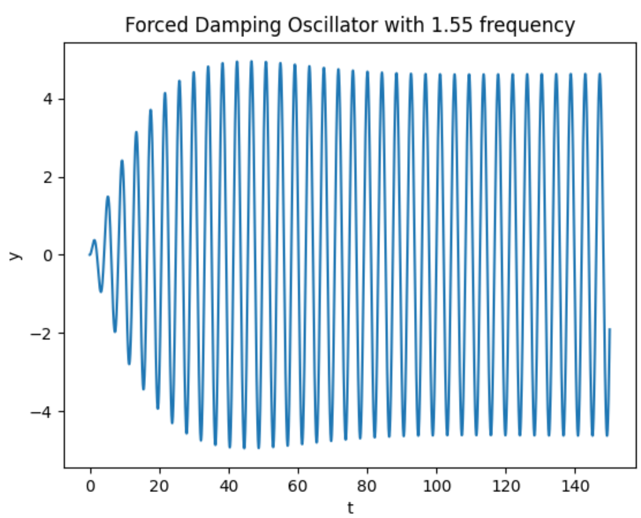


Figure 6: System Response with frequency = 1.55

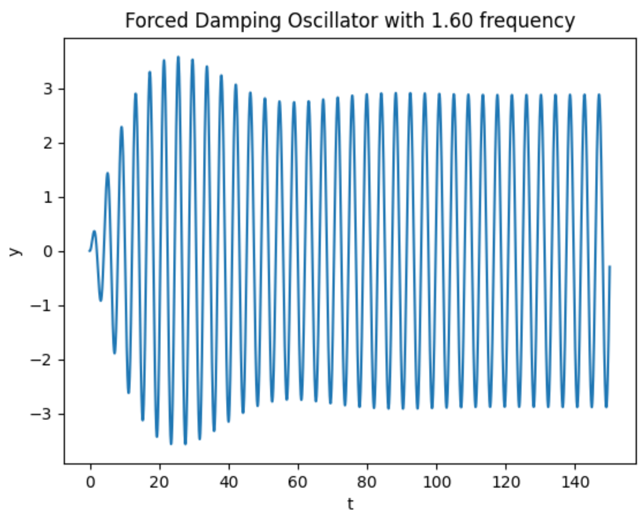


Figure 7: System Response with frequency = 1.60

### 1.1.4 Question 4

We now consider a coupled Differential system

$$\ddot{x} + (x - y) = 0 \tag{3}$$

and

$$\ddot{y} + 2(y - x) = 0 \tag{4}$$

with the initial conditions: $\dot{x}(0) = 0, \dot{y}(0) = 0, x(0) = 1, y(0) = 0$. Taking Laplace Transform and solving for $X(s)$ and $Y(s)$, We get:

$$X(s) = \frac{s^2 + 2}{s^3 + 3s} \tag{5}$$

$$Y(s) = \frac{2}{s^3 + 3s} \tag{6}$$

```
#solve for X in coupling equation
X = sp.lti([1,0,2],[1,0,3,0])
t,x = sp.impulse(X,None,np.linspace(0,20,5001))
plot_function(t,x,"t","x","Coupled Oscilations: X",)

#solve for Y in coupling equation
Y = sp.lti([2],[1,0,3,0])
t,y = sp.impulse(Y,None,np.linspace(0,20,5001))
plot_function(t,y,"t","y","Coupled Oscilations: Y")
```

We notice that the outputs of this system are 2 sinusoids which are out of phase.This system can be realized by creating an undamped single spring double mass system.

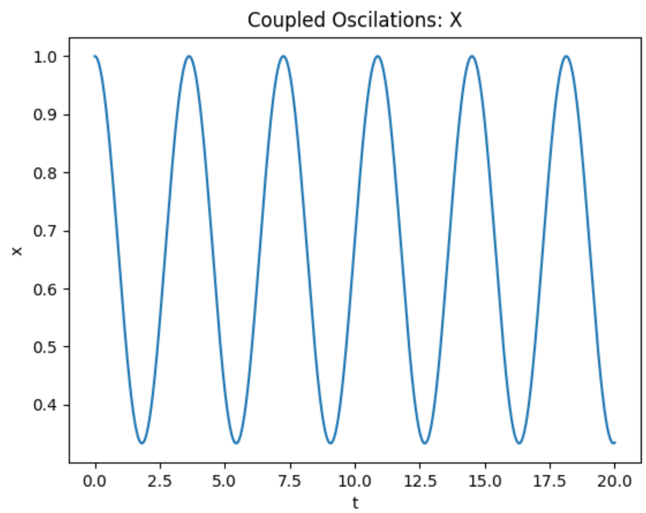


Figure 8: Coupled Oscillations of X

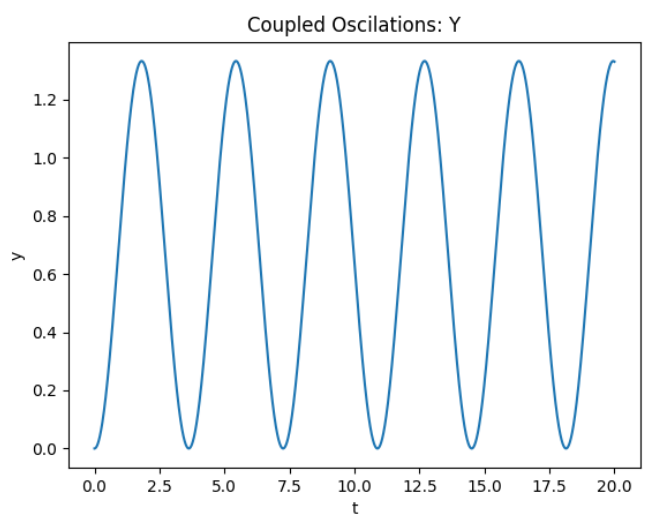


Figure 9: Coupled Oscillations of Y

### 1.1.5 Question 5

Now we try to create the bode plots for the low pass filter defined in the question 5

```
#returns Low pass filter response for given input and time period
def RLC(time,R,L,C,f):
    H = sp.lti([1],[L*C,R*C,1])
    w,S,phi = H.bode()
    plt.subplot(2,1,1)
    plt.title("Magnitude response")
    plt.xlabel("w")
    plt.ylabel(r'$|H(s)|$')
    plt.semilogx(w,S)
    plt.show()
    plt.subplot(2,1,2)
    plt.title("Phase response")
    plt.xlabel("w")
    plt.ylabel(r'$\angle(H(s))$')
    plt.semilogx(w,phi)
    plt.show()
    return sp.lsim(H,f,time)

t=np.linspace(0,30e-6,1000)
R=100
L=1e-6
C=1e-6
function= np.cos(1000*t) - np.cos(1e6*t)
t,y,svec = RLC(t,R,L,C,function)
```

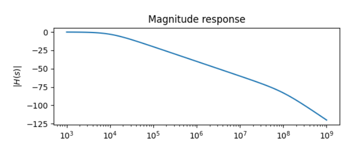


Figure 10: Magnitude response

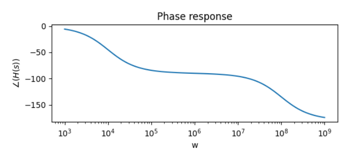


Figure 11: Phase response

#### 1.1.6 Question 6

We now plot the response of the low pass filter to the given input:

$$V_i(t) = (cos(10^3 t) - cos(10^6 t))u(t)$$

for $0 < t < 30\mu s$ and $0 < t < 10ms$

```
t=np.linspace(0,30e-6,1000)
R=100
L=1e-6
C=1e-6
function= np.cos(1000*t) - np.cos(1e6*t)
H = sp.lti([1],[L*C,R*C,1])
w,S,phi = H.bode()
t,y,svec = sp.lsim(H,function,t)
plot_function(t,y,"t",r'$v_{o}(t)$',"Output of RLC for t<30u sec")

t=np.linspace(0,10e-3,100000)
function= np.cos(1000*t) - np.cos(1e6*t)
H = sp.lti([1],[L*C,R*C,1])
w,S,phi = H.bode()
t,y,svec = sp.lsim(H,function,t)
plot_function(t,y,"t",r'$v_{o}(t)$',"Output of RLC for t<10m sec")
```

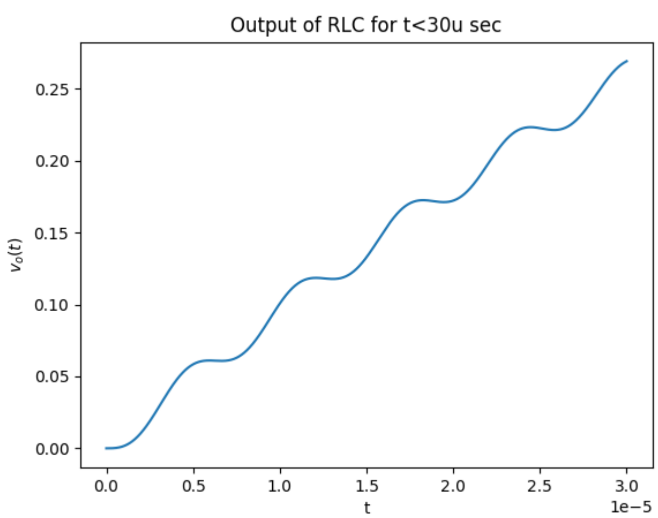


Figure 12: System response for t¡30us

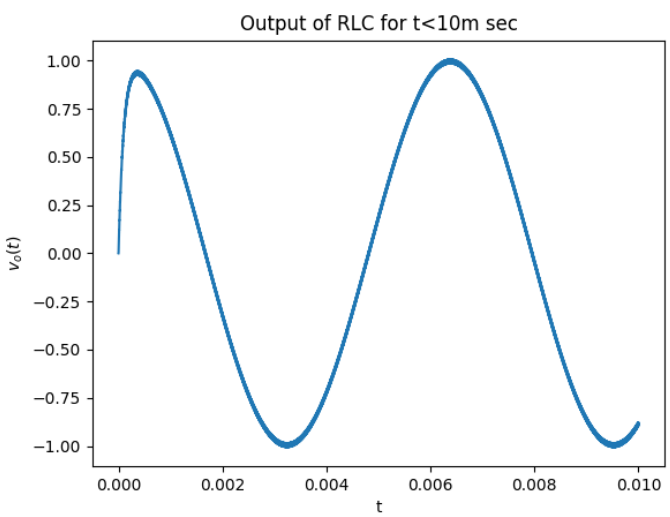


Figure 13: System response for t¡30ms

## 2 Conclusion

LTI systems are observed in all fields of engineering and are very important. In this assignment, we have used scipy's signal processing library to analyze a wide range of LTI systems. Specifically we analyzed forced oscillatory systems, single spring, double mass systems and Electric filters