Assignment 7: Laplace transform

G Ch V Sairam , EE19B081

16-04-2021

Introduction

In this assignment , we learn to use the scipy.signal module in python to analyze Linear Time-Invariant systems. We use 3 systems: A osciallator undergoing forced damping , A coupled differential equation system and an RLC filter. All these systems only have rational transfer functions.

Question 1

We are given a forced oscillatory system with 0 initial conditions.

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

Converting into the laplace domain , we get

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

Then , X(s) can be written as,

$$X(s) = \frac{F(s)}{s^2 + 2.25} \tag{3}$$

We then find out the impulse response of $X(s)$ and plot it.

```
def transfer_func(frequency,decay):
        num=([1,-1*decay])                          #Numerator polynomial of
        denom= np.polymul([1.0,0,2.25],[1,-2*decay,frequency*frequency +
        return sp.lti(num,denom)            #Returns the transfer function wi

H1=transfer_func(1.5,-0.5)
t,x = sp.impulse(H1,None,np.linspace(0,50,5001))            #Impulse response
plt.figure()
plot_graphs(t,x,'Damping oscillator with 0.5 decay','t','x')      #Plot the
```

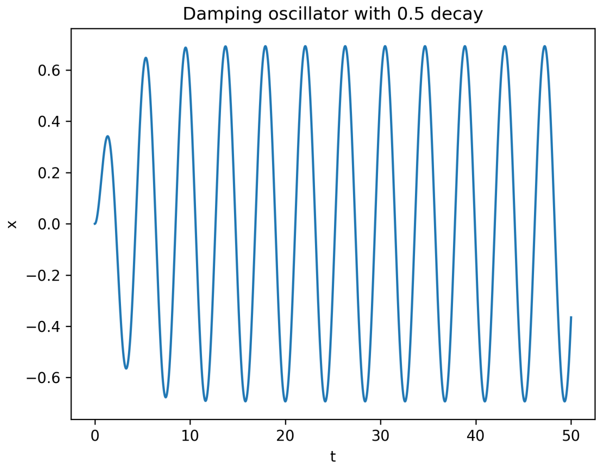

Figure 1: Damping oscillator with decay=0.5

Question 2

We do repeat the above block with a different amount of delay , 0.05.

```
H2=transfer_func(1.5,-0.05)
t,x = sp.impulse(H2,None,np.linspace(0,50,5001))          #Impulse
plt.figure()
plot_graphs(t,x,'Damping oscillator with 0.05 decay','t','x')     #Plot the
```

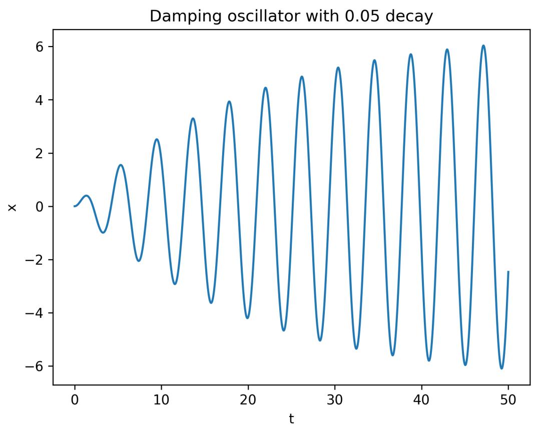

Figure 2: Damping oscillator with decay=0.05

We notice 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.

Question3

We now vary the frequency keeping the delay the same(i.e;-0.05). We plot the graphs with the frequencies 1.4,1.45,1.5,1.55 and 1.6 Hz.

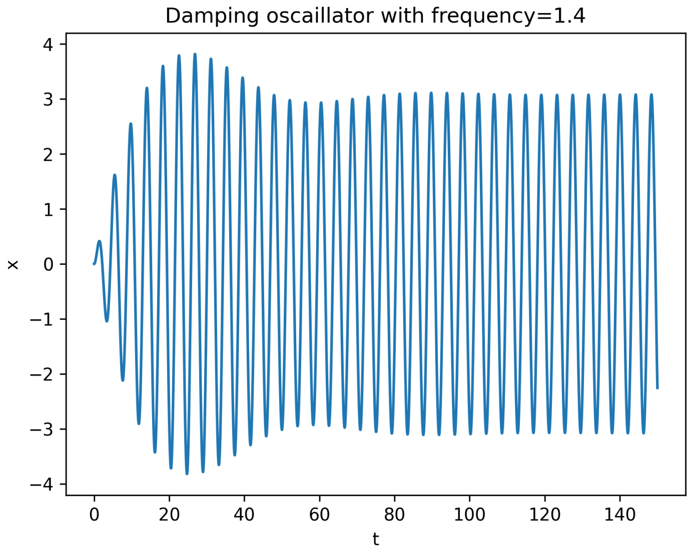

Figure 3: Damping oscillator with decay=1.4

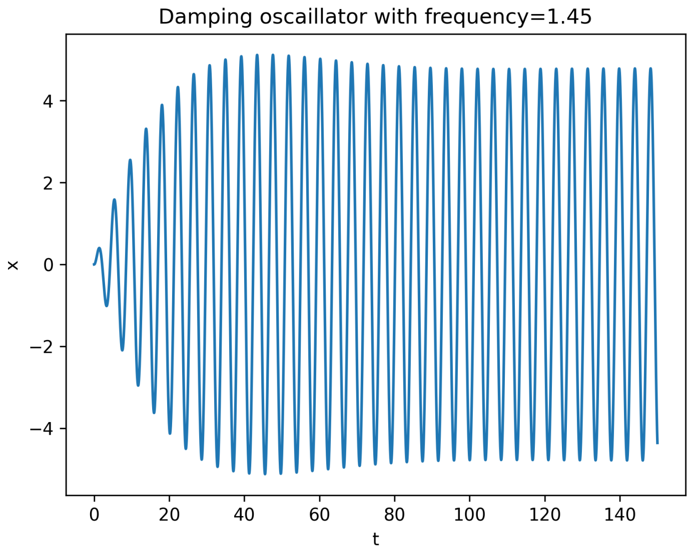

Figure 4: Damping oscillator with decay=1.45

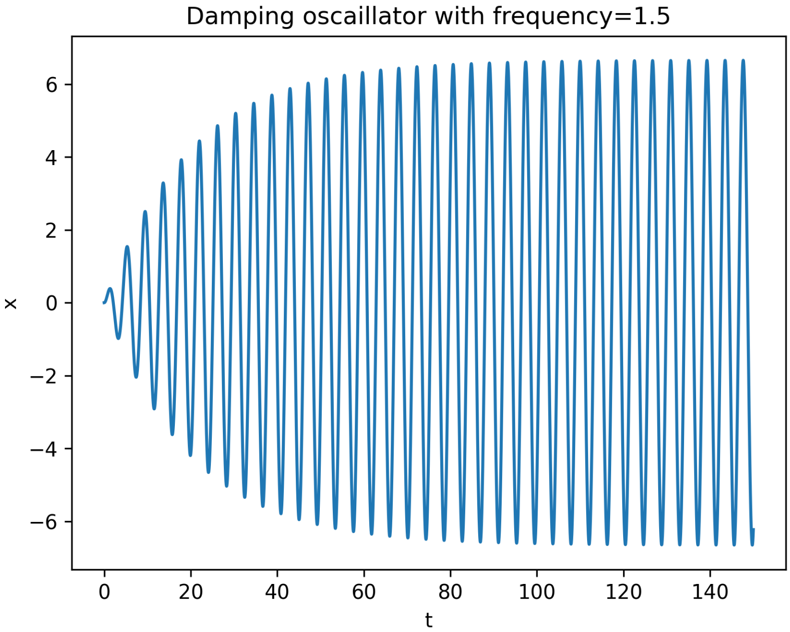

Figure 5: Damping oscillator with decay=1.5

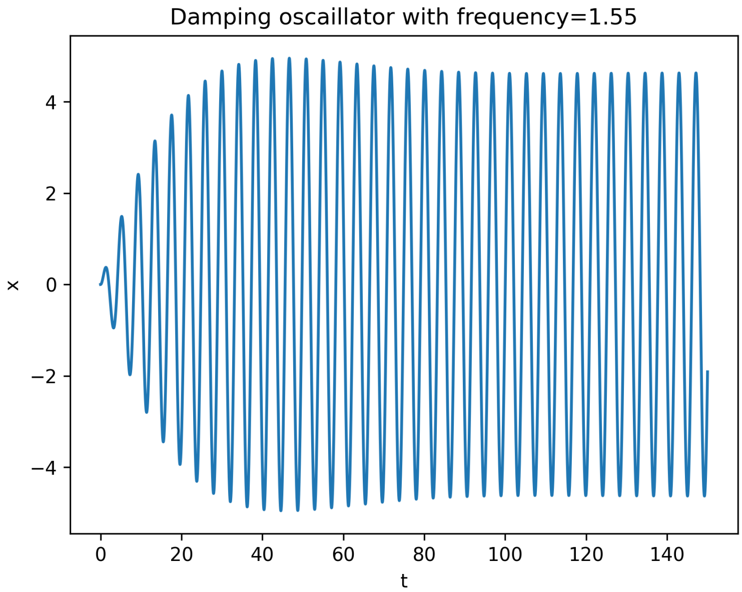

Figure 6: Damping oscillator with decay=1.55

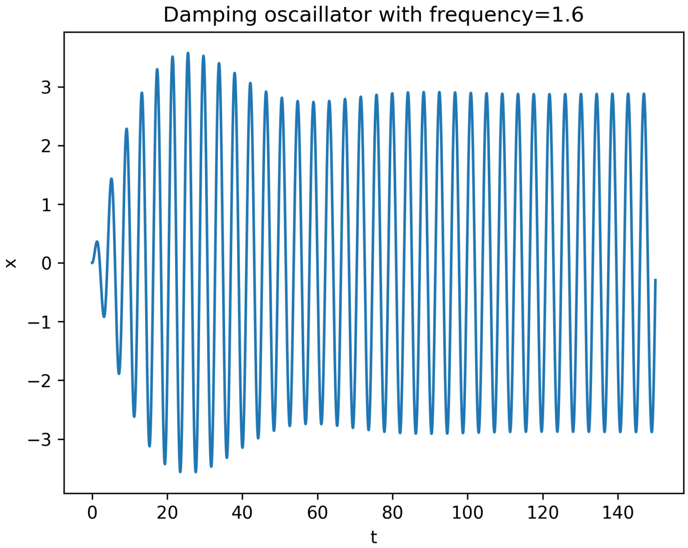

Figure 7: Damping oscillator with decay=1.6

We notice that the amplitude is maximum at frequency=1.5 Hz

Question 4

Here , we are given a coupled system of differential equations. The given equations are:

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

$$\ddot{y} + 2(y - x) = 0 \tag{5}$$

Transforming these equations into laplace domain , we get:

s^2X(s)+X(s)-Y(s)=0

s^2Y(s)+2(Y(s)-X(s))=0

Solving these equations , we get the solution:

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

$$Y(s) = \frac{2}{s^3 + 3s} \tag{7}$$

We plot their impulse responses.

```
X = sp.lti([1,0,2],[1,0,3,0])
t,x = sp.impulse(X,None,np.linspace(0,50,5001))
plot_graphs(t,x,"Coupled Oscilations","t","x")
Y = sp.lti([2],[1,0,3,0])
```

```
t,y = sp.impulse(Y,None,np.linspace(0,50,5001))
plot_graphs(t,y,"Coupled Oscilations","t","y")
```

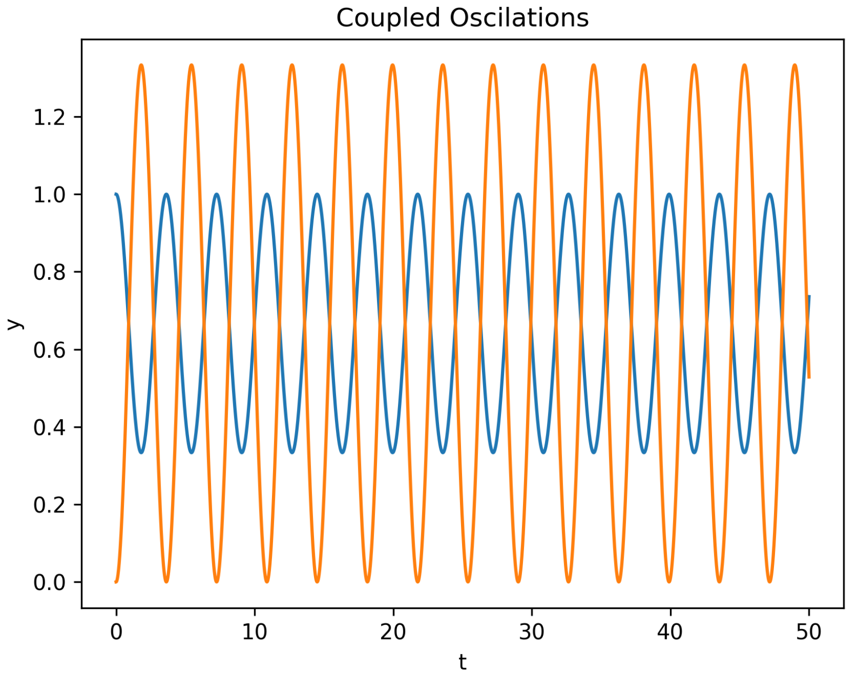

Figure 8: Coupled oscillations

We notice that the outputs of this system are 2 sinusoids which are out of phase.

Question 5

A 2-port RLC filter network is given. We calculate its impulse responses and plot its magnitude and phase responses.

```
H = sp.lti([1],[L*C,R*C,1])                 #Steady state transfer function o
w,S,phi = H.bode()
fig,(ax1,ax2) = plt.subplots(2,1)
ax1.set_title("Magnitude response")
ax1.semilogx(w,S)                      #Plot its magnitude response
ax2.set_title("Phase response")
ax2.semilogx(w,phi)                    #Plot its phase response
```

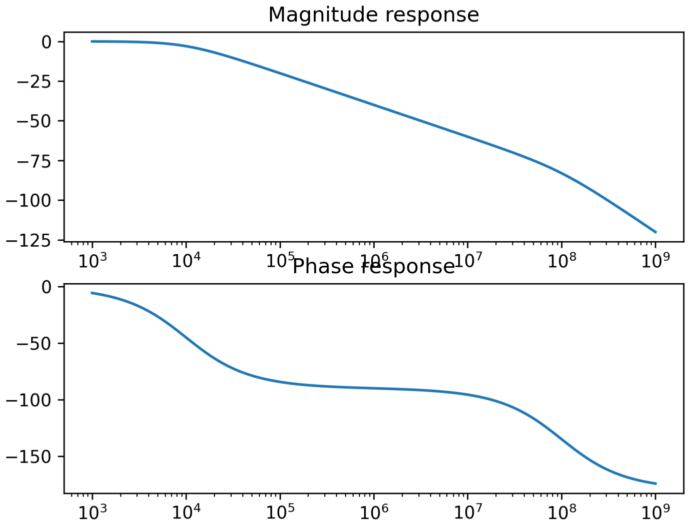

Figure 9: Bode plots

Question 6

In the above question , an input signal V_i(t) is supplied to the system.

$$V_i(t) = (cos(10^3 t) - cos(10^6 t))u(t) \tag{8}$$

We obtain the output voltage V_o(t) by defining the transfer function as a system and obtaining the output using signal.lsim.

We calculate the output signal for times on different timescales- in microseconds and in milliseconds.

```
def func(t):
        return np.cos(1000*t) -np.cos(1e6*t)                #The input signal

time=np.linspace(0,30e-6,10000)
t,y,_ = sp.lsim(H,func(time),time)
plot_graphs(t,y,"Output of RLC for t<30us","t","x")               #Plots ou
time=np.linspace(0,30e-3,10000)
plt.savefig("assgn7_plot10.png",dpi=300)
plt.figure()
t,y,_ = sp.lsim(H,func(time),time)
plot_graphs(t,y,"Output of RLC for t<30ms","t","x")               #Plots ou
plt.savefig("assgn7_plot11.png",dpi=300)
```

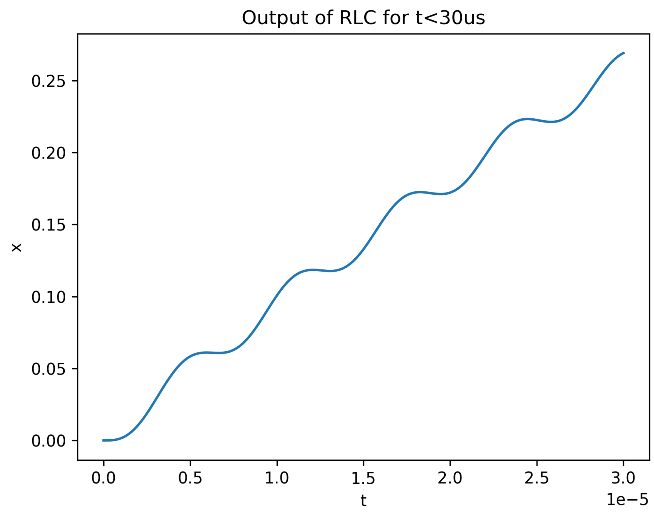

Figure 10: Output signal in a timescale of microseconds

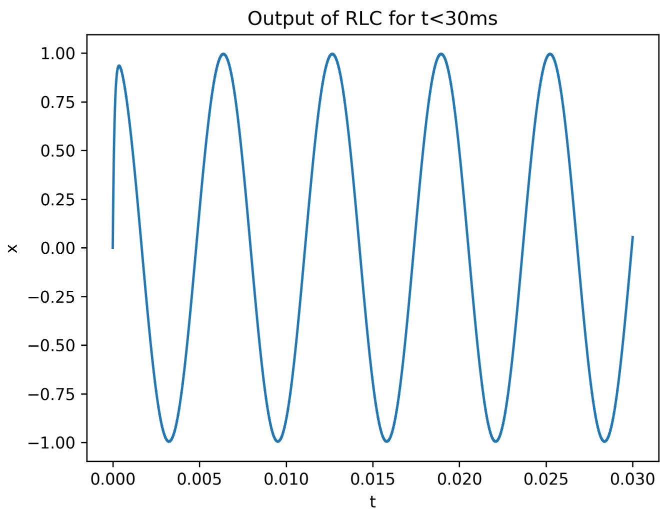

Figure 11: Output signal in a timescale of milliseconds

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.