EE2703 Week 6

Allu Yaswanth,EE20B007

Mar 27, 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 of Differential Equations and an 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.

```
# Time response of spring with decay 0.5
p1_num = pl.poly1d([1,0.5]) #Defining numerator polynomial
p1_den = pl.polymul([1,1,2.5],[1,0,2.25]) #Defining denominator polynomial
S1 = sp.lti(p1_num,p1_den)
t1,x1 = sp.impulse(S1,None,pl.linspace(0,50,500))

# The plot x(t) vs t for decay 0.5
pl.figure(0)
pl.plot(t1,x1)
pl.title("X(t) for decay 0.5")
pl.xlabel(r'$t\rightarrow$')
pl.ylabel(r'$x(t)\rightarrow$')
pl.grid(True)
pl.show()
```

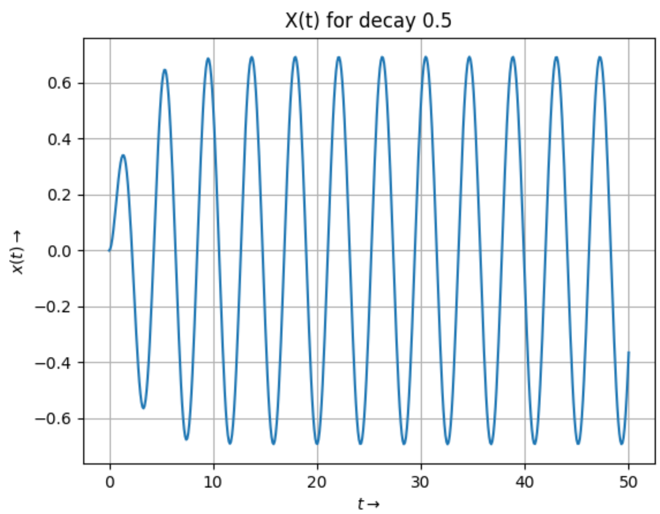

Figure 1: System Response with Decay = 0.5

1.1.2 Question 2

We now see what happens with a smaller Decay Constant.

```
# The plot x(t) vs t for decay 0.05
p2_num = pl.poly1d([1,0.05]) #Defining numerator polynomial
p2_den = pl.polymul([1,0.1,2.2525],[1,0,2.25]) #Defining denominator polynomial
S2 = sp.lti(p1_num,p2_den)
t2,x2 = sp.impulse(S2,None,pl.linspace(0,50,500))

# The plot x(t) vs t for decay 0.05
pl.figure(1)
pl.plot(t2,x2)
pl.title("X(t) for decay 0.05")
pl.xlabel(r'$t\rightarrow$')
pl.ylabel(r'$x(t)\rightarrow$')
pl.grid(True)
pl.show()
```

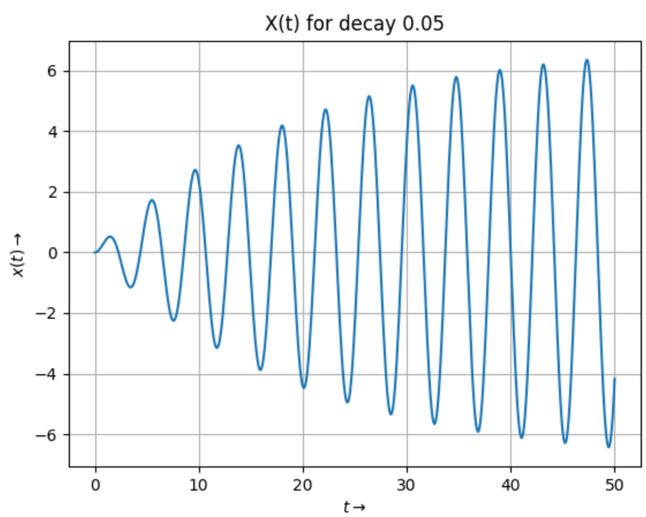

Figure 2: System Response with Decay = 0.05

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.

1.1.3 Question 3

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

```
# Transfer function and responces by varying frequencies from 1.4 to 1.6
H = sp.lti([1],[1,0,2.25])

for OMEGA in np.arange(1.4, 1.6, 0.05):
    t3 = np.linspace(0, 50, 500)
    f = np.cos(OMEGA*t3)*np.exp(-0.05*t3)
    t3,x3,svec = sp.lsim(H,f,t3)

# The plot of x(t) for various frequencies vs time.
    pl.figure(2)
    pl.plot(t3,x3,label='w = ' + str(OMEGA))
    pl.title("x(t) for different frequencies ranged from 1.4 to 1.6")
    pl.xlabel(r'$t\rightarrow$')
    pl.ylabel(r'$x(t)\rightarrow$')
    pl.legend(loc = 'upper left')
    pl.grid(True)
pl.show()
```

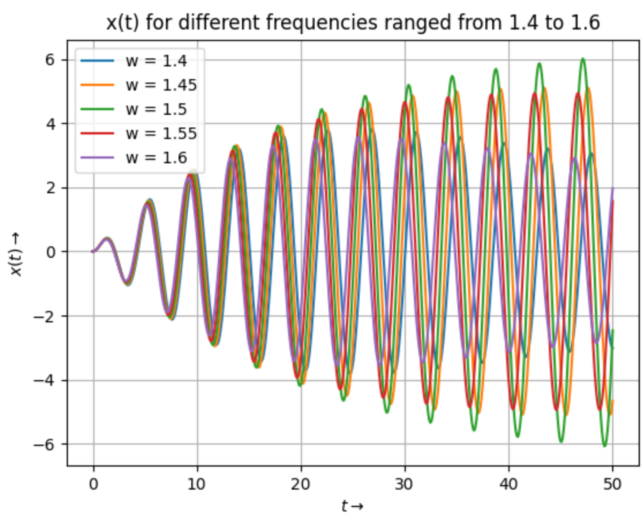

Figure 3: System Response with frequency ranging from 1.4 to 1.6

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:

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

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

```
# The python code snippet for Q.4
t4 = pl.linspace(0,20,500)
X4 = sp.lti([1,0,2],[1,0,3,0])
Y4 = sp.lti([2],[1,0,3,0])
t4,x4 = sp.impulse(X4,None,t4)
t4,y4 = sp.impulse(Y4,None,t4)

pl.figure(3)
pl.plot(t4,x4,label='x(t)')
pl.plot(t4,y4,label='y(t)')
pl.title("x(t) and y(t)")
pl.xlabel(r'$t\rightarrow$')
pl.ylabel(r'$functions\rightarrow$')
pl.legend(loc = 'upper right')
pl.grid(True)
pl.show()
```

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.

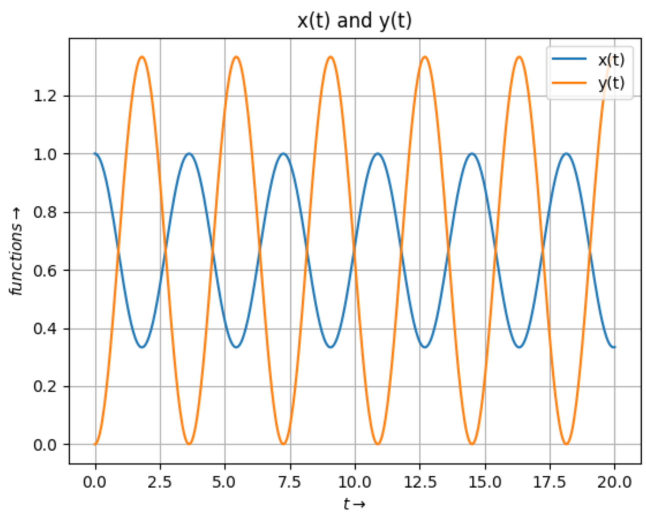

Figure 4: Coupled Oscillations

1.1.5 Question 5

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

```
#To find the Transfer Equation of two port network and plotting bode plots of magnitude and ph
R = 100
L = 1e-6
C = 1e-6

OMEGA = 1/np.sqrt(L*C)
Qf = 1/R * np.sqrt(L/C)
ZETA = 1/(2*Qf)

num = pl.poly1d([OMEGA**2])
den = pl.poly1d([1,2*OMEGA*ZETA,OMEGA**2])

H = sp.lti(num,den)

w,S,phi=H.bode()

# The magnitude bode plot
pl.figure(4)
pl.semilogx(w,S)
pl.title("Magnitude Bode plot")
pl.xlabel(r'$\omega\rightarrow$')
pl.ylabel(r'$20\log|H(j\omega)|\rightarrow$')
pl.grid(True)
pl.show()

# The phase bode plot
pl.figure(5)
pl.semilogx(w,phi)
pl.title("Phase Bode plot")
pl.xlabel(r'$\omega\rightarrow$')
pl.ylabel(r'$\angle H(j\omega)\rightarrow$')
pl.grid(True)
pl.show()
```

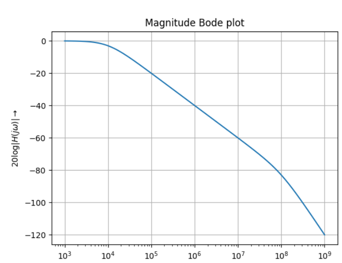

Figure 5: Magnitude Bode plot

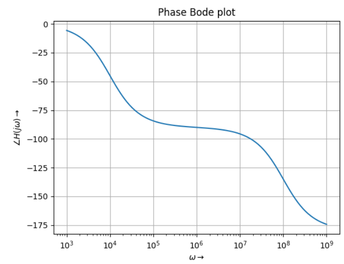

Figure 6: Phase Bode plot

1.1.6 Question 6

We know plot the response of the low pass filter to the input:

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

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

```
#To Find the output voltage from transfer function and input voltage for short term and long t
t6 = pl.arange(0,25e-3,1e-7)
vi = np.cos(1e3*t6) - np.cos(1e6*t6)
t6,vo,svec = sp.lsim(H,vi,t6)

# The plot of Vo(t) vs t for large time interval.
pl.figure(6)
pl.plot(t6,vo)
pl.title("The Output Voltage for large time interval")
pl.xlabel(r'$t\rightarrow$')
pl.ylabel(r'$V_o(t)\rightarrow$')
pl.grid(True)
pl.show()

# The plot of Vo(t) vs t for small time interval.
pl.figure(7)
pl.plot(t6[0:300],vo[0:300])
pl.title("The Output Voltage for small time interval")
pl.xlabel(r'$t\rightarrow$')
pl.ylabel(r'$V_o(t)\rightarrow$')
pl.grid(True)
pl.show()
```

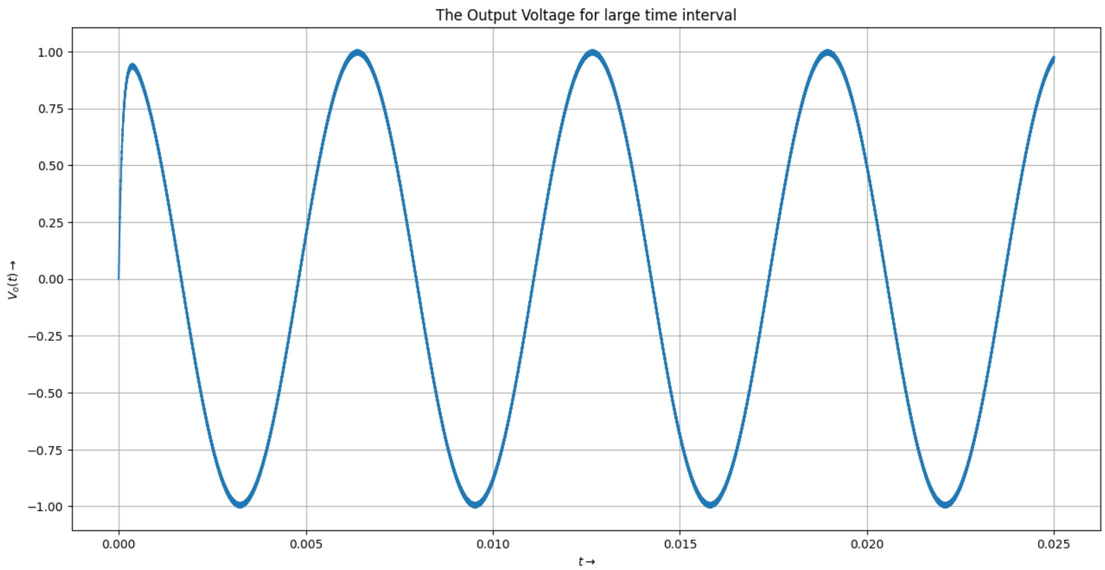

Figure 7: The output voltage for large time interval

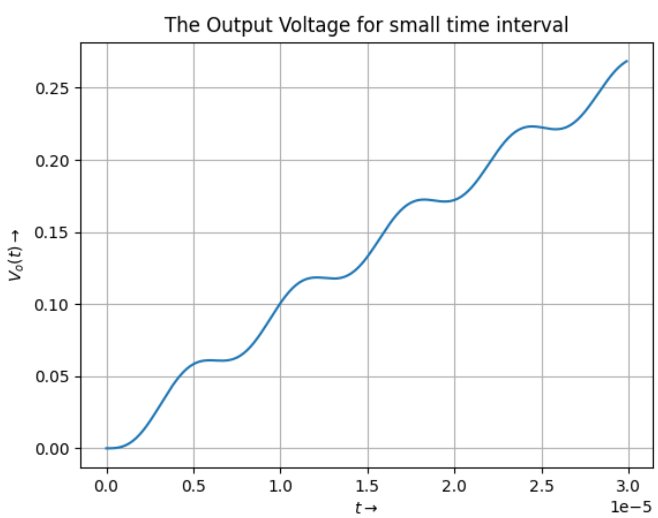

Figure 8: The output voltage for small time interval

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