EE2703: Assignment 7

Tanay Dixit
EE19B123

April 15, 2021

Introduction

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

Systems

Question 1

We 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.

```python
def H(self, freq , alpha):
    '''
    @Params:
    freq:  frequecy of the system
    alpha: decay constant
    '''
    p= np.polymul(self.coef,[1,-2*alpha,freq*freq + alpha*alpha])
    return sp.lti([1,-1*alpha],p)
```

We use the following code to plots the results in-order to understand them better.

```python
#Q1
laplace_solver = spring([1.0,0,2.25])
h = laplace_solver.H(1.5, -0.5)
t,x = sp.impulse(h,None,np.linspace(0,50,5001))
laplace_solver.plot(t,x, \
title = 'Forced Damping Oscillator with decay = 0.5' , \
xlabel = 't',\
ylabel = 'x',\
path = 'imgs/oscillation_1')

#Q2
laplace_solver = spring([1.0,0,2.25])
h = laplace_solver.H(1.5, -0.05)
t,x = sp.impulse(h,None,np.linspace(0,50,5001))
```

```
laplace_solver.plot(t,x, \
title = 'Forced Damping Oscillator with decay = 0.05' , \
xlabel = 't',\
ylabel = 'x',
path ='imgs/oscillation_2')
```

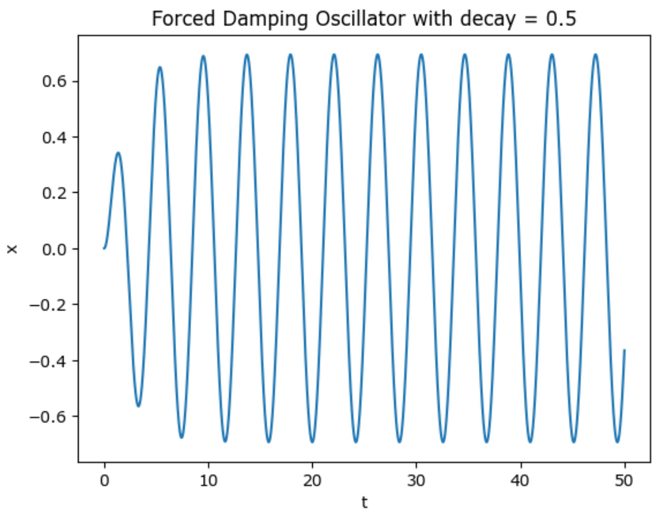

Figure 1: System Response with Decay = 0.5

Question 2

We now see what happens with a smaller Decay Constant.

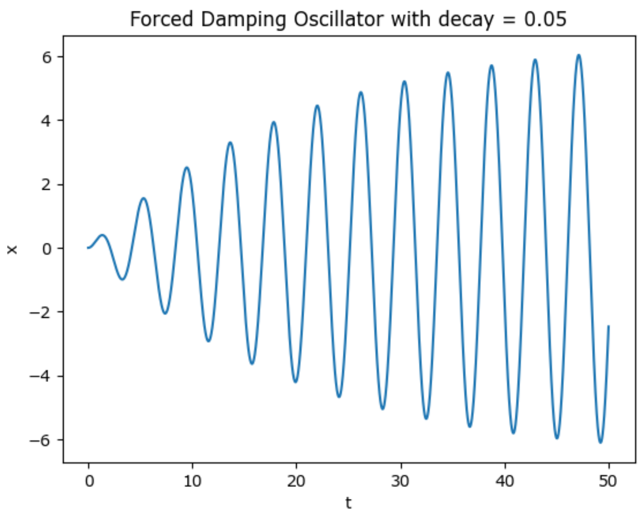

Figure 2: System Response with Decay = 0.05

We can notice that the result is similar to that of Figure 1, except with a different amplitude. This is because the system takes longer to reach a steady state.

Question 3

We now vary frequency. We note the the amplitude is maximum at frequency = 1.5, which is the natural frequency of the given system

```python
def LTI_system(self, FreqRange):
    '''
    @Params:
    FreqRange : list of possible frequencies : np.linspace(1.4,1.6,5)
    '''
    for f in FreqRange:
        h = sp.lti([1],self.coef)
        time = np.linspace(0,50,5001)
        u = np.cos(f*time)*np.exp(-0.05*time)
        t,x,_ = sp.lsim(h,u,time)
        self.plot(t, x, \
        title =f'Forced Damping Oscillator with {f}',\
        xlabel ='t',\
        ylabel ='x',\
        path =f'imgs/LTI_plots{f}')
```

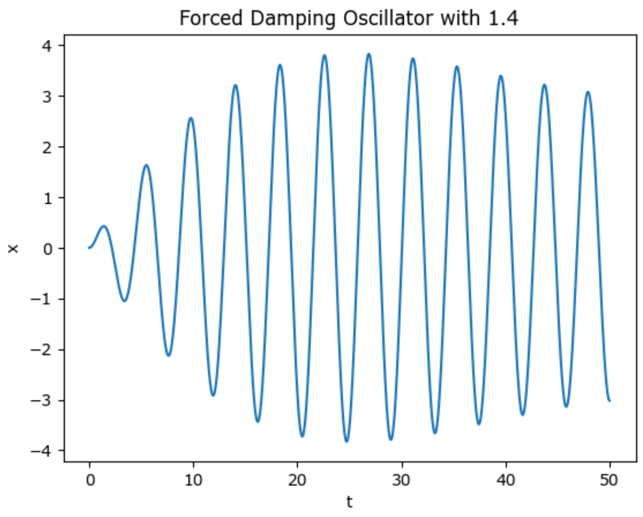

Figure 3: System Response with frequency = 1.4

When the input frequency is at the natural frequency the output amplitude is maximum.In the other cases the output amplitude decreases.This phenomenon is known as resonance.

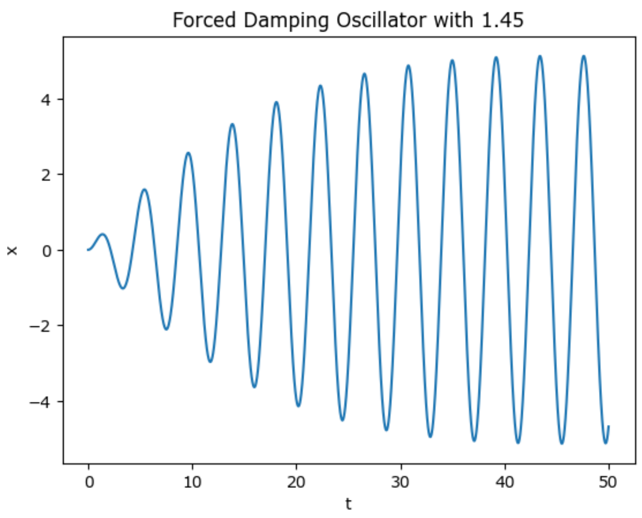

Figure 4: System Response with frequency = 1.45

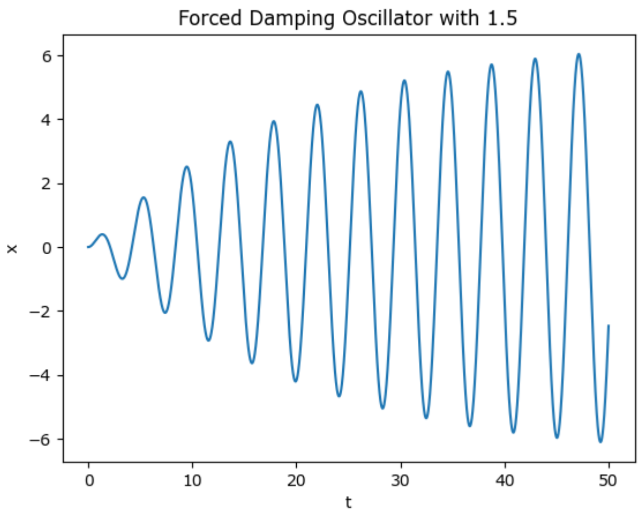

Figure 5: System Response with frequency = 1.5

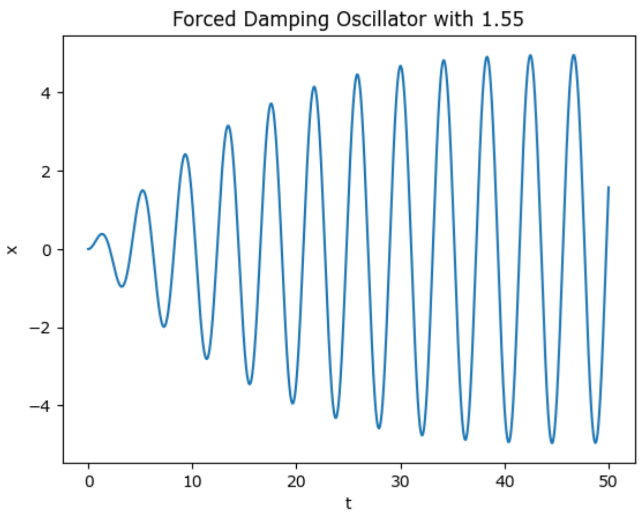

Figure 6: System Response with frequency = 1.55

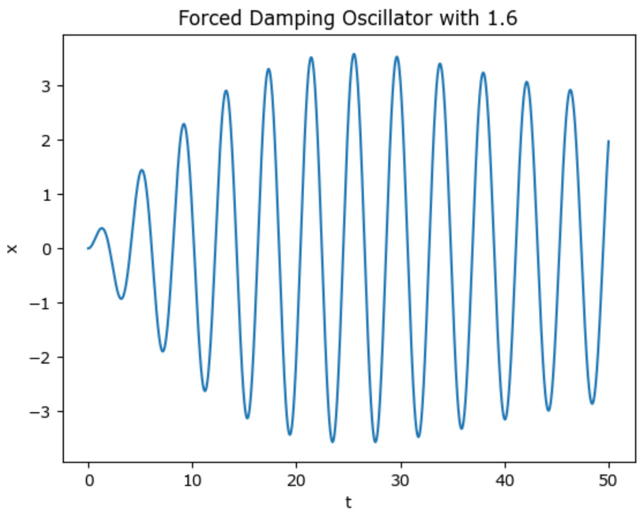

Figure 7: System Response with frequency = 1.6

Question 4

We now consider a system of coupled Differential systems

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

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:

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

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

```python
def system(num, den):
    '''
    @Params:
    num : numerator coefficients
    den : denominator coefficients
    '''
    X = sp.lti(num,den)
    t,x = sp.impulse(X,None,np.linspace(0,50,5001))
    return t, x
```

We can see 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 system.

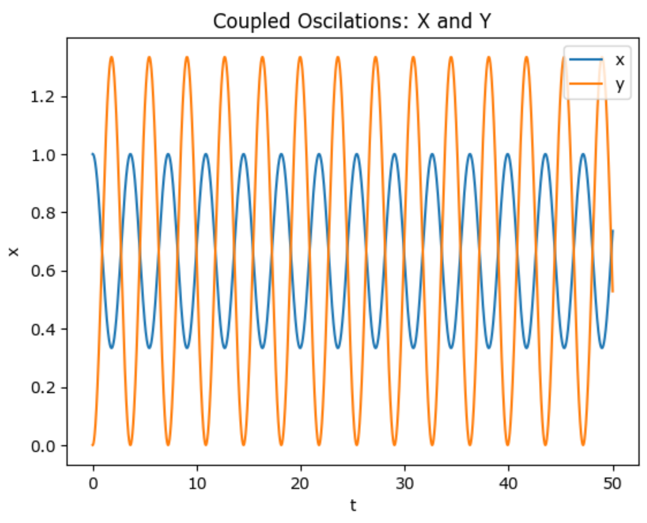

Figure 8: Coupled Oscillations

Question 5

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

```python
def TwoPortNetwork(self,time,R, L,C,vi = lambda t : np.cos(1000*t) -np.cos(1e6*t)
    ):
    H = sp.lti([1],[L*C,R*C,1])
    w,S,phi = H.bode()
    def plotting(w, S, phi):
        fig,(ax1,ax2) = plt.subplots(2,1)
        fig.set_figheight(8)
        fig.set_figwidth(12)
        ax1.set_title("Magnitude response")
        ax1.semilogx(w,S)
        ax2.set_title("Phase response")
        ax2.semilogx(w,phi)
        plt.savefig('imgs/2port_plots.png',bbox_inches='tight')

    plotting(w,S,phi)

    return sp.lsim(H,vi(time),time)
```

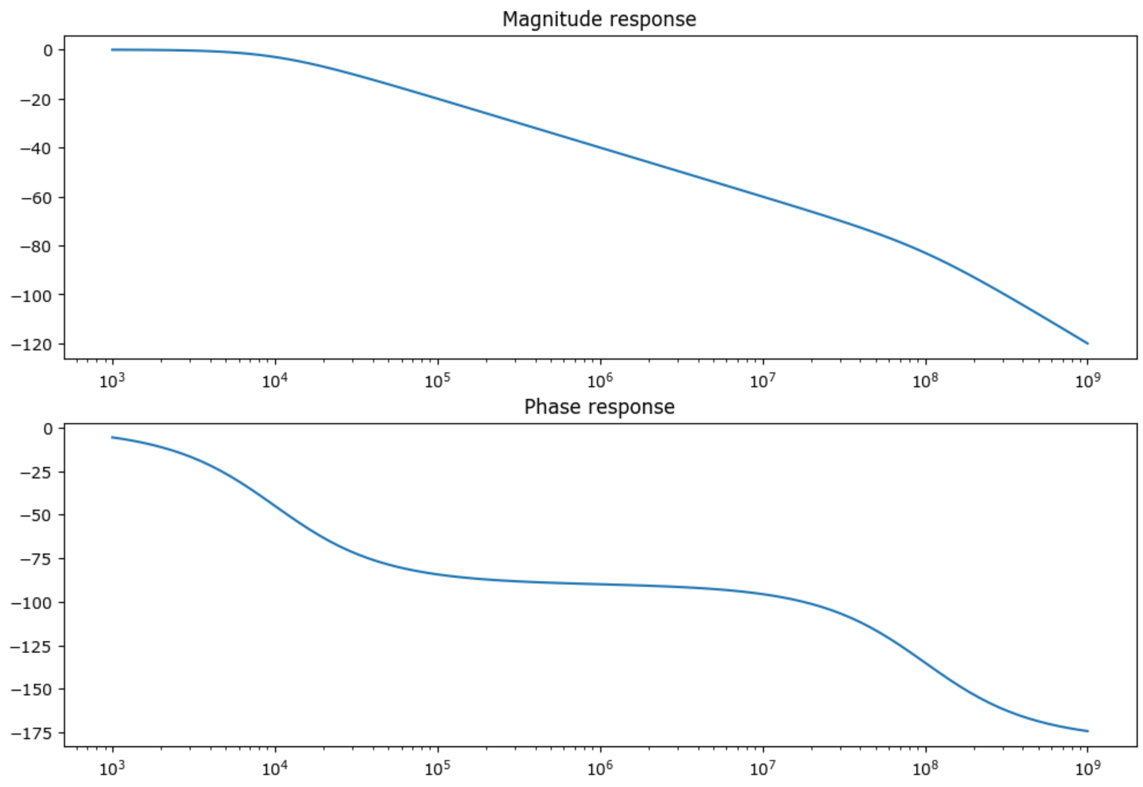

Figure 9: Bode Plots For RLC Low pass filter

Question 6

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

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

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

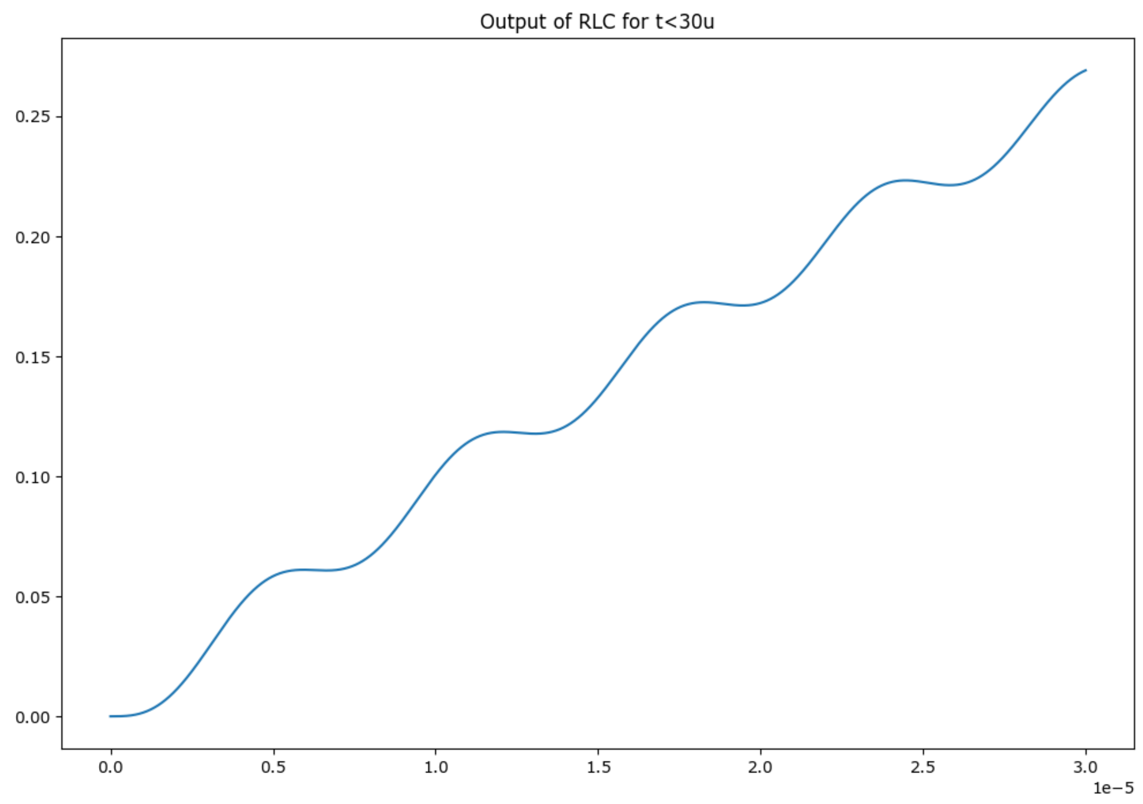

Figure 10: System response for $t < 30us$

Figure 10 plot shows that the capacitor is charging up to meet the input amplitude. The high frequency component can be seen as a ripple in the plot. This component is highly attenuated and hence not visible in the Figure 11 plot. In the fast time plot, we see that the low frequency component passes almost unchanged, the amplitude is almost 1.

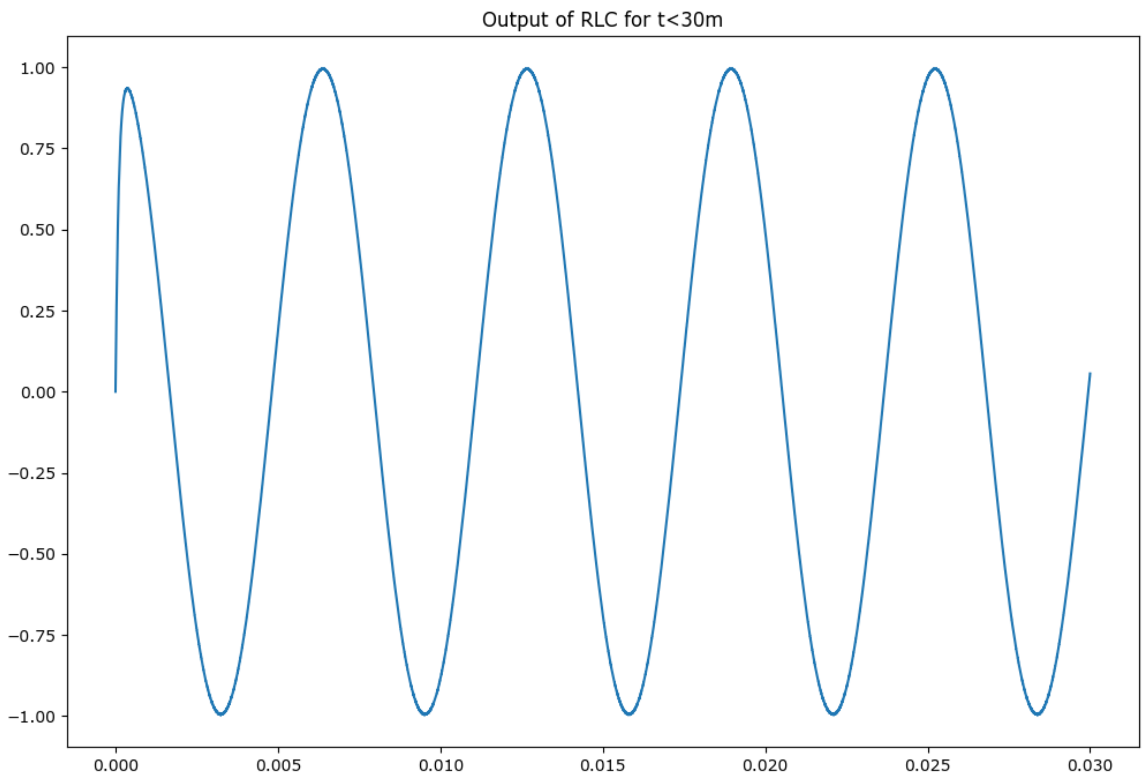

Figure 11: System response for $t < 30ms$

Conclusion

In this assignment, we have used *scipy's* signal processing library to analyze LTI systems. We analysed an RLC circuit and made use of *scipy* functions to analyse Laplace Transforms.