# EE2703 : Applied Programming Lab
# Assignment 6
# Report

Kesava Aruna Prakash R L
EE20B061

March 26, 2022

## Abstract

This week's assignment focuses on analysing linear time-invariant systems (LTI systems) using python tools. Transfer functions of systems are solved in time domain using functions in `scipy.signal` module.

## Question 1

An external time varying input $f(t)$ is applied to a spring system.

$$f(t) = \cos(1.5t)\exp(-0.5t)u(t)$$

Laplace transform of $f(t)$ is $F(s)$

$$F(s) = \frac{s + 0.5}{(s + 0.5)^2 + 2.25}$$

The differential equation for the position of the spring $x$ is :

$$x''(t) + 2.25x(t) = f(t)$$

The conditions for $x$ are $x(0) = 0$ and $x'(0) = 0$. The solution to the above equation can be obtained using `signal.impulse()` which finds the impulse response to a LTI system. Giving the Laplace domain equation of the differential equation as input to `signal.impulse()` will effectively return the time-domain solution of $x$. Time response of the spring between t = 0 to t = 50 seconds is:

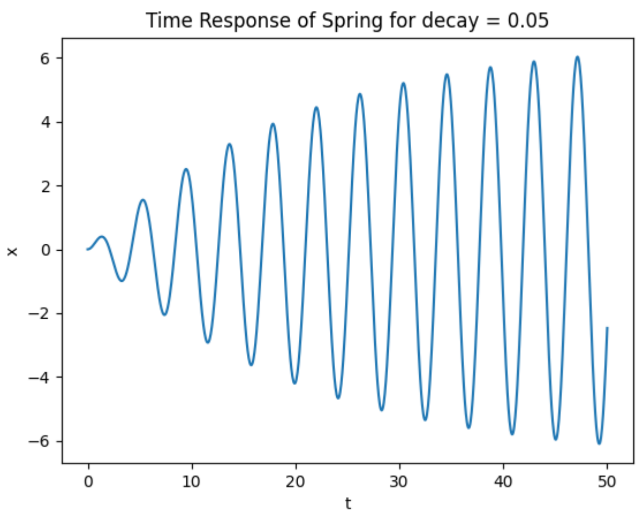


Figure 1: Time response of the spring: t = 0 to t = 50 seconds

Python code for execution of the above:

```
def f_Transfer(decay):
    num = np.poly1d([1,decay])
    denom = num**2+2.25
    return num, denom

num1,denom1 = f_Transfer(0.05)

f = np.poly1d([1,0,2.25])
H1 = sp.lti(num1,f*denom1)
t = np.linspace(0,50,1001)

t1,h1 = sp.impulse(H1,None,t)
pt.title("Time Response of Spring for decay = 0.05")
pt.xlabel("t")
pt.ylabel("x")
pt.plot(t1,h1)
pt.show()
```

## Question 2

The time response of the spring is calculated for the same system with a different decay of 0.05 in the input.

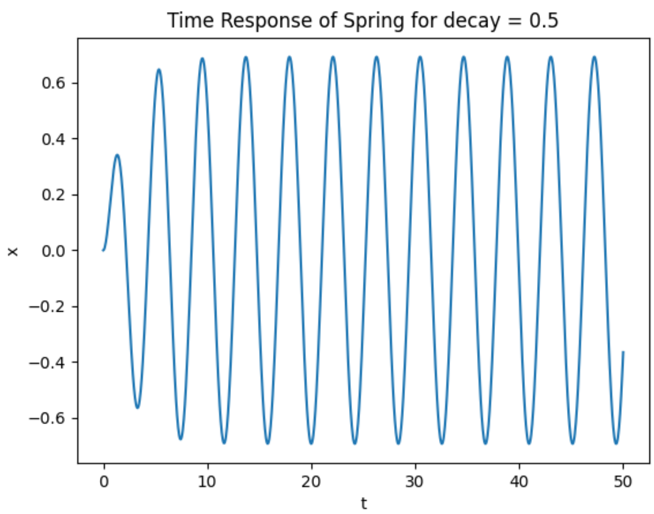


Figure 2: Time response of the spring: t = 0 to t = 50 seconds

## Question 3

The time response of the spring is calculated in a different approach by analysing the spring as a LTI system and finding the output of the system($x(t)$) from the input using the transfer function of the system using `signal.lsim()` function.
$f(t)$ is the input and $x(t)$ is the output.
Transfer function of the system is $\frac{X(s)}{F(s)}$ while $X(s)$ is calculated from the equation:

$$x''(t) + 2.25x(t) = u(t)$$

The same calculation is done for different frequencies of the cosine term in $f(t)$.
Python code for execution of the above:

```
def f_t(t,decay,freq):
    return np.cos(freq*t)*np.exp(-1*decay*t)
X = sp.lti([1],[1,0,2.25])

w = np.arange(1.4,1.6,0.05)

leg = []
for i in w:
    ft = f_t(t,0.05,i)
    t,y,svec = sp.lsim(X,ft,t)
    pt.plot(t,y)
    leg.append(f"w= {i}")
pt.title("Time Response of Spring for different frequencies")
pt.legend(leg)
pt.xlabel("t")
pt.ylabel("x")
pt.show()
```

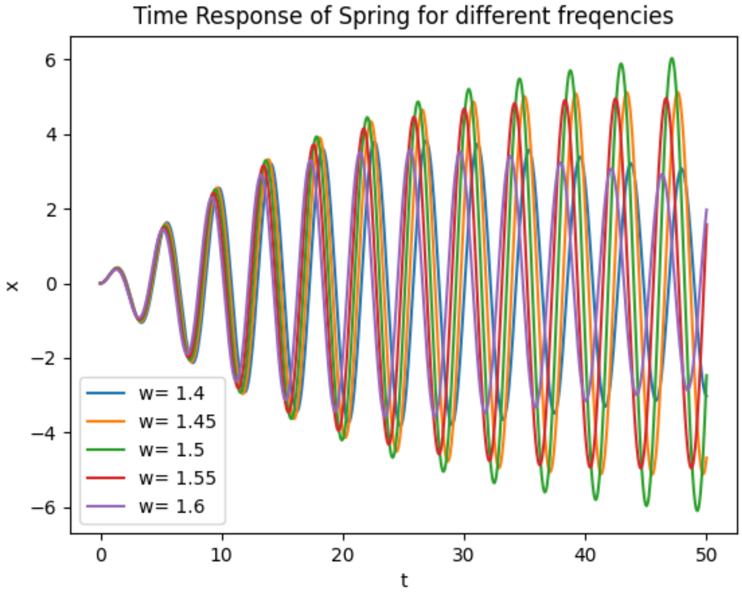


Figure 3: Time response of the spring: t = 0 to t = 50 seconds for different frequencies

## Question 4

A system of two coupled springs is analysed. The system is resolved into two different outputs $x(t)$ and $y(t)$, and transfer functions $X(s)$ and $Y(s)$.

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

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

Python code for solving them:

```
X_s = sp.lti([1,0,2],[1,0,3,0])
Y_s = sp.lti([2],[1,0,3,0])

tn = np.linspace(0,20,201)
tx,x = sp.impulse(X_s,None,tn)
ty,y = sp.impulse(Y_s,None,tn)

pt.plot(tx,x)
pt.plot(ty,y)
```

```
pt.xlabel("t")
pt.ylabel("Position")
pt.title("Time Response for Coupled Strings")
leg1 = ["x","y"]
pt.legend(leg1)
pt.show()
```

Solving them for time interval t = 0 to t = 20 seconds:

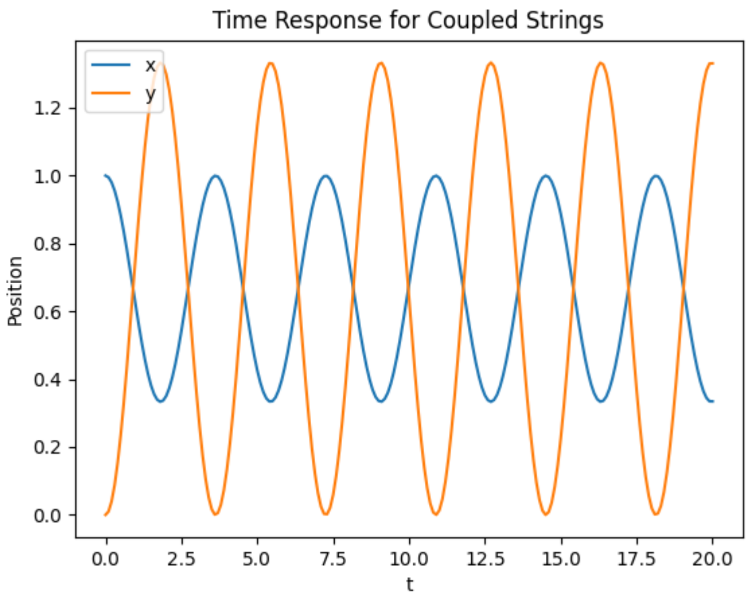


Figure 4: Time response of the coupled springs: t = 0 to t = 20 seconds

## Question 5

The given series RLC circuit is a LTI system with a steady state transfer function given by:

$$H(s) = \frac{1}{1 + s(RC) + s^2(LC)}$$

Where,
$R$, $C$ and $L$ are the corresponding resistance, capacitance and inductance of the linear elements respectively.
Bode plots for magnitude and phase:

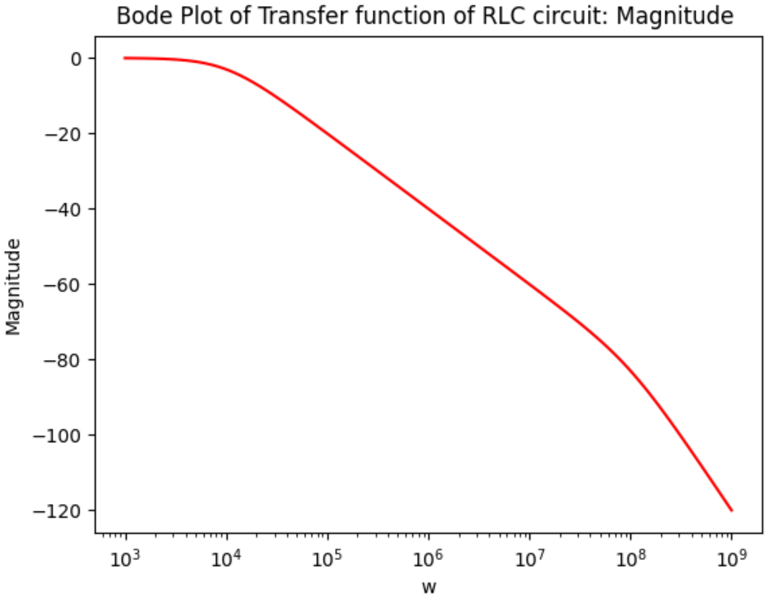


Figure 5: Bode Magnitude plot for RLC Circuit

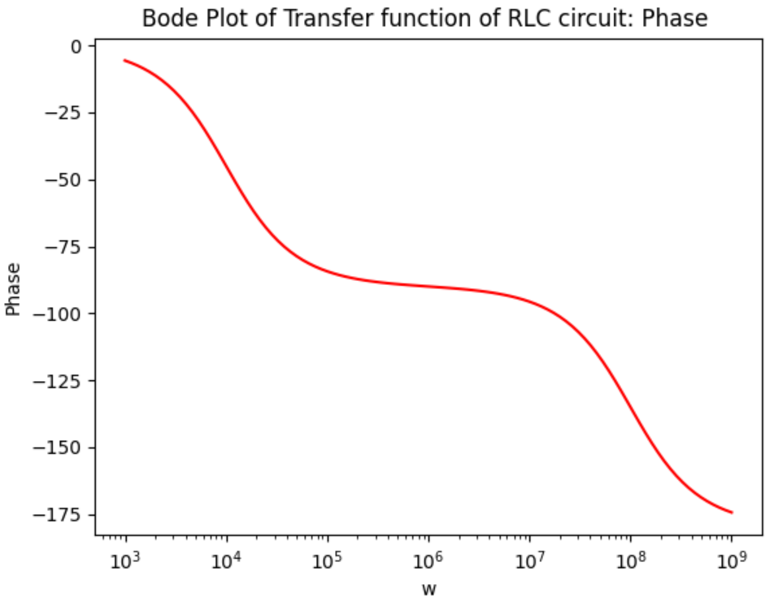


Figure 6: Bode phase plot for RLC Circuit

## Question 6

An input $v_i(t)$ is given to the RLC circuit and output $v_o(t)$ is analysed for two different time frames.

$$v_i = \cos{(10^3 t)}u(t) - \cos{(10^6 t)}u(t)$$

Python code for the calculation:

```
t_large = np.arange(0,1e-2,1e-7)
t_small = np.arange(0,30e-6,1e-7)
vi = lambda t: np.cos((10**3)*t)-np.cos((10**6)*t)

tl,vo_large,svec = sp.lsim(H_RLC,vi(t_large),t_large)
pt.title("Plot of Output Voltage : 0 to 10ms")
pt.xlabel("Time")
pt.ylabel("Voltage")
pt.plot(tl,vo_large)
pt.show()

ts,vo_small,svec = sp.lsim(H_RLC,vi(t_small),t_small)
pt.title("Plot of Output Voltage : 0 to 30us")
pt.xlabel("Time")
pt.ylabel("Voltage")
pt.plot(ts,vo_small)
pt.show()
```

Plots for $v_o(t)$ :

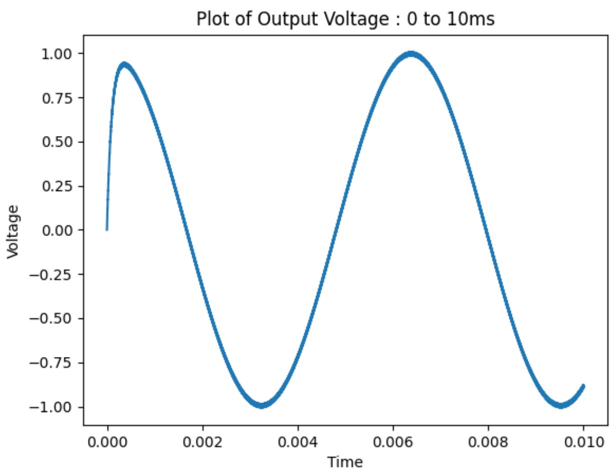


Figure 7: $v_o(t)$ plot for RLC Circuit: 0 to 10 ms

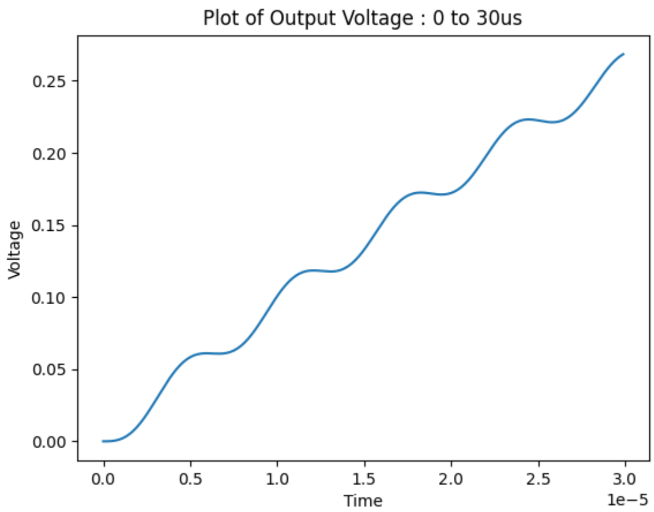


Figure 8: $v_o(t)$ plot for RLC Circuit: 0 to 30 us

## Inference for Questions 3 and 6

- In Question 3, amplitude of the displacement of the spring reaches the maximum when the frequency of the input $f(t)$ is 1.5 rad/s because the natural frequency of the spring is also 1.5 rad/s.
- In Question 6, the system acts a low pass filter and filters out the high frequency term in $v_i(t)$ to give the low frequency term in $v_i(t)$ as output with very small phase difference and almost now scaling.
- The smaller time frame plot can be explained as the lag between output and input.