The Laplace Transform

Mohammed Muqeeth
EE16B026
Electrical Engineering
IIT Madras

March 19, 2018

Abstract

Laplace Transform is a very powerful mathematical tool applied in various areas of engineering.We use laplace transform here to analyse linear time invariant systems with initial conditions using signal toolbox of Python.

1 Introduction

- Python has a way to define polynomials of single variable.We can define a polynomial sending coefficients as arguments. Eg polyld([1,2,3]) constructs $1.x^2 + 2.x + 3$.Similarly polyadd polymul commands add,multiply two polynomials given as argument. Eg polyadd([1,2],[1,2,3]) returns array([1,3,5]) which is $1.x^2 + 3.x + 5$, polymul([[1,2],[1,2,3]) retuns array([1,4,7,6]) $1.x^3 + 4.x^2 + 7.x + 6$.
- The signal toolbox of python has different functions. Eg lti,impulse,lsim required under scipy.signal module.Import scipy.signal as sp. Now sp has all functions listed in example.
- sp.lti helps to define a transfer function in rational form. Eg .H=sp.lti([1,2,3],[3,2,0,1]) defines transfer function of form $H(s) = \frac{s^2+2s+3}{3s^3+2s^2+1}$.The bode function on H returns frequency, magnitude , phase .Eg. w,S,phi = H.bode().which can be used for bode plots.
- sp.impulse functions retuns impulse response for a given transfer function.Eg. t,x=sp.impulse(H,None, linspace(0,10,101))
- sp.lsim does convolution of given input signal with impulse response of transfer function. Eg. t=linspace(0,10,101). u=sin(t). t,y,svec=sp.lsim(H,u,t)
- Here we take certain examples to illustrate above listed function of scipy.signal module.

2 Time response of Spring .

2.1 Method and Discussion

- Let us consider an lti system with input $f(t)$and output $x(t)$.The initial conditions are $x(0) = 0$ and $\dot{x}(0) = 0$.Laplace transform of Input $f(t) = \cos(1.5t)\exp(-0.5t)u_o(t)$ is given by $F(s) = \frac{s+0.5}{(s+0.5)^2+2.25}$The system satisfies differential equation given as

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

$$s^2X(s) + 2.25X(s) = F(s)$$

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

- We can use sp.impulse to find $x(t)$for above given $X(s)$.This method is repeated for $f(t) = \cos(1.5t)\exp(-0.05t)u_o(t)$ with laplace given as $F(s) = \frac{s+0.05}{(s+0.05)^2+2.25}$.
- Now we vary frequency of cosine in $f(t)$ from 1.4 to 1.6 in steps of 0.05.The transfer function of system $H(s) = X(s)/F(s) = \frac{1}{s^2+2.25}$.We use sp.lsim with arguments $H(s)$and input $f(t)$ to calculate $x(t)$.

Figure 1: Time response of spring for decay=0.5

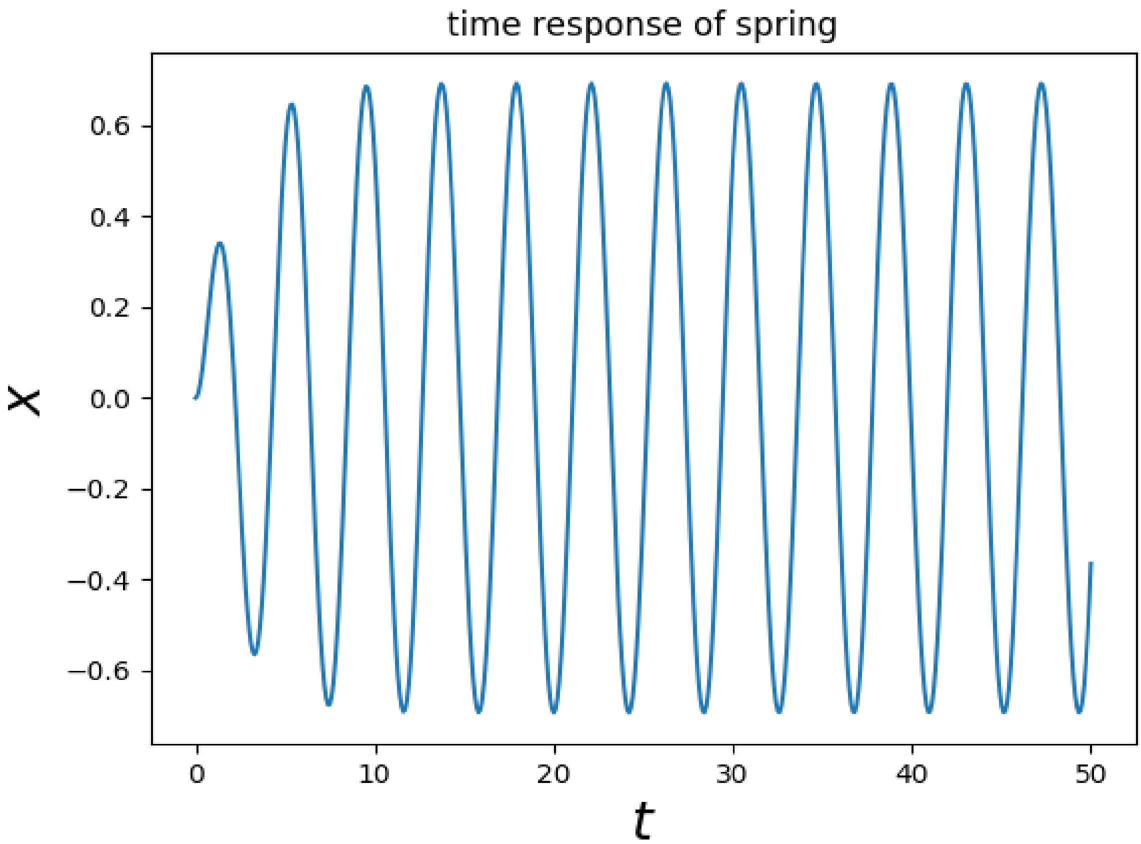


Figure 2: Time response of spring for smaller decay = 0.05

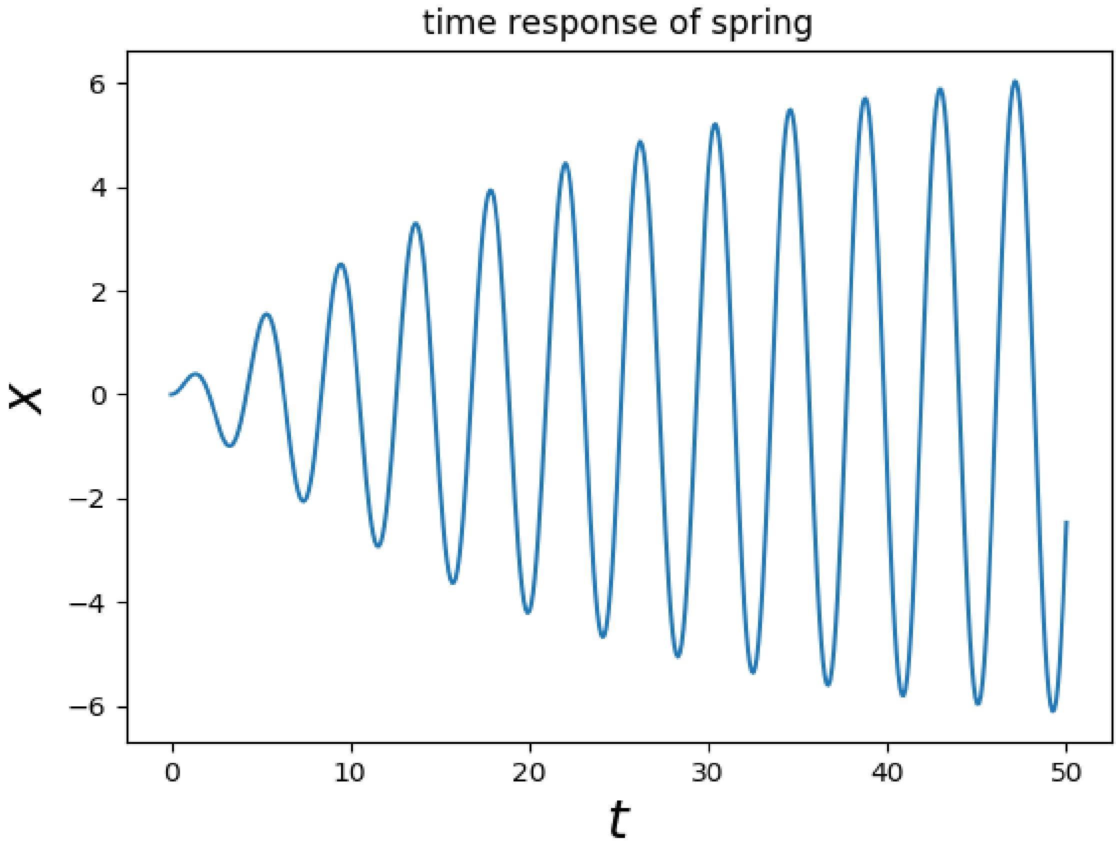
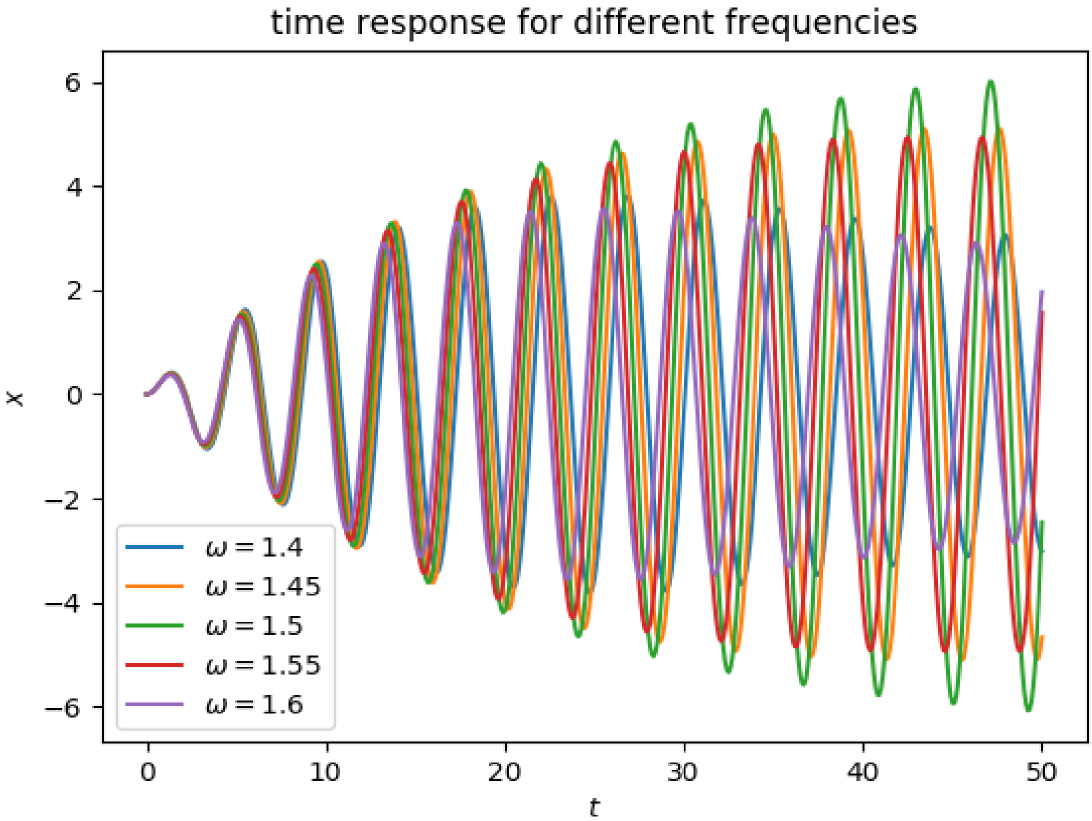
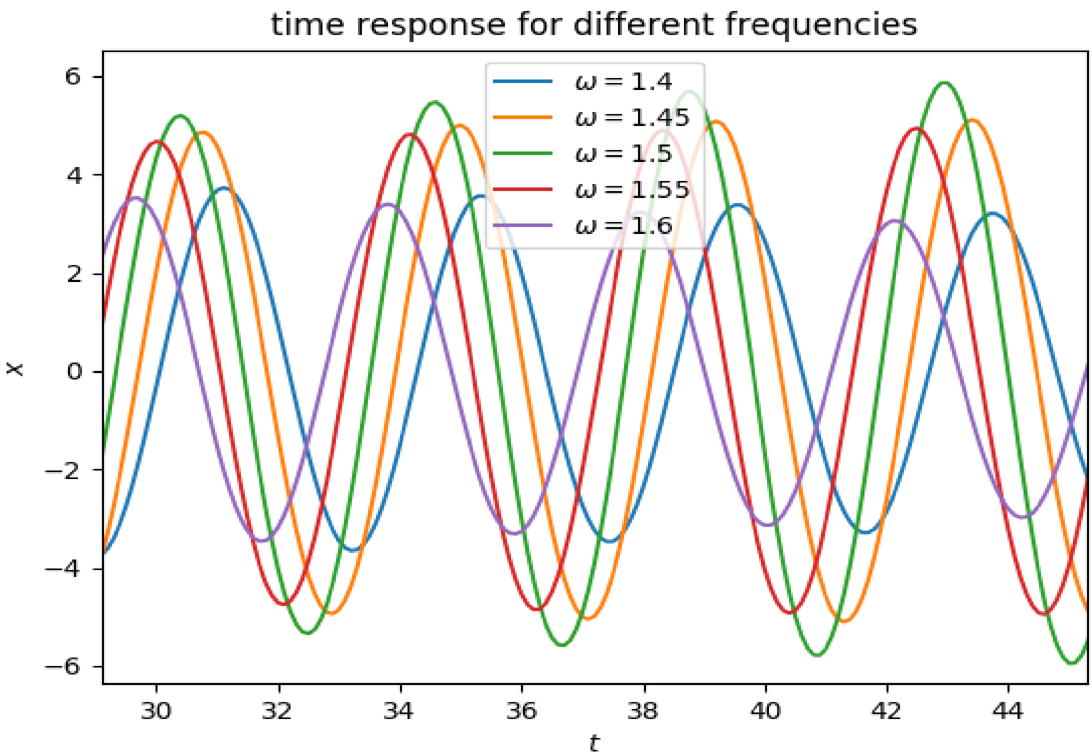


Figure 3: Time response varying frequency of cosine from 1.4 to 1.6





- The given system has poles on imaginary axis 1.5j and -1.5j. Therefore natural response never dies out.Since forcing function $f(t)$ decays with time . After long time, output will be dominated by natural response.
- From figure 1 and figure 2 we observe that response with smaller decay slowly rises whereas that of decay $= 0.5$ rises at beginning itself.

- Figure 3 has time response for different frequencies.For frequency of 1.5 the amplitude peak is highest of all.

2.2 Code

```
#define transfer function X(s)
#X(s) = s+0.5/(((s+0.5)*(s+0.5)+2.25]*[s*s+2.25))
num = poly1d([1,0.5])
den = polymul([1,1,2.5],[1,0,2.25])
X = sp.lti(num,den)
t,x=sp.impulse(X,None,linspace(0,50,501))
plt.figure(0)
title('time response of spring')
xlabel('$t$',fontsize=20)
ylabel('$x$',fontsize=20)
plot(t,x)
#define transfer function X(s)
#X(s) = s+0.05/(((s+0.05)*(s+0.05)+2.25]*[s*s+2.25))
num = poly1d([1,0.05])
den = polymul([1,0.1,2.2525],[1,0,2.25])
X = sp.lti(num,den)
t,x=sp.impulse(X,None,linspace(0,50,501))
plt.figure(1)
title('time response of spring')
xlabel('$t$',fontsize=20)
ylabel('$x$',fontsize=20)
plot(t,x)
#probmem3
for a in arange(1.4,1.65,0.05):
    t =linspace(0,50,501)
    u = cos(a*t)*exp(-0.05*t)
    num = poly1d([1])
    den = poly1d([1,0,2.25])
    H = sp.lti(num,den)
    t,x,svec=sp.lsim(H,u,t)
    plt.figure(2)
    plot(t,x)
title('time response for different frequencies')
plt.legend(['$\omega=1.4$','$\omega=1.45$','$\omega=1.5$','$\omega=1.55$','$\omega=1.6
xlabel('$t$')
ylabel('$x$')
```

3 Coupled Spring Problem.

3.1 Method and Discussion

- Consider a coupled spring which satisfies the differential equation with initial conditions $x(0) = 1, \dot{x}(0) = \dot{y}(0) = y(0) = 0$ given below

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

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

- Applying laplace transform on above equation results : $s^2X(s) - s + X(s) - Y(s) = 0$,$s^2Y(s) + 2Y(s) - 2X(s) = 0$ solving these two equations result in $X(s) = \frac{2+s^2}{s^3+3s}$ $Y(s) = \frac{2}{s^3+3s}$.

- These transfer functions can be used as input of sp.impulse functions which returns impulse response ie $x(t)$ $y(t)$. The plot of $x(t)y(t)$ are in figure 3.

$$2\ddot{x} + \ddot{y} = 0$$

$$2\dot{x} + \dot{y} = 0$$

$$2x + y = 2$$

- The relation between the two differential equations is $2x(t) + y(t) = 2$ which is linear . The corresponding plot is shown in figure 4.

Figure 4: $x(t), y(t)$satisfying $\ddot{x} + (x - y) = 0, \ddot{y} + 2(y - x) = 0$ with initial conditions $x(0) = 1, \dot{x}(0) = \dot{y}(0) = y(0) = 0$

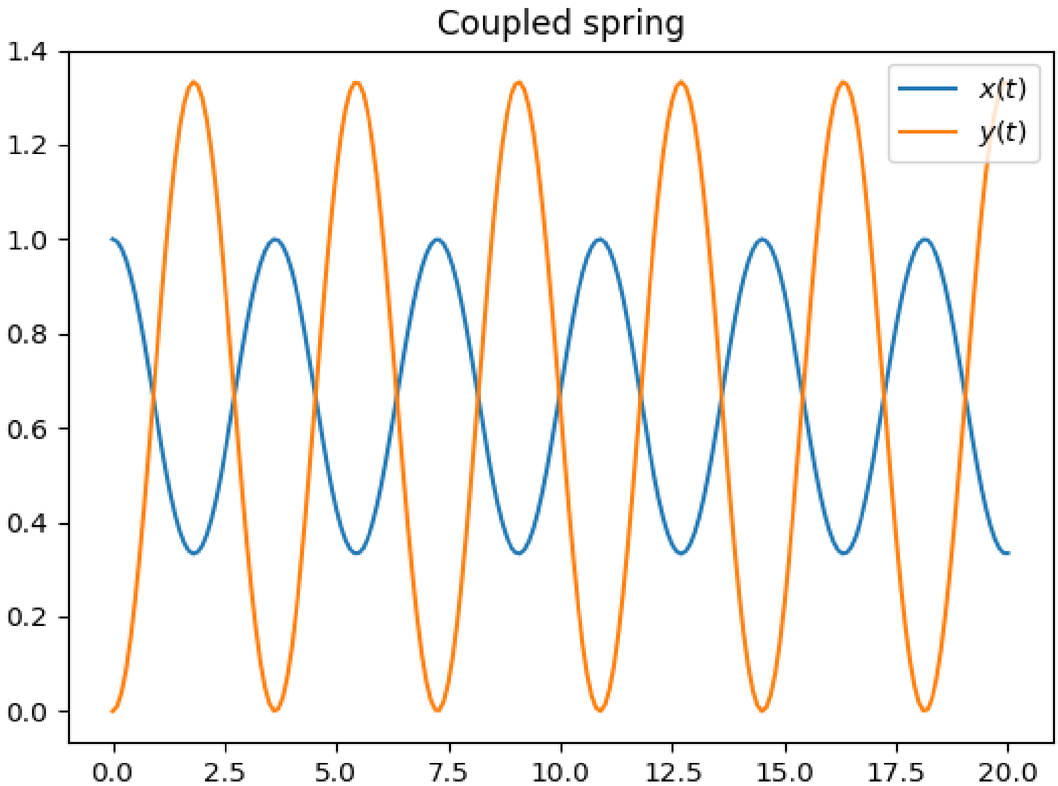
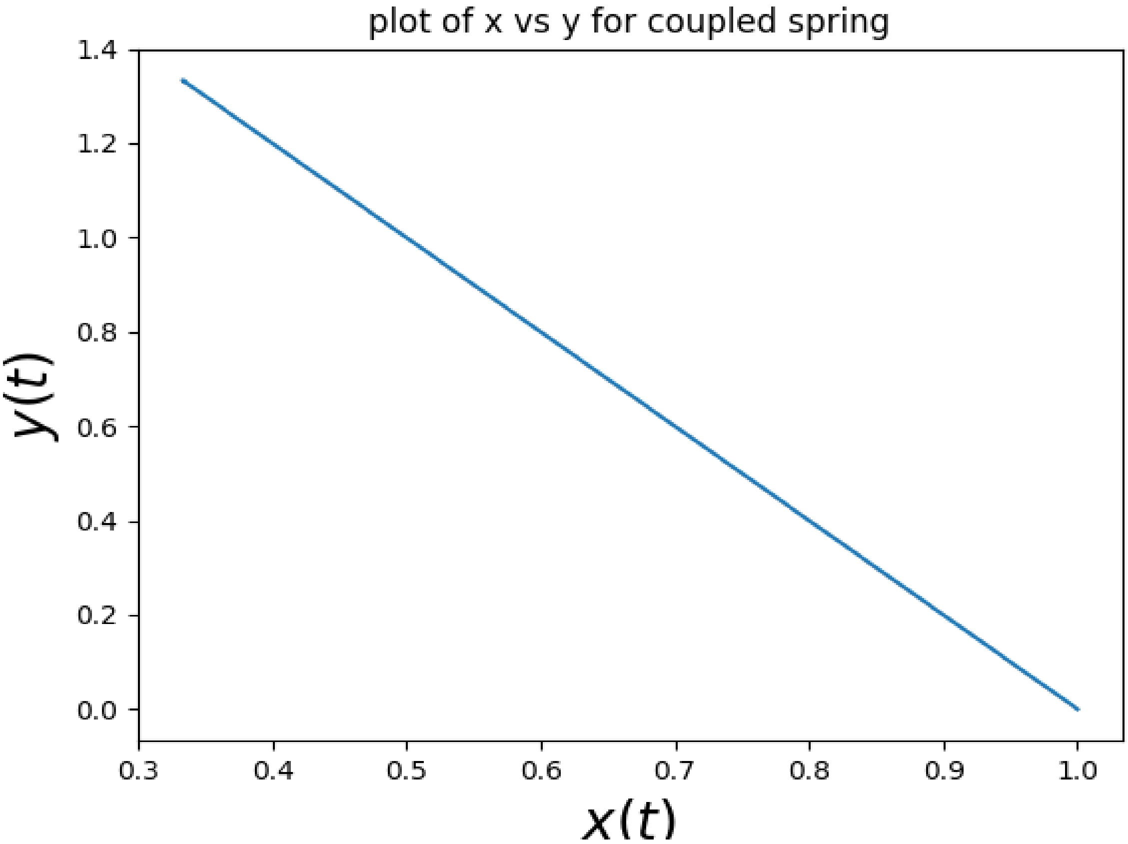


Figure 5: $x(t)$vs$y(t)$for given coupled spring



3.2 Code

```
#problem 4
numx = poly1d([1,0,2])
denx = poly1d([1,0,3,0])
X = sp.lti(numx,denx)
numy = poly1d([2])
deny = poly1d([1,0,3,0])
Y = sp.lti(numy,deny)
t,x=sp.impulse(X,None,linspace(0,20,201))
t,y=sp.impulse(Y,None,linspace(0,20,201))
plt.figure(3)
title('Coupled spring')
plot(t,x)
plot(t,y)
plt.legend(['$x(t)$','$y(t)$'])
plt.figure(4)
title('plot of x vs y for coupled spring')
xlabel('$x(t)$',fontsize=20)
ylabel('$y(t)$',fontsize=20)
plot(x,y)
```

4 RLC network .

4.1 Method and Discussion

- The transfer function of series lcr network is given as $H(s) = \frac{1}{s^2LC+sCR+1}$ where $LC = 1e^{-12}$ $RC = 1e^{-4}$.The input is sinusoid of two frequencies $v_i(t) = \cos(10^3t)u(t) - \cos(10^6t)u(t)$.

- The output voltage is obtained by giving $H(s)v_i(t)$ to sp.lsim function.

- The magnitude and phase can be obtained by using bode function. It is implemented as $w, S, phi = H.bode()$.S has magnitude in dB, phi has phase in degree.

Figure 6: Bode plot of $H(s) = \frac{1}{s^2 10^{-12} + s 10^{-4} + 1}$

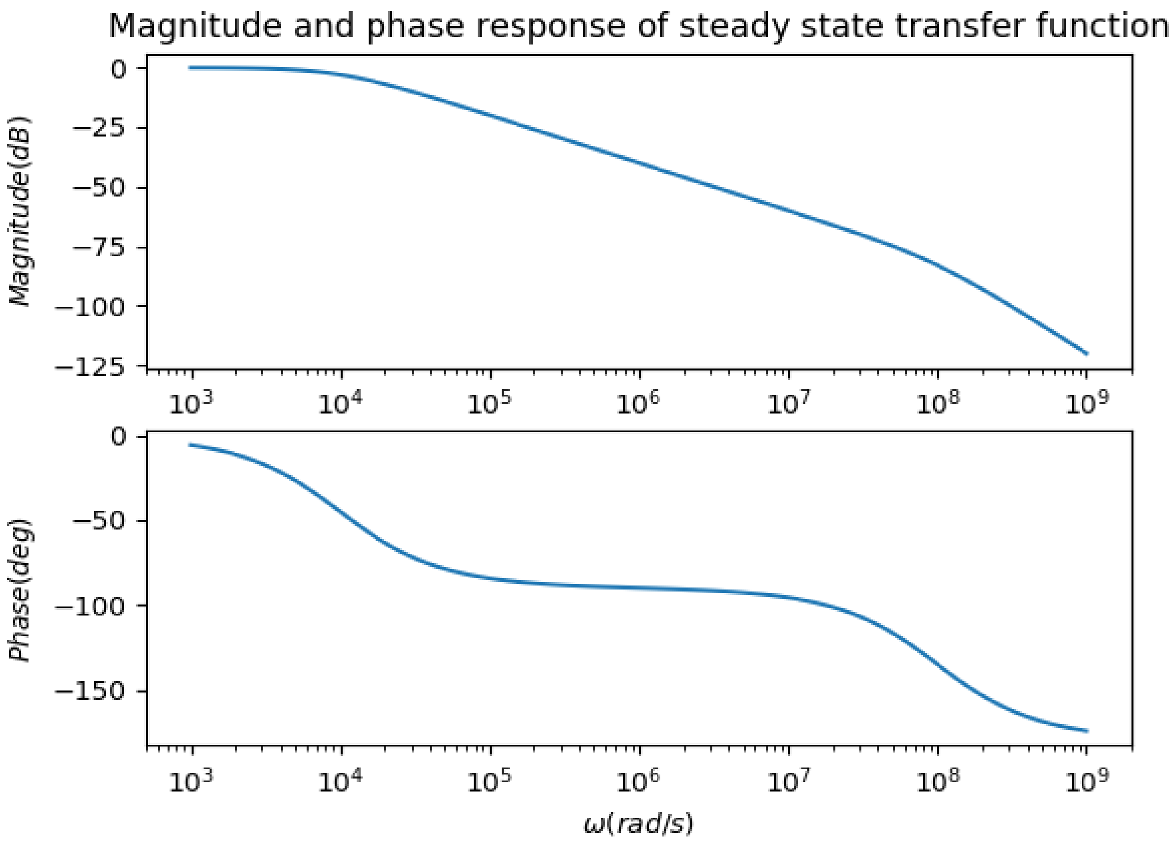


Figure 7: Voltage across capacitor

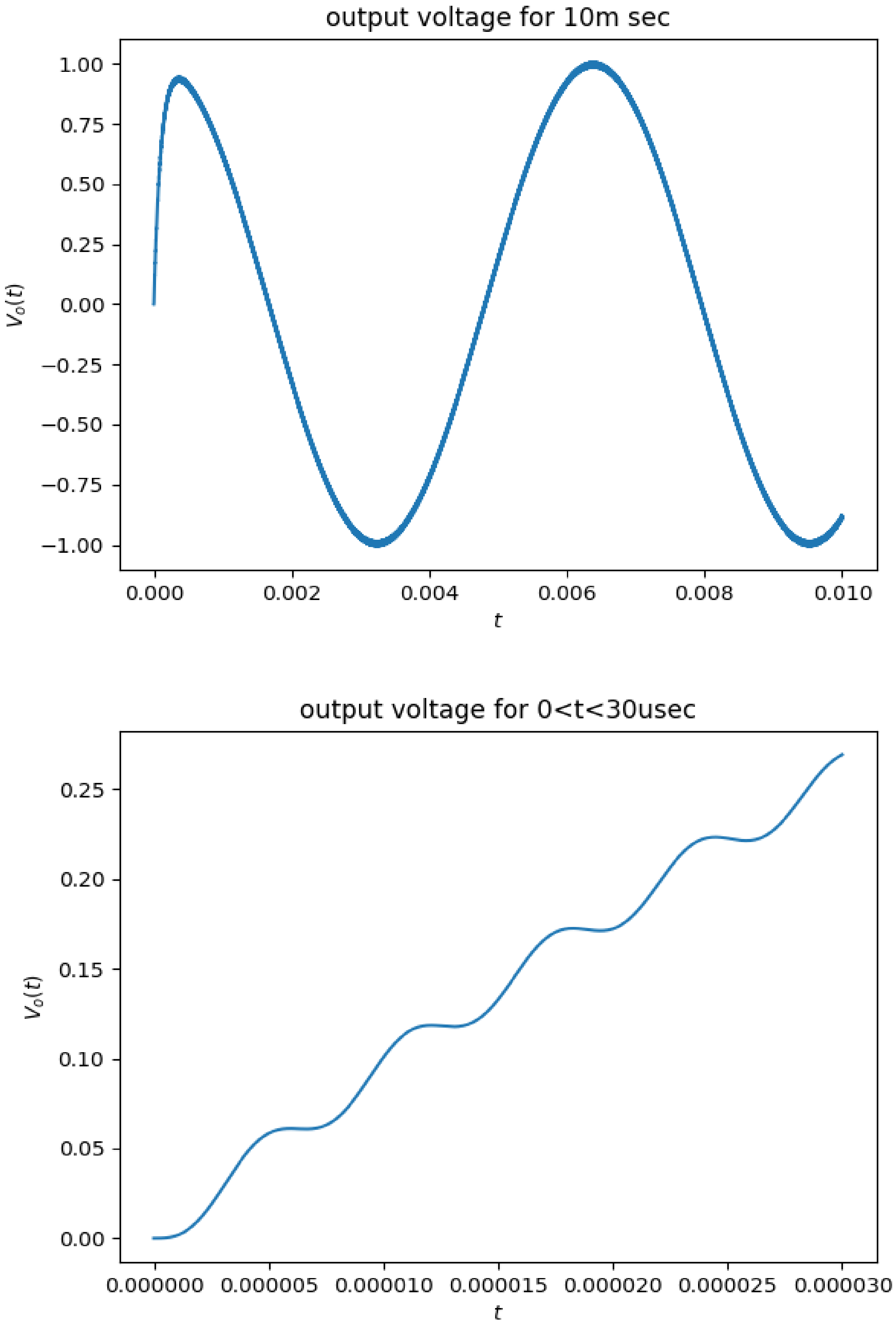
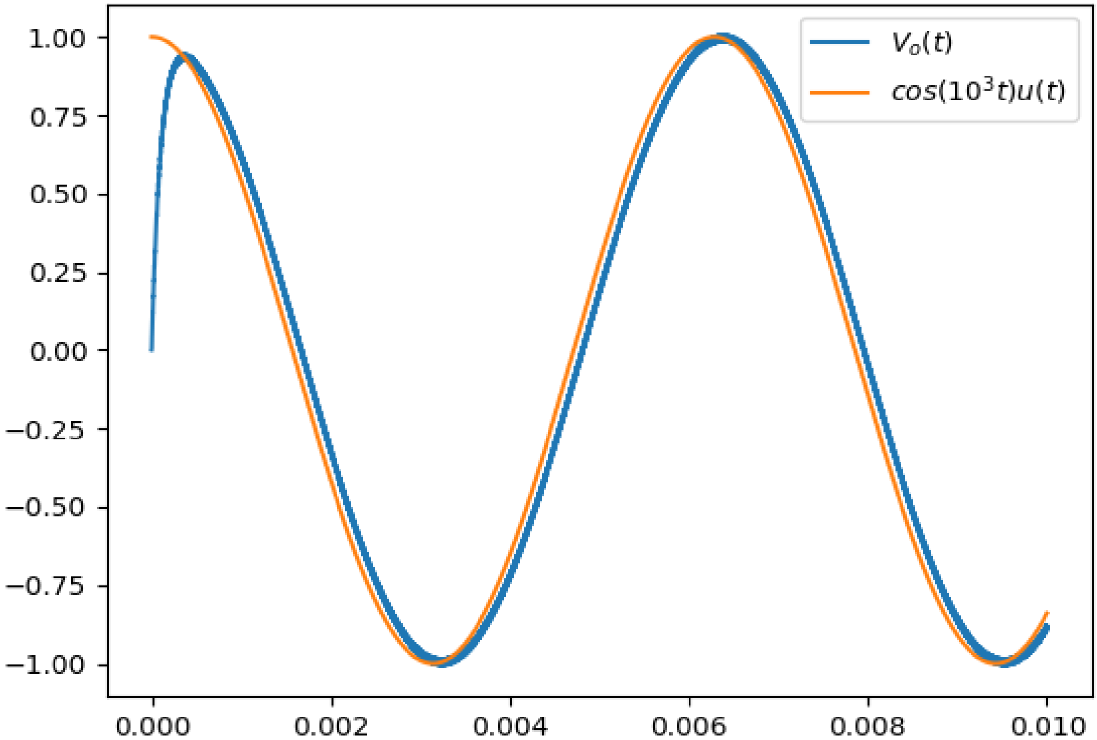


Figure 8: Plot of $v_o(t)$and $\cos(10^3t)$



- The output of system for sinusoid of frequency $\omega$is $|H(j\omega)|\cos(\omega t + \angle H(j\omega)$.From bode plot it is clear that for frequency of $10^3H(j\omega) \approx 1$ and $\angle H(j\omega) \approx 0$ but frequency $10^6$is attenuated more. Therefore the output is nearly $\cos(10^3t)$ with some phase shift shown in figure 8.
- The output has ripples in it as shown in figure 7 because $\cos(10^6t)$is not completely attenuated to zero.It results in small variation in output shown as ripples.
- Close to zero in figure 8. The input is zero since input has difference of two cosine terms. whereas the output is dominated by cosine of frequency 10^3 which is close to 1.

4.2 Code

```
#problem5
#H(s) = 1/(s2LC+SCR+1) LC = e-12,RC = e-4
num = poly1d([1])
den = poly1d([pow(10,-12),pow(10,-4),1])
H = sp.lti(num,den)
w,S,phi=H.bode()
title('Magnitude and phase response of steady state transfer function')
plt.figure(5)
subplot(2, 1, 1)
ylabel('$Magnitude(dB)$')
semilogx(w,S)
subplot(2, 1, 2)
ylabel('$Phase(deg)$')
xlabel('$\omega(rad/s)$')
semilogx(w,phi)
t =arange(0,0.01,pow(10,-7))
```

```
u = cos(pow(10,3)*t) - cos(pow(10,6)*t)
num = poly1d([1])
den = poly1d([pow(10,-12),pow(10,-4),1])
H = sp.lti(num,den)
t,y,svec=sp.lsim(H,u,t)
plt.figure(6)
title('output voltage for 10m sec')
ylabel('$V_o(t)$')
xlabel('$t$')
plot(t,y)
plt.figure(8)
plot(t,y)
plot(t,cos(pow(10,3)*t))
plt.legend(['$V_o(t)$','$cos(10^3t)u(t)$'])
t =arange(0,3*pow(10,-5),pow(10,-7))
u = cos(pow(10,3)*t) - cos(pow(10,6)*t)
num = poly1d([1])
den = poly1d([pow(10,-12),pow(10,-4),1])
H = sp.lti(num,den)
t,y,svec=sp.lsim(H,u,t)
plt.figure(7)
title('output voltage for 0<t<30usec')
ylabel('$V_o(t)$')
xlabel('$t$')
plot(t,y)
show()
```