Analysis of Circuits using Laplace Transforms

Name: Sankalp Saoji (EE16B063)

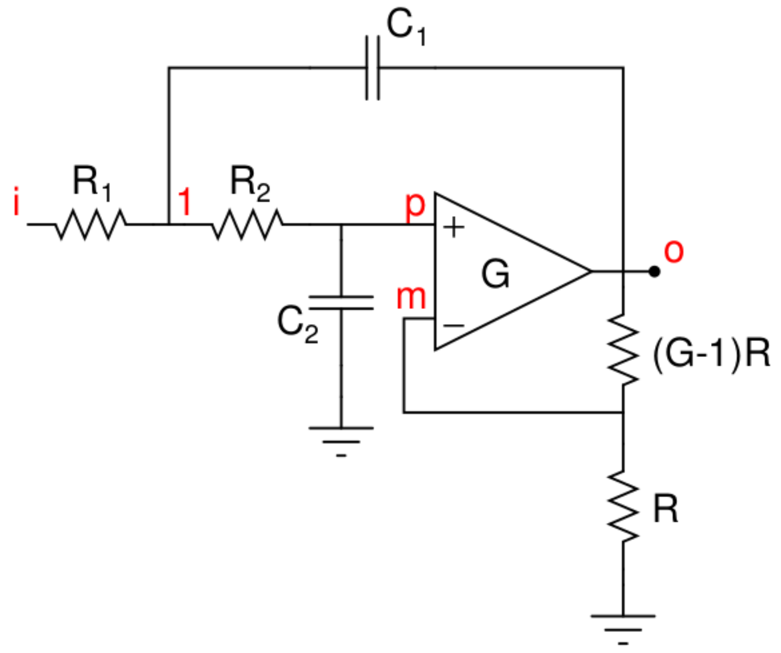
Date: 6/04/2018

Abstract

In this week, we looked upon two powerful capabilities of Python:
1) Symbolic Algebra
2) Analysing Cicuits using Laplace Transforms

1 INTRODUCTION

We considered the following low pass filter circuit,



So, we get a Butterworth filter of cutoff frequency $\frac{1}{2\pi}$ MHz. The circuit equations are,

V$_m$ = $\frac{V_0}{G}$, -(1)
V$_p$ = V$_1\frac{1}{1+j\varpi R_2C_2}$, -(2)
V$_0$ = G(V$_p - V_m$), -(3)

$\frac{V_i - V_1}{R_1} + \frac{V_p - V_1}{R_2} + \text{j}\varpi C_1(\text{V}_0 - V_1) = 0$ -(4)
Solving for V_0 in equation 3, we get,
$\text{V}_0 = \frac{GV_1}{2}\frac{1}{1+J\varpi R_2 C_2}$
Equation 4 becomes,
$\frac{V_i}{R_1} + \text{V}_1(-\frac{1}{R_1} + \frac{1}{R_2}\frac{1}{1+J\varpi R_2 C_2} - \frac{1}{R_2} + \text{j}\varpi C_1 \frac{G}{2}\frac{1}{1+J\varpi R_2 C_2} * \text{j}\varpi C_1) = 0$
The term involving G will dominate, and so we obtain,
$\text{V}_1 \approx \frac{2V_i}{G}\frac{1+j\varpi R_2 C_2}{j\varpi R_1 C_1}$
Substituting back into the expression for V_0 we finally get,
$\text{V}_0 \approx \frac{V_i}{j\varpi R_1 C_1}$

1.1 Using sympy to solve the above circuit problem :

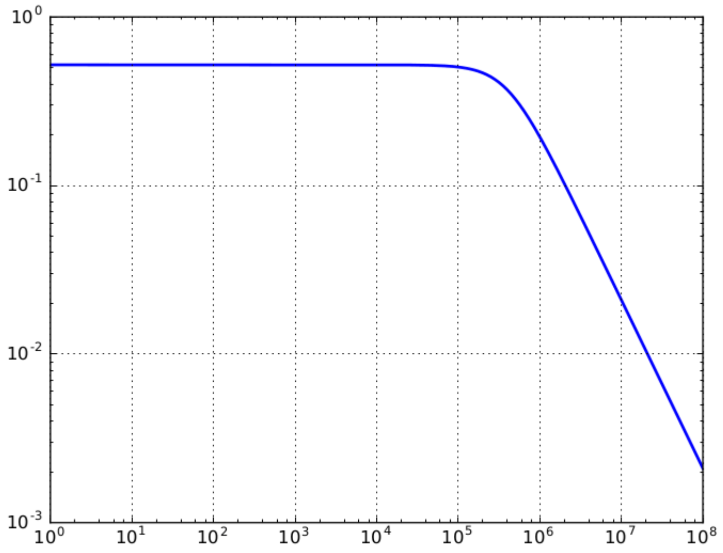
The above circuit problem reduces to the following matrix after doing nodal analysis. It is as shown below,

$$\begin{pmatrix} 0 & 0 & 1 & -\frac{1}{G} \\ -\frac{1}{1+sR_2C_2} & 1 & 0 & 0 \\ 0 & -G & G & 1 \\ -\frac{1}{R_1} - \frac{1}{R_2} - sC_1 & \frac{1}{R_2} & 0 & sC_1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_p \\ V_m \\ V_o \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ V_i(s)/R_1 \end{pmatrix}$$

The above matrix is solved as shown in below code. (This above analysis and the below code were all given in the document).

```
# Solving of the above problem

#Importing modules

from sympy import *
import pylab as p

#Defining function

def lowpass(R1,R2,C1,C2,G,Vi):
        s = symbols('s')
        A = Matrix([[0,0,1,-1/G],[-1/(1+s*R2*C2),1,0,0],
    [0,-G,G,1],[-1/R1-1/R2-s*C1,1/R2,0,s*C1]])
        b = Matrix([0,0,0,Vi/R1])
        V = A.inv()*b
    # Above solution is in s space.
        return (A,b,V)

#Plotting

A,b,V = lowpass(10000,10000,1e-9,1e-9,1.586,1)
print 'G=1000'
Vo = V[3]
print Vo
```

```
w = p.logspace(0,8,801)
ss = 1j*w
hf = lambdify(s,Vo,numpy)
v = hf(ss)
p.loglog(w,abs(v),lw = 2)
p.grid(True)
p.show()Varying the frequency of the cosine and plotting
```



Above is the way we have been given the plot of the output voltage for the low pass filter in the document.

2 QUESTIONS

2.1 Obtaining and plotting the step and impulse response of the low pass filter :

2.1.1 Getting step response H(jϖ) of above low pass filter circuit and plotting,

From above circuit, using nodal analysis, we get the following matrix,

$$\begin{bmatrix} 0 & 0 & 1 & \frac{-1}{G} \\ \frac{-1}{1+sR_2C_2} & 1 & 0 & 0 \\ 0 & -G1 & G1 & 1 \\ -sC_1 - \frac{1}{R_1} - \frac{1}{R_2} & \frac{1}{R_2} & 0 & sC_1 \end{bmatrix} x \begin{bmatrix} V' \\ V_p \\ V_n \\ V_o \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ V_i/R \end{bmatrix}$$

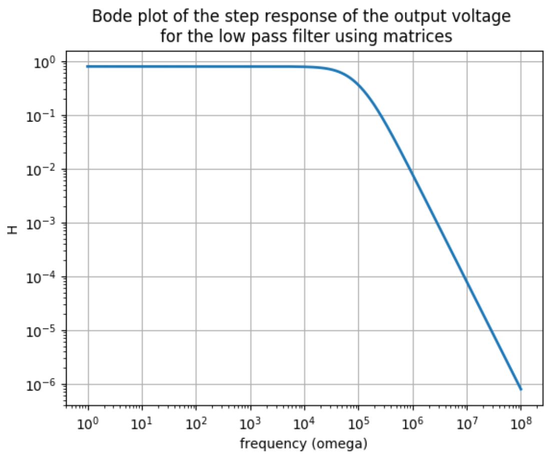
Here,
$R_1 = R_2 = 10k\Omega$,
$C_1 = C_2 = 10pF$,
G(open loop gain of opamp and the feedback factor) = 1.586

```
#Importing modules and libraries

import pylab as py
import numpy
import math
import matplotlib.pyplot as plt
import scipy.signal as sp
from sympy import *
```

In the above code, I have imported all the modules and libraries I will require in this assignment.

So, below is the code for the plotting of step response of the low pass filter.

```
#Creating a function and solving the matrices for the low pass filter

s = symbols('s')
def lowpass(R1,R2,C1,C2,G,Vi):
    A = Matrix([[0,0,1,-1/G],[(-1)/(1+s*R2*C2),1,0,0],
  [0,-G,G,1],[((-1)/(R1))-(1/R2)-s*C1,1/R2,0,s*C1]])
    b = Matrix([0,0,0,Vi/R1])
    V = A.inv()*b
    return (A,b,V)

#Obtaining output voltage, H(jw) graph for the low pass filter and plotting

A,b,V = lowpass(10000,10000,1e-9,1e-9,1.586,1)
Vo = V[3]
w = py.logspace(0,8,801)
ss = 1j*w
hf = lambdify(s,Vo,'numpy')
v = hf(ss)

plt.figure(0)
py.loglog(w,abs(v),lw = 2)
py.title('Bode plot of the step response of the output
voltage \n for the low pass filter using matrices')
py.xlabel('frequency (omega)')
py.ylabel('H')
py.grid(True)
py.show()
```



Above is the plot I obtained for H(jϖ) for the low pass filter. We easily see the pole at around 0.5 x 10^5.

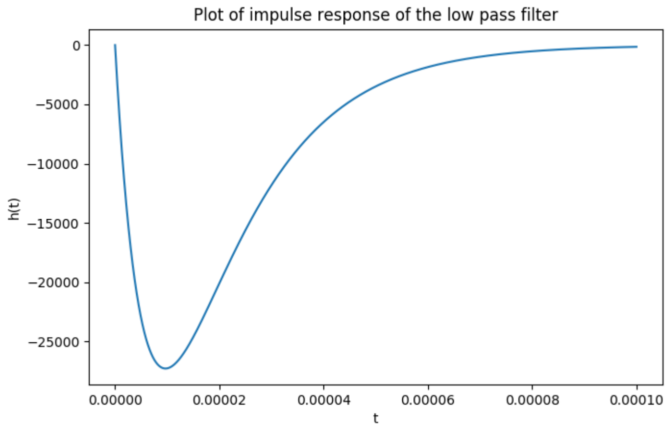
2.1.2 Getting impulse response h(t) of above low pass filter circuit and plotting,

Now, below, we obtain the response of the circuit in time domain and plot it.

Below is the code for it,

```
#Getting impulse response of the low pass filter

Vo = simplify(Vo)
num, denom = fraction(Vo)
Num = Poly(num,s)
Denom = Poly(denom,s)
num_coeff = Num.all_coeffs()
denom_coeff = Denom.all_coeffs()
num_coeff=[float(i) for i in num_coeff]
denom_coeff=[float(i) for i in denom_coeff]
H_num1 = numpy.poly1d(num_coeff)
H_denom1 = numpy.poly1d(denom_coeff)
H_1 = sp.lti(H_num1, H_denom1)
t_vo_low, h1 = sp.impulse(H_1, None, numpy.linspace(0,1e-4,1e5))

#Plotting impulse response of the low pass filter
```

```
plt.figure(1)
plt.plot(t_vo_low,h1)
plt.title('Plot of impulse response of the low pass filter')
plt.xlabel('t')
plt.ylabel('h(t)')
plt.show()
```



As seen above, this is the impulse response of the low pass filter in time domain.

2.2 Determining output voltage when a given input is applied and plotting the output voltage waveform for the low pass filter :

For the above low pass filter circuit, now the input is,

$v_i(t) = (\sin(2000\pi t) + \cos(2 \text{ x } 10^6\pi t))u_0(t)$ volts

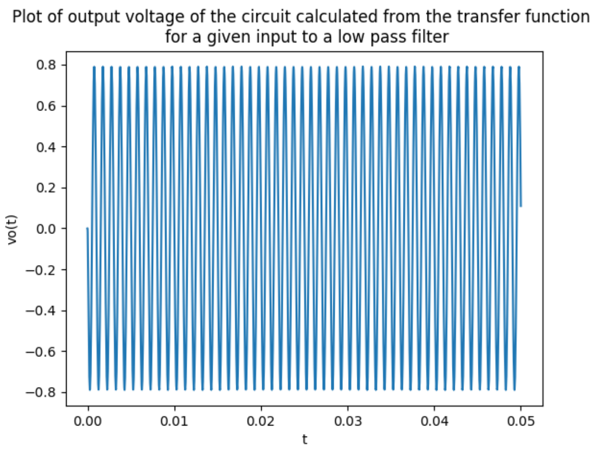
f(sine wave) = $\varpi_0/2\pi = 1000$

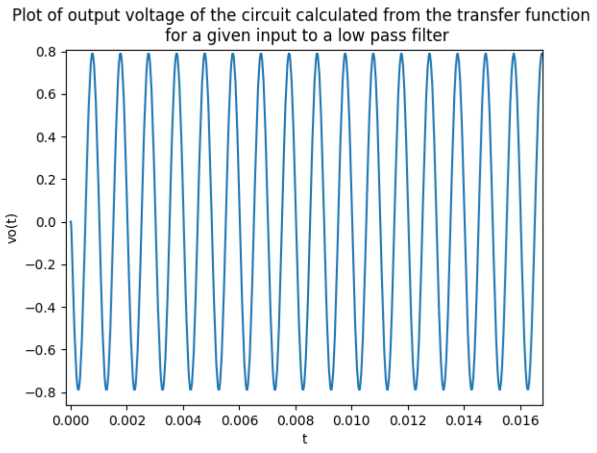
f(cosine wave) = $\varpi_0/2\pi = 1000000$

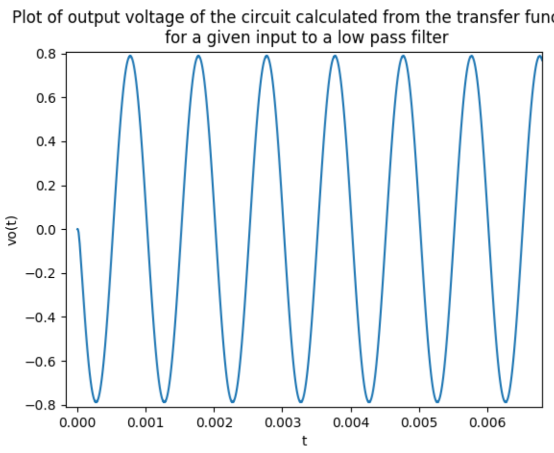
```
#Obtaining vo(t) from the transfer function of a low pass
filter when a given function is the input

t_vo1 = numpy.linspace(0,0.05,1e5)
vi = numpy.sin(2000*(math.pi)*t_vo1) + numpy.cos(2*(1e6)*(math.pi)*t_vo1)
t_transfer1, vo1, vec_transfer1 = sp.lsim(H_1, vi, t_vo1)

#Plotting vo(t) for the low pass filter

plt.figure(2)
plt.plot(t_vo1, vo1)
plt.title('Plot of output voltage of the circuit calculated
from the transfer function \n for a given input to a low pass filter')
```

```
plt.xlabel('t')
plt.ylabel('vo(t)')
plt.show()
```

In the above code, I have obtained the output voltage v_0(t) using 'sp.lsim' from the signal toolbox. Following is the plot I have obtained for the output voltage of the low pass filter,





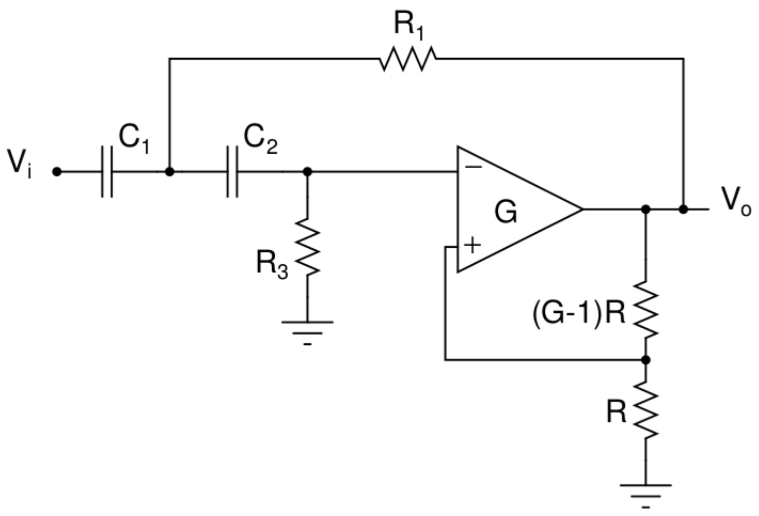


As we see from the above graph, the time period is very quite fine
T = 10^{-3} seconds
f = $\frac{1}{T}$ = 1 kHz
The graph looks like a sine wave!

Even when our input had two types of sinusoids, the sine sinusoid succeeded to pass through the filter as we have got the output frequency as 1 kHz which belongs to the sine sinusoid. So, as this frequency has been able to pass, we are sure that the circuit is a low pass filter.

2.3 Obtaining and plotting the step, impulse response snd output when a particular input is given to the high pass filter :

We have a high pass filter circuit now,



2.3.1 Getting step response H(jϖ) of above high pass filter circuit and plotting :

In this problem, I analysed the circuit using sympy and created a function similar to the one defined above. For the same choice of components and gain, this circuit is a highpass filter.

From above circuit, using nodal analysis, we get the following matrix,

$$\begin{bmatrix} 0 & 0 & 1 & \frac{-1}{G} \\ -sC_2 & sC_2+\frac{1}{R_3} & 0 & 0 \\ 0 & G & -G & -1 \\ sC_1+sC_2+\frac{1}{R_1} & -sC_2 & 0 & \frac{-1}{R_1} \end{bmatrix} x \begin{bmatrix} V' \\ V_p \\ V_n \\ V_o \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ sC_1V_i \end{bmatrix}$$

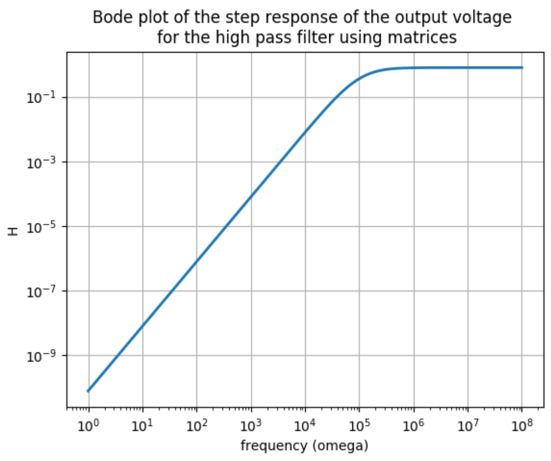
We have,

$R_1 = R_3 = 10k\Omega$,

$C_1 = C_2 = 1nF$,

G(open loop gain of opamp and the feedback factor) = 1.586

```
#Creating function and solving the matrices for the high pass filter

s = symbols('s')
def highpass(R1,R3,C1,C2,G,Vi):
        A = Matrix([[0,0,1,-1/G],[-s*C2,s*C2+1/R3,0,0],
[0,G,-G,-1],[1/R1+s*C2+s*C1,-s*C2,0,-1/R1]])
        b = Matrix([0,0,0,s*C1*Vi])
        V = A.inv()*b
        return (A,b,V)

#Obtaining output voltage, H(jw) graph for the high pass filter and plotting

A,b,V = highpass(10000,10000,1e-9,1e-9,1.586,1)
Vo = V[3]
w = py.logspace(0,8,801)
ss = 1j*w
hf = lambdify(s,Vo,'numpy')
v = hf(ss)

plt.figure(3)
py.loglog(w,abs(v),lw=2)
py.title('Bode plot of the step response of the output
voltage \n for the high pass filter using matrices')
py.xlabel('frequency (omega)')
py.ylabel('H')
py.grid(True)
py.show()
```



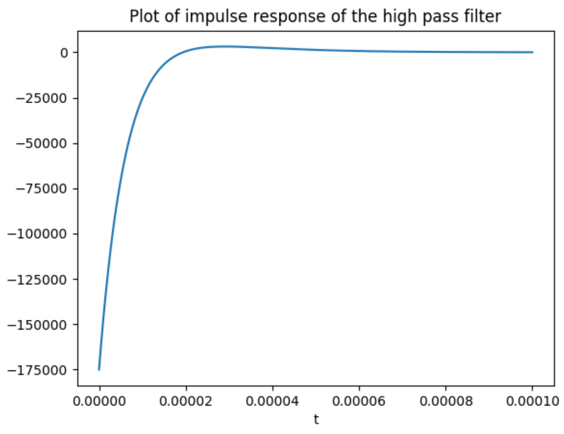
Above is the plot I obtained for H(jϖ) for the high pass filter. We easily see the pole at around 10^5.

2.3.2 Getting impulse response h(t) of above high pass filter circuit and plotting :

Now, below, we obtain the response of the circuit in time domain and plot it.

Below is the code for it,

```
#Getting impulse response of the high pass filter

Vo = simplify(Vo)
num, denom = fraction(Vo)
Num = Poly(num,s)
Denom = Poly(denom,s)
num_coeff = Num.all_coeffs()
denom_coeff = Denom.all_coeffs()
num_coeff=[float(i) for i in num_coeff]
denom_coeff=[float(i) for i in denom_coeff]
H_num2 = numpy.poly1d(num_coeff)
H_denom2 = numpy.poly1d(denom_coeff)
H_2 = sp.lti(H_num2, H_denom2)
t_vo_high, h2 = sp.impulse(H_2, None, numpy.linspace(0,1e-4,1e5))

#Plotting impulse response of the low pass filter

plt.figure(4)
plt.plot(t_vo_high,h2)
plt.title('Plot of impulse response of the high pass filter')
plt.xlabel('t')
plt.ylabel('h(t)')
plt.show()
```



As seen above, this is the impulse response of the high pass filter in time domain.

2.3.3 Determining output voltage when a given input is applied and plotting the output voltage waveform for the high pass filter :

Now, the input v_i to the circuit is,

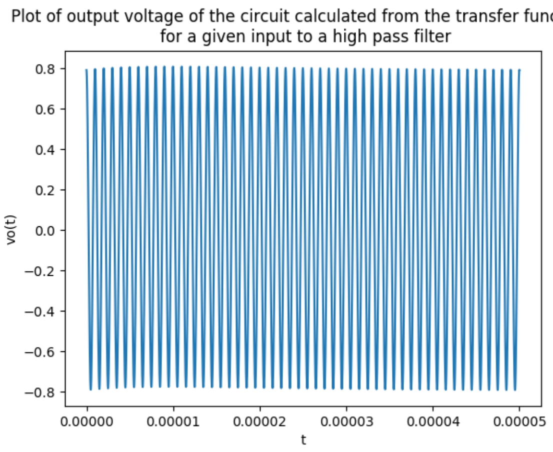
$v_i(t) = (\sin(2000\pi t) + \cos(2 \text{ x } 10^6\pi t))u_0(t)$ volts

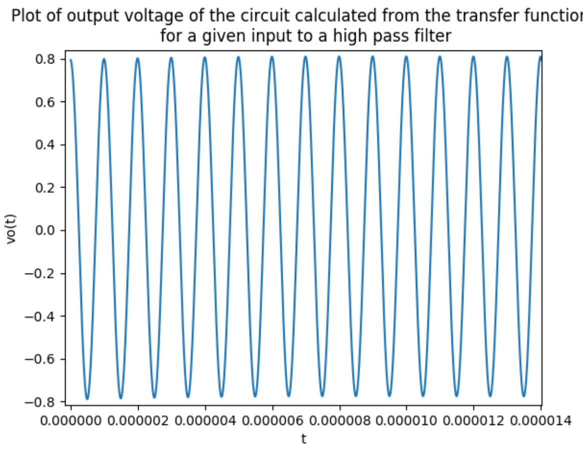
f(sine wave) = $\varpi_0/2\pi = 1000$

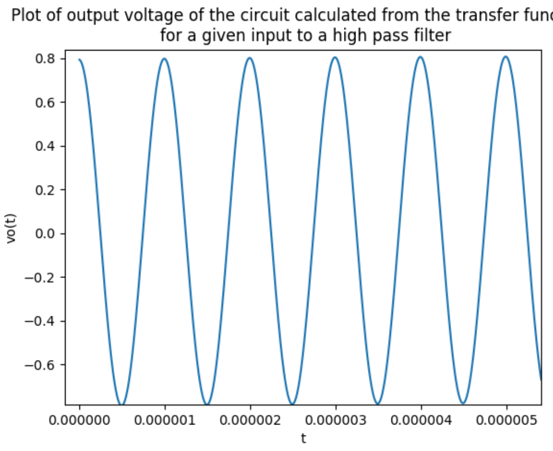
f(cosine wave) = $\varpi_0/2\pi = 1000000$

```
#Obtaining vo(t) from the transfer function of a high pass
 filter when a given function is the input

t_vo2 = numpy.linspace(0,5e-5,1e4)
vi = numpy.sin(2000*(math.pi)*t_vo2) + numpy.cos(2*(1e6)*(math.pi)*t_vo2)
t_transfer2, vo2, vec_transfer2 = sp.lsim(H_new2, vi, t_vo2)

#Plotting vo(t) for the high pass filter

plt.figure(5)
plt.plot(t_vo2, vo2)
plt.title('Plot of output voltage of the circuit calculated from
the transfer function \n for a given input to a high pass filter')
plt.xlabel('t')
plt.ylabel('vo(t)')
plt.show()
```

In the above code, I have obtained the output voltage $v_0(t)$ using 'sp.lsim' from the signal toolbox. Following is the plot I have obtained for the output voltage of the high pass filter,







As we see from the above graph, the time period is very low,
T $= 10^{-6}$ seconds

f = $\frac{1}{T}$ = 1 MHz
The graph looks like a cosine wave!

Even when our input had two types of sinusoids, the cosine sinusoid succeeded to pass through the filter as we have got the output frequency as 1 MHz which belongs to the cosine sinusoid. So, as this frequency has been able to pass, we are sure that the circuit is a high pass filter.

2.4 Obtaining the response of the circuit to a damped sinusoid :

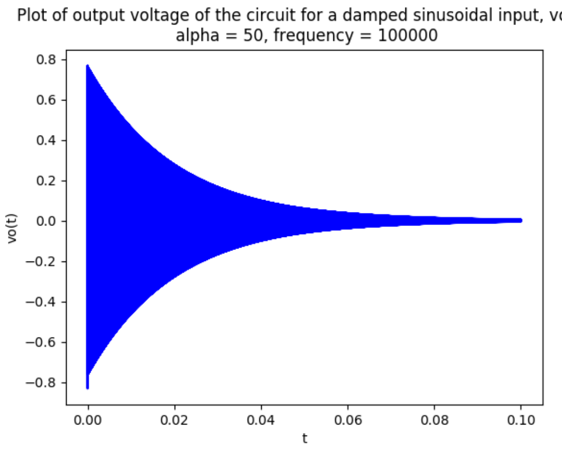
In this question, I have assumed a damped sinusoidal input as,
$v_i = e^{-50t}\sin(200000\pi t)$
$f = \varpi_0/2\pi = 100000$

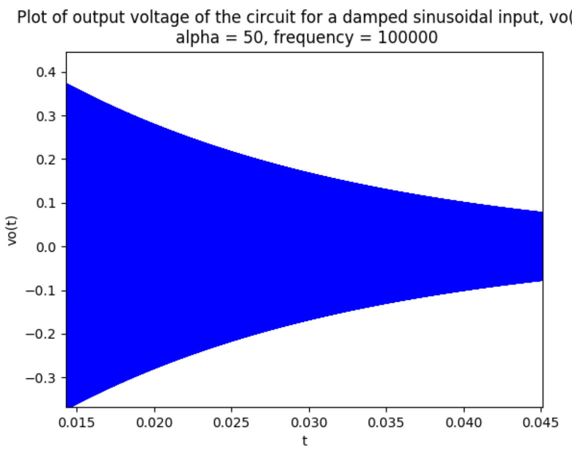
```
#Obtaining the repsonse of the circuit for a damped sinusoidal input

t_vo3 = numpy.linspace(0,0.1,1e5)
vi2 = numpy.exp(-50*t_vo3)*(numpy.sin(200000*(math.pi)*t_vo3))
t_transfer3, vo3, vec_transfer3 = sp.lsim(H_2, vi2, t_vo3)
```

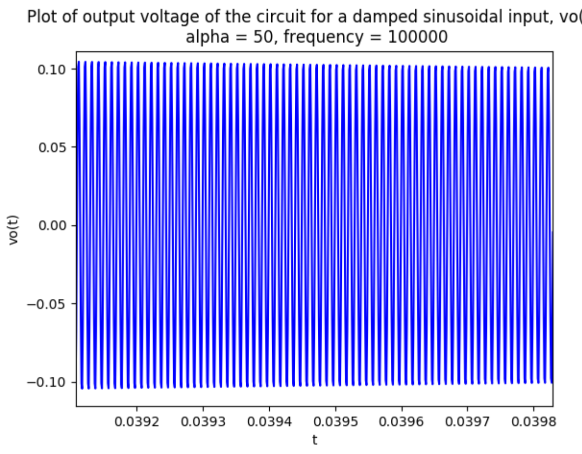
Following is the way I plotted,

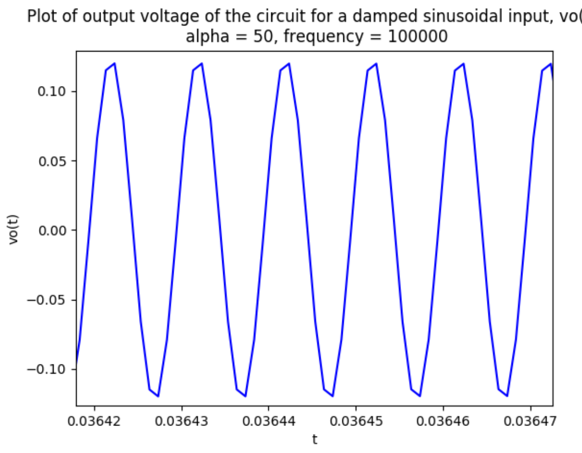
```
#Plotting output of the damped input

plt.figure(6)
plt.plot(t_vo3, vo3, 'b')
plt.title('Plot of output voltage of the circuit for
a damped sinusoidal input, vo(t) \n alpha = 50, frequency = 100000')
plt.xlabel('t')
plt.ylabel('vo(t)')
plt.show()
```

After giving this as input, we get the response of the circuit as,









In above graphs, I have shown the various outputs graphs and their zoomed versions. As we see for a high frequency and huge damping, as the input amplitude decreases rapidly, the output amplitude also decreases but the frequency is still the same as, $\varpi_0 = 20000\pi$ rad/sec.

Now, I have assumed another damped sinusoidal input as,

$v_i = e^{-50t}\sin(500\pi t)$

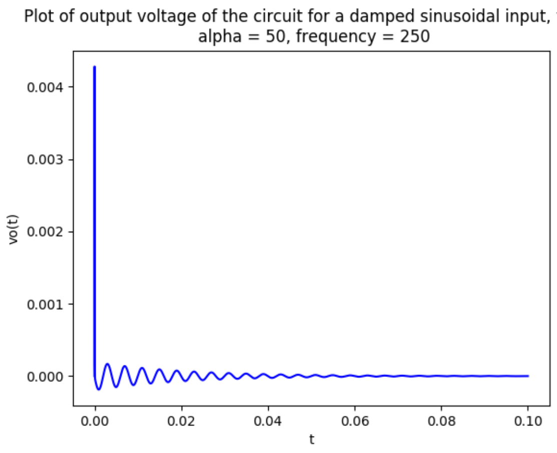
$f = \varpi_0/2\pi = 250$

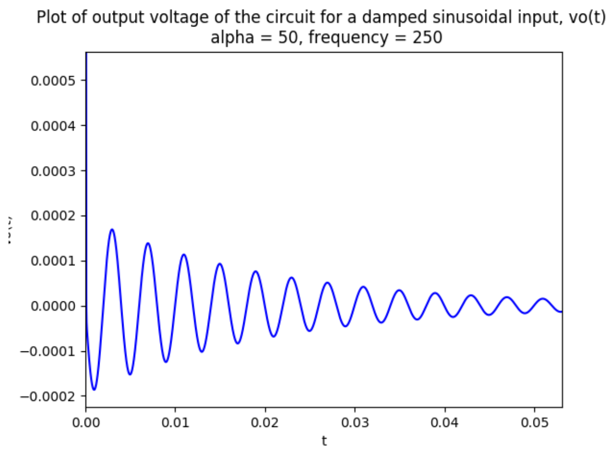
```
#Obtaining the repsonse of the circuit for a damped sinusoidal input

t_vo3 = numpy.linspace(0,0.1,1e5)
vi2 = numpy.exp(-50*t_vo3)*(numpy.sin(500*(math.pi)*t_vo3))
t_transfer3, vo3, vec_transfer3 = sp.lsim(H_new2, vi2, t_vo3)
```

Following is the way I plotted,

```
#Plotting output of the damped input

plt.figure(4)
plt.plot(t_vo3, vo3, 'b')
plt.title('Plot of output voltage of the circuit for
a damped sinusoidal input, vo(t) \n alpha = 50, frequency = 250')
plt.xlabel('t')
plt.ylabel('vo(t)')
plt.show()
```

After giving this as input, we get the response of the circuit as,





In above graphs, I have shown the output graph and its zoomed version. As we see for a low frequency and huge damping, as the input amplitude decreases rapidly, the output amplitude is almost becoming zero but the frequency is still the same as, $\varpi_0 = 250\pi$ rad/sec.

2.5 Obtaining the response of the circuit for a unit step function :

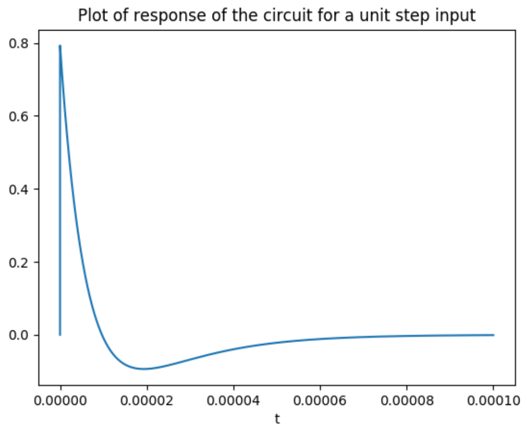
In this question we have input as a unit step function whose Laplace Transform can be written as follows,

$V_i(s) = \frac{1}{s}$

```
#Obtaining the repsonse of the circuit for a unit step response

Vi_damp_num = numpy.poly1d([1])
Vi_damp_denom = numpy.poly1d([1,0])
Vi_damp = sp.lti(Vi_damp_num, Vi_damp_denom)
t_vo4, vi3 = sp.impulse(Vi_damp, None, numpy.linspace(0,1e-4,1e5))
vi3[0] = 0
t_transfer4, vo4, vec_transfer4 = sp.lsim(H_2, vi3, t_vo4)
```

Above is the code for obtaining the output in time domain. Below is the way I have plotted the response.

```
#Plotting output of the unit step input

plt.figure(5)
plt.plot(t_vo4, vo4)
plt.title('Plot of response of the circuit for a unit step input')
plt.xlabel('t')
plt.ylabel('vo(t)')
plt.show()
```



As we see from the above graph, the peculiar shape of the graph with discontinuity is due to the fact that there are a lot of frequencies present in the unit function. Due to this, what happens is that initially the higher frequencies will be allowed to pass through the high pass filter (lower frequencies removed) resulting in the peak as shown in the beginning and then when the higher frequencies are gone (lower frequencies are left), the graph shows lower values of output voltage due to removal of lower frequenices by the high pass filter.

In the graph, we see a depression as well. That depression happens due to the quality factor being less than 1 (we get it from the transfer function).