EE2703 Applied Programming Lab - Assignment No 7

Name: Atishay Ganesh
Roll Number: EE17B155

March 17, 2019

1 Abstract

The goal of this assignment is the following.

- To analyze Filters using Laplace Transform.
- To see how python can be used for symbolic Algebra.
- To plot graphs to understand high pass and low pass analog filters.

2 Assignment

2.1 Setting up the variables

Importing the standard libraries

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
import scipy.signal as sp
from sympy import *
```

We initialize sympy and define a plotting function to help simplify the code.

```
init_printing()
s = symbols('s')
plt.rcParams.update({'mathtext.default':  'regular' })

def plotter(fig_no,arg1,arg2,label_x,label_y,type=plt.plot,arg3='b-',title="",cmap =
    plt.figure(fig_no)
    plt.grid(True)
    if type==plt.contourf:
        type(arg1,arg2,arg3,cmap=cmap)
```

```
        plt.colorbar()
    else:
        type(arg1,arg2,arg3)
    plt.xlabel(label_x,size =19)
    plt.ylabel(label_y,size =19)
    plt.title(title)
```

We also define a function to make bode plots.

```
def bodeplot(figure,H):
    plt.figure(figure)
    w,s,phi = H.bode()
    plt.subplot(2,1,1)
    plt.semilogx(w,s)
    plt.grid(True)
    plt.ylabel(r'$|H(j\omega)|$',size =17)
    plt.subplot(2,1,2)
    plt.semilogx(w,phi)
    plt.grid(True)
    plt.xlabel(r'$\omega$',size=17)
    plt.ylabel(r'$\angle(H(j\omega))$',size =17)
```

2.2 Defining the filters

We write the matrix equation of the given two systems and use sympy to symbolically solve for the transfer functions.

```
def lowpass(R1,R2,C1,C2,G,Vi=1):
    '''Active 2nd order low pass butterworth filter using opamp'''
    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)


def highpass(R1,R2,C1,C2,G,Vi = 1):
    '''Active 2nd order high pass filter using opamp'''
    A = Matrix([[0,0,1,-1/G],[-1/(1+1/(s*R2*C2)),1,0,0],
                [0,-G,G,1],[-s*C1-s*C2-1/R1,s*C2,0,1/R1]])
    b = Matrix([0,0,0,Vi*s*C1])
    V = A.inv()*b
    return(A,b,V)
```

2.3 Converting to scipy signals form

We convert the transfer functions thus got to scipy signals form, which will help us in further calculations.

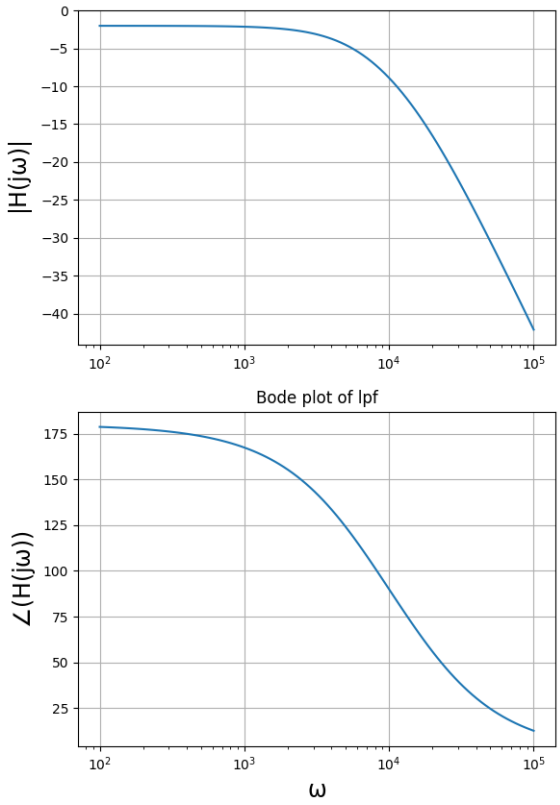
```
def simplifytoH(V):
    '''extracts H from the result of the matrix inversion'''
    Vo = V[3]
    #since the third element in the V column is actually the output voltage
    Vo = expand(simplify(Vo))#converting the simple rational form
    H = sympytoscipytf(Vo)
    return H

def sympytoscipytf(Vo):
    '''converts sympy tf to scipy tf'''
    v1 = fraction(Vo)#converting to numerator and denominator
    n,d = Poly(v1[0],s),poly(v1[1],s)
    numer,denom = n.all_coeffs(), d.all_coeffs()#extracting coeffs
    numer,denom = [float(f) for f in numer], [float(f) for f in denom]
    H = sp.lti(numer,denom)#converting to scipy lti form
    return H

A,b,V = lowpass(10000,10000,1e-8,1e-8,1.586,1)
'''We use 10 nF as it gives us nice figures'''
Hl = simplifytoH(V)
t = np.arange(0,1e-2,1e-7)
A,b,V = highpass(1e4,1e4,1e-9,1e-9,1.586)
'''We use 1 nF as it gives us nice values'''
Hh = simplifytoH(V)
```

2.4 Bode plots

We draw bode plots of the filters, to verify indeed that they are low and high pass filters respectively.

```
#Drawing Bode Plots of the filters
bodeplot(0,Hl)
plt.title("Bode plot of lpf")
bodeplot(1,Hh)
plt.title("Bode plot of hpf")
plt.show()
```

We observe that the impulse response of the first filter drops offs towards higher frequencies and that of the second filter increases towards higher frequencies, hence they are low and high pass filters respectively.



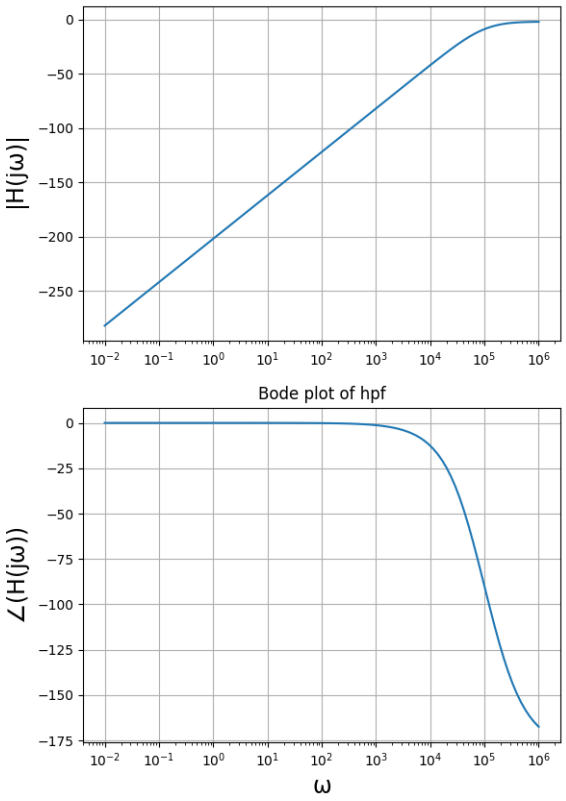
2.5 Step Response

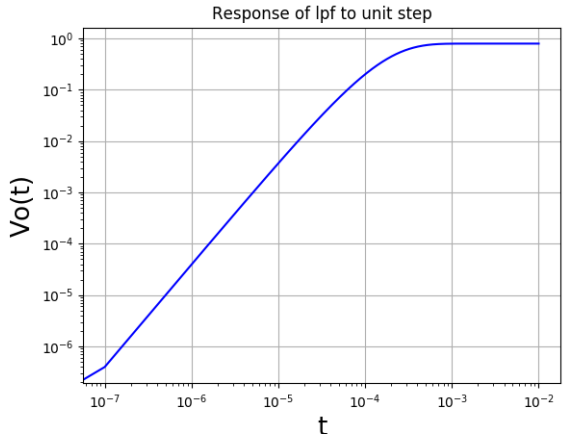
We plot the step response of both the filters to see what it shows.

```
#Plotting step response of Analog filters
t,vtd,svec = sp.lsim(Hl,np.ones_like(t),t)
plotter(0,t,abs(vtd),r"t",r'Vo(t)',
       plt.loglog,title ='Response of lpf to unit step')

t,vtd,svec = sp.lsim(Hh,np.ones_like(t),t)
plotter(1,t,(vtd),r"t",r'Vo(t)',
       plt.semilogx,title ='Response of hpf to unit step')
plt.show()
```

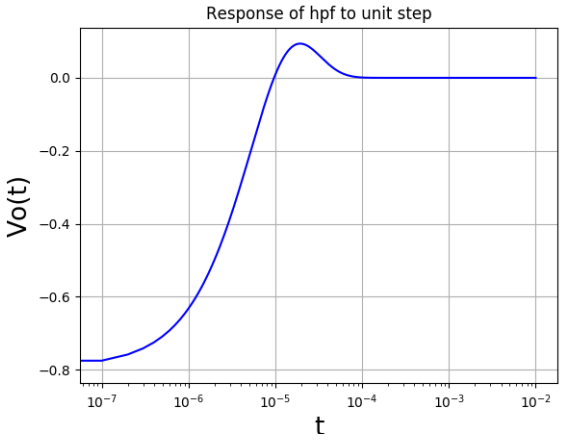
Using the initial and final value theorem, we understand the following.

- The low pass filter's step response has a final value of about 0.79, indicating, it allows DC almost intact.





- The high pass filter's step response has a final value of 0, which shows it does not allow DC to pass through



- The low pass filter's step response has a initial value of 0, indicating, it does not allow high frequency AC to pass.
- The high pass filter's step response has a initial value of magnitude 0.79, which shows it does allow high frequency AC to pass.
- No overshoot in lpf case as it is a butterworth filter
- Overshoot is observed in hpf case.

2.6 Response to sum of two sinusoids

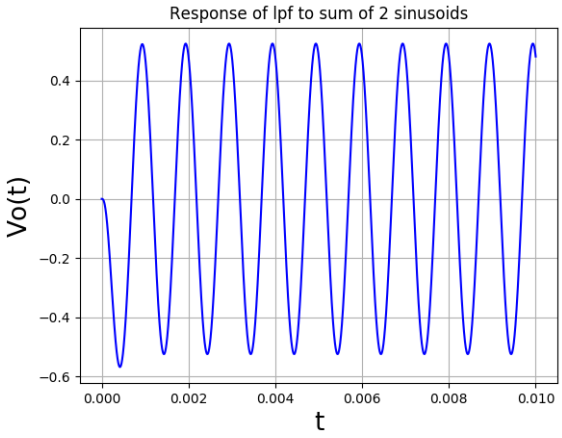
We plot the response of the filters to a sum of two input sinusoids.

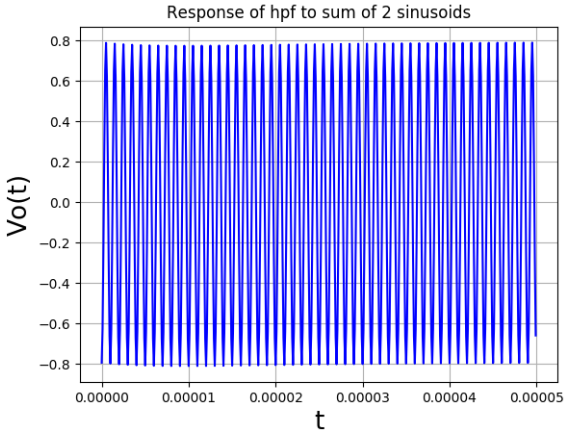
```
#Response of high and low pass filter to sum of sinusoids.
#Frequency of one is in the pass band of lpf and stop band of hpf.
#Frequency of other is in the pass band of hpf and stop band of hpf.
t,vtd,svec = sp.lsim(Hl,(np.sin(2000*np.pi*t)+np.cos(2*10**6*np.pi*t)),t)
plotter(0,t,vtd,r"t",r'Vo(t)',
       plt.plot,title ='Response of lpf to sum of 2 sinusoids')
t,vtd,svec = sp.lsim(Hh,(np.sin(2000*np.pi*t)+np.cos(2*10**6*np.pi*t)),t)
plotter(1,t[0:500],vtd[0:500],r"t",r'Vo(t)',
       plt.plot,title ='Response of hpf to sum of 2 sinusoids')
plt.show()
```

As expected we observe the lpf only allows the low frequency sinusoid to pass and the hpf only allows the high frequency sinusoid to pass. Observe the differing time scales in both the images.

2.7 Response to damped sinusoids

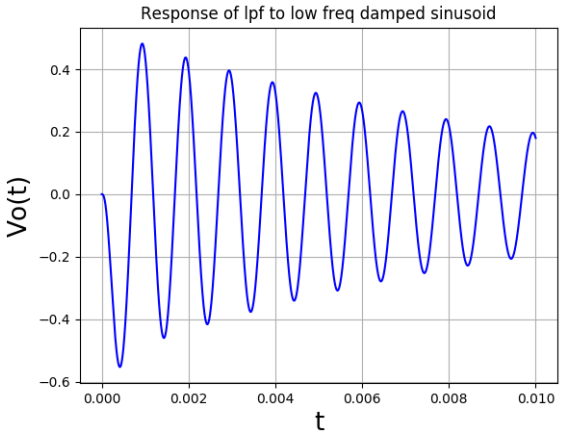
We input damped sinusoidal inputs of low and high frequencies to observe how the filters will respond.






```
#Response of of lpf and hpf to damped sinusoids
#Damping factor is chosen such that
#damping is visible in the time frame.
#Both low and high freq inputs are given.
damping_factor = 100
t = np.arange(0,1e-2,1e-7)
t,vtd,svec = sp.lsim(Hh,
    np.exp(-damping_factor*t)*(np.cos(2*10**6*np.pi*t)),t)
plotter(0,t,vtd,r"t",r'Vo(t)',
       plt.plot,title ='Response of hpf to high freq damped sinusoid')


t = np.arange(0,1e-2,1e-7)
t,vtd,svec = sp.lsim(Hl,
    np.exp(-damping_factor*t)*(np.cos(2*10**6*np.pi*t)),t)
plotter(1,t,vtd,r"t",r'Vo(t)',
```

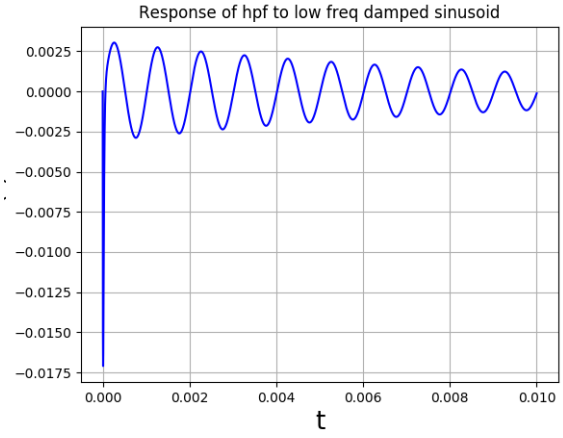
```
       plt.plot,title ='Response of lpf to high freq damped sinusoid')


t = np.arange(0,1e-2,1e-7)
t,vtd,svec = sp.lsim(Hl,
    np.exp(-damping_factor*t)*(np.sin(2000*np.pi*t)),t)
plotter(2,t,vtd,r"t",r'Vo(t)',
       plt.plot,title ='Response of lpf to low freq damped sinusoid')


t = np.arange(0,1e-2,1e-7)
t,vtd,svec = sp.lsim(Hh,
    np.exp(-damping_factor*t)*(np.sin(2000*np.pi*t)),t)
plotter(3,t,vtd,r"t",r'Vo(t)',
       plt.plot,title ='Response of hpf to low freq damped sinusoid')
plt.show()
```

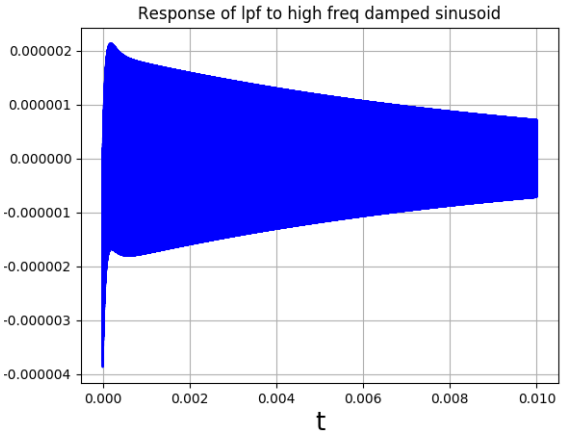


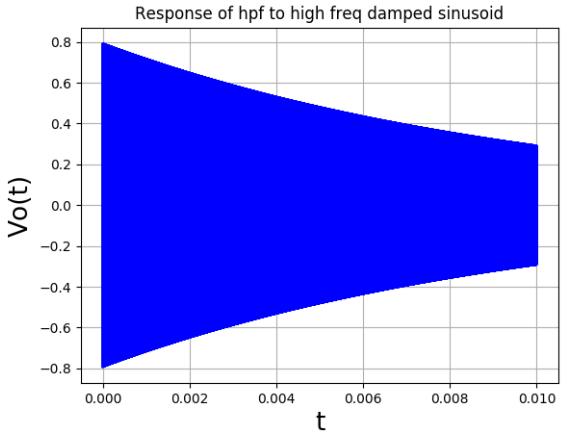
The low frequency sinusoid when passed through the high pass filter shows much quicker decay than when passed through the low pass filter. This is expected as the hpf acts in a damping fashion to the low frequency sinusoid. The opposite is observed for the high frequency sinusoid.

3 Conclusions

- We analyzed analog filters in the Laplace domain using sympy.
- We used the scipy signals to analyse the response to various input signals.
- We plotted graphs to understand the above.


Response of hpf to low freq damped sinusoid
0.0025
0.0000
−0.0025
−0.0050
−0.0075
−0.0100
−0.0125
−0.0150
−0.0175
0.000
0.002
0.004
0.006
0.008
0.010
t


Response of lpf to high freq damped sinusoid
0.000002
0.000001
0.000000
-0.000001
-0.000002
-0.000003
-0.000004
0.000
0.002
0.004
0.006
0.008
0.010
t


Response of hpf to high freq damped sinusoid
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
Vo(t)
0.000
0.002
0.004
0.006
0.008
0.010
t