EE2703: Applied Programming Lab

Assignment # 7

Sourav Sahoo, EE17B040

March 16, 2019

Contents

1 Introduction

In signal processing, a filter is a device or process that removes some unwanted components or features from a signal. Filtering is a class of signal processing, the defining feature of filters being the complete or partial suppression of some aspect of the signal. Most often, this means removing some frequencies or frequency bands. Filters are widely used in electronics and telecommunication, in radio, television, audio recording, radar, control systems, music synthesis, image processing, and computer graphics. [1]
Analogue filters are a basic building block of signal processing much used in electronics. Amongst their many applications are the separation of an audio signal before application to bass, mid-range and tweeter loudspeakers; the combining and later separation of multiple telephone conversations onto a single channel; the selection of a chosen radio station in a radio receiver and rejection of others. [2] In this assignment we use SymPy to analyse Low Pass Filters and High Pass Filters.

2 Problem 1

Import the following libraries.

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

First the LPF is designed.

```
s = sp.symbols('s')
def LPF(R1,R2,C1,C2,G,Vi):
    A = sp.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 = sp.Matrix([0,0,0,Vi/R1])
    V = A.inv()*b
    return (A,b,V)
```

Next, the system impulse and step response is calculated using `scipy.signal.impulse` and plotted.

```
A,b,V = LPF(10000,10000,1e-9,1e-9,1.586,1)
A1,b1,V1 = LPF(10000,10000,1e-9,1e-9,1.586,1/s)
Vo = V[3]
Vp = V1[3]
w = np.logspace(-1,8,100)
ss = 1j*w
hf = sp.lambdify(s,Vo,'numpy')
hf1 = sp.lambdify(s,Vp,'numpy')
v = hf(ss)
v1 = hf1(ss)

plt.figure(figsize=(16,8))
plt.grid()
plt.title(r'$Impulse\;and\;step\;response\;of\;LPF$')
plt.xlabel(r'$\omega\rightarrow$')
plt.ylabel(r'$|H(j\omega)|\rightarrow$')
plt.loglog(w,abs(v))
plt.loglog(w,abs(v1))
plt.legend(['impulse response','step impulse'])
plt.show()
```

In the time domain ,the response can be be found out by executing the following code.

[1] https://en.wikipedia.org/wiki/Filter_(signal_processing)
[2] https://en.wikipedia.org/wiki/Analogue_filter

```
H = sig.lti([-0.0001586],[2e-14,4.414e-9,0.0002])
t = np.linspace(0,0.001,2e5)
u = (t>0)
t,y,_ = sig.lsim(H,u,t)
plt.figure(figsize=(16,8))
plt.grid()
plt.title(r'$Input\;and\;output\;to\;the\;filter$')
plt.xlabel(r'$t\rightarrow$')
plt.ylabel(r'$y\rightarrow$')
plt.plot(t,u)
plt.plot(t,y)
plt.legend(['input','output'])
plt.show()
```

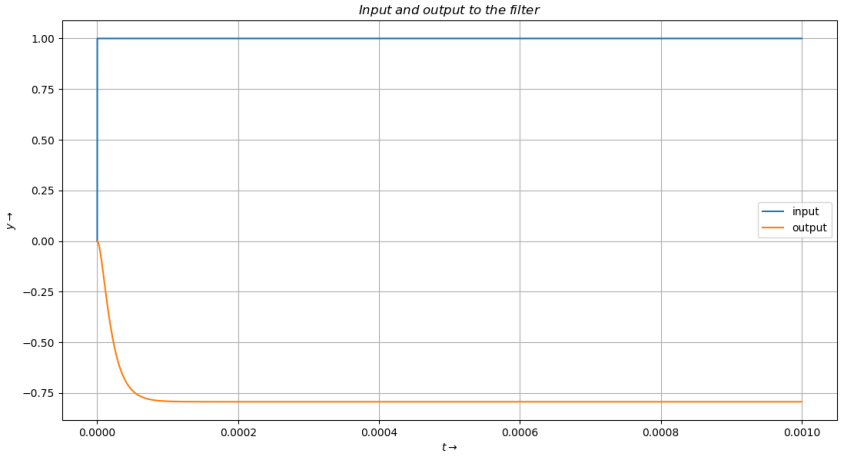

Figure 1: Input and output to the filter

3 Problem 2

We have

$$v_i(t) = (sin(2000\pi t) + cos(2 \times 10^6 \pi t))u(t)$$

The low and high frequency components of the input atre plotted separately.

```
t = np.linspace(0,0.001,2e5)
u1 = np.sin(2000*np.pi*t) # t = 1ms
u2 = np.cos(200000*np.pi*t) # t = 1us
Vo = sp.simplify(Vo)
n,d = sp.fraction(Vo)
print(n,d)
plt.figure(figsize=(16,8))
plt.grid()
plt.title(r'$Components\;of\;the\;input\;to\;the\;filter$')
plt.xlabel(r'$t\rightarrow$')
plt.ylabel(r'$y\rightarrow$')
plt.plot(t,u1)
plt.plot(t,u2)
plt.legend(['low frequency input','high frequency input'])
plt.show()
```

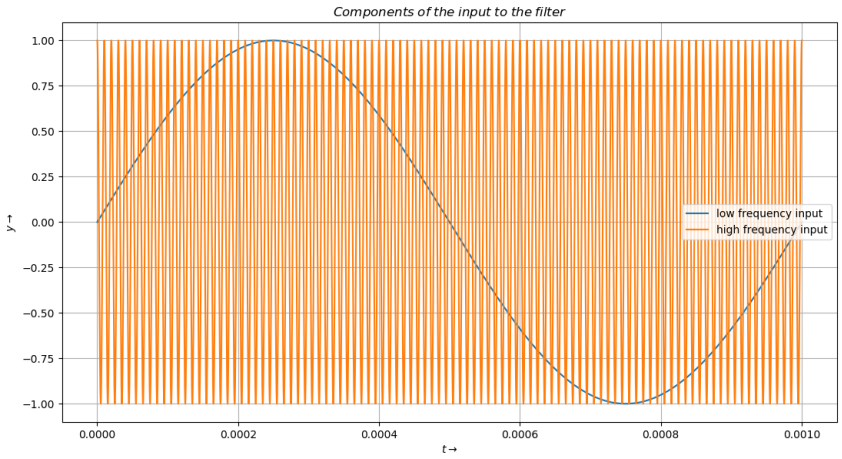

Figure 2: Components of the input to the LPF

Then the output is calculated using `scipy.signal.lsim` function and plotted.

```
H = sig.lti([-0.0001586],[2e-14,4.414e-9,0.0002])
t,y,_ = sig.lsim(H,u1+u2,t)
plt.figure(figsize=(16,8))
plt.grid()
plt.title(r'$Output\;to\;the\;filter$')
plt.xlabel(r'$t\rightarrow$')
plt.ylabel(r'$y\rightarrow$')
plt.plot(t,y)
plt.show()
```

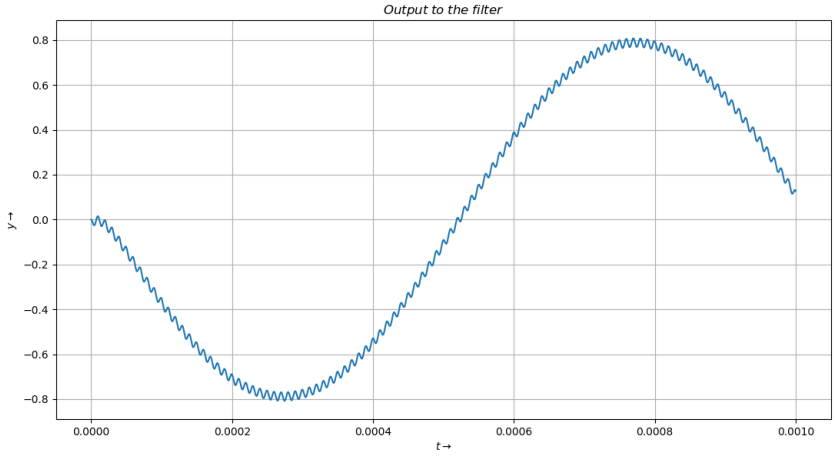

Figure 3: Output of the LPF

4 Problem 3

The HPF is designed and the transfer function is plotted.

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

A,b,V = HPF(10000,10000,1e-9,1e-9,1.586,1)
Vo = V[3]
Vo = sp.simplify(Vo)
n,d = sp.fraction(Vo)
print(n,d)
w = np.logspace(-1,8,1000)
ss = 1j*w
hf = sp.lambdify(s,Vo,'numpy')
v = hf(ss)

plt.figure(figsize=(16,8))
plt.semilogx(w,abs(v))
plt.grid()
plt.title(r'$Impulse\;response\;of\;HPF$')
plt.xlabel(r'$\omega\rightarrow$')
plt.ylabel(r'$|H(j\omega)|\rightarrow$')
plt.show()
```

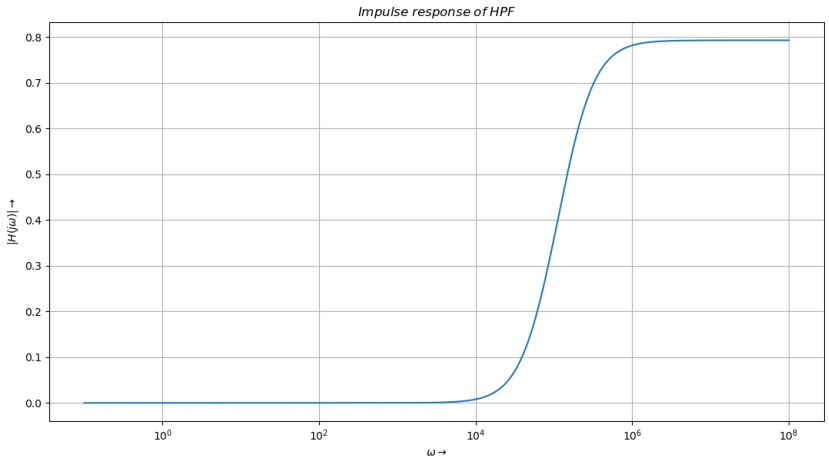

Figure 4: Impulse response of the HPF

5 Problem 4

Let the input to the HPF be:

$$v_i(t) = (sin(2000t) + sin(2 \times 10^6))e^{-50000t}u(t)$$

The low and high frequency components of the input atre plotted separately.

```
t = np.linspace(0,0.0001,2e5)
u1 = np.sin(2000*np.pi*t)*np.exp(-50000*t)
u2 = np.sin(2000000*np.pi*t)*np.exp(-50000*t)
```

```
plt.figure(figsize=(16,8))
plt.grid()
plt.title(r'$Components\;of\;the\;input\;to\;the\;filter$')
plt.xlabel(r'$t\rightarrow$')
plt.ylabel(r'$y\rightarrow$')
plt.plot(t,u1)
plt.plot(t,u2)
plt.legend(['low frequency input','high frequency input'])
plt.show()
```

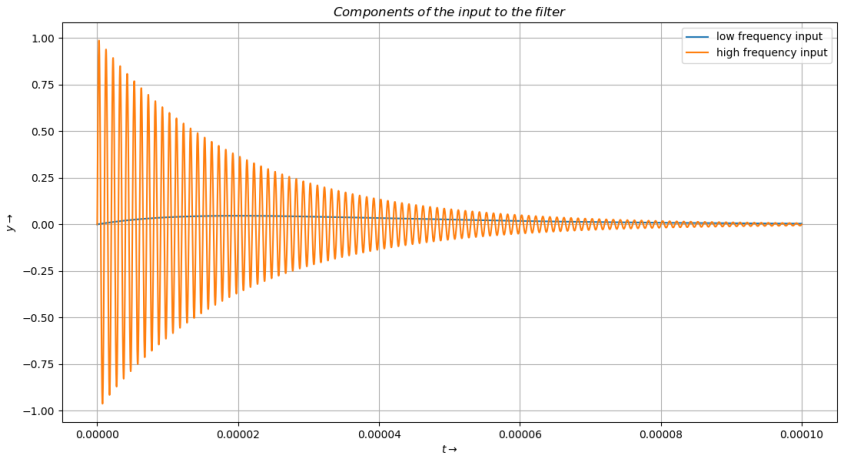

Figure 5: Components of the input to the HPF

Then the output is calculated using `scipy.signal.lsim` function and plotted.

```
H = sig.lti([-1.586e-9,0,0],[2e-9,4.414e-4,20.0])
t,y,_ = sig.lsim(H,u1+u2,t)
plt.figure(figsize=(16,8))
plt.grid()
plt.title(r'$Output\;to\;the\;filter$')
plt.xlabel(r'$t\rightarrow$')
plt.ylabel(r'$y\rightarrow$')
plt.plot(t,y)
plt.show()
```

6 Problem 5

The input to the HPF is now a unit step function. The input and output of the HPF is calculated using `scipy.signal.lsim` and plotted.

```
H = sig.lti([-1.586e-9,0,0],[2e-9,4.414e-4,20.0])
u = (t>0)
t,y,_ = sig.lsim(H,u,t)
plt.figure(figsize=(16,8))
plt.grid()
plt.title(r'$Input\;and\;output\;to\;the\;filter$')
plt.xlabel(r'$t\rightarrow$')
plt.ylabel(r'$y\rightarrow$')
plt.plot(t,u)
plt.plot(t,y)
```

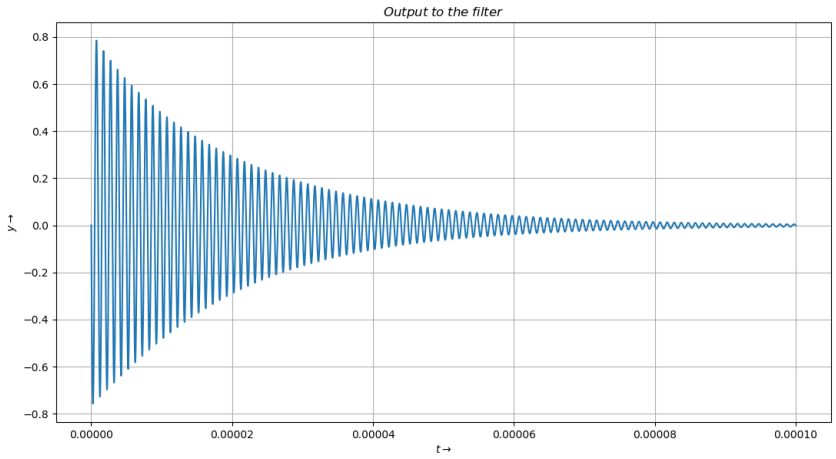

Figure 6: Output of the HPF

```
plt.legend(['input','output'])
plt.show()
```

6.1 Inline Question

The output of the HPF is as expected. Initially,the input contains some high frequency components which can be calculated from the Fourier Transform of the input.So, we see a spike at the beginning.The unit step function is as good as a DC input at later times ,so the output falls down to zero.

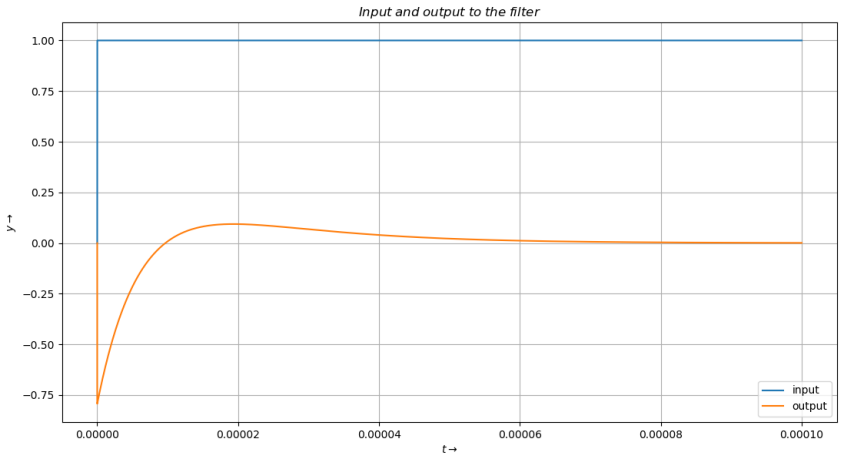

Figure 7: Input and output of the HPF

7 Conclusion

In this assignment we analyzed two important types of filters namely Low Pass Filter and High Pass Filters using SymPy by providing different inputs and calculating the corresponding outputs.