EE2703 : Applied Programming Lab
Assignment 8
Circuit Analysis Using Sympy

Dhruv Goyal, EE19B077

April 30, 2021

1. Abstract

This week's assignment involves the analysis of filters using laplace transforms. Python's symbolic solving library, sympy is a tool we use in the process to handle our requirements in solving Modified Nodal Analysis equations. Coupled with scipy's signal module, we are able to analyse both High pass and low pass analog filters, both second order, realised using a single op amp.

2. The Low Pass Filter

The low pass filter that we use gives the following matrix equation after simplification of the modified nodal equations.

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

The python code snippet that declares the low pass function and solves the matrix equation to get the V matrix is as shown below:

```
def lowpass(R1,R2,C1,C2,G,Vi):          # Lowpass filter takes circuit paramete
s=sym.symbols('s')
A=sym.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=sym.Matrix([0,0,0,-Vi/R1])
V=A.inv()*b
return A,b,V
```

The plot for the magnitude of the transfer function (magnitude bode plot) is as shown below:

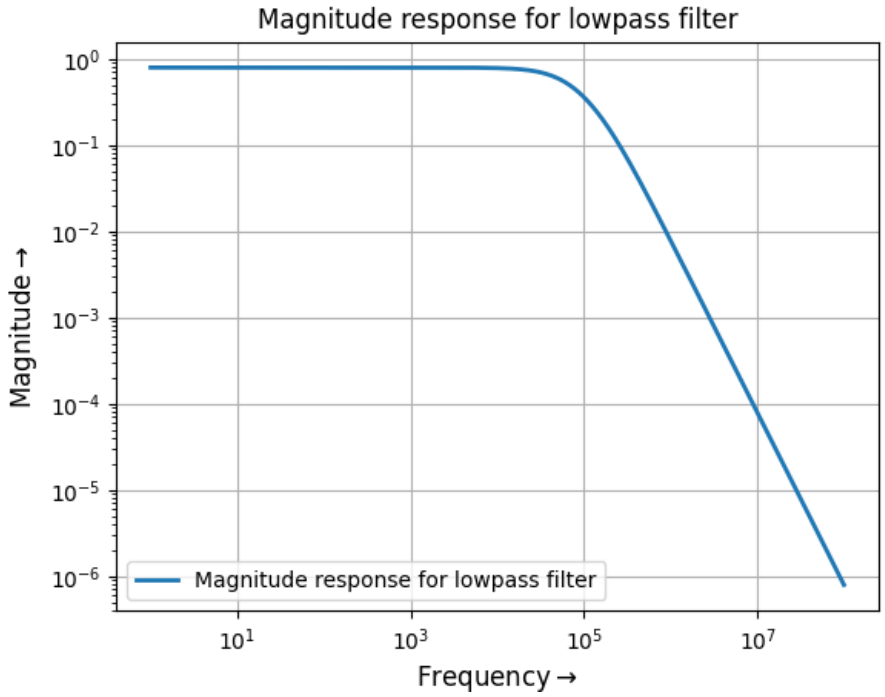

Figure 1: Magnitude bode plot of the low pass filter

3. High Pass Filter

The high pass filter we use gives the following matrix equations after simplification of the modified nodal equations

$$\begin{pmatrix} 0 & 0 & 1 & -\frac{1}{G} \\ -\frac{-sR_2C_2}{1+sR_2C_2} & 1 & 0 & 0 \\ 0 & -G & G & 1 \\ -1-(sR_1C_1)-(sR_2C_2)) & sC_2R_1 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_p \\ V_m \\ V_o \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -V_i(s)sR_1C_1 \end{pmatrix}$$

The python code snippet that declares the high pass function and solves the matrix equation to get the V matrix is as shown below:

```
def highpass(R1,R2,C1,C2,G,Vi):                    # Highpass Filter
    s=sym.symbols('s')
    A=sym.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=sym.Matrix([0,0,0,-Vi*s*C1])
    V = A.inv()*b
    return A,b,V
```

The plot for the magnitude of the transfer function (magnitude bode plot) is as shown below:

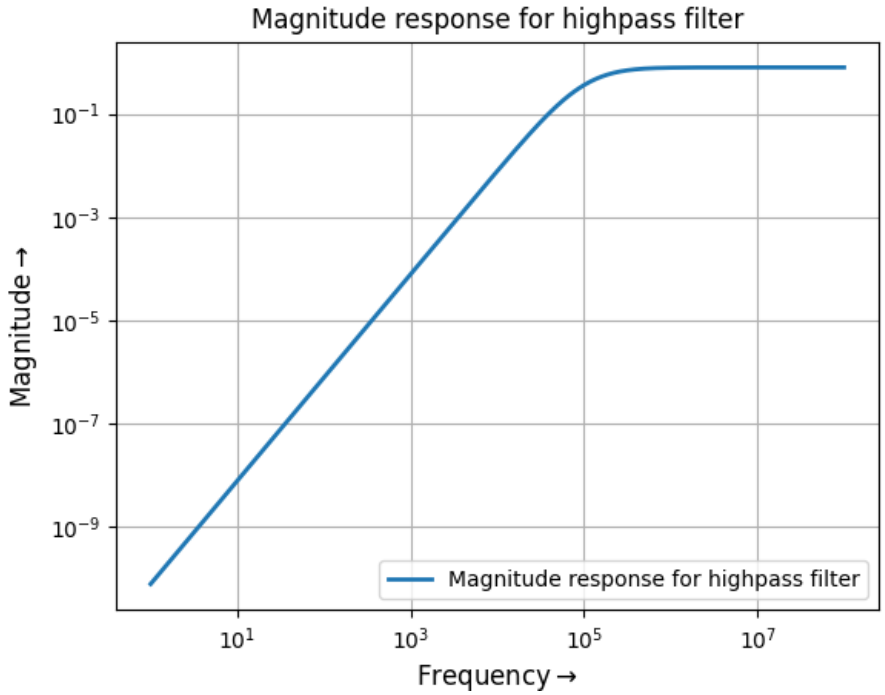

Figure 2: Magnitude bode plot of the high pass filter

4. Assignmemnt

1. Step Response of low pass filter

The sympy functions,that are expressed in terms of 's', must be converted to another form that is understood by sp.signal. This is done using the *sympyToH()* function.
The python code snippet is as shown:

```
def symToH(H):
    n,d = sym.simplify(H).as_numer_denom()
```

```
    n,d = sym.poly(n,s), sym.poly(d,s)
    num,den = n.all_coeffs(), d.all_coeffs()
    num,den = [float(f) for f in num], [float(f) for f in den]
    return num,den

def convolve(Vo,T,f):              #Function to perform convolution using sp.lsim
        num, den = symToH(Vo)
        H = sp.lti(num,den)
        t,y,svec = sp.lsim(H,f,T)
        return t,y
```

The plot for the step response of the low pass circuit is as shown:

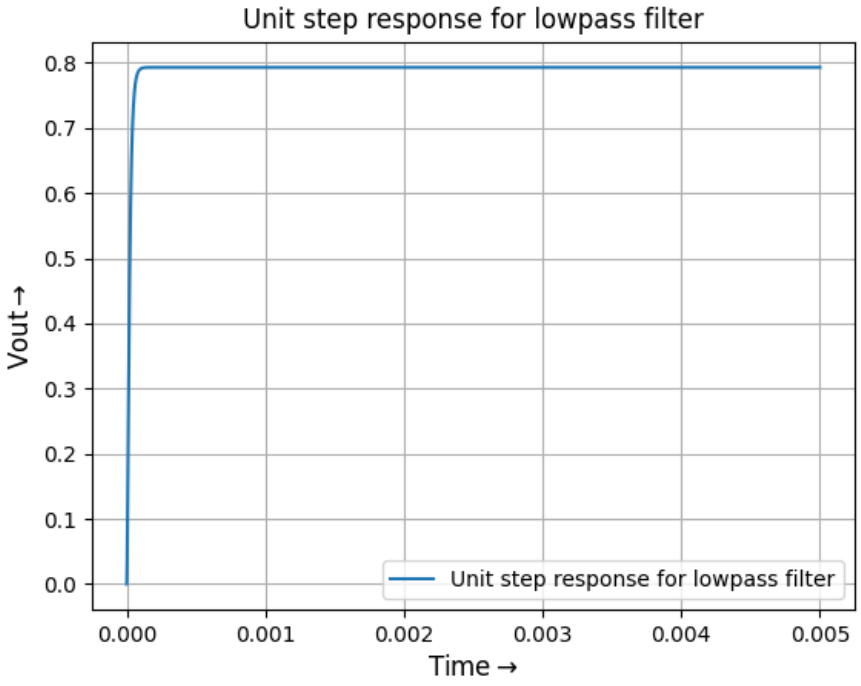

Figure 3: Step response of low pass filter.

2. Response to Sum Of Sinusoids

When the input is a sum of sinusoids like,

$$T = np.linspace(0, 0.001, 100000)$$
$$Sin = (np.sin(2000 * np.pi * T) + np.cos(2 * np.pi * T * 1e6))$$

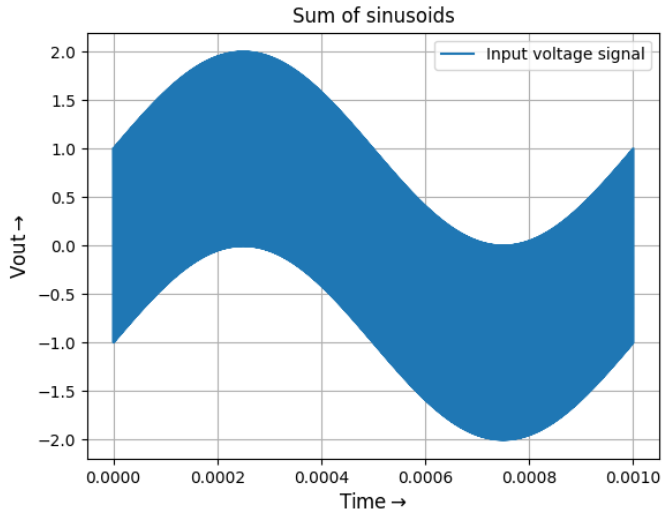

Figure 4: Output response to the sum of sinusoids.

Then the output response for the lowpass filter can easily be found by using convolve. The output response to the sum of sinusoids for the low pass filter is as shown:

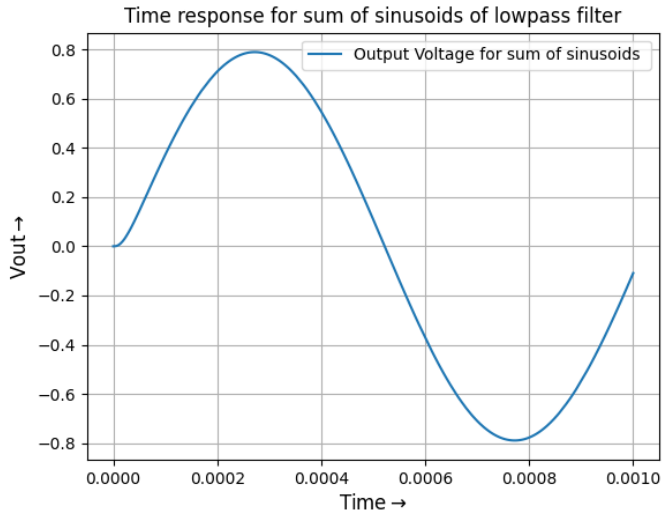

Figure 5: Output response to the sum of sinusoids.

The output response to the sum of sinusoids for the high pass filter is as shown:

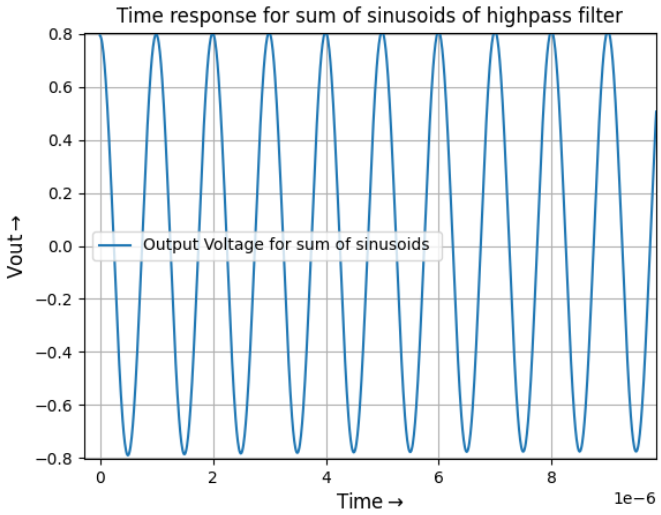

Figure 6: Output response to the sum of sinusoids.

3. Response to Damped Sinusoids

In this case we assign the input voltage as a damped sinusoid like,
Low frequency,

$$T1 = np.linspace(0, 0.5, 1000)$$
$$dSin2 = dampedsin(T1, 1e2, 50)$$

High frequency,

$$dSin = dampedsin(T, 1e7, 3000)$$

The high frequency and low frequency plots of the input damped sinusoids are as shown:

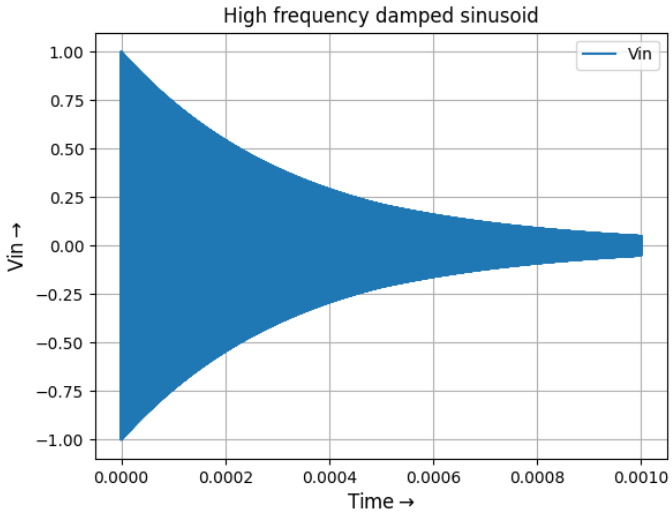

Figure 7: High frequency damped sinusoid

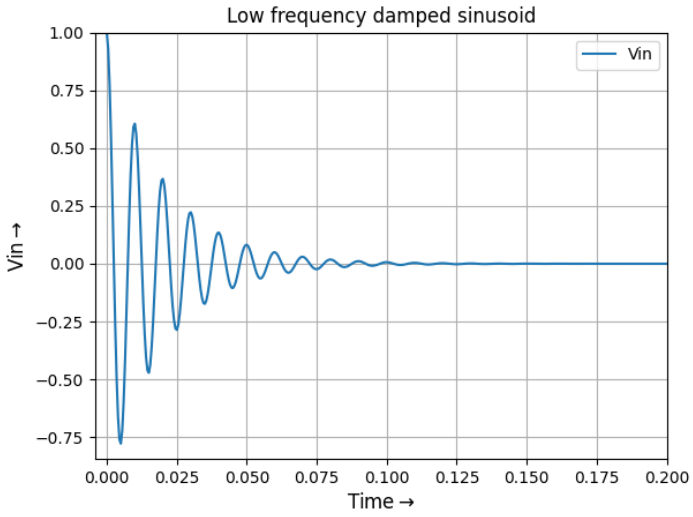

Figure 8: Low frequency damped sinusoid

The python code snippet to execute the above is as shown:

```
def damped_sin(t, freq, decay):
        return np.cos(2*np.pi*freq*t)*np.exp(-decay*t)

# Finding response to decaying sinusoid with high frequency
```

```
dSin=damped_sin(T,1e7,3000)

t_dsinl,y_dsinl=convolve(Vo_il,T,dSin)
t_dsinh,y_dsinh=convolve(Vo_ih,T,dSin)



# Finding response to decaying sinusoid with low frequency
T1 = np.linspace(0,0.5,1000)
dSin2=damped_sin(T1,1e2,50)

t_dsinl2,y_dsinl2=convolve(Vo_il,T1,dSin2)
t_dsinh2,y_dsinh2=convolve(Vo_ih,T1,dSin2)
```

The output responses for both cases in a high pass filter is as shown:

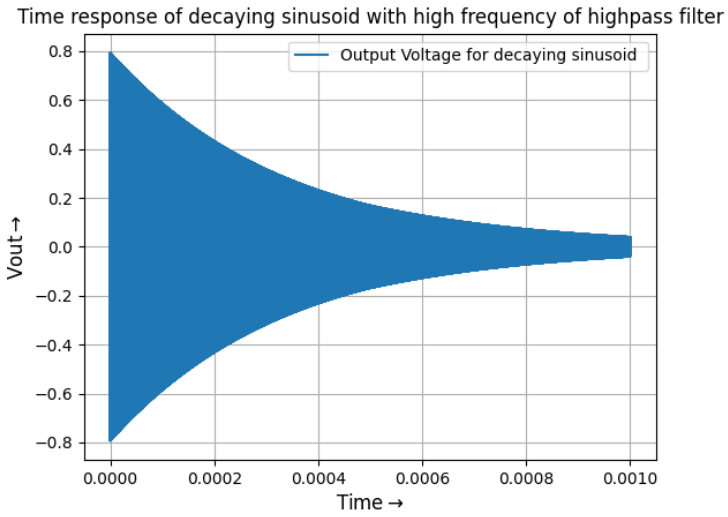

Figure 9: Output response of a high frequency damped sinusoid

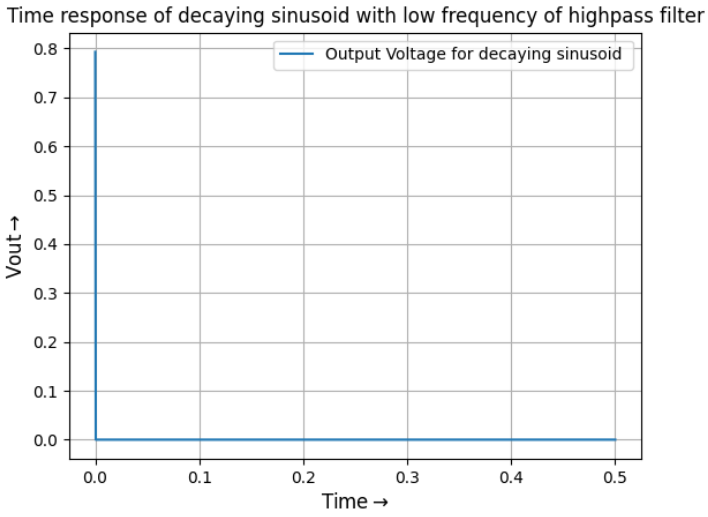

Figure 10: Output response of a low frequency damped sinusoid

The output responses for both cases in a low pass filter is as shown:

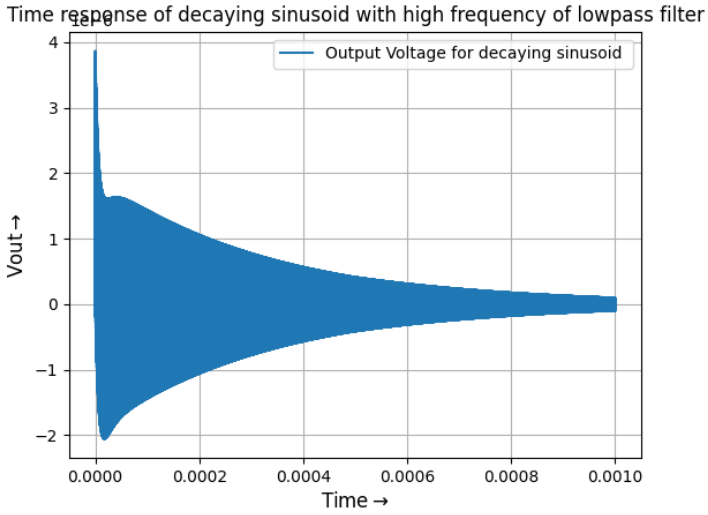

Figure 11: Output response of a high frequency damped sinusoid

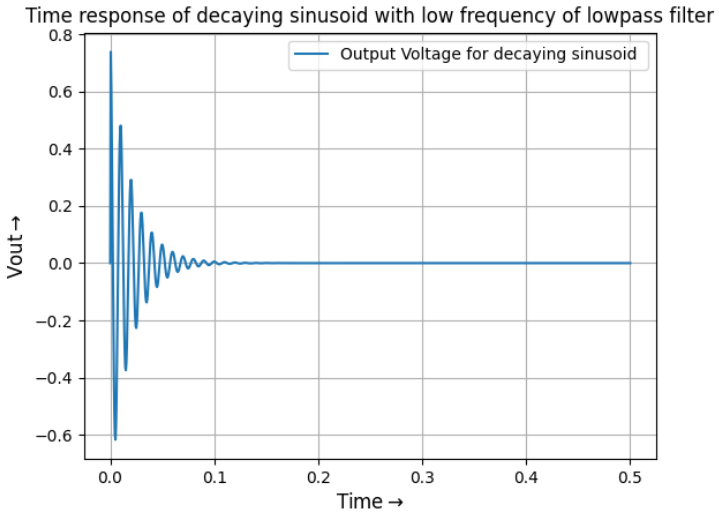

Figure 12: Output response of a low frequency damped sinusoid

4. Step Response of High Pass Filter

Since there is no other change at large time values outside the neighbourhood of 0, the Fourier transform of the unit step has values near 0 frequency, which the high pass filter attenuates. At t=0, the change is abrupt and hence a large value is seen around t=0 in the high pass filter step response.

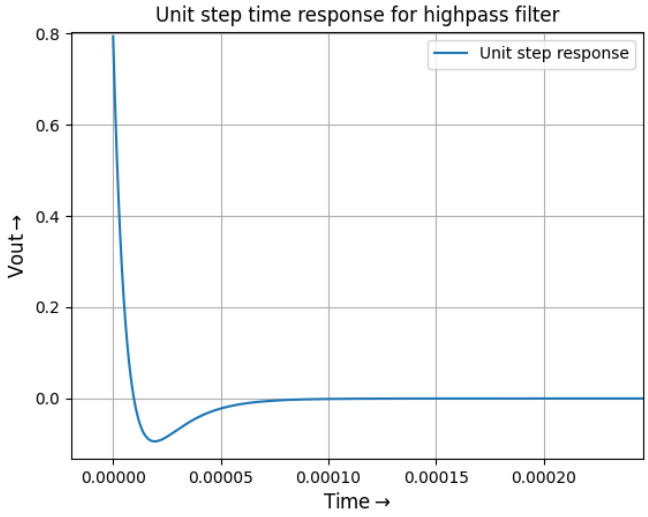

Figure 13: Step response for a high pass filter

Conclusion

In conclusion, the sympy module has allowed us to analyse quite complicated circuits by analytically solving their node equations. Their magnitude response was visualised.

The low pass filter allows sinusoids upto an angular frequency of 1e5 rad/s(for the values of components given) approximately. The high pass filter allows sinusoids beyond an angular frequency of 1e5 rad/s.

In the first question, the step response gets attenuated around t=0, as there is an abrupt jump in the input at t=0 (which corresponds to a high frequency). While in the long run the input behaves like DC and hence doesn't get attenuated in the long run.

In question 2 the high frequency component got filtered because it's frequency lies beyond the cutoff of the LPF.

In question 4, for the high frequency damped sinusoid, it passes almost unaffected through the HPF, while it gets attenuated when passed through the LPF. Similarly the low frequency damped sinusoid get's attenuated when it passes through the HPF and passes through almost unaffected through the LPF.

In question 5, for t ¿¿0 the step function behaves like a constant(DC), and hence gets attenuated by the high pass filter. At t=0, the change is abrupt and hence a large value is seen around t=0 in the high pass filter step response

We interpreted the solutions by plotting time domain responses using the signals toolbox. Thus, sympy combined with the scipy.signal module is a very useful toolbox for analyzing complicated systems like the active filters in this assignment.