EE2703 : Applied Programming Lab Assignment 7

Sasanapuri Uttam Raj
EE20B118

April 2022

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 Low Pass Filter

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

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

The following python code snippet declares the low pass function and solves the matrix equation to get the V matrix:

```
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
        return A,b,V
```

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

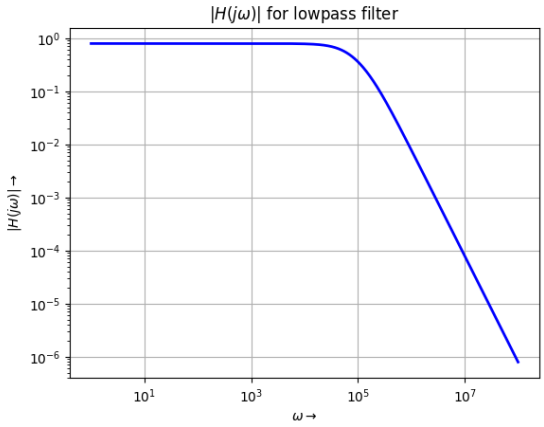

Figure 1: Magnitude bode plot of the low pass filter

3 High Pass Filter

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

$$\begin{bmatrix} 0 & 0 & 1 & -\frac{1}{G} \\ -\frac{-sR_3C_2}{1+sR_3C_2} & 1 & 0 & 0 \\ 0 & -G & G & 1 \\ -1-(sR_1C_1)-(sR_3C_2)) & sC_2R_1 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_1 \\ V_p \\ V_m \\ V_o \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -V_i(s)/R_1 \end{bmatrix}$$

The following python code snippet declares the high pass function and solves the matrix equation to get the V matrix:

```
def highpass(R1,R3,C1,C2,G,Vi):
    s = symbols("s")
    A = Matrix([[0,-1,0,1/G],
        [s*C2*R3/(s*C2*R3+1),0,-1,0],
        [0,G,-G,1],
        [-s*C2-1/R1-s*C1,0,s*C2,1/R1]])
    b = 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:

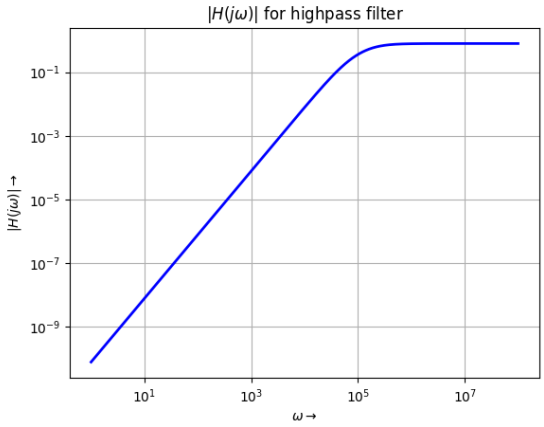

Figure 2: Magnitude bode plot of the high pass filter

4 Assignment Questions

4.1 Question 1

In order to find the step response of the low pass filter, we need to assign $Vi(s) = 1/s$. The python code below explains it:

```
A1,b1,V1 = lowpass(10000,10000,1e-9,1e-9,1.586,1/s)
Vo1 = V1[3]
H1 = sympytoscipy(Vo1)
t,y1 = sp.impulse(H1,None,linspace(0,5e-3,10000))
```

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

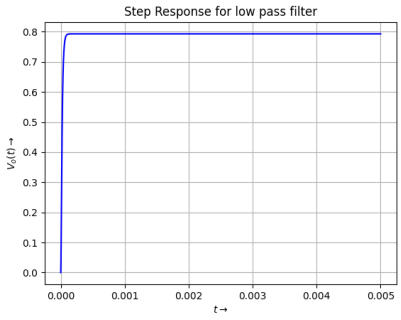

Figure 3: Step response of low pass filter

4.2 Question 2

Now the given in input $V_i(t)$ is:

$$V_i(t) = (\ \sin(2000\pi t) + \cos\big(2 * 10^6 \pi t\big)\ ) u_o(t)\ Volts$$

Then the output response for the lowpass filter can be found using the following code snippet:

```
vi = sin(2000*pi*t) + cos(2e6*pi*t)
t,y2,svec = sp.lsim(H,vi,t)
```

The output response for sum of the sinusoids is as shown:

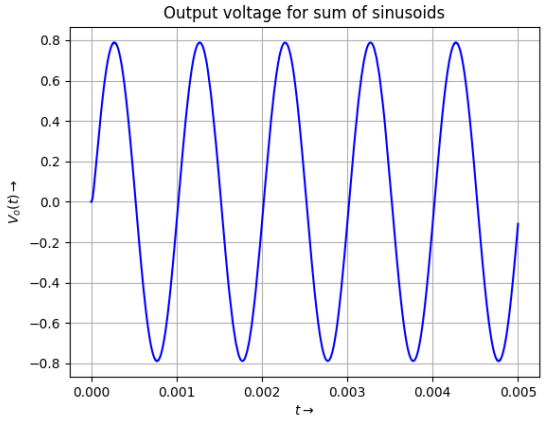

Figure 4: Output response to the sum of sinusoids.

4.3 Question 3

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

$$V_i(t) = e^{-500t}(\ \cos(2000\pi t)\ ) u_o(t)\ Volts$$

High frequency,

$$V_i(t) = e^{-500t}(\ \cos\big(2 * 10^6 \pi t\big)\ ) u_o(t)\ Volts$$

The high frequency and low frequency plots of the input damped sinusoids are as shown:
The python code snippets to execute is as shown:

```
# High frequency damped sinusoid.
t2 = arange(0,1e-2,1e-5)
vi_d_1 = exp(-500*t2)*cos(2e6*pi*t2)
t2,y4,svec = sp.lsim(H3,vi_d_1,t2)
```

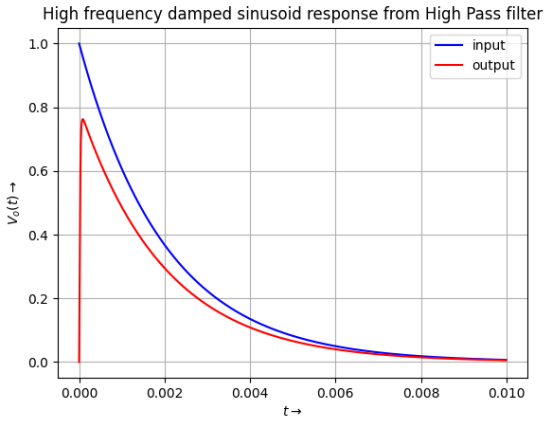

Figure 5: High frequency damped sinusoid

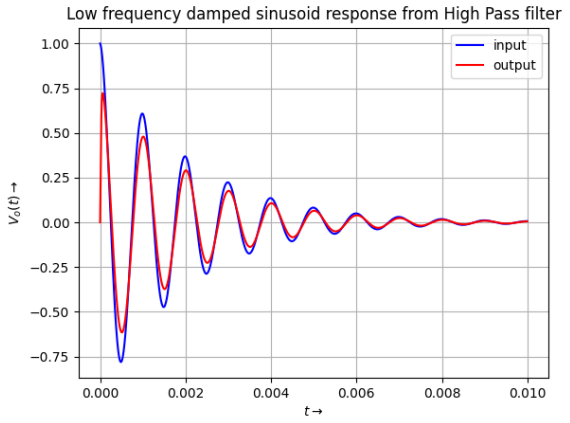

Figure 6: Low frequency damped sinusoid

```
# Low frequency damped sinusoid.
t3 = arange(0,1e-2,1e-5)
vi_d_2 = exp(-500*t3)*cos(2e3*pi*t3)
t3,y4_2,svec = sp.lsim(H3,vi_d_2,t3)
```

4.4 Question 5

In order to find the step response of the high pass filter, we need to assign $Vi(s) = 1/s$. The python code below explains it.

```
A5,b5,V5 = highpass(10000,10000,1e-9,1e-9,1.586,1/s)
Vo5 = V5[3]
H5 = sympytoscipy(Vo5)
t5,y5 = sp.impulse(H5,None,linspace(0,5e-3,10000))
```

The plot of the step response for a high pass filter is as shown:

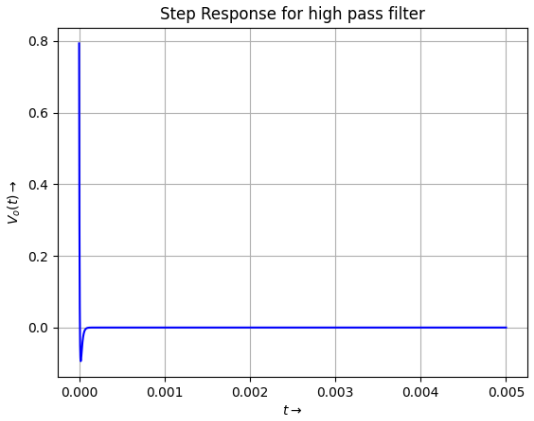

Figure 7: Step response for a high pass filter

The unit step response, as expected is high at t=0 when there is an abrupt change in the input. Since there is no other change at large time values outside the neighbourhood of 0, the Fourier transform of the unit step has high values near 0 frequency, which the high pass filter attenuates.

5 Conclusion

In conclusion, the sympy module has allowed us to analyse quite complicated circuits by analytically solving their node equations. We then 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.