EE2703 : Applied Programming Lab
Assignment 7

Andapally Snehitha
EE20B008

April 8, 2022

Introduction

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. Besides this the library also includes useful classes to handle the simulation and response to inputs.

Coupled with scipy's signal module, we are able to analyse both High pass and low pass filters, both second order, realised using a single opamp.

Assignment Questions

Question 1

In order to find the step response of the low pass filter, we first need to define the system. The following piece of code implements a low pass filter:

```
# Declaring the sympy function for a lowpass filter.
def lowpass(R1, R2, C1, C2, G, Vi):
    s = sy.symbols('s')
    # Creating the matrices and solving them to get the output voltage.
    A = sy.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 = sy.Matrix([0,0,0,-Vi/R1])
    V = A.inv()*b
    return A, b, V
```

Since the system is an expression in s, the following piece of code is used to convert it into an object that can be used by the signal toolbox.

```
# Creating a function to convert a sympy function into a version that is understood by sp.signal
def TFconverter(h):
    s = sy.symbols('s')
    n, d = sy.fraction(h)
    N = sy.Poly(n, s).all_coeffs()
    D = sy.Poly(d, s).all_coeffs()
    N, D = [float(f) for f in N], [float(f) for f in D]
    return sp.lti(N, D)
```

The step response is found by using lsim() and is plotted in Figure 1.

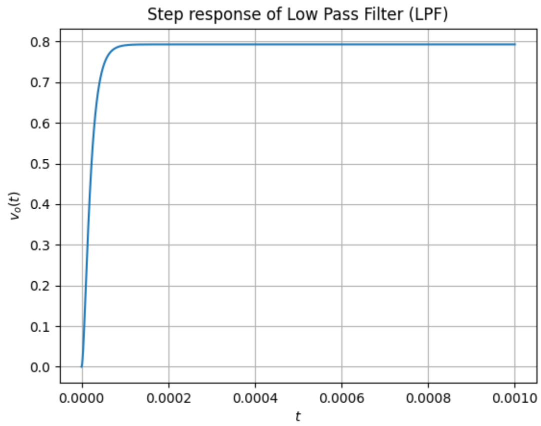

Figure 1: Step Response of Low Pass Filter

Question 2

The given input is

$$\mathrm{vi(t)} = (\sin(200\pi \mathrm{t}) + \cos(2 \times 106\pi \mathrm{t}))\mathrm{u0(t)}$$

The response to this input should be a signal with low frequency, since the high frequencies are blocked by the LPF. The response is given in Figure 2.

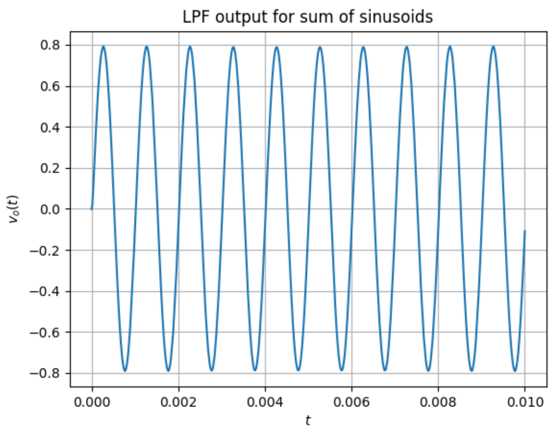

Figure 2: LPF response for sum of sinusoids

Question 3

In this section we simulate a highpass filter. The system is defined using the piece of code given below:

```
# Declaring the sympy function for a highpass filter.
def highpass(R1, R2, C1, C2, G, Vi):
    s = sy.symbols('s')
    # Creating the matrices and solving them to get the output voltage.
    A = sy.Matrix([[0,-1,0,1/G], [s*C2*R2/(s*C2*R2+1),0,-1,0], [0,G,-G,1], [-1/R1-s*C1-s*C2,0,s*C2,1/R1]])
    b = sy.Matrix([0,0,0,-Vi*s*C1])
    V = A.inv()*b
    return A, b, V
```

The magnitude response of the filter is shown in Figure 3:

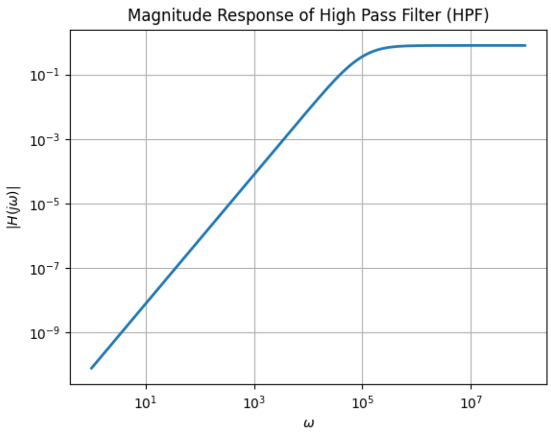

Figure 3: $|H(j\omega)|$ for high pass filter

Question 4

In this section, we observe the response of HPF to damped sinusoids. We consider two sinusoids,one with low frequency and the other with high frequency. The decay is the same for both sinusoids. The sinusoids are

$$\mathrm{e}^{-1000t}\cos(2\times 106\pi \mathrm{t}),\ \mathrm{e}^{-1000t}\cos(2000\pi \mathrm{t})$$

The responses of the High pass filter for these signals are given in Figures 4 and 5.

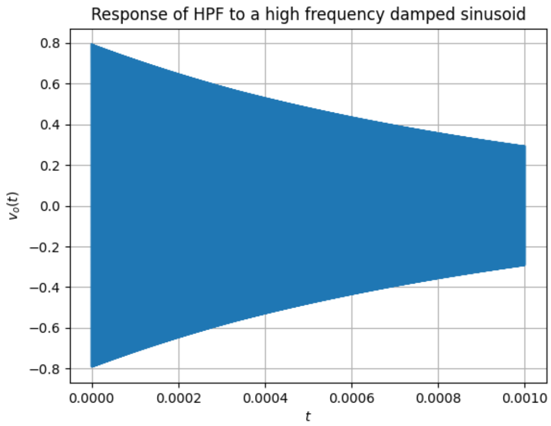

Figure 4: HPF output for $e^{-1000t}\cos(2 \times 106\pi t)$

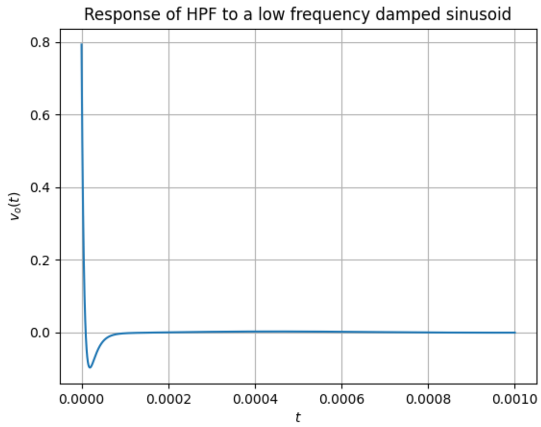

Figure 5: HPF output for $e^{-1000t}\cos(2000\pi t)$

Question 5

The step response of the HPF is given in Figure 6. The fourier transform of u(t) is given as

$$U(j\omega) = \frac{1}{j\omega} + \pi\delta(\omega)$$

We see that the high frequency components of u(t) are very small because of the $\frac{1}{j\omega}$ variation.Hence, the output of HPF coverges to 0.

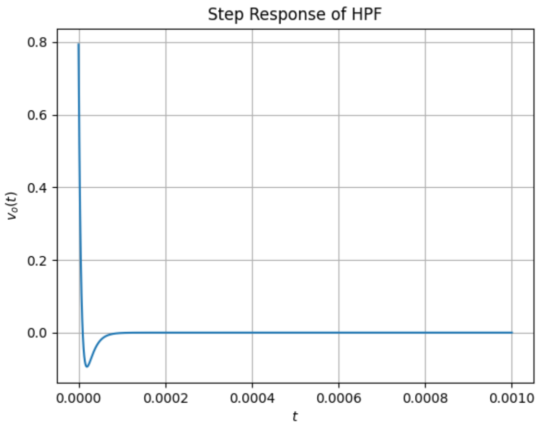

Figure 6: Step Response of High Pass Filter

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.