EE2703 : Applied Programming Lab
Assignment 7

P N Neelesh Sumedh
ee19b047

May 4, 2021

1 Introduction

In this assignment, the focus will be mostly on two powerful capabilities of Python:
1) Symbolic Algebra
2)Analysis of Circuits using Laplace Transform

2 Low Pass Filter

A Low Pass Filter is a filter which allows only low frequencies to pass and does not allow higher frequencies in the output. The transfer function of a low pass filter is of the form

$$H(s) = \frac{1}{1 + \frac{s}{w_p}} \tag{1}$$

The magnitude plot of LPF is as follows

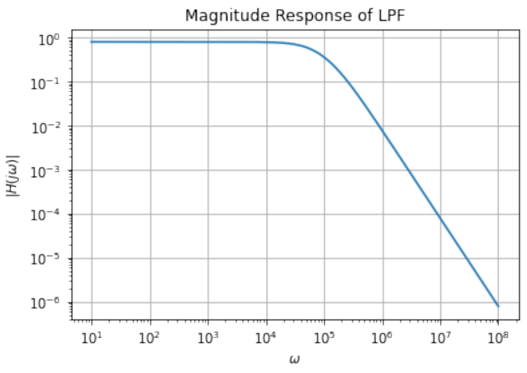

Figure 1: Magnitude Response of LPF

3 Assignment

3.1 Step Response of the given LPF circuit

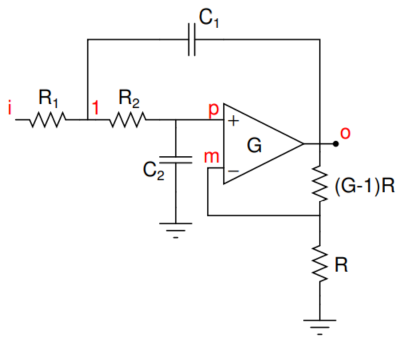

Figure 2: Low Pass Filter

The above given circuit is a LPF. First we define a function *lowpassfilter* to give the necessary inputs and get the transfer function as the output.

```
    def lowpassfilter(R1,R2,C1,C2,G,Vi):
        s = sy.symbols('s')
        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)
```

The output of the above function is of type sympy. But to use signal toolbox of scipy, we have to convert this into scipy. For this we write another function.

```
    def TFconvt(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)
```

This function gives the transfer function in type scipy. This can be used to find the step response using signal toolbox. The code to find the step response is as follow:

```
    s = sy.symbols('s')
    (A1,b1,V1) = lowpassfilter(10000,10000,1e-9,1e-9,1.586,1/s)
    H1 = TFconvt(V1[3])
    t,SR1 = sp.impulse(H1,None,np.linspace(0,1e-3,10000))
    p.plot(t,SR1)
    p.grid(True)
    p.xlabel(r'$t$')
    p.ylabel(r'$v_o(t)$')
    p.title('Step response of LPF')
    p.show()
```

The step response is shown in the figure below

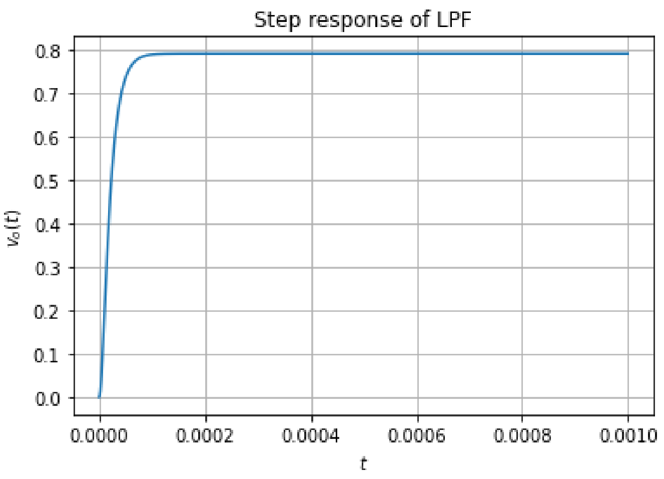

Figure 3: Step Response of LPF

3.2 Find output of LPF given the input

The given input is

$$v_i(t) = \sin(2000\pi t)u_0(t) + \cos(2 * 10^6\pi t)u_0(t) \tag{2}$$

The output of LPF when the above input is given contains only the low frequency part of the signal. The higher frequencies are neglected by the filter.

```
    t2 = np.linspace(0,1e-3,1000000)
    vi2 = np.sin(2000*np.pi*t2)+np.cos(2e6*np.pi*t2)
    (A2,b2,V2) = lowpassfilter(10000,10000,1e-9,1e-9,1.586,1)

    H2 = TFconvt(V2[3])
    t2,Vo2,svec = sp.lsim(H2,vi2,t2)
    p.plot(t2,vi2)
    p.plot(t2,Vo2)
    p.grid()
    p.xlabel(r'$t$')
    p.ylabel('V')
    p.title('LPF output for sum of sinusoids along with input')
    p.show()
```

The output is shown below

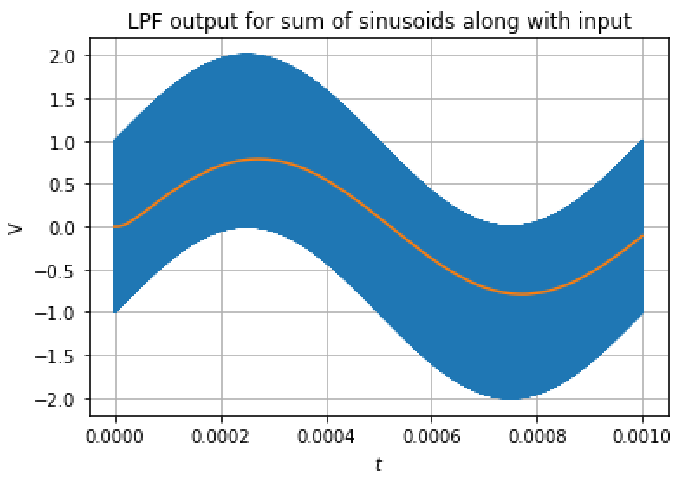

Figure 4: LPF output for input with a high frequency and a low frequency

The blue plot in the above graph is the input signal and the orange plot is the output signal. We can see that the output contains only the small frequency while the higher frequency is neglected.

3.3 Transfer Function of the given HPF

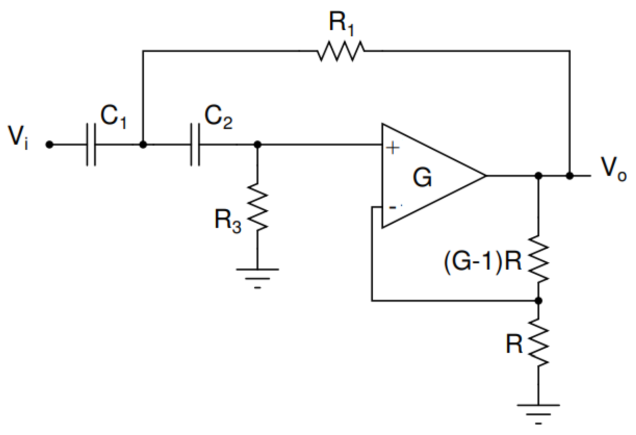

Figure 5: High Pass Filter

The above circuit is a HPF. To find the transfer function of this filter, we define a function *highpassfilter*.

```
    def highpassfilter(R1,R2,C1,C2,G,Vi):
        s = sy.symbols('s')
        A = sy.Matrix([[0,0,1,-1/G],[-s*R2*C2/(1+s*R2*C2),1,0,0],[0,-G,G,1],[-s*
    C1-s*C2-1/R1,s*C2,0,1/R1]])
```

```
        b = sy.Matrix([0,0,0,-Vi*s*C1])
        V = A.inv()*b
        return (A,b,V)
```

Again the output of this function is of type sympy. We use the *TFconvt* function to convert into scipy. This gives us the transfer function of the HPF. The magnitude plot of the HPF is shown below.

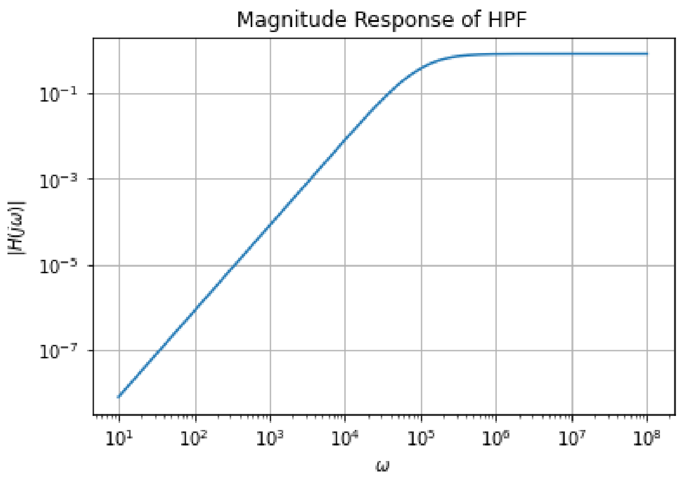

Figure 6: Magnitude Response of HPF

3.4 Output of HPF to a damped sinusoid input

When a sinusoid is given as input to a HPF, the high frequency components are passed where as the lower frequencies are rejected. Therefore, depending on the input frequency, the Output will differ.
In Figure 7, the plot is the output of HPF with input equal to $\cos(1.7 * 10^6 \pi t)e^{-100t}$. The output is almost equal to input with a little decrease in amplitude.
In Figure 8, the plot is the output of HPF with input equal to $\cos(1 * 10^3 \pi t)e^{-100t}$. Since the frequency is small, the output is almost equal to zero.

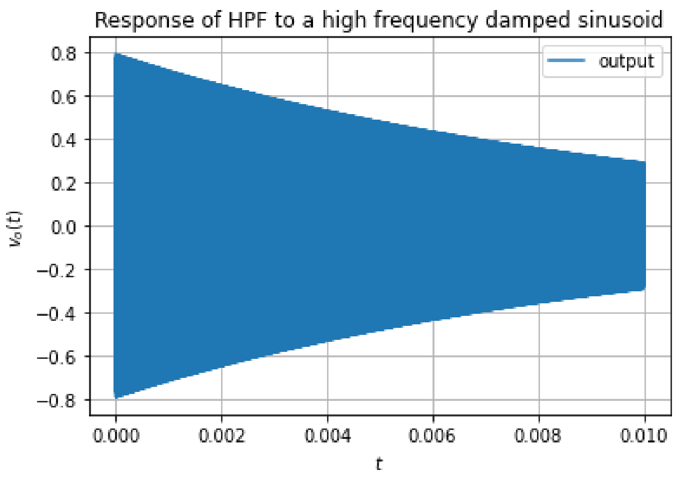

Figure 7: Output of HPF with high frequency damping sinusoid

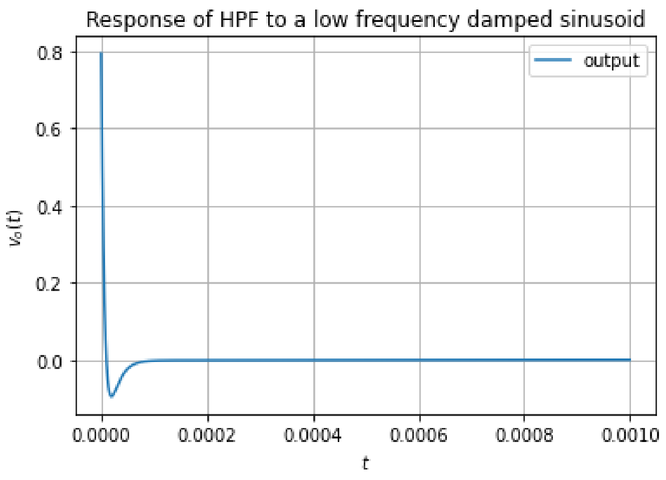

Figure 8: Output of HPF with low frequency damping sinusoid

3.5 Step Response of HPF

The step response of HPF can be found out similar to the step response of LPF. For the input signal, $u_0(t)$, the fourier transform is

$$U(j\omega) = \frac{1}{j\omega} + \pi\delta(\omega) \tag{3}$$

We can see that, higher frequencies have very less amplitude in the input, therefore the output also quickly reaches to zero as we can see in Figure 9.

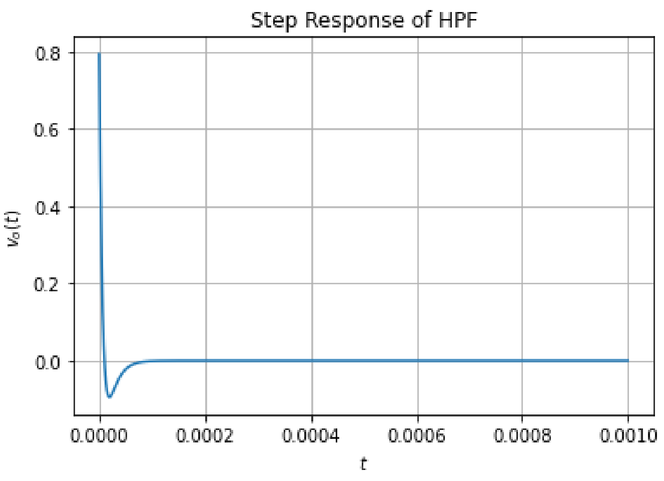

Figure 9: Step Response of HPF

4 Conclusion

We implemented High Pass Filter and Low Pass Filter using sympy and scipy modules. We found out the step responses of both the filters and also the outputs when the input is a damping sinusoid.