Assignment No 7

Shivanshu Shekhar

April 6, 2022

1 Introduction

In this assignment, we learnt:

- Symbolic Algebra capabilities of python.
- Analysis of Circuits using Laplace Transforms.

We also touched upon the basics of high and low pass filter.

2 Problems and Solutions

2.1 Question 1

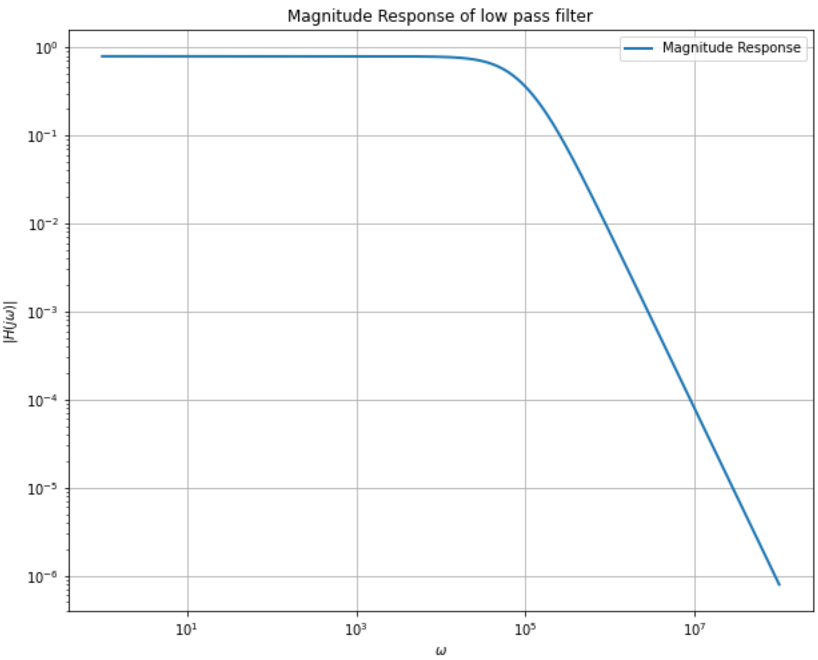
The low pass filter that we used, gave the following matrix equations on solving:

$$\begin{bmatrix} 0 & 0 & 1 & -1/G \\ \frac{-1}{1+sc_2R_2} & 1 & 0 & 0 \\ 0 & -G & G & 1 \\ -\frac{1}{R_1}-\frac{1}{R_2}-sC_1 & \frac{1}{R_2} & 0 & sC_1 \end{bmatrix} \begin{bmatrix} V1 \\ Vp \\ Vm \\ V0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\frac{Vi(s)}{R_1} \end{bmatrix}$$

We implemented the following equation and solved for the voltage vector, we did the following by using `sympy`, we also plotted the magnitude response for the same:

```
def lowpass(R1,R2,C1,C2,G,Vi):

    A = simp.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 = simp.Matrix([0,0,0,-Vi/R1])
```

```
    V = A.inv()*b

    return A, V, b

R1, R2, C1, C2, G, Vi = 1e4, 1e4, 1e-9, 1e-9, 1.586, 1
A,V,b=lowpass(R1, R2, C1, C2, G, Vi)
print(f"G = {G}")
Vo = V[3]
print(Vo)
ww = np.logspace(0, 8, 801)
ss = simp.CC(1j)*ww
hf = simp.lambdify(s, Vo, "numpy")
v = hf(ss)

plt.figure(figsize=(10,8))
plt.title("Magnitude Response of lowpass filter")
plt.loglog(ww,abs(v), lw = 2, label = "Magnitude Response")
plt.xlabel(r"$|H(j\omega)|$")
plt.ylabel(r"$\omega$")
plt.grid()
plt.legend()
plt.show()
```

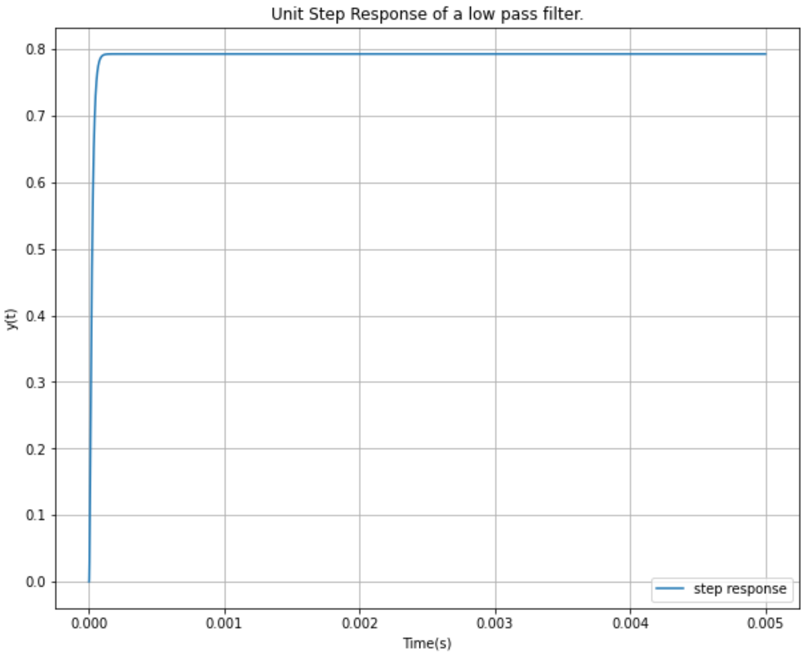


Once we have obtained the system response from `sympy` in the symbolic form, we need to get the coefficients of the function to feed it to `scipy.signal.lti` in order to work with it further down the line, we used the following function for the same:

```
def coeff(expr, var = s):
    num, denom = expr.as_numer_denom()
    num = [float(i) for i in simp.Poly(num, var).all_coeffs()]
    denom = [float(i) for i in simp.Poly(denom, var).all_coeffs()]
    return num, denom
```

We then plotted the unit step response of the lowpass filter using `scipy.impulse` and plotted the same.

```
A,V,b=lowpass(R1, R2, C1, C2, G, 1/s)
Vo = V[3]
hs = sig.lti(*coeff(Vo))
t = np.linspace(0, 5e-3, 10**4)
t, v = sig.impulse(hs,T = t)

plt.figure(figsize=(10,8))
plt.title("Unit Step Response of a lowpass filter.")
plt.plot(t, v, label = "step response")
plt.legend()
plt.xlabel("y(t)")
plt.ylabel("Time(s)")
plt.grid()
plt.show()
```



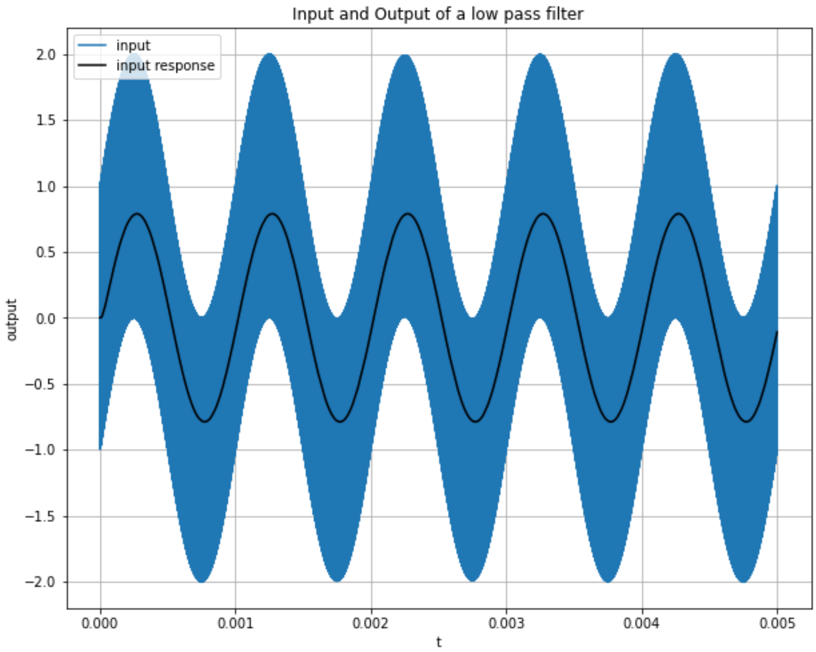
2.2 Question 2

We had to find the Low pass system response for the input signal:

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

This is accomplished by the following:

```
A,V,b = A,V,b=lowpass(R1, R2, C1, C2, G, Vi = 1)
Vo = V[3]
hs = sig.lti(*coeff(Vo))
t = np.linspace(0, 5e-3, 10**5)
t, v, _ = sig.lsim(hs, U = inSignal(t), T = t)

plt.figure(figsize=(10,8))
plt.title("Input and Output of a low pass filter")
plt.plot(t, inSignal(t), label="input")
plt.plot(t, v, "k", label = "input response")
plt.legend()
plt.grid()
plt.show()
```



This can be explained by the basic logic of a low pass filter, i.e. it passes only those frequencies that are less than its pole frequency that is 10^5, so we can clearly see that it passed 10^3 almost unchanged but rejected 10^6.

2.3 Question 3

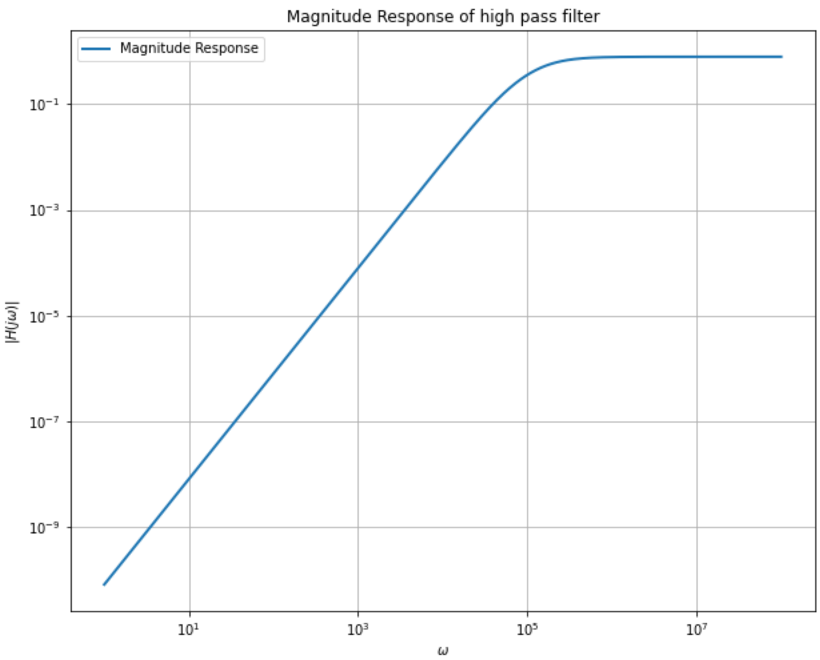
Next we do the same analysis for the High pass filter:

$$\begin{bmatrix} 0 & -1 & 0 & -1/G \\ \frac{sc_2R_3}{sc_2R_3+1} & 0 & -1 & 0 \\ 0 & G & -G & 1 \\ -sc_2 - \frac{1}{R_1} - sc_1 & 0 & sc_2 & \frac{1}{R_1} \end{bmatrix} \begin{bmatrix} V1 \\ Vp \\ Vm \\ V0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ Vi(s)c_1 \end{bmatrix}$$

```
def highpass(R1,R3,C1,C2,G,Vi):

    A=simp.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=simp.Matrix([0,0,0,-Vi*s*C1])

    V = A.inv()*b

    return (A,V,b)
```

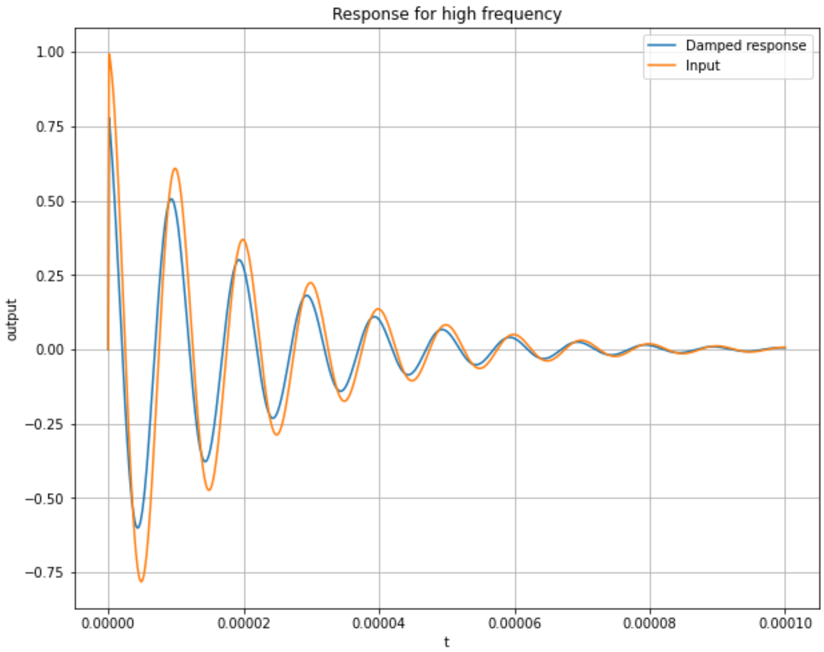
```
A,V,b=highpass(R1, R2, C1, C2, G, Vi)
print(f"G = {G}")
Vo = V[3]
print(Vo)
ww = np.logspace(0, 8, 801)
ss = simp.CC(1j)*ww
hf = simp.lambdify(s, Vo, "numpy")
v = hf(ss)

plt.figure(figsize=(10,8))
plt.title("Magnitude Response of high pass filter")
plt.loglog(ww,abs(v), lw = 2, label = "Magnitude Response")
plt.xlabel(r"$|H(j\omega)|$")
plt.ylabel(r"$\omega$")
plt.grid()
plt.legend()
plt.show()
```

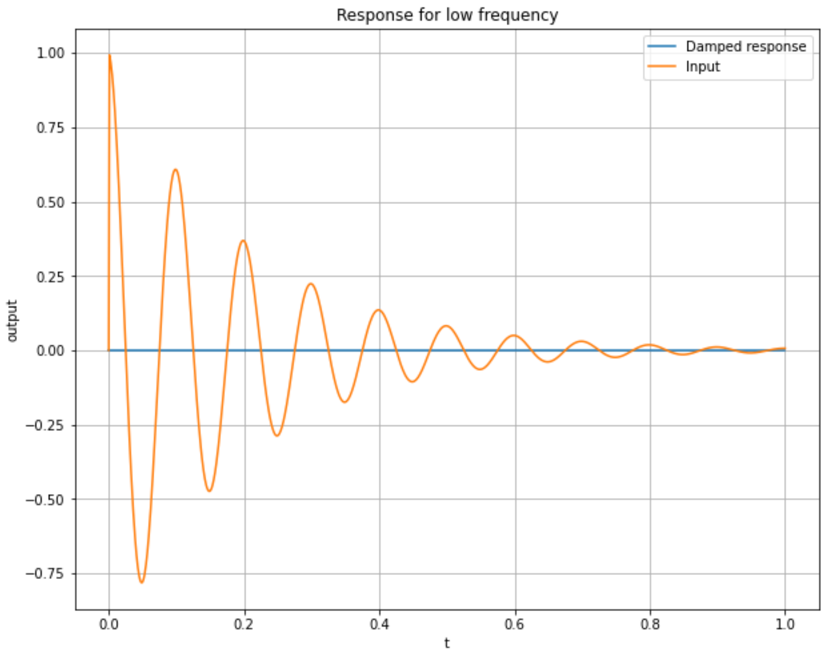
We also plotted the magnitude response of the filter:



2.4 Question 4

Now we plot the response of the high pass filter to a high frequency decaying signal and low frequency decaying signal:






```
def indampedS(t,decay=5e4,freq=1e8):
    return np.cos(2*np.pi*freq*t)*np.exp(-decay*t) * (t>0)


A,V,b=highpass(R1, R2, C1, C2, G, Vi)
```

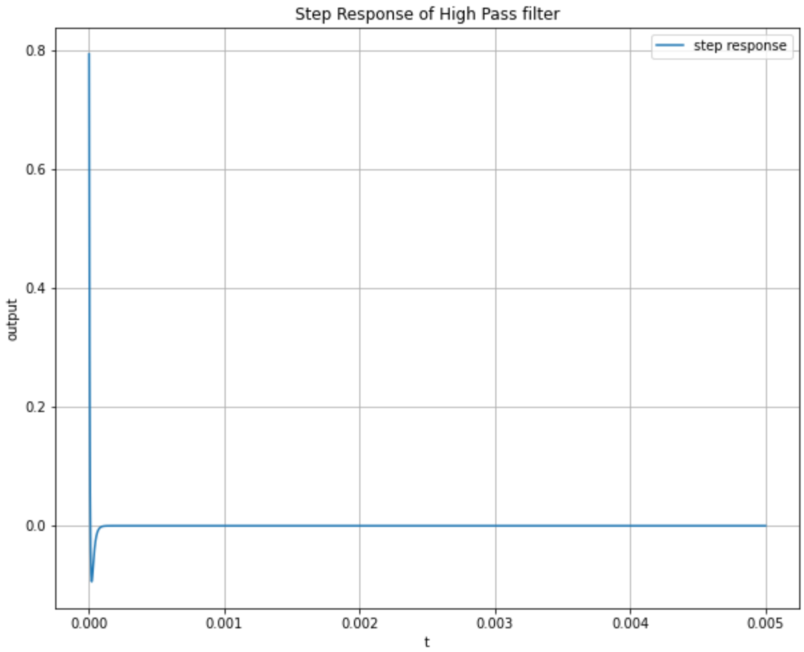
```
t = np.linspace(0, 1e-4, 1000)
hs = sig.lti(*coeff(Vo))
t, v, _ = sig.lsim(hs, U=indampedS(t), T = t)
plt.figure(figsize=(10,8))
plt.title("Response for high frequency")
plt.plot(t, v, label = "Damped response")
plt.plot(t, indampedS(t), label = "Input")
plt.legend()
plt.grid()
plt.show()

t = np.linspace(0, 1, 1000)
hs = sig.lti(*coeff(Vo))
t, v, _ = sig.lsim(hs, U=indampedS(t, 5, 10), T = t)
plt.figure(figsize=(10,8))
plt.title("Response for low frequency")
plt.plot(t, v, label = "Damped response")
plt.plot(t, indampedS(t, 5, 10), label = "Input")
plt.legend()
plt.grid()
plt.show()
```

We can clearly see that the low frequency signal is not allowed to pass by the high pass filter and the high frequency component is almost unchanged by the filter.

2.5 Question 5

To get the unit step response we pass 1/s to the highpass function and then compute the response using `scipy.signal.impulse`, we did this from the following lines of code:

```
A,V,b=highpass(R1, R2, C1, C2, G, 1/s)
Vo = V[3]
hs = sig.lti(*coeff(Vo))
t = np.linspace(0, 5e-3, 10**4)
t, v = sig.impulse(hs,T = t)

plt.figure(figsize=(10,8))
plt.title("Step Response of High Pass filter")
plt.plot(t, v, label = "step response")
plt.legend()
plt.grid()
plt.show()
```



As soon as the voltage is applied, i.e at t=0+, the capacitors behaves as conducting wires, and thus we see a positive voltage at the output node, whereas at $t = \infty$ (for practical purposes, this time is not very large), the capacitors would behave as an open-circuit for a DC voltage and thus we would see zero volts at the output node.

3 Conclusion

- We learnt how to use `sympy` library to do symbolic algebra.
- We also learnt how to integrate it with `scipy.signal` to solve circuits.
- We also learnt about basic circuits components and filters