EE2703 : Applied Programming Lab
Assignment 8
Circuit Analysis using Laplace Transforms

Bachotti Sai Krishna Shanmukh
EE19B009

April 27, 2021

Introduction

In this assignment, we use a module called SymPy for symbolic algebra to solve circuit equations and get the symbolic expressions of required Transfer functions. Along with it, we use the Signals tool box from SciPy for better analysis of systems and their behaviour.

1 Low Pass Butterworth Filter

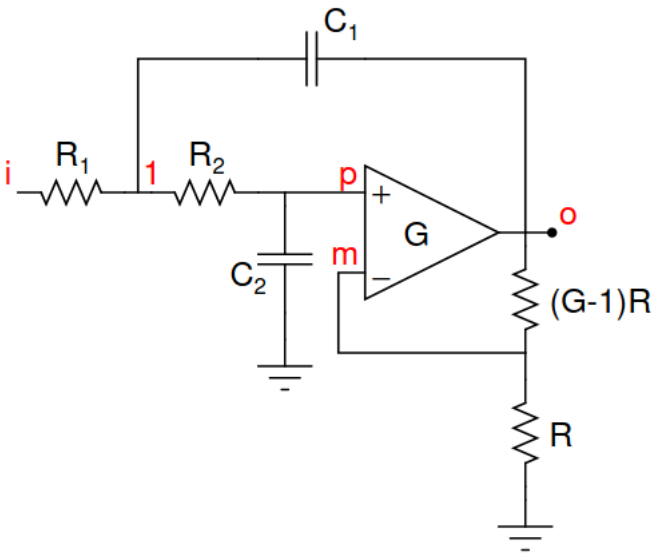

Figure 1: Low Pass Butterworth Filter

1.1 Circuit parameters and Matrix Equation

Above is the circuit representation of the famously known Butter-worth Low Pass filter. Given, G = 1.586, R1 = R2 = $10k\Omega$, C1 = C2 = 1nF. This circuit gives the following nodal matrix upon writing the current and voltage equations.

$$\begin{bmatrix} 0 & 0 & 1 & -\frac{1}{G} \\ \frac{-1}{sR_2C_2} & 1 & 0 & 0 \\ 0 & -G & G & 1 \\ -\frac{1}{R_1} - \frac{1}{R_2} - sC_1 & \frac{1}{R_2} & 0 & sC_1 \end{bmatrix} \begin{bmatrix} V_1(s) \\ V_p(s) \\ V_m(s) \\ V_o(s) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\frac{V_i(s)}{R1} \end{bmatrix}$$

Below is the code snippet for low pass filter

```
def lowpass(R1,R2,C1,C2,G,Vi):
    s = sympy.symbols('s')
    A = sympy.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 = sympy.Matrix([0,0,0,-Vi/R1])
    V = A.inv()*b
    return A,b,V
```

1.2 Bode Plots

Using the lambdify function from sympy, we can convert the symbolic expression to a numpy compatible function. This can be used for the Bode plots of the system transfer function. Below is the code snippet for bode plots

```
def bode_plotter(vo, title,plt_num):
    s = sympy.symbols('s')
    w = np.logspace(0,8,801)
    ss = 1j*w
    hf = sympy.lambdify(s,vo,'numpy')
    v_val = hf(ss)
    fig,ax = plt.subplots(num=plt_num,figsize=(8,7))
    ax.loglog(w,np.abs(v_val),lw=2)
    plt.xlabel(r' $\omega\longrightarrow$')
    plt.ylabel(r' Magnitude$\longrightarrow$')
    plt.title('Magnitude response of '+title+' Filter')
    plt.grid()
    fig,ax = plt.subplots(num= plt_num+1,figsize=(8,7))
    ax.semilogx(w,np.degrees(np.angle(v_val)),lw=2)
    plt.xlabel(r' $\omega\longrightarrow$')
    plt.ylabel(r' $\phi\longrightarrow$')
    plt.title('Phase response of '+title+' Filter')
    plt.grid()
```

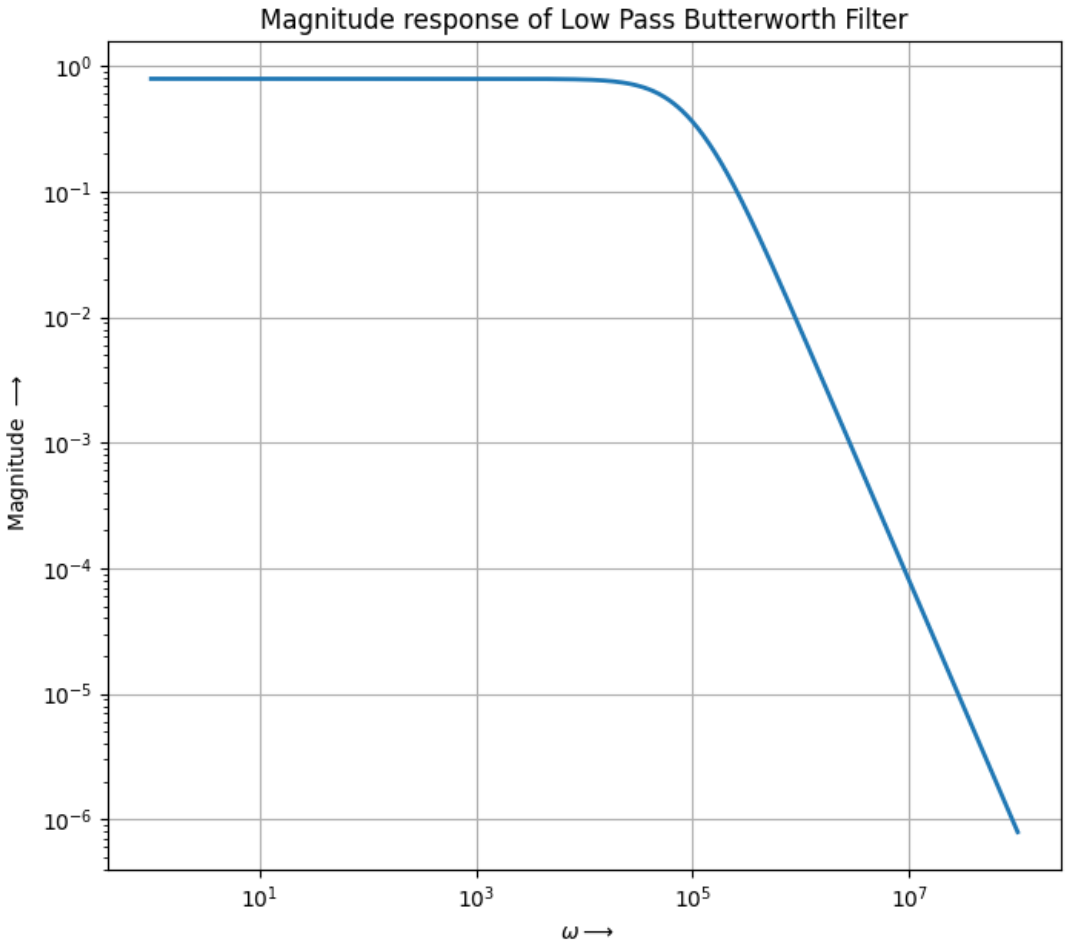

Figure 2: Magnitude Response

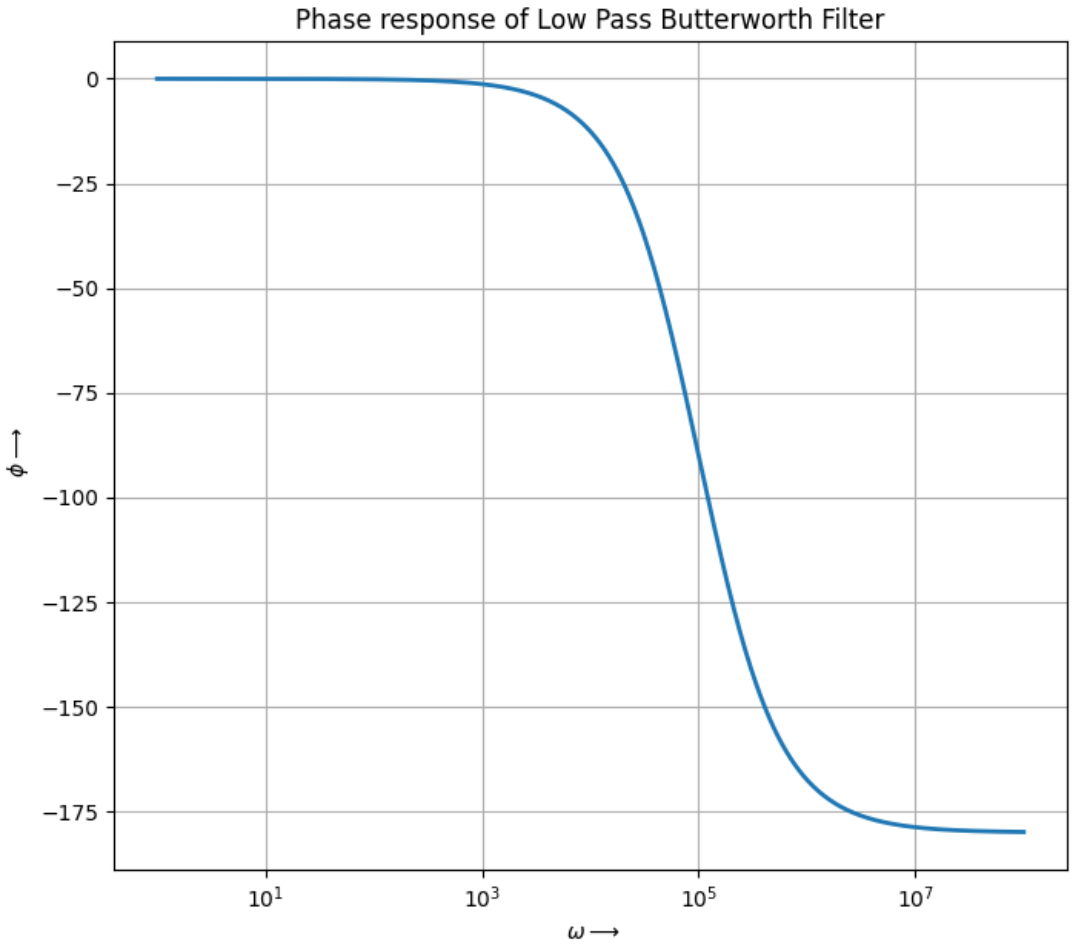

Figure 3: Phase response in degrees

1.3 Unit Step Response (Question 1)

If $H(s)$ is the system transfer function, then the Laplace transform of the unit step response $U(s)$ is given by

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

Below is the code snippet for finding the step response

```
def sym2sig(SymX):
    s = sympy.symbols('s')
    X = sympy.simplify(SymX)
    n,d = sympy.fraction(X)
    n,d = sympy.Poly(n,s), sympy.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
#Refer to source code for documentation
def stepresponse(SymX,tstop):
    num,den = sym2sig(SymX)
    den.append(0)
    H1 = sp.lti(num,den)
    return sp.impulse(H1,None,np.linspace(0,tstop,100000))
```

Below is the plot of the unit step response for Low Pass Filter

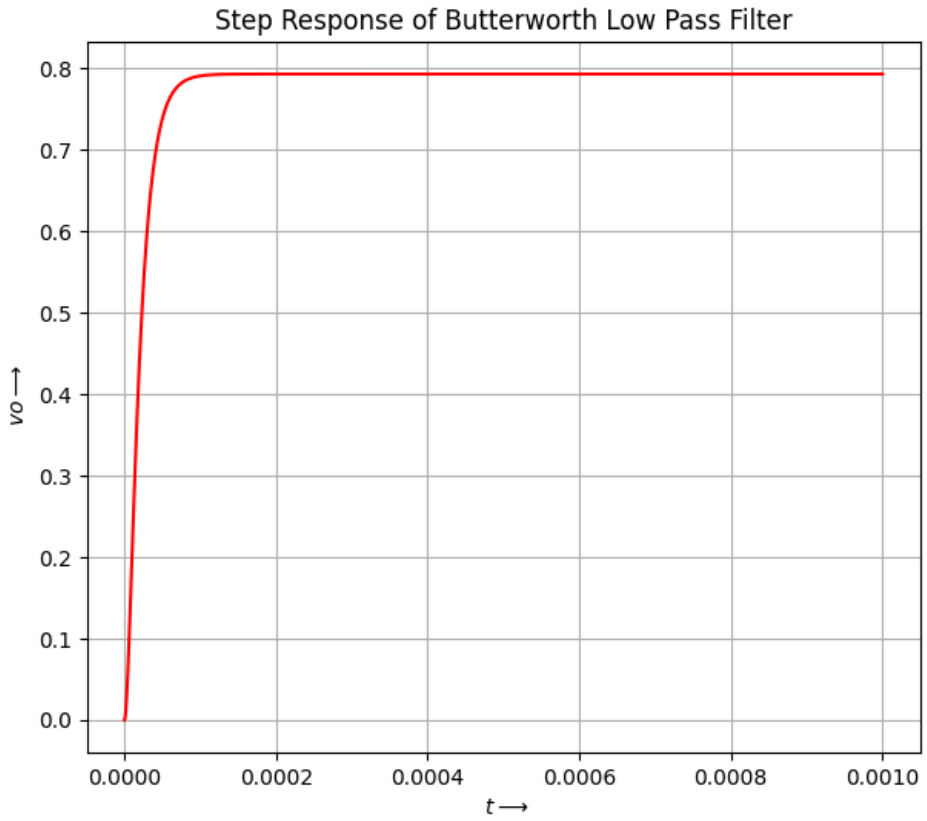

Figure 4: Unit Step Response of LPF

1.4 Output for Two Sinusoids of different frequencies(Question 2)

Given,

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

Below, is the plot of the signal $v_i(t)$

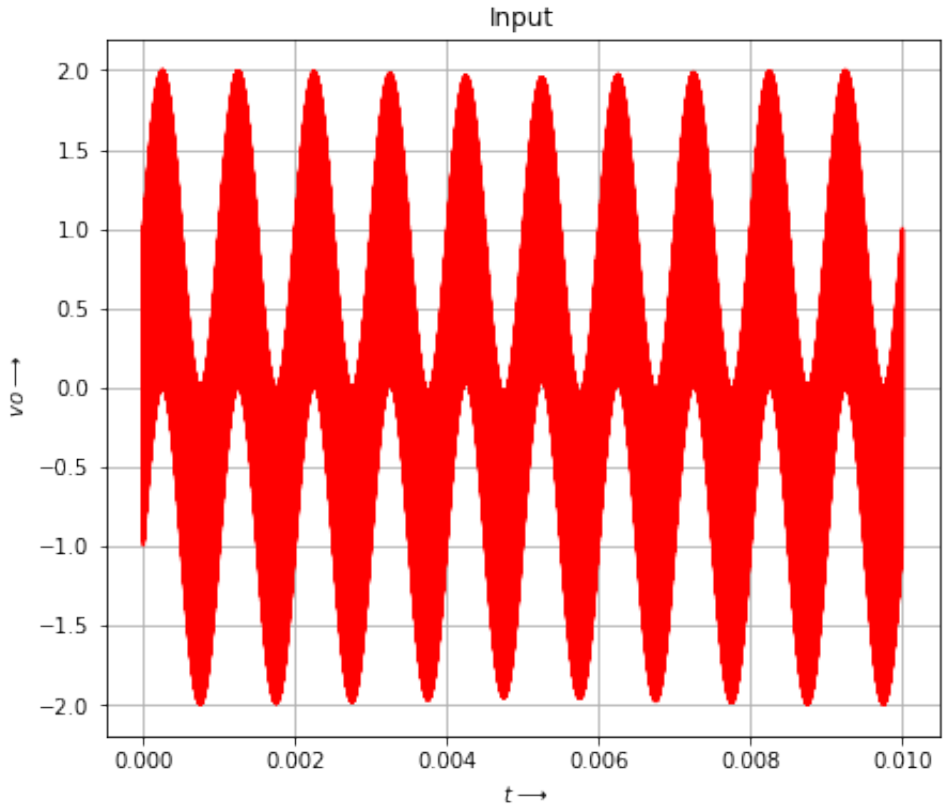

Figure 5: Plot of the signal $v_i(t)$

The voltage $v_o(t)$ can be determined by using signal.lsim() function. Below is the code snippet for response of any input signal

```
def input_response(SymX,f,tstop):
    num,den = sym2sig(SymX)
    H = sp.lti(num,den)
    t = np.linspace(0,tstop,100000)
    t,y,temp = sp.lsim(H,f(t),t)
    return t,y
```

Below is the plot of output of the Low Pass filter for given input

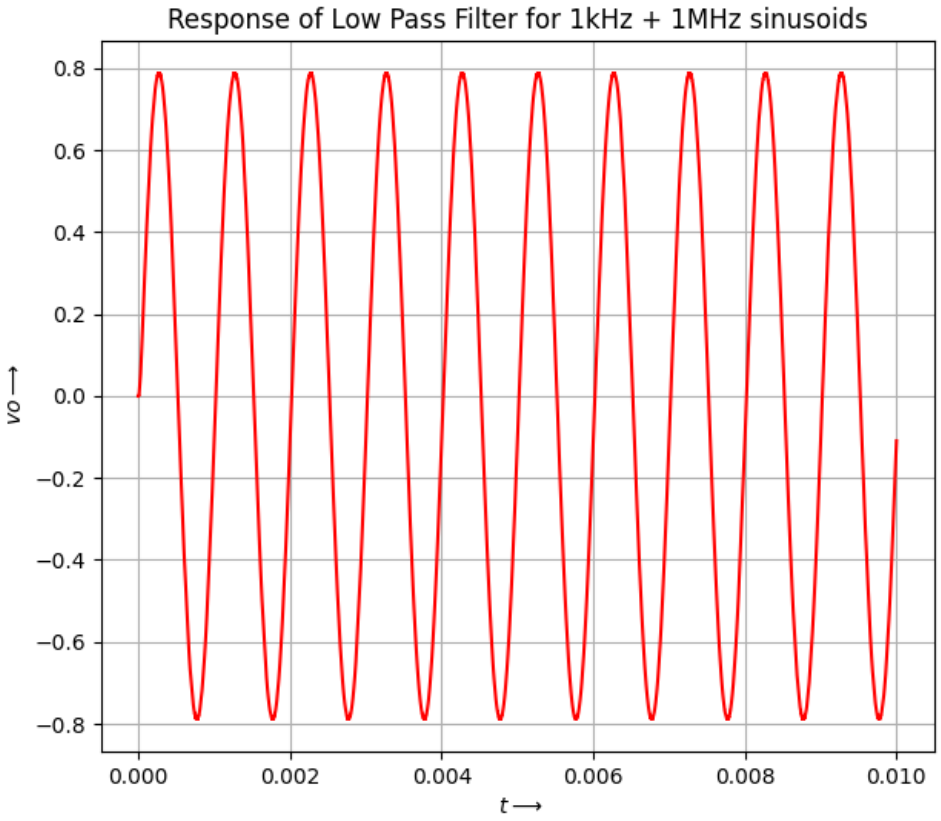

Figure 6: Output voltage of Low Pass filter for 10ms

We can infer that the 1MHz sinusoid is attenuated largely and the 1kHz sinusoid is seen as output of the Low Pass Filter.

2 High Pass Butterworth Filter

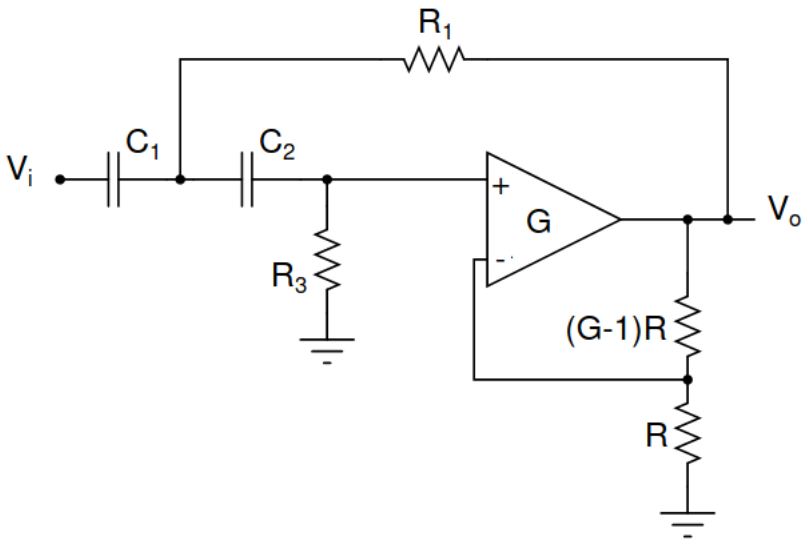

Figure 7: Circuit diagram of High Pass Butterworth Filter

2.1 Circuit parameters and Matrix Equation

Above is the circuit representation of the famously known Butter-worth High Pass filter. Given, G = 1.586, R1 = R2 = $10k\Omega$, C1 = C2 = 1nF. This circuit gives the following nodal matrix upon writing the current and voltage equations.

$$\begin{bmatrix} 0 & 0 & 1 & -\frac{1}{G} \\ -sC_2 & \frac{1}{R_3}+sC_2 & 0 & 0 \\ 0 & -G & G & 1 \\ -sC_1-sC_2-\frac{1}{R_1} & sC_2 & 0 & \frac{1}{R1} \end{bmatrix} \begin{bmatrix} V_1(s) \\ V_p(s) \\ V_m(s) \\ V_o(s) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -sC_1V_i(s) \end{bmatrix}$$

Below is the code snippet for High pass filter. (Question 3)

```
def highpass(R1,R3,C1,C2,G,Vi):
    s = sympy.symbols('s')
    A = sympy.Matrix([[0,0,1,-1/G],[0,-G,G,1],[-s*C2, 1/R3+s*C2, 0, 0],[-s*C1-s*
    C2-1/R1, s*C2, 0, 1/R1]])
    b = sympy.Matrix([0,0,0,-Vi*s*C1])
    V = A.inv()*b
    return A,b,V
```

2.2 Bode Plots

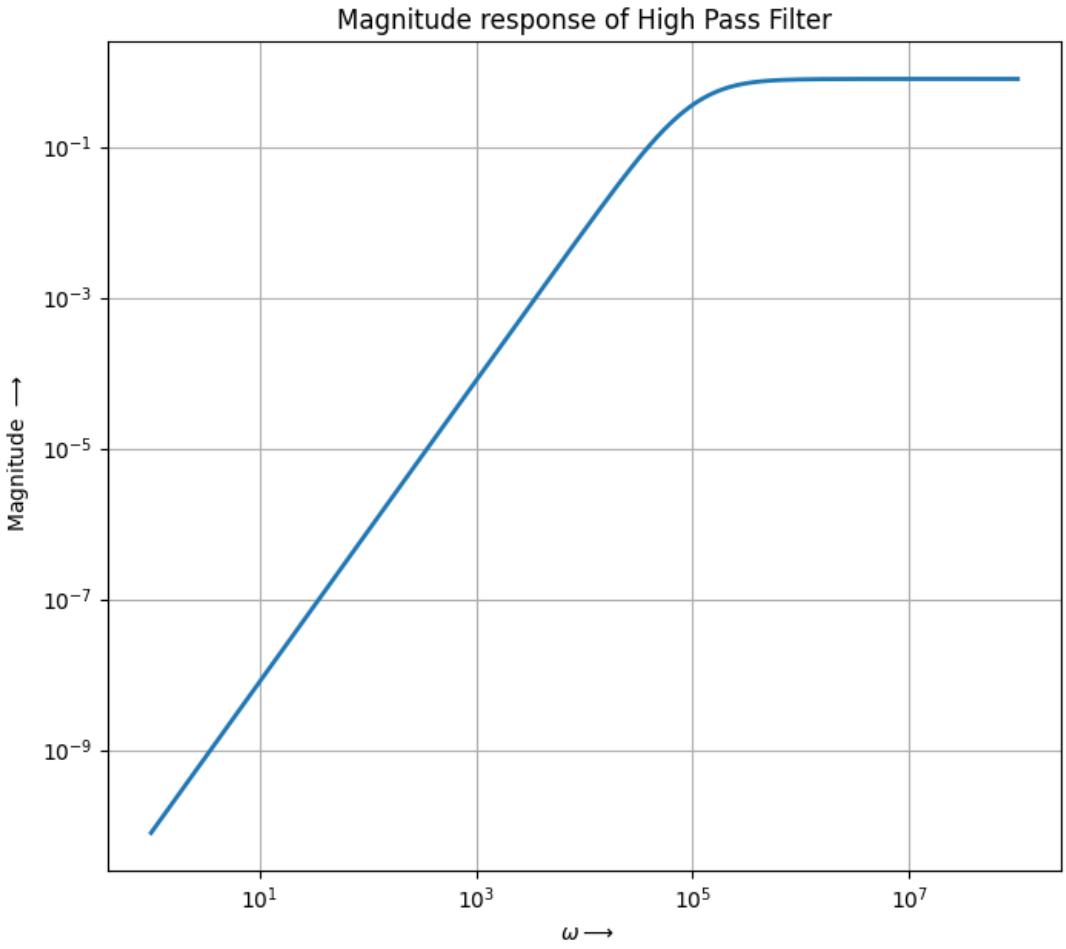

Figure 8: Magnitude response

We can infer from the magnitude response that the circuit behaves as a High pass filter

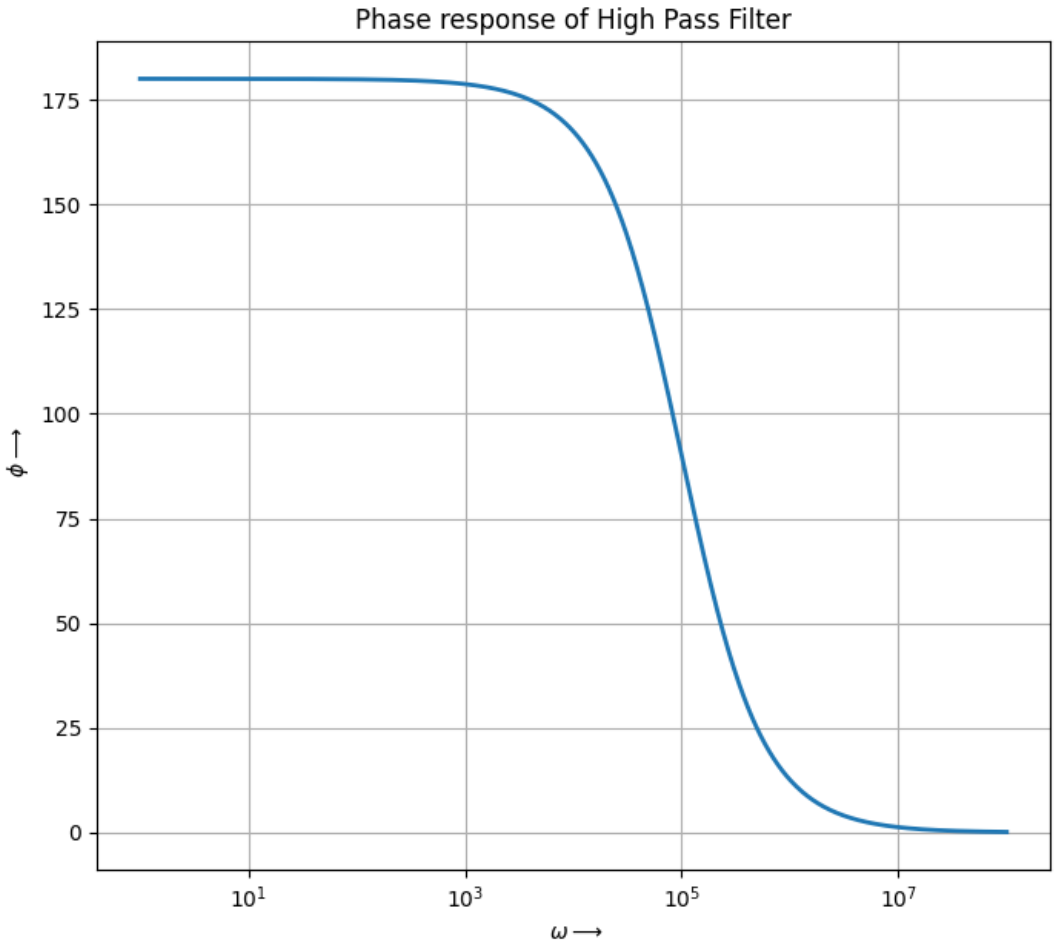

Figure 9: Phase response in degrees

2.3 Output for Two Sinusoids of different frequencies

If we use the same input $v_i(t)$ as in section 1.4 for the High Pass Butterworth filter, the output contains only the 1MHz sinusoid and the 1kHz sinusoid is largely attenuated. This explains the behaviour of High Pass Filter.

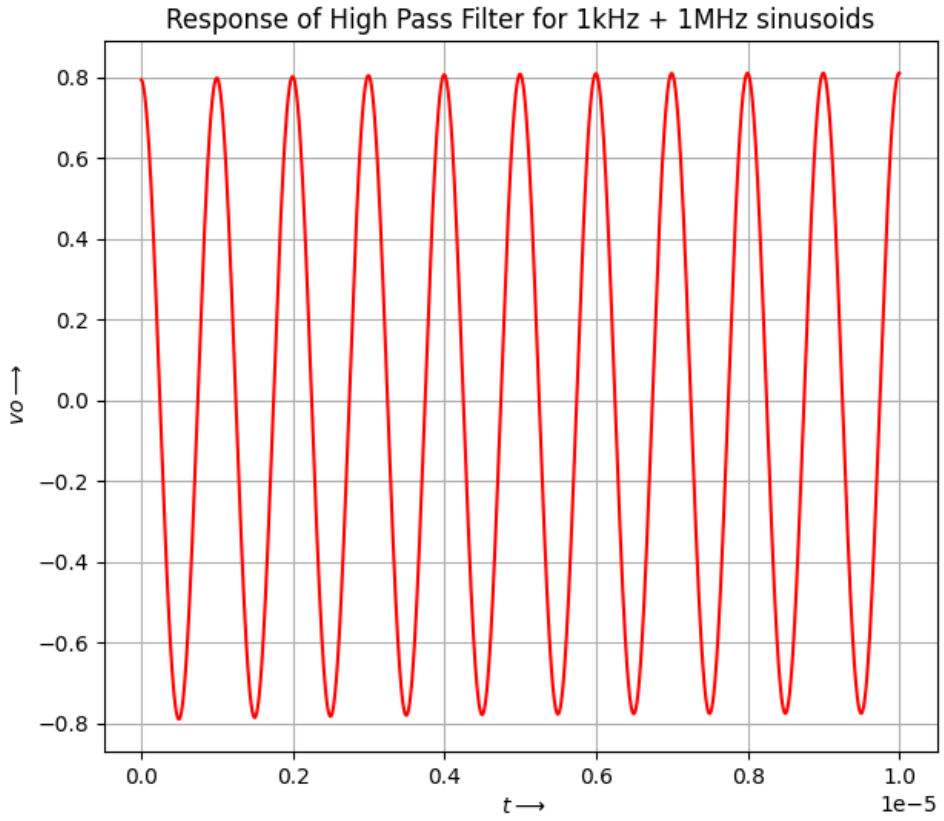

Figure 10: Output $v_o(t)$ of High Pass Filter

2.4 Output for Exponentially decaying sinusoid (Question 4)

To find the output of signal for a given system, we repeat the same process done in section 1.4 and 2.3 i.e, we use a function defined in the source code and pass the specific input signal function to it. Let's consider the signal,

$$v_i(t) = e^{-5t}cos(2000\pi t)u_o(t) \tag{3}$$

The above signal is exponentially decaying sinusoid oscillating at a frequency of 1kHz. Below is the plot of the output of the high pass filter for above input.
We can infer that the signal peaks at the start during the transient. Later, the signal gets attenuated and a small ripple is seen in steady state.

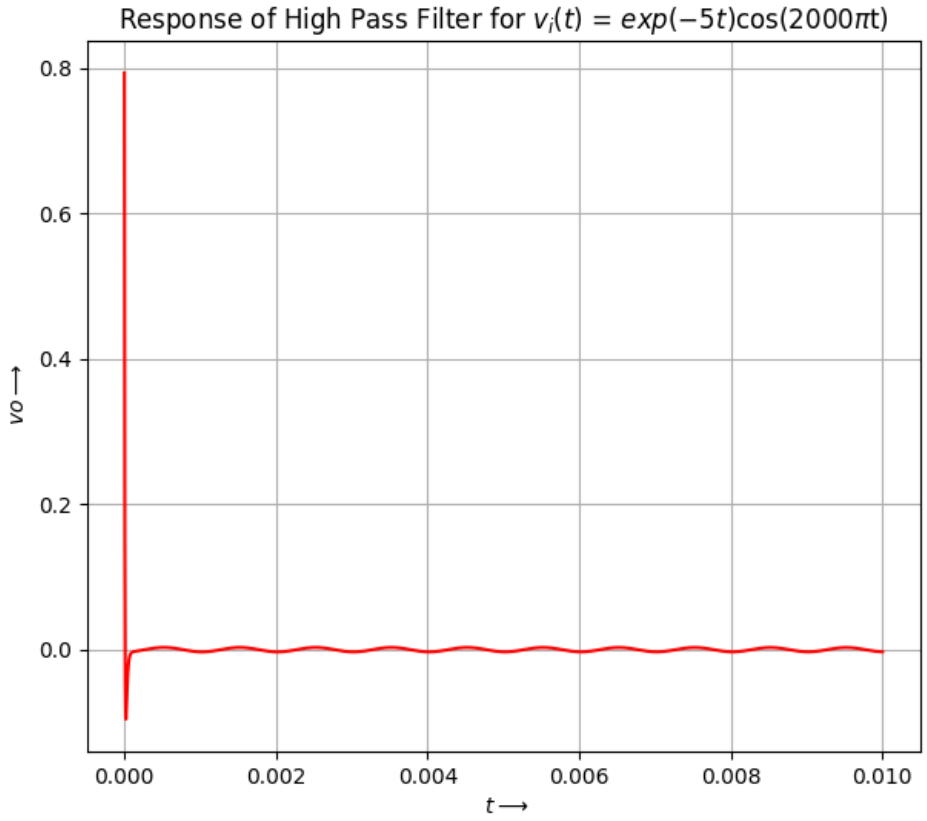

Figure 11: Output $v_o(t)$ is attenuated

Let's consider another signal,

$$v_i(t) = e^{-5000t} cos(2 \times 10^6 \pi t) u_o(t) \tag{4}$$

The above signal has a larger decay factor and oscillates at 1MHz frequency. When passed through the High Pass filter, the signal remains almost unchanged. Below is the plot of the output signal.

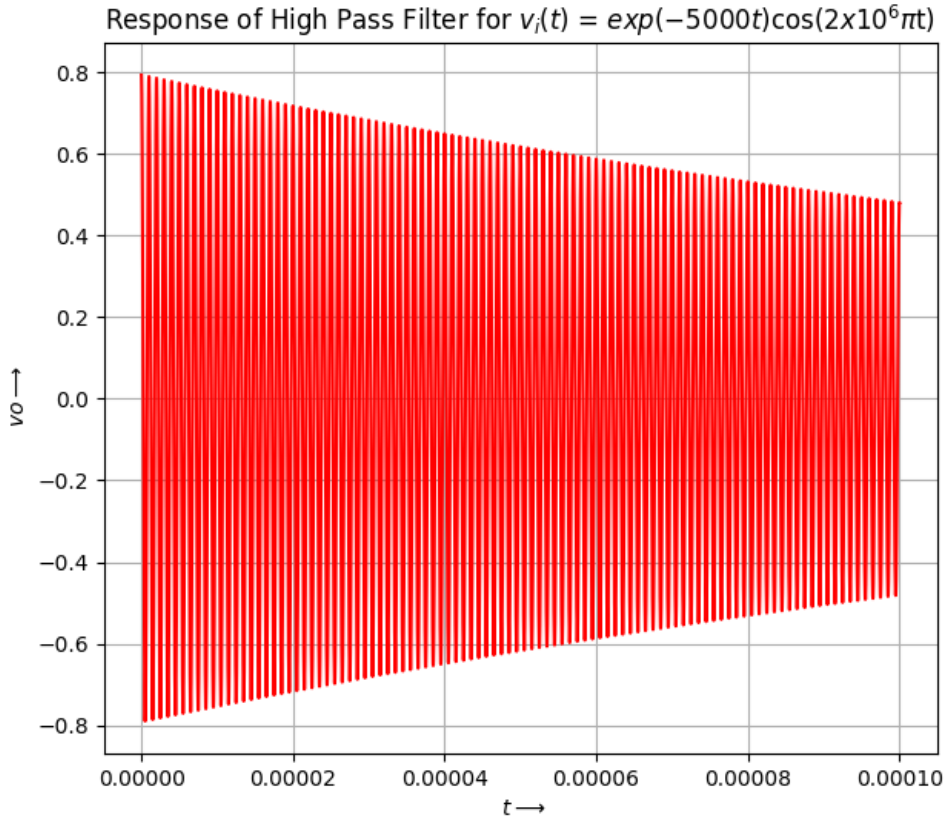

Figure 12: Output $v_o(t)$ is plotted for 100 cycles

2.4.1 Step Response (Question 5)

Using the function defined in section 1.3, we can find the step response of the High Pass Filter too.

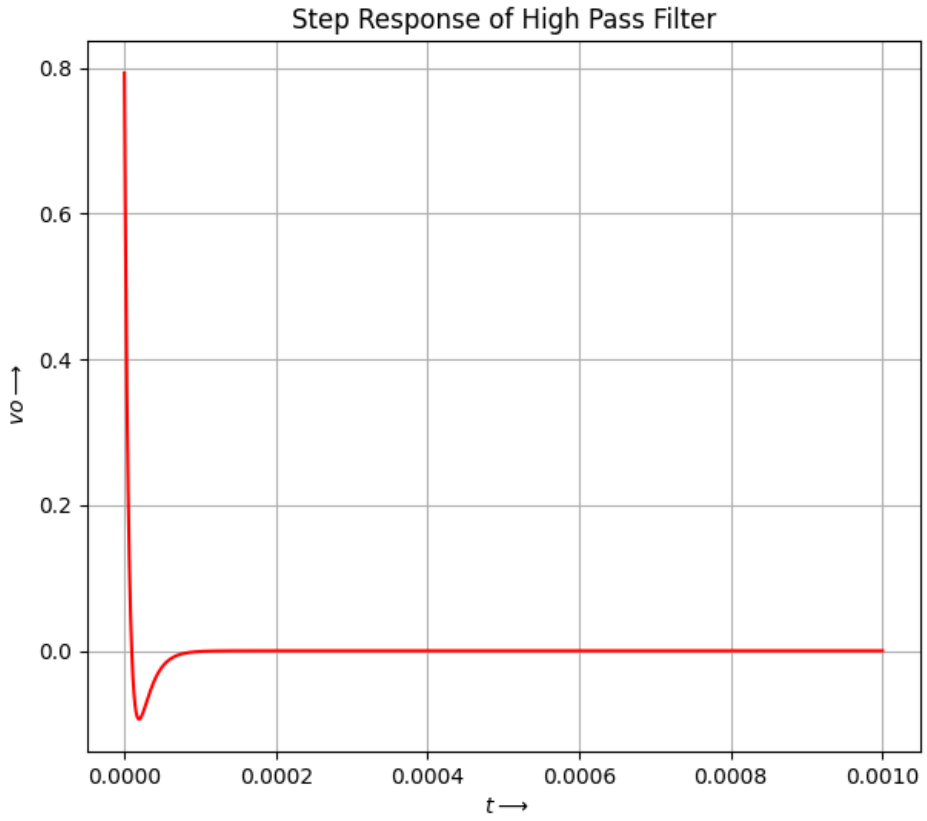

Figure 13: Step Response of High Pass Filter

The unit step response is high at t=0 due to abrupt change in input, and after some transient it settles to a steady state zero voltage. The input is a constant dc source after t=0, hence there is no oscillation in input after start at t=0. Hence, the output settles to zero after some transient time, as dc input is blocked by the High pass filter.

Conclusion

By using sympy and signals tool box, circuit analysis becomes a simpler problem to solve.