EE2703 : Applied Programming Lab
Assignment 8

Potta Muni Asheesh
EE19B048

May 23, 2021

1 Introduction

In this assignment, circuit analysis using Laplace transforms is done using `sympy` as a tool for symbolic algebra.

2 Low pass filter

The following low pass filter is considered.

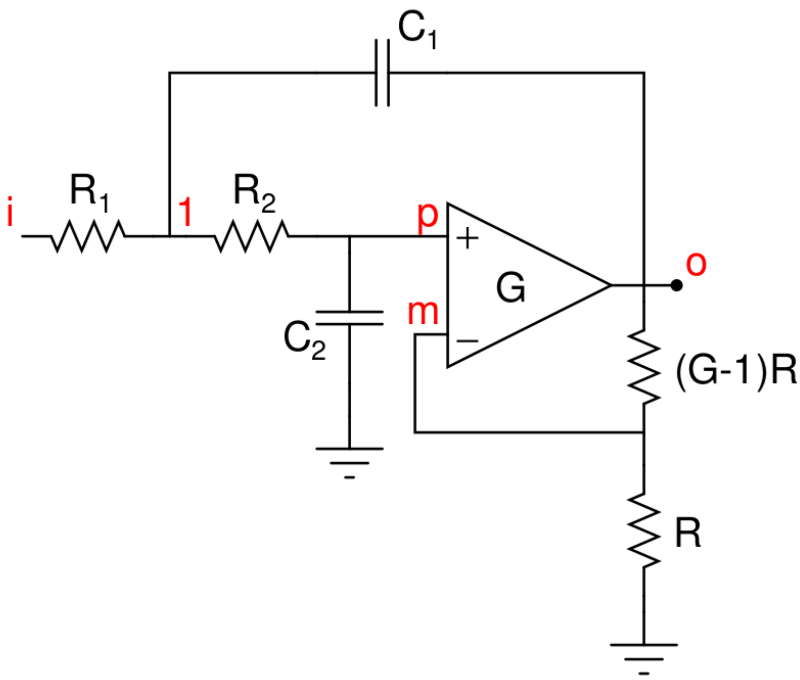

Figure 1: Circuit diagram of lowpass filter

where, the Op-Amp is ideal and $G = 1.586$, $R_1 = R_2 = 10k\Omega$ and $C_1 = C_2 = 10pF$. This gives a second order Butter-worth filter with cutoff frequency of $10/2\pi$MHz. As the op-amp is considered ideal, the circiut equations are

$$V_m = \frac{V_o}{G} \tag{1}$$

$$V_p = \frac{V_1}{1 + sC_2R_2} \tag{2}$$

$$V_p = V_m \tag{3}$$

$$\frac{V_i - V_1}{R_1} + \frac{V_p - V_1}{R_2} + sC_1(V_o - V_i) = 0 \tag{4}$$

These equations upon solving give the V_i to V_o transfer function as

$$\frac{V_o}{V_i} = \frac{1.586}{1 + \frac{1.414s}{10^7} + \frac{s^2}{10^{14}}}$$

This circuit analysis is done in python using sympy module.

The four circuit equations are solved by rewriting them as a matrix equation and using sympy to solve the matrix equation.

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

A function is defined in python takes, the circuit parameters and Laplace transform of input signal, as arguments and returns the matrix as `A`, the input vector as `b` and the voltage vector as `V`.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sp
import sympy as sym

def lowpass(R1,R2,C1,C2,G,Vi):
    s = sym.symbols('s')
    A = sym.Matrix([[0,0,1,-1/G],[1/(1 + s*C2*R2),-1,0,0],[0,1,-1,0],[1/
    ↪ R1+1/R2+s*C1,-1/R2, 0, -s*C1]])
    b = sym.Matrix([0,0,0,Vi/R1])
    V = A.inv()*b
    return (A, b, V)
```

The Frequency response of the low pass filter as obtained by extracting the coefficients of the numerator and the denominator of the rational expression for V_o in the array `V`. This is done as shown below. The frequency response obtained is shown below. The DC gain of the filter is $20log(G) = 4dB$.

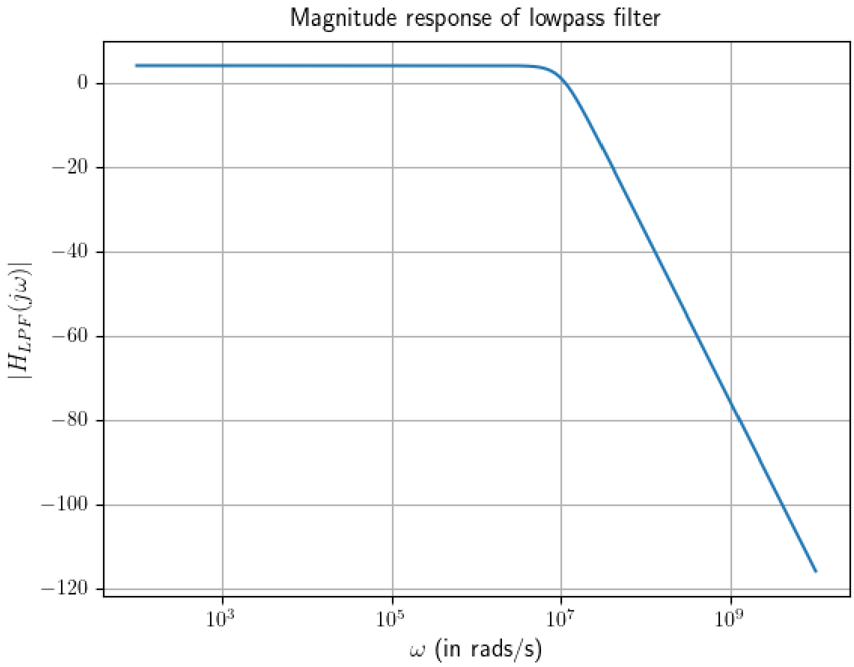

Figure 2: Magnitude response of lowpass filter

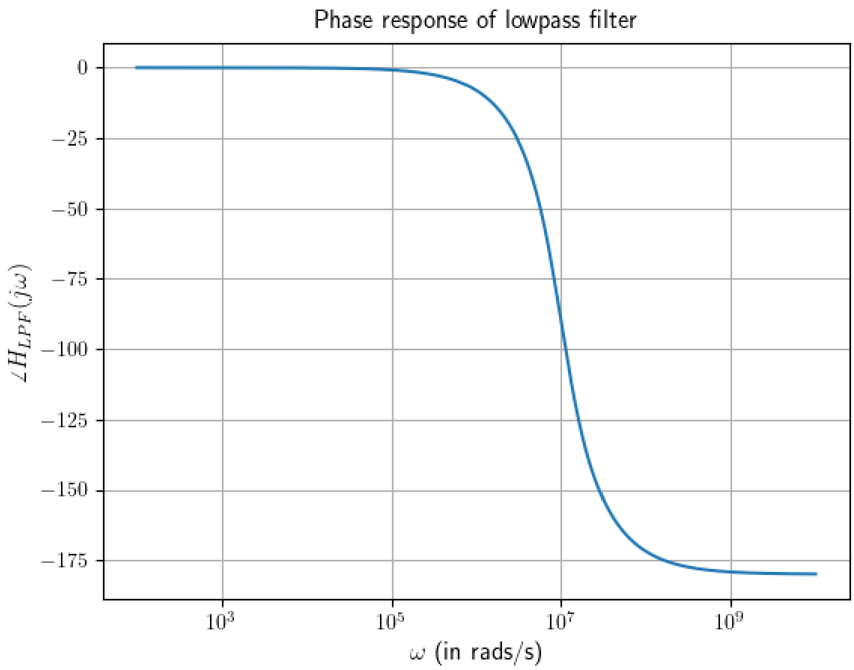

Figure 3: Phase response of lowpass filter

```
def freq_resp_lpf():
    s = sym.symbols('s')
```

```
    A, b, V = lowpass(10000, 10000, 1e-11, 1e-11, 1.586, 1)
    Vo = V[3]
    # print(Vo)
    num , den = Vo.as_numer_denom()
    num, den = np.array(sym.Poly(num, s).all_coeffs(), dtype=float), np.
   ↪ array(sym.Poly(den, s).all_coeffs(), dtype=float)
    # num, den = [1.586], [1e-10, 1.414e-5, 1]
    H = sp.lti(num, den)

    global fignum

    plt.figure(fignum)
    fignum += 1
    w = np.logspace(2,10,801)
    w, mag, phi = H.bode(w)
    plt.semilogx(w, mag)
    plt.grid(True)
    plt.title(r'Magnitude response of lowpass filter')
    plt.xlabel(r'$\omega$ (in rads/s)', size=12)
    plt.ylabel(r'$|H_{LPF}(j\omega)|$', size=12)

    plt.figure(fignum)
    fignum += 1
    plt.semilogx(w, phi)
    plt.grid(True)
    plt.title(r'Phase response of lowpass filter')
    plt.xlabel(r'$\omega$ (in rads/s)', size=12)
    plt.ylabel(r'$\angle H_{LPF}(j\omega)$', size=12)
```

To get the step response $s(t)$, simply giving $V_i(s) = 1/s$ and using `impulse` to get the time domain output voltage $v_o(t)$.

```
def q1():
    s = sym.symbols('s')
    A, b, V = lowpass(10000, 10000, 1e-11, 1e-11, 1.586, 1/s)
    Vo = V[3]
    num , den = Vo.as_numer_denom()
    num, den = np.array(sym.Poly(num, s).all_coeffs(), dtype=float), np.
   ↪ array(sym.Poly(den, s).all_coeffs(), dtype=float)
    H = sp.lti(num, den)

    t = np.arange(0, 1e-5, 1e-8)
    t, y = sp.impulse(H, None, t)
    global fignum
    plt.figure(fignum)
    fignum += 1
    plt.plot(t, y)
```

```
    plt.title(r'Step response ($s(t)$) of lowpass filter')
    plt.xlabel(r"time $t$ (in seeconds)", size=12)
    plt.ylabel(r'$s(t)$', size=12)
    plt.grid(True)
```

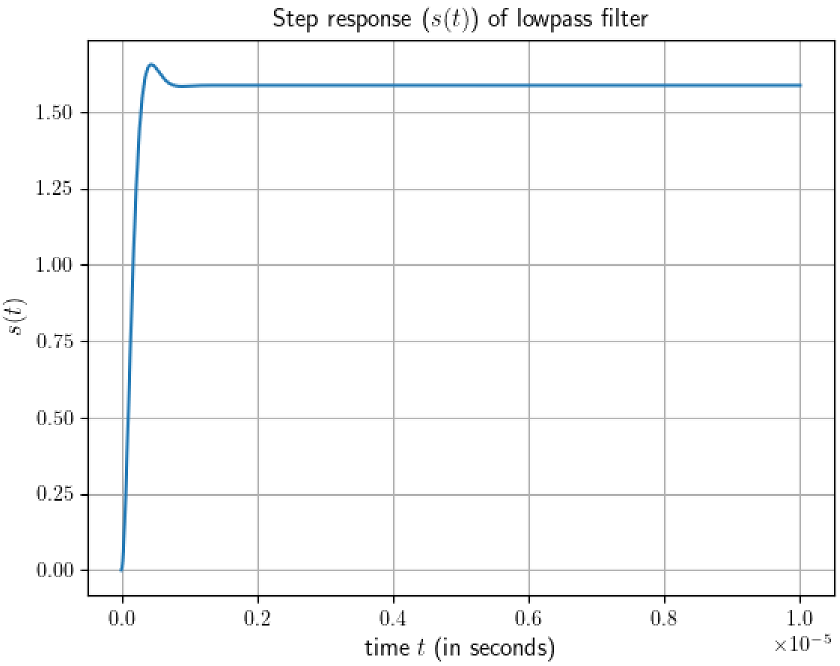

Figure 4: Step response of lowpass filter

3 High pass filter

The following circuit is considered.

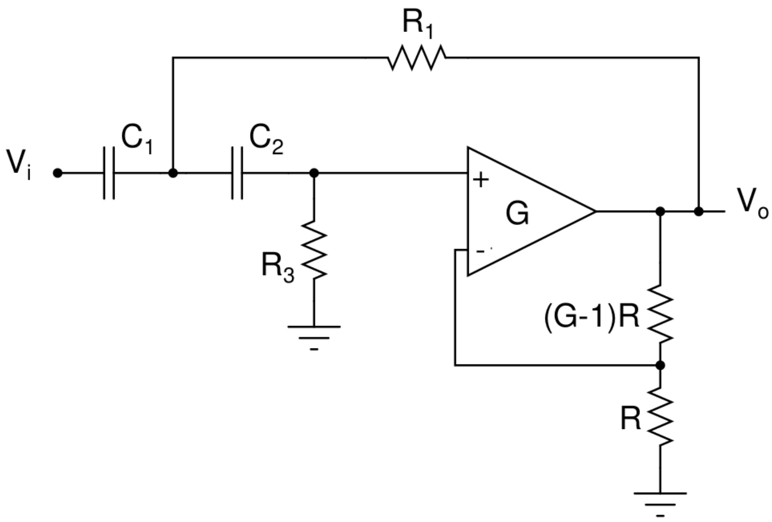

Figure 5: Circuit diagram of high pass filter

where, the Op-Amp is ideal and $G = 1.586$, $R_1 = R_3 = 10k\Omega$ and $C_1 = C_2 = 1nF$. This gives a second order Butter-worth filter with cutoff frequency of $1/20\pi$MHz.

As the op-amp is considered ideal, the circuit equations are

$$V_m = \frac{V_o}{G} \tag{5}$$

$$V_p = V_1 \frac{sC_2R_3}{1 + sC_2R_3} \tag{6}$$

$$V_p = V_m \tag{7}$$

$$(V_i - V_1)sC_1 + (V_p - V_1)sC_2 + \frac{V_o - V_i}{R_1} = 0 \tag{8}$$

These equations upon solving give the V_i to V_o transfer function as

$$\frac{V_o}{V_i} = 1.586 \frac{\frac{s}{10^{10}}}{1 + \frac{1.414s}{10^5} + \frac{s^2}{10^{10}}}$$

A similar function for highpass filter is also defined.

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

When the following input voltage is given at V_i

$$v_i(t) = (sin(2000\pi t) + cos(2 \times 10^6 t))u(t)$$

The transfer function is extracted using sympy and `lsim` is used to get the output voltage $v_o(t)$. This is done as shown below.

```
def q2():
    s = sym.symbols('s')
    A, b, V = highpass(10000, 10000, 1e-9, 1e-9, 1.586, 1)
    Vo = V[3]
    # print(Vo)
    num , den = Vo.as_numer_denom()
    num, den = np.array(sym.Poly(num, s).all_coeffs(), dtype=float), np.
    ↪ array(sym.Poly(den, s).all_coeffs(), dtype=float)
    H = sp.lti(num, den)

    t = np.arange(0, 10e-3, 1e-7)
    vi = np.sin(2*np.pi*1e3*t) + np.cos(2*np.pi*1e6*t)
    t, vo = sp.lsim(H, vi, t)[:2]
```

The plots $v_i(t)$ and $v_o(t)$ for $t < 10\mu s$ and $t < 1ms$ are given below.

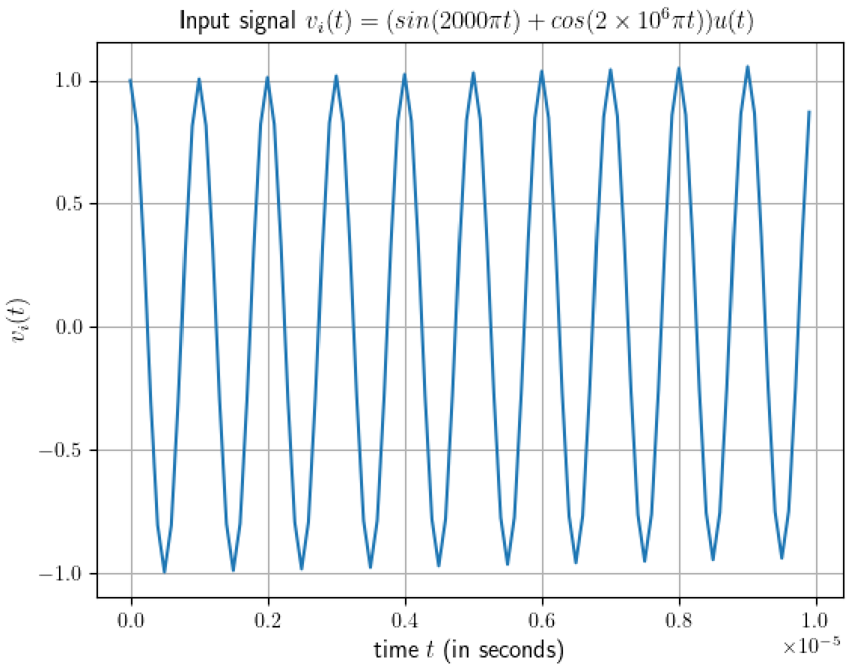

Figure 6: Input voltage for $t < 10\mu s$

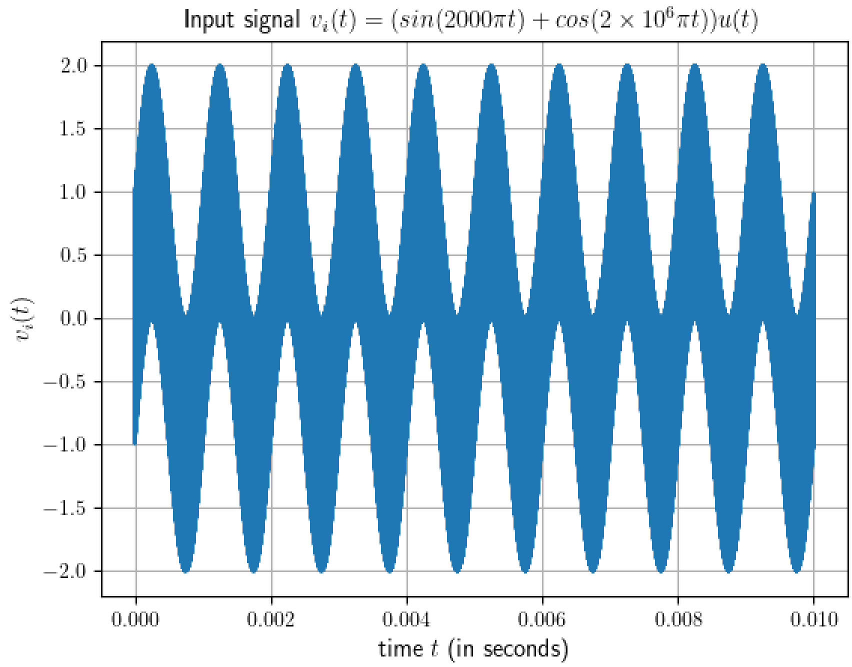

Figure 7: Input voltage for $t < 1ms$

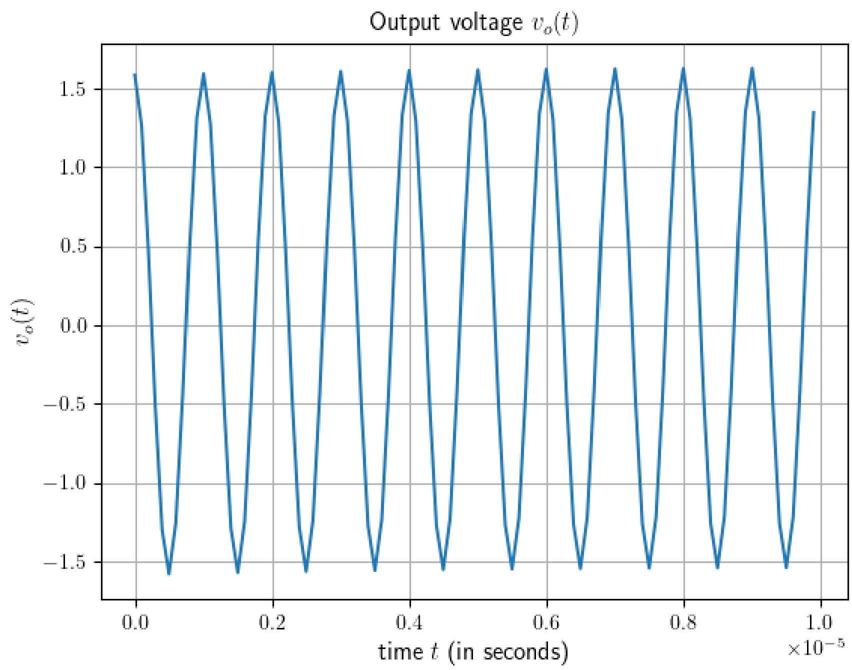

Figure 8: Output voltage for $t < 10\mu s$

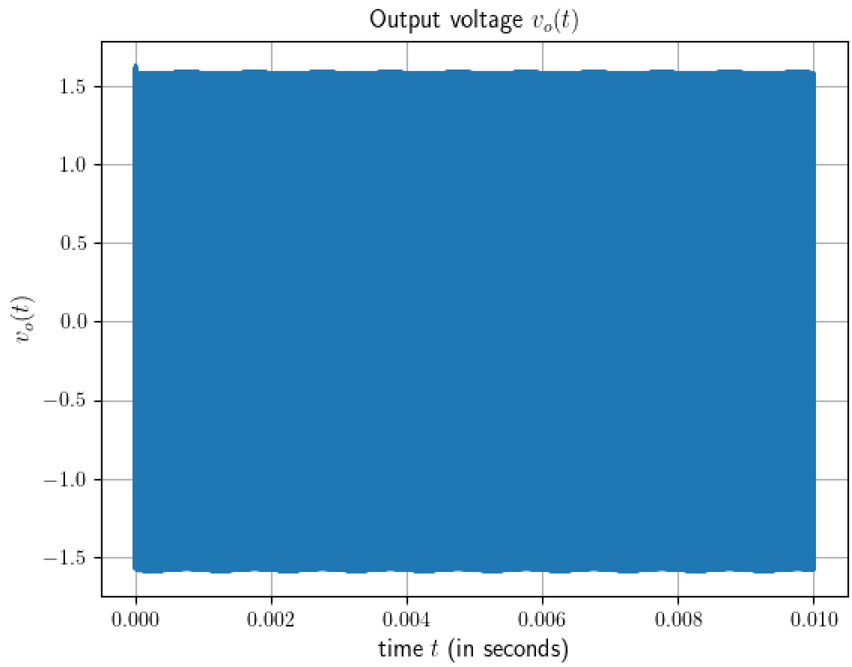

Figure 9: Output voltage for $t < 1ms$

It can be seen clearly that the lower frequency $\omega = 2000\pi$ component of v_i is attenuated and the higher frequency $\omega = 2\pi \times 10^6$ component is amplified by 1.586 times.

The coefficients of the transfer function obtained using sympy are used to create the system transfer function using `lti` from `scipy.signal`.

From this, the frequency response of the system can be obtained. The plots of the frequency response are shown below.

```
def q3():
    s = sym.symbols('s')
    A, b, V = highpass(10000, 10000, 1e-9, 1e-9, 1.586, 1)
    Vo = V[3]
    # print(Vo)
    num , den = Vo.as_numer_denom()
    num, den = np.array(sym.Poly(num, s).all_coeffs(), dtype=float), np.
    ↪ array(sym.Poly(den, s).all_coeffs(), dtype=float)
    H = sp.lti(num, den)

    w = np.logspace(0,8,801)
    w, mag, phi = H.bode(w)
```

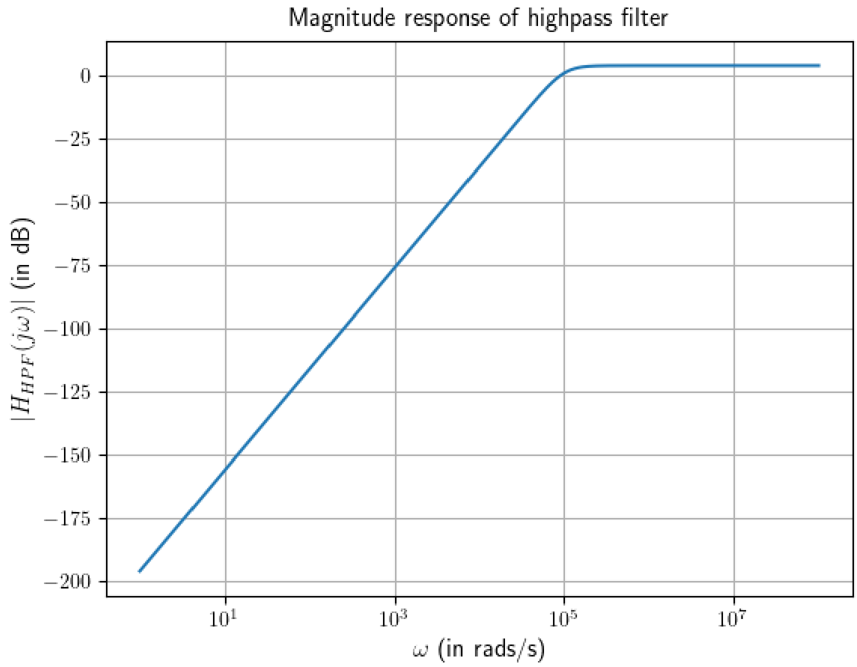

Figure 10: Magnitude response of high pass filter

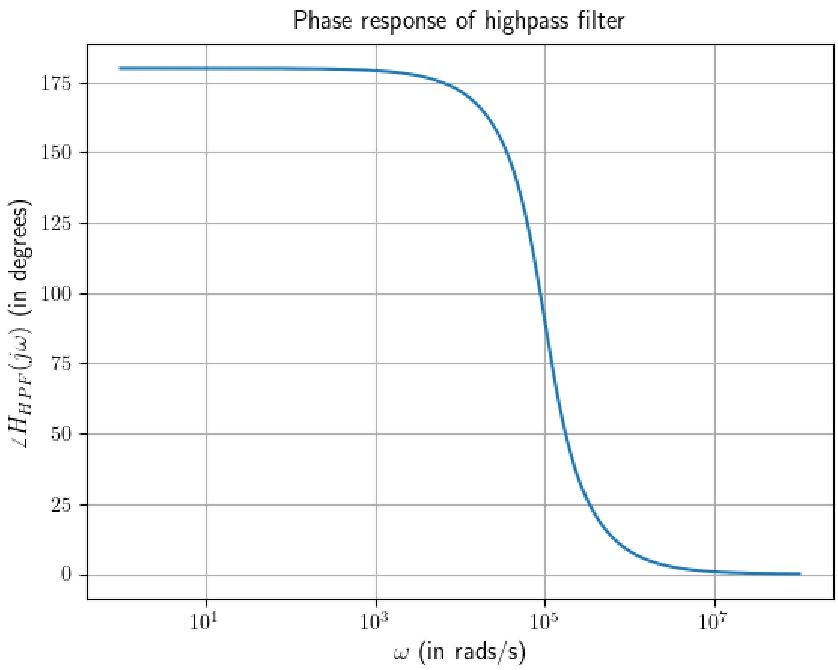

Figure 11: Phase response of high pass filter

Consider the input voltage $v_i(t) = exp(-10^4 t)cos(10^7 t)$, then it's laplace transform

$V_i(s)$ is given as

$$V_i(s) = \frac{s + 10^4}{(s + 10^4)^2 + 10^{14}}$$

This is given as input to the function `highpass` and the laplace transform of output voltage $v_o(t)$ is obtained. `impulse` command is used to obtain the time domain signal.

```
def q4():
    s = sym.symbols('s')
    a, w0 = 1e4, 1e7
    Vi = (s+a)/((s+a)**2 + w0**2)
    A, b, V = highpass(10000, 10000, 1e-9, 1e-9, 1.586, Vi)
    Vo = V[3]
    # print(Vo)
    Vi_TF = sp.lti([1, a], [1, 2*a, w0**2 + a**2])
    num , den = Vo.as_numer_denom()
    num, den = np.array(sym.Poly(num, s).all_coeffs(), dtype=float), np.
    ↪ array(sym.Poly(den, s).all_coeffs(), dtype=float)
    H = sp.lti(num, den)

    t = np.arange(0, 1e-4, 1e-8)
    vi = np.exp(-a*t)*np.cos(w0*t)
    t, h = sp.impulse(H, None, t)
```

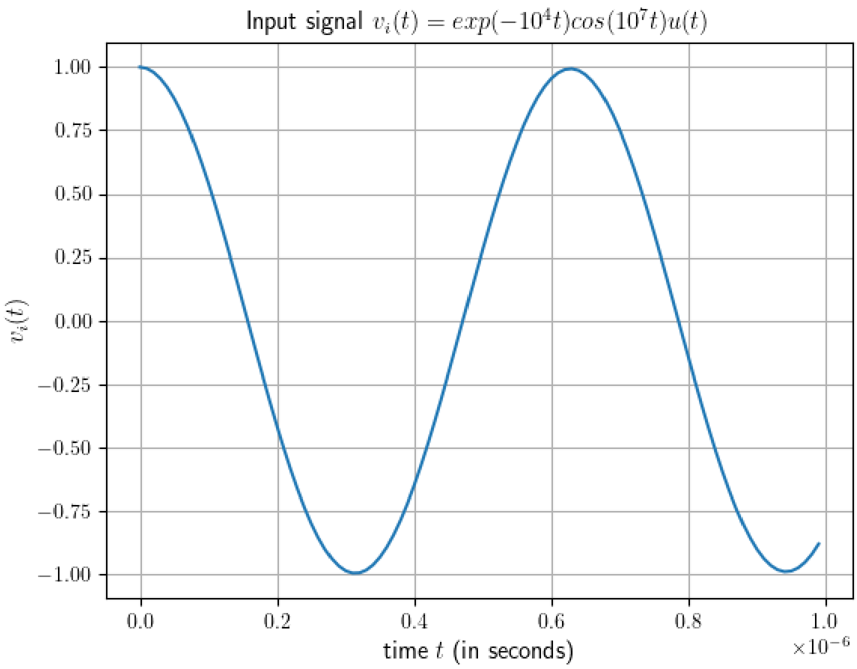

Figure 12: Input volatge for $t < 1\mu s$

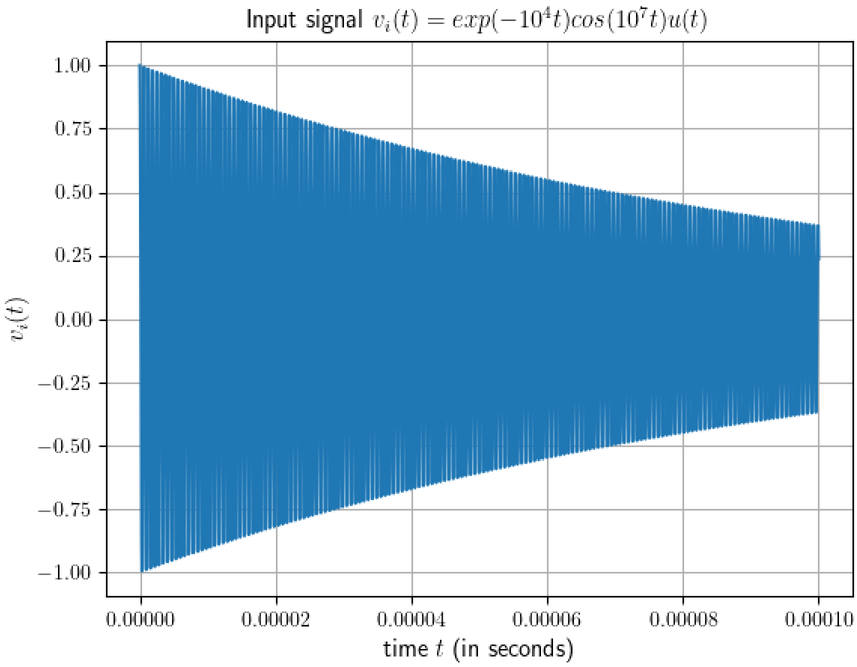

Figure 13: Input volatge for $t < 100\mu s$

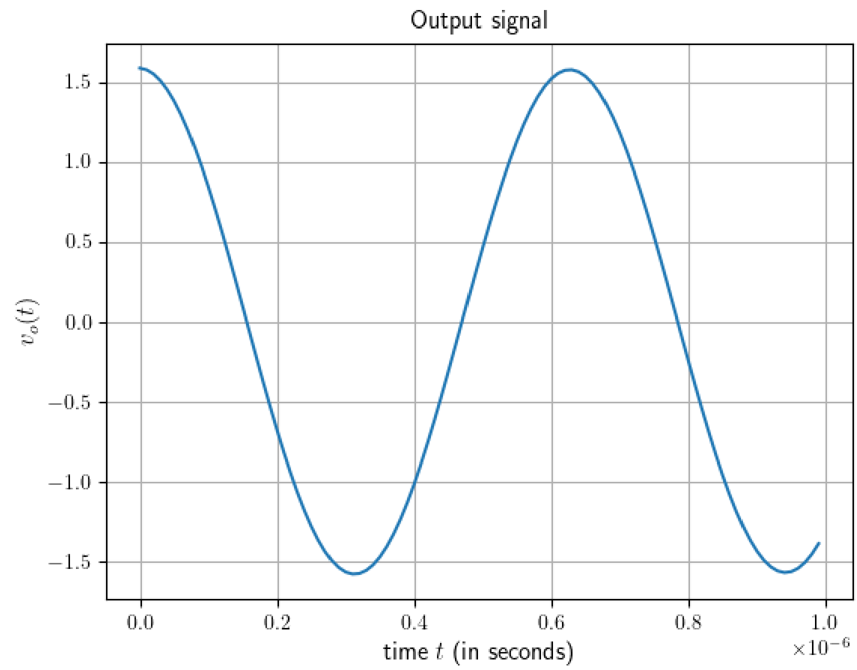

Figure 14: Output volatge for $t < 1\mu s$

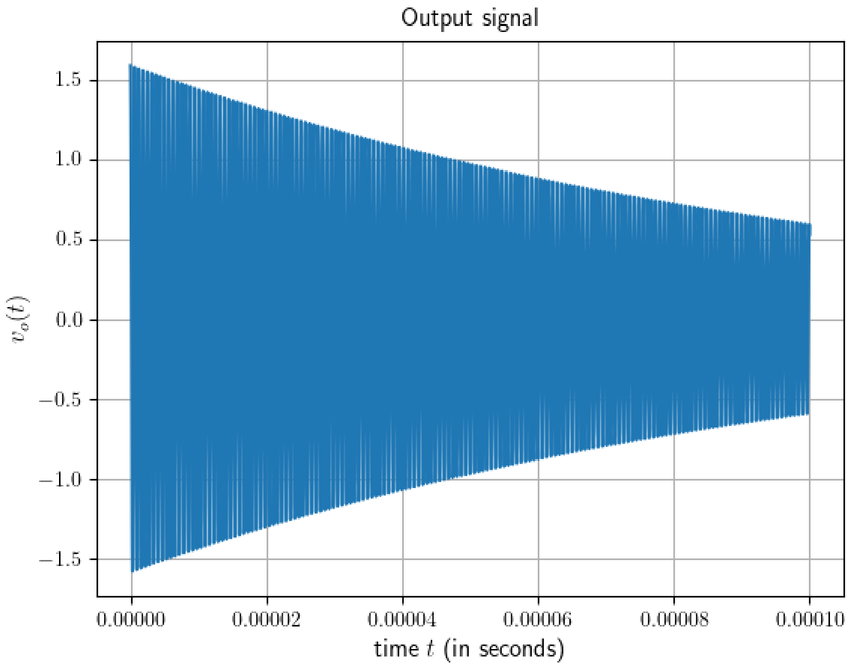

Figure 15: Output volatge for $t < 100\mu s$

It can be seen that the signal is amplified by 1.586 times.

When the input voltage given is $v_i(t) = exp(-10t)cos(10^3t)$, then the laplace transform $V_i(s)$ is given by

$$V_i(s) = \frac{s + 10}{(s + 10)^2 + 10^6}$$

The output voltage signal is obtained similarly.

```
def q4():
    s = sym.symbols('s')
    a, w0 = 1e1, 1e3
    Vi = (s+a)/((s+a)**2 + w0**2)
    A, b, V = highpass(10000, 10000, 1e-9, 1e-9, 1.586, Vi)
    Vo = V[3]
    # print(Vo)
    Vi_TF = sp.lti([1, a], [1, 2*a, w0**2 + a**2])
    num , den = Vo.as_numer_denom()
    num, den = np.array(sym.Poly(num, s).all_coeffs(), dtype=float), np.
    ↪ array(sym.Poly(den, s).all_coeffs(), dtype=float)
    H = sp.lti(num, den)

    t = np.arange(0, 1, 1e-4)
    vi = np.exp(-a*t)*np.cos(w0*t)
    t, h = sp.impulse(H, None, t)
```

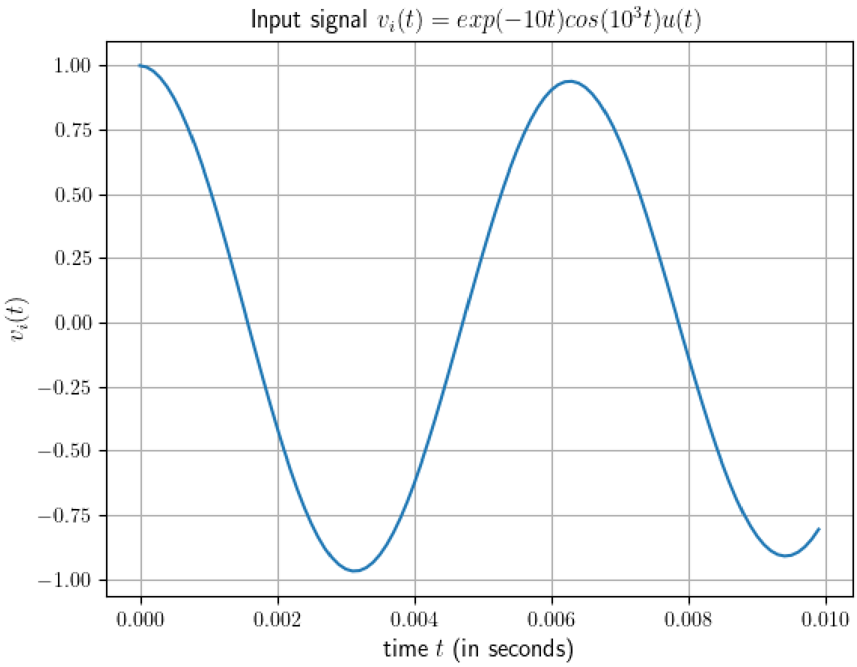

Figure 16: Input volatge for $t < 10ms$

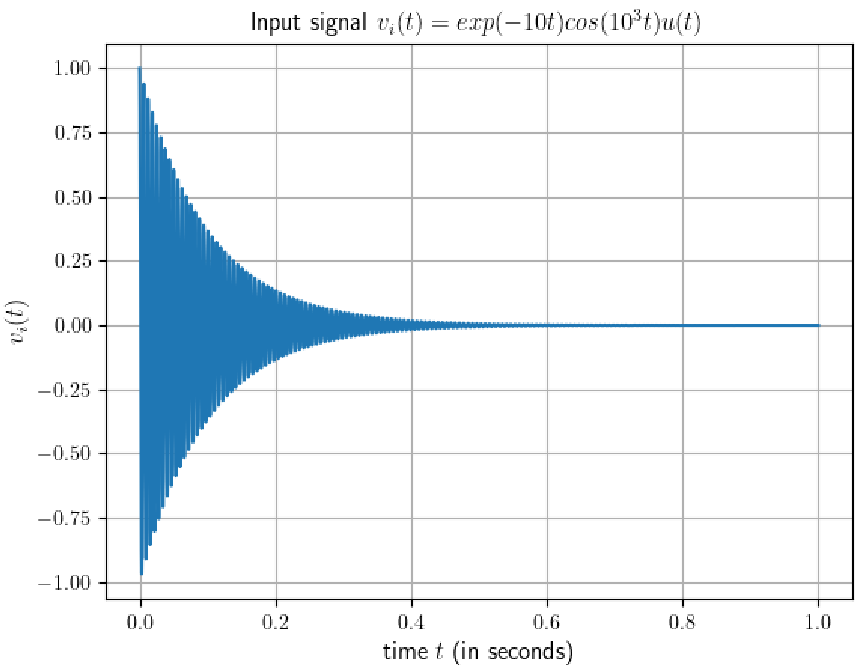

Figure 17: Input volatge for $t < 1s$

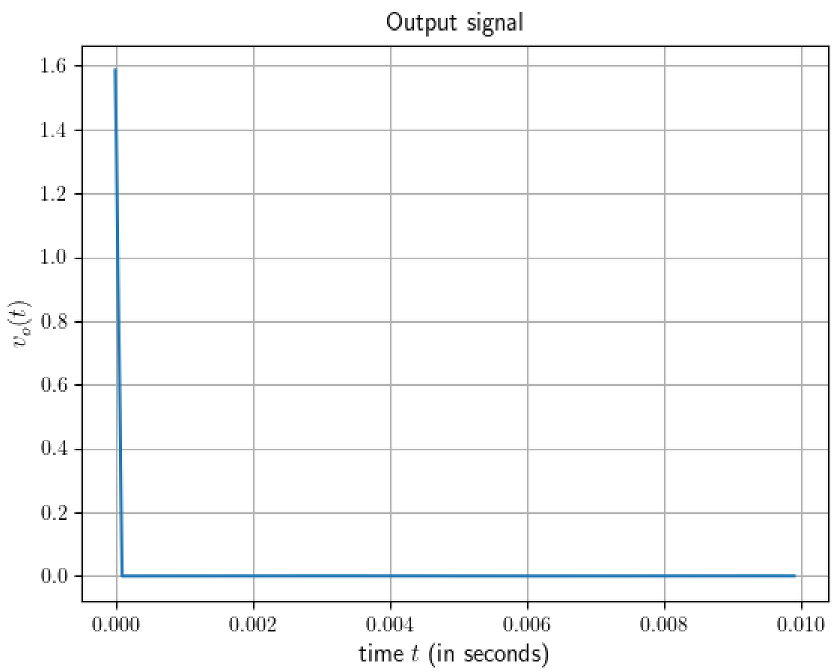

Figure 18: Output volatge for $t < 10ms$

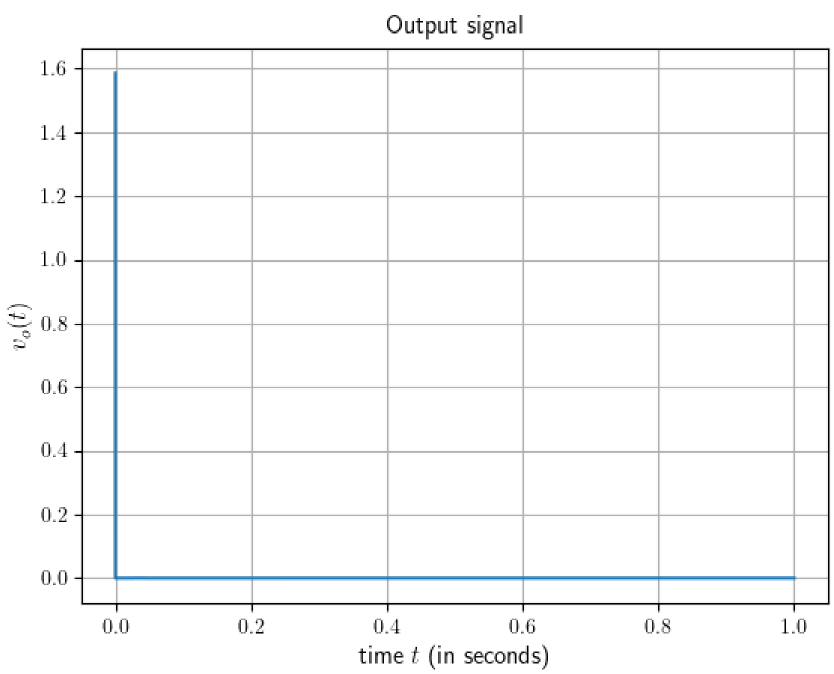

Figure 19: Output volatge for $t < 1s$

It can be seen that the signal is attenuated.

Similarly, to get the step response, the laplace transform of input is given as $V_i(s) = 1/s$ to the function `highpass` to get the laplace transform of output voltage signal.

```
def q5():
    s = sym.symbols('s')
    A, b, V = highpass(10000, 10000, 1e-9, 1e-9, 1.586, 1/s)
    Vo = V[3]
    # print(Vi)
    # Vi_TF = sp.lti([1, 0.01],[1, 2e-2, 1e7])
    num , den = Vo.as_numer_denom()
    num, den = np.array(sym.Poly(num, s).all_coeffs(), dtype=float), np.
    ↪ array(sym.Poly(den, s).all_coeffs(), dtype=float)
    H = sp.lti(num, den)

    t = np.arange(0, 1e-3, 1e-7)
    t, h = sp.impulse(H, None, t)
```

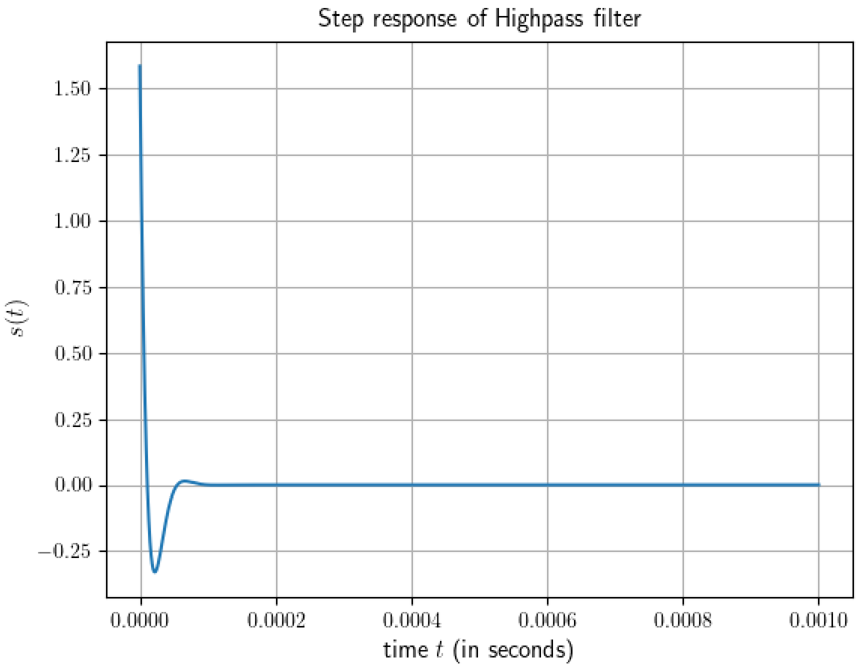

Figure 20: Step response of high pass filter