EE2703: Applied Programming Lab
Assignment 7
Analysis of Circuits using Laplace Transforms

Kaushik Ravibaskar
EE20B057

April 4, 2022

Contents

1 Aim **2**

2 Introduction **2**
2.1 The Low-Pass Filter 2
2.2 The High-Pass Filter 3

3 The Assignment Problems **3**
3.1 P1: Obtaining the Step Response of the Butterworth Filter . 3
3.2 P2: Obtaining Sinusoidal Response of the Butterworth Filter 6
3.3 P3: Analysing the High Pass Filter 7
3.4 P4: Obtaining Response of Filters for Damped Sinusoids . . . 9
3.5 P5: Obtaining the Step Response for the High Pass Filter . . 14

4 Conclusion **14**

1 Aim

The assignment mainly deals with the following points:

- To utilize the **sympy** module in Python to represent the circuit analysis equations in a meaningful manner and to solve them accordingly.
- To convert the **sympy** functions into Python compatible functions so as to plot their time variations using **scipy.signal** module functions.
- To understand the behaviour of the low-pass and high-pass filters given in the assignment by noting their response to unit step and damped sinusoidal inputs.
- To plot all the signals obtained using **pylab** module and to analyse the same.

2 Introduction

Circuit analysis is inevitable in Electrical Engineering domain. In Electrical Engineering, we deal with a wide variety of analog circuits, some of which act like filters. Two basic kinds of filters are **low-pass** and **high-pass** filters. In this assignment, we deal with both of these filters, will plot the magnitude response of their transfer function in frequency domain, and understand their behaviour to different types of inputs.

2.1 The Low-Pass Filter

In this assignment, we are given a **3dB Butterworth Filter**, which is a low-pass filter. Technically what it does is, it allows low frequency inputs to pass and blocks or diminishes high frequency inputs. The circuit diagram is given in the question. The equations (in Laplace Domain) pertaining to the circuit given is as follows:

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

$$V_p = \frac{V_1}{(1 + sR_2C_2)}$$

$$V_o = G(V_p - V_m)$$

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

The Matrix representation for the same is as follows:

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

The details regarding how it will be expressed using **sympy** module and analysed further, will be taken up in subsequent sections.

2.2 The High-Pass Filter

In one of the problems in the assignment, we are given a high-pass filter. Technically what it does is, it allows high frequency inputs to pass through and blocks or diminishes low frequency inputs. The circuit diagram is given in the question. The equations (in Laplace Domain) pertaining to the circuit given is as follows:

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

$$V_p = \frac{V_1 sC_2 R_3}{(1 + sR_3C_2)}$$

$$V_o = G(V_p - V_m)$$

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

The Matrix representation for the same is as follows:

$$\begin{pmatrix} 0 & 0 & 1 & \frac{-1}{G} \\ \frac{sR_3C_2}{1+sR_3C_2} & 0 & -1 & 0 \\ 0 & -G & G & 1 \\ -\frac{1}{R_1} - sC_2 - sC_1 & sC_2 & 0 & \frac{1}{R_1} \end{pmatrix} \begin{pmatrix} V_1(s) \\ V_p(s) \\ V_m(s) \\ V_o(s) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -sC_1V_i(s) \end{pmatrix} \tag{2}$$

The details regarding how it will be expressed using **sympy** module and analysed further, will be taken up in subsequent sections.

3 The Assignment Problems

3.1 P1: Obtaining the Step Response of the Butterworth Filter

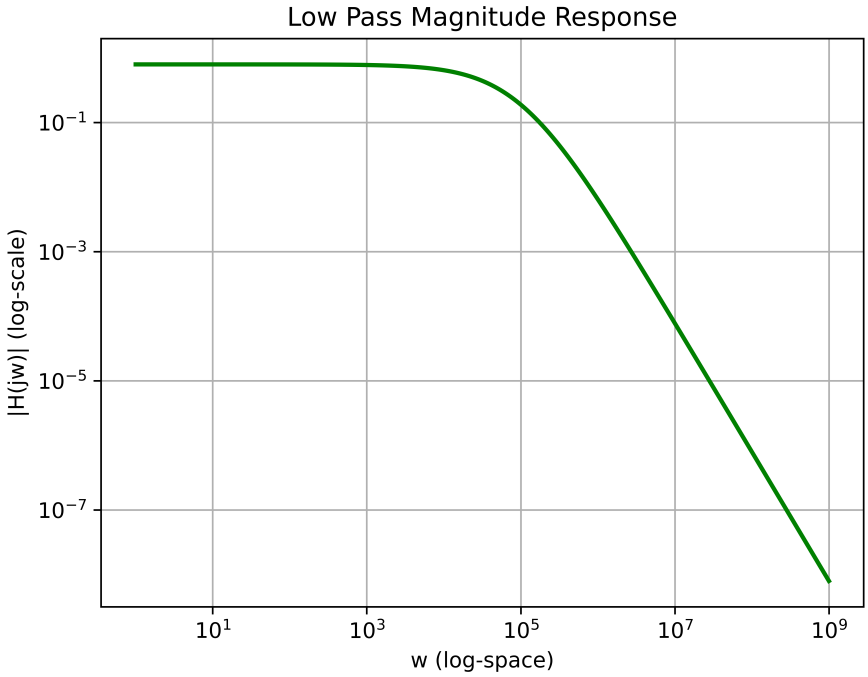
Firstly, we will plot the magnitude response in frequency domain of the transfer function of the filter. For that, we can define a **low-pass** function which solves the matrix mentioned above in a symbolic manner. The function defined is as follows:

```
#defining the low-pass filter function
def lowpass(R1, R2, C1, C2, G, Vi):
    t = 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]])
    o = sy.Matrix([0, 0, 0, -Vi/R1])
    x = t.inv()*o
    return (t, o, x)
```

We will give the data given in the problem to the function and make $V_i(s) = 1$ in the function. It returns the vector x, which is the solution vector. The 4^{th} entry in the vector gives the transfer function $H(s)$. We will then use the **lambdify** function in the **sympy** module to make the transfer function compatible with python. The code that does this all is shown as follows:

```
#given data
r1_1 = 10000
r1_2 = 10000
c1_1 = 1e-9
c1_2 = 1e-9
G1 = 1.586

#obtaining magnitude response
(t_1, o_1, h_1) = lowpass(r1_1, r1_2, c1_1, c1_2, G1, 1)
h_1l = h_1[3]
w_1 = py.logspace(0, 9, 2001)
h_1l = sy.lambdify(s, h_1l, 'numpy')
h_1l = h_1l(w_1)
```

The magnitude plot is as follows:



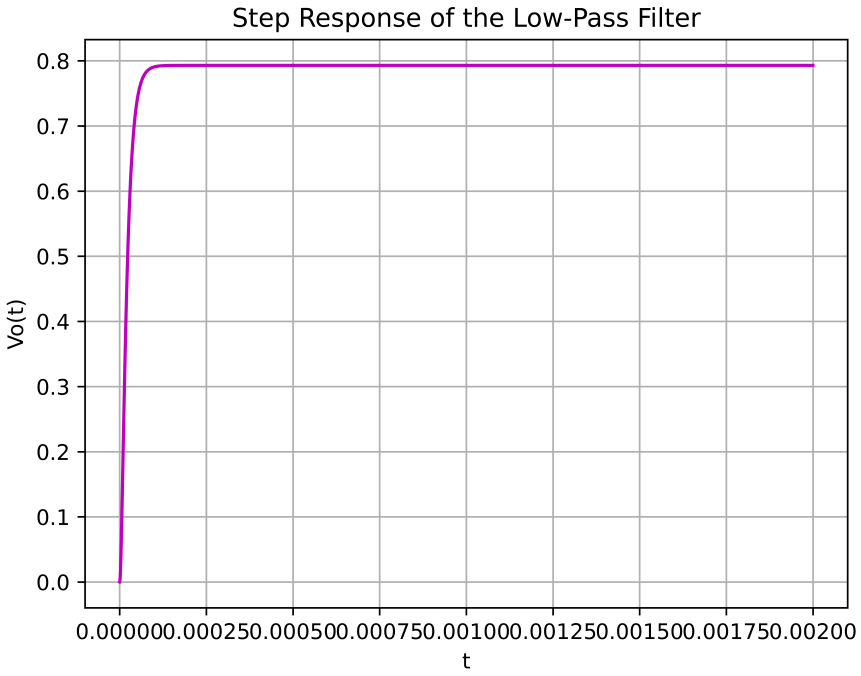
To get the step response, we will simply put $V_i(s) = 1/s$ in the function. The 4^{th} entry in the matrix will give the corresponding $V_o(s)$. Now we will define a function using **sympy** module to get the numerator and denominator coefficients of any transfer function. The function defined is as follows:

```
#defining a function to get numerator and denominator
coefficients of a symbolic expression
def sym_extract(K):
    K = sy.simplify(K)
    (num, den) = sy.fraction(K)
    num = sy.Poly(num, s)
    den = sy.Poly(den, s)
    num = num.all_coeffs()
    den = den.all_coeffs()
    num = [float(temp) for temp in num]
    den = [float(temp) for temp in den]
    return (num, den)
```

We will now pass the obtained $V_o(s)$ above into the function to get its numerator and denominator coefficients. Next we will use the **scipy.signal** module to write it in the Laplace form (non symbolic form) and correspondingly get its time response. The code which does all this is as follows:

```
#obtaining and plotting step response
(t_1, o_1, vi_1) = lowpass(r1_1, r1_2, c1_1, c1_2, G1,
1/s)
vi_1 = vi_1[3]
(num_vi_1, den_vi_1) = sym_extract(vi_1)
vi_1 = sig.lti(num_vi_1, den_vi_1)
(t1_x, vo_1) = sig.impulse(vi_1, None, py.linspace(0,
0.002, 7001))
```

The plot for the step response is as follows:



3.2 P2: Obtaining Sinusoidal Response of the Butterworth Filter

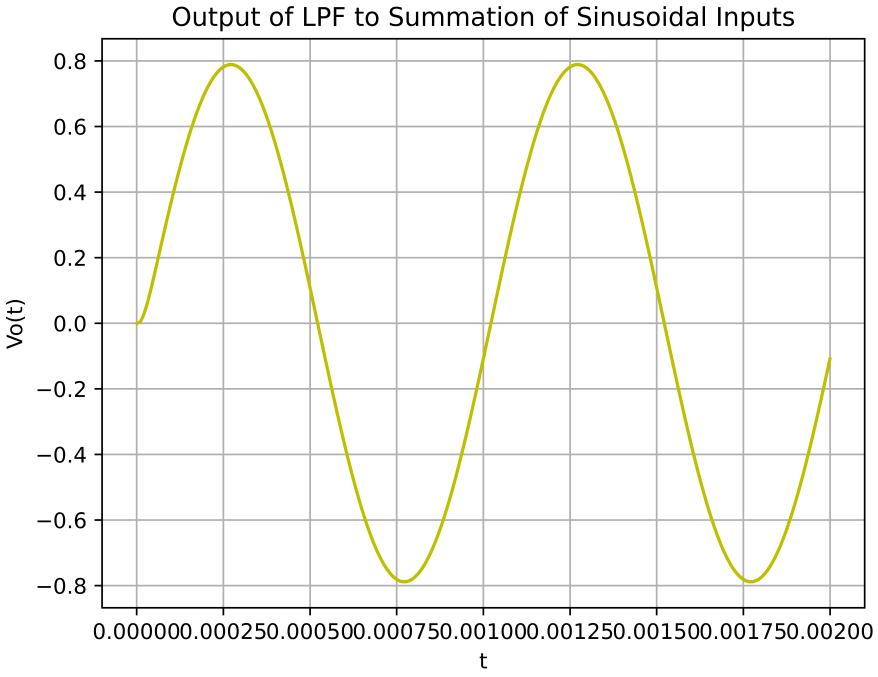
We are given the input signal as follows:

$$V_i(t) = (\sin(2000\pi t) + \cos(2 \times 10^6 \pi t))u_o(t)V$$

Now to find the response, we can firstly find the numerator and denominator coefficients of the transfer function of the system i.e. $H(s)$. Then we can use **signal.lsim** to get the response by passing the above input $V_i(t)$, transfer function and time array as parameters. The code that does all this is as follows:

```
#creating the transfer function in python domain
(t_1, o_1, h_1) = lowpass(r1_1, r1_2, c1_1, c1_2, G1, 1)
h_1s = h_1[3]
(num_h1s, den_h1s) = sym_extract(h_1s)
h_1s = sig.lti(num_h1s, den_h1s)

#obtaining the required output response
t2_x = py.linspace(0, 0.002, 7001)
vi2_t = py.sin(2000*pi*t2_x) + py.cos(2000000*pi*t2_x)
(t2_x, vo_2, svec) = sig.lsim(h_1s, vi2_t, t2_x)
```

The plot of the output response is as follows:



3.3 P3: Analysing the High Pass Filter

As was shown in the introduction, we had formulated the matrix for the high-pass system. The function defined for the same is as follows:

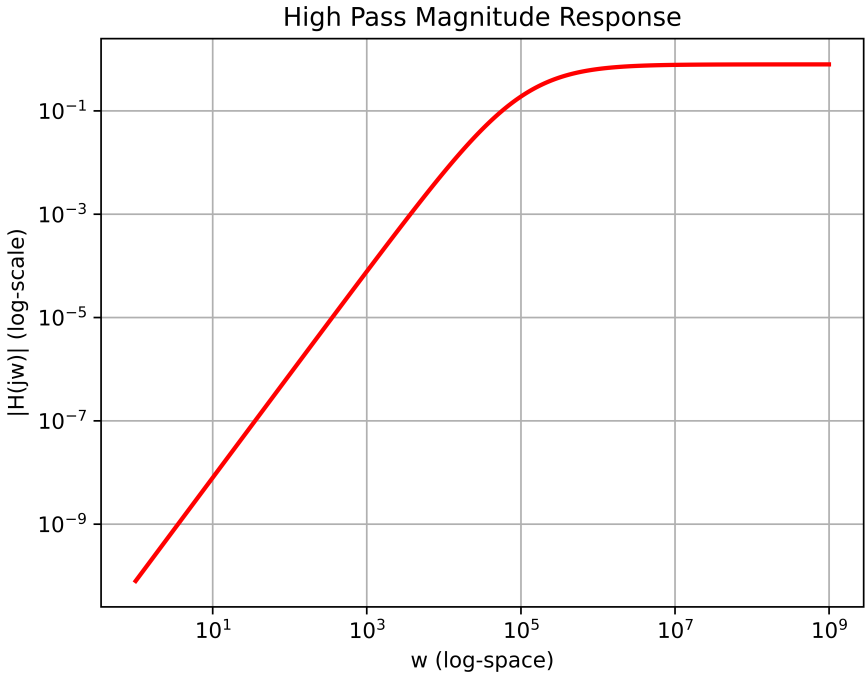
```
#defining the high-pass filter function
def highpass(R1, R3, C1, C2, G, Vi):
    t = sy.Matrix([[0, 1, 0, -1/G], [0, -G, G, -1],
    [-s*C2, 0, (s*C2 + (1/R3)), 0],  [s*(C1 + C2) +
```

```
    (1/R1), 0, -s*C2, -(1/R1)]])
    o = sy.Matrix([0, 0, 0, Vi*s*C1])
    x = t.inv()*o
    return (t, o, x)
```

Now the procedure to obtain the magnitude response in the frequency domain remains the same as for the low-pass case, the code for the same is as follows:

```
#given data
r3_1 = 10000
r3_3 = 10000
c3_1 = 1e-9
c3_2 = 1e-9
G3 = 1.586

#obtaining magnitude response
(t_3, o_3, h_3) = highpass(r3_1, r3_3, c3_1, c3_2, G3, 1)
h_3l = h_3[3]
w_3 = py.logspace(0, 9, 2001)
h_3l = sy.lambdify(s, h_3l, 'numpy')
h_3l = h_3l(w_3)
```

The plot for the magnitude response is as follows:



3.4 P4: Obtaining Response of Filters for Damped Sinusoids

We will not take two damped sinusoidal inputs, namely:

$$u_1(t) = e^{-100t}\sin(2000\pi t)$$
$$u_2(t) = e^{-50000t}\sin(2\times 10^6\pi t)$$

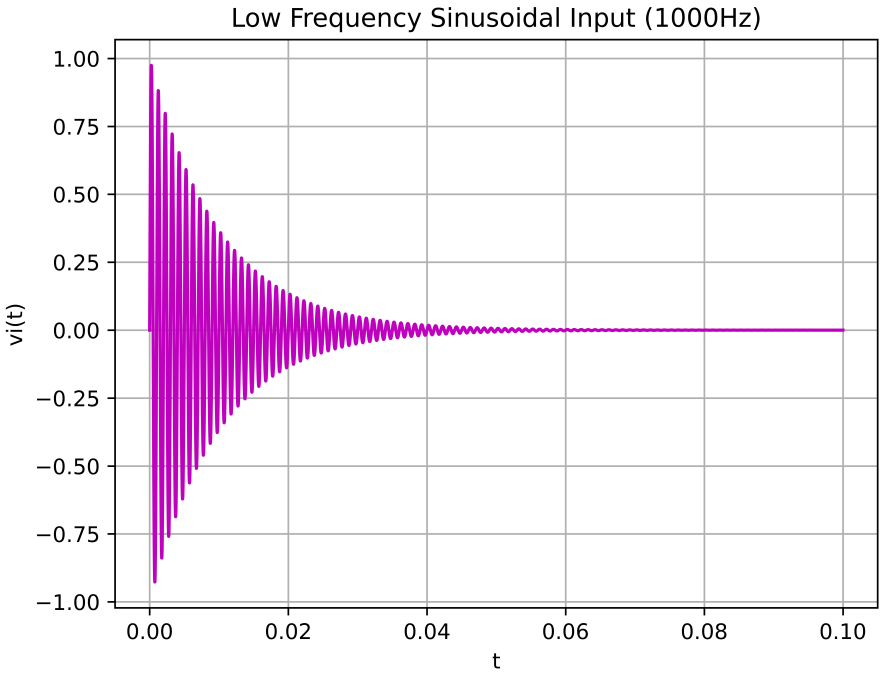
As we can see the former one is low-frequency low-damped response and the latter one is high-frequency high-damped response. We will now pass them both through the filters discussed above. The procedure to get the output is exactly the same as getting the step response for a low-pass filter discussed earlier. The code that generates the responses is as follows:

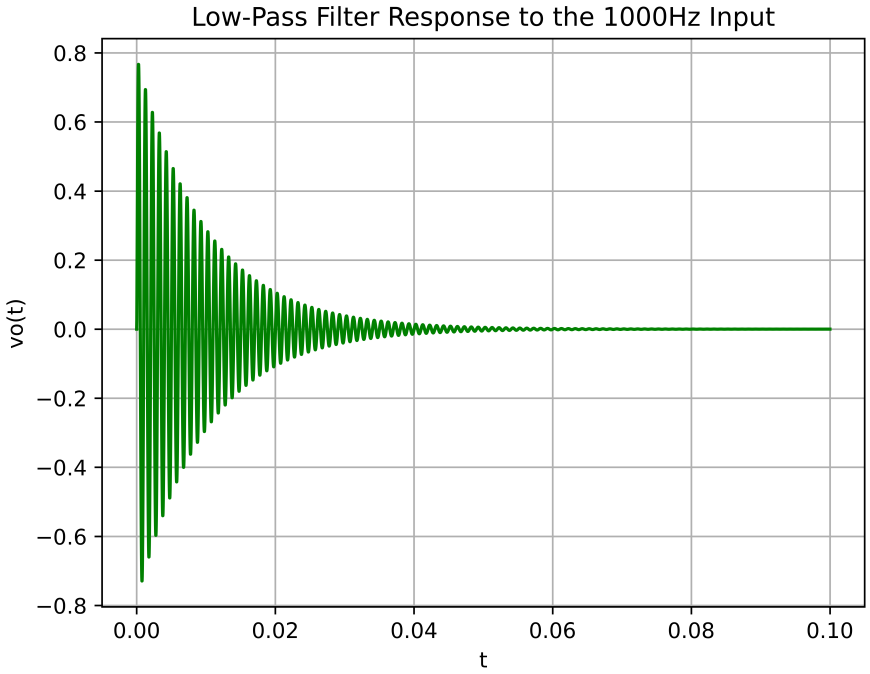
```
t_4l = py.linspace(0, 0.1, 5001)
t_4h = py.linspace(0, 0.0001, 5001)
vl_t = py.exp(-100*t_4l)*py.sin(2000*pi*t_4l)
vh_t = py.exp(-50000*t_4h)*py.sin(2000000*pi*t_4h)

(t_4, vo_lpl, svec) = sig.lsim(h_1s, vl_t, t_4l)
(t_4, vo_hpl, svec) = sig.lsim(h_3s, vl_t, t_4l)
(t_4, vo_lph, svec) = sig.lsim(h_1s, vh_t, t_4h)
(t_4, vo_hph, svec) = sig.lsim(h_3s, vh_t, t_4h)
```

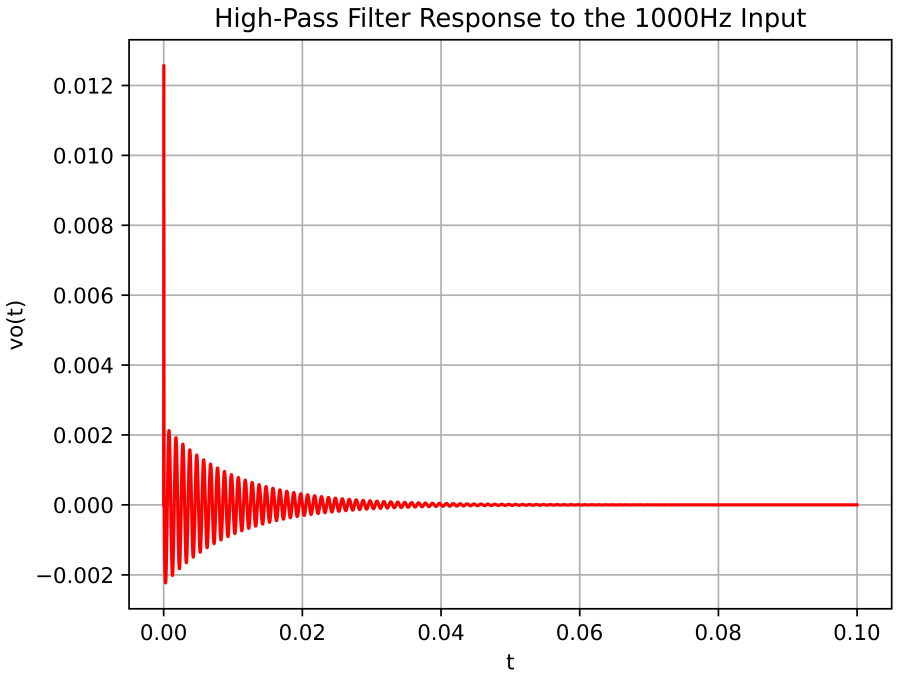
where `h1s` (low-pass case) and `h3s` (high-pass case) are the transfer functions obtained by first solving the respective matrices to get the **sympy** form, and then using **sym extract** function to get the numerator and denominator coefficients and finally using **signal.lti** to generate the Laplace forms of the transfer functions.

The plots obtained for the two inputs by each of the filters, along with the respective input plots, is as follows:

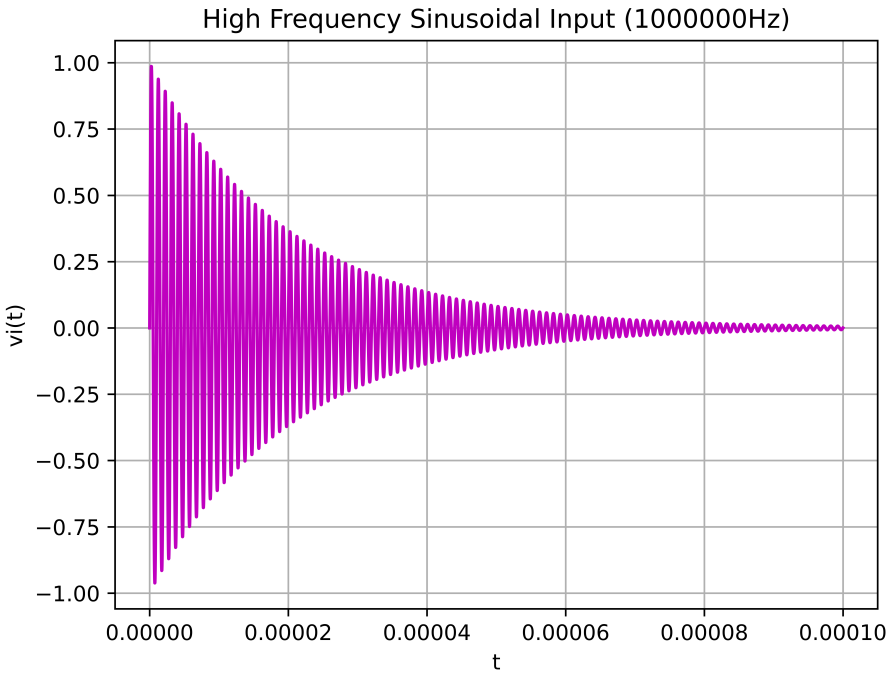
For the low-frequency input case,

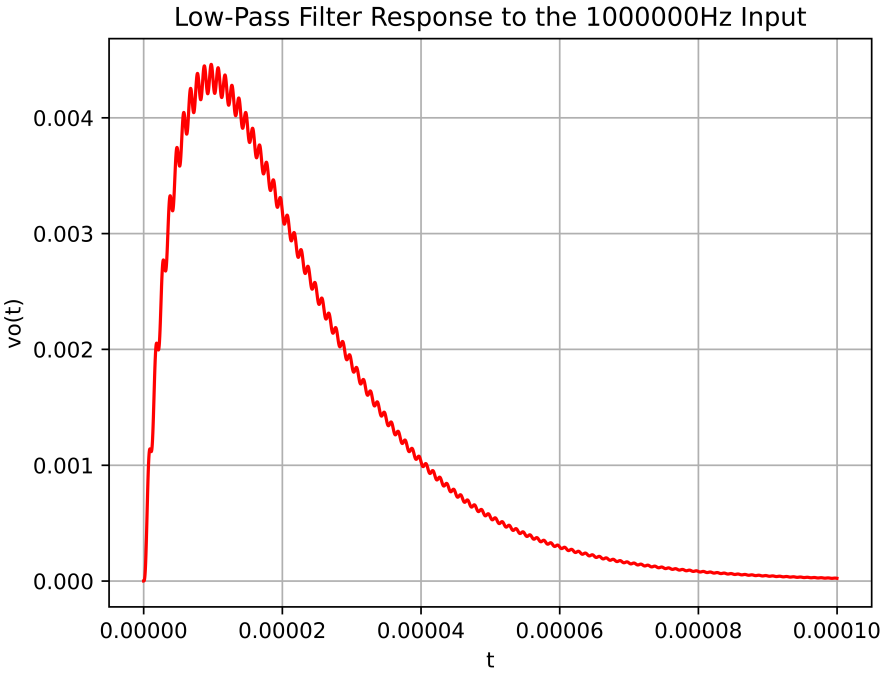


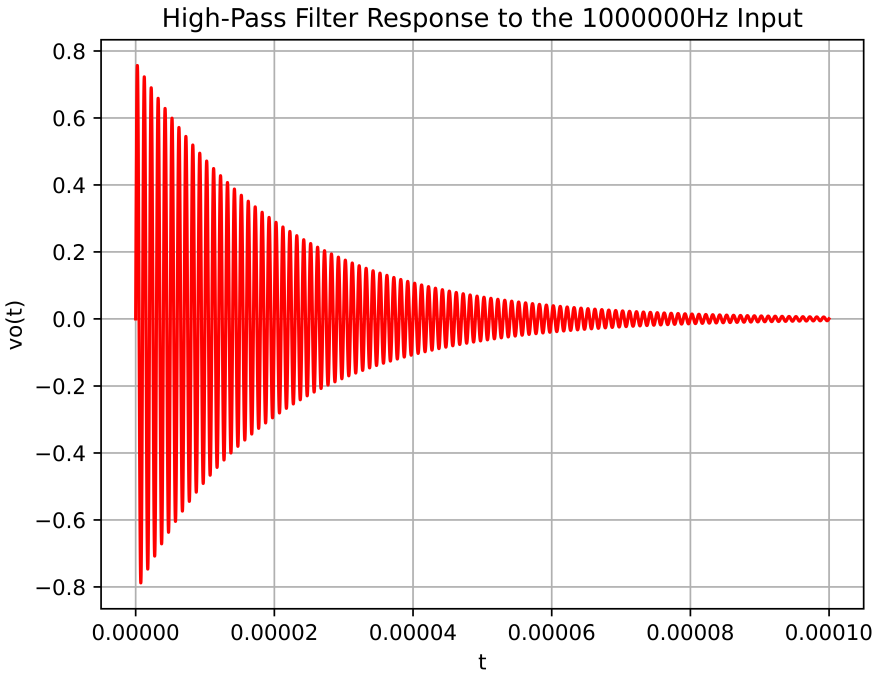

Low-Pass Filter Response to the 1000Hz Input
vo(t)
t
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0.00
0.02
0.04
0.06
0.08
0.10


High-Pass Filter Response to the 1000Hz Input
vo(t)
t
0.012
0.010
0.008
0.006
0.004
0.002
0.000
−0.002
0.00
0.02
0.04
0.06
0.08
0.10

For the high-frequency input case,

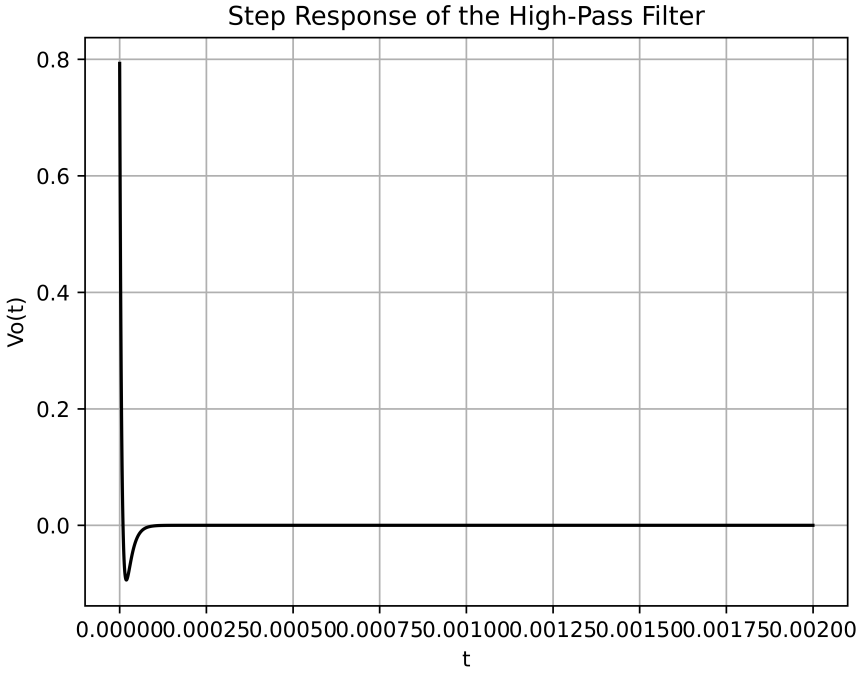



Low-Pass Filter Response to the 1000000Hz Input
vo(t)
0.004
0.003
0.002
0.001
0.000
0.00000
0.00002
0.00004
0.00006
0.00008
0.00010
t


High-Pass Filter Response to the 1000000Hz Input
vo(t)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0.00000
0.00002
0.00004
0.00006
0.00008
0.00010
t

3.5 P5: Obtaining the Step Response for the High Pass Filter

The step response for the high pass filter is as follows:



The reason for such a response is that, initially the capacitors can be considered to be shorted in the circuit. When the step input was given to the circuit, the output can be given as follows,

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

$$V_o = G(V_p - V_m)$$

$$V_o = G(1 - \frac{V_o}{G})$$

Thus, we can say that, $V_o = \frac{G}{2}$, which is approximately 0.8. After that, the output drastically decays, slightly undershoots until finally hitting zero. The reason why it hits zero at steady state is that, at steady state, the capacitors become open circuited, hence making $V_o = 0$.

4 Conclusion

The concluding points are as follows:

- The **sympy** module is a very powerful tool to symbolically solve the circuit equations and analyse various responses.
- The Low-pass filter handsomely allows the low-frequency signals like the low-frequency damped sinusoid shown above and highly attenuates the high-frequency signal as also noted above.
- The High-pass filter handsomely allows the high-frequency signals like the high-frequency damped sinusoid shown above and highly attenuates the low-frequency signal as also noted above.
- The step response in both the cases take some finite time to reach their steady state value, the time which is decided by circuit parameters.