EE2703 Assignment 7: Analysis of Circuits using Laplace Transforms

Anvith Pabba EE19B070

15th April 2021

1 Introduction

In this assignment, we use symbolic algebra to analyse filters in the Laplace domain. We use the **sympy** library and its powerful capabilities to solve the modified nodal equations. We then find the outputs of the filter with different inputs and plot the necessary graphs and analyse our findings.

2 The Low Pass Filter

First, we analyse the Low pass filter given below:

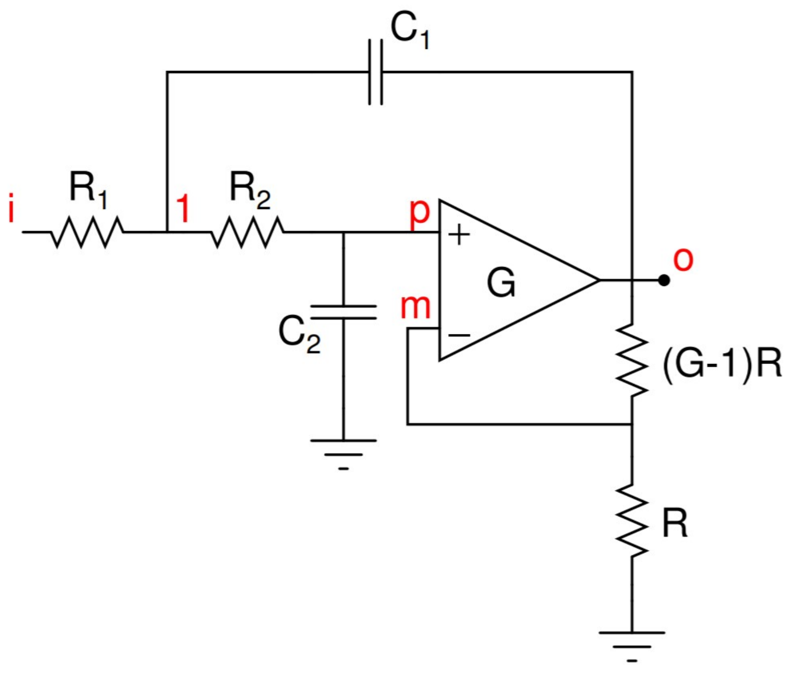

Figure 1: Diagram of the Low Pass Filter

Upon simplifying the equations obtained through modified nodal analysis, we get the following Matrix Equations:

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

Figure 2: Matrix equations for the LPF

2.1 Solving the Equations

We now define a function that takes in the parameters of the given filter and solves the equations and returns the output voltage (in terms of symbol 's').

2.1.1 Code:

```
#Defining a Low pass filter in sympy with necessary parameters
def lowpass(R1,R2,C1,C2,G,Vi):
  s=symbols('s')
  # Creating the matrices through nodal analysis then solvng them to get the output
  A=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=Matrix([0,0,0,-Vi/R1])
  V=A.inv()*b
  return(A,b,V)
```

2.2 Bode plot of Magnitude

Below we find the Bode magnitude plot of the LPF transfer function:

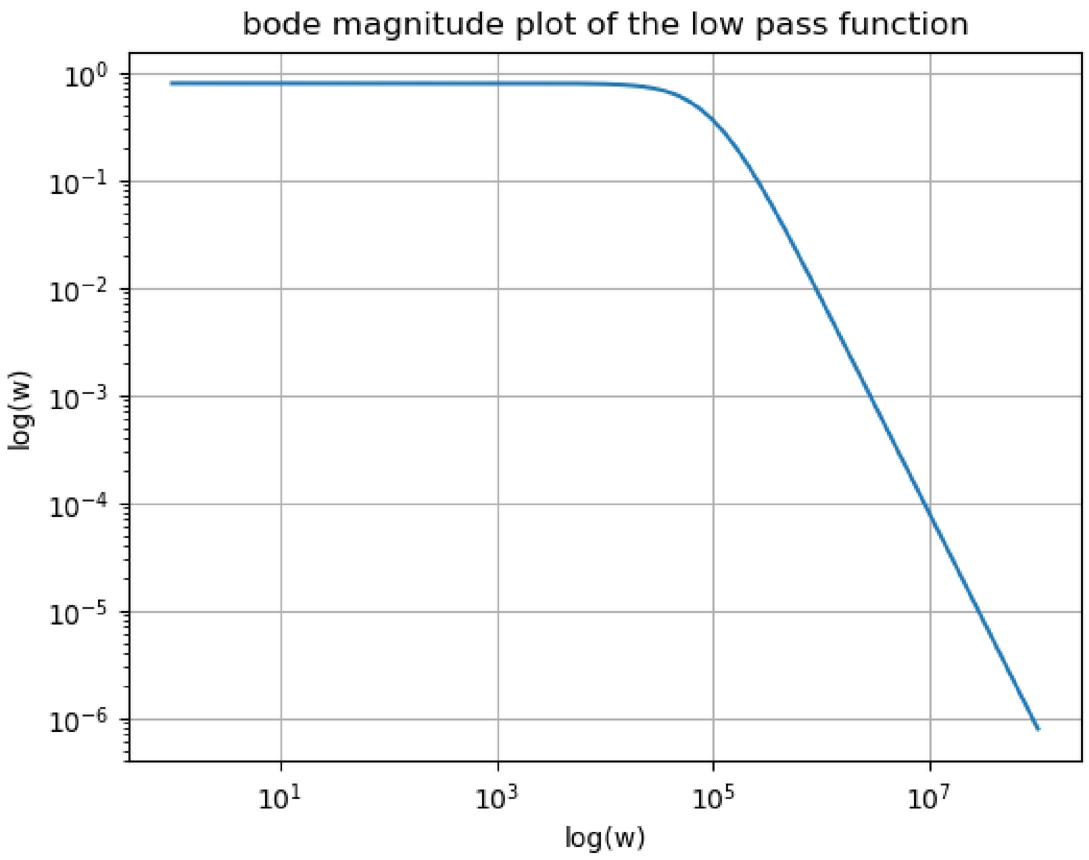

Figure 3: LPF magnitude bode plot

3 The High Pass Filter

First, we analyse the High pass filter given below:

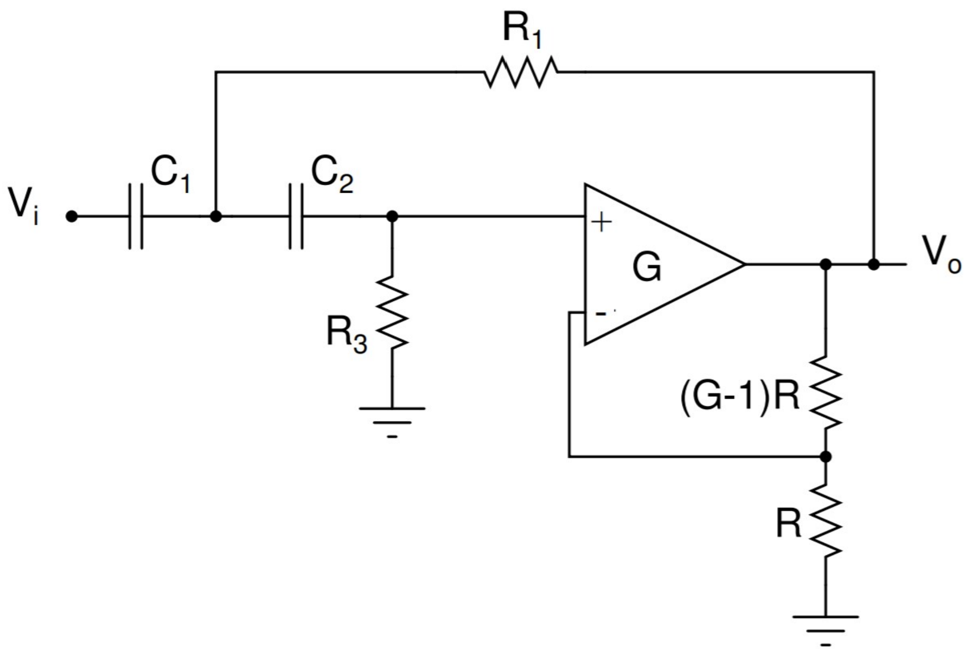

Figure 4: Diagram of the High Pass Filter

Upon simplifying the equations obtained through modified nodal analysis, we get the following Matrix Equations:

$$\begin{pmatrix} 0 & 0 & 1 & -\frac{1}{G} \\ -\frac{-sR_3C_2}{1+sR_3C_2} & 1 & 0 & 0 \\ 0 & -G & G & 1 \\ -1-(sR_1C_1)-(sR_3C_2)) & sC_2R_1 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_p \\ V_m \\ V_o \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -V_i(s)sR_1C_1 \end{pmatrix}$$

Figure 5: Matrix equations for the HPF

3.1 Solving the Equations

We now define a function that takes in the parameters of the given filter and solves the equations and returns the output voltage (in terms of symbol 's').

3.1.1 Code:

```
#Defining a High pass filter in sympy with necessary parameters

def highpass(R1,R3,C1,C2,G,Vi):
    s = symbols("s")
  # Creating the matrices through nodal analysis then solvng them to get the
                                                      output
    A = 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 = Matrix([0,0,0,-Vi*s*C1])
    V = A.inv()*b
    return A,b,V
```

3.2 Bode plot of Magnitude

Below we find the Bode magnitude plot of the HPF transfer function:

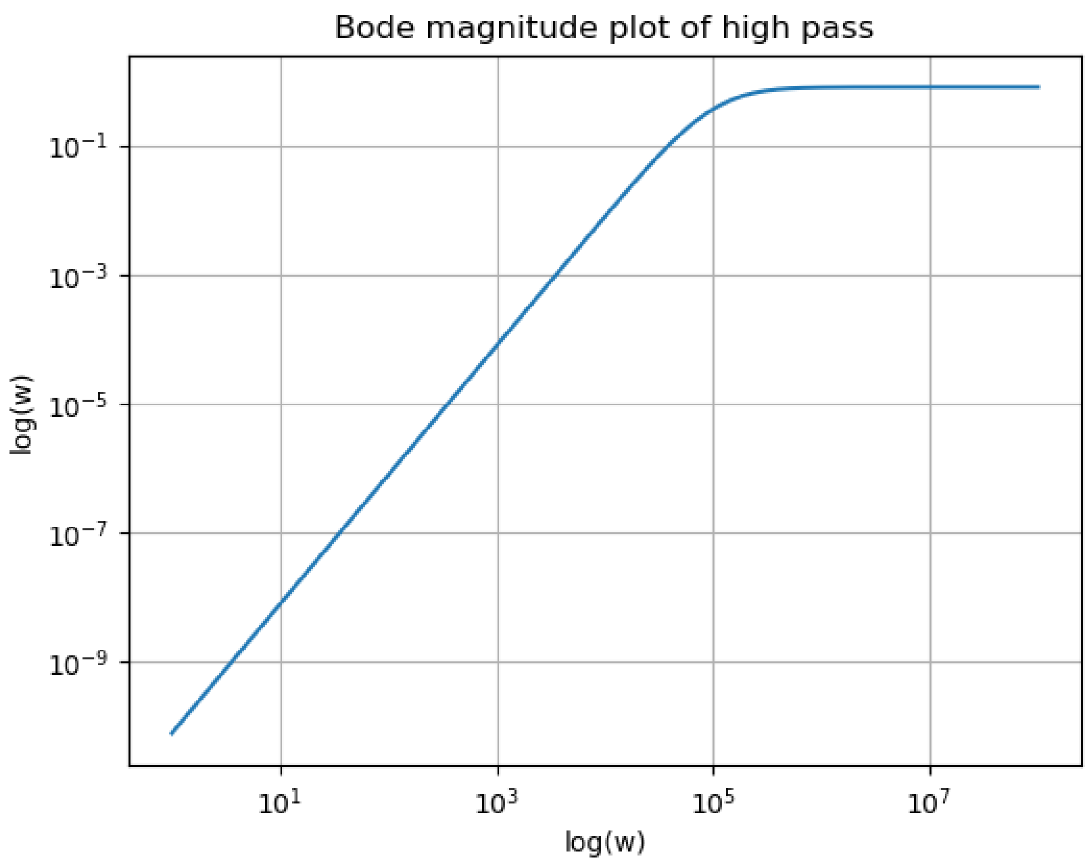

Figure 6: HPF magnitude bode plot

4 changing a sympy syntax into a scipy acceptable syntax:

We do this through simple arithmetic manipulations present in the sympy toolkit. We create a function that converts any expression with a sympy variable **s** into a scipy transfer function. The function is given by: **sympy_to_H_lti()**

4.1 Code:

```
#converts an equation containing sympy variables into a function that scipy
                                        accepts
def sympy_to_H_lti(sympy_func,s):
  s=symbols('s')
  n,d = fraction(sympy_func)  #getting the numerator and denominator
  num_c = Poly(n,s).all_coeffs()  #getting the coeff
  den_c = Poly(d,s).all_coeffs()
  numf = p.array(num_c, dtype=float)  #converting the arrays into numpy arrays
  denf = p.array(den_c, dtype=float)
  H = sp.lti(numf,denf) #creating the lti systems transfer response
  return(H)
```

5 The Assignment

5.1 Question 1: Step response of the LPF

To do this, we simply change the input function in the formula. As it is a step function, we use the following as the input.

$$V_i(s) = \frac{1}{s} \tag{1}$$

The response we get is from using the **lambdify** and the **sp.impulse** commands.

5.1.1 Code:

```
#Question 1:

#step impulse response, Vi(s) = 1/s
A,b,V1=lowpass(10000,10000,1e-9,1e-9,1.586,1/s)
Vo1 = V1[3]
w = p.logspace(0,8,801)
ss=1j*w
hf1=lambdify(s,Vo1,'numpy')
v=hf1(ss)
H1 = sympy_to_H_lti(Vo1,s)

#plotting the magnitude plot of the step response of the low pass filter
plot_display(w,abs(v),'w','magnitude','Magnitude bode plot of step response of
                                       low pass filter','log','log')

t,y=sp.impulse(H1,None,p.linspace(0,0.001,1000))
plot_display(t,y,'t','y','step response of low pass filter','linear','linear')
```

5.1.2 Magnitude plot of the transfer function

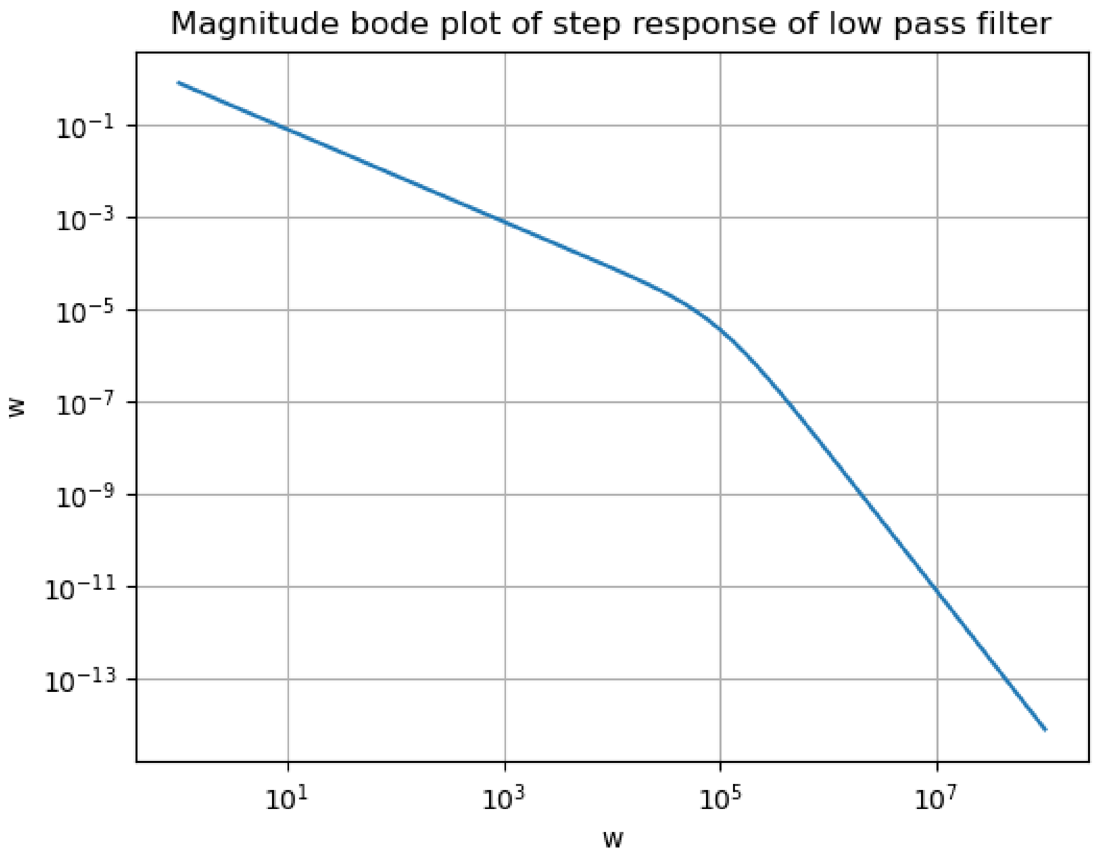

Figure 7: step response of LPF magnitude bode plot

5.1.3 The Step response

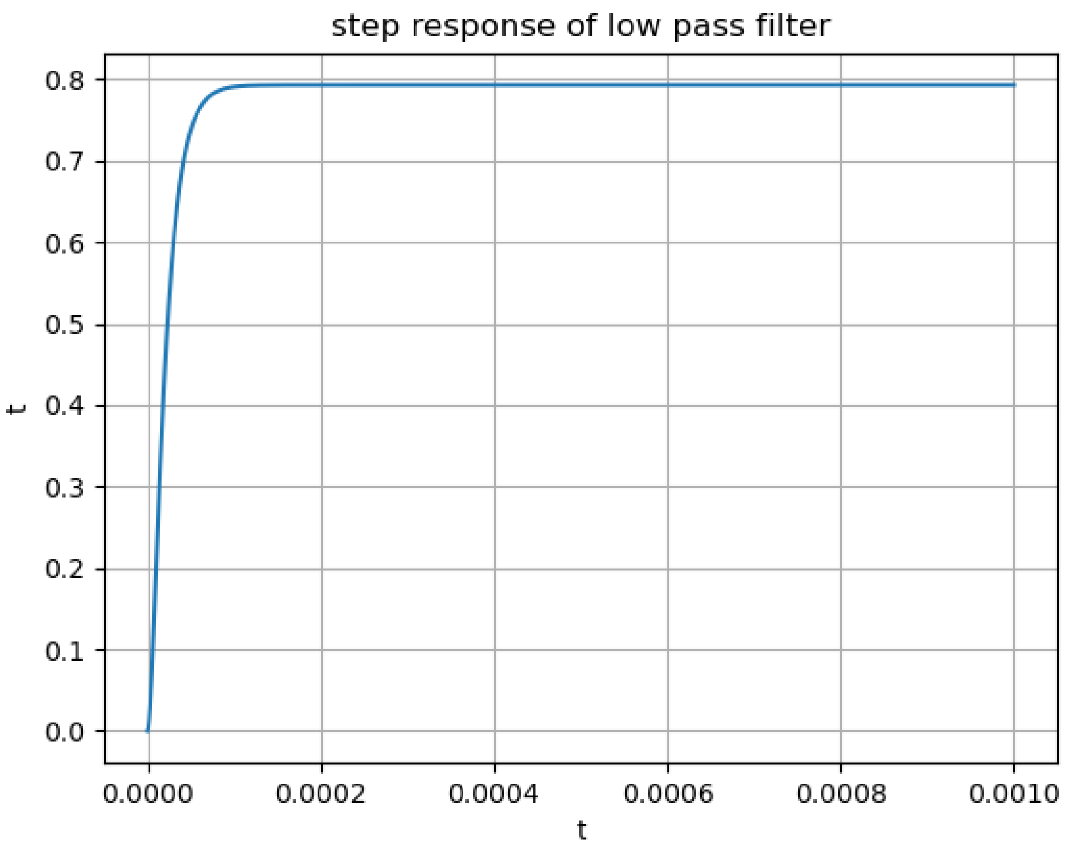

Figure 8: The step response in the time domain

5.2 Question 2: Response to a Sum of Sinusoidal inputs

Now, the input is given by:

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

so to find the output we use the **sp.lsim** command, which gives out the convolution of a given transfer function and input signal.

5.2.1 Code:

```
#finding the solutoion matrix
A,b,V=lowpass(10000,10000,1e-9,1e-9,1.586,1)
Vo=V[3]
w=p.logspace(0,8,801)
ss=1j*w
hf=lambdify(s,Vo,'numpy')
v=hf(ss)
```

```
#Question 2:

#getting the input signal
t = p.linspace(0,0.005,10000)
```

```
v_i = p.multiply(p.sin(2000*p.pi*t) + p.cos(2*1e6*p.pi*t),p.heaviside(t,0.5))

#getting the transfer function of the LPF
H2 = sympy_to_H_lti(Vo,s)

#performing the convolution
t,y,svec=sp.lsim(H2,v_i,t)
plot_display(t,y,'t','y','output response of input sinusoids','linear','linear'
                                        )
```

5.2.2 Output Signal

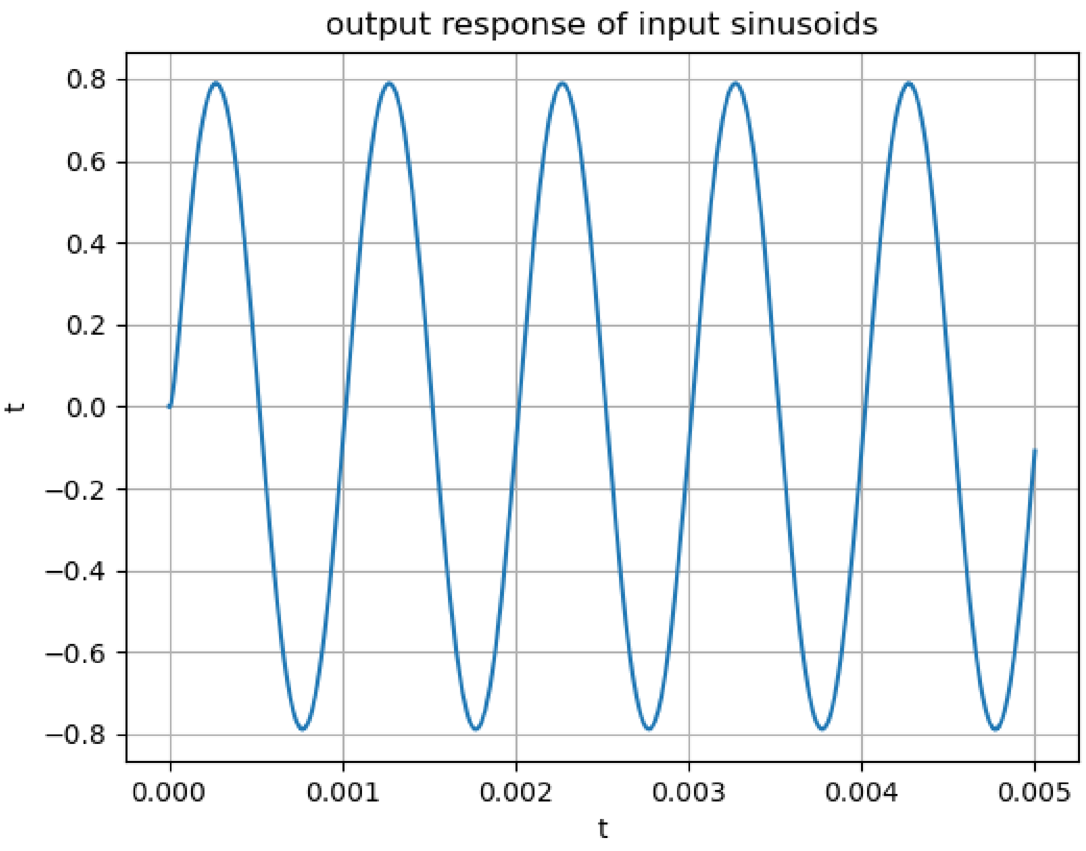

Figure 9: Response of the LPf with mixed sinusoidal signals

5.3 Observations

We can clearly see that the high frequency 10^6Hz sinusoid gets filtered out leaving only the 10^3Hz signal left.

5.4 Question 3:

This question has already been answered in the **The High Pass Filter** section near the beginning of this report.

5.5 Question 4: Response of a Damped sinusoid

We take 2 signals, first is a high freq, high decay coeff signal and the second is a low freq, high decay coeff signal.

5.5.1 Code:

```
#Question 4:

#defining a function that gives the output as damped sinusoid
#with all the necessary input parameters
def damped_sinusoid(freq,type,decay,t):
  if type == 'sin':
    return p.sin(freq*2*p.pi*t)*p.exp(-1*decay*t)
  if type == 'cos':
    return p.cos(freq*2*p.pi*t)*p.exp(-1*decay*t)


#plotting the input and output graphs for a low freq, high decay sinusoid
t = p.linspace(0,0.01,10000)
v_i3 = damped_sinusoid(2000,'sin',1000,t)
t,y,svec=sp.lsim(H3,v_i3,t)
plot_display(t,damped_sinusoid(2000,'sin',1000,t),'t','$v_i3$','damped sinusoid
                                         input response (low freq)','linear','
                                        linear')
plot_display(t,y,'t','y','output response','linear','linear')


#plotting the input and output graphs for a high freq, high decay sinusoid
t = p.linspace(0,0.01,10000)
v_i4 = damped_sinusoid(20000,'sin',1000,t)
t,y,svec=sp.lsim(H3,v_i4,t)
plot_display(t,damped_sinusoid(20000,'sin',1000,t),'t','$v_i4$','damped
                                        sinusoid input response (high freq)','
                                        linear','linear')
plot_display(t,y,'t','y','output response','linear','linear')
```

5.5.2 Signal 1

for signal 1,

$$V_i(t) = (sin(2000 * t))e^{(-1000*t)} * u(t) \tag{3}$$

5.5.3 Input of the low freq damped sinusoid

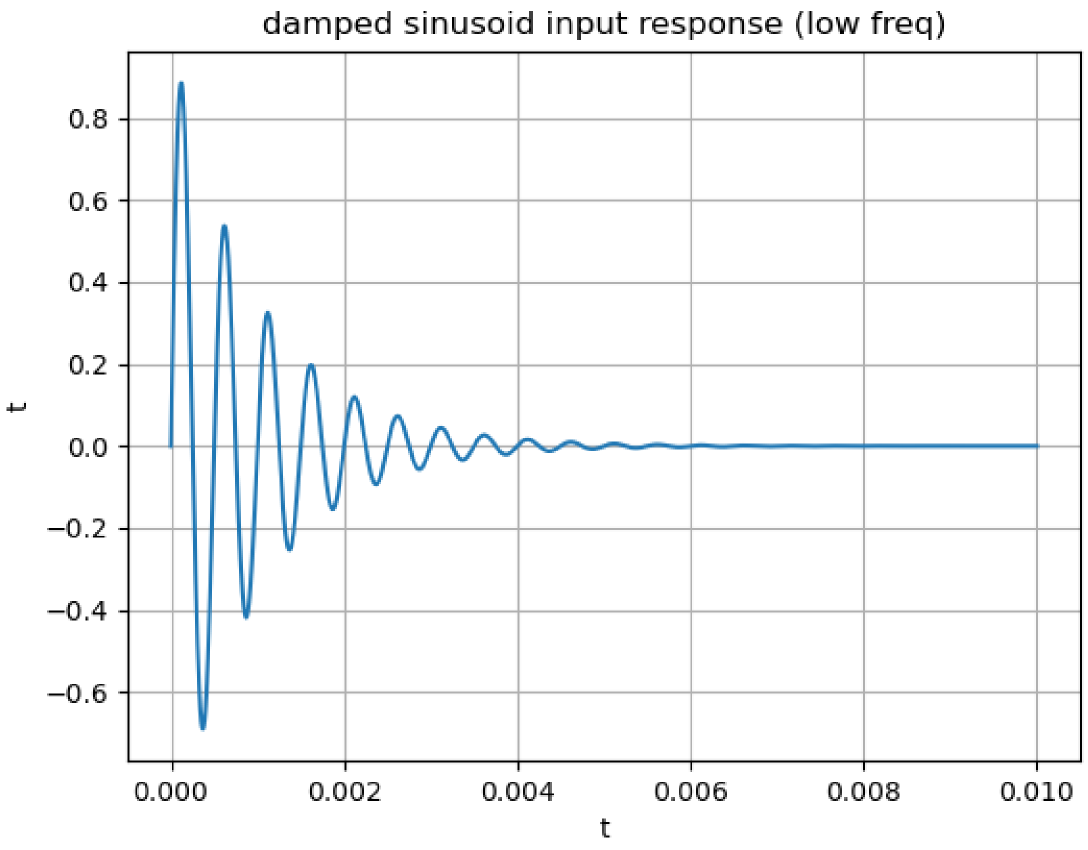

Figure 10: Low freq damped sinusoid

5.5.4 Output of the Low freq damped sinusoid

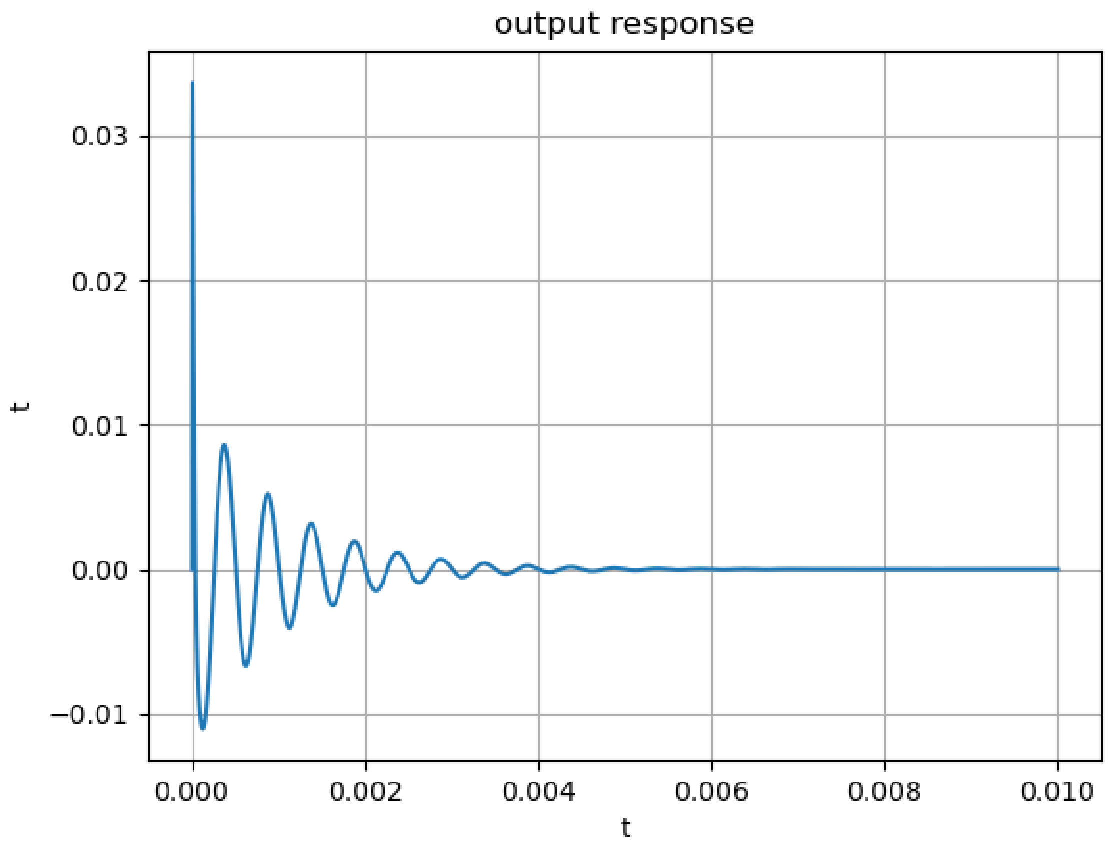

Figure 11: Output of the Low freq damped sinusoid

5.6 Signal 2

for signal 2,

$$V_i(t) = (sin(20000 * t))e^{(-1000*t)} * u(t) \tag{4}$$

5.6.1 Input of the high freq damped sinusoid

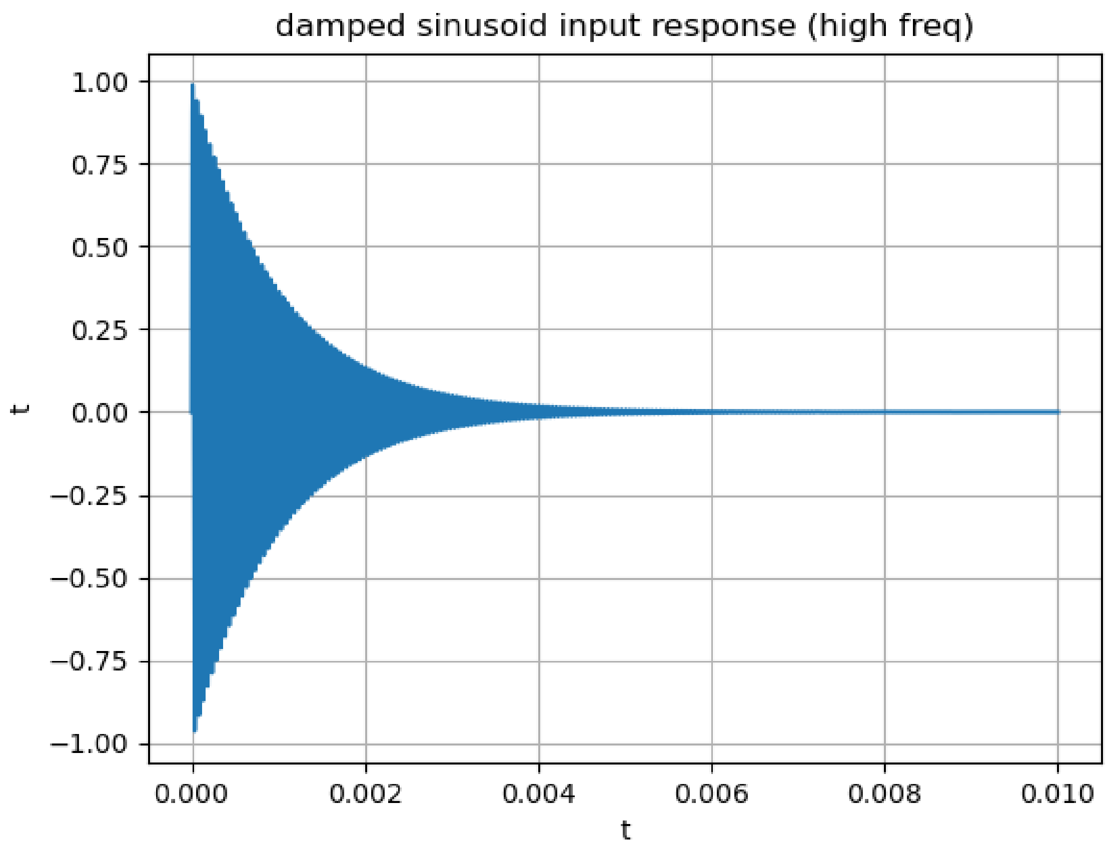

Figure 12: High freq damped sinusoid

5.6.2 Output of the High freq damped sinusoid

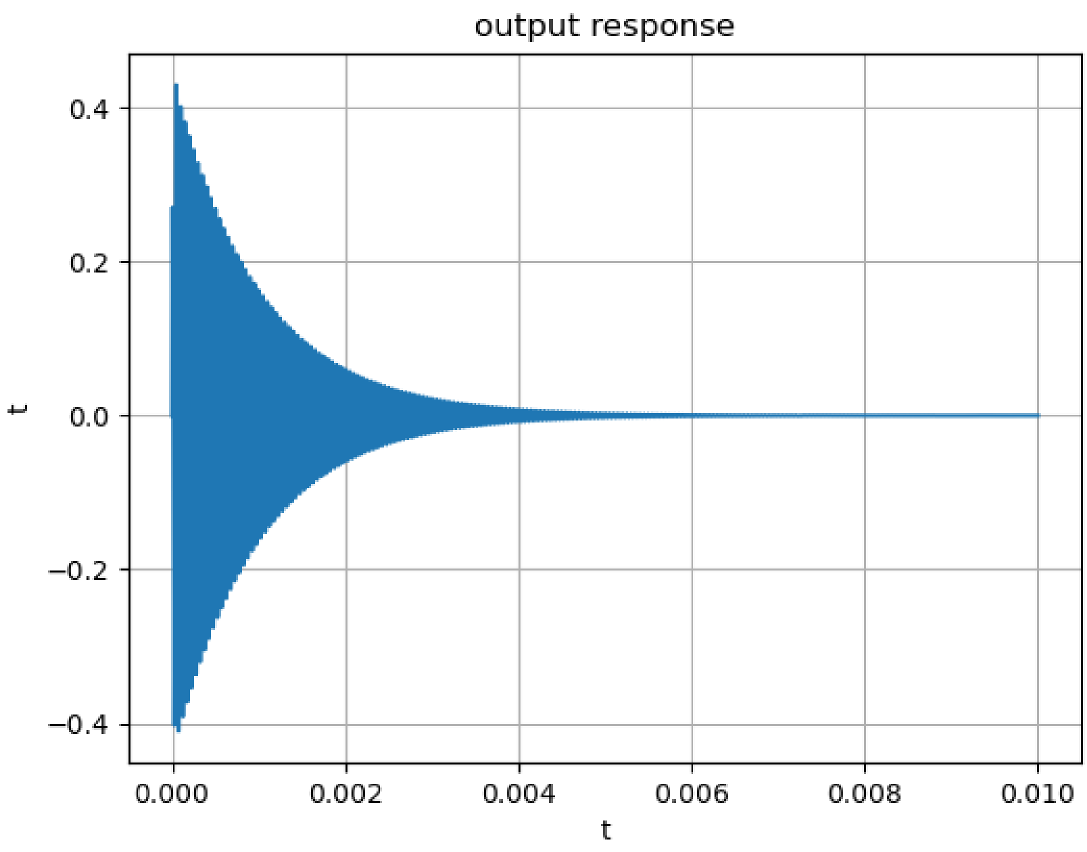

Figure 13: Output of the High freq damped sinusoid

5.7 Question 5: Unit Step Response of the HPF

The plots of the step response are given by:

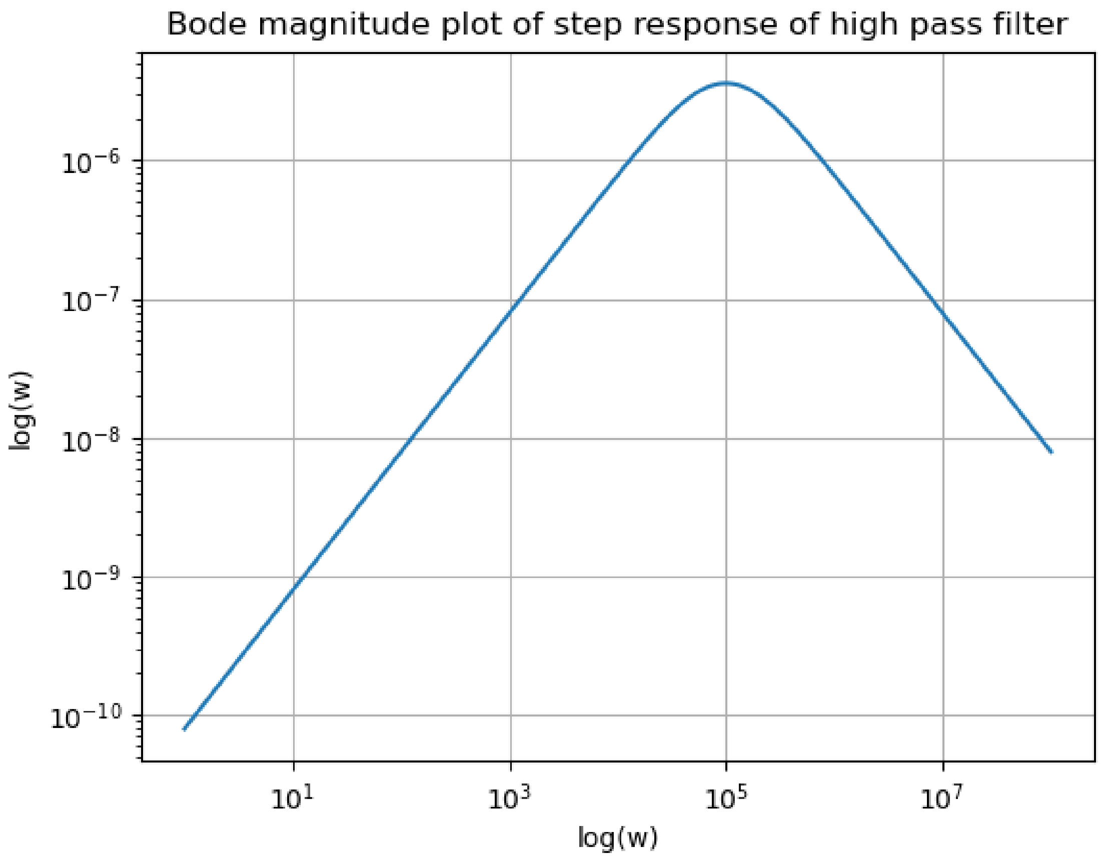

Figure 14: Bode magnitude plot of the Step Response

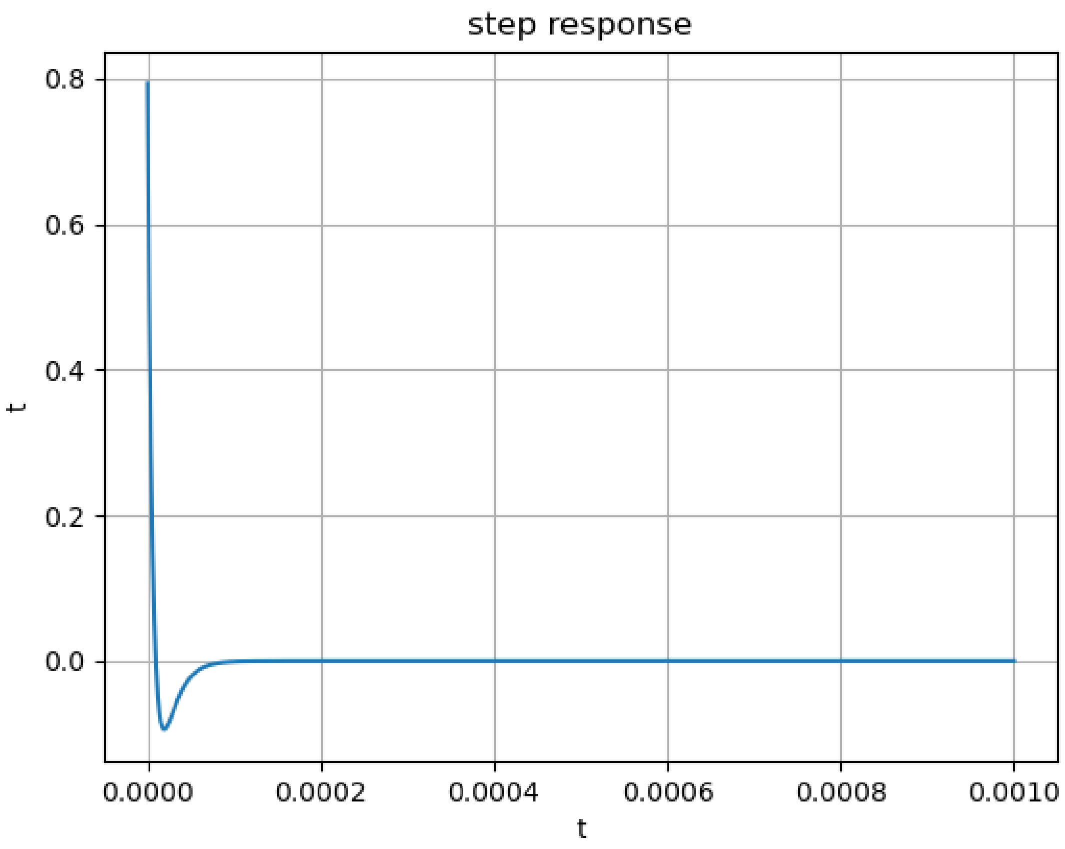

Figure 15: Plot of the Step Response

5.7.1 Analysis of the response

Initially, there is a jump in V_o as the capacitors act like a short circuit just after the abrupt change in the input voltage, but as time goes on, the capacitors act like open circuits and the final output voltage drops to 0.

6 Conclusions

In this assignment, we've learnt how to:

- Analyse complicated circuits in the Laplace domain through nodal analysis
- Use the sympy library to obtain simplified versions of the transfer functions
- Converting expressions between sympy and other syntax's (such as numpy or scipy)
- obtaining the graphs of outputs and the respective bode plots