# Symbolic Python and Laplace transforms
# Assignment 8

Rohithram R, EE16B031
B.Tech Electrical Engineering, IIT Madras

April 6, 2018
First created on March 27,2018

**Abstract**

This report will discuss about how to solve system equations analytically instead of numerical approach which is followed till now, using Symbolic python. And it also delve deeper into circuit analysis using Laplace transforms and analysing them and converting them back to time domain,etc.

## 1 Introduction

- We analyse the infamous LTI systems in continuous time using Laplace transform to find the solutions to the equations governing the system with the help of python tools such as Symbolic python known as *Sympy* and Signal toolbox.

- *symbols* $\rightarrow$ used to declare symbols using sympy.
- *lambdify* $\rightarrow$ Converts the sympy expression into one numpy expression which can be evaluated.
- *system.impulse* $\rightarrow$ Computes the impulse response of the transfer function
- *sp.lti* $\rightarrow$ defines a transfer function from polynomial coefficients of numerator and denominator as inputs.
- In this assignment we solve all equations analytically using expressions with variables like we use to solve manually with variables which is unlike the way we generally follow to solve i.e by solving for some specific cases (Numerically)) rather than solving for general case with variables.
- So we use Symbolic python to do the above mentioned and we analyse the circuits by solving them analytically and analyse the systems finally using numpy.

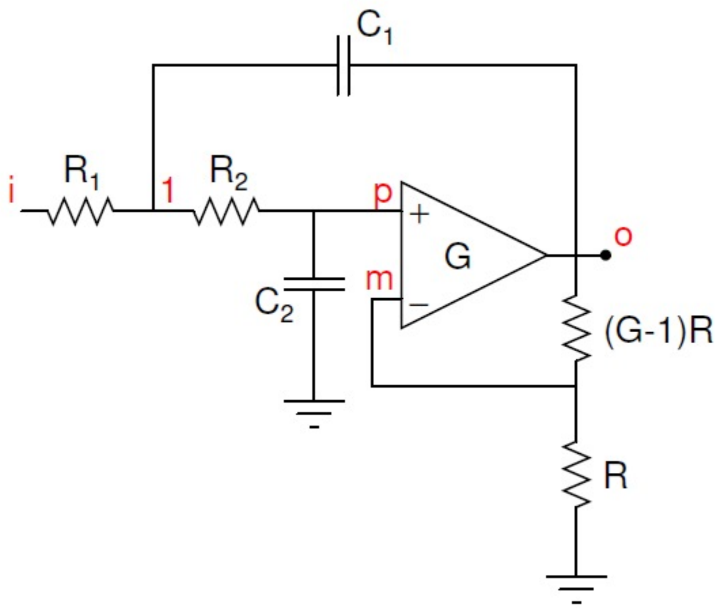
```
In [1]: import writefile_run as writefile_run
```

```
In [2]: # load libraries and set plot parameters
        %matplotlib inline
        from  tabulate import tabulate
        import scipy.signal as sp
        from pylab import *
        from sympy import *

        from IPython.display import set_matplotlib_formats
        set_matplotlib_formats('pdf', 'png')
        plt.rcParams['savefig.dpi'] = 75

        plt.rcParams['figure.autolayout'] = False
        plt.rcParams['figure.figsize'] = 12, 9
        plt.rcParams['axes.labelsize'] = 18
        plt.rcParams['axes.titlesize'] = 20
        plt.rcParams['font.size'] = 16
        plt.rcParams['lines.linewidth'] = 2.0
        plt.rcParams['lines.markersize'] = 6
        plt.rcParams['legend.fontsize'] = 18
        plt.rcParams['legend.numpoints'] = 2
        plt.rcParams['legend.loc'] = 'best'
        plt.rcParams['legend.fancybox'] = True
        plt.rcParams['legend.shadow'] = True
        plt.rcParams['text.usetex'] = True
        plt.rcParams['font.family'] = "serif"
        plt.rcParams['font.serif'] = "cm"
        plt.rcParams['text.latex.preamble'] = r"\usepackage{subdepth}, \usepackage{type1cm}"
```

```
In [3]: init_printing(use_latex='mathjax')
```

## 2 Question 1:

- We analyse a Low pass filter circuit given below using symbolic python and numpy.
- Observe and analyse the responses of the systems for various inputs.



Low Pass Filter circuit realised using Opamp

- Using sympy we can represent the nodal equations of the ciruit in the form of matrix and solve it to find Vo(t).

$$\begin{pmatrix} 0 & 0 & 1 & -\frac{1}{G} \\ -\frac{1}{1+sR2C2} & 1 & 0 & 0 \\ 0 & -G & G & 1 \\ -\frac{1}{R_1} - \frac{1}{R_2} - sC_1 & \frac{1}{R2} & 0 & sC_1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_p \\ V_m \\ V_o \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -V_i(s)/R_1 \end{pmatrix}$$

- Obtain the Transfer function of the network, which is determined by finding laplace transform of impulse response ($V_i(t) = \delta(t)$ whose $Laplace\ Transform$ is $\mathcal{V}_\rangle(s) = \frac{1}{s}$).
- From transfer function obtain the Quality factor of the system, which essentially says how sharp the system is certain range frequencies.

- If $Q < 0.5$ system is overdamped since damping factor $\zeta = \frac{1}{2Q} > 1$ for $Q < 0.5$ which means the unit step response will raise slowly from 0 to $V_{max}$ exponentially unlike immediately changing from 0 $to V_{max}$.
- To observe this behaviour we give unit step as input and analyse the output.
- Obtain unit step response of the system i.e

$$V_i(t) = u(t)\ Volts \tag{1}$$

$$\mathcal{V}_{\rangle}(s) = \frac{1}{s} \tag{2}$$

- Obtain and analyse the response for sinusoid with a low frequency and high frequency component of $\omega_1 = 2000\pi\ rads^{-1}\ and\ \omega_2 = 2 * 10^6\pi\ rads^{-1}$.

$$V_i(t) = (\ \sin(2000\pi t) + \cos(2x10^6\pi t)\ )u_o(t)\ Volts \tag{3}$$

- Laplace transform of input sinusoid in generalised form with frequencies of $\omega_1$ and $\omega_2$ for sine and cosine respectively is

$$\mathcal{V}_{\rangle}(s) = \frac{\omega_1}{(s^2 + \omega_1^2)} + \frac{s}{(s^2 + \omega_2^2)} \tag{4}$$

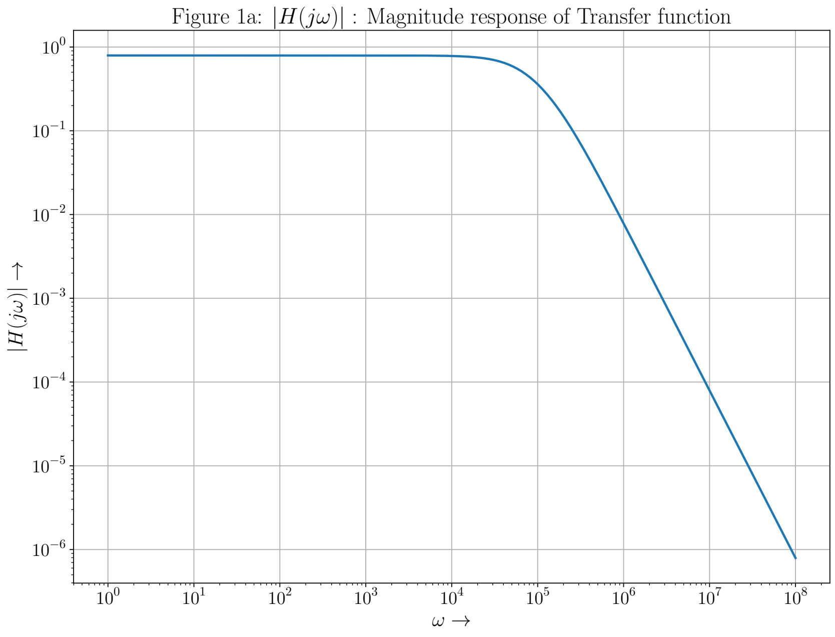
- *sp.impulse* is used for converting $\mathcal{V}_{\rangle}(s)$ into time domain.
- Determine and Plot the output voltage $V_o(t)$ for the cases above and analyse them.

In [4]:
```
'''
function to solve for V(s) by Matrix inversion
This function used for Low pass filter
arguments : R1,R2,C1,C2,G   - parameters of the circuit
            Vi - Laplace transform of Input.
'''

def LpfResponse(R1,R2,C1,C2,G,Vi):
    s=symbols('s')
    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)
```

In [5]:
```
'''
function to solve for Transfer function H(s)
To convert sympy polynomial to sp.lti polynomial
Arguments : num_coeff   - array of coefficients of denominator polynomial
            den_coeff   - array of coefficients of denominator polynomial
Returns   : Hs          - Transfer function in s domain
'''

def SympyToLti(num_coeff,den_coeff):
    num_coeff = np.array(num_coeff, dtype=float)
    den_coeff = np.array(den_coeff,dtype=float)
    H_num = poly1d(num_coeff)
    H_den = poly1d(den_coeff)
    Hs = sp.lti(H_num,H_den)
    return Hs
```

```
In [6]: '''
        function to solve for Output voltage for given circuit
        Arguments : R1,R2,C1,C2,G   - parameters of the circuit
                    Vi - Laplace transform of input Voltage
                    circuitResponse - function defined which is either lpf or Hpf
        Returns   : v,Vlti         - v is array of values in jw domain
                                   - Vlti is sp.lti polynomial in s
        '''

        def solver(R1,R2,C1,C2,G,Vi,circuitResponse):
            s = symbols('s')
            A,b,V = circuitResponse(R1,R2,C1,C2,G,Vi)
            Vo = V[3]
            num,den = fraction(simplify(Vo))
            num_coeffs = Poly(num,s).all_coeffs()
            den_coeffs = Poly(den,s).all_coeffs()
            Vlti = SympyToLti(num_coeffs,den_coeffs)

            w = logspace(0,8,801)
            ss = 1j*w
            hf = lambdify(s,Vo,"numpy")
            v = hf(ss)

            #Calculating Quality factor for the system
            if(Vi == 1):          # Vi(s)=1 means input is impulse
                Q  = sqrt(1/(pow(den_coeffs[1]/den_coeffs[2],2)/(den_coeffs[0]/den_coeffs[2])))
                print("Quality factor of the system : %g"%(Q))
                return v,Vlti,Q
            else:
                return v,Vlti

In [7]: #Declaring params of the circuit1
        R1 = 10000
        R2 = 10000
        C1 = 1e-9
        C2 = 1e-9
        G  = 1.586

        # w is x axis of bode plot
        s = symbols('s')
        w = logspace(0,8,801)

        Vi_1 = 1      #Laplace transform of impulse
        Vi_2 = 1/s    #Laplace transform of u(t)

        #Finding Vo(t) for these given two inputs
        Vo1,Vs1,Q = solver(R1,R2,C1,C2,G,Vi_1,LpfResponse)

        # To find Output Voltage in time domain
        t1,Vot1 = sp.impulse(Vs1,None,linspace(0,1e-2,10000))

        Vo2,Vs2 = solver(R1,R2,C1,C2,G,Vi_2,LpfResponse)

        # To find Output Voltage in time domain
        t2,Vot2 = sp.impulse(Vs2,None,linspace(0,1e-3,100000))

Quality factor of the system : 0.453104


In [8]: #plot of Magnitude response of Transfer function
```

```
fig1 = figure()
ax1 = fig1.add_subplot(111)
ax1.loglog(w,abs(Vo1))
title(r"Figure 1a: $|H(j\omega)|$ : Magnitude response of Transfer function")
xlabel(r"$\omega \to $")
ylabel(r"$ |H(j\omega)| \to $")
grid()
savefig("Figure1a.jpg")
show()
```

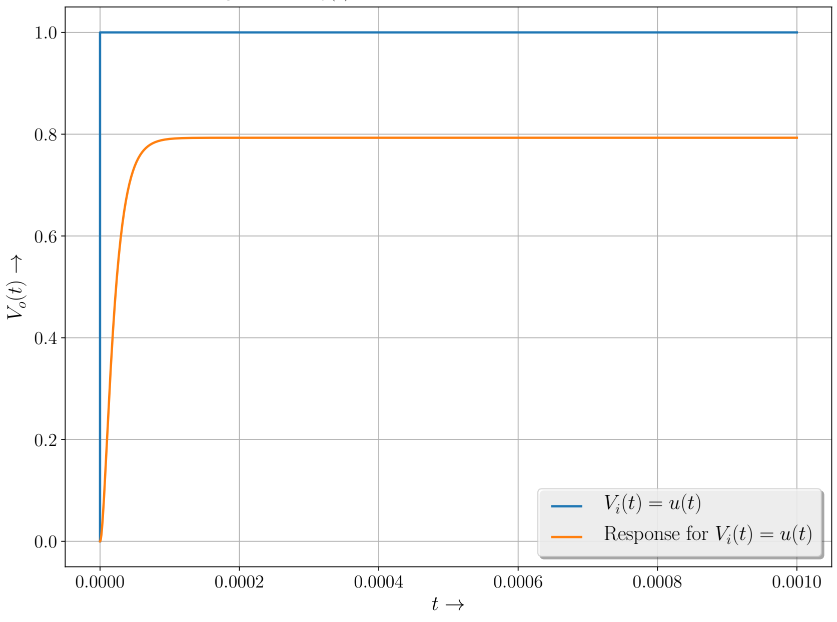


### 2.0.1 Results and Discussion:

- As we observe the plot and the circuit that we know it is a low pass filter with bandwidth $0 < \omega < 10^4$.
- So the circuit will only pass input with frequencies which are in range of bandwidth only and attenuates other frequencies largely since its second order filter with -40dB/dec drop in gain

```
In [9]: #plot of unit Step response

        fig1b = figure()
        ax1b = fig1b.add_subplot(111)
        # Input - Unit step function
        ax1b.step([t2[0],t2[-1]],[0,1],label=r"$V_{i}(t) = u(t)$")

        ax1b.plot(t2,Vot2,label=r"Response for $V_{i}(t) = u(t)$")

        ax1b.legend()
        title(r"Figure 1b: $V_{o}(t)$ : Unit Step response in time domain")
        xlabel(r"$t \to $")
        ylabel(r"$ V_{o}(t) \to $")
        grid()
        savefig("Figure1b.jpg")
        show()
```

Figure 1b: $V_o(t)$ : Unit Step response in time domain



### 2.0.2 Results and Discussion :

- As we observe the plot that $V_o(t)$ increases quickly from 0 to 0.8 and settles at 0.8 for after some time and remains constant.
- Because since the network is lowpass filter, the output must be dominated by DC gain at steady state.
- And we determined Quality factor of the system as $Q = 0.453.. < \frac{1}{\sqrt{2}}$ Which implies that the gain of the system never exceeds DC Gain and always less than that. This observation comes by analysing the general form of second order transfer function.
- So with this we see that unit step response is always less that the DC Gain of 0.8 which is obtained by putting $s = 0$ in the $\mathcal{V}_\wr(s)$.
- Also If $Q < 0.5$ system is overdamped since damping factor $\zeta = \frac{1}{2Q} > 1$ for $Q < 0.5$ which means the unit step response will raise slowly from 0 to $V_{max}$ exponentially unlike immediately changing from 0 $to V_{max}$
- So this is also observed in the plot as it slowly raises from 0 to 0.8 and settles.

## 3 Question 2:

- Now we give different input to the same circuit given above : $V_i(t)$ as follows

$$V_i(t) = (\ \sin(2000\pi t) + \cos(2x10^6\pi t)\ )u_o(t)\ Volts \tag{5}$$

- So the laplace transform of $V_i(t)$ in generalised form with frequencies of $\omega_1$ and $\omega_2$ for sine and cosine respectively is

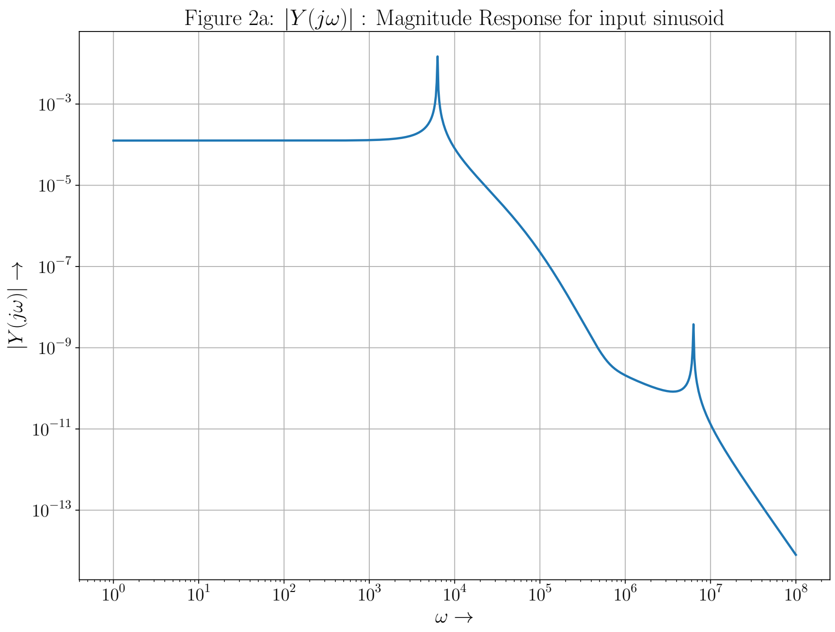
$$\mathcal{V}_\rangle(s) = \frac{\omega_1}{(s^2 + \omega_1^2)} + \frac{s}{(s^2 + \omega_2^2)} \tag{6}$$

- Determine the output voltage $V_o(t)$ using sp.impulse.

- Plot the Magnitude Response of $V_o(t) \to |V_o(j\omega)|$ and $V_o(t)$
- Using this we analyse the results.

```
In [10]: #input sinusoid frequencies in rad/s
         w1 = 2000*pi
         w2 = 2*1e6*pi

         #Laplace transform of given input sinusoid
         Vi_3 =  w1/(s**2 + w1**2) + s/(s**2 + w2**2)
         Vo3,Vs3 = solver(R1,R2,C1,C2,G,Vi_3,LpfResponse)
         #Vo(t) for sinusoid input
         t3,Vot3 = sp.impulse(Vs3,None,linspace(0,1e-2,10000))
```

```
In [11]: #plot of Magnitude Response of sinusoidal input

         fig2 = figure()
         ax2 = fig2.add_subplot(111)
         ax2.loglog(w,abs(Vo3))

         title(r"Figure 2a: $|Y(j\omega)|$ : Magnitude Response for input sinusoid")
         xlabel(r"$\omega \to $")
         ylabel(r"$ |Y(j\omega)| \to $")
         grid()
         savefig("Figure2a.jpg")
         show()
```

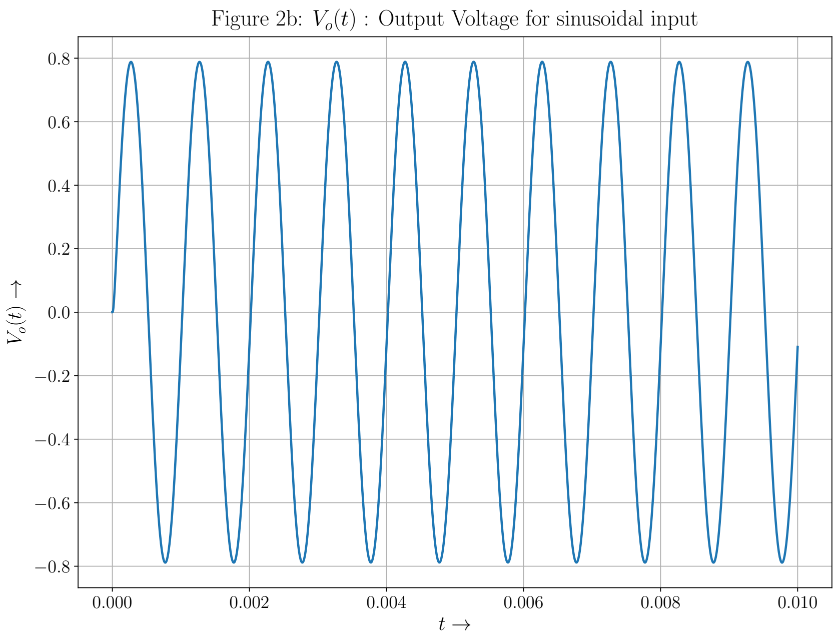


### 3.0.3 Results and Discussion:

- As we observe the plot of transer function's magnitude response and the circuit that we know it is a Low pass filter with bandwidth $0 < \omega < 10^4$.
- So the circuit will only pass input with frequencies which are in range of bandwidth only. But since its not a ideal low pass filter as its gain doesn't drop abruptly at $10^4$ rather gradual decrease which is observed from magnitude response plot of the transfer function.
- Thats why higher frequency of $\omega = 2 * 10^6\pi$ is attenuated largely compared to lower one, which can be observed from above plot.

- And the peaks are due to poles on $j\omega$ axis from both sin *and* cos terms in the input sinusoid.
- So the output $V_o(t)$ will be mainly of $\sin(2000\pi t)$ with higher frequencies riding over it in long term response i.e Steady state solution.
- This behaviour is observed in the plot that,the $v_o(t) \approx \sin(2000\pi t)$.
- With higher frequencies attenuated largely since its second order filter so gain drops 40dB/dec.

```
In [12]: #plot of Vo(t) for sinusoidal input

         fig2b = figure()
         ax2b = fig2b.add_subplot(111)
         ax2b.plot(t3,(Vot3))
         ax2b.legend()
         title(r"Figure 2b: $V_{o}(t)$ : Output Voltage for sinusoidal input")
         xlabel(r"$t \to $")
         ylabel(r"$ V_{o}(t) \to $")
         grid()
         savefig("Figure2b.jpg")
         show()
```



### 3.0.4 Results and Discussion:

- As we observe the plot and the circuit that we know it is a Low pass filter with bandwidth $0 < \omega < 104$.
- So the circuit will only pass input with frequencies which are in range of bandwidth only. But since its not a ideal low pass filter as its gain doesn't drop abruptly at $10^4$ rather gradual decrease which is observed from magnitude response plot.
- So the output $V_o(t)$ will be mainly of $\sin(2000\pi t)$ with higher frequencies attenuated largely since its second order filter so gain drops 40dB/dec.
- Since its sin function, we can observe that $V_o(t)$ starts from 0

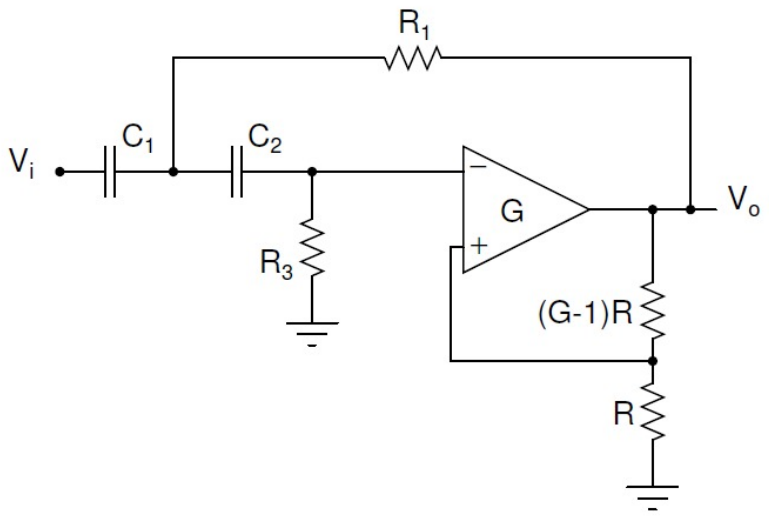
## 4 Question 3 ,4 & 5:

- Now we analyse a High pass filter circuit given below using symbolic python.
- observe the responses of the systems for various inputs.

- Using sympy we can represent the nodal equations of the ciruit in the form of matrix and solve it to find Vo(t).

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

- $R_1 = 10k$,$R_2 = 10k$, $C1 = C2 = 10^{-9}F$



High Pass Filter circuit realised using Opamp

- Obtain the Transfer function of the network, which is determined by finding laplace transform of impulse response($V_i(t) = \delta t$).
- Obtain and analyse the response for undamped and damped sinusoids.

$$V_i(t) = (\ \sin(2000\pi t) + \cos(2x10^6\pi t)\ )u_o(t)\ Volts \tag{7}$$

$$V_i(t) = e^{-10^5 t}(\ \sin(2000\pi t) + \cos(2x10^6\pi t)\ )u_o(t)\ Volts \tag{8}$$

- Laplace transform of undamped sinusoid in generalised form with frequencies of $\omega_1$ and $\omega_2$ for sine and cosine respectively is

$$\mathcal{V}_{\rangle}(s) = \frac{\omega_1}{(s^2+\omega_1^2)} + \frac{s}{(s^2+\omega_2^2)} \tag{9}$$

- Laplace transform for damped sinusoid in generalised form with damping factor denoted as $a$ whose value is taken as $10^5$

$$\mathcal{V}_{\rangle}(s) = \frac{\omega_1}{((s+a)^2+\omega_1^2)} + \frac{s+a}{((s+a)^2+\omega_2^2)} \tag{10}$$

- *sp.impulse* is used for converting $\mathcal{V}_{\rangle}(s)$ into time domain.
- Determine and Plot the output voltage $V_o(t)$ for both the cases above and analyse them.

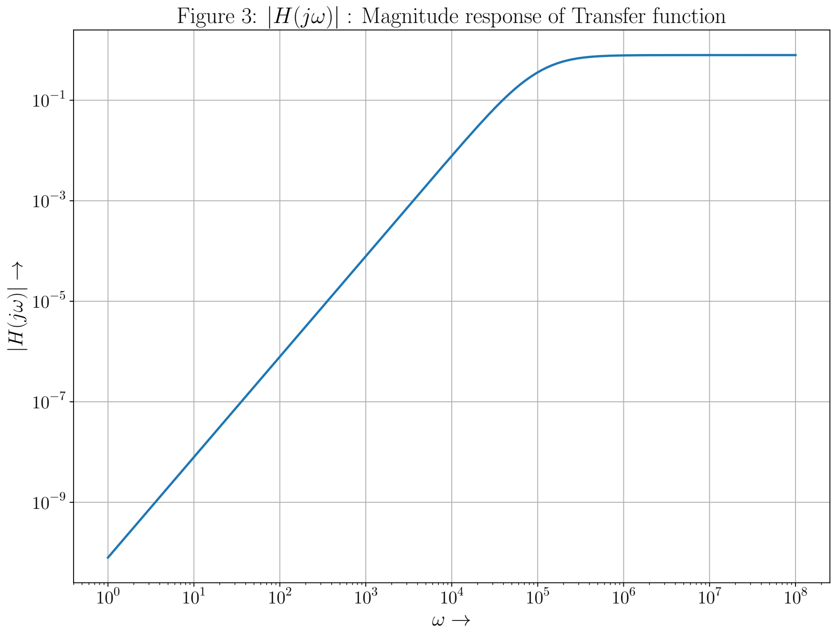
- Using results obtained from this network and previous network compare them and analyse the differences.

```
In [13]: '''
         function to solve for V(s) by Matrix inversion
         This function used for High pass filter
         arguments : R1,R3,C1,C2,G   - parameters of the circuit
                     Vi - Laplace transform of Input.
         '''

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

```
In [14]: #Params for 2nd circuit
         R1b = 10000
         R3b = 10000
         C1b= 1e-9
         C2b = 1e-9
         Gb = 1.586

         #input frequencies for damped sinusoids
         w1 = 2000*pi
         w2 = 2e6*pi
         #Decay factor for damped sinusoid
         a = 1e5


         Vi_1b = 1   # Laplace transform of impulse

         #Laplace transform of damped sinusoid
         Vi_2b =  w1/((s+a)**2 + w1**2) + (s+a)/((s+a)**2 + w2**2)
         #Laplace of unit step
         Vi_3b = 1/s

         #Laplace transform of undamped input sinusoid
         Vi_4b =  w1/(s**2 + w1**2) + s/(s**2 + w2**2)


         '''
         Solving for Output voltage for these inputs
         Qb is the quality factor of this system
         '''
         Vo1b,Vs1b,Qb = solver(R1b,R3b,C1b,C2b,Gb,Vi_1b,HpfResponse)
         t1b,Vot1b = sp.impulse(Vs1b,None,linspace(0,1e-2,10000))

         Vo2b,Vs2b = solver(R1b,R3b,C1b,C2b,Gb,Vi_2b,HpfResponse)
         t2b,Vot2b = sp.impulse(Vs2b,None,linspace(0,5e-5,1000001))

         Vo3b,Vs3b = solver(R1b,R3b,C1b,C2b,Gb,Vi_3b,HpfResponse)
         t3b,Vot3b = sp.impulse(Vs3b,None,linspace(0,5e-4,10001))

         Vo4b,Vs4b = solver(R1b,R3b,C1b,C2b,Gb,Vi_4b,HpfResponse)
         t4b,Vot4b = sp.impulse(Vs4b,None,linspace(0,1e-1,10000))
```

```
Quality factor of the system : 0.453104
```

```
In [15]: #plot of Magnitude response of Transfer function

         fig3 = figure()
         ax3 = fig3.add_subplot(111)
         ax3.loglog(w,abs(Vo1b))
         ax3.legend()
         title(r"Figure 3: $|H(j\omega)|$ : Magnitude response of Transfer function")
         xlabel(r"$\omega \to $")
         ylabel(r"$ |H(j\omega)| \to $")
         grid()
         savefig("Figure3.jpg")
         show()
```

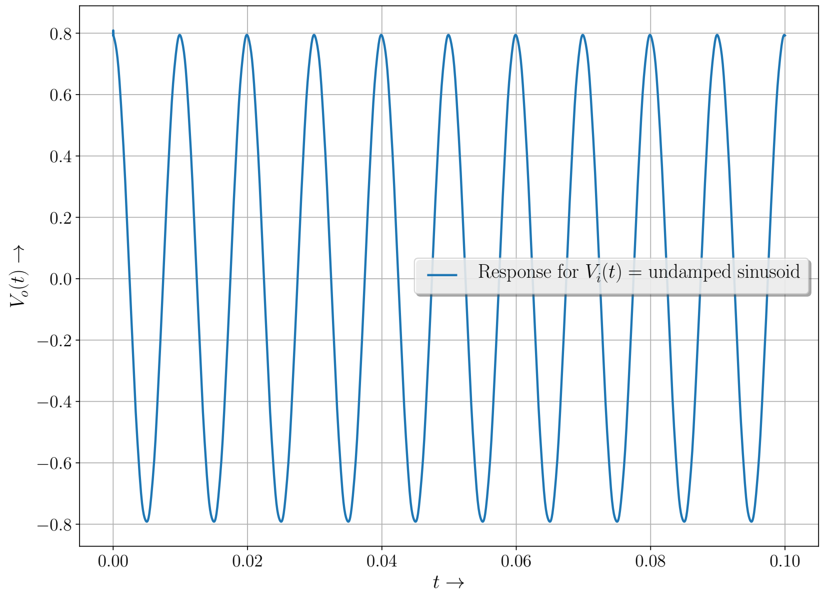


### 4.0.5 Results and Discussion:

- As we observe the plot and the circuit that we know it is a high pass filter with bandwidth $\omega > 10^5$.
- So the circuit will only pass input with frequencies which are in range of bandwidth only.

```
In [16]: #plot of Vo(t) for damped sinusoidal input

         fig6a = figure()
         ax6a = fig6a.add_subplot(111)
         ax6a.plot(t4b,(Vot4b),label=r"Response for $V_{i}(t) = $ undamped sinusoid")
         ax6a.legend()
         title(r"Figure 6a: $V_{o}(t)$ : Output Voltage for undamped sinusoid input")
         xlabel(r"$t \to $")
         ylabel(r"$ V_{o}(t) \to $")
         grid()
         savefig("Figure6a.jpg")
         show()
```

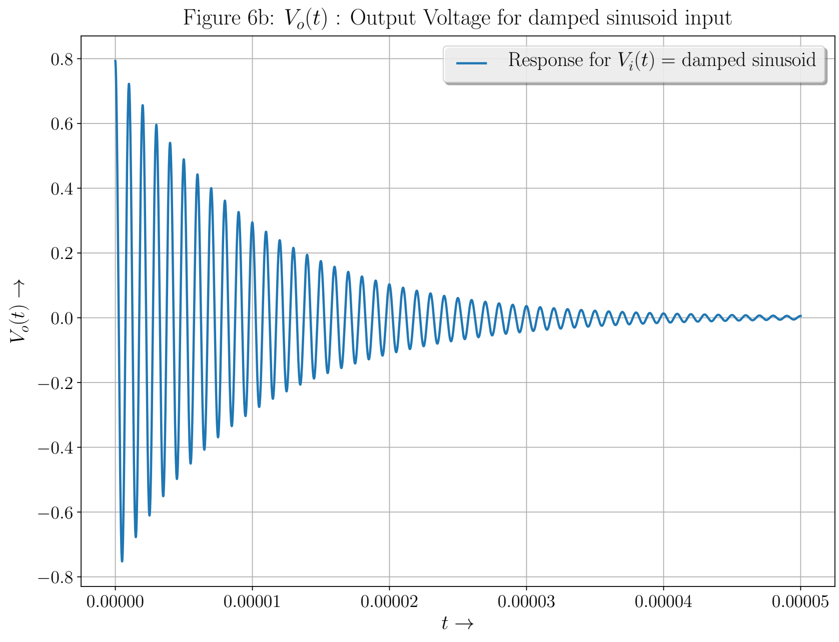
Figure 6a: $V_o(t)$ : Output Voltage for undamped sinusoid input through High Pass filter



### 4.0.6 Results and Discussion:

- As we observe the plot and the circuit that we know it is a high pass filter with bandwidth $\omega > 10^5$.
- So it allows only frequency component which is more than $10^5$ $rads^{-1}$ with gain of 0.8.
- Since $\omega_1 = 2000\pi < 10^5$ so its not passed through the filter,so only cos term passes through with gain of 0.8.
- Thats why at $t = 0$, $V_o(t) > 0$ since its cos function and Since its a undamped sinusoid, its amplitude does not change w.r.t time

```
In [17]: #plot of Vo(t) for damped sinusoidal input

         fig6 = figure()
         ax6 = fig6.add_subplot(111)
         ax6.plot(t2b,(Vot2b),label=r"Response for $V_{i}(t) = $ damped sinusoid")
         ax6.legend()
         title(r"Figure 6b: $V_{o}(t)$ : Output Voltage for damped sinusoid input")
         xlabel(r"$t \to $")
         ylabel(r"$ V_{o}(t) \to $")
         grid()
         savefig("Figure6b.jpg")
         show()
```

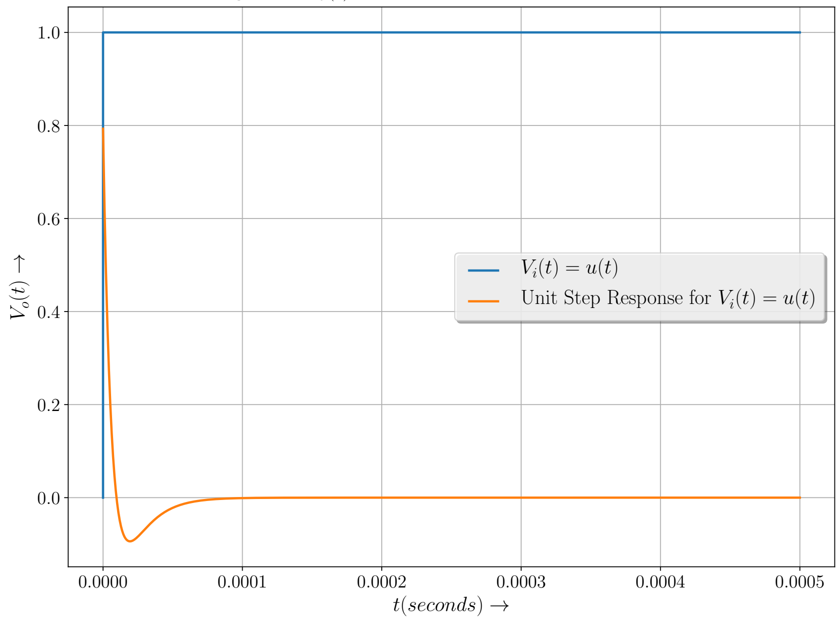


### 4.0.7 Results and Discussion:

- As we observe the plot and the circuit that we know it is a high pass filter with bandwidth $\omega > 10^5$.
- So it allows only frequency component which is more than $10^5$ $rads^{-1}$ with gain of 0.8.
- Since $\omega_1 = 2000\pi < 10^5$ so its not passed through the filter,so only cos term passes through with gain of 0.8.
- Thats why at $t = 0$, $V_o(t) > 0$ since its cos function and Since its a damped sinusoid, it decays as time grows.

```
In [18]: #Plot vo(t)  for unit step input

         fig7 = figure()
         ax7 = fig7.add_subplot(111)
         # Input - Unit step function
         ax7.step([t3b[0],t3b[-1]],[0,1],label = r"$V_{i}(t) = u(t)$")
         ax7.plot(t3b,(Vot3b),label=r"Unit Step Response for $V_{i}(t) = u(t)$")
         ax7.legend()
         title(r"Figure 7: $V_{o}(t) $ : Unit step response in time domain")
         xlabel(r"$ t (seconds) \to $")
         ylabel(r"$ V_{o}(t) \to $")
         grid()
         savefig("Figure7.jpg")
         show()
```

Figure 7: $V_o(t)$ : Unit step response in time domain



### 4.0.8 Results and Discussion:

- As we observe the plot that $V_o(t)$ decreases quickly from 0.8 to 0 and settles at 0.8 for after some time and remains constant.
- But interestingly it **crosses zero and goes negative** for some period of time.
- Because since the network is high pass filter, the output must not allow DC at steady state and since the input is unit step which is constant for $t > 0$ so at steady state $V_0(t)$ should be zero. Also since its Highpass filter its $Mean = 0$ which means $\int_0^\infty V_o(t)dt = 0$.
- So thats why we observe that the voltage becomes negative to make averge zero.
- And we determined Quality factor of the system as $Q = 0.453.. < \frac{1}{\sqrt{2}}$ Which implies that the gain of the system never exceeds DC Gain and always less than that. This observation comes by analysing the general form of second order transfer function.
- So with this we see that unit step response is always less that the DC Gain of 0.8 which is obtained by putting $s = 0$ in the $\mathcal{V}_\wr(s)$.
- Also If $Q < 0.5$ system is overdamped since damping factor $\zeta = \frac{1}{2Q} > 1$ for $Q < 0.5$ which means the unit step response will decrease slowly from 0.8 to 0 exponentially since its high pass filter and steady state voltage must be zero ,unlike immediately changing from 0.8 $to0$
- So this is also observed in the plot as it slowly decays from 0.8 to 0 and settles.

## 5 Conclusion :

- We successfully analysed circuits using laplace transform by solving analytically instead of numerical solutions using Symbolic python.