Assignment 7: Circuit Analysis using SymPy

J.Phani Jayanth - EE19B026

May 2, 2021

Abstract

In this assignment, we focus on Circuit Analysis using Symbolic Algebraic capabilities of Python in solving circuits using Laplace Transfrorms. We make use of SymPy and the Signals Toolbox of SciPy for this purpose. We analyse Low-Pass and High-Pass filters and their responses to sinusoidal and damped inputs.

Analysis of Low Pass Filter

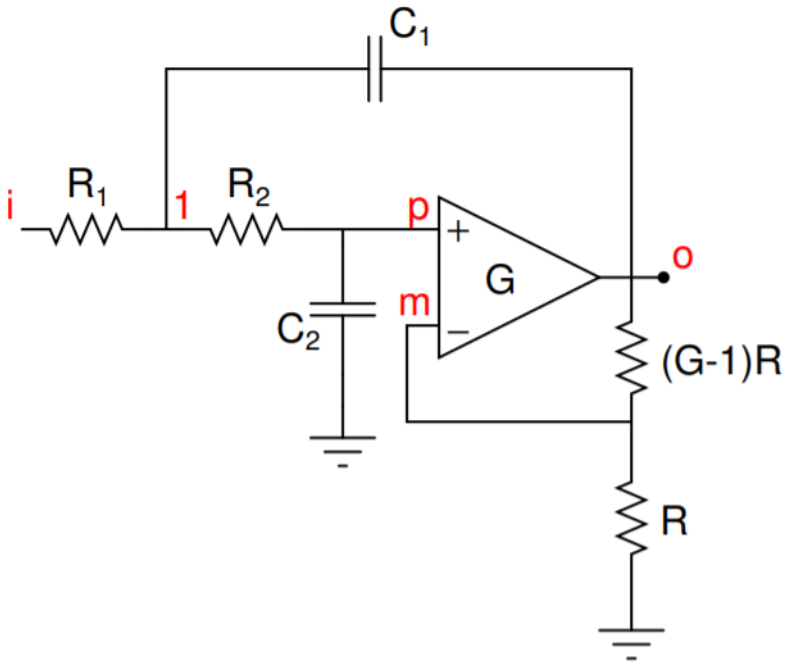

Figure 1: Low Pass Filter - 3dB Butterworth Filter with a cutoff frequency of $1/2\pi$ MHz

For the given circuit: G = 1.586, $R_1 = R_2 = 10\text{k}\Omega$ and $C_1 = C_2 = 1\text{nF}$
We can write the KCL equations for the above circuit as follows:

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

$$V_p = \frac{V_1}{1 + j\omega R_2 C_2}$$

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

$$\frac{V_i - V_1}{R_1} + \frac{V_p - V_1}{R_2} + j\omega C_1(V_o - V_1) = 0$$

These equations can be written in matrix form as follows: (A.V = b)

$$\begin{pmatrix} 0 & 0 & 1 & \frac{-1}{G} \\ \frac{-1}{1+sC_2R_2} & 1 & 0 & 0 \\ 0 & -G & G & 1 \\ \frac{-1}{R_1}+\frac{-1}{R_2}-sC_1 & \frac{1}{R_2} & 0 & sC_1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_p \\ V_m \\ V_o \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{V_i(s)}{R_1} \end{pmatrix}$$

We solve for the Voltage matrix as V = $A^{-1}b$.

We obtain the solution as a function in s-domain. So we need to extract the output in time domain, by finding the Inverse Laplace Transform of the s-domain output function. We achieve this using the Signals Toolbox of *SciPy* Library.

After defining the corresponding A and b matrices, we find V_o from the Voltage matrix. This will a rational polynomial in the variable 's'. To obtain the Impulse response of the system, we use the *lambdify* function to find the values of the Output for Input = 1 (i.e., Impulse function). The Impulse response so obtained will be in the frequency domain. This is plotted against frequency(ω) in loglog scale to get the Bode plot of Impulse response of the system as shown in Figure 2.

The code is as follows:

```
# Defining a Low Pass Filter
def lowpass(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)

# Impulse Response of the Low-Pass Filter in Frequency Domain
A,b,V = lowpass(10000,10000,1e-9,1e-9,1.586,1) # Vi = 1 - Impusle function in
    frequency domain
HSLP = V[3]
w = p.logspace(0,8,801) # Omega
ss = 1j*w
hf = lambdify(s,HSLP,'numpy') #Substituting omega values in the function HSLP
v = hf(ss)
p.loglog(w,abs(v),'r',lw=2) #Plotting Impulse response in the frequency domain
p.title("Impulse response of the Low-Pass filter in Frequency Domain")
p.xlabel("Frequency(\u03C9) $\longrightarrow$")
p.ylabel("|H(j\u03C9)| - Low Pass")
p.grid(True)
p.show()
```

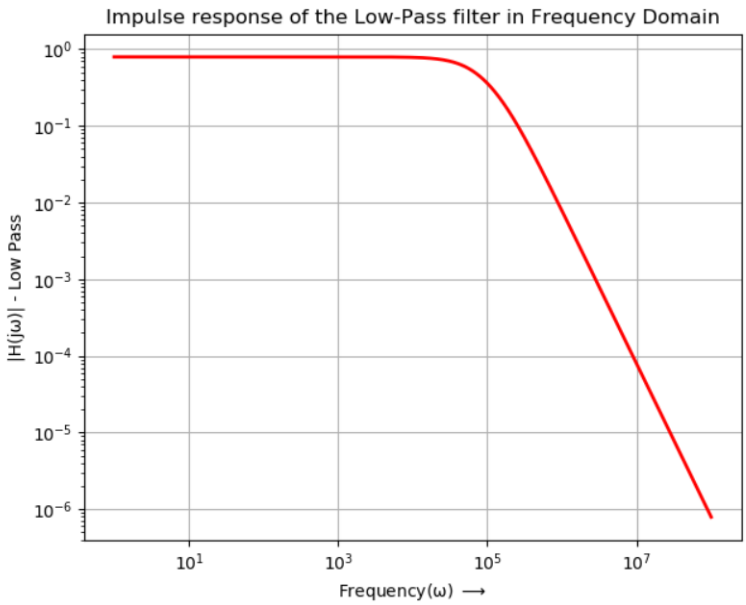

Figure 2: Frequency Response or Transfer Function of the Low-Pass Filter

The plot clearly depicts the Low-pass Filter nature of the circuit shown in Figure 1.

We now obtain the Step response of the Low-Pass Filter in frequency domain as well as in time domain. For this, we give the input function as $\frac{1}{s}$ - Step function in frequency domain. We obtain the Bode Plot of the Step response as mentioned above using *lambdify*. We plot the Step Response in Frequency Domain as shown in Figure 3

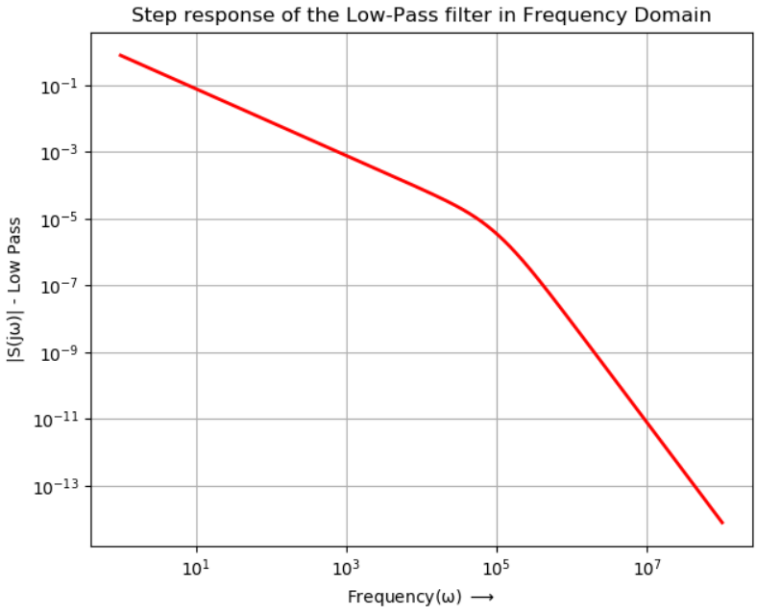

Figure 3: Step Response of the Low-Pass Filter in Frequency Domain

To obtain the step response of the filter in the time domain, we first define a function **S2LTI()** to convert the symbolic Transfer function (in variable s) to an LTI function. We then use *sp.impulse()* function to obtain the time-domain step response of the system. We plot it as shown in Figure 4

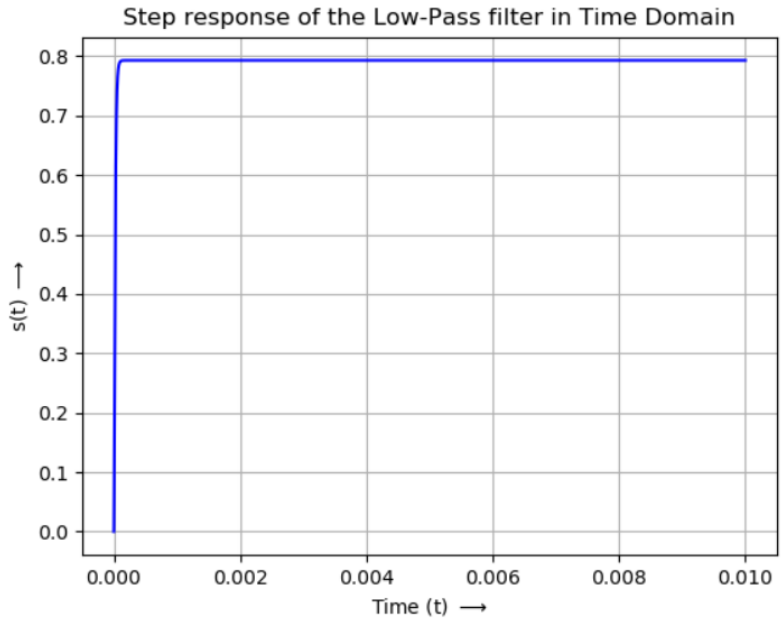

Figure 4: Step Response of the Low-Pass Filter in Time Domain

The code for this is as follows:

```
def S2LTI(func):
  num,den = fraction(func)
  num = Poly(num,s)
  den = Poly(den,s)
  num_coeffs = num.all_coeffs()
  den_coeffs = den.all_coeffs()
  num_coeffs = [float(num) for num in num_coeffs]
  den_coeffs = [float(den) for den in den_coeffs]
  return sp.lti(num_coeffs,den_coeffs)

# Step Response of the Low-Pass Filter in Time Domain
t,Vstep = sp.impulse(S2LTI(SSLP),None,t)
p.plot(t,Vstep,'b')
p.title("Step response of the Low-Pass filter in Time Domain")
p.xlabel("Time (t) $\longrightarrow$")
p.ylabel("s(t) $\longrightarrow$")
p.grid(True)
p.show()
```

Obtaining the Output of the Filter for a given Input

Input to the filter is given as:

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

We use the *sp.lsim()* function to obtain the output signal for the given input. The function *sp.lsim()* convolves the given time domain input signal with the impulse response of the Low-Pass filter system to give the output signal in time domain i.e., $v_o(t)$. This is plotted against time as shown in Figure 5.

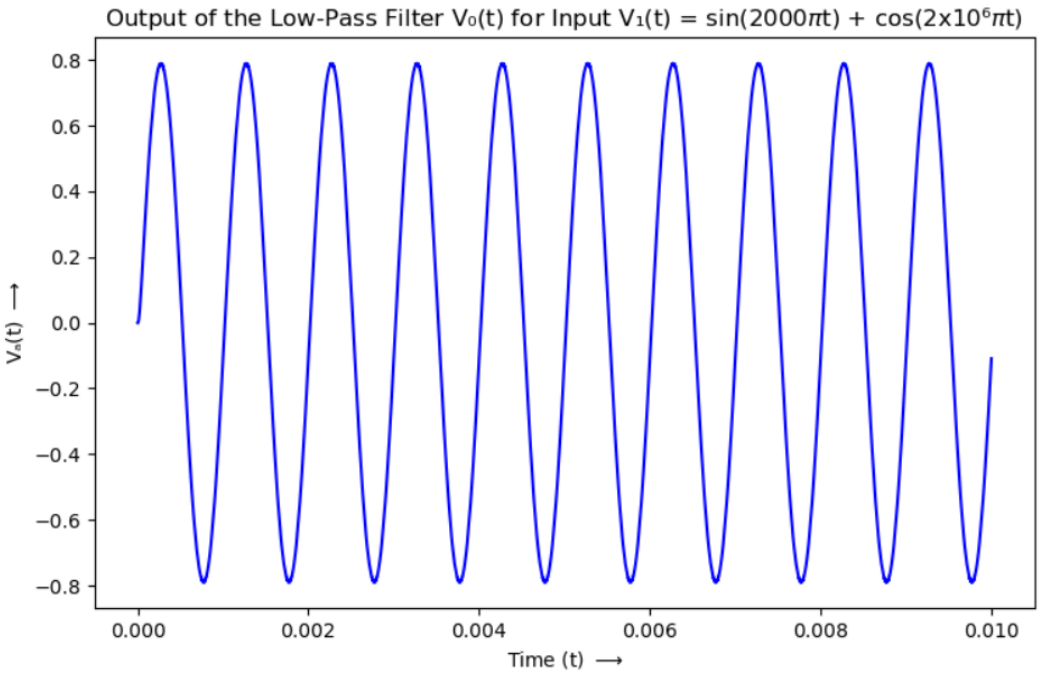

Figure 5: $v_o(t)$ - Output Voltage of the Low-Pass Filter Circuit for the given Input

The given input voltage $v_i(t)$ has two frequencies - 1 kHz and 1 MHz. Since the cutoff frequency of the filter is close to $1/2\pi$ MHz, the sinusoidal component with the frequency of 1 MHz is attenuated/blocked, and the output will only consist the sinusoidal component of frequency 1 kHz as shown in the above figure. The code is as follows:

```
Vi = (np.sin(2e3*np.pi*t)+np.cos(2e6*np.pi*t))*np.heaviside(t,1)
# Determining the Output Voltage Vo(t)
t,Vo,svecLP = sp.lsim(S2LTI(HSLP),Vi,t)
p.plot(t,Vo,'b')
p.title('Output of the Low-Pass Filter V\u2080(t) for Input V\u2081(t) = sin(2000
    $\pi$t) + cos(2x10\u2076$\pi$t)')
p.xlabel('Time (t) $\longrightarrow$')
p.ylabel('V\u2090(t) $\longrightarrow$')
p.show()
```

Analysis of High Pass Filter

The system shown Figure 6 represents a High-Pass filter with a cutoff frequency around $1/2\pi$ MHz

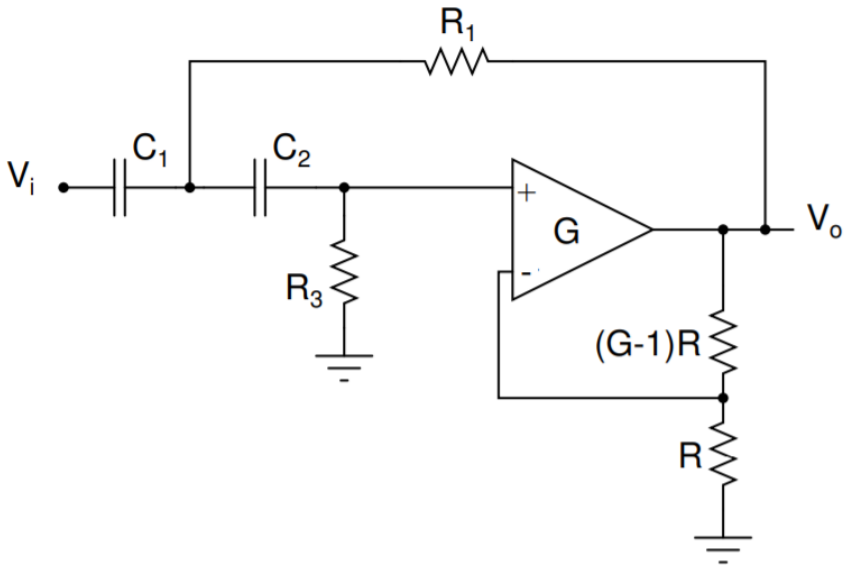

Figure 6: High Pass Filter with a cutoff frequency of $1/2\pi$ MHz

For the given circuit: G = 1.586, $R_1 = R_3 = 10\text{k}\Omega$ and $C_1 = C_2 = 1\text{nF}$
We obtain the Nodal Analysis matrix for the above circuit as follows: A.V = b

$$\begin{pmatrix} 0 & -1 & 0 & \frac{1}{G} \\ \frac{sC_2R_3}{1+sC_2R_3} & 0 & -1 & 0 \\ 0 & G & -G & 1 \\ \frac{-1}{R_1} - s(C_1+C_2) & 0 & sC_2 & \frac{1}{R_1} \end{pmatrix} \begin{pmatrix} V_1 \\ V_m \\ V_p \\ V_o \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -sC_1V_i \end{pmatrix}$$

We solve for the Voltage matrix as V = $A^{-1}b$.

As mentioned before, we obtain the solution as a function in s-domain. So we need to extract the output in time domain, by finding the Inverse Laplace Transform of the s-domain output function.
After defining the corresponding A and b matrices, we find V_o from the Voltage matrix. This will a rational polynomial in the variable 's'. Again, to obtain the Impulse response of the system, we use the *lambdify* function to find the values of the Output for Input = 1 (i.e., Impulse function). The Impulse response so obtained will be in the frequency domain. This is plotted against frequency(ω) in loglog scale to get the Bode plot of Impulse response of the High-Pass Filter system as shown in Figure 7.

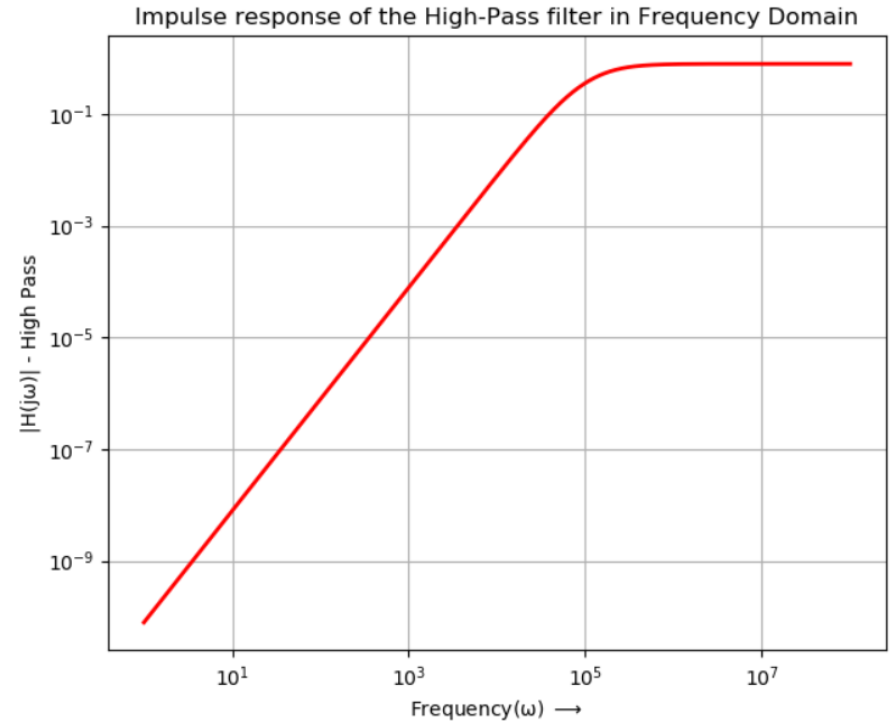

Figure 7: Impulse Response of the High Pass Filter in Frequency Domain

Similarly, we also obtain the Step Response (Response to Input = $\frac{1}{s}$ in both Frequency and Time Domain, as discussed before. These are plotted as shown in Figure 8 and Figure 9.

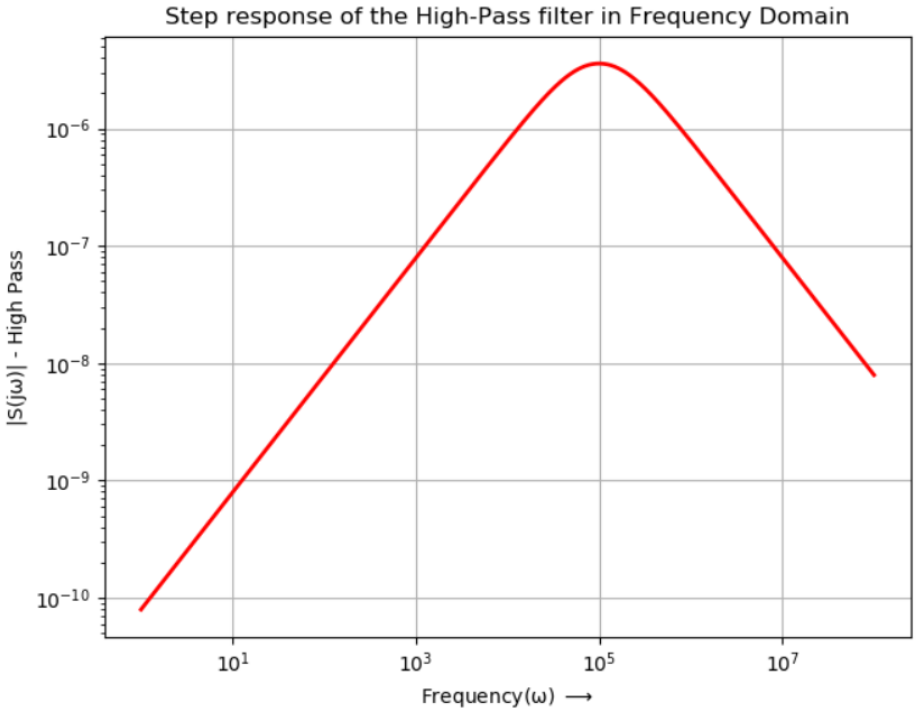

Figure 8: Step Response of the High Pass Filter in Frequency Domain

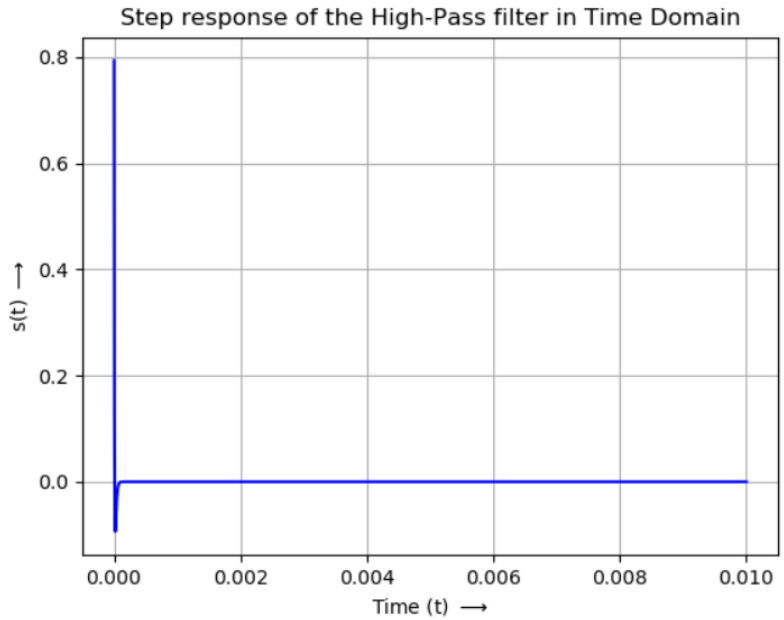

Figure 9: Step Response of the High Pass Filter in Time Domain

The code is as follows:

```
def highpass(R1,R3,C1,C2,G,Vi):
  s = symbols('s')
  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)

# Impulse Response of High-Pass Filter in Frequency Domain
A,b,V = highpass(1e4,1e4,1e-9,1e-9,1.586,1) # Vi = 1 - Impusle function in
    frequency domain
HSHP = V[3]
w = p.logspace(0,8,801) # Omega
ss=1j*w
hf = lambdify(s,HSHP,'numpy') #Substituting omega values in the function HSHP
v=hf(ss)
p.loglog(w,abs(v),'r',lw=2) #Plotting Impulse response in the frequency domain
p.title("Impulse response of the High-Pass filter in Frequency Domain")
p.xlabel("Frequency(\u03C9) $\longrightarrow$")
p.ylabel("|H(j\u03C9)| - High Pass")
p.grid(True)
p.show()
```

The code for obtaining the Step Response is as follows:

```
# Step Response of the High-Pass Filter in Frequency Domain
A,b,V = highpass(10000,10000,1e-9,1e-9,1.586,1/s) # Vi = 1/s - Step function in
    frequency domain
SSHP = V[3]
w = p.logspace(0,8,801) # Omega
ss = 1j*w
hf = lambdify(s,SSHP,'numpy') #Substituting omega values in the function SSLP
v = hf(ss)
p.loglog(w,abs(v),'r',lw=2) #Plotting Step response in the frequency domain
p.title("Step response of the High-Pass filter in Frequency Domain")
p.xlabel("Frequency(\u03C9) $\longrightarrow$")
p.ylabel("|S(j\u03C9)| - High Pass")
p.grid(True)
p.show()

# Step Response of the High-Pass Filter in Time Domain
t,Vstep = sp.impulse(S2LTI(SSHP),None,t)
p.plot(t,Vstep,'b')
p.title("Step response of the High-Pass filter in Time Domain")
p.xlabel("Time (t) $\longrightarrow$")
p.ylabel("s(t) $\longrightarrow$")
p.grid(True)
p.show()
```

Obtaining the Output of the Filter for a given Input

We use the following input again for the High-Pass Filter:

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

We use the *sp.lsim()* function to obtain the output signal $v_o(t)$ for the given input as mentioned before. This is plotted against time as shown in Figure 10.
Since the cutoff frequency of the filter is close to $1/2\pi$ MHz, the sinusoidal component with the frequency of 1 kHz is attenuated/blocked, and the output will only consist of the high frequency sinusoidal component of frequency 1 MHz as shown in Figure 10. The code is as follows:

```
# Determining the Output Voltage Vo(t)
t2 = np.arange(0,5e-6,5e-8)
Vi = (np.sin(2e3*np.pi*t2)+np.cos(2e6*np.pi*t2))*np.heaviside(t2,1)
t2,Vo,svecHP = sp.lsim(S2LTI(HSHP),Vi,t2)
p.plot(t2,Vo,'b')
p.title('Output of the High-Pass Filter V\u2080(t) for Input V\u2081(t) = sin
    (2000$\pi$t) + cos(2x10\u2076$\pi$t)')
p.xlabel('Time (t) $\longrightarrow$')
p.ylabel('V\u2090(t) $\longrightarrow$')
p.show()
```

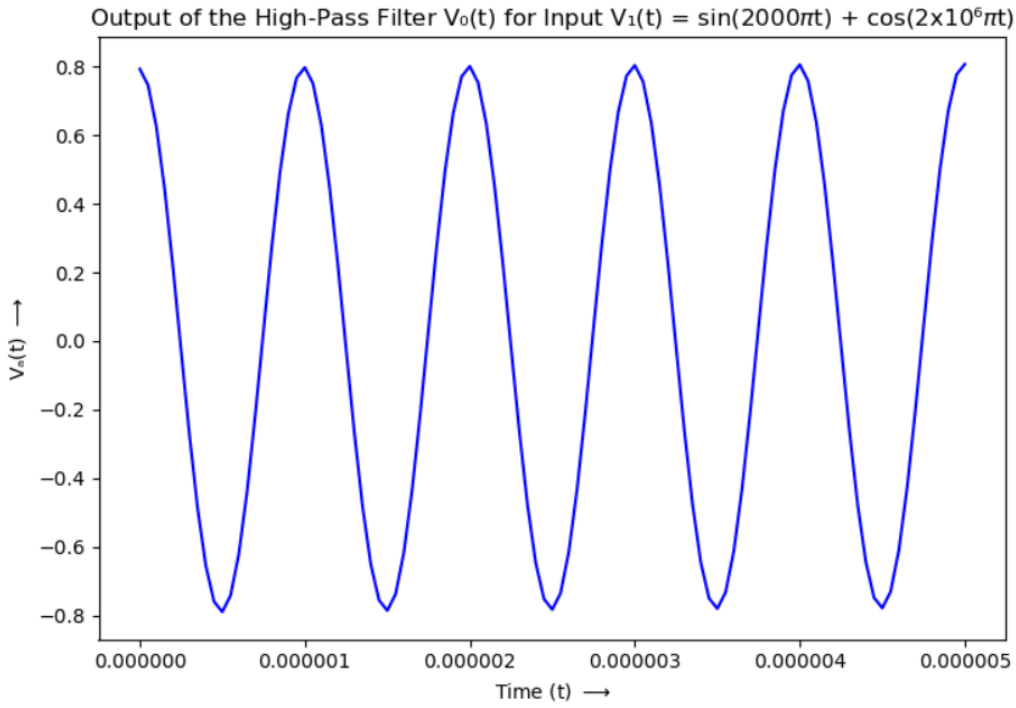

Figure 10: $v_o(t)$ - Output Voltage of the High-Pass Filter Circuit for the given Input

Response of the High-Pass Filter to Damped Sinusoidal Inputs

We consider two damped sinusoidal inputs - one of low frequency (1 kHz) and another of high frequency (1 MHz) as follows:

$$v_{i,dampedLF} = sin(2000t)e^{-500t}u(t)$$

$$v_{i,dampedHF} = sin(2 \times 10^6 t)e^{-500t}u(t)$$

Impulse response of the High-Pass filter is convolved with the inputs in time domain using *sp.lsim*, to find the corresponding outputs in time domain. These are plotted against time as shown in Figure 11 and Figure 12.

For the damped sinusoidal input of frequency 1kHz, output converges to zero. This is expected, because the system is a High-Pass Filter and hence blocks/attenuates low frequency signals.

For the damped sinusoidal input of frequency 1MHz, output follows the input as expected, because the system is a High-Pass Filter and frequencies above $1/2\pi$ MHz are allowed to pass through. The amplitude is not exactly the same since the gain of the filter at a frequency of 1MHz is not unity.

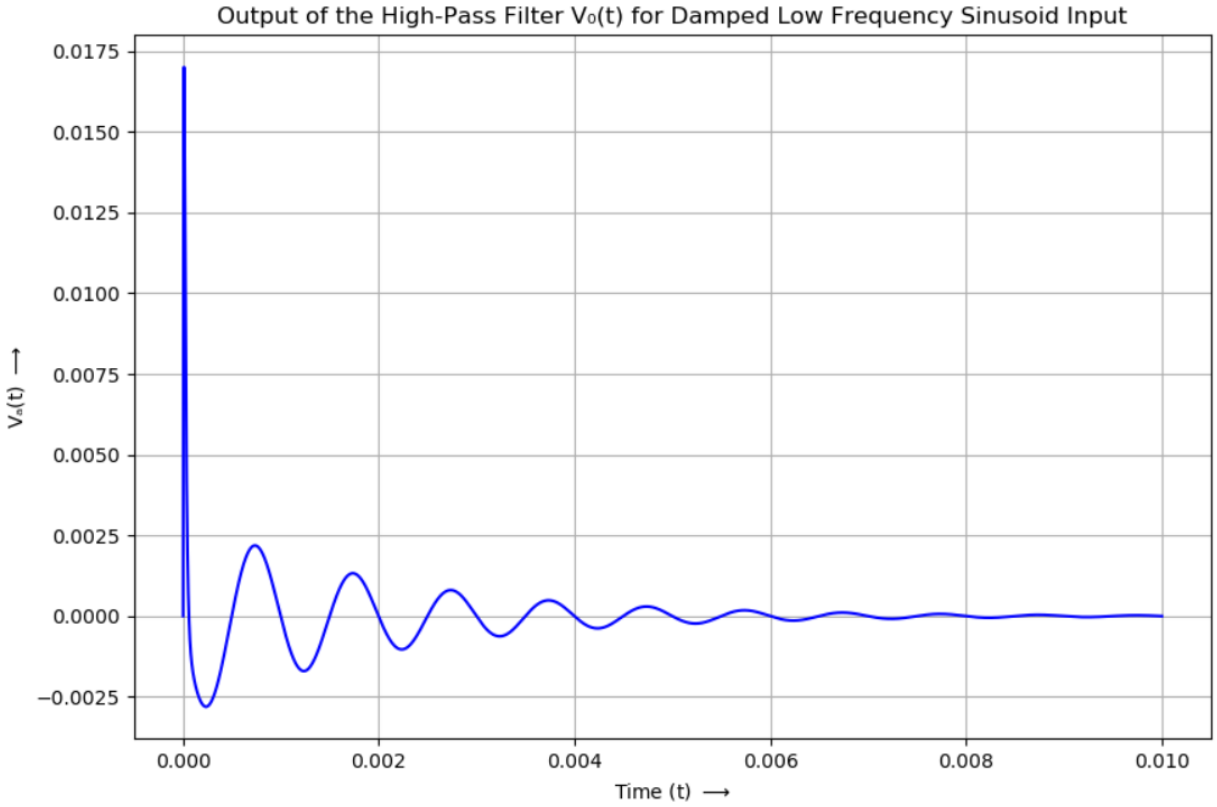

Figure 11: Output of the High Pass Filter for Damped Input Sinusoid of 1 kHz Frequency

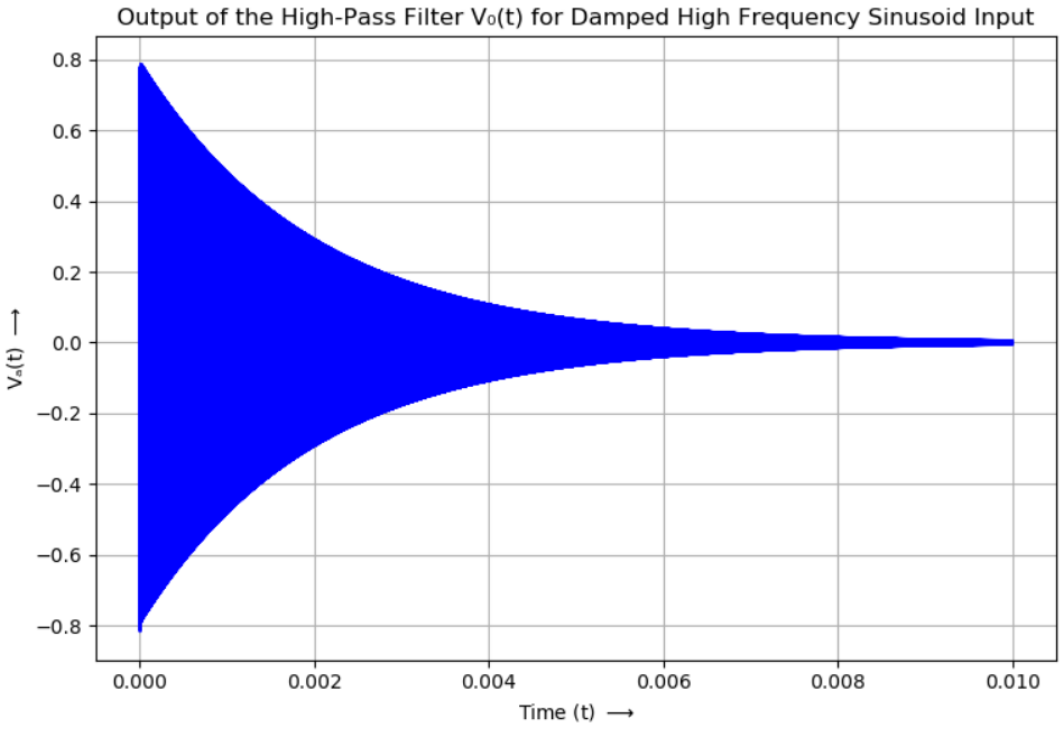

Figure 12: Output of the High Pass Filter for Damped Input Sinusoid of 1 MHz Frequency

The code is as follows:

```
Vi_LFdamp = (np.sin(2e3*PI*t))*np.exp(-500*t)*np.heaviside(t,1)
Vi_HFdamp = (np.sin(2e6*PI*t))*np.exp(-500*t)*np.heaviside(t,1)

# Output for Damped Sinusoid of Low frequency
t,Vo,svecHP = sp.lsim(S2LTI(HSHP),Vi_LFdamp,t)
p.plot(t,Vo,'b')
p.title('Output of the High-Pass Filter V\u2080(t) for Damped Low Frequency
    Sinusoid Input')
p.xlabel('Time (t) $\longrightarrow$')
p.ylabel('V\u2090(t) $\longrightarrow$')
p.grid(True)
p.show()

# Output for Damped Sinusoid of High frequency
t,Vo,svecHP = sp.lsim(S2LTI(HSHP),Vi_HFdamp,t)
p.plot(t,Vo,'b')
p.title('Output of the High-Pass Filter V\u2080(t) for Damped High Frequency
    Sinusoid Input')
p.xlabel('Time (t) $\longrightarrow$')
p.ylabel('V\u2090(t) $\longrightarrow$')
p.grid(True)
p.show()
```

Conclusion

We have learned how to use various tools Python includes for Analysis of complex circuits, such as Signals Toolbox of SciPy, Symbolic Algebraic capabilities of Python (SymPy), etc. We analysed Low-Pass and High-Pass filters and plotted their Impulse and Step Responses. We have also plotted and observed the responses of the filters to undamped and damped inputs of varied frequencies.