Assignment 7

Kathir Pagalavan EE20B056

April 4, 2022

Aim

To analyse the given low pass and high pass filter circuits using SymPy python module.

Low Pass Filter

The low pass filter that we use gives the following matrix equation after simplification of the modified nodal equations.

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

The above matrices are put as SymPy matrices in the following code and solved.

```python
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,
    b=Matrix([0,0,0,-Vi/R1])
    V=A.inv()*b
    return(A,b,V)
```

The plot for the magnitude of the transfer function (magnitude bode plot) is as shown below:

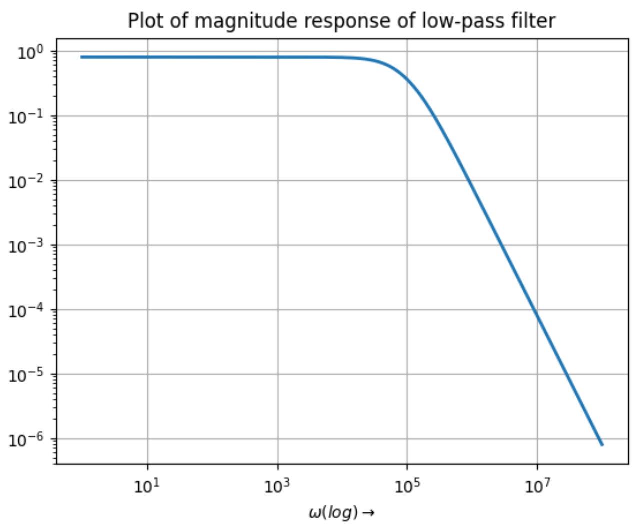

Figure 1: Magnitude bode plot of the low pass filter

High Pass Filter

The high pass filter we use gives the following matrix equations after simplification of the modified nodal equations

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

The above matrices are put as SymPy matrices in the following code and solved.

```
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*
    b = Matrix([0,0,0,-Vi*s*C1])
    V = A.inv()*b
    return A,b,V
```

The plot for the magnitude of the transfer function (magnitude bode plot) is as shown below:

SymPy function to SciPy LTI system

The python code snippet is as shown: SymPy functions in terms of the SymPy symbol 's' must be converted to SciPy LTI system by extracting co-

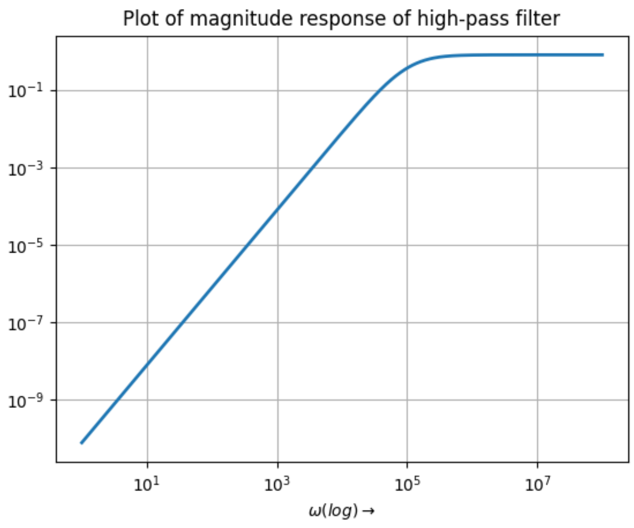

Figure 2: Magnitude bode plot of the high pass filter

efficients from the numerator and denominator; which is done in the function *coeffextract()*

```
def coeffextract(Vo):
    s=symbols('s')
    top, bot = [[float(i) for i in Poly(j, s).all_coeffs()] for j in Vo.as_numer_de
    return(sp.lti(top,bot))
```

Step Response Of the Low Pass Filter

We find the step response of the low-pass filter by using the *sp.step()* function in the following code

```
A,b,V=lowpass(10000,10000,1e-9,1e-9,1.586,1)
Vo=V[3]
H1=coeffextract(Vo)

t=p.linspace(0,1e-3,1001)
t,y1=sp.step(H1,None,t)
p.title('Plot of unit step response of low-pass filter')
p.plot(t,y1,label='Vo')
p.plot(t,p.ones(len(t)),label='Vi')
p.xlabel(r'$t\rightarrow$')
p.legend()
p.show()
```

The plot for the step response of the low pass circuit is as shown:

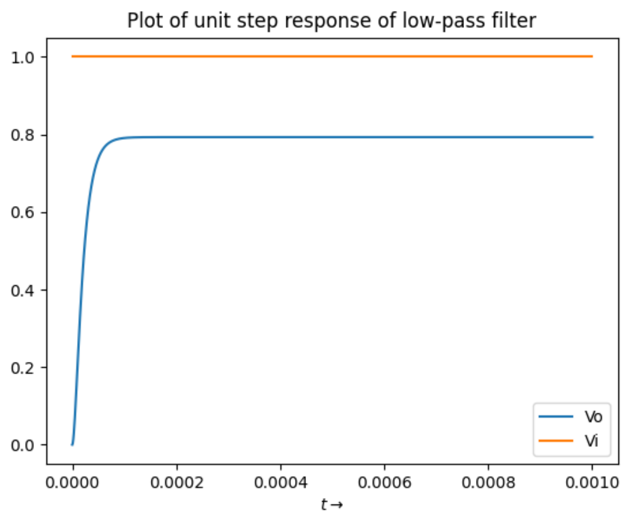

Figure 3: Step response of low pass filter.

Response to Sum Of Sinusoids

When the input is a sum of sinusoids, as given

$$V_i(t) = (\ \sin(2000\pi t) + \cos\big(2 * 10^6\pi t\big)\ )u_o(t)\ Volts$$

Then the output response for the lowpass filter is calculated as

```
t=p.linspace(0,1e-2,100001)
Vi=p.sin(2000*p.pi*t) + p.cos(2e6*p.pi*t)
t,y2,svec = sp.lsim(H1,Vi,t)
p.title('Plot of low-pass filter response to multiple sinusoid')
p.plot(t,y2,label='Vo',color='r')
p.plot(t,Vi,label='Vi',color='b',alpha=0.5)
p.xlabel(r'$t\rightarrow$')
p.legend()
p.show()
```

The output response to the sum of sinusoids for the low pass filter is as shown:

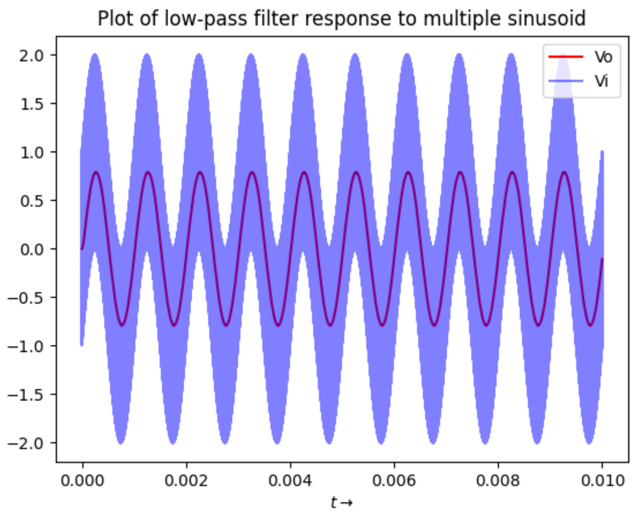

Figure 4: Output response to the sum of sinusoids.

Response to Damped Sinusoids

We pass two damped sinusoids of low and high frequencies to the high pass filter and observe what happens.

$$V_i(t) = e^{-1000t}(\ \cos(1000\pi t)\ )u_o(t)\ Volts$$

High frequency,

$$V_i(t) = e^{-1000t}(\ \cos\left(1 * 10^6 \pi t\right)\ )u_o(t)\ Volts$$

```
t=p.linspace(0,1e-2,1001)
Vi=p.sin(1e3*p.pi*t)*p.exp(-damp*t)
t,y3,svec=sp.lsim(H2,Vi,t)
p.title('Plot of high-pass filter response to low frequency damped sinusoid')
p.plot(t,y3,label='Vo')
p.plot(t,Vi,label='Vi')
p.xlabel(r'$t\rightarrow$')
p.legend()
p.show()


t=p.linspace(0,1e-5,1001)
Vi=p.sin(1e6*p.pi*t)*p.exp(-damp*t)
t,y4,svec=sp.lsim(H2,Vi,t)
p.title('Plot of high-pass filter response to high frequency damped sinusoid')
```

```
p.plot(t,y4,label='Vo')
p.plot(t,Vi,label='Vi')
p.xlabel(r'$t\rightarrow$')
p.legend()
p.show()
```

The output responses for both cases in a high pass filter is as shown:

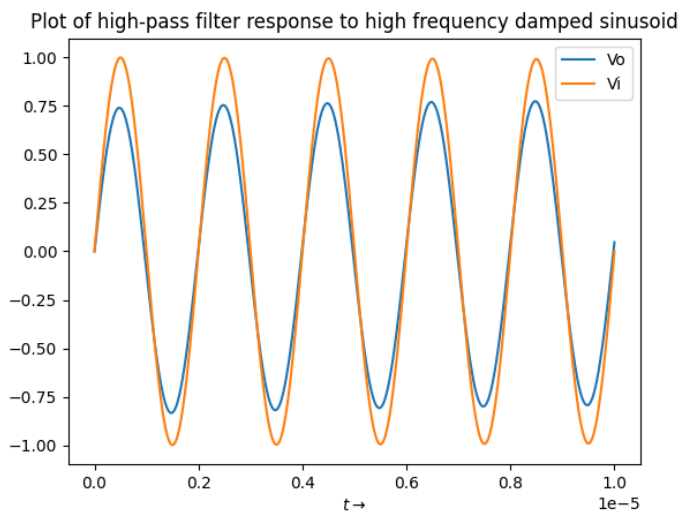

Figure 5: Output response of a high frequency damped sinusoid

Step Response of High Pass Filter

As done previously, to find the step response of the high-pass filter we use *sp.step()* function.

```
t=p.linspace(0,1e-3,1001)
t,y1=sp.step(H2,None,t)
p.title('Plot of unit step response of high-pass filter')
p.plot(t,y1,label='Vo')
p.plot(t,p.ones(len(t)),label='Vi')
p.xlabel(r'$t\rightarrow$')
p.legend()
p.show()
```

The plot of the step response for a high pass filter is as shown:

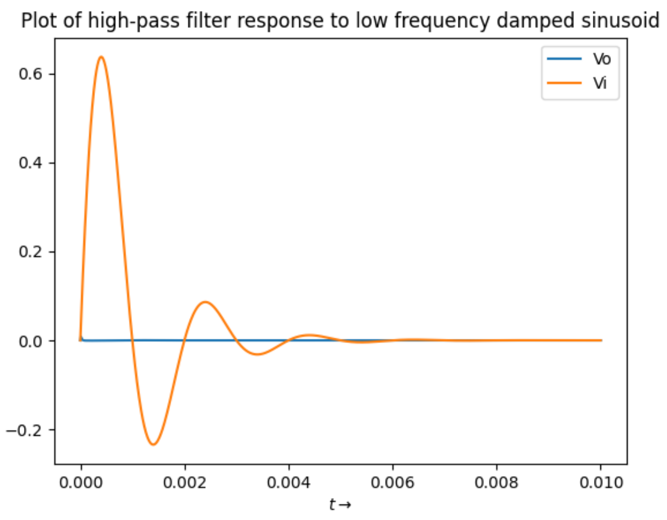

Figure 6: Output response of a low frequency damped sinusoid

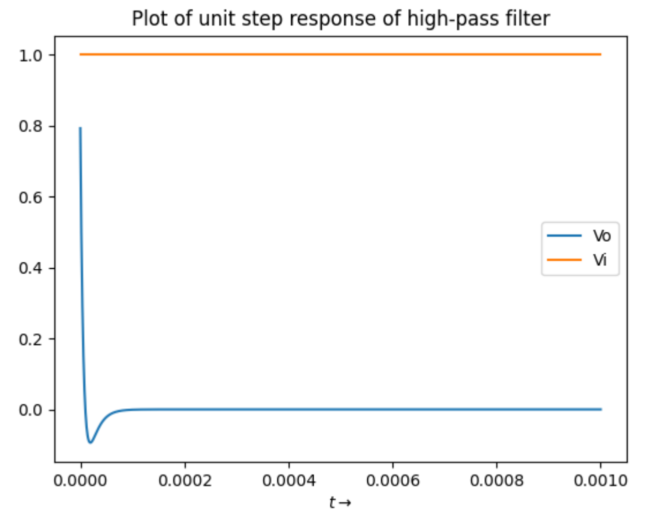

Figure 7: Step response for a high pass filter

The unit step response, as expected is high at t=0 when there is an abrupt change in the input. Since there is no other change at large time values outside the neighbourhood of 0, the Fourier transform of the unit step has high values near 0 frequency, which the high pass filter attenuates.

The unit step has almost all near zero frequency components which are bound to get filtered out by the high-pass filter.The spike near zero explains the only large jump in the unit step function

Conclusion

The Sympy module in conjunction with SciPy signal toolbox has been greatly useful in analysing complicated systems by the simple idea of putting a 'variable' symbol to use.