Assignment No. 7

Srivenkat A (EE18B038)

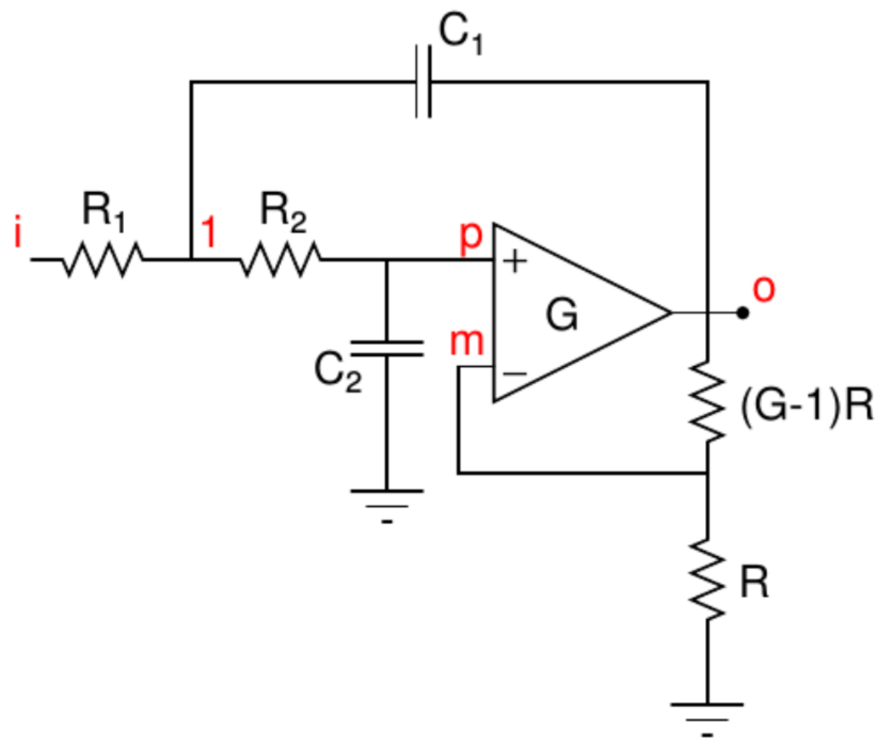
April 8, 2020

Introduction

The assignment focusses on using **sympy** library to analyse the laplace trasform of transfer functions generated by RLC and opamp based circuits

Low Pass filter circuit

Given is the circuit of a low pass filter:



On applying Kirchoff's laws and analysing the circuit, we get the output laplace transform from the following code:

```
def lowpass(R1,R2,C1,C2,G,Vi):
    A = sym.Matrix([[0,0,1,-1/G],[-1/(1+s*R2*C2),1,0,0],[0,-G,G,1],[-1/R1-1/R2-s*C1
    b = sym.Matrix([0,0,0,-Vi/R1])
    V = A.inv()*b
    return (A,b,V)
```

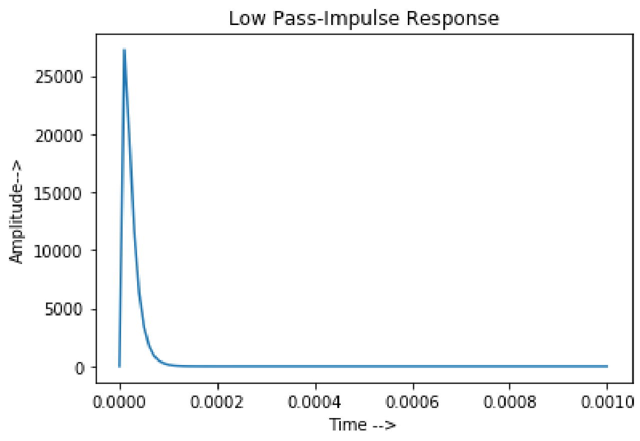
We are given values $G = 1.586, R_1 = R_2 = 10k\Omega, C_1 = C_2 = 10pF$
For these values, the circuit behaves as a 3dB **Butterworth filter**.
We get the impulse response of the circuit from

```
Vi = 1
A,b,V = lowpass(10000,10000,1e-9,1e-9,1.586,Vi)
Vl=V[3]
```

The following code converts the laplace impulse response into a polynomial form:

```
def Hcalc(Lap):
    Lap = Lap.simplify()
    Lap = Lap.expand()
    num,den = sym.fraction(Lap)
    num_deg = sym.degree(num)
    den_deg = sym.degree(den)
    num_c = np.empty(num_deg+1)
    den_c = np.empty(den_deg+1)
    for i in range(num_deg+1):
        num_c[i] = num.coeff(s,i)
    for i in range(den_deg+1):
        den_c[i] = den.coeff(s,i)
    num_c = num_c[::-1]
    den_c = den_c[::-1]
    H = sig.lti(num_c,den_c)
    return(H)
```

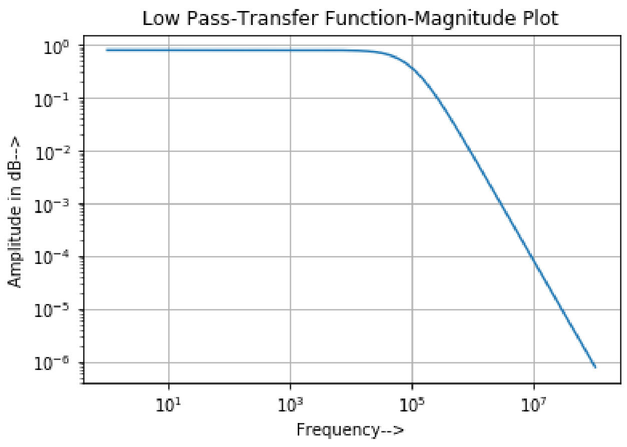
On plotting the impulse response, we get



We call the plotbode function to get the bode plots for the impulse response:

```
def Hcalc(Lap):
    Lap = Lap.simplify()
    Lap = Lap.expand()
    num,den = sym.fraction(Lap)
    num_deg = sym.degree(num)
    den_deg = sym.degree(den)
    num_c = np.empty(num_deg+1)
    den_c = np.empty(den_deg+1)
    for i in range(num_deg+1):
        num_c[i] = num.coeff(s,i)
    for i in range(den_deg+1):
        den_c[i] = den.coeff(s,i)
    num_c = num_c[::-1]
    den_c = den_c[::-1]
    H = sig.lti(num_c,den_c)
    return(H)
```

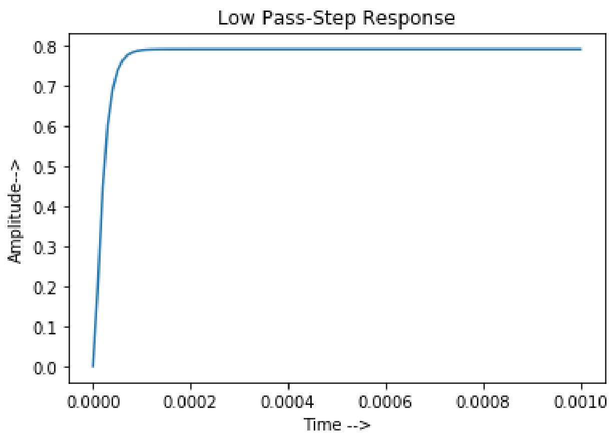
We get the magnitude plot as



To analyse the step response, we use the folowing code:

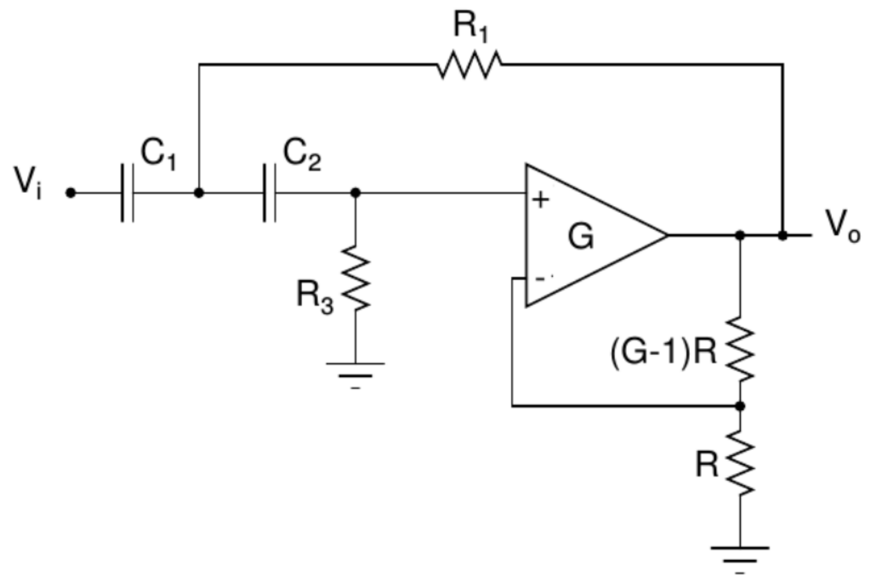
```
Vi = 1/s
A,b,V = lowpass(10000,10000,1e-9,1e-9,1.586,Vi)
Vl_step=V[3]
Vl_step = Hcalc(Vl_step)
t,Vl_step = sig.impulse(Vl_step,None,t)
plotsig(Vl_step,"Low Pass-Step Response",t)
```

We get the step response as



High Pass filter circuit

Given is the circuit of a high pass filter:



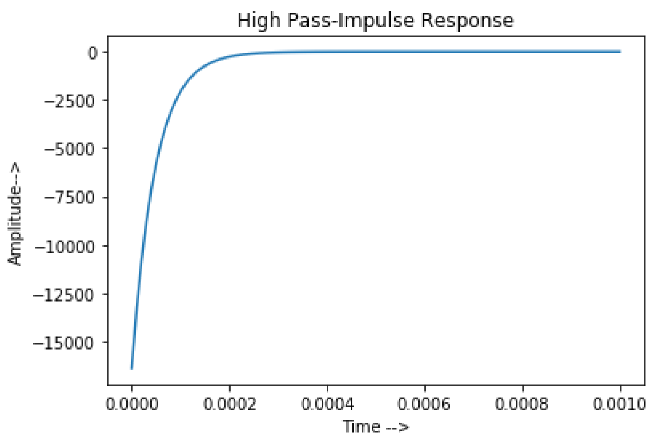
On applying Kirchoff's laws and analysing the circuit, we get the output laplace transform from the following code:

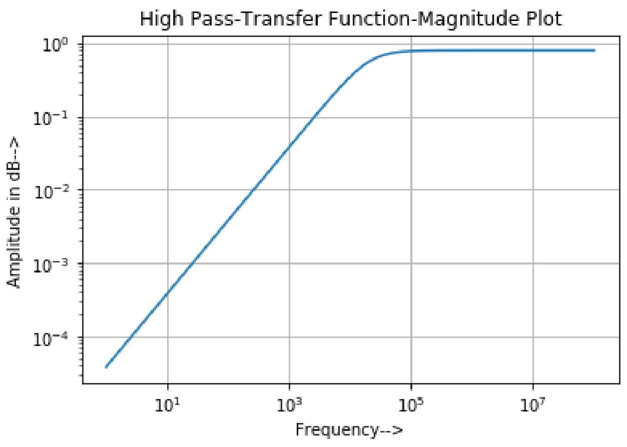
```
def highpass(R1,R3,C1,C2,G,Vi):
    A = sym.Matrix([[0,0,1,-1/G],[-1/(1+(1/s*C2*R3)),1,0,0],[0,-G,G,1],[-s*C1-s*C2-1
    b = sym.Matrix([0,0,0,-Vi*s*C1])
    V = A.inv()*b
    return (A,b,V)
```

We are given values $G = 1.586, R_1 = R_2 = 10k\Omega, C_1 = C_2 = 10pF$
For these values, the circuit behaves as a **High Pass filter**.
We get the impulse response of the circuit from

```
Vi = 1
A,b,V = highpass(10000,10000,1e-9,1e-9,1.586,Vi)
Vh=V[3]
```

On plotting the impulse response and its magnitude response, we get

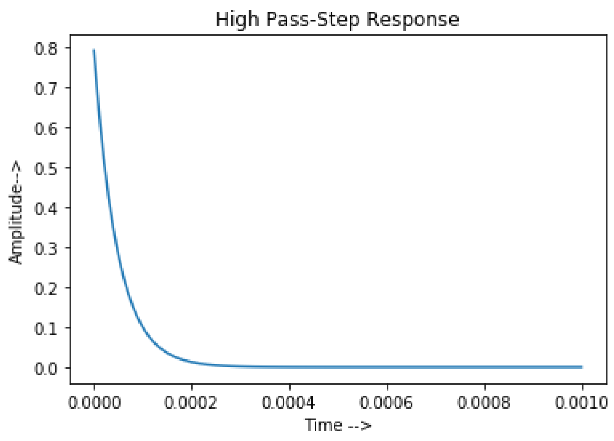




To analyse the step response, we use the following code:

```
Vi = 1/s
A,b,V = highpass(10000,10000,1e-9,1e-9,1.586,Vi)
Vh_step=V[3]
Vh_step = Hcalc(Vh_step)
t,Vh_step = sig.impulse(Vh_step,None,t)
plotsig(Vh_step,"High Pass-Step Response",t)
```

We get the step response as



Mixed frequency response

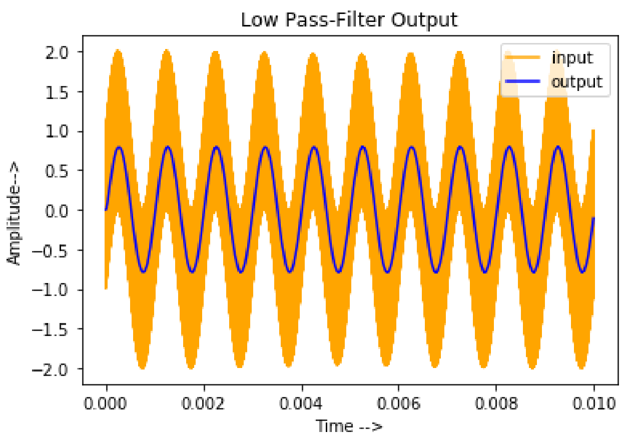
We analyse the response for the following mixed frequency input

$$V_i(t) = sin(2000\pi t) + cos(2000000\pi t)$$

For the low pass filter: The following code convolves the input laplace transorm with the impulse response to get the output response:

```
Vl_input = np.sin(2e3*ma.pi*t) + np.cos(2e6*ma.pi*t)
t,Vl_output,svec = sig.lsim(Vl,Vl_input,t)
plotsig(Vl_output,"Low Pass-Filter Output",t,Vl_input)
```

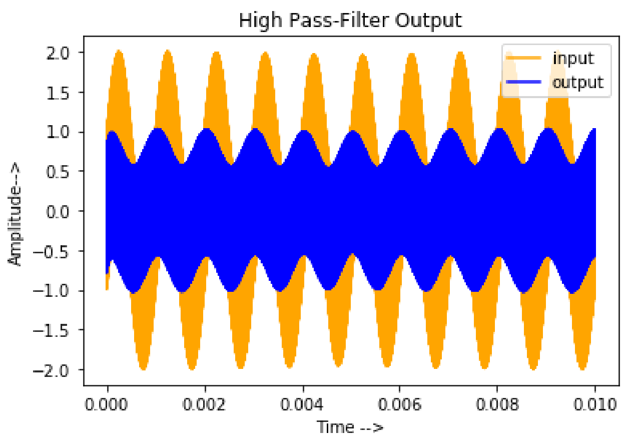
We get the output response for the lowpass filter to be: From the plot it



is visible that on passing the sum of two different sinusoids of varying frequencies and phase, the lowpass filter has allowed only the lower frequency 2000Hz component to pass through. So, the high frequency ringing is eliminated.
For the high pass filter: The following code convolves the input laplace transorm with the impulse response to get the output response:

```
Vh_input = np.sin(2e3*ma.pi*t) + np.cos(2e6*ma.pi*t)
t,Vh_output,svec = sig.lsim(Vh,Vh_input,t)
plotsig(Vh_output,"High Pass-Filter Output",t,Vh_input)
```

We get the output response for the highpass filter to be: From the plot,



we can conclude that, as opposed to the lowpass filter which only allowed the base carrier sinusoid to pass through, here, only the high frequency vibrations are passed through. The amplitude modulated part gets converted into the high frequency signal. The low frequency component is eliminated.

Response of the high pass filter to a Damped sinusoid

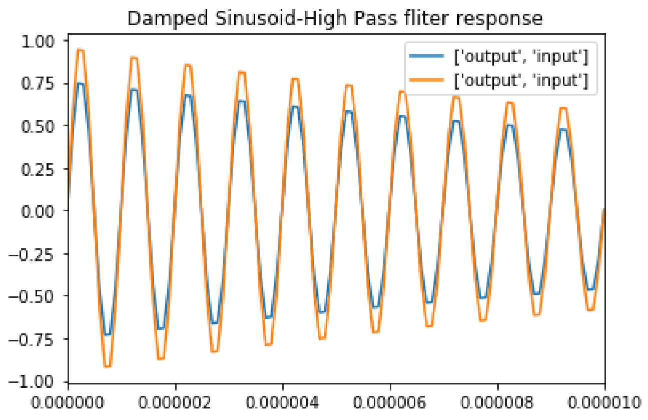
We give the following input to the circuit

$$V_i(t) = sin(2000000\pi t)e^{-50000t} \tag{1}$$

The following code calculated the output response:

```
Vh_ds = np.sin(2e6*np.pi*t)*np.exp(-5e4*t)
t,Vh_output,svec = sig.lsim(Vh,Vh_ds,t)
```

On plotting the obtained response, we get It is observed from the plot that the output has a reduced magnitude than the input. So, the high pass filter, apart from allowing only high frequency components to pass through,



also scales them down in amplitude. The damping term, doesn't affect the filtering action. The output damps in the same way as the input.

Conclusion

The sympy library provides a convenient way to analyse transfer functions and it can be used as a symbolic computation engine to solve equations involving variables. For the same opamp based circuit, by exchanging the resistors and capacitors in the input system, the circuit behaves as a low-pass and high-pass circuit respectively. The step response and mix frequency response for the above filters were analysed. The response of the high pass filter to a damped sinusoid input was analysed.