Assignment No 7: Circuit Analysis using sympy

Aravint Annamalai, EE18B125

March 17, 2020

1 Abstract

In this assignment, I am going to analyse electrical circuits using the `sympy` library. This library makes the analysis of circuits, especially in the s-domain, relatively much easier compared to `scipy.signal` library as we can define symbols (like s as far as analysis in Laplace domain is concerned) and hence, we can express the transfer functions and output in the s-domain itself. Here, we are going to analyse two opamp circuits(one acts as a high-pass filter and the other acts as a lowpass filter) in the s-domain, plotting the Bode plot of their transfer functions and plotting the outputs in time domain for sinusoidal signals of appropriate frequencies.

2 Introduction

As discussed earlier, I am going to explore different features of `sympy` library in this assignment. Here, many symbols can be created as we wish. In this assignment, since we are going to analyse circuits in the s-domain, the symbol which we are going to use the most is 's', which can be defined as follows:

```
s = symbols('s')
```

So, wherever `s` is referred, the symbol s is put in that place. `sympy` library can also be used to declare matrices in the s-domain as well. This can be done using the `Matrix` method and the matrix defined can be initialised in the exact same way as we define `numpy` matrices. Using the `inv` method, the matrix can be inverted as well and thus, we can solve equations represented in the form of matrices in the s-domain as well using `sympy`

3 Analysis of first circuit (lowpass filter)

3.1 Defining the circuit

The first circuit is as shown in the figure below:

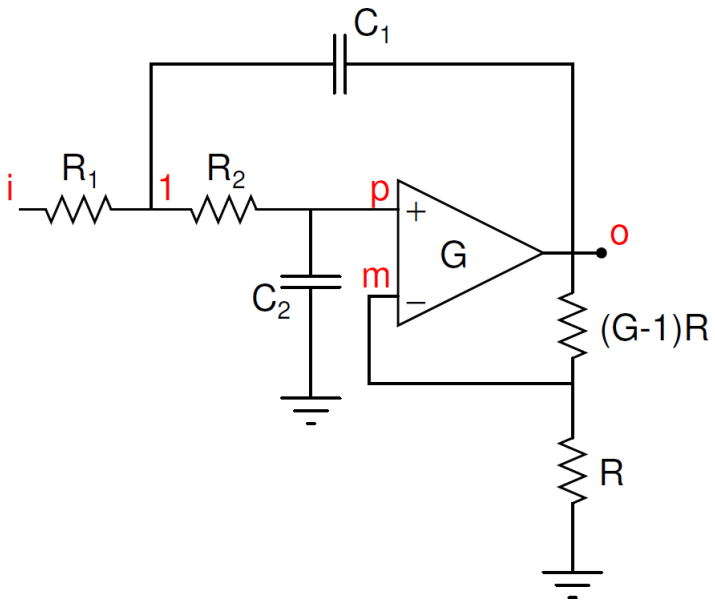

Figure 1: Lowpass filter circuit

The corresponding equations in order to get the output in s-domain for any input in s-domain are as follows:

$$V_m = \frac{V_o}{G}$$
$$V_p = V_1 \frac{1}{1 + sR_2C_2}$$
$$V_o = G(V_p - V_m)$$
$$\frac{V_i - V_1}{R1} + \frac{V_p - V_1}{R_2} + sC_1(V_o - V_1) = 0$$

This set of equations is represented in matrix form and is solved. This is performed for the most general value of all the circuit parameters using the function `lowpass(R1, R2, C1, C2, G, Vi)`. For our circuit, the value of all the parameters are as follows:

$$G = 1.586, R_1 = R_2 = 10kOhm, C_1 = C_2 = 10pF$$

3.2 Plotting the transfer function

We give the input as an impulse (which is 1 in s-domain) and we plot the Bode magnitude plot of the transfer function. This is done in this code:

```
A,b,V=lowpass(10000,10000,1e-9,1e-9,1.586,1)
print('G=%f' % G)
```

```
Vo=V[3]
print(Vo)
ww=p.logspace(0,8,801)
ss=1j*ww
hf=lambdify(s,Vo,'numpy')
v=hf(ss)
p.loglog(ww,abs(v),lw=2)
p.grid(True)
p.show()
```

The resultant Bode plot is shown in the figure below:

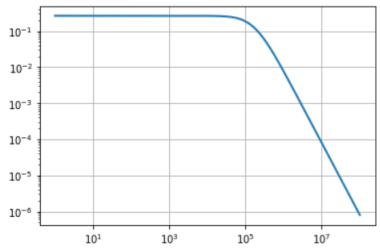

Figure 2: Lowpass filter Bode plot

4 Converting s-domain function to num,den form

Isolating the numerator and denominator functions from a *s*-domain transfer function is of great importance as we will be switching between `sympy` and `scipy.signal` ways of analysing. The way of inter-converting between the two forms involve simplifying the function such that both the numerator and denominator are proper polynomials of s, extracting the coefficients of s and converting them into floating point numbers. The entire process is illustrated in this code:

```
H = V[3]
H1 = expand(simplify(H))
n, d = fraction(H1)
n, d = Poly(n, s), Poly(d, s)
num, den = n.all_coeffs(), d.all_coeffs()
num, den = [float(f) for f in num], [float(f) for f in den]
```

The above algorithm is used multiple times in this assignment.

5 Step Response

We apply the above algorithm to convert the s-domain response of the output (given by V[3]) into num, den form. Using `sp.impulse()` function, the time

domain output is plotted from 0 to 10ms. This is shown in the figure below:

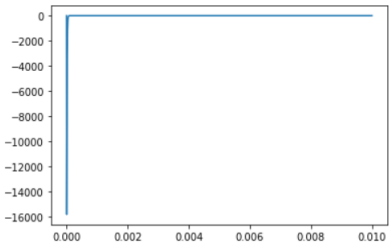

Figure 3: Step response to lowpass filter circuit

6 Output to sum of sinusoids

Here, we define the time-domain input function

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

Here, we can see that the input is sum of two sinusoids, one having a relatively low frequency and the other having a high frequency. Using `sp.lsim()` function, the time domain output when the above signal is passed through the lowpass filter is obtained. This is plotted as well. The plot is obtained as follows:

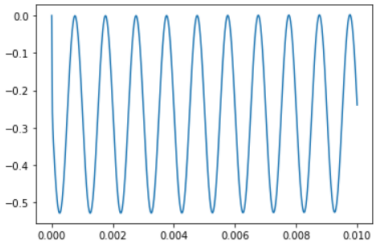

Figure 4: Output to sum of sinusoids

This concludes our analysis for lowpass filter.

7 Highpass filter

7.1 Defining the circuit

Next, we are going to analyse the high pass filter circuit which is shown below:

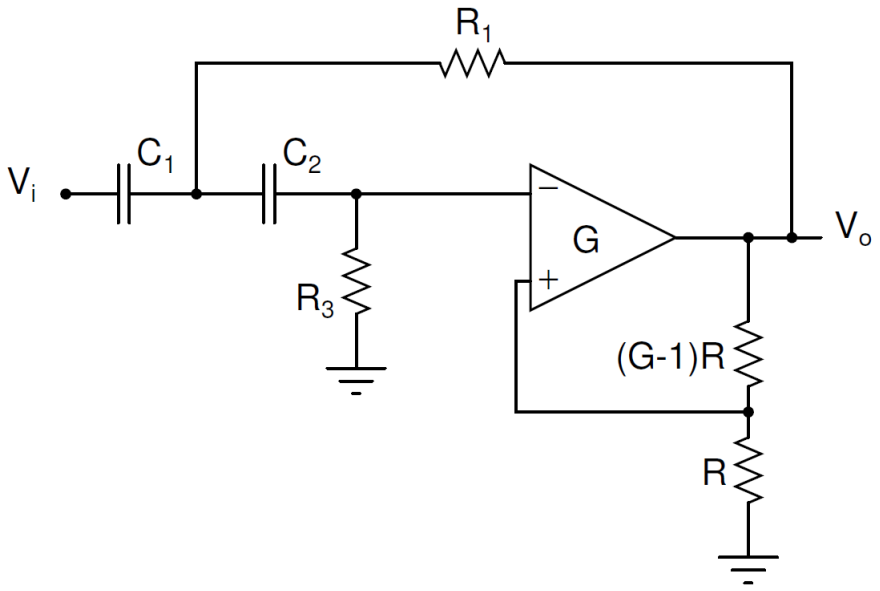

Figure 5: Highpass filter circuit

This circuit can be modelled in s-domain using the function `highpass(C1, C2, R1, R3, G, Vi)`.

7.2 Bode plot of transfer function

The values of all the components of the highpass filter are the same as the corresponding values of the lowpass filter. For obtaining the transfer function, $V_i(s) = 1$. We use the similar code as before in order to plot the Bode plot of the transfer function. The Bode plot obtained is as shown below:

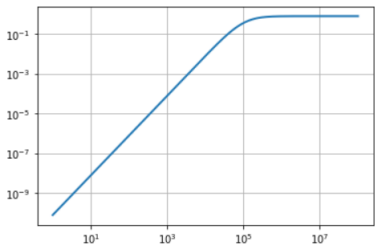

Figure 6: Highpass filter - Bode plot

8 Responses to decaying sinusoids of different frequencies

To the highpass filter circuit, we pass a couple of functions as input - both of them are decaying sinusoids, but one is of a higher frequency, whereas the other is of a lower frequency. We plot the time domain input and output functions and try to reason out the same. Since we are going to use the signals toolbox here, the transfer function is converted into num, den form.

8.1 Response to a decaying sinusoid of a higher frequency

We pass the following time-domain function as an input to our circuit:

$$V_i(t) = cos(2 * 10^5 t)exp(-0.5t)$$

Using `sp.lsim()`, the output in time domain can be obtained. This is obtained for t running from 0 to $3 * 10^{-4}$ sec. The time-domain input and output is plotted with appropriate legends. The plot obtained is as follows:

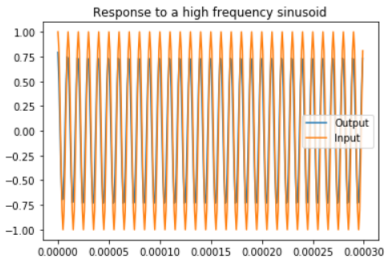

Figure 7: Output to a high frequency decaying sinusoid

8.2 Response to a decaying sinusoid of a lower frequency

We perform the same exercise as above, but now the time-domain input function is:

$$V_i(t) = sin(2\pi t)exp(-0.5t)$$

Here, we obtain the plot for t running from 0 to 10 secs. We plot the input and output time domain functions, and the graphs obtained are as below:

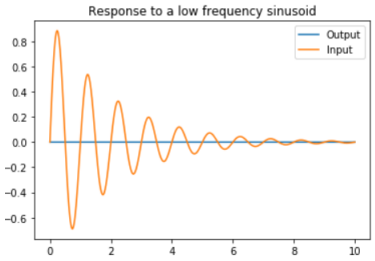

Figure 8: Output to a high frequency decaying sinusoid

Here, we can see that the output to the decaying sinusoid almost resembles the input function, except for perhaps some scaling factor for a high frequency sinusoid and it is almost zero as the frequency decreases. This shows that inputs of higher frequencies are completely passed whereas the input of lower frequencies are completely bolcked by the circuit.

9 Step response

For the unit step function $u(t)$, the s-domain function is $1/s$. We pass that as the input to the `highpass()` function. Since we want the time-domain output to the step function, the s-domain output is converted first to the num, den form to use the signals toolbox, and then to the time domain using the `sp.impulse()` function. The time domain output is plotted from t=0 to 10ms. The obtained plot is as follows:

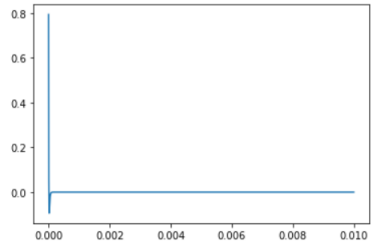

Figure 9: Step response

10 Conclusion

In this assignment, I have learnt about the `Sympy` library and the procedure to use differemt features of the library to solve the circuits in the Laplace domain without much fuss. The tools learnt in the last two assignments, `sympy` and `scipy.signal()` are really powerful tools for an electrical engineer to solve circuits much quickly and analyse the properties of the circuit by plotting the time-domain outputs for different time-domain inputs.