Indian Institute of Technology, Madras
Department of Electrical Engineering
Applied Programming Lab

Lab Report
Assignment 7

Nithin Uppalapati
EE18B035

05-06-2020

Contents

1. Abstract

Electrical engineers deal with various circuits very often. Sympy in python enables us to use symbols to describe components and write equations using these symbols.

Sympy proves to be very useful while solving for circuit parameters as we can deal directly with the symbols and enter the values later.

2. Introduction

In this assignment we are going to use a new module called sympy in python to write transfer functions of High Pass as well as Low Pass filters and use our knowledge from our previous experiment to solve for output given the input of these systems and also find out the step response and Transfer functions of the systems.

3. How to use Sympy

The first thing to be done is to import the required modules. In this assignment we are going to use a module called sympy.

• To define symbols we use the command symbol(). Once we use the symbol function on certain variables, the variables are symbols and no longer python variables.

Example Code:

```
s,R2,C2=symbols('s R_2 C_2')
h=-1/(1+s*R2*C2)
```

- To create matrices we use **Matrix([[1,2],[3,4]])**

So, the above command creates a matrix with two rows and two columns

We can also do inverse operations with is this library.

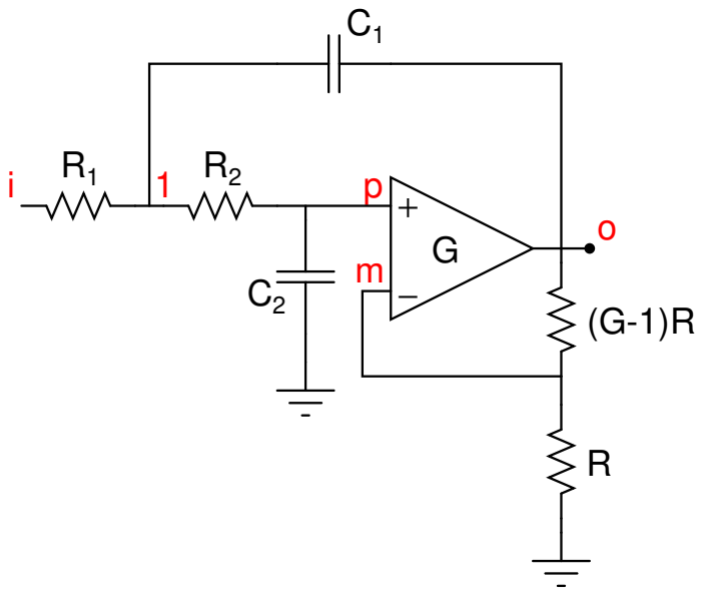
- We also use a function called lambdify(s,h,'numpy').What lambdify does is it takes the sympy function "h" and converts it into a python function which is then applied to the array 'ss'.

```
ww=p.logspace(0,8,801)
ss=1j*ww
hf=lambdify(s,Vo,'numpy')
v=hf(ss)
```

3.1 The Assignment

3.1.1 Transfer Function

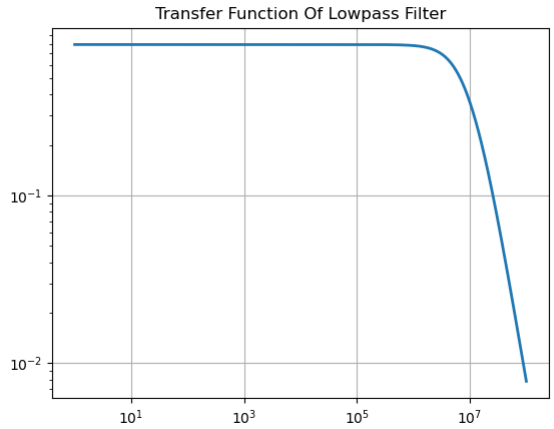
We can solve directly for the exact result from Python. Let us define$s = j\omega$, and rewrite the equations in a matrix equation.The below equation is for a Low Pass Filter.



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

Now we just create a function in python using this matrix and use it to get outputs when an input is fed to the system.

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



3.1.2 Problem 1

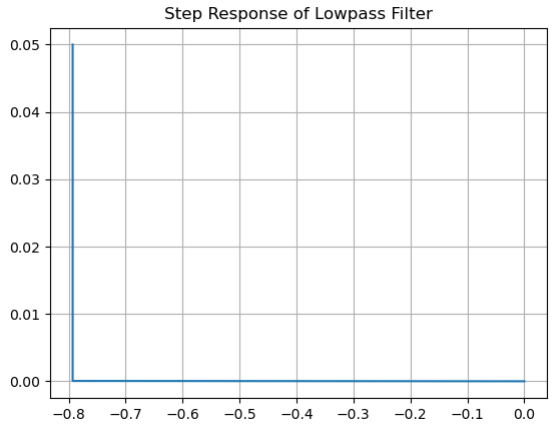
In the first problem we have to find the step response of a Low Pass Filter. We can use the function **sp.step.**

We first import all the required modules.

```
from __future__ import division from sympy import *
import pylab as p
import scipy.signal as sp
```

The following code gives us the step response.

```
p.figure(1)
p.title("Step Response of Lowpass Filter")
Y=sympy_to_lti(Vo)
t1=p.linspace(0,0.05,1000)
t,x=sp.step(Y,None,t1)
p.plot(x,t)
```



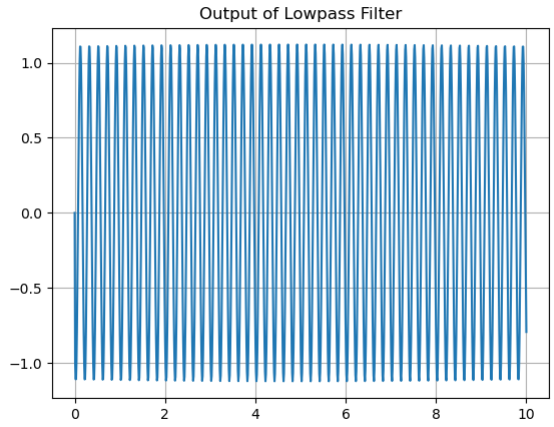
3.1.3 Problem 2

In the next problem we have to find output of the Low Pass Filter when an input containing two different frequencies are passed.

$$V_i(t) = (sin(2000\pi t) + cos(2 * 106\pi t))u_0[t]$$

```
p.figure(2)
w_1=2e3*p.pi
w_2=2e6*p.pi
Vin=(w_1/(s**2+w_1**2))+(s/(s**2+w_2**2))

A0,b0,V0=lowpass(10000,10000,1e-11,1e-11,1.586,Vin)
Vout=V0[3]
t2=p.linspace(0,0.1,1000)
# t2,x=sp.impulse(sympy_to_lti(Vout),None,t2)
u1=p.sin(w_1*t)+p.cos(w_2*t)
t,y,svec=sp.lsim2(Y,u1,t2)
p.title('Output of Lowpass Filter') p.plot(t,x)
```

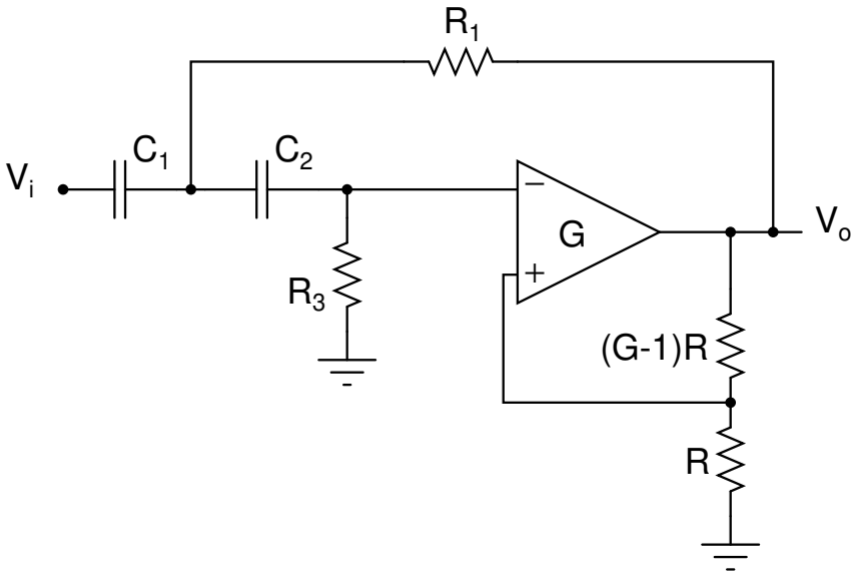


3.1.4 Problem 3

In this problem we are supposed to find the transfer function of a similar High Pass Filter. The process is similar compared to what we did for the Low Pass Filter before problem 1.

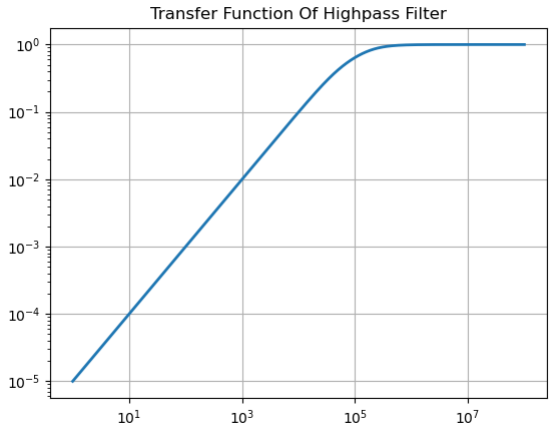
Defining High pass filter Function:

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



Using the lamdify function we can define the transfer function for an array of frequencies $ss = j * ww$ and then find the transfer function.

```
a1,b1,V1=highpass(10000,10000,1e-9,1e-9,1.586,1)
hf1=lambdify(s,V1[3],'numpy')
ww=p.logspace(0,8,801)
ss=j*ww
v1=hf1(ss)
p.figure(3)
p.title('Transfer Function Of Highpass Filter')
p.loglog(ww,abs(v1),lw=2)
```

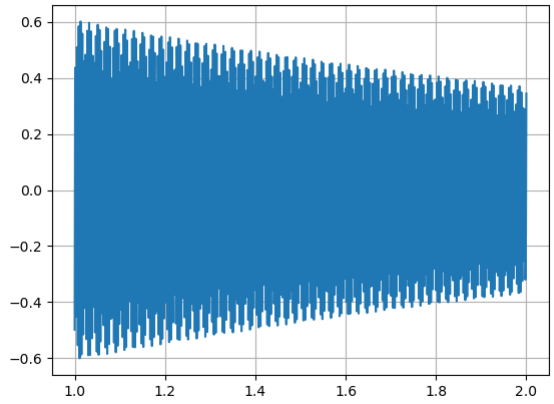


3.1.5 Problem 4

In this problem we send a damped sinusoid as an input to the High Pass Filter and plot its output. We use signal.lsim to simulate the output.

$$V_i = sin(2\pi wt)exp(at)$$

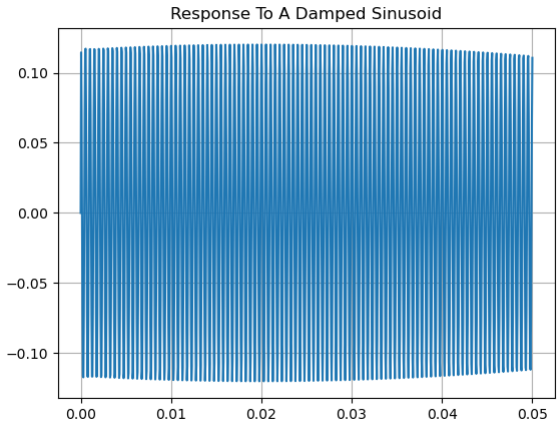
And w_2=$2\pi * 10^6$



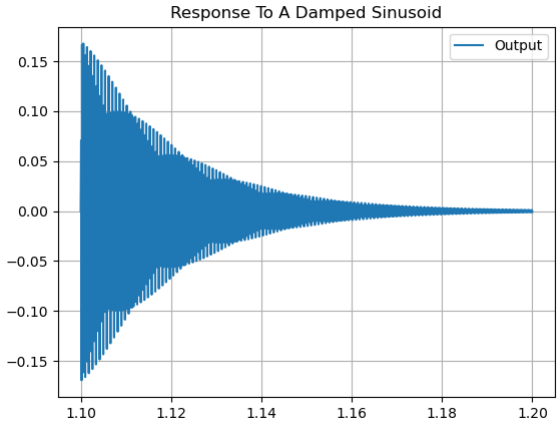
We have the transfer function in V1[3]. We use the function sympy-to-lti and then enter in the lsim function.

```
Vda=V1[3]
Vd=sympy_to_lti(Vda)
a=0.5
u=p.sin(w_2*p.pi*t)*p.exp(-a*t)
t3=p.linspace(1,2,1000)
t,y,svec=sp.lsim(Vd,u,t3)
p.figure(4)
p.title("Response To A Damped Sinusoid")
p.plot(t,y)
```

The figure is a zoomed in version of the output signal. The damping is not much evident.



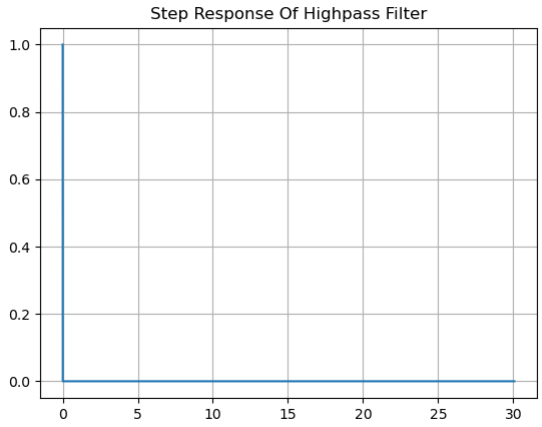
This is the output plot for a long period.



3.1.6 Problem 5

Finally we have to find step response of the High Pass Filter.

```
a2,b2,V2=highpass(10000,10000,1e-9,1e-9,1.586,1/s)
Vst=V2[3]
p.figure(5)
p.title("Step Response Of Highpass Filter")
t4=p.linspace(0,30.07,100000)
Y1=sympy_to_lti(Vst)
t,y=sp.impulse(Y1,None,t4) p.plot(t,y)
p.show()
```



3.2 Observations

From this assignment we can observe that sympy is a very useful module in python as we can now work with symbols instead of variables. We can think of many applications for this as we can continuously change the values given to symbols and that maybe useful for certain applications.