EE2703 : Applied Programming Lab
Experiment 7

Ch Venkat Sai
EE18B042

April 08, 2020

Overview

The experiment is about:
Learning the two powerful tools in python:
1. Symbolic Algebra
2. Analysis of circuits using laplace transform
We majorly use sympy in this experiment.

Theory

If the laplace transform of the input f is F(s) and the laplace transform of the impulse response h(t) is H(s) then the laplace transform of the output is H(s)F(s). The transform of any function can be found out using the formula:

$$H(s) = \int_{-\infty}^{\infty} h(t)e^{-st}dt \tag{1}$$

Though some of the functions have classic laplace transforms and corresponding inverses.
After importing sympy we use a datatype called symbols. Instead of using the, as variables we use them as symbols like those in polynomials and as those in different expressions. Further we have some allowance of using these expressions and polynomials in mathematics like those in matrices and more. Then we can plot these expressions in plots too.

Q1

We now use symbols to use a variable as a symbol as in the code. We correspondingly define the filter function i.e the matrices in terms of the symbol and find the corresponding inverse and thus finding the transfer function. We now use lamdify so that we can plot these in graphs give some value to the symbols in the expression.

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

s = symbols('s')
def lowpass(R1,R2,C1,C2,G,Vi):
    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)


A,b,V = lowpass(10000,10000,1e-9,1e-9,1.586,1)
Vout = V[3]
lp = lambdify(s,Vout,'numpy')
ww = p.logspace(0,8,801)
ss = 1j*ww
v = lp(ss)
h2 = simplify(Vout)
numer,denom = h2.as_numer_denom()
num = poly(numer, s)
den = poly(denom, s)
Hlp = sp.lti([float(i) for i in num.all_coeffs()],
[float(i) for i in den.all_coeffs()])
p.loglog(ww,abs(v),lw=2)
p.title("Low pass filter")
p.grid(True)
p.show()


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


A,B,V = highpass(10000,10000,1e-9,1e-9,1.586,1)
Vout = V[3]
hf = lambdify(s,Vout,'numpy')
ww = p.logspace(0,8,801)
ss = 1j*ww
v = hf(ss)
h3 = simplify(Vout)
numer,denom = h3.as_numer_denom()
num = poly(numer, s)
den = poly(denom, s)
```

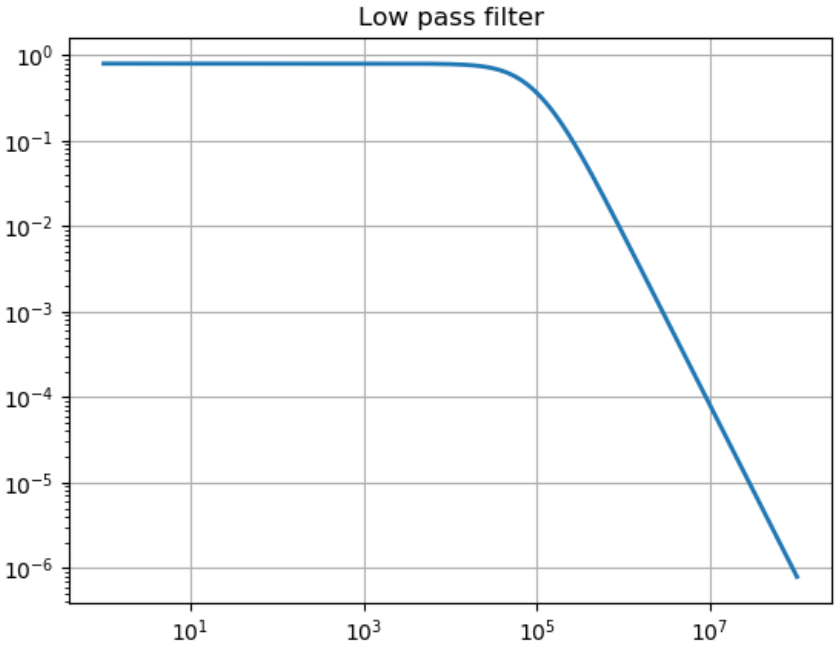

Figure 1: Low pass

```
Hhp = sp.lti([float(i) for i in num.all_coeffs()],
[float(i) for i in den.all_coeffs()])
p.loglog(ww,abs(v),lw=2)
p.title("High pass filter")
p.grid(True)
p.show()
```

As seen we are giving the s the value of jw.
The plots are as shown in figure.
The transfer function obtained is for the following circuit:

Q2

Having obtained the transfer functions we now send in the following input:

$$x = cos(2x10^6\pi t) + sin(2000\pi t) \tag{2}$$

```
v = p.cos(2*(10**6)*p.pi*t) + p.sin(2000*p.pi*t)
t = p.linspace(0,1.5e-3,50000)
```

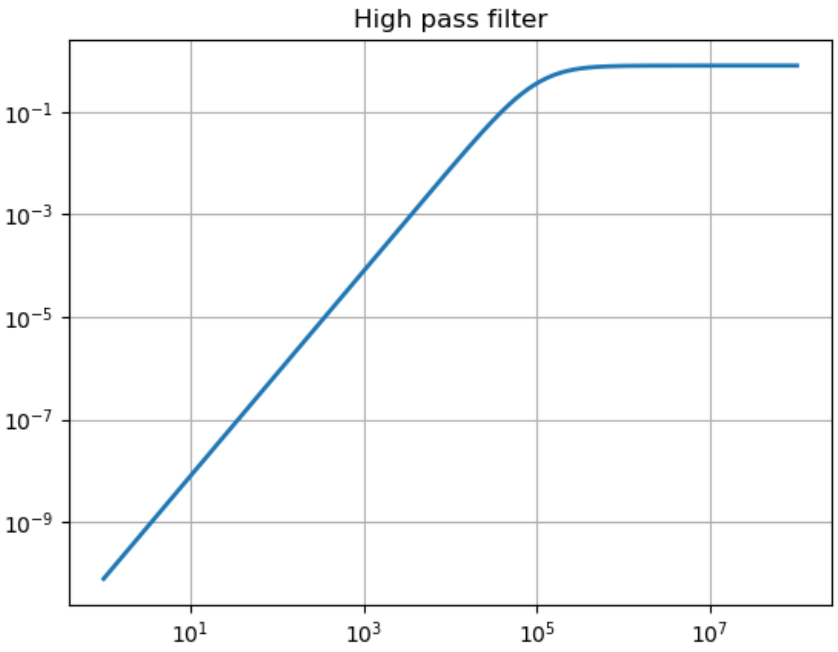

Figure 2: High pass

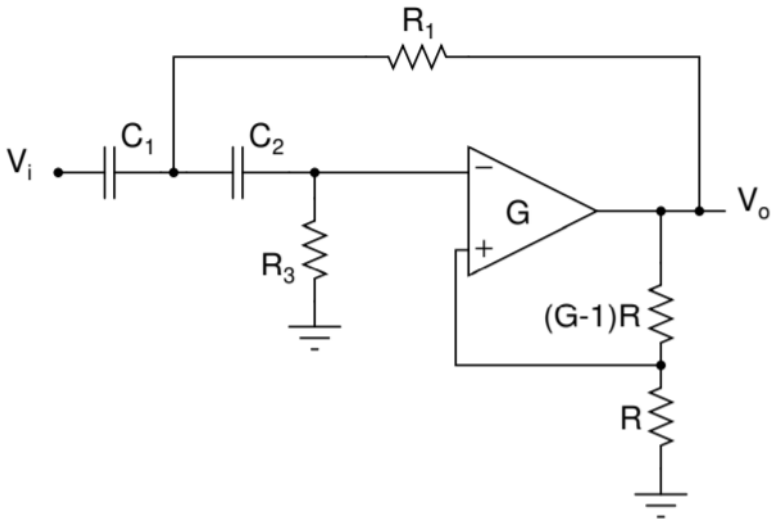

Figure 3: Circuit for high pass

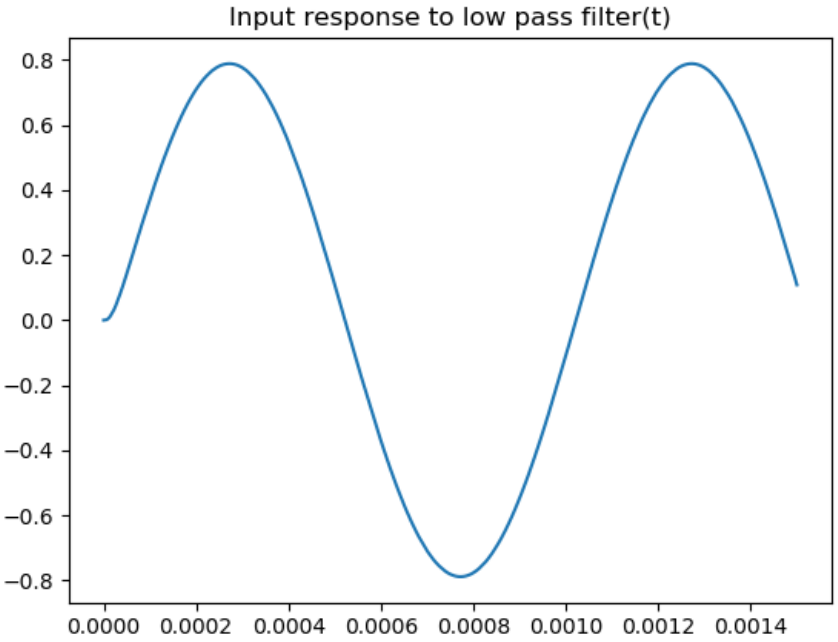

Figure 4: Response with low pass

```
y = sp.lsim(Hlp, v, t)[1]
p.plot(t,y)
p.title("Input response to low pass filter(t)")
p.show()
```

As this is a low pass filter the second term remains and thus the figure is as shown in fig.4

Q3

Having obtained the transfer functions we now send in the following input:

$$x = cos(2x10^6\pi t) + sin(2000\pi t) \tag{3}$$

```
y1 = sp.lsim(Hhp, v, t)[1]
p.plot(t,y1)
p.title("Input response to high pass filter(t)")
p.show()
```

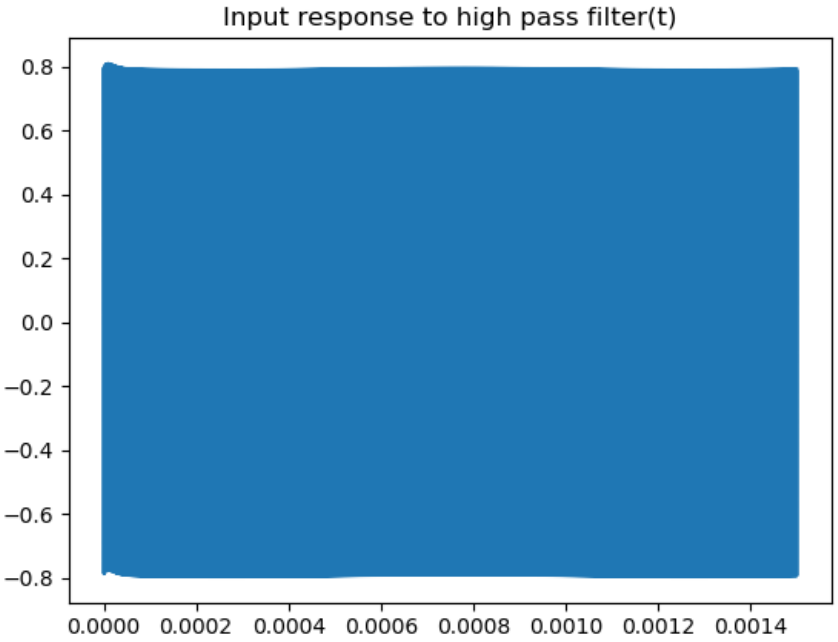

Figure 5: Response with high pass

As this is a high pass filter the first term remains and thus the figure is as shown in fig.5

Q4

Now we try to find the step response. This means the input is simply an array of 1s with the size that of the time t.

```
v = p.array([1 for i in t])
y = sp.lsim(Hlp, v, t)[1]
p.title("Step response to low pass filter(t)")
p.plot(t,y)
p.show()
```

The graph turns out be as shown in fig:6.

Q5

Similarly here also.

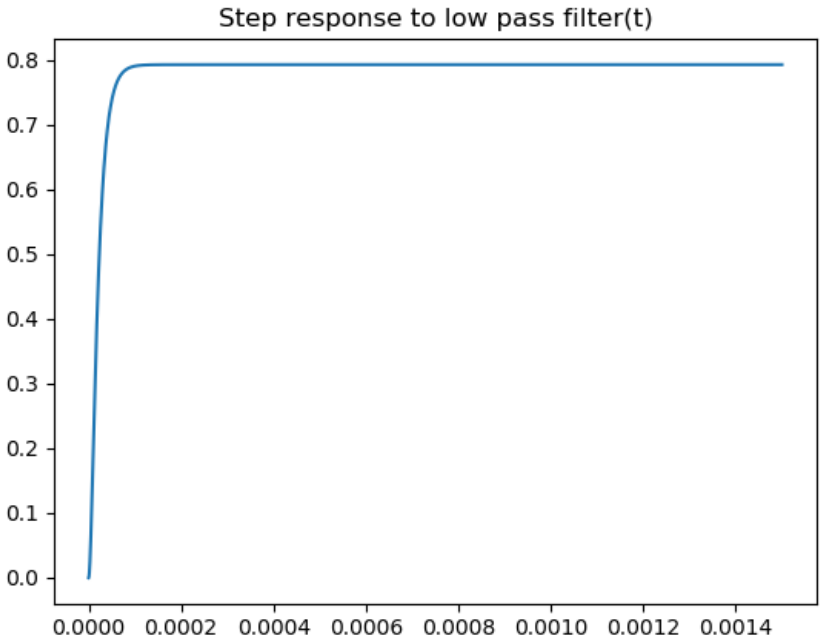

Figure 6: Step response

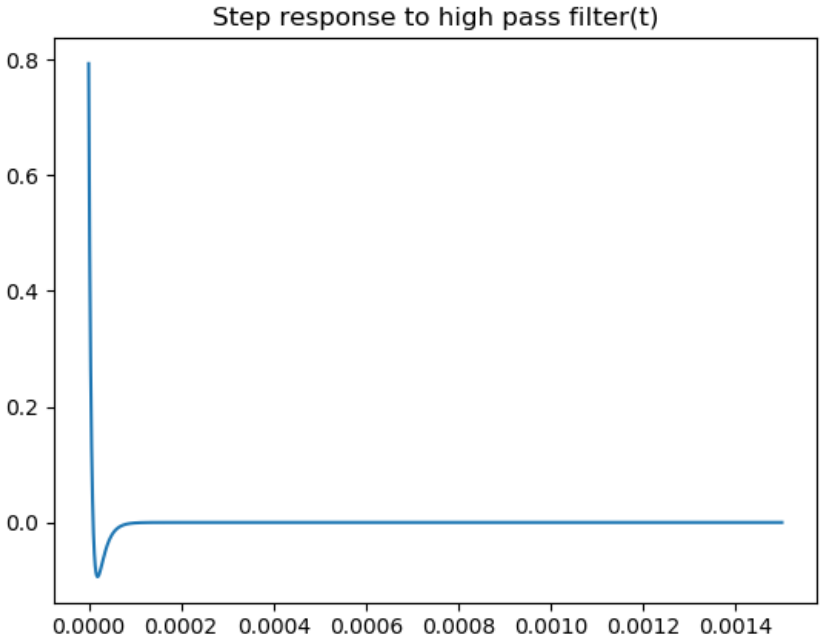

Figure 7: Step response

```
y1 = sp.lsim(Hhp, v, t)[1]
p.title("Step response to high pass filter(t)")
p.plot(t,y1)
p.show()
```

The figure turns out to be as shown in fig:7.

Q6

Now we take an damped sinusoidal input as one shown:

$$v_(t) = sin(500t)\exp -0.05t \tag{4}$$

And this is feeded to the high pass filter circuit as shown.

```
v = p.sin(500*t)*(p.e**(-0.05*t))
y = sp.lsim(Hhp, v, t)[1]
p.title("Damped sin response to high pass filter(t)")
p.plot(t,y)
```

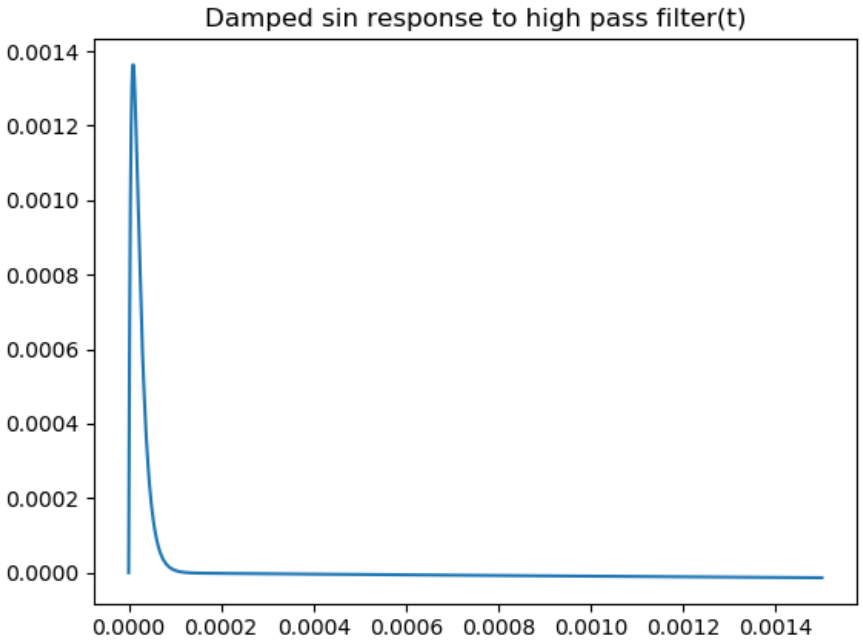

Figure 8: Response with a damped sinusoid

This gives us a plot like this: Here the step response goes from 0 to 1 at t=0 which is very sudden and hence the frequency is very high and therefore is allowed. Later it becomes constant which implies a zero frequency and not allowed therefore the response is reaches 0.

Conclusion

As this experiment comes to an end we were able to able to solve any circuit given to us by converting it into laplace domain and we were able to plot these in loglog scale as well as in time domain with the help of signal tool box.