Circuit Analysis Using Sympy

EE16B014
G.HARSHA KANAKA ESWAR

April 06,2018

Abstract

There are many types of filters present ahvailable. such asFIR Filters,IIR Filters,High-pass,Low-pass,Band-pass,Stop-band,Notch,Comb Filter,All-pass,etc....IN the above filters we are going to discuss two filters.
They are:
1) *Low pass filter*
2) *high pass filter*
Here we will use symbolic python for writing the transfer function directly as a matrix with Frequency *'s'* as a variable so that we can get the magnitude plot of transfer function and ,its response to different inputs of having different frequencies.

1 Lowpass Filter:

In this section We have the low pass filter equation in the form of matrix,which we will solve using symbolic python with matrix inversion.

The equation is given as:

$$\begin{pmatrix} 0 & 0 & 1 & -\frac{1}{G} \\ -\frac{1}{1+sR_2C_2} & 1 & 0 & 0 \\ 0 & -A_0 & A_0 & 1 \\ -\frac{1}{R_1}-\frac{1}{R_2}-sC1 & \frac{1}{R_2} & 0 & sC1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_p \\ V_m \\ V_0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{V_i(s)}{R_1} \end{pmatrix}$$

We now use the function 'Matrix' in sympy to create two matrices A,b -cofficient matrix and source matrix Inside a function named as lowpass where there a variable 's' which is declared using symbols.

```
def lowpass(R1,R2,C1,C2,G,Vi,Ao):

        A= Matrix([[0,0,1,-1/G],[-1/(1+s*R2*C2),1,0,0],[0,-Ao,Ao,1], \
           [-(1/R1)-(1/R2)-(s*C1),1/R2,0,s*C1]])
        b= Matrix([0,0,0,-(Vi/(R1))])
        print (A)
        print (b)
```

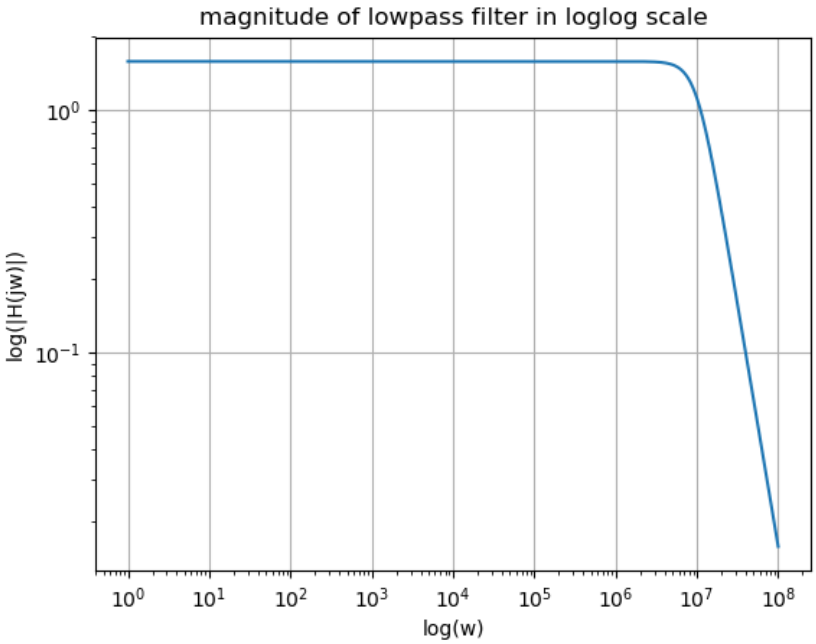
```
V=A.inv()*b
print (V)
return (A,b,V)
```

The above function takes the input values for constants and gives out $V_o(s)$ as the output along with A,b matrices.So we got the output equation interms of *'s'(Frequency).*
We can see the plot of magnitude of H(s) , by defining 'hf' using lambdif and array of frequencies(w) in log scale using logspace and 'ss' as jw then plotting hf(ss) vs w.[1]

```
A,b,V=lowpass(10**4,10**4,10**-11,10**-11,1.586,1,1000)
Vo=V[3]
w=p.logspace(0,8,801)
ss=1j*w
hf=lambdify(s,Vo,'numpy')
v=hf(ss)
p.loglog(w,abs(v))
p.xlabel('w')
p.ylabel('|H(jw)|')
p.title('magnitude of lowpass filter in loglog scale')
p.grid(True)
p.show()
```

And the plot looks like:

[1] H(jw),H(s) mean the same.(s=jw)



Since we got $V_o(s)$ we will now define a function called coff to get the cofficients of 's' in numerator and denominator to define transfer function H(s),using the cofficients.

```
def coff(Vo):
Vo=simplify(Vo)
N,D=fraction(Vo)
As=simplify(N)
bs=simplify(D)
As1=Poly(As,s)
bs1=Poly(bs,s)
print (As1,bs1)
Num=As1.all_coeffs()
Den=bs1.all_coeffs()
Num1=list(map(float,Num))
Den1=list(map(float,Den))
return Num1,Den1
```

After getting cofficients using above function H(s) is given as:

```
H=sp.lti(Num1,Den1)
```

1.1 Step Response:

The response of the system to input step function is known as step response.So we give input step which is nothing but one when the time is greater than zero.

So we define 'u(t)' with function ones.

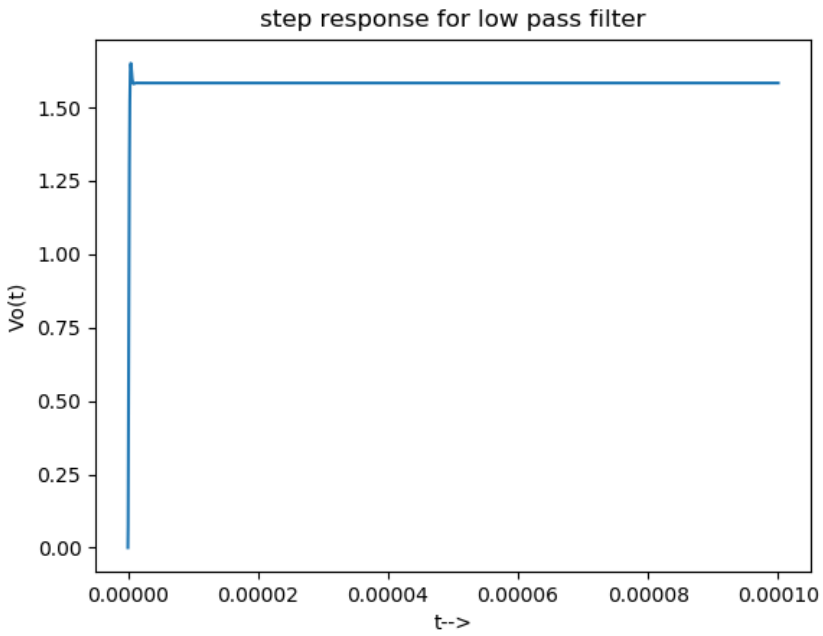
```
def u(t):
        return p.ones(len(t))
```

Since we got H(s),u(t) lets get the step response By using sp.lsim from scipy

```
H=sp.lti(Num1,Den1)
t=p.linspace(0,0.0001,3000)

t,y,svec=sp.lsim(H,u(t),t)
p.plot(t,y)
p.xlabel('t-->')
p.ylabel('Vo(t)')
p.title('step response for low pass filter')
p.show()
```

Now we plot the graph of step response :



u(t) is nothing but a signal of frequency '0'.As we Can See the response is just an amplification of the u(t),the Amplification is due to gain of 'u(t)' which is nearly 1.58, which can be seen from the magnitude plot of H(s).[2].
At t=0 instant u(t) can be taken as very high frequency because of its sudden change in value from 0 to at t=0, which makes the output zero at zero for low-pass fiter.We can also observe a small ripple at zero,it is just because of gibbs phenomena,its magnitude depends on quality factor of H(s).

[2]Gain of a response at zero frequency is called as DC gain

1.2 Output for an input sine wave:

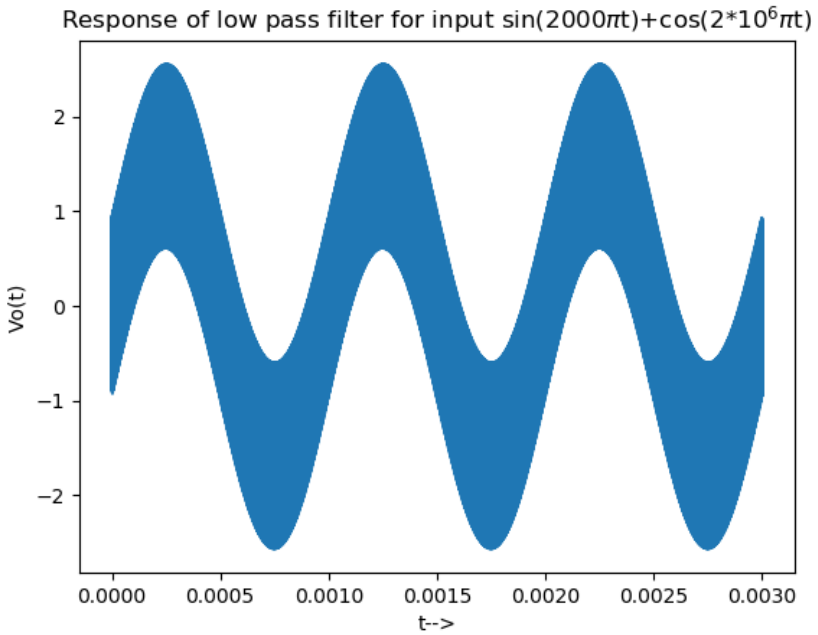
Lets now take input combination of two sinusoids of frequencies: 2×10^3,2×10^6 as show below:

$$v_i(t) = (sin(2000\pi t) + cos(2\times10^6\pi t))u_0(t)Volts$$

We can find the output in a simlar method as we have done step response,with a small change. just replace the function 'u(t)' in the above case with 'x(t)', which is defined for the above input.

```
def x(t):
        return p.sin(2000*(p.pi)*t)+p.cos(2*1e10*(p.pi)*t)
t1=p.linspace(0,0.002,10000)
t1,y2,svec=sp.lsim(H,x(t1),t1)
p.xlabel('t-->')
p.ylabel('Vo(t)')
p.title('Response of low pass filter for input sin(2000*(pi)*t)+cos(2*1e6*(pi)*t)')
p.plot(t1,y2)
p.show()
```

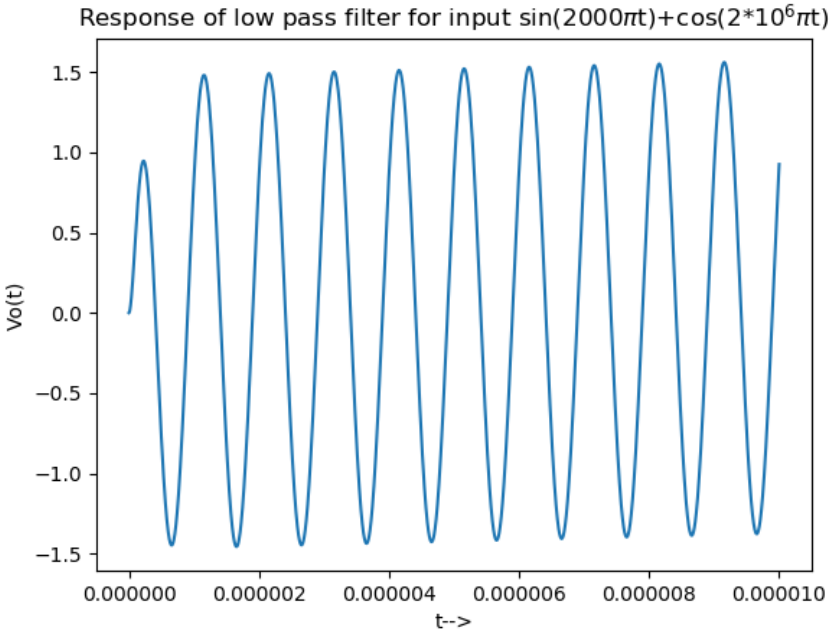
Now lets see the plot of output for various t1's:



As we can see its like a sinusoid with some thickness, and the frequency of the sine wave is '2000'.But observe the amplitude of the wave its around '3' but

gain of sinusoid of frequency is just 1.5.OK now lets take a close look of whats happening
Now lets get more inference by decreasing the value of 't1'.

```
t1=p.linspace(0,0.00001,10000)
```



Now we are able to see an anthore sinusiod, Which is the cause of thickness in the first output we saw,Which means both sinusiod frequencies are passed at the output successfully.Thus it makes sense of amplitude greater than one sinusoid gain,because of two sinusoid outputs.

1.3 Conclusions:

- Thus we got see that how low pass is allowing the DC signals like u(t) which was seen at calculation of step response.
- And the sinusiods are passed freely to the output because of the gain observed from H(s) graph is greater than '1',thus telling that those two frequencies are low for H(s),and allowed to pass.
- And the peek seen in the step response is due to gibbs phenomena,depends mainly on quality factor of H(s)

2 Highpass filter:

In the above case we have clearly discussed about using Sympy ,Lowpass filter and its Characteristics,Now we will discusss about HighPass filter and its response to different inputs.

We have the circuit equation in the form of matrix equation as shown below which is solved by similar code used for Lowpass filter:

$$\begin{pmatrix} 0 & 0 & 1 & -\frac{1}{G} \\ -\frac{sR_3C_2}{1+sR_3C_2} & 1 & 0 & 0 \\ 0 & -A_0 & A_0 & 1 \\ -\frac{1}{R_1} - sC_2 - sC_1 & sC_2 & 0 & sC_1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_p \\ V_m \\ V_0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ V_i(s) * sC_1 \end{pmatrix}$$

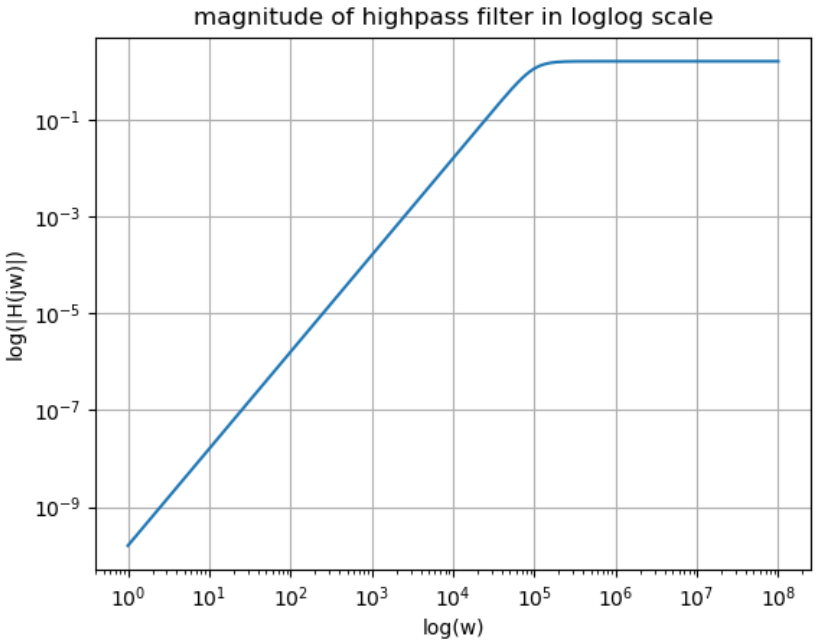
The Equation is simalir to the Lowpass in some fashion that R_1replaced by $\frac{1}{sC_1}$ and viceversa,R_2replaced by $\frac{1}{sC_2}$ and $\frac{1}{sC_2}$ byR_3.Thats just because of the circuit where the similar replacement we can see that some resistors replaced by capcitors and capacitors by resistors.
Ok now lets see the code for solving this:

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

There is not more new things to explain here as we already saw getting coffecients from $V_o$and getting its transfer function by scipy.
Here is the code:

```
A,b,V=highpass(10**4,10**4,10**-9,10**-9,1.586,1,1000)
Vo=V[3]
w=p.logspace(0,8,801)
ss=1j*w
hf=lambdify(s,Vo,'numpy')
v=hf(ss)
p.loglog(w,abs(v))
p.xlabel('log(w)')
p.ylabel('log(|H(jw)|)')
p.title('magnitude of highpass filter in loglog scale')
p.grid(True)
p.show()
```

So lets take a direct look at the graph of *magnitude of highpass vs frequency* and get inferences:



So from seeing the graph we can directly conclude that this filter doesn't all low frequencies,and passes all frequencies above some limit with a constant gain.

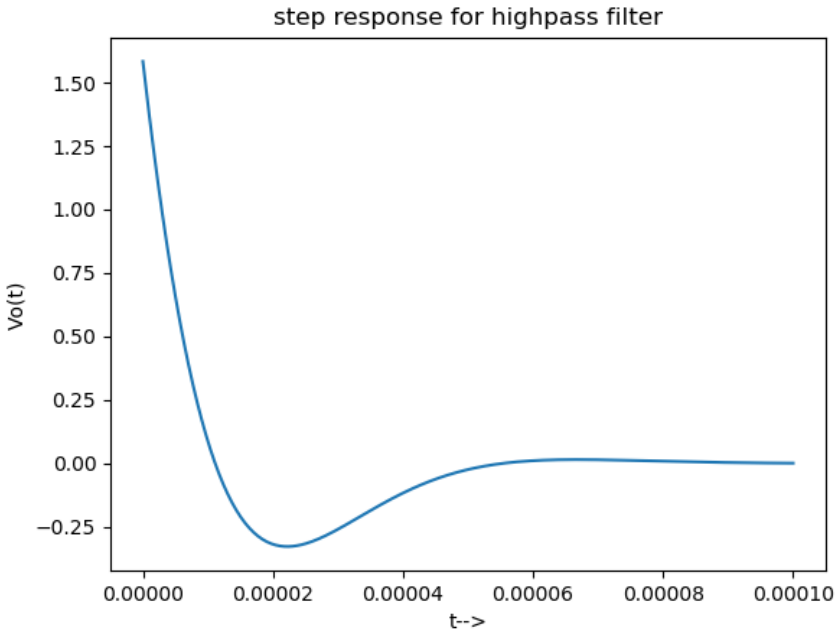
2.1 Step Response:

As we have discussed earlier the response of system to step is called as step response.
Lets see the code first and then the graph:

```
def u(t):
        return p.ones(len(t))

H=sp.lti(Num1,Den1)
t=p.linspace(0,0.0001,300)
t,y,svec=sp.lsim(H,u(t),t)
p.plot(t,y)
p.xlabel('t-->')
p.ylabel('Vo(t)')
p.title('step response for highpass filter')
p.show()
```

Here is the Output graph:



Lets take a close look at graph it seems that at zero the output has some gain near to 1.5 and the amplitude became nearly zero in steady state.This is because of the reasons that we have already known, They are at zero u(t) is acting like high frequecy wave. So it will get the gain of 1.5 but as the time becomes greater than zero its just a dc signal with frequency zero,For which the high pass has a very least which doesn't allow the signal to pass at all.

2.2 Output for a Damped sinusoid:

Lets take a damped sinsusoid example as shown below:

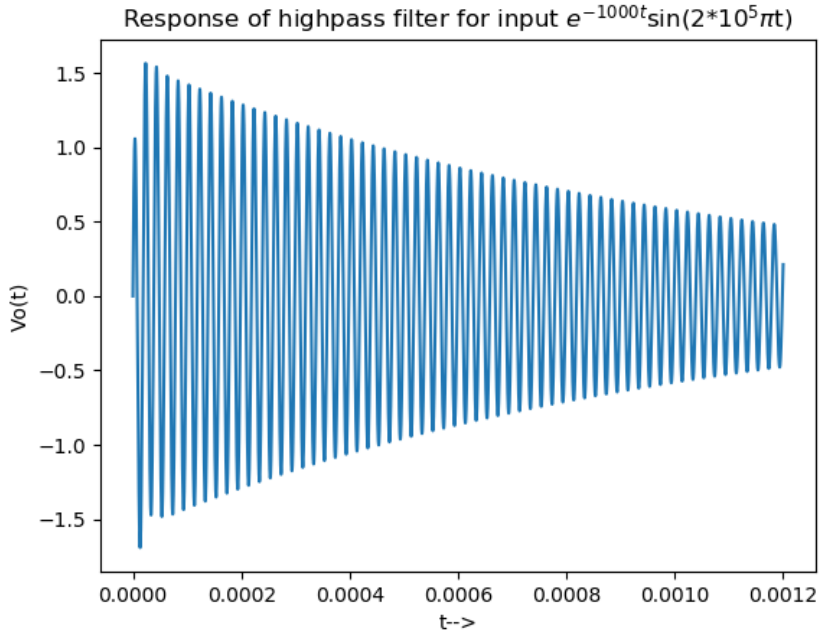
$$V_i(s) = (e^{-1000t}) * sin(10^5 \Pi t)$$

Here the signal is taken in such a way that it is passed by the highpass filter ,So that we can see the graph passed as a whole and decrease of amplitude is observed as time increases not because of H(s)(i.e, System) But because of the damping e^{-1000t}.
Here is the Code:

```
def x(t):
        return (e**(-1000*t))*sin(100000*pi*t)
t1=p.linspace(0,0.0012,5000)
t1,y2,svec=sp.lsim(H,x(t1),t1)
p.xlabel('t-->')
```

```
p.ylabel('Vo(t)')
p.title('Response of highpass filter for input (e**(-10*t))*sin(200000*pi*t)')
p.plot(t1,y2)
p.show()
```

It is done in the simalar fashion to lowpass,i.e just by defining a function x(t).
Lets take a look at the graph:



It just came like the input just amuktiplication factor of 1.5 is applied ,which can be seen as the gain from the system.So the damping is just because of the exponential with negative power infront of sinusoid.

2.3 Conclusions:

- The step Response of the system is having some value at zero time, which is eventually going to zero because of its zero frequency and highpass nature of the filter.
- The damping sinusoid has no effect due to filter except some gain ,but its main is due to its own exponent power multiplying it.