Assignment No 7

Subhankar Chakraborty

March 19, 2019

1 Analysing circuits using Laplace transforms

The system response of an LTI system, $H(s)$ links its input $X(s)$ and its output $Y(s)$ by the following relation.

$$Y(s) = X(s) * H(s)$$

or

$$H(s) = Y(s)/X(s)$$

$H(s)$ is a property of the system and does not depend on the input. Knowing the value of $H(s)$ can help us to analyse the system for different inputs. Since a lot of real life systems can be approximated as LTI systems, this also helps us to predict the approximate behaviour of a lot of real life systems.

2 The Low Pass Filter

The circuit given is shown in Figure-1. Solving equations in all the nodes keeping in mind that current does not enter inside an op amp,we get

$$\frac{V_o}{V_i} \approx \frac{1}{j\omega R_1 C_1}$$

We can see that the circuit is roughly a low pass filter.

2.1 Solving the equations in Python

The circuit equations are

$$V_m = \frac{V_o}{G} \tag{1}$$

$$V_p = \frac{V_1}{1 + j\omega R_2 C_2} \tag{2}$$

$$V_o = G(V_p - V_m) \tag{3}$$

$$\frac{V_i - V_1}{R_1} + \frac{V_p - V_1}{R_2} + j\omega C_1(V_o - V_1) = 0 \tag{4}$$

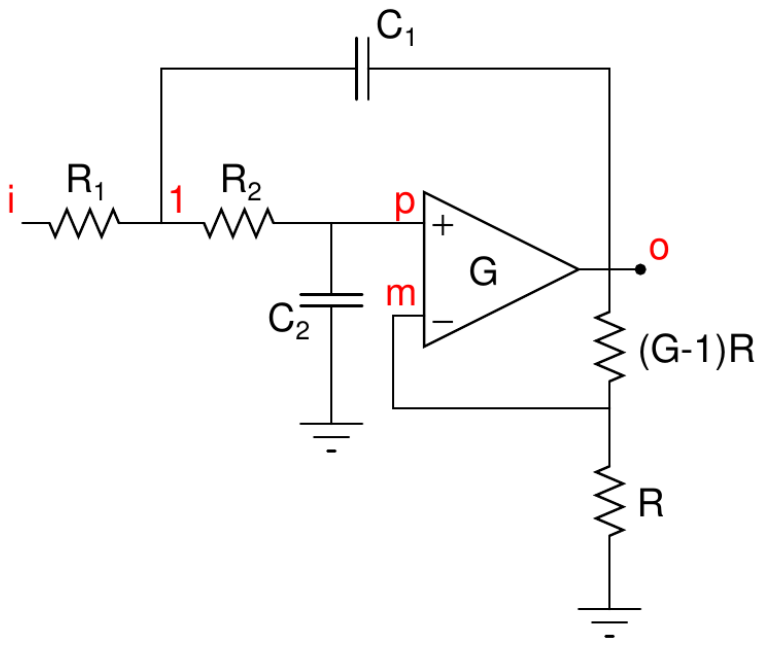

Figure 1: Circuit diagram for the low pass filter

We can solve it using Matrix inversion.We define a function called **lowpass** which calculates the output voltage for different values of $R_1, R_2, C_1, C_2 and G$.

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

We call it with some given values of $R_1, R_2, C_1, C_2 and G$.

```
A,b,V=lowpass(10000,10000,1e-11,1e-11,1.586,1)
Vo=V[3]
```

Using the lambdify function of Sympy, we convert this from a Sympy function to a Python function.

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

Since the input to the lowpass filter, $V_i(s)$ is taken to be 1, this will essentially give the transfer function of the system (as the Laplace Transform of the *dirac delta* is 1). Plotting the magnitude of the transfer function with respect to ω, we get Figure-2. We can clearly see that the gain is roughly 1 for ω

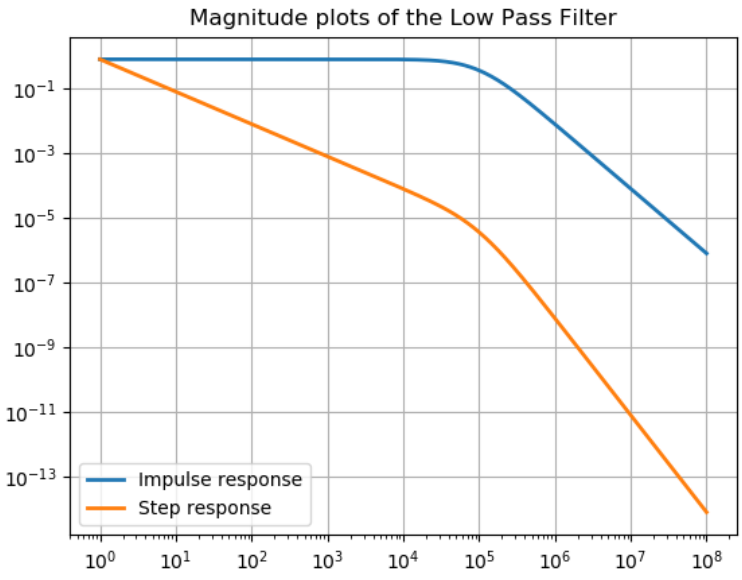

Figure 2: Magnitude plots of the Low pass filter

below 10^5 and takes a sharp dip after that. This is as expected of a low pass filter. The bandwidth of this filter is $\omega = 10^5$ rads^{-1}.

2.2 The filter in action

We pass input to the filter given by

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

$v_i(t)$ can be seen as a superposition of two sinusoids of frequencies $1 and 1000kHz$ respectively.We expect the low frequency component of the filter to stay as it is and the high frequency component to be filtered out.Figures 3 upto 6 will make all of this clear.

The following code does the above mentioned.

```
H_lowpass_impulse = sp.lti(numerators_impulse_response_lowpass,
denominators_impulse_response_lowpass)
low_freq_component = sin(2000*pi*time)
high_freq_component = cos(2*pi*1e6*time)
```

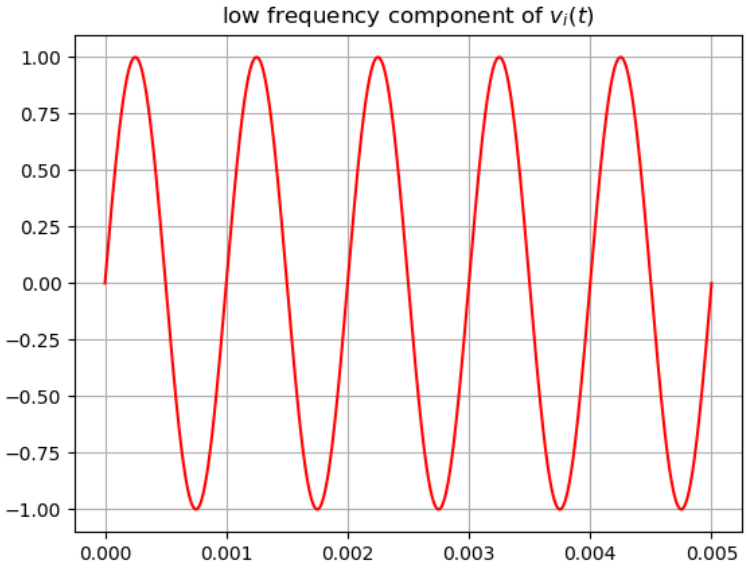

Figure 3: $v_i(t)$ and it's low frequency component

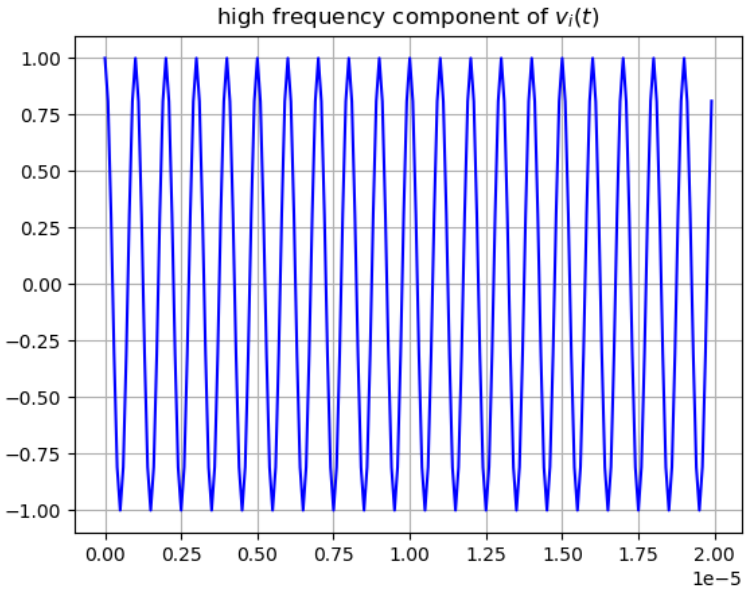

Figure 4: $v_i(t)$ and it's high frequency component

```
v_i =low_freq_component + high_freq_component
time,y,svec=sp.lsim(H_lowpass_impulse,v_i,time)
```

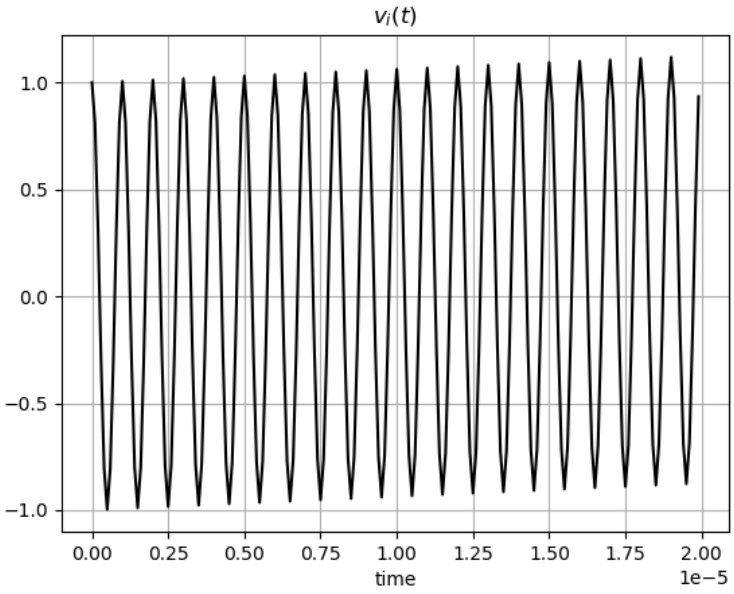

Figure 5: $v_i(t)$

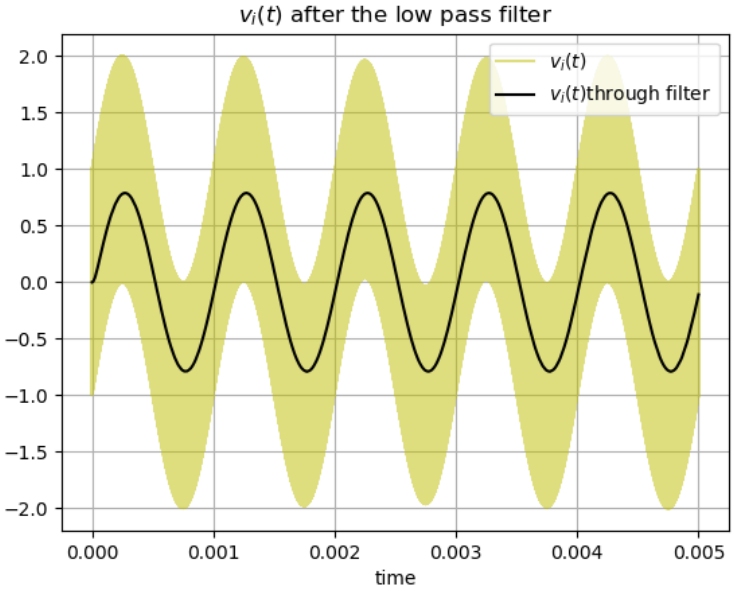

Figure 6: $v_i(t)$ after the low pass filter.

We can see that the high frequency component has been almost completely filtered out. One thing to note that the x limits in the high frequency graphs are lower so that the curve and its changes are visible to the naked eye.

3 The High Pass Filter

A few simple changes to the circuit of figure 1 changes it from a high pass filter to a low pass filter.It is shows in Figure-7.

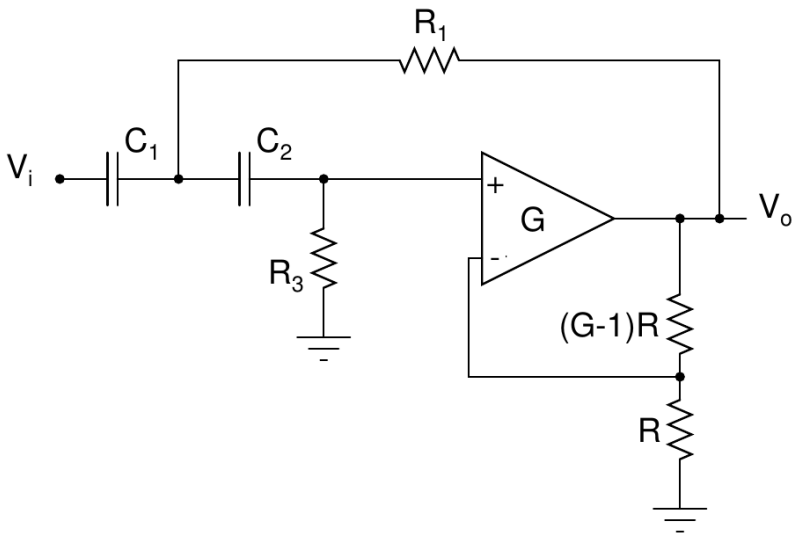

Figure 7: Circuit of the high pass filter

The transfer function of this high pass filter is found out exactly how we find out the transfer function for the low pass filter.

```
def highpass(c1,c2,r1,r2, G,Vi):
s = Symbol('s')
A = Matrix([[0,0,1,-1/G], [-1/(1+1/s*c2*r2), 1,0,0], [0,-G, G,1],\
[-s*c1-s*c2-1/r1, s*c2,0,1/r1]])
b = Matrix([0,0,0,Vi*s*c1])
V = A.inv()*b
return (A,b,V)
```

We call it with a particular set of values.

```
A,b,V = highpass(1e-9,1e-9, 1e4, 1e4,1.586,1)
```

```
Vo=V[3]
```

The method to plot the magnitude response is exactly similar to as we plotted the low pass filter. Figure-8 shows the impulse and step response of the high pass filter.

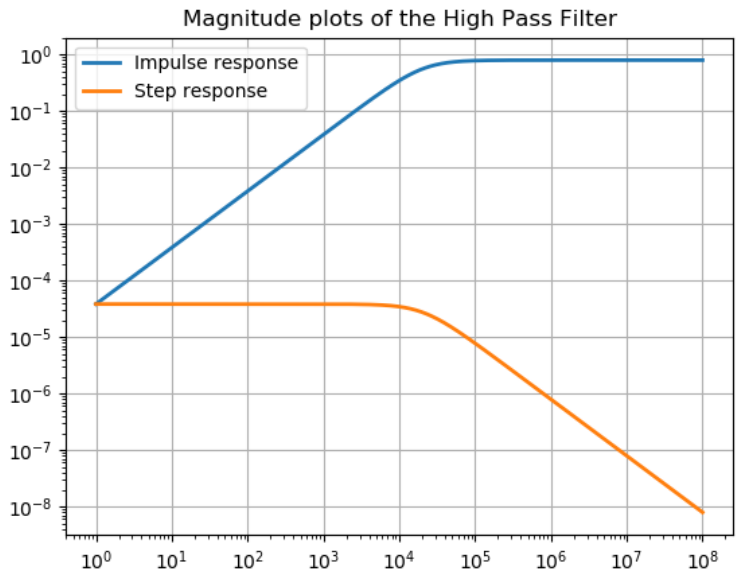

Figure 8: Magnitude plots of the High Pass Filter

From the figure it is clear that the gain of the high pass filter is very small for ω less than 10^5 and about 1 for frequencies beyond that.

3.1 The high pass filter in action

We pass $v_i(t)$ as defined earlier through the high pass filter. Figure-9 shows what happens to it.
As expected only the high frequency content remains.

3.2 Damped Sinusoid Through a High Pass Filter

Now we give two decaying sinusoids with different frequencies to the high pass filter and see its effect.Figures 10 and 11 show the desired values.

We expect the damped sinusoid of low frequency to be severely attenuated and the one with high frequency to go through with the same gain as the high pass filter(which we have realized by now, is approximately 0.8). This is exactly what happens in the plots.

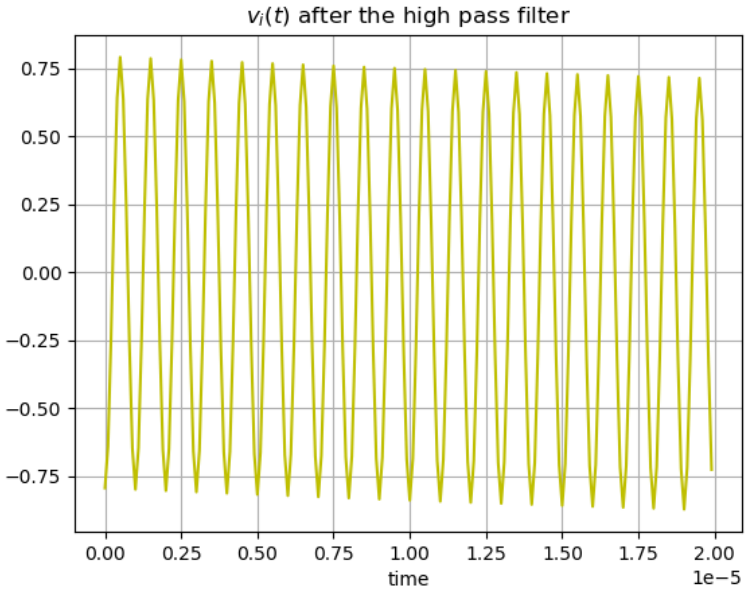

Figure 9: $v_i(t)$ through the high pass filter

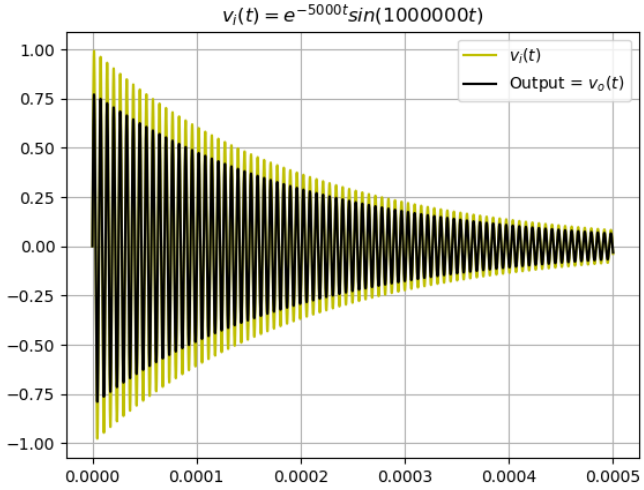

Figure 10: A high frequency damped sinusiod input

3.3 The Unit Step through the High Pass Filter

We pass the unit step function through the filters. We should expect to get a value of almost zero everywhere as the unit step is basically a DC and the high pass filter will cut it off completely.With a similar analogy, the low

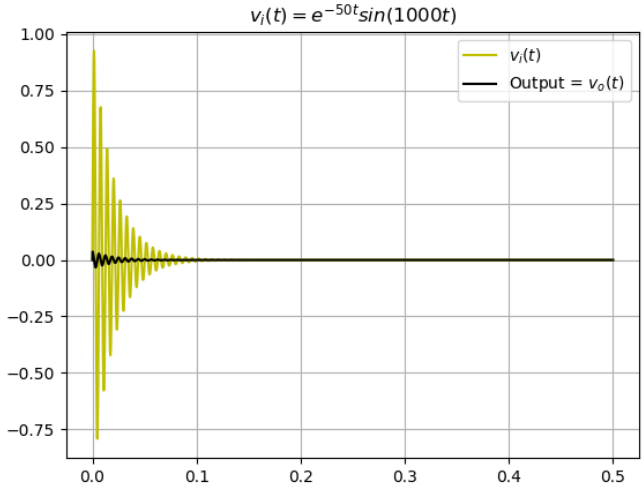

Figure 11: A low frequency damped sinusiod input

pass filter passes the entirety of the step response with some gain as a DC is basically a sinusoid of frequency zero. The following piece of code does that.

```
time = np.arange(100)
step_function = np.ones_like(time)
step_function[0]=0
step_function[1]=0.5
time,y,svec=sp.lsim(H_highpass_impulse,step_function,time)
time,y1,svec=sp.lsim(H_lowpass_impulse,step_function,time)
```

NOTE:
All parts of the assignment where we had to pass a given signal through a certain filter, it has been done in the like in the code mentioned for passing $v_i(t)$ through the low pass filter.

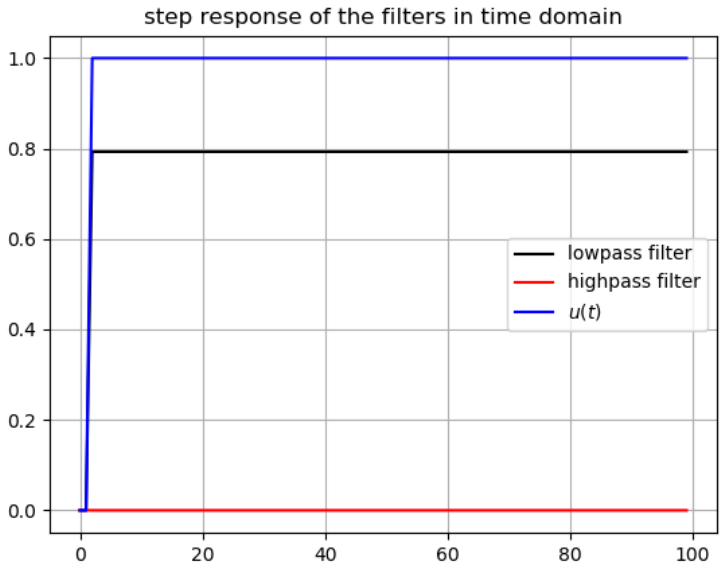

Figure 12: Unit step through the filters