Assignment 7: Circuit Analysis using Sympy

Name of the student: Balasubramaniam M C
Roll number- EE18B155

April 29, 2020

Abstract:

This assignment deals with solving analog circuits with KCL and KVL after converting them into the laplace domain. We will be using symbolic python to easily evaluate the output(using Matrix multiplication) in laplace domain . For plotting magnitude and phase response of a transfer function easily we use to 'lambdify' function to find the python function equivalent.
Given below are some functions of sympy we will be using in this assignment:

- .lti(numerator,denominator)- This function takes in the numerator and denominator coefficients of the transfer function and returns a form of the transfer function that can be used for convolution.
- .lsim(Transfer function, input signal, time array)- This function computes the convolution of a given input signal and a transfer function(it's time domain equivalent) after taking in three inputs- a time vector, input signal as a function of the time vector, an s-domain function which represents the transfer function.

Introduction:

This assignment can be roughly divided into two parts-

- The analysis of the low pass filter circuit shown in fig.1. We are given the transfer function Vo/Vin after some approximations done in the calculations. Using this we can plot the step response using the signals toolbox.
- The analysis of the high pass filter circuit as shown in fig.2. In this section we define a function to calculate the output Vo(in s-domain) given any input Vi(in s-domain) using KCL, KVL and matrix multiplication. More specifically, we do this for a given input function having low and high frequency sinusoids, a step function and a damped sinusoid.

The low pass filter circuit:

Evaluating the transfer function:

As seen from figure 1. and from KVL, KCL and the error equation for a non-ideal opamp, we have the following four equations:

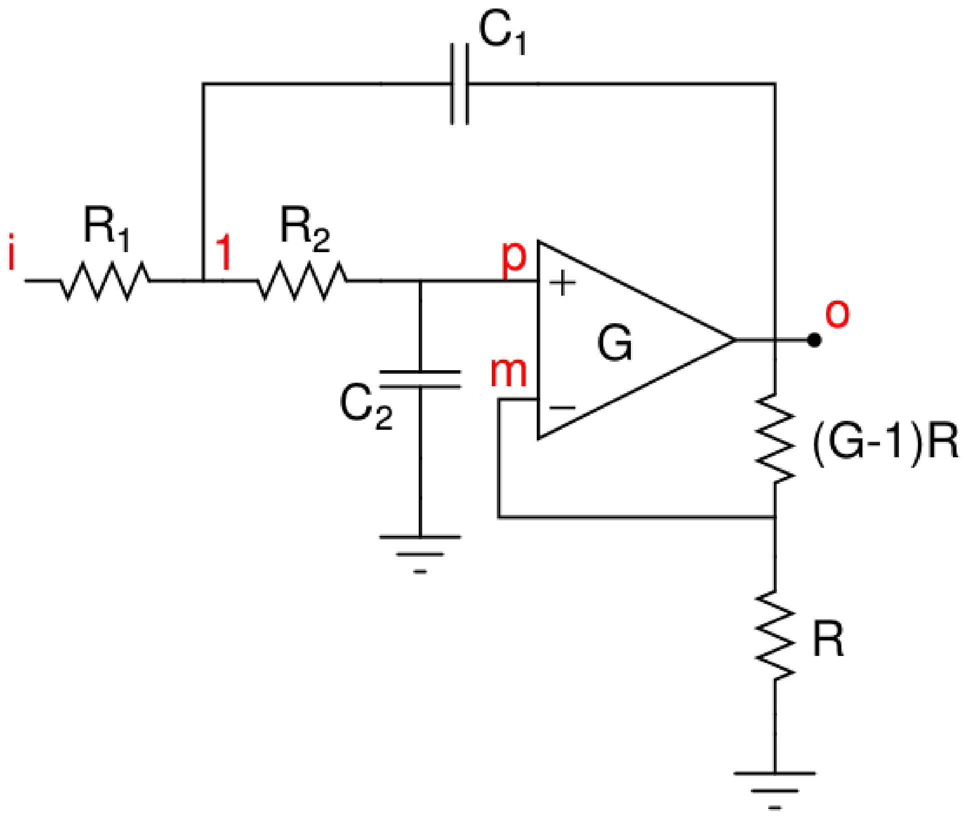

Figure 1: The low pass filter circuit

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

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

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

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

From equation 3, we get:

$$V_o = \frac{GV_1}{2} \frac{1}{1 + j\omega R_2 C_2} \tag{5}$$

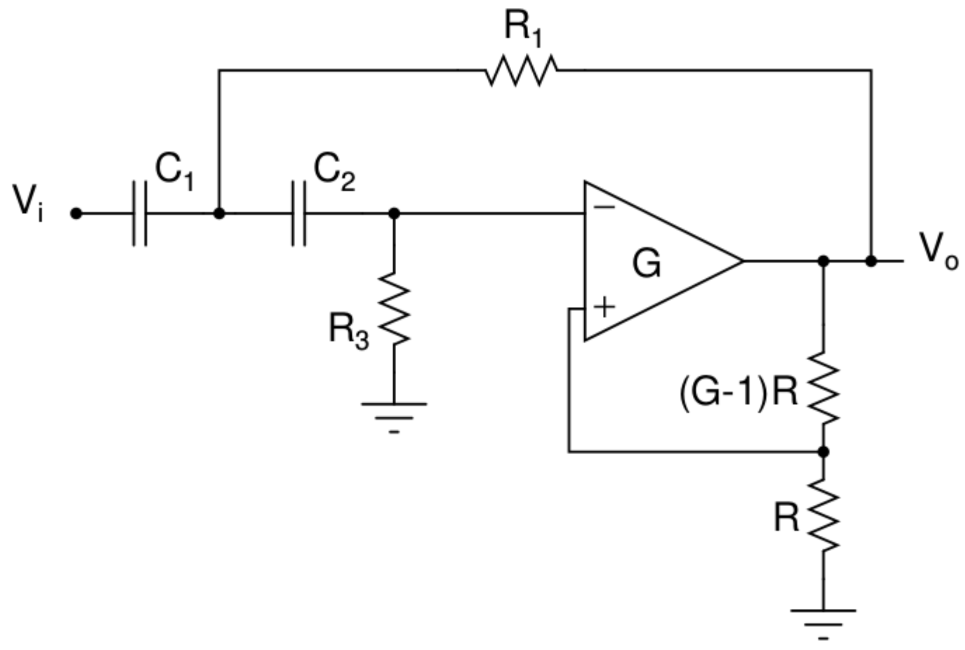

Figure 2: The high pass filter circuit

After assuming G is really high, we get an equation involving V1 and Vi:

$$V_1 = \frac{2V_i}{G}\frac{(1 + j\omega R_2 C_2)}{j\omega R_1 C_1} \tag{6}$$

Hence we get the following relation between Vi and Vo or the transfer function:

$$V_o = \frac{V_i}{j\omega R_1 C_1} \tag{7}$$

Finding unit step response:

We make the transfer function using the 'lti' function after defining the numerator and denominator coefficients and convolve this transfer function with a unit step function using the 'lsim' function and plot the result. This can be seen below:

```
S_T_Num = poly1d([1])
S_T_Den= poly1d([R1*C1,0])
S_T = sp.lti(S_T_Num, S_T_Den) #calculating transfer function
t1,y,svec= sp.lsim(S_T,u_t,t1) #convolving u_t and the transfer function
plot(t1,y) #plotting step response
```

The step response is seen in fig. 3.

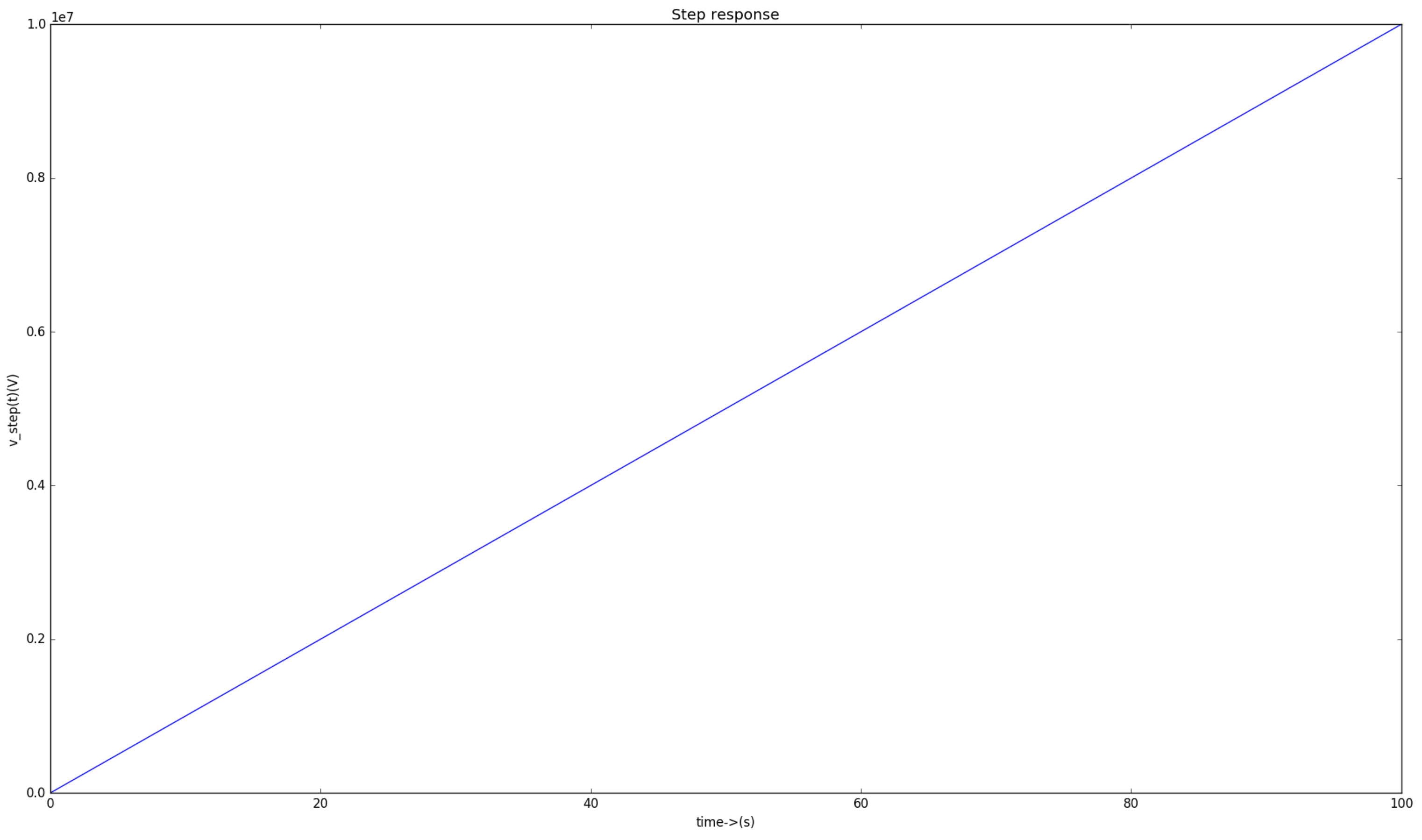

Figure 3: The step response of the low pass filter circuit

Analysis of the high pass filter circuit:

The high pass filter circuit as shown in fig.2 can be solved in a similiar way , but instead of approximations enter the coefficients of the equations in a matrix and solve it in the laplace domain. The 'symbols' function creates a variable s which can be multiplied divided and used in expressions as shown below:

```
s= symbols('s')# defining a symbol for s
init_printing() #enables a certain format of printing out equations
```

The following code shows how we solve the equations for the high pass filter for any input signal Vi,

```
def hpf(R1,C1,C2,R3,G,Vi,k):
s= symbols('s') #enables multiplication of expressions involving 's'

#defining the conduction matrix
A = Matrix([ [s*C1 +1/R1 +s*C2, -s*C2,0,-1/R1],
            [0, -1,1, 1/G],
            [0 , 0, 1, -1/k],
           [-(R3), (1/(s*C2)+R3), 0, 0] ])

#defining the matrix having RHS of all eqns.
b = Matrix( [s*C1*Vi,0,0,0])
#stores values of voltages at all involved nodes
V = A.inv()*b # stores output matrix having V1, Vp, Vm, Vo
return(A,b,V)
```

Hence, this function returns the output Voltages at all the nodes (in-s domain).

Evaluating response for input having high and low frequencies:

The input signal is as shown below:

$$v_i(t) = (sin(2000\pi t) + cos(2 * 10^6 \pi t))u_o(t) \tag{8}$$

In order to make things easier, we send in the laplace transform of each sinusoid (of lower and higher frequency) separately to the high pass filter function. Following which, we evaluate the inverse output signals individually and add them. This can be seen from the code shown below:

```
A,b,Vo2_cos = hpf(R1,C1,C2,R3,G,s/(s**2+(2*(1e6)*pi)**2),G1)
#receiving Transfer function cos part
coeff_num= poly1d([float(num_cos.coeff(s,3)),0,0,0])
coeff_den= poly1d([float(den_cos.coeff(s,i)) for i in range(4,-1,-1)])
#calculating coefficients of expanded numerator and denominator
Vo2_cos = sp.lti(coeff_num, coeff_den)
# transfer function cos

t,y2_cos3,svec = sp.lsim(Vo2_cos,d_t,t)
evaluating o/p signal

A,b,Vo2_sin = hpf(R1,C1,C2,R3,G,(2000*pi)/(s**2+(2000*pi)**2),G1)
#receiving Transfer function sin part
.
.
.
Similarly for sine


plot(t,y2_sin3+y2_cos3)
#plotting the sum of signals
```

The output signal can be seen in fig. 4

Evaluating response for damped sinusoidal input:

The damped signal we will be using is as shown below:

$$e^{(-0.05t)} * cos(1e5t)u(t) \tag{9}$$

We send in the laplace transform of the above to the 'hpf' function and receive the output function(Vo) as shown below:

```
Vi_3_T = (s+0.05)/( (s+0.05)**2 + (1e5)**2 )
A,b,Vo_damp = hpf(R1,C1,C2,R3,G,Vi_3_T,G1)
#receiving Transfer function
```

The output is calculated in a similar manner as for the earlier case i.e, by finding the coefficients of all powers of 's' in the numerator and denominator and evaluating the transfer function and using lsim to evaluate the output signal. This can be seen below:

```
num_damp,den_damp=Vo_damp[3].as_numer_denom()
coeff_num_damp= poly1d([float(num_damp.coeff(s,i)) for i in range(3,-1,-1)])
coeff_den_damp = poly1d([float(den_damp.coeff(s,i)) for i in range(4,-1,-1)])
#finding coefficients
Vo2_damp = sp.lti(coeff_num_damp, coeff_den_damp)
#transfer function for plotting
t2,y_damp,svec = sp.lsim(Vo2_damp,d_t,t2) #finding o/p signal
```

The output can be seen in fig.5

Evaluating response for unit step input:

We send in 1/s as input to the 'hpf' function and receive the output function (Vo). This can be seen below:

```
A,b,Vo_unit = hpf(R1,C1,C2,R3,G,1/s,G1) #receiving Transfer function
```

Again, the output is calculated in a similar manner as for the earlier case i.e, by finding the coefficients of all powers of 's' in the numerator and denominator and evaluating the transfer function and using lsim to evaluate the output signal. This can be seen below:

```
num_unit,den_unit=Vo_unit[3].as_numer_denom()
num_unit = num_unit.expand()
den_unit = den_unit.expand()
#splitting into numerator and denominator
coeff_num_unit= poly1d([float(num_unit.coeff(s,i)) for i in range(1,-1,-1)])
coeff_den_unit = poly1d([float(den_unit.coeff(s,i)) for i in range(2,-1,-1)])
#finding coefficients

Vo2_unit = sp.lti(coeff_num_unit, coeff_den_unit)
#transfer function for plotting
t3,y_unit,svec = sp.lsim(Vo2_unit,d_t3,t3)
#finding o/p signal
```

The output can be seen in fig.6

Observation and Conclusions:

- The step response for the low pass filter is justified as shown in fig.3 as we know that it is of the form $1/(s^2)$ whose inverse laplace transform is a linear function in time.

- The output signal for the high pass filter for a given input of two sinusoids(that of a high and low frequency) is shown in fig. 4 where we can see that the cosine(freq-10^6 Hz) is amplified and the sine function (freq- 10^3 Hz) is attenuated for obvious reasons

- The response to a damped sinusoid input is as shown in fig.5 where the high frequency wave appears in a modulated form (probably arising from the properties of the circuit) and it is ofcourse damped as the input is damped.

- The step response of the high pass filter circuit is shown in fig.6 . It looks like a graph with a steep drop and a slow rise upto a particular stable voltage. This is because of slight ringing in the circuit whose zeta (damping factor) would be ¡ 1. Hence, it's not a very good high pass filter circuit whose output is reached as fast as possible.

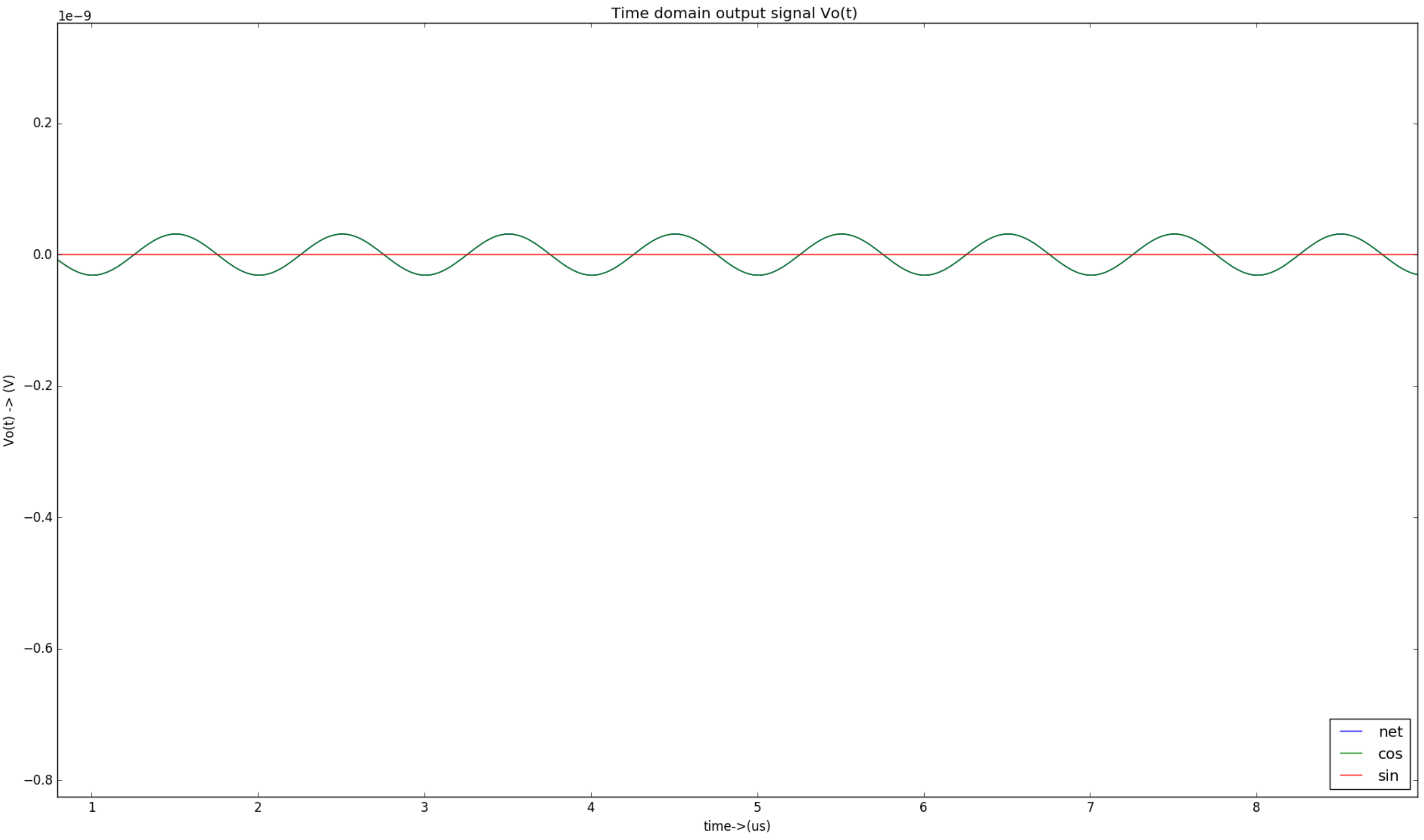
Time domain output signal Vo(t)
1e−9
0.2
0.0
−0.2
−0.4
−0.6
−0.8
Vo(t) -> (V)
1
2
3
4
5
6
7
8
time->(us)
net
cos
sin

Figure 4:

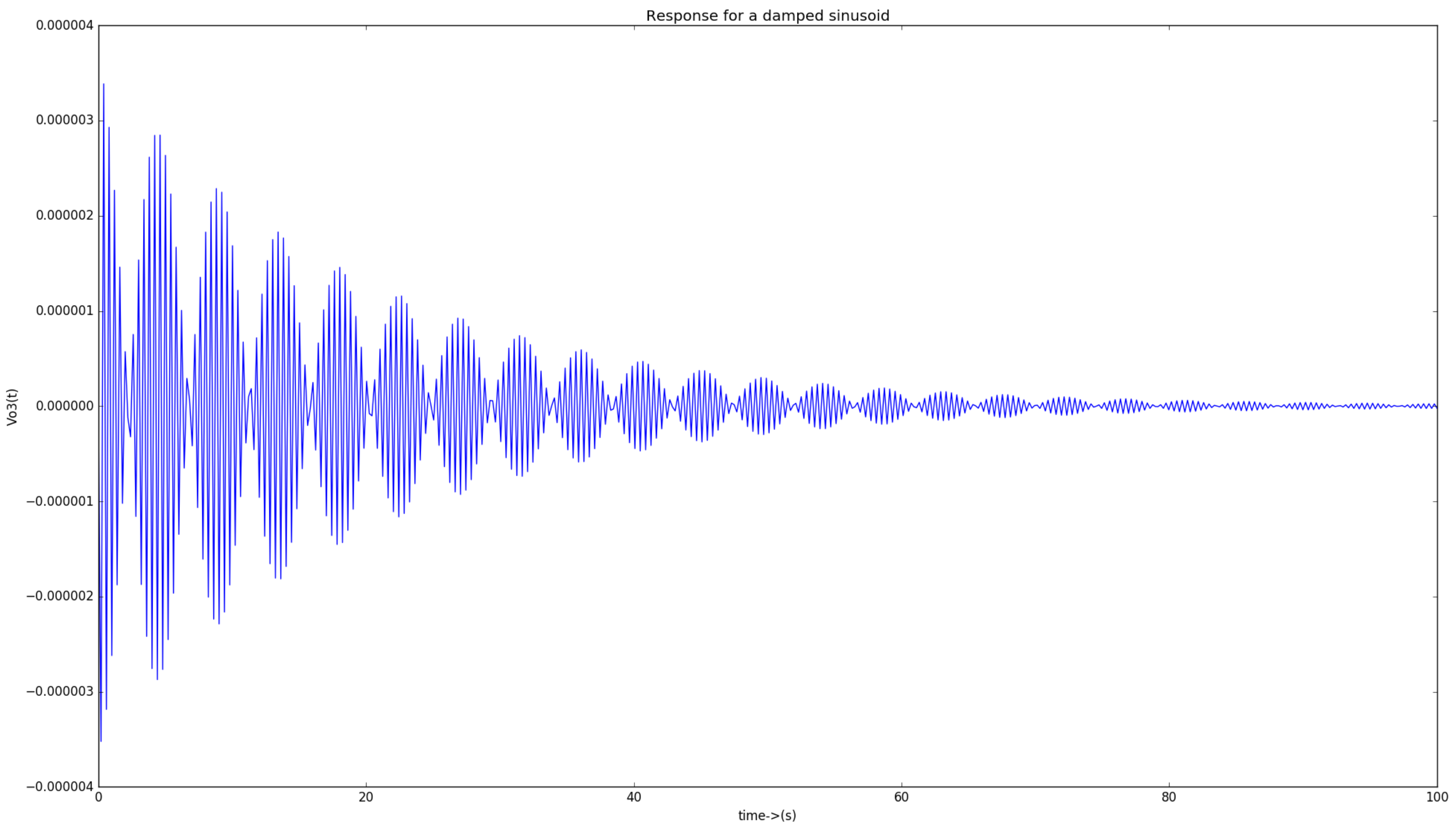
Response for a damped sinusoid
0.000004
0.000003
0.000002
0.000001
0.000000
−0.000001
−0.000002
−0.000003
−0.000004
Vo3(t)
0
20
40
60
80
100
time->(s)

Figure 5:

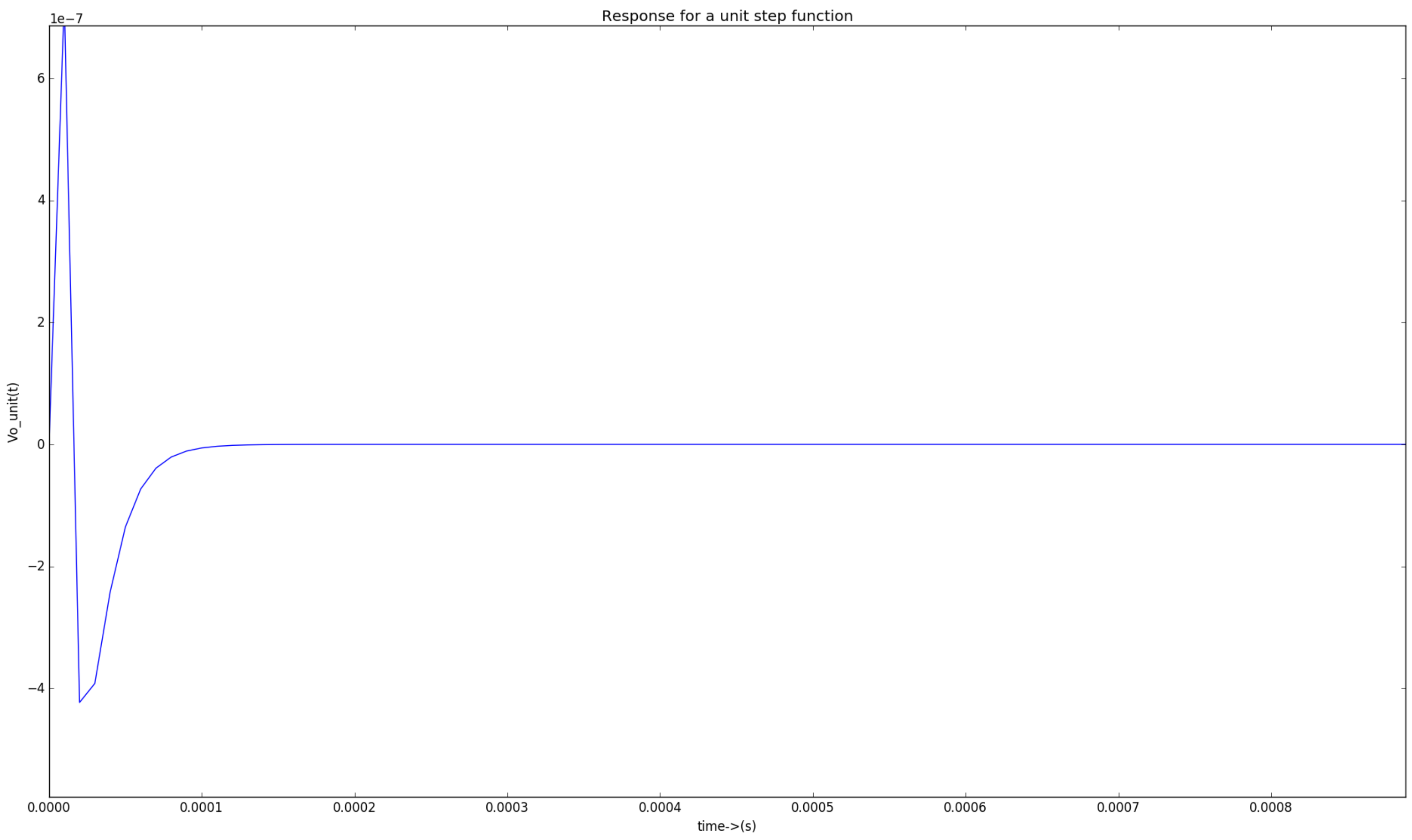

Figure 6: Step Response for High pass filter