Assignment 7: Circuit Analysis using Sympy

Manognya Param Koolath

March 19, 2019

The task

In this assignment, we foucus on two powerful capabilities of python which are symbolic algebra and analysis of circuits using Laplace Transforms.

Question 1

Consider the circuit shown in Figure 1. Writing KVL and KCL equations in matrix form, we get:

$$\begin{bmatrix} 0 & 0 & 1 & \frac{-1}{G} \\ \frac{-1}{1+sR_2C_2} & 1 & 0 & 0 \\ 0 & -G & G & 1 \\ \frac{-1}{R_1} - \frac{1}{R_2} - sC_2 & \frac{1}{R_2} & 0 & sC_1 \end{bmatrix} \begin{bmatrix} V_1 \\ V_p \\ V_m \\ V_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{V_i(s)}{R_1} \end{bmatrix}$$

The given circuit is clearly a low-pass filter. A function is defined which

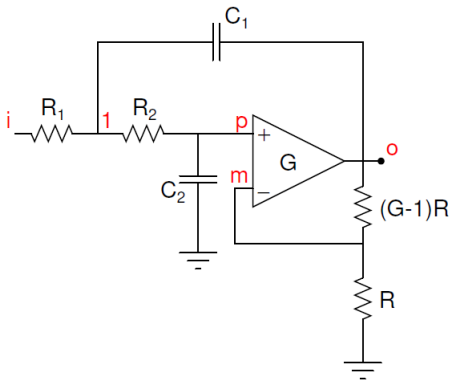

Figure 1: Circuit for Questions 1 and 2

gives the expression of $V_0(s)$ in terms of other known variables. For an input $V_i(s)$ of 1, the $V_0(s)$ we get is the transfer function of the circuit. It's magnitude response is plotted in Figure 2. For input $V_i(s) = \frac{1}{s}$, the response (step response) is as shown in Figure 3.

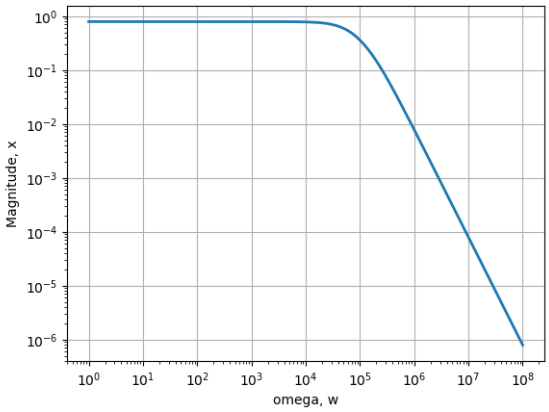

Figure 2: Transfer function of the low pass filter

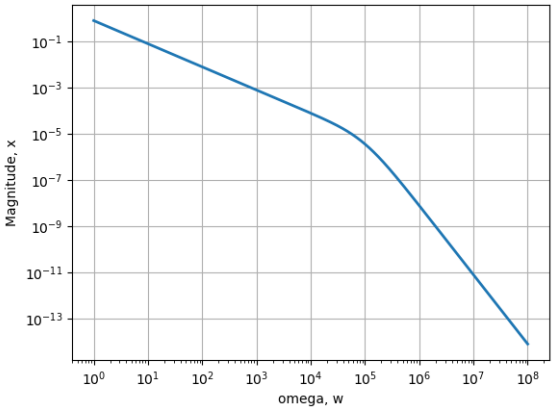

Figure 3: Unit step response of the low pass filter(Question 1)

Question 2

For the given input $v_i(t)$, we have to find the output voltage. For this, the input in s-domain is calculated as:

$$V_i(s) = \frac{s^3 + 2e3\pi s^2 + 4e6\pi^2 s + 8e15\pi^3}{s^4 + 4e12\pi^2 s^2 + 16e18\pi^4}$$

We give this input to the function `lowpass` to get the expression for $V_0(s)$ which is converted to time domain. The magnitude response of this is shown in Figure 4.

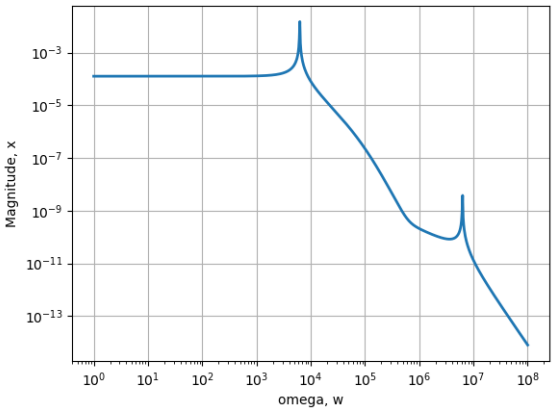

Figure 4: Magnitude response of $V_0(s)$(Question 2)

Question 3

From this question onwaards, we focus on circuit 2, shown in Figure 5, which is clearly a high pass filter. The KVL, KCL matrix for this equation is as follows, which is put in a function to solve for $V_i(s)$ in s-domain.

$$\begin{bmatrix} 0 & 0 & G & -1 \\ sC_2R_3 & -1-sC_2R_3 & 0 & 0 \\ 0 & G & -G & 1 \\ 1+sC_2R_1+sC_1R_1 & -sC_2R_1 & 0 & -1 \end{bmatrix} \begin{bmatrix} V_1 \\ V_p \\ V_m \\ V_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ sC_1R_1V_i \end{bmatrix}$$

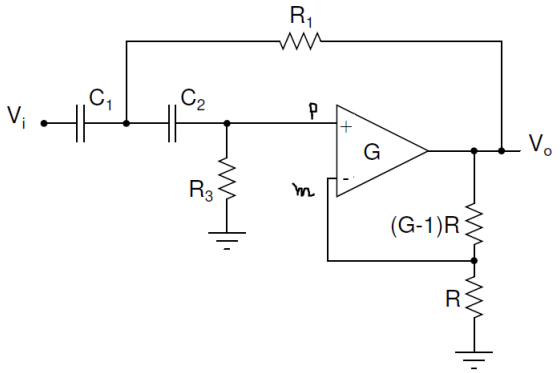

Figure 5: Circuit 2 for questions 3,4 and 5

From this, we can find the transfer function of the system which is shown in Figure 6, from which we can infer that the circuit is a high pass filter.

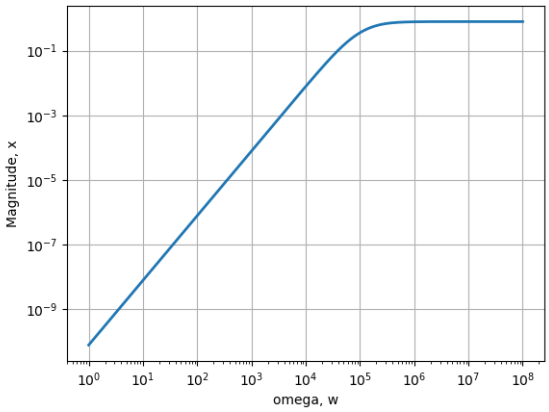

Figure 6: Magnitude response of $V_0(s)$ (Question 3)

Question 4

The follwing a damped sinusoid is used as input:

$$v_i(t) = \sin(2000t)e^{-t}$$

$v_i(t)$ is plotted in Figures 7 and 8. Its s-domain expression is given as:

$$V_i(s) = \frac{2e3}{(s+1)^2 + 4e6}$$

$V_i(s)$ is given as input to the function `highpass` and the output in time domain is plotted in Figures 10 and 11. Figure 9 shows the magnitude response of the output $V_0(s)$.

Question 5

In this, we find the unit step response of the circuit. This is plotted in Figure 12.

Python code

The code is properly commented and completely vectorised.

Assignment 7 Code

```
#Assignment 7
import numpy as np
import matplotlib.pyplot as plt
```

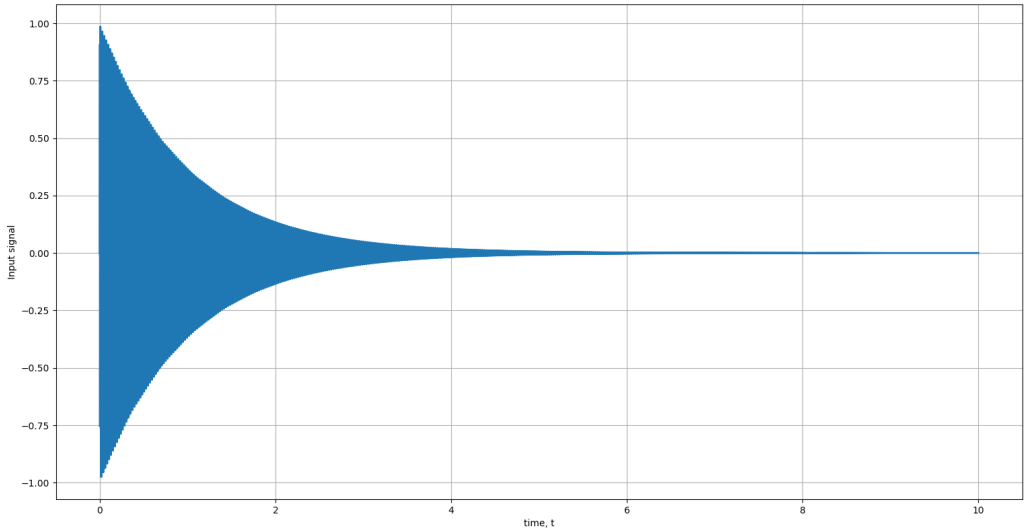

Figure 7: $v_i(t)$ v/s t in Question 4

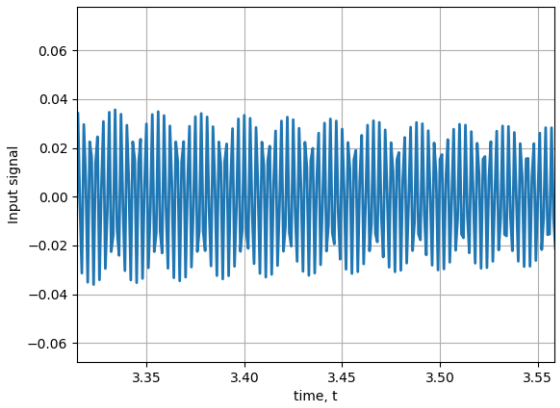

Figure 8: Zoomed plot of $v_i(t)$ v/s t in Question 4

```
import scipy.signal as sp
from scipy.linalg import lstsq
import scipy.integrate as spint
import mpl_toolkits.mplot3d.axes3d as p3
from scipy import ndimage
import sympy as spy
```

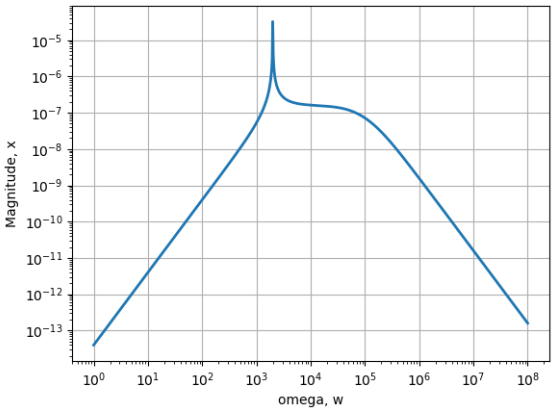

Figure 9: Magnitude response of $V_0(s)$ (Question 4)

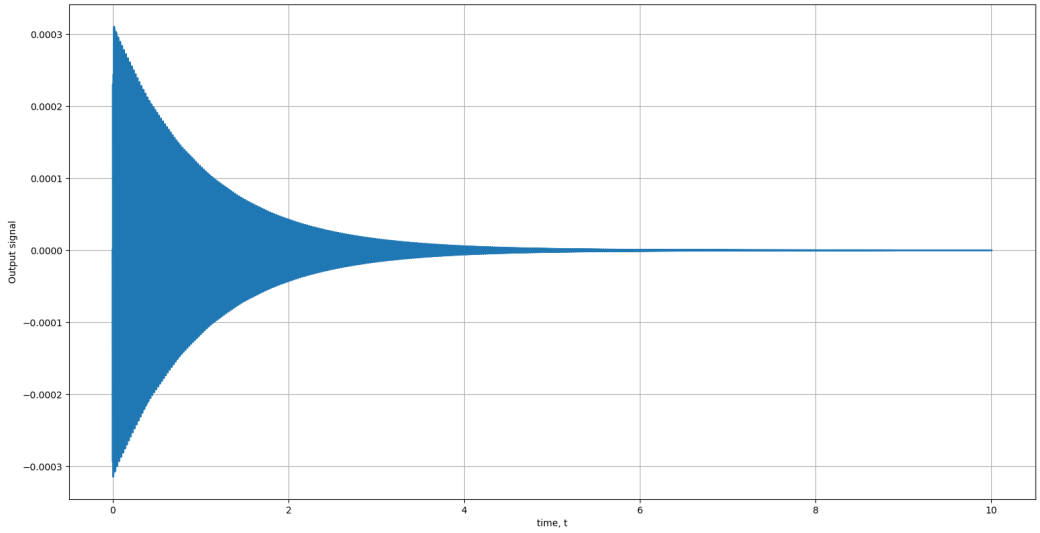

Figure 10: $v_0(t)$ v/s t in Question 4

```
s = spy.symbols('s')

def lowpass(R1,R2,C1,C2,G,Vi):
        A = spy.Matrix([[0,0,1,-1/G],[-1/(1+s*R2*C2),1,0,0],[0,-G,G,1],[-
        b = spy.Matrix([0,0,0,Vi/R1])
        V = A.inv()*b #Output voltage is V[3]
        return (A,b,V)
```

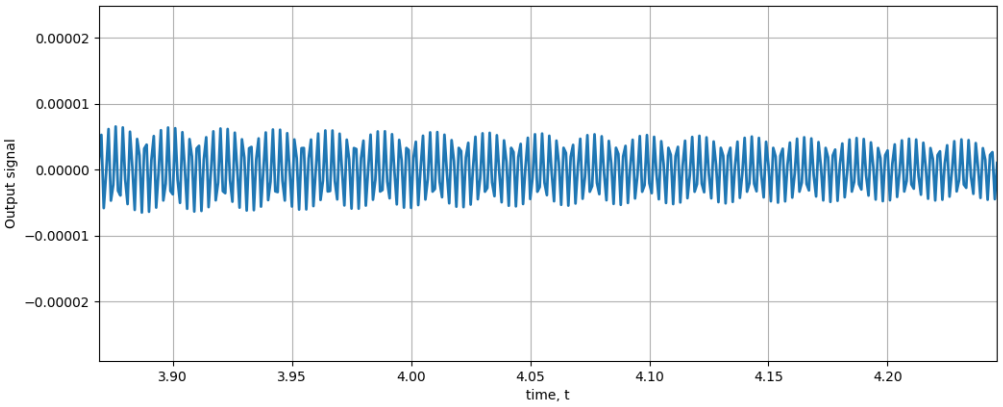

Figure 11: Zoomed plot of $v_0(t)$ v/s t in Question 4

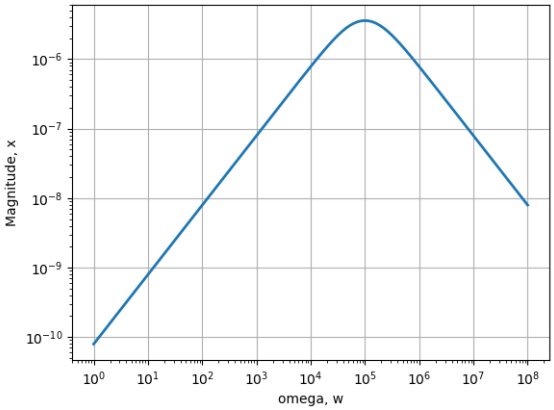

Figure 12: Unit step response of $V_0(s)$ (Question 5)

```
def highpass(R1,R3,C1,C2,G,Vi):
        A = spy.Matrix([[0,0,G,-1],[s*C2*R3,-1-s*C2*R3,0,0],[0,G,-G,-1],[1
        b = spy.Matrix([0,0,0,s*C1*R1*Vi])
        V = A.inv()*b #Output voltage is V[3]
        return (A,b,V)

def get_coeffs(expr):
        num,den = expr.as_numer_denom()
        return [spy.Poly(num,s).all_coeffs(), spy.Poly(den,s).all_coeffs(

A,b,V = lowpass(10000,10000,1e-9,1e-9,1.586,1)
```

```
# print ('G=1000')
Vo=V[3]
# print(Vo)
w=np.logspace(0,8,801)
ss=1j*w
hf = spy.lambdify(s,Vo, 'numpy')
v=hf(ss)
plt.loglog(w,abs(v),lw=2)
plt.xlabel('omega, w')
plt.ylabel('Magnitude, x')
plt.grid(True)
plt.show()

#Q1:
A,b,V = lowpass(10000,10000,1e-9,1e-9,1.586,1/s)
# print ('G=1000')
Vo=V[3]
# print(Vo)
w=np.logspace(0,8,801)
ss=1j*w
hf = spy.lambdify(s,Vo, 'numpy')
v=hf(ss)
plt.loglog(w,abs(v),lw=2)
plt.xlabel('omega, w')
plt.ylabel('Magnitude, x')
plt.grid(True)
plt.show()

#Q2:
#First term:
# Num = np.poly1d([2000*np.pi])
# Den = np.poly1d([1,0,(2000*np.pi)**2])
# Vi1 = sp.lti(Num,Den)
# #Second term:
# Num = np.poly1d([1,0])
# Den = np.poly1d([1,0,(2e6*np.pi)**2])
# Vi2 = sp.lti(Num,Den)
# Vi = Vi1 + Vi2
# print(Vi)
Vi = (s**3 + (2e3*np.pi*s**2) + (4e6*(np.pi**2)*s + 8e15*np.pi**3))/(s**4
A,b,V = lowpass(10000,10000,1e-9,1e-9,1.586,Vi)
# print ('G=1000')
Vo=V[3]
print(Vo)
```

```
w=np.logspace(0,8,801)
ss=1j*w
hf = spy.lambdify(s,Vo,'numpy')
v=hf(ss)
plt.loglog(w,abs(v),lw=2)
plt.xlabel('omega, w')
plt.ylabel('Magnitude, x')
plt.grid(True)
plt.show()

# n,d=get_coeffs(Vo)
# # print(n)
# # print(d)
# n=[float(x) for x in n]
# d=[float(x) for x in d]
# Vo_s = sp.lti(n,d)
# # print(n,d)
# # print("hello")
# t,v_o = sp.impulse(Vo_s,None,np.linspace(0,10,10))
# print(v_o)
# # Output time domain signal:
# plt.plot(t,v_o,lw=2)
# plt.xlabel('time, t')
# plt.ylabel('Output signal')
# plt.grid(True)
# plt.show()

#Q3:
A,b,V = highpass(10000,10000,1e-9,1e-9,1.586,1)
# print ('G=1000')
Vo=V[3]
# print(Vo)
w=np.logspace(0,8,801)
ss=1j*w
hf = spy.lambdify(s,Vo,'numpy')
v=hf(ss)
plt.loglog(w,abs(v),lw=2)
plt.xlabel('omega, w')
plt.ylabel('Magnitude, x')
plt.grid(True)
plt.show()

#Q4:
```

```
#Damped sinusoid: sin 2000t e**(-t)
Vi=2e3/((s+1)**2 + 4e6)
n,d=get_coeffs(Vi)
# print(n)
# print(d)
n=[float(x) for x in n]
d=[float(x) for x in d]
Vi_s = sp.lti(n,d)
# print(n,d)
# print("hello")
t,v_i = sp.impulse(Vi_s,None,np.linspace(0,10,10001))
# Plot of input time domain signal:
plt.plot(t,v_i,lw=2)
plt.xlabel('time, t')
plt.ylabel('Input signal')
plt.grid(True)
plt.show()

A,b,V = highpass(10000,10000,1e-9,1e-9,1.586,Vi)
# print ('G=1000')
Vo=V[3]
# print(Vo)
w=np.logspace(0,8,801)
ss=1j*w
hf = spy.lambdify(s,Vo,'numpy')
v=hf(ss)
plt.loglog(w,abs(v),lw=2)
plt.xlabel('omega, w')
plt.ylabel('Magnitude, x')
plt.grid(True)
plt.show()
# print(hf)

n,d=get_coeffs(Vo)
# print(n)
# print(d)
n=[float(x) for x in n]
d=[float(x) for x in d]
Vo_s = sp.lti(n,d)
# print(n,d)
# print("hello")
t,v_o = sp.impulse(Vo_s,None,np.linspace(0,10,10001))
# Output time domain signal:
plt.plot(t,v_o,lw=2)
```

```
plt.xlabel('time, t')
plt.ylabel('Output signal')
plt.grid(True)
plt.show()

# #Damped sinusoid: sin t e**(-t)
# Vi=1/((s+1)**2 + 1)
# n,d=get_coeffs(Vi)
# # print(n)
# # print(d)
# n=[float(x) for x in n]
# d=[float(x) for x in d]
# Vi_s = sp.lti(n,d)
# # print(n,d)
# # print("hello")
# t, v_i = sp.impulse(Vi_s,None,np.linspace(0,10,10001))
# # Plot of input time domain signal:
# plt.plot(t,v_i,lw=2)
# plt.xlabel('time, t')
# plt.ylabel('Input signal')
# plt.grid(True)
# plt.show()

# A,b,V = highpass(10000,10000,1e-9,1e-9,1.586,Vi)
# # print ('G=1000')
# Vo=V[3]
# # print(Vo)
# w=np.logspace(0,8,801)
# ss=1j*w
# hf = spy.lambdify(s,Vo,'numpy')
# v=hf(ss)
# plt.loglog(w,abs(v),lw=2)
# plt.xlabel('omega, w')
# plt.ylabel('Magnitude, x')
# plt.grid(True)
# plt.show()
# # print(hf)

# n,d=get_coeffs(Vo)
# # print(n)
# # print(d)
# n=[float(x) for x in n]
# d=[float(x) for x in d]
# Vo_s = sp.lti(n,d)
```

```
## print(n,d)
## print("hello")
# t,v_o = sp.impulse(Vo_s,None,np.linspace(0,10,10001))
## Output time domain signal:
# plt.plot(t,v_o,lw=2)
# plt.xlabel('time, t')
# plt.ylabel('Output signal')
# plt.grid(True)
# plt.show()

##Damped sinusoid: sin 2e10t e**(-t)
# Vi=2e10/((s+1)**2 + 4e20)
# n,d=get_coeffs(Vi)
## print(n)
## print(d)
# n=[float(x) for x in n]
# d=[float(x) for x in d]
# Vi_s = sp.lti(n,d)
## print(n,d)
## print("hello")
# t,v_i = sp.impulse(Vi_s,None,np.linspace(0,10,10001))
## Plot of input time domain signal:
# plt.plot(t,v_i,lw=2)
# plt.xlabel('time, t')
# plt.ylabel('Input signal')
# plt.grid(True)
# plt.show()

# A,b,V = highpass(10000,10000,1e-9,1e-9,1.586,Vi)
## print ('G=1000')
# Vo=V[3]
## print(Vo)
# w=np.logspace(0,8,801)
# ss=1j*w
# hf = spy.lambdify(s,Vo,'numpy')
# v=hf(ss)
# plt.loglog(w,abs(v),lw=2)
# plt.xlabel('omega, w')
# plt.ylabel('Magnitude, x')
# plt.grid(True)
# plt.show()
## print(hf)

# n,d=get_coeffs(Vo)
```

```
## print(n)
## print(d)
# n=[float(x) for x in n]
# d=[float(x) for x in d]
# Vo_s = sp.lti(n,d)
## print(n,d)
## print("hello")
# t,v_o = sp.impulse(Vo_s,None,np.linspace(0,10,10001))
## Output time domain signal:
# plt.plot(t,v_o,lw=2)
# plt.xlabel('time, t')
# plt.ylabel('Output signal')
# plt.grid(True)
# plt.show()


#Q5:
A,b,V = highpass(10000,10000,1e-9,1e-9,1.586,1/s)
# print ('G=1000')
Vo=V[3]
# print(Vo)
w=np.logspace(0,8,801)
ss=1j*w
hf = spy.lambdify(s,Vo,'numpy')
v=hf(ss)
plt.loglog(w,abs(v),lw=2)
plt.xlabel('omega, w')
plt.ylabel('Magnitude, x')
plt.grid(True)
plt.show()

# THE END
```