ECE 351 - 52

Signals and Systems 1

Lab 4

Submitted By :
Owen Blair

Contents

1 Important Notes

It is important to note that plt.show() is needed at the end of the python code to properly show the plots of each function. It was not included in every section of code that plots a function(s) because it is assumed that its included at the end the code.

The web address to the GitHub where LaTeX code is stored is here:
https://github.com/Blairis123/ECE351_Reports

The web address to the GitHub where the Python code is here:
https://github.com/Blairis123/ECE351_Code

2 Part 1 Deliverables

The goal of part one is to plot the impulse response of the circuit in *Figure 1: Circuit*. This was done in two ways. One was a hand solved time domain response implemented as a function and the second is using scipy.signal.impulse() function with the s-domain transfer function of the circuit. The function that was found by hand can be found in *Listing 1: Hand Solved Function Using Python* and the plot of the hand solved impulse response copared to the scipy.signal.impulse() plot can be found in *Figure 2: Hand-Solved Vs signal.impulse()*.

3 Part 2 Deliverables

Part two is about comparing the step response of the circuit to the impulse response of the circuit. The step response is found by usig the scipy.signal.step() function and compared to the impulse response found in Part 1. The code to generate and plot the scipy.signal.step() compared with the scipy.signal.impulse() can be found in *Listing 2: Step Response* and the plot in *Figure 3: Step Vs Impulse*. The equation for the final value theorem can be found in the *Equations Used* section of this report. A further discussion of the final value theorem can be fond in the *Equations Used* section as well.

4 Equations

Equation 1: Definition of the Final Value Theorem

$$\lim_{t\to\infty} h(t) = \lim_{s\to 0} sH(s)$$

Given this we can then input the values of R, L, and C from *Figure 1.* This results in about:

$$\lim_{t\to\infty}(\frac{g}{\omega} * e^{-\frac{1}{2}*t} * sin(\omega t + \angle g)) = \lim_{s\to 0}(\frac{10^4 * s}{s^2 * 10^4 * s + 3.7037 * 10^8})$$

Knowing that $g = -\frac{1}{2} * 18584j = 18584\angle 90.002$ and $\alpha = -\frac{1}{2}$ and is multiplied by t, it is safe to say that $\lim_{t\to\infty} h(t) = 0$. Regarding $\lim_{s\to 0} sH(s)$ it can be seen that the numerator goes to zero and the denominator goes to $\frac{1}{L*C}$ so $sH(s)$ must also go to zero.

5 Results

Figure 1: Circuit

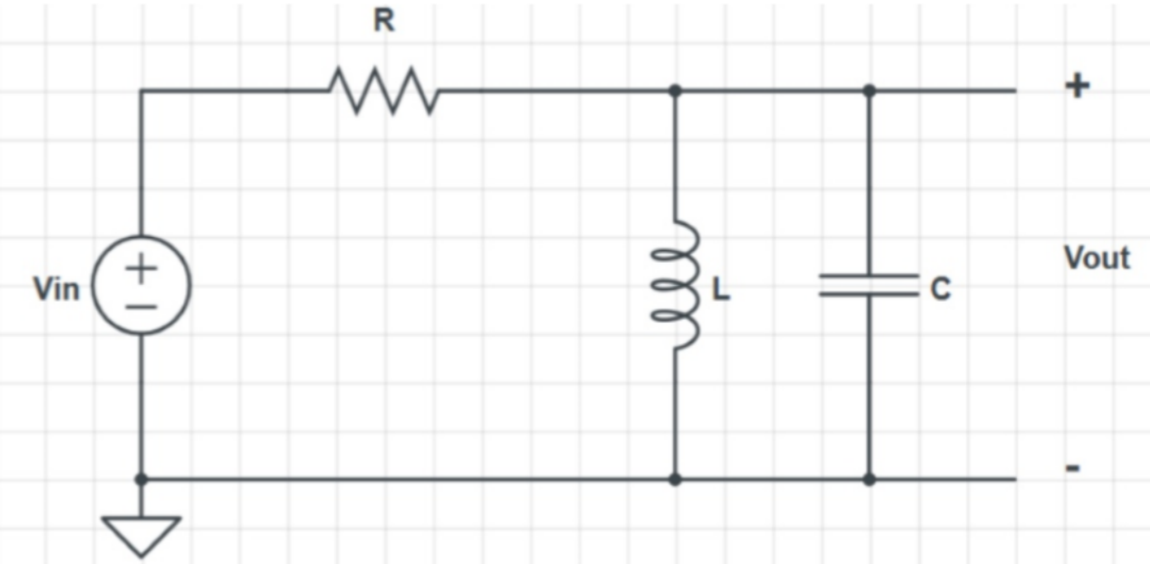


Figure 1: R = 1kΩ, L = 27 mH, C = 100 nF

Figure 2: Hand-Solved Vs signal.impulse()

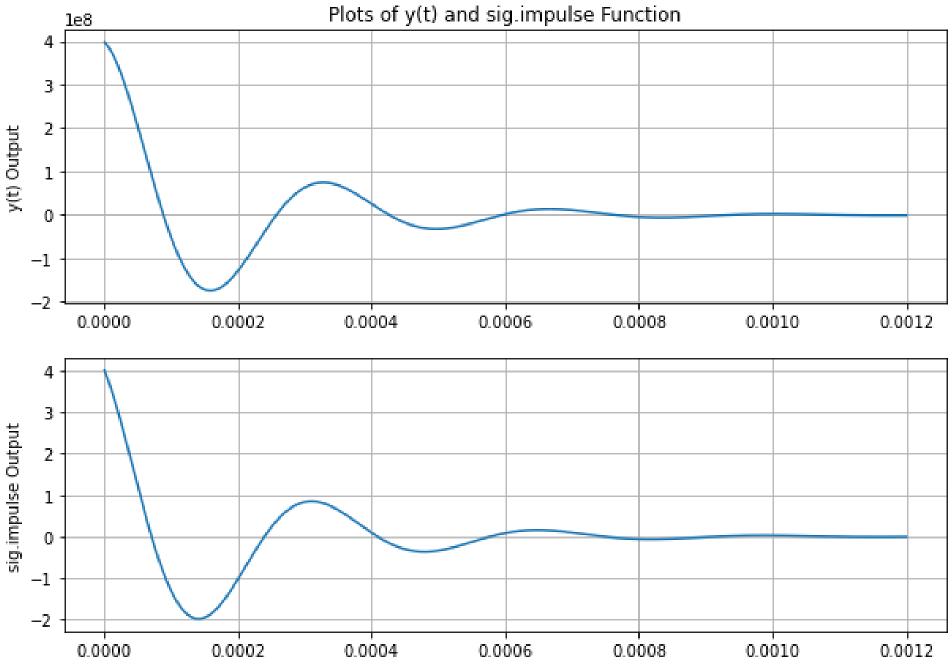
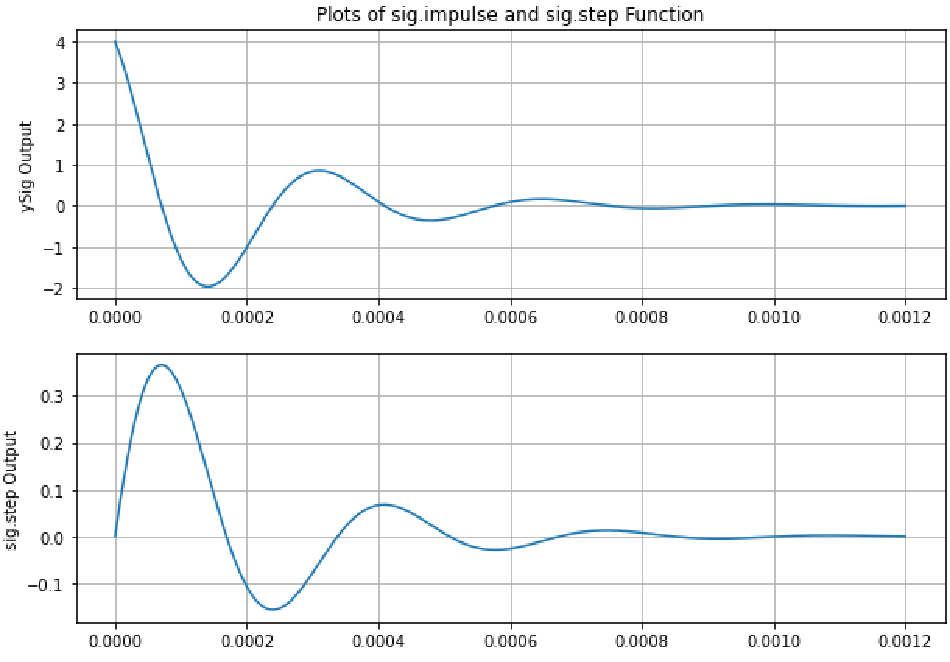


Figure 3: Step Vs Impulse



6 Listings

Listing 1: Hand Solved Function Using Python

```
    #DEFINE FUNCTION TO USE FOR HAND SOLVED PLOT
def func2Plot(t, R, L, C):

    X = (1/(R*C))
    tmpX = -0.5 * ((1/(R*C))**2)
    tmpY = 0.5 * X * (m.sqrt((X**2)) - (4/(L*C)))

    magOfG = m.sqrt( (tmpX**2) + (tmpY**2))

    tmpTanX = 0.5 * (m.sqrt((X**2)) - (4/(L*C)))
    tmpTanY = -0.5 * ((1/(R*C))**2)

    degOfG = m.atan(tmpTanX / tmpTanY)

    omega = 0.5 * np.sqrt(X**2 - 4*(1/np.sqrt(L*C))**2 + 0*1j)
    alpha = (-0.5) * X

    y = (magOfG/np.abs(omega)) * np.exp(alpha * t) * np.sin(np.abs(
   omega) * t + degOfG) * stepFunc(t, 0, 1)

    return y
#--------------END FUNCTIONS-----------------------------#

    #Make steps for t! From 0 to 1.2 ms this would be
steps = 1e-6

    #t for part 1
start = 0
stop = 1.2e-3
    #Define a range of t_pt1. Start @ 0 go to 20 (+a step) w/
    #a stepsize of step
t = np.arange(start, stop + steps, steps)

#-------PT1----------------------

    #For circuit-------------R L C DEF----------
R = 1e3
L = 27e-3
C = 100e-9

y = func2Plot(t, R, L, C)

num = [0, 1/(R*C), 0] #Creates a matrix for the numerator
den = [1, 1/(R*C), (1/(L*C))] #Creates a matrix for the denominator
```

```
tout , ySig = sig.impulse ((num , den), T = t)

ySig = ySig * (4e-4)
    #Make plots
plt.figure(figsize=(10,7))
plt.subplot(2,1,1)
plt.plot(t,y)
plt.grid()
plt.ylabel('y(t) Output')
plt.title('Plots of y(t) and sig.impulse Function')

plt.subplot(2,1,2)
plt.plot(t,ySig)
plt.grid()
plt.ylabel('sig.impulse Output')
```

Listing 2: Step Response

```
    #For circuit-------------RLC DEF----------
R = 1e3
L = 27e-3
C = 100e-9

y = func2Plot(t, R, L, C)

num = [0, 1/(R*C), 0] #Creates a matrix for the numerator
den = [1, 1/(R*C), (1/(L*C))] #Creates a matrix for the denominator

tout , ySig = sig.impulse ((num , den), T = t)

ySig = ySig * (4e-4)
#-----------------------PT2 CODE-----------
tout2 , yStep = sig.step ((num , den), T = t)

#---------------PT 2 PLOTS----------------
plt.figure(figsize=(10,7))
plt.subplot(2,1,1)
plt.plot(t,ySig)
plt.grid()
plt.ylabel('ySig Output')
plt.title('Plots of sig.impulse and sig.step Function')

plt.subplot(2,1,2)
plt.plot(t,yStep)
plt.grid()
plt.ylabel('sig.step Output')
```

[language=Python]

7 Questions

7.1 Question 1