ECE 2260 - Fundamentals of Electrical Circuits: Lab 6

Hunter Van Horn

March 13, 2025

Purpose

This lab introduces resonance in an RLC circuit.

Preliminary

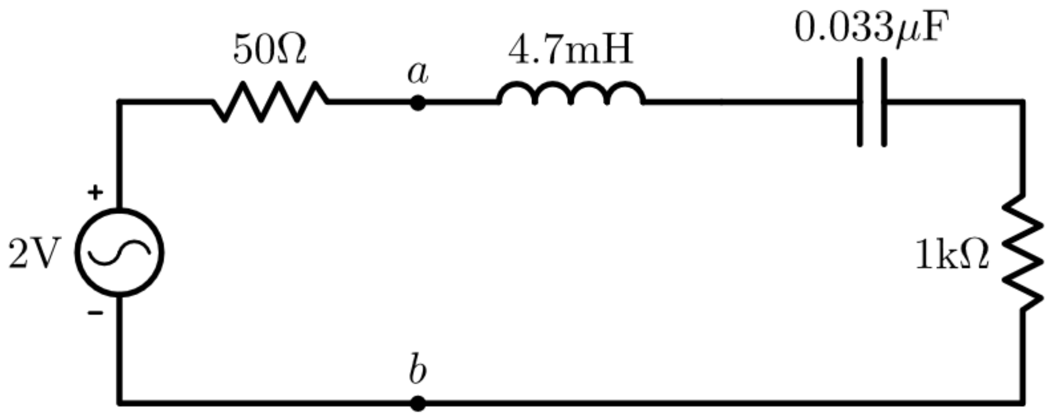
1. Write a Python program that computes the complex impedances of both the parallel (Z_p) and series (Z_s) RLC circuits with a frequency range of $f \in [1kHz, 100kHz]$.
2. Simulate both RLC circuits in LTspice and generate complex impedance plots for $f \in [1kHz, 100kHz]$.
3. Create the series RLC circuit on a breadboard and measure the impedance of the circuit. Use python to graph the experimentally obtained impedance values.

Equipment

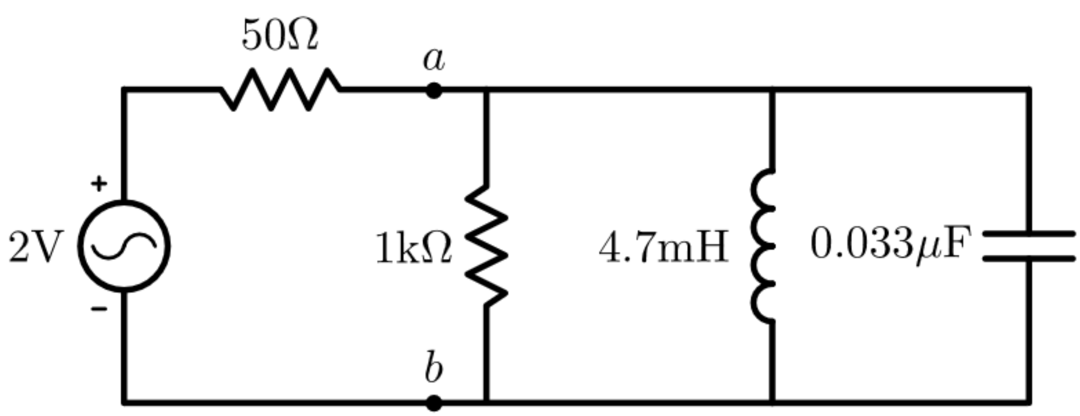
- Breadboard
- Multimeter
- Wave Generator
- Oscilloscope
- Resistor: $1k\Omega$
- Inductor: $4.7mH$
- Capacitor: $0.033\mu F$

The RLC Circuits

Series RLC Circuit



Parallel RLC Circuit



Python Code

[110]:
```
'''
Author(s): Dexter Ward, Hunter Van Horn
Date: 02/27/2025
This program will graph a Series and Parallel impedance as functions of
frequency
'''
```

[110]:
```
'\nAuthor(s): Dexter Ward, Hunter Van Horn  \nDate: 02/27/2025  \nThis program
will graph a Series and Parallel impedance as functions of \nfrequency\n'
```

Dependancies

[111]:
```
import numpy as np
import matplotlib.pyplot as plt
import cmath
```

Finding Resonant Frequency

[112]:
```
"""
Parameters
----------
Z: array_like
Array of impedance values
omega : array_like
Array of frequency values in radians / second
circuit_type : string
This can either be 'series ' or 'parallel '
"""

def find_omega_0 (z , omega, circuit_type) :
    if (circuit_type == "series"):
        omega_0 = omega[np.argmin(z)]
        f_0 = omega_0 / (2 * np.pi)
    else:
        omega_0 = omega[np.argmax(z)]
        f_0 = omega_0 / (2 * np.pi)
    return omega_0 , f_0
```

Impedance Calculations

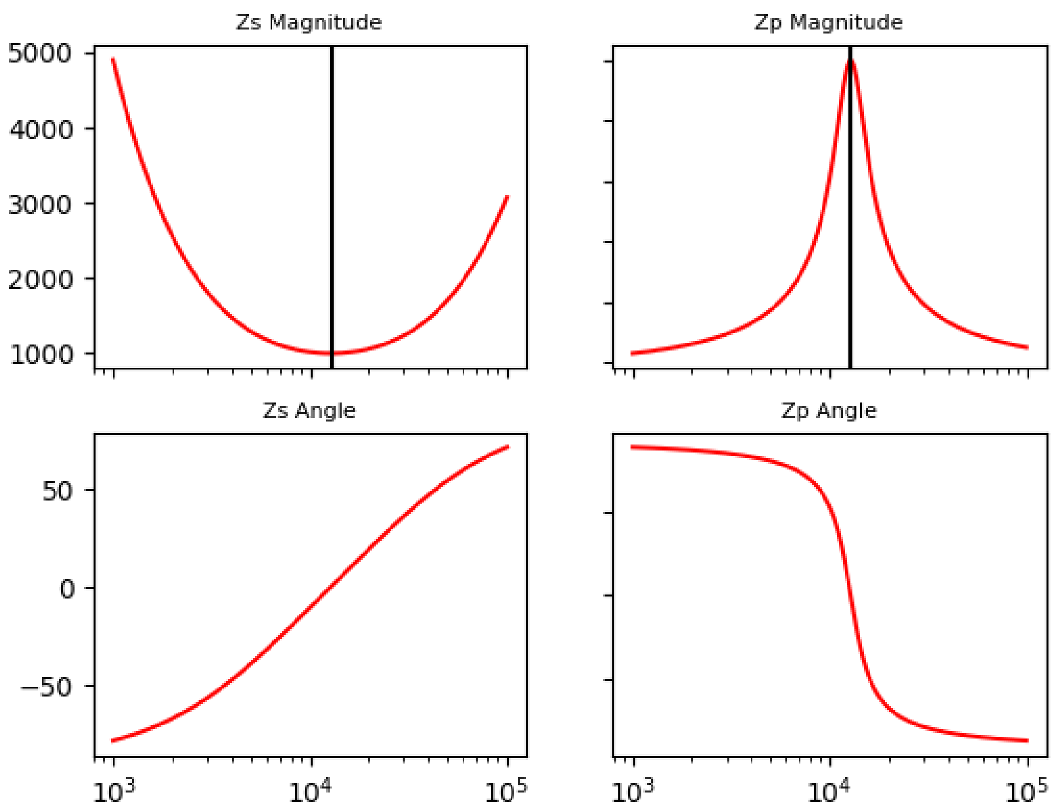
[113]:
```
"""
Parameters
----------
r: float, value of the resistor (Ohms)
l: float, value of the inductor (H)
c: float, value of the capacitor (F)
omega: float, value of the freq (rad)
"""
def imped_s(r, l, c, omega):
    zs = r + ((omega * l * 1j) + 1 / (omega * c * 1j))
    return zs

"""
Parameters
----------
r: float, value of the resistor (Ohms)
```

```
l: float, value of the inductor (H)
c: float, value of the capacitor (F)
omega: float, value of the freq (rad)
"""
def imped_p(r, l, c, omega):
    zp = 1 / ((1 / r) + (1 / (omega * l * 1j)) + (omega * c * 1j))
    return zp
```

Graphing

[114]:
```
r = 1000
l = 0.0047
c = 0.000000033
freq = np.arange(1000, 100001, 100)
omega = 2 * np.pi * freq

zsm = []
zsa = []
zpm = []
zpa = []

for i in omega:
    zs = imped_s(r, l, c, i)
    zp = imped_p(r, l, c, i)

    zsm.append(abs(zs))
    zsa.append(np.degrees(cmath.phase(zs)))

    zpm.append(abs(zp))
    zpa.append(np.degrees(cmath.phase(zp)))

fig, axs = plt.subplots(2, 2)


axs[0, 0].plot(freq, zsm, 'r')
axs[0, 0].axvline(find_omega_0(zsm, omega, "series")[1], color = 'k')
axs[0, 0].set_title('Zs Magnitude', fontsize=8)
axs[0, 0].semilogx()
axs[0, 0].set_xticklabels([])
axs[0, 1].plot(freq, zpm, 'r')
axs[0, 1].axvline(find_omega_0(zpm, omega, "parallel")[1], color = 'k')
axs[0, 1].set_title('Zp Magnitude', fontsize=8)
axs[0, 1].semilogx()
axs[0, 1].set_xticklabels([])
axs[1, 0].plot(freq, zsa, 'r')
axs[1, 0].set_title('Zs Angle', fontsize=8)
```

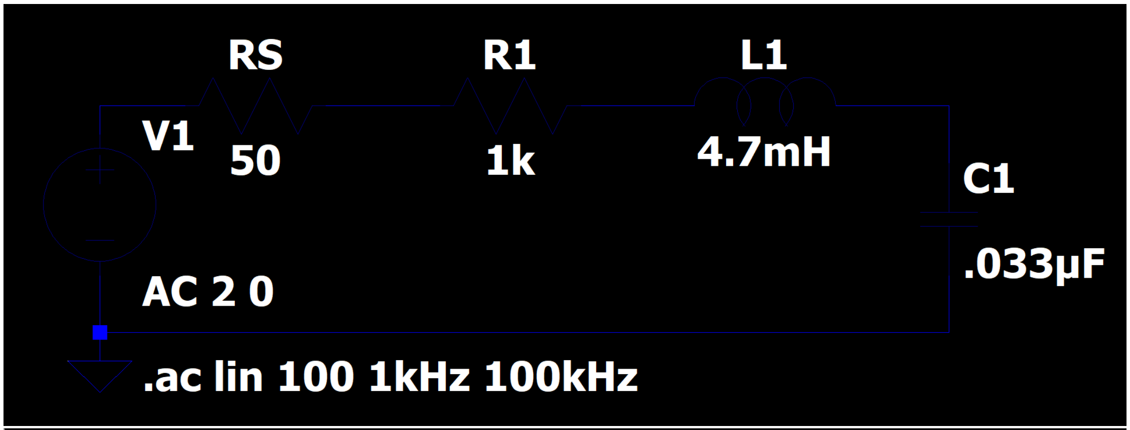
```
axs[1, 0].semilogx()
axs[1, 1].plot(freq, zpa, 'r')
axs[1, 1].set_title('Zp Angle', fontsize=8)
axs[1, 1].semilogx()

for ax in fig.get_axes():
    ax.label_outer()
```

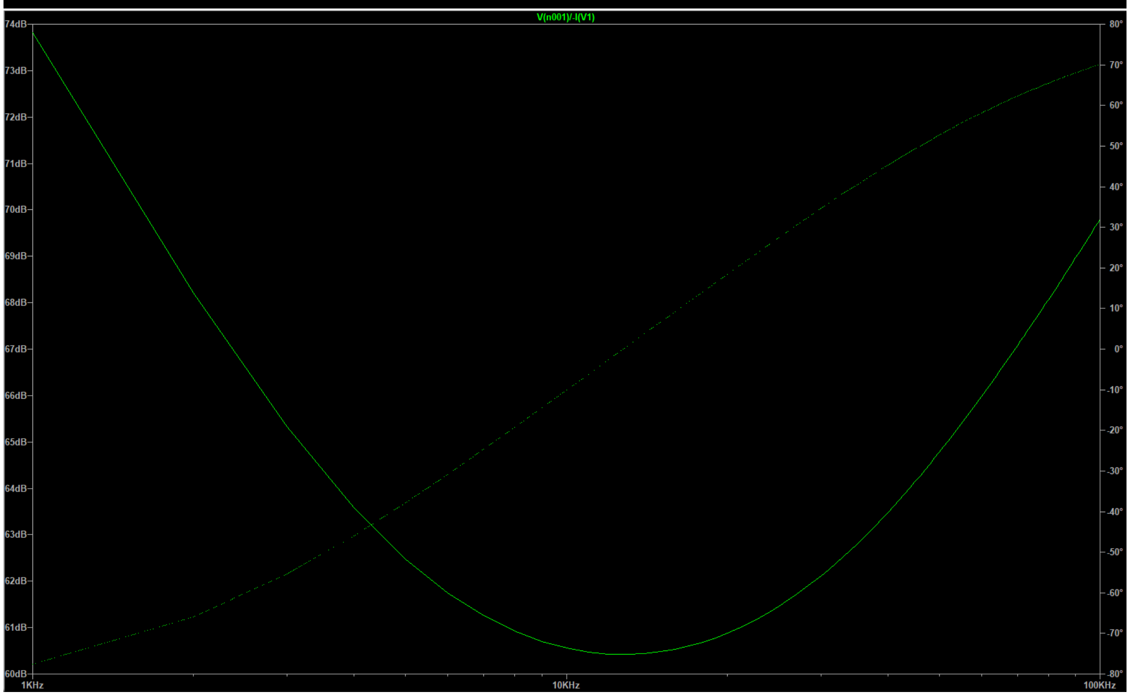


LTspice Simulation

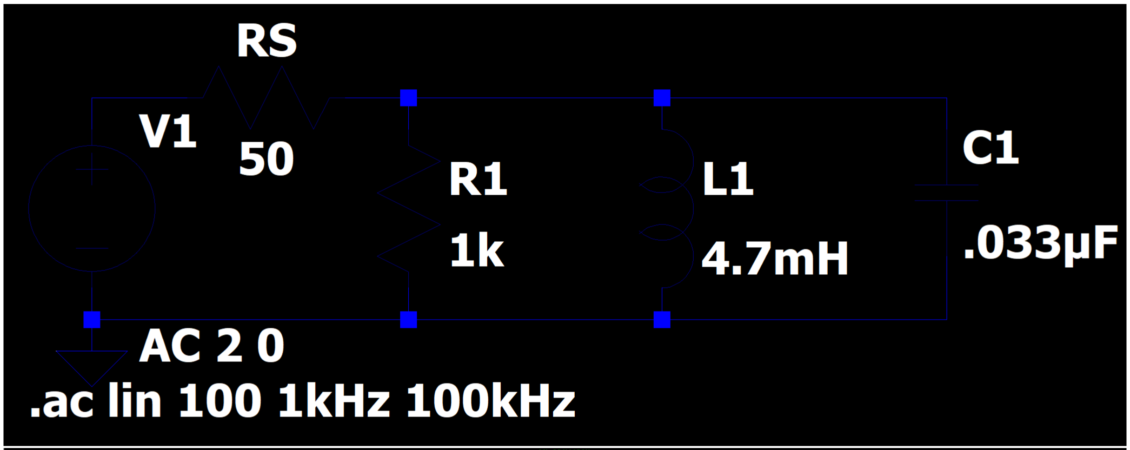
Using LTspice we simulated both a series and parallel RLC circuit. Using the trace feature of LTspice we were able to generate a graph of both the magnitude (represented by the solid line) and the angle (represented by the dashed line) vs. frequency of both circuits.

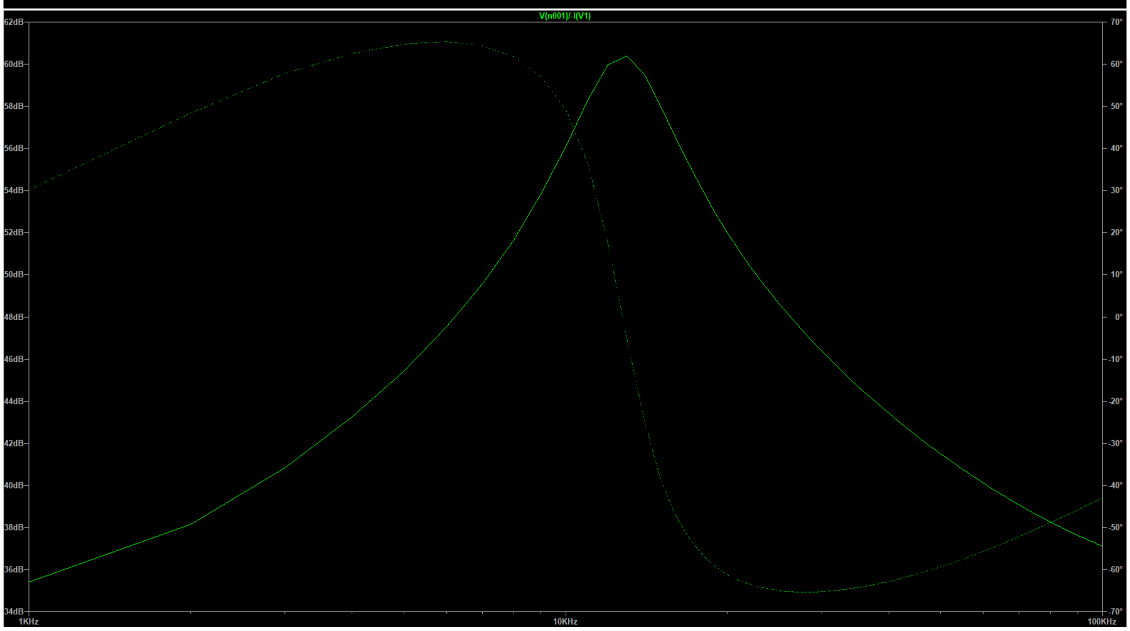
Series RLC Circuit





Parallel RLC Circuit





Experimental Circuit

Experimental Data

Number	Frequency	V_R	$\angle$
1	250Hz	120mV	-83.8°
2	500Hz	220mV	-81.4°
3	750Hz	320mV	-79.8°
4	1kHz	440mV	-76.5°
5	1.5kHz	600mV	-71.8°
6	2kHz	760mV	-66.8°
7	2.5kHz	920mV	-61.6°
8	3kHz	1.05V	-57.0°

Number	Frequency	V_R	$\angle$
9	3.5kHz	1.17V	$-52.5°$
10	4kHz	1.29V	$-48.2°$
11	4.5kHz	1.37V	$-44.1°$
12	5kHz	1.49V	$-40.3°$
13	7.5kHz	1.71V	$-25.4°$
14	10kHz	1.85V	$-12.4°$
15	12.5kHz	1.87V	$-3.9°$
16	14kHz	1.87V	0°
17	15kHz	1.87V	3.4°
18	20kHz	1.81V	15.2°
19	25kHz	1.73V	25.0°
20	30kHz	1.63V	33.0°
21	34kHz	1.55V	37.3°
22	38kHz	1.47V	41.8°
23	42kHz	1.39V	46.3°
24	46kHz	1.31V	49.9°
25	50kHz	1.25V	52.8°
26	54kHz	1.16V	54.7°
27	58kHz	1.13V	57.7°
28	60kHz	1.09V	57.3°
29	80kHz	860mV	67.1°
30	100kHz	720mV	73.6°

Graphing Experimental Data

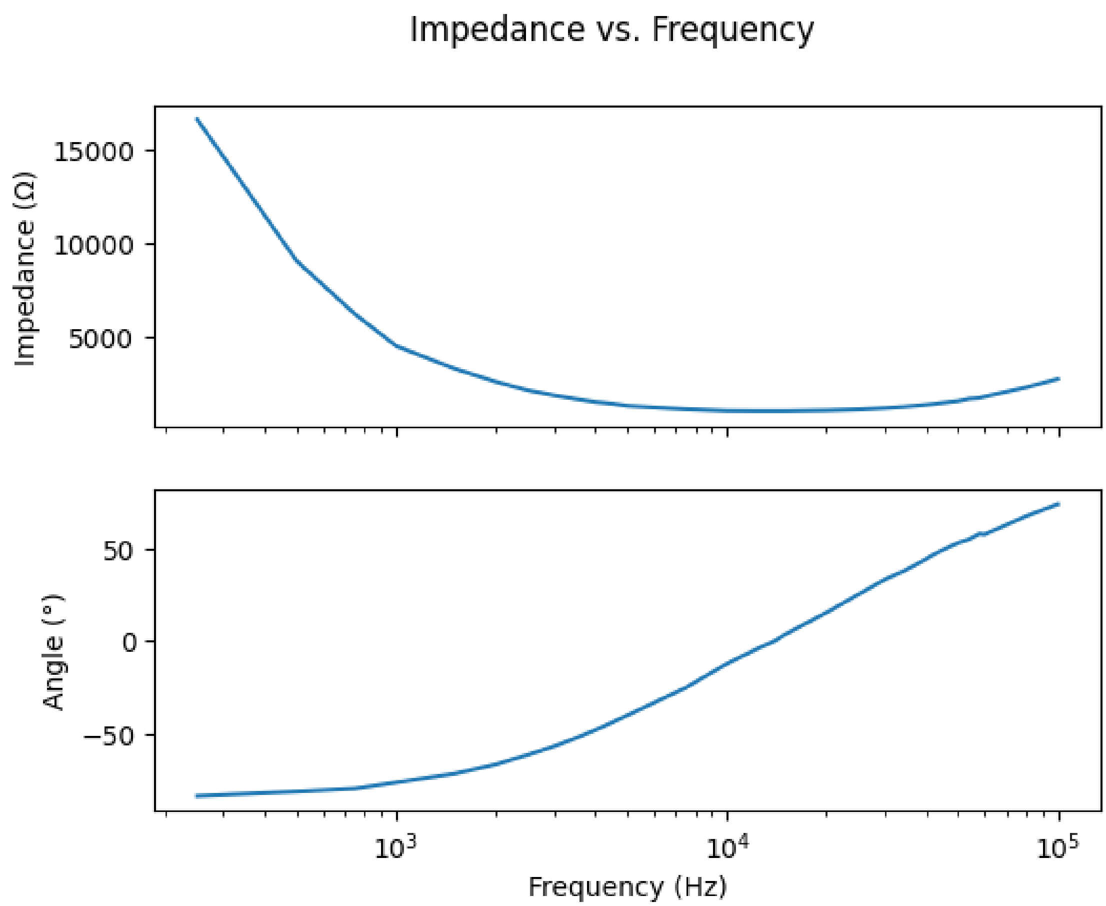
Dependancies

[115]:
```
import numpy as np
import matplotlib.pyplot as plt
```

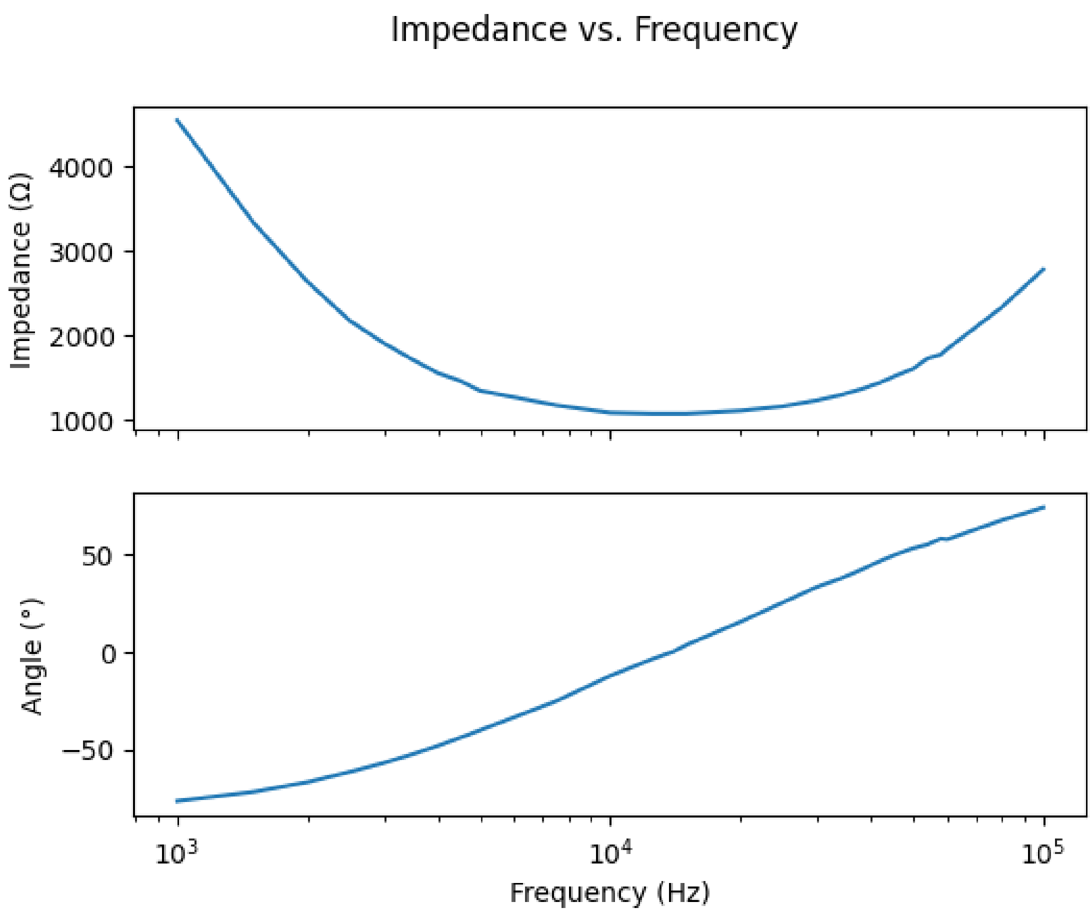
Graphing

[116]:
```
readFile = open('results.txt', 'r')
values = readFile.readlines()
readFile.close()

temp = []

for iter in values:
    temp.append(iter.replace(' ','').replace('\n','').split('|'))

allValues = np.array(temp, dtype='d')
frequency = allValues[:,1]*1000
current = (allValues[:,2]/1000)
```

```python
impedance = 2/current
angle = allValues[:,3]

fig1, axs1 = plt.subplots(2,1,sharex=True)
fig2, axs2 = plt.subplots(2,1,sharex=True)


axs1[0].semilogx(frequency, impedance)
axs1[1].semilogx(frequency, angle)
axs1[0].set_ylabel('Impedance (Ω)')
axs1[1].set_xlabel('Frequency (Hz)')
axs1[1].set_ylabel('Angle (°)')
fig1.suptitle('Impedance vs. Frequency')

axs2[0].semilogx(frequency[3:],impedance[3:])
axs2[1].semilogx(frequency[3:], angle[3:])
axs2[0].set_ylabel('Impedance (Ω)')
axs2[1].set_xlabel('Frequency (Hz)')
axs2[1].set_ylabel('Angle (°)')
fig2.suptitle('Impedance vs. Frequency')
plt.show()
```



Impedance vs. Frequency
Impedance (Ω)
15000
10000
5000
Angle (°)
50
0
−50
10^3
10^4
10^5
Frequency (Hz)

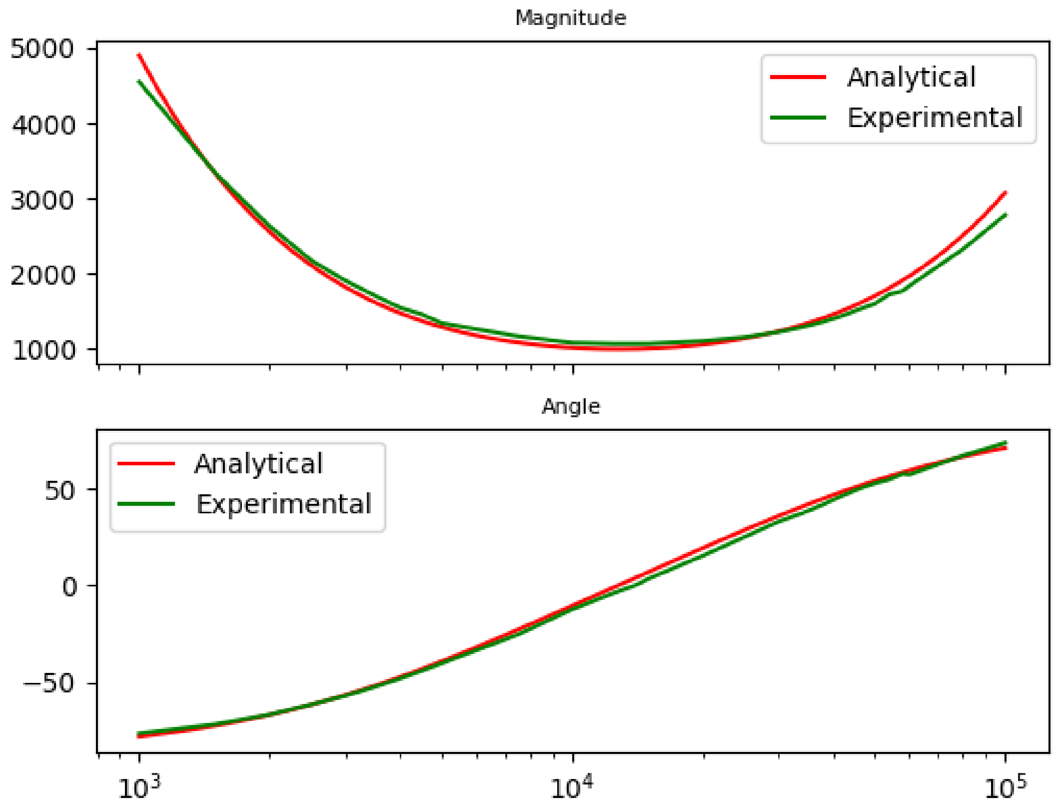


This portion of the program reads and parses the voltage and angle data from a file (results.txt). Using the data the current through the circuit is calculated. Once the current is known the impedance can be calculated using the input voltage. The graphs that are outputted are a measure of impedance vs. frequency (both magnitude and angle) the first set shows a frequency range of $f \in [0kHz, 100kHz]$ and the second set shows a frequency range of $f \in [1kHz, 100kHz]$.

Comparison

[117]:
```
r = 1000
l = 0.0047
c = 0.000000033
freq = np.arange(1000, 100001, 100)
omega = 2 * np.pi * freq

zsm = []
zsa = []
```

```python
zpm = []
zpa = []

for i in omega:
    zs = imped_s(r, l, c, i)
    zp = imped_p(r, l, c, i)

    zsm.append(abs(zs))
    zsa.append(np.degrees(cmath.phase(zs)))

    zpm.append(abs(zp))
    zpa.append(np.degrees(cmath.phase(zp)))

fig, axs = plt.subplots(2, 1, sharex=True)


axs[0].semilogx(freq, zsm, 'r', label='Analytical')
axs[0].set_title('Magnitude', fontsize=8)
axs[1].semilogx(freq, zsa, 'r', label='Analytical')
axs[1].set_title('Angle', fontsize=8)

readFile = open('results.txt', 'r')
values = readFile.readlines()
readFile.close()

temp = []

for iter in values:
    temp.append(iter.replace(' ','').replace('\n','').split('|'))

allValues = np.array(temp, dtype='d')
frequency = allValues[:,1]*1000
current = (allValues[:,2]/1000)
impedance = 2/current
angle = allValues[:,3]

axs[0].plot(frequency[3:],impedance[3:],'g',label='Experimental')
axs[1].plot(frequency[3:], angle[3:],'g',label='Experimental')
axs[0].legend()
axs[1].legend()
plt.show()

for ax in fig.get_axes():
    ax.label_outer()
```



Conclusion

In this lab my partner and I created a Python program that was able to calculate the impedance of both a series and parallel RLC circuit. Using the calculated values the program then graphed the magnitude and angle vs. frequency. We then recreated the circuit in LTspice and simulated the oscilloscope trace of the circuit that produced both the magnitude and angle of the impedance. Finally we built a series RLC circuit on a breadboard, with components in the following order: voltage source, inductor, capacitor, resistor. We then used an oscilloscope to measure the voltage and phase angle across the 1kΩ resistor. With this data in a file we created a python file that could import the lab data, parse it and use it to calculate the impedance. We then used the data to graph another magnitude and angle vs. frequency graph based on the experimental values. Finally, we combined both python programs to create a graph that compares the analytical and experimental data. As you can see from the final graphs the experimental data closely follows the analytical data.