LRC Circuit

KDSMIL001 PHY2004W

1 September 2020

Contents

1 Introduction

In this report we will investigate the LRC circuit in two different forms; the parallel resonant circuit and the series resonant circuit. More on what those are a bit later. For now we just need to know that an LRC circuit is comprised of an inductor (L), a resistor (R), and a capacitor(C), as well as some kind of applied voltage. In this experiment we will be using an alternating voltage as it results in some interesting behaviours. We will get into these behaviours more in later sections but for now all we need to know is that any kind of LRC circuit will have a resonant driving frequency at which the current (and thus the voltage) as well as the impedance in the circuit will be at an extremum. In order to see how each configuration will behave, let's investigate them analytically first.

2 LRC Circuit Theory

2.1 The Parallel Resonance LRC Circuit

The parallel resonance set-up for an LRC circuit is one in which the inductor and capacitor are wired in parallel with each other and then that inductor-capacitor module is wired in series with the resistor, as shown below.

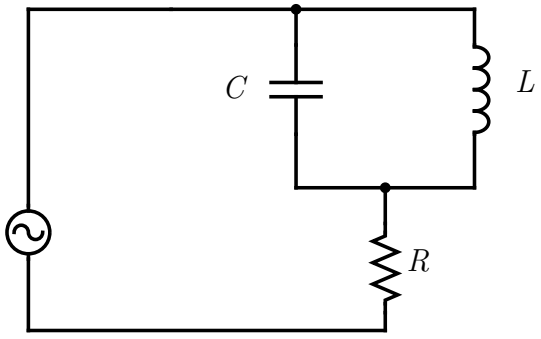


Figure 2.1: Parallel Resonance Circuit

In order to mathematically describe these circuits we use what we call a Transfer Function, in this case denoted $H(\omega)$, which is the magnitude of the ratio of output voltage across the resistor V_{out} over the applied voltage V_{in}. Finding these voltages in terms of inductance and so on leads us to

$$H(\omega) = \frac{R(1-\omega^2 LC)}{\sqrt{R^2(1-\omega^2 LC)^2 + \omega^2 L^2}} \tag{2.1}$$

which, when differentiated, has a minimum at $\omega_0 = \frac{1}{\sqrt{LC}}$. Note that this ω is $2\pi f$ where f is the driving frequency. If we plot this function with some values for L, C, R, we get something like Figure 2.2.

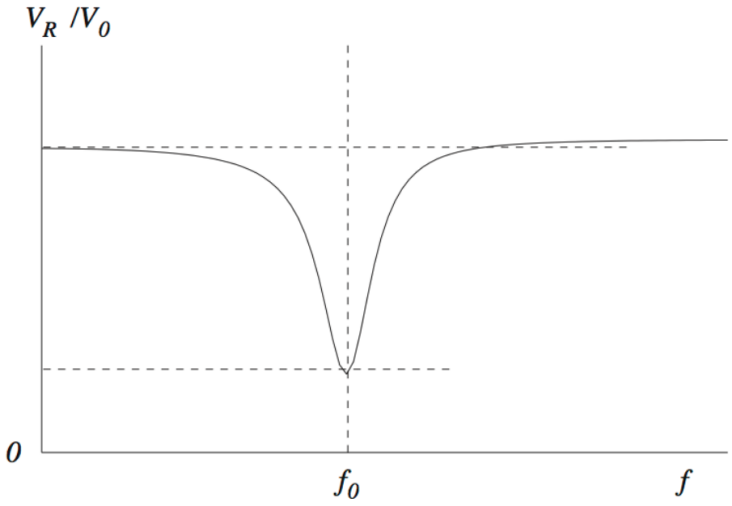


Figure 2.2: Parallel Resonance Curve

We'll discuss this more in the following section.

2.2 The Series Resonance LRC Circuit

In this set-up we have all three components, the inductor, capacitor, and resistor, in series, as below.

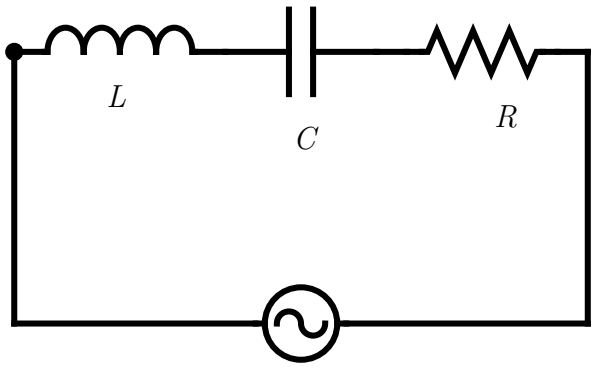


Figure 2.3: Series Resonance Circuit

As you might imagine, the transfer function for this configuration is slightly different to Equation 2.1. In fact we will derive it now.

We begin with the definition of the transfer function:

$$H(\omega) = \frac{V_{out}}{V_{in}}$$

where V_{in} is the voltage supplied by our AC source and V_{out} is the voltage drop across the resistor, which we know will be

$$V_{out} = iR$$

but as this is a series circuit we know that the current in all the components will always be the same value

$$i = \frac{V_{in}}{|Z(\omega)|}$$

where $Z(\omega)$ is the impedance of this circuit. Again this is a series circuit so the total impedance is merely the sum of the impedance of the respective components.

$$\begin{aligned} Z(\omega) &= Z_L + Z_C + Z_R \\ &= (0 + jL\omega) + (0 - j\frac{1}{C\omega}) + (R + 0j) \\ &= R + j(L\omega - \frac{1}{C\omega}) \end{aligned}$$

where we're using j as the imaginary unit. Current is not complex, which is why we have $|Z|$ previously. We can calculate this value

$$\begin{aligned} |Z(\omega)| &= \sqrt{Z \cdot Z*} \\ &= \sqrt{(R + j(L\omega - \frac{1}{C\omega}))(R - j(L\omega - \frac{1}{C\omega}))} \\ &= \sqrt{R^2 + (L\omega - \frac{1}{C\omega})^2} \\ &= \sqrt{R^2 + L^2\omega^2 - 2\frac{L}{C} + \frac{1}{C^2\omega^2}} \\ &= \frac{1}{C\omega}\sqrt{C^2\omega^2R^2 + L^2C^2\omega^4 - 2LC\omega^2 + 1} \\ &= \frac{1}{C\omega}\sqrt{(C\omega R)^2 + (1 - LC\omega^2)^2} \end{aligned}$$

Which means that we have

$$\begin{aligned} i &= \frac{V_{in}}{\frac{1}{C\omega}\sqrt{(C\omega R)^2 + (1 - LC\omega^2)^2}} \\ &= \frac{V_{in}C\omega}{\sqrt{(C\omega R)^2 + (1 - LC\omega^2)^2}} \\ \implies V_{out} &= \frac{RV_{in}C\omega}{\sqrt{(C\omega R)^2 + (1 - LC\omega^2)^2}} \\ \implies H(\omega) &= \frac{RV_{in}C\omega}{V_{in}\sqrt{(C\omega R)^2 + (1 - LC\omega^2)^2}} \end{aligned}$$

So we have our transfer function for the series resonance circuit:

$$H(\omega) = \frac{RC\omega}{\sqrt{(C\omega R)^2 + (1 - LC\omega^2)^2}} \tag{2.2}$$

This equation has the same extremum as the parallel circuit, except it has a maximum at $\omega_0 = \frac{1}{\sqrt{LC}}$ rather than a minimum. Plotting this function we did in Figure 2.2 we get

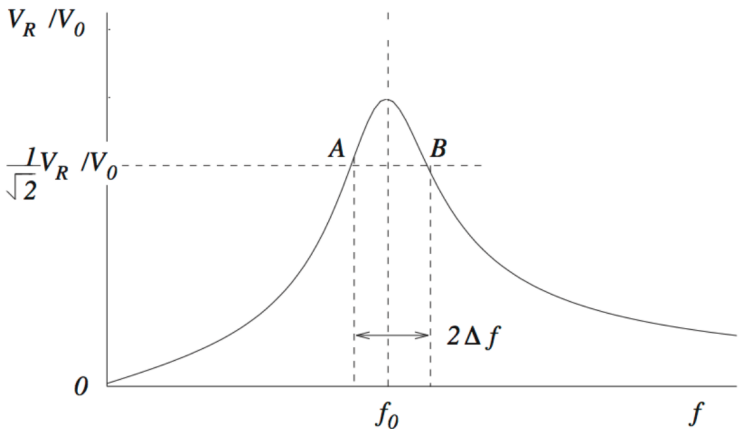


Figure 2.4: Series Resonance Curve

These plots have f along the horizontal axis as that's the input frequency but that is no problem as $f = \frac{\omega}{2\pi}$. From the values of $2\Delta f$ and f_0 we can define a parameter we call the quality factor

$$Q = \frac{f_0}{2\Delta f} = \frac{\omega_0 L}{R} \tag{2.3}$$

that describes the "sharpness" of this curve, the larger Q is, the sharper the curve. From Equation 2.3 we see that the resistance and inductance in the circuit has an impact on Q and thus on the shape of the resonance curve. Note that this quality factor and its associated usefulness is not unique to the series resonance circuit, it applies to the parallel circuit too and we will use this in later sections to analyse specific circuits.

3 Experiment

In the experimental section of this report we will be investigating the behaviour of both the parallel and series resonance circuits by way of setting them up on a breadboard to see how they behave in reality, as well as using a simulation of each.

3.1 Apparatus

We used the following items, all with standard measurement uncertainty of 2%:

- 1 1 kΩ resistor (actual measured value 984.4 Ω).
- 2 100 Ω resistors (actual measured value 98.0 and 98.6 Ω).
- 1 96.51 nF capacitor (actual measured value 93.94 nF).
- 1 70 mH inductor (actual measured value 73.62 mH).
- 1 myDAQ.
- 1 screwdriver.
- 1 function generator.
- 1 digital oscilloscope.
- 1 breadboard.
- Wire (with negligible resistance).

Our parallel circuit was set up with the 1 kΩ resistor and the series circuit was set up with the two 100 Ω resistors in series to achieve a total resistance of 200 Ω. They were set up in the same configuration as in Figure 2.1 and Figure 2.3.

3.2 Method and Results

To collect our data (thanks Prof Blumenthal) we started off using the function generator to apply a sinusoidal alternating voltage of 1 Volt peak-to-peak across the parallel and series circuits with the oscilloscope measuring the voltage across the resistor. By adjusting the frequency of the function generator we were able to find the point at which the voltage across the resistor was minimised or maximised, depending on the type of circuit. The oscilloscope displays for each circuit are pictured in Figure 3.1 and Figure 3.2

Starting with Figure 3.1, we counted one wavelength to be 11 small divisions, with one division on either side as the bounds of the pdf. One division is 0.5×10^{-4} s and this is an analogue reading, so our uncertainty is $u = \frac{a}{2\sqrt{6}}$. That gives us a wavelength of $\lambda = (5.5000 \pm 0.2641) \times 10^{-4}$ s. As we know, $f = \frac{1}{\lambda}$, and the uncertainty is given by $u(f) = \frac{u(\lambda)}{\lambda^2}$, which was derived from the multiplicative propagation equation. This gives us a resonant frequency of $f_0 = 1818.18 \pm 67.48$ Hz.

For the series circuit, Figure 3.2, we counted 12 small divisions with the same upper and lower bound. Doing the same calculations and uncertainty propagations we

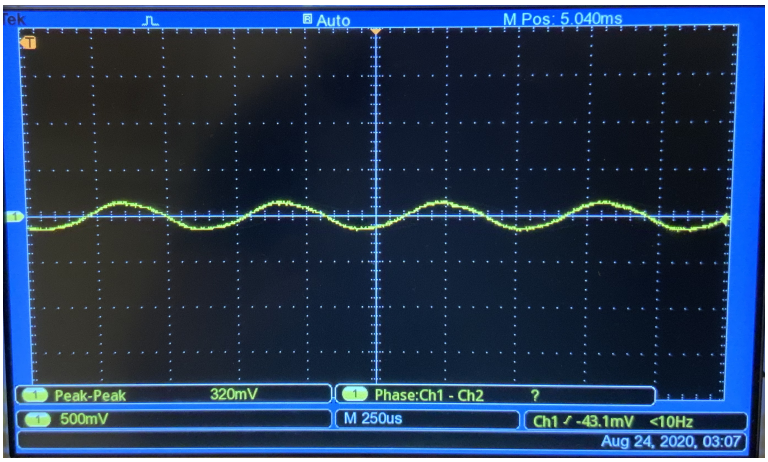


Figure 3.1: Parallel Resonance Circuit at resonance

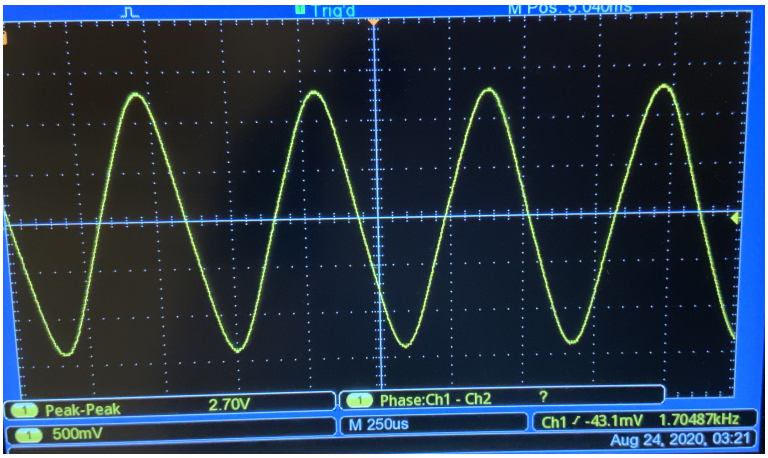


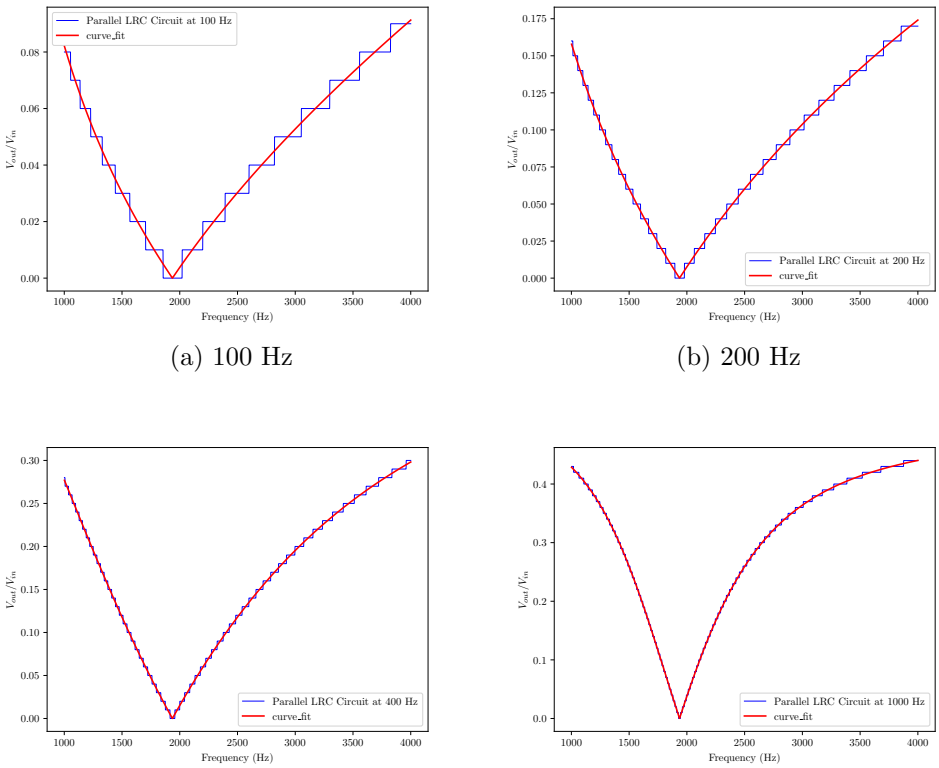
Figure 3.2: Series Resonance Circuit at resonance

found the resonant frequency to be $f_0 = 1666.67 \pm 73.37$ Hz. The expected value for both circuits can be found from $\omega_0 = 2\pi f_0 = \frac{1}{\sqrt{LC}}$. If we use the measured values for L and C with uncertainties of 2% we get $f_0 = 1913.802 \pm 27.064$ Hz, where the uncertainty is given by

$$u(f) = \sqrt{\frac{L^2 u(C)^2 + C^2 u(L)^2}{4C^3 L^3}}$$

Next we used the `LCsim.exe` program to simulate the parallel and series circuits with a range of different resistance values to investigate how the value of R affects the value of Q. We simulated two circuits with the same inductance and capacitance values as the real circuits we used (70 mH and 96.51 nF) and ran each circuit separately with 100, 200, 400, and 1000 Ωs of resistance, 1 Volt peak-to-peak, and from 1000 to 4000 Hz with 2000 data points. The program can export as text so

we copied the outputs into .txt files and did our usual plotting procedure. We then used `scipy.optimize.curve_fit` to fit the data to the relevant transfer function. The results are in Figure 2.1 and Figure 2.3.



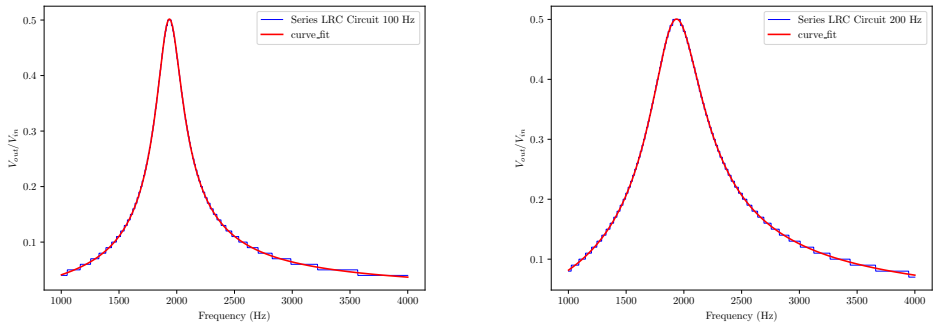
(a) 100 Hz

(b) 200 Hz

(c) 400 Hz

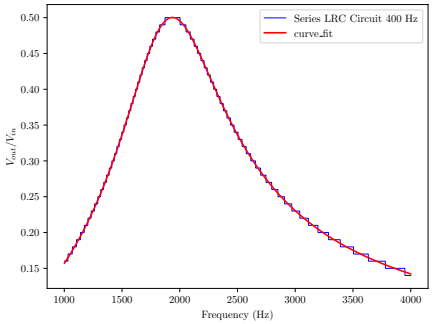
(d) 1000 Hz

Figure 3.3: Parallel Circuit Simulation

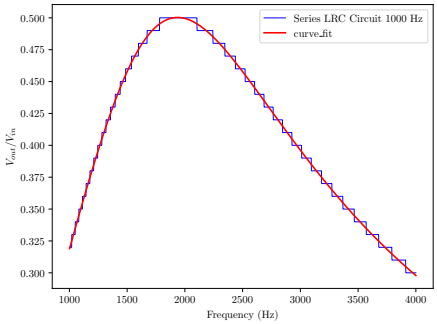


(a) 100 Hz

(b) 200 Hz



(c) 400 Hz



(d) 1000 Hz

Figure 3.4: Series Circuit Simulation

We'd like to mention here that the `LCsim.exe` program seems to have a maximum resolution of 0.01 Volts, which is the reason that the data above looks choppy. Despite this it seems that the `curve_fit` function can still approximate the function well because the steps seem to even each other out.

And finally, in order to investigate the transfer function for these two circuits, we used the myDAQ to apply a range of frequencies of alternating voltage with 1 Volt peak-to-peak to the circuits described in subsection 3.1. The myDAQ also recorded the voltage across the resistor. We could then plot that data and use `curve_fit` to fit it to the relevant transfer function. The results are in Figure 3.5, Figure 3.6, and Table 1. The Q values were calculated from Equation 2.3.

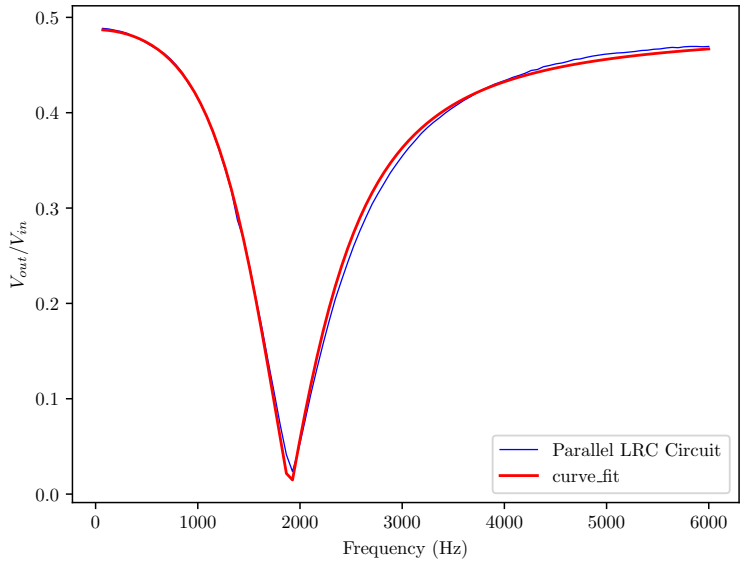


Figure 3.5: Parallel Frequency Sweep

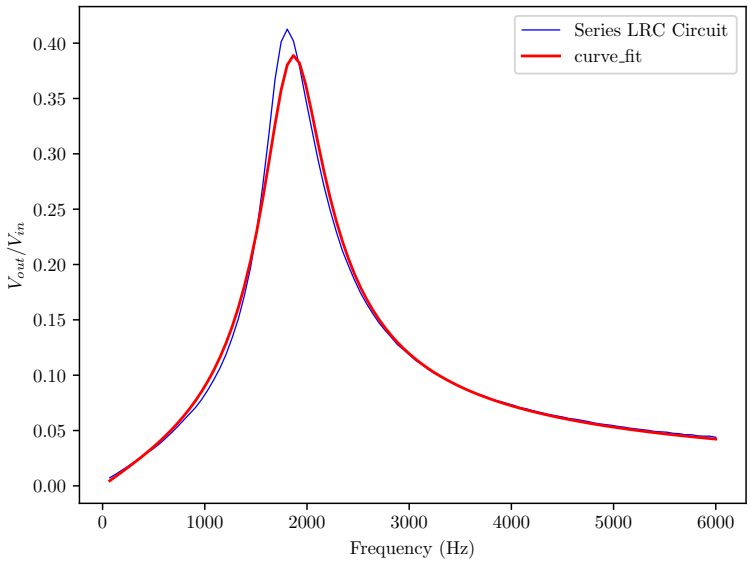


Figure 3.6: Series Frequency Sweep

Parallel		Series	
f_0:	1926.80 ± 38.54 Hz	f_0:	1866.90 ± 37.34 Hz
L:	$(65.78 \pm 3.33) \times 10^{-3}$ H	L:	$(73.31 \pm 3.25) \times 10^{-3}$ H
R:	$931.98 \pm 49.22\ \Omega$	R:	$272.95 \pm 12.66\ \Omega$
C:	$(106.41 \pm 5.41) \times 10^{-9}$ F	C:	$(98.954 \pm 4.097) \times 10^{-9}$ F
Scale Factor:	0.48716 ± 0.00059	Scale Factor:	0.3888 ± 0.0014
Q:	1.17 ± 0.082	Q:	3.15 ± 0.18

Table 1: Resonant Frequency and Quality Factor for Parallel and Series Resonant Circuits

On the uncertainties for these values, we used the 2% accuracy rating for the equipment used in order to get the uncertainty of f_0, the resonant frequency. The uncertainties of L, R, C, and the Scale Factor we used Monte Carlo methods, particularly the Jackknife method to estimate the uncertainties as these values came from the optimal fitting parameters that `curve_fit` gave us. Q was determined from Equation 2.3 and the uncertainty was determined from propagating the Jackknife uncertainties through that equation using the multiplicative propagation equation:

$$z = xy \implies u(z) = |xy| \sqrt{\left(\frac{u(x)}{x}\right)^2 + \left(\frac{u(y)}{y}\right)^2}$$

All of the code that did this analysis is in Appendix 1 and Appendix 2, being modified slightly to loop through different files.

4 Discussion and Recommendations

4.1 Resonance Frequency Search

The two resonant frequencies found in subsection 3.2 from examining Figure 3.1 and Figure 3.2 were

$$\begin{aligned} \text{Parallel:} \quad & f_0 = 1818.18 \pm 67.48 \text{ Hz} \\ \text{Series:} \quad & f_0 = 1666.67 \pm 73.37 \text{ Hz} \\ \text{Expected:} \quad & f_0 = 1913.802 \pm 27.064 \text{ Hz} \end{aligned}$$

These values do not agree. The reason for this discrepancy is likely due to the lack of control we have when finding the resonant frequency as the effective resolution on the oscilloscope when modulating the driving frequency is quite low, so it's hard to find the exact resonant frequency. To improve this experiment we recommend using the methods we used when analysing the myDAQ data as it's much more definite than just guessing while looking at an oscilloscope.

4.2 Circuit Simulation

To investigate the relationship between resistance and the quality factor, we look at Figure 3.5 and Figure 3.6. Looking at the parallel circuit first, the function clearly becomes more sharp the higher the resistance is, so $Q \propto R$ for the parallel resonance circuit. The opposite is true for series; as the resistance increases the function becomes less sharp, so we have $Q \propto \frac{1}{R}$ for the series resonance circuit.

5 Appendix

Appendix 1: Parallel Frequency Sweep

```
from matplotlib import pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
from scipy.signal import find_peaks
from numpy import cos, pi, sin, sqrt, exp, random, power
import matplotlib
matplotlib.use('pgf')
matplotlib.rcParams.update({
    'pgf.texsystem': 'pdflatex',
    'font.family': 'serif',
    'text.usetex': True,
    'pgf.rcfonts': False,
})
# Extracting all the relevant data, sorting it into our data array,
   for use later
file = open('Parallel_Freq_Sweep.txt', 'r')
header = file.readline()
lines = file.readlines()
N = len(lines)
i=0
# data[0] is the frequency
# data[1] is the voltage
data = np.zeros((2,N))
# Reading the file, getting the data into the data array
for line in lines:
    line = line.strip()
    columns = line.split()
    data[0][i] = float(columns[0])
    data[1][i] = float(columns[1])
    i += 1
file.close()

p0=[73.62e-3,984.4,93.94e-9,0.5]
pNames=['L','R','C','scaleFactor']
u=data[1]*0.02
tmodel = np.linspace(data[0][0], data[0][-1], N, endpoint=True)
def transfer(omega,L,R,C,scaleFactor):
    return scaleFactor*abs(R*(1-power(2*pi*omega,2)*L*C))/(sqrt((
       power(R,2)*(power((1-power(2*pi*omega,2)*L*C),2)))+(power(2*pi
       *omega,2)*power(L,2))))
# Removes random values from the data sets
jackknifeData = np.zeros((4, N, N-1))
for c in range(N):
    r = random.randint(0, N)
    jackknifeData[0, c] = np.delete(data[0], r)
    jackknifeData[1, c] = np.delete(data[1], r)
```

```
    jackknifeData[2, c] = np.delete(u, r)
    jackknifeData[3, c] = np.delete(tmodel, r)
# Fitting to the jackknifed datasets
jackknifeFits = np.zeros((N, N-1))
popts = []
for k in range(N):
    popt, pcov = curve_fit(transfer, jackknifeData[0, k],
       jackknifeData[1, k], p0, sigma=jackknifeData[2, k],
       absolute_sigma=True)
    jackknifeFits[k] = transfer(jackknifeData[3, k], *popt)
    popts.append(popt)
# Isolates arrays of each optimal fitting parameter
poptNp = np.zeros((4, N))
for d, ds in enumerate(popts):
    poptNp[0, d] = ds[0]
    poptNp[1, d] = ds[1]
    poptNp[2, d] = ds[2]
    poptNp[3, d] = ds[3]
# Calculates means and standard uncertainties
pOptimals = np.zeros((2, 4))
for j, js in enumerate(poptNp):
    mean = np.mean(js)
    pOptimals[0][j] = mean
    sumI = 0
    for i in js:
        sumI += (i-mean)**2
    pOptimals[1][j] = sqrt(float(((N-1)/N)*sumI))/sqrt(N-1)
yfit=transfer(tmodel,*pOptimals[0])
# Printing optimal fit parameters
for p in range(len(p0)):
    print(pNames[p],'=',pOptimals[0][p],'+/-',pOptimals[1][p])
# Determining Q=resFreq*L/R
resIndex=find_peaks(-yfit)[0][0]
resFreq=tmodel[resIndex]
resAngularFreq=2*pi*resFreq
resAngularFreqL=resAngularFreq*pOptimals[0][0]
Q=pOptimals[0][1]/resAngularFreqL
# Uncertainty calculations
resAngularFreqUn=0.02*resAngularFreq
resAngularFreqLUn=resAngularFreqL*sqrt((resAngularFreqUn/
   resAngularFreq)**2 + (pOptimals[1][0]/pOptimals[0][0])**2)
QUn=Q*sqrt((resAngularFreqLUn/resAngularFreqL)**2 + (pOptimals[1][1]/
   pOptimals[0][1])**2)

print('The Quality factor is',Q,'+/-',QUn)
print('The Resonant Frequency is',resFreq,'+/-',resFreq*0.02)
print('The Resonant Angular Frequency is',resAngularFreq,'+/-',
   resAngularFreqUn)
# Plotting the data
plt.figure()
```

```
plt.plot(data[0],data[1],color='b',linewidth=0.7,label='Parallel LRC Circuit')
plt.plot(tmodel, yfit,c='r',label='curve\_fit')
plt.xlabel('Frequency (Hz)')
plt.ylabel('$V_{out}/V_{in}$')
plt.legend()
# plt.show()
# plt.savefig('Parallel_Freq_Sweep.pgf')
```

Appendix 2: Series Frequency Sweep

```
from matplotlib import pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
from scipy.signal import find_peaks
from numpy import cos, pi, sin, sqrt, exp, random, power
import matplotlib
matplotlib.use('pgf')
matplotlib.rcParams.update({
    'pgf.texsystem': 'pdflatex',
    'font.family': 'serif',
    'text.usetex': True,
    'pgf.rcfonts': False,
})
# Extracting all the relevant data, sorting it into our data array, for use later
file = open('Series_Freq_Sweep.txt', 'r')
header = file.readline()
lines = file.readlines()
N = len(lines)
i=0
# data[0] is the frequency
# data[1] is the "voltage"
data = np.zeros((2,N))
# Reading the file, getting the data into the data array
for line in lines:
    line = line.strip()
    columns = line.split()
    data[0][i] = float(columns[0])
    data[1][i] = float(columns[1])
    i += 1
file.close()
# Using the jackknife method to calculate uncertainties on curve_fir determined optimal parameters
p0=[73.62e-3,196.6,93.94e-9,0.35]
pNames=['L','R','C','scaleFactor']
u=data[1]*0.02
tmodel = np.linspace(data[0][0], data[0][-1], N, endpoint=True)
def transfer(omega,L,R,C,scaleFactor):
```

```
    # return scaleFactor*(R*2*pi*omega*C)/sqrt((power(1-C*L*power(2*
        pi*omega,2),2))+power(C*2*pi*omega*R,2))
    return scaleFactor*(R*2*pi*omega*C)/sqrt(((1-C*L*power(2*pi*omega
        ,2))**2) + ((C*2*pi*omega*R)**2))
# Removes random values from the data sets
jackknifeData = np.zeros((4, N, N-1))
for c in range(N):
    r = random.randint(0, N)
    jackknifeData[0, c] = np.delete(data[0], r)
    jackknifeData[1, c] = np.delete(data[1], r)
    jackknifeData[2, c] = np.delete(u, r)
    jackknifeData[3, c] = np.delete(tmodel, r)
# Fitting to the jackknifed datasets
jackknifeFits = np.zeros((N, N-1))
popts = []
for k in range(N):
    popt, pcov = curve_fit(transfer, jackknifeData[0, k],
        jackknifeData[1, k], p0, sigma=jackknifeData[2, k],
        absolute_sigma=True)
    jackknifeFits[k] = transfer(jackknifeData[3, k], *popt)
    popts.append(popt)
# Isolating arrays of each optimal fitting parameter
poptNp = np.zeros((4, N))
for d, ds in enumerate(popts):
    poptNp[0, d] = ds[0]
    poptNp[1, d] = ds[1]
    poptNp[2, d] = ds[2]
    poptNp[3, d] = ds[3]
# Calculating means and standard uncertainties
pOptimals = np.zeros((2, 4))
for j, js in enumerate(poptNp):
    mean = np.mean(js)
    pOptimals[0][j] = mean
    sumI = 0
    for i in js:
        sumI += (i-mean)**2
    pOptimals[1][j] = sqrt(float(((N-1)/N)*sumI))/sqrt(N-1)
# Plotting the function with the optimal parameters
yfit=transfer(tmodel,*pOptimals[0])
# Printing fit parameters
for p in range(len(p0)):
    print(pNames[p],'=',pOptimals[0][p],'+/-',pOptimals[1][p])
# Determining Q=resFreq*L/R
resIndex=find_peaks(yfit)[0][0]
resFreq=tmodel[resIndex]
resAngularFreq=2*pi*resFreq
resAngularFreqL=resAngularFreq*pOptimals[0][0]
Q=resAngularFreqL/pOptimals[0][1]
# Uncertainty calculations
resAngularFreqUn=0.02*resAngularFreq
```

```
83 resAngularFreqLUn=resAngularFreqL*sqrt((resAngularFreqUn/
       resAngularFreq)**2 + (pOptimals[1][0]/pOptimals[0][0])**2)
84 QUn=Q*sqrt((resAngularFreqLUn/resAngularFreqL)**2 + (pOptimals[1][1]/
       pOptimals[0][1])**2)
85 # Printing results
86 print('The Quality factor is',Q,'+/-',QUn)
87 print('The Resonant Frequency is',resFreq,'+/-',resFreq*0.02)
88 print('The Resonant Angular Frequency is',resAngularFreq,'+/-',
       resAngularFreqUn)
89 # Plotting the data
90 plt.figure()
91 plt.plot(data[0],data[1],color='b',linewidth=0.7,label='Series LRC
       Circuit')
92 plt.plot(tmodel, yfit,c='r',label='curve\_fit')
93 plt.xlabel('Frequency (Hz)')
94 plt.ylabel('$V_{out}/V_{in}$')
95 plt.legend()
96 # plt.show()
97 # plt.savefig('Series_Freq_Sweep.pgf')
```