DC And AC Currents Through LCR Circuits

PHY224 Fall 2021

Fredrik Dahl Bråten, Pankaj Patil

November 26, 2021

Abstract

The aim of this experiment is to study the behavior of different circuit elements like Resistor, Capacitor and Inductor in DC and AC circuits. We observed that capacitor opposes change in Voltage, while Inductor opposes in the current. As a result Capacitor in DC circuit is an open circuit while Inductor in DC circuit is a short circuit. In this report we present these behaviors with the help of Oscilloscope measurements.

1 Introduction

A. Background Theory

In RC circuit, the capacitance of the capacitor is given by $V_0 = \frac{Q_0}{C}$, and as we know $I_0 = \frac{V_R}{R} \implies V_R = V_0 e^{-\frac{t}{\tau}}$, , where $\tau = RC$. This is also the fitting equation for this case. Here total Voltage V_0 is independent variable, and V_R across the resistor is dependent variable. The capacitance (C) and the the resistance (R) are auxiliary parameters.

In LR circuit, we have $I_R(t) = I_0(1 - e^{\frac{t}{L/R}}) \implies V_R = V_0(1 - e^{\frac{t}{L/R}})$, where $I_0 = \frac{V_0}{R}$. Our fitting equation is the same, and again total Voltage V is our independent variable, and voltage across the Resistor V_R, is the dependent variable. L and R are auxiliary parameters.

In an LCR circuit, the total impedance Z is defined as $Z = \sqrt{(\omega L - \frac{1}{\omega C})^2 + R^2}$. Z can be found by measuring total Voltage V, compared to voltage $V_R = RI \implies Z = \frac{V}{V_R} R$.

It is clear from the fitting equation for RC circuit that we would observe exponential decay for the Voltage across the resistor, as can be seen in Figure 1. And in case of LR, the relationship is reversed, and we observe exponential increase in voltage across the Resistor as seen in Figure 3.

B. Materials and Methods

In this experiment we used various circuit elements namely Resistors, Capacitors, and Inductor, in the setups provides in the lab manual [1]. The measurements were made by connecting to the appropriate channels of Oscilloscope.

We followed the steps provided in the lab manual [1] for this experiment. We also made adjustments based on TA's suggestion to the circuits as the AC circuits in the manual were applicable to old Oscilloscopes.

2 Results

A. Transient Decay (RC)

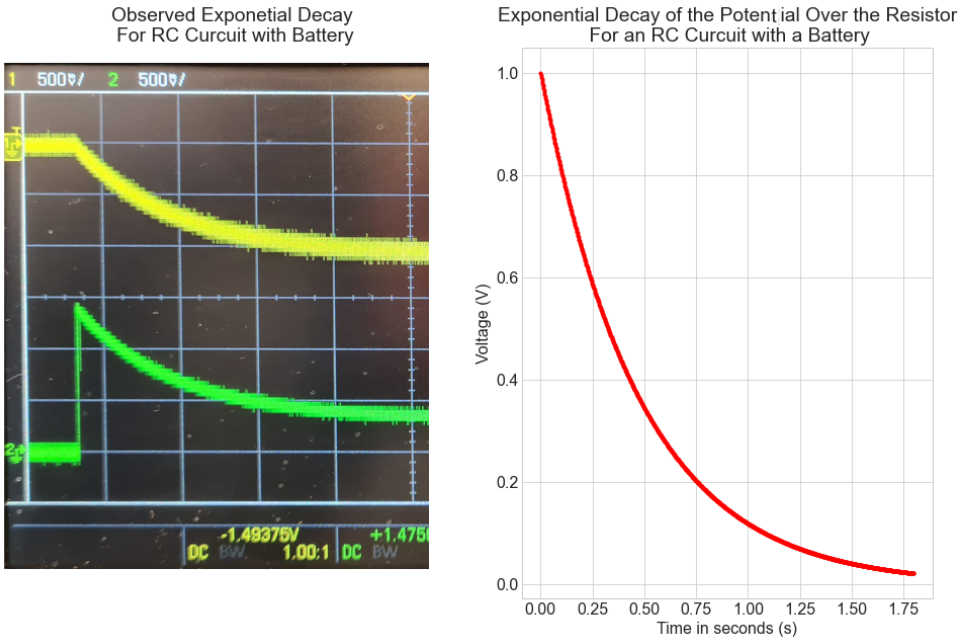


Figure 1: Transient Potential Decay in RC circuit (Measurement Vs. Simulation)

$$\tau\ (measured) = 0.4\ s, \quad RC = 0.47\ s$$

The observed or measured value of time constant matches closely with the theoretical value RC.

B. DC Square Wave

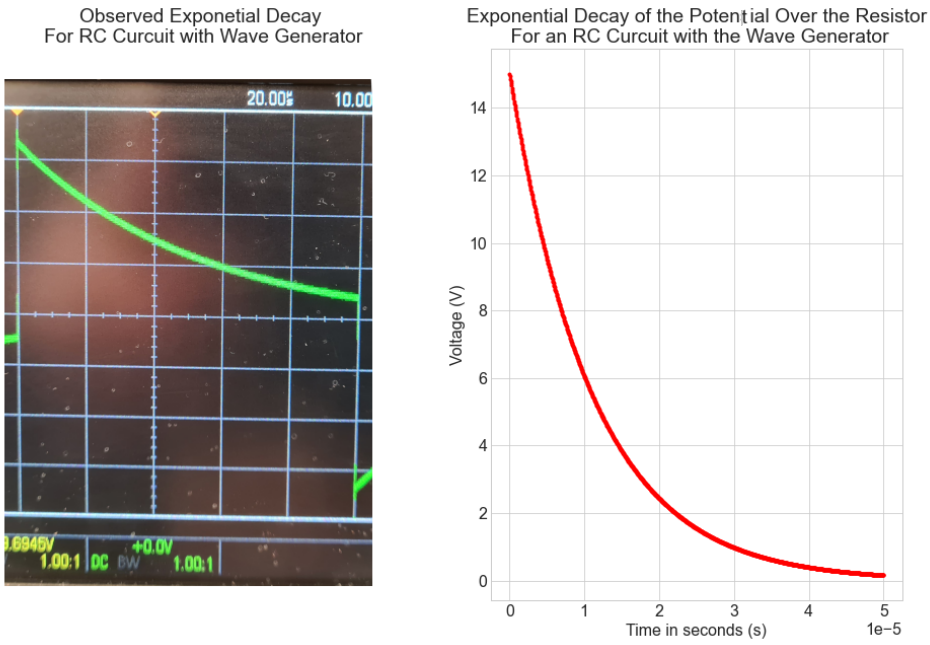


Figure 2: Exponential Potential Decay in RC circuit (Measurement Vs. Simulation)

$$\tau\ (measured) = 8 \times 10^{-5}\ s$$
$$RC = 1.1 \times 10^{-5}\ s$$

The observed or measured value of time constant matches with the theoretical value RC in order of magnitude.

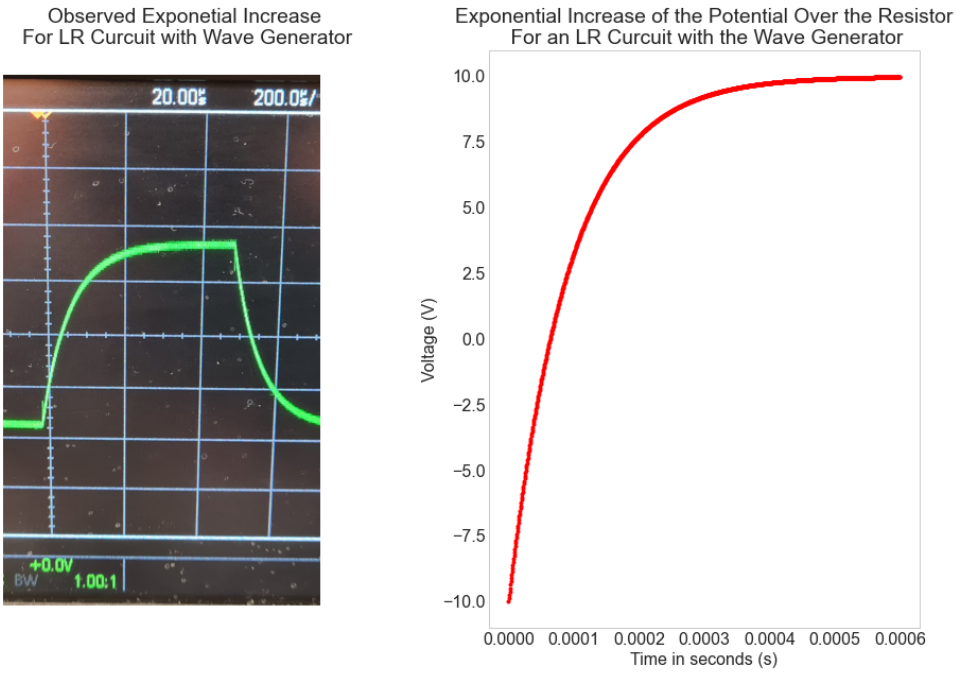


Figure 3: Exponential Potential Increase in LR circuit (Measurement Vs. Simulation)

$$\tau\ (measured) = 5.0 \times 10^{-5}\ s$$
$$L/R = 9.2 \times 10^{-5}\ s$$

The observed or measured value of time constant matches with the theoretical value L/R in order of magnitude.

C. AC Sine Wave

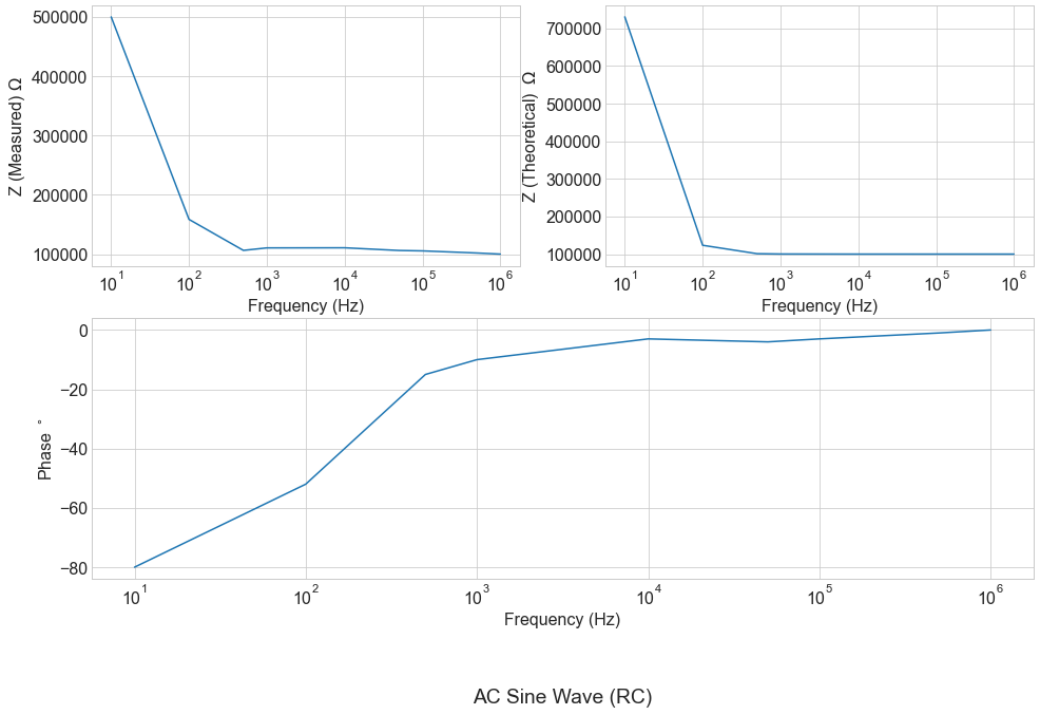


Figure 4: AC Sine Wave (RC): Z vs Frequency and Phase vs Frequency Plots

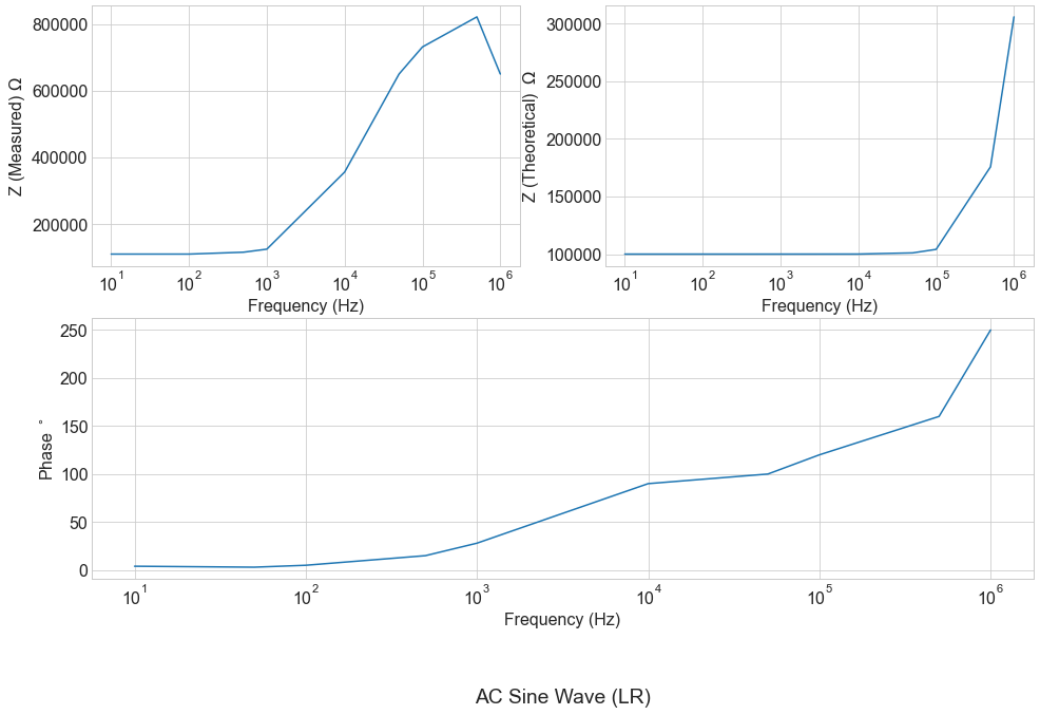


Figure 5: AC Sine Wave (LR): Z vs Frequency and Phase vs Frequency Plots

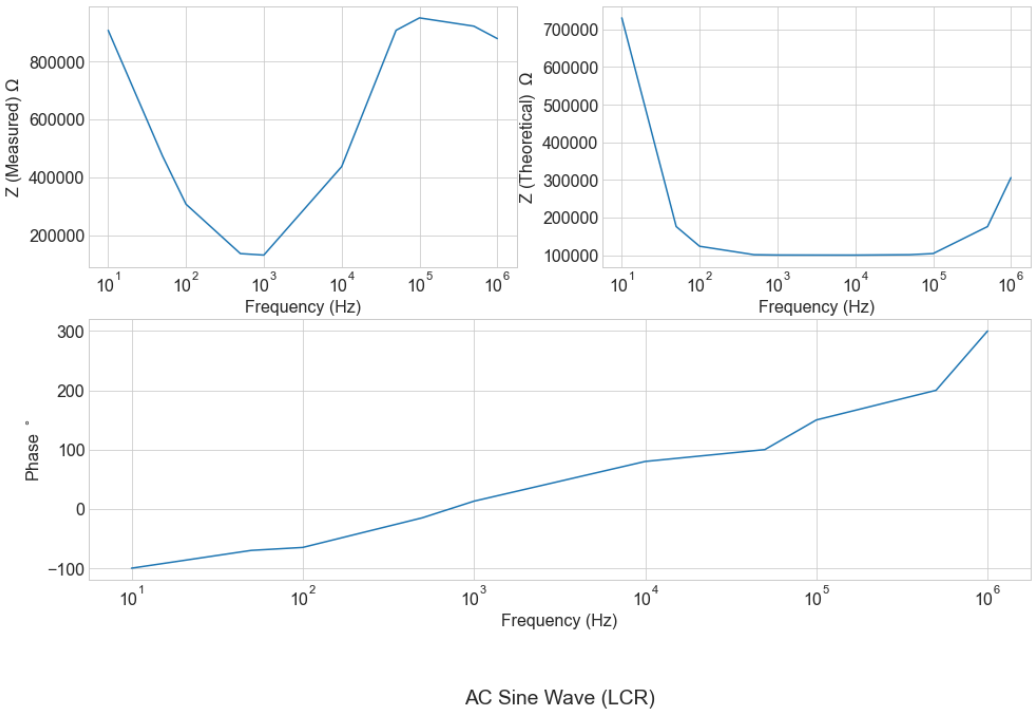


Figure 6: AC Sine Wave (LCR): Z vs Frequency and Phase vs Frequency Plots

3 Uncertainty

For the purpose of this experiment we estimated the uncertainty in V_R as the thickness of the voltage line plotted on the Oscilloscope for channel 2, which measures the voltage across the Resistor. In case of DC circuit, with RC we observe that $\Delta V_R = \frac{1}{4} \times$ Voltage Division $= .25 \times 500\ mV = 125\ mV$. And for DC Square Wave, it is 2 V. Similarly for LR Square Wave case, $\Delta V_R = 1\ V$. Overall for the experiment we can assume the uncertainty in the measurement of V_R to be of the order of 1 V.

4 Discussion

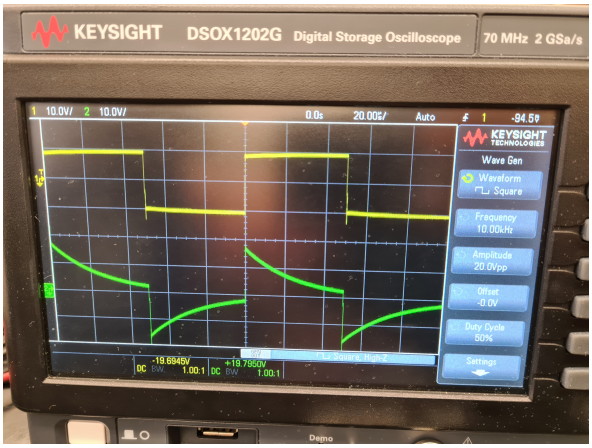
In RC circuit, the accuracy of the measurement of V_R in DC case is given by $\frac{|1.4/e - 1.|}{1.4/e} \times 100 = 35\%$. And the precision of measurement of V_R is given by by $\frac{0.125}{0.7} \times 100 = 17\%$. The accuracy value being large can be attributed to the instrument.

In LR circuit, the accuracy of V_R in DC case is given by $\frac{|12.64 - 10.0|}{12.64} \times 100 = 20\%$. And the precision of measurement of V_R is given by by $\frac{1}{10} \times 100 = 10\%$.

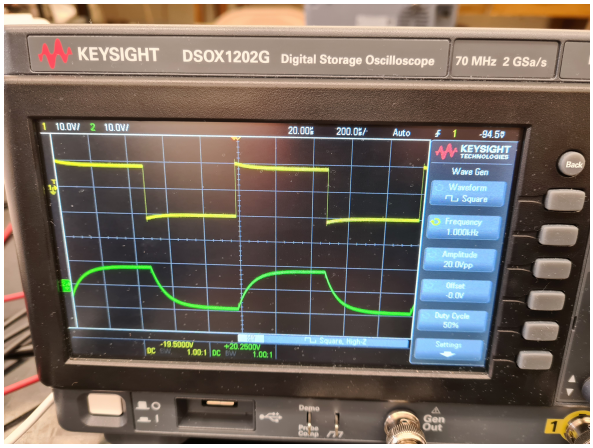
In the LCR circuit for lower frequencies the phase difference is negative, and as the frequency is increased the phase gradually shifts to positive values. The resonance frequency can be estimated to be around 1 kHz. And as per theory $\omega_r = 1/\sqrt{LC} = 1.4$ kHz.

A Appendix

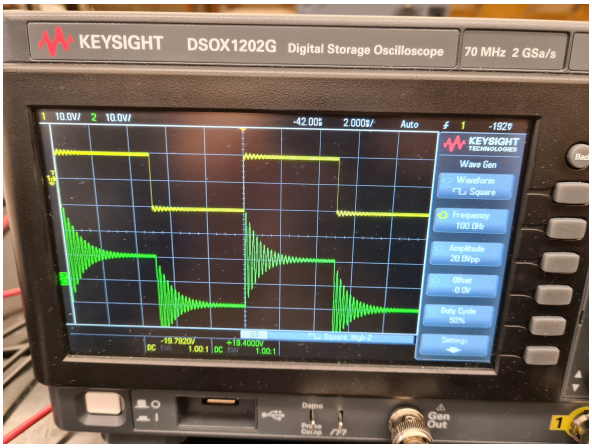
A.1 Oscilloscope Readings



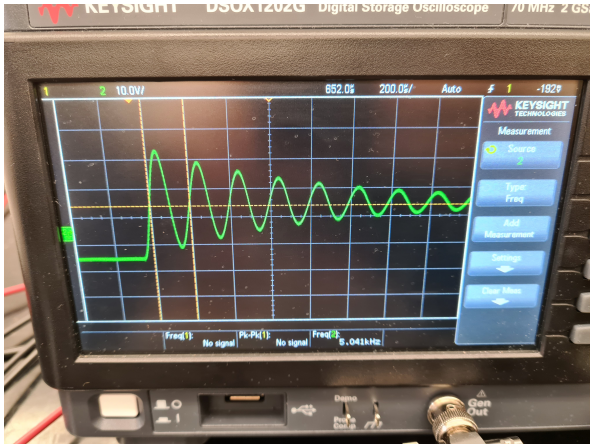
(a) RC Circuit with Square Wave



(b) LR Circuit with Square Wave



(a) LC Circuit with Square Wave



(b) LC Circuit with Square Wave (Zoomed In)

A.2 Python Code

A.3 Simulated Curves

```
# -*- coding: utf-8 -*-
"""
Created on Thu Nov 25 16:47:07 2021

@author: Fredrik
"""

#Importing modules
import numpy as np
import matplotlib.pyplot as plt

plt.style.use("seaborn-whitegrid")

#Defining constants
R_RC_B = 470*10**3 #Ohm
C_RC_B = 1*10**-6 #Ferad
V0_RC_B = 1

R_RC_W = 500
C_RC_W = 0.022*10**-6
V0_RC_W = 15

R_LR_W = 500
L_LR_W = 46*10**-3 #Henry
V0_LR_W = 20

#Tau:
Tau_RC_B = R_RC_B*C_RC_B
Tau_RC_W = R_RC_W*C_RC_W
Tau_LR_W = L_LR_W/R_LR_W

#Making array of points for plotting graphs:
t_RC_B = np.linspace(0,0.2*9,1000)
V_RC_B = V0_RC_B*np.exp(-t_RC_B/Tau_RC_B)
t_RC_W = np.linspace(0,10*10**-6*5,1000)
V_RC_W = V0_RC_W*np.exp(-t_RC_W/Tau_RC_W)
t_LR_W = np.linspace(0,200*10**-6*3,1000)
V_LR_W = V0_LR_W*np.exp(-t_LR_W/Tau_LR_W)


#we then plot the corresponding curves of exponential decay or increase
#For the RC circuit with battery:
fig = plt.figure(figsize=(16,10))
ax = fig.add_subplot(1,2,1)
ax.axis('off')
ax.set_title("Observed Exponetial Decay \nFor RC Curcuit with Battery")
ax.grid()
```

```
ax = fig.add_subplot(1,2,2)
ax.set_title("Exponential Decay of the Potensial Over the Resistor"+
             "\n For an RC Curcuit with a Battery")
ax.plot(t_RC_B,V_RC_B,c='r', ls='', marker='.',lw=1)
ax.set_xlabel("Time in seconds (s)")
ax.set_ylabel("Voltage (V)")
ax.figure.savefig("Exponential Decay of the Potential Over the Resistor"+
                  "For an RC Curcuit with a Battery"+".png")

#For the RC cirvuit with wave generator
fig = plt.figure(figsize=(16,10))
ax = fig.add_subplot(1,2,1)
ax.axis('off')
ax.set_title("Observed Exponetial Decay \nFor RC Curcuit with Wave Generator")
ax.grid()

ax = fig.add_subplot(1,2,2)
ax.set_title("Exponential Decay of the Potensial Over the Resistor"+
             "\n For an RC Curcuit with the Wave Generator")
ax.plot(t_RC_W,V_RC_W,c='r', ls='', marker='.',lw=1)
ax.set_xlabel("Time in seconds (s)")
ax.set_ylabel("Voltage (V)")
ax.figure.savefig("Exponential Decay of the Potential Over the Resistor"+
                  "For an RC Curcuit with the Wave Generator"+".png")


#For the LR circuit with wave generator
fig = plt.figure(figsize=(16,10))
ax = fig.add_subplot(1,2,1)
ax.axis('off')
ax.set_title("Observed Exponetial Increase \nFor LR Curcuit with Wave Generator")
ax.grid()

ax = fig.add_subplot(1,2,2)
ax.set_title("Exponential Increase of the Potential Over the Resistor"+
             "\n For an LR Curcuit with the Wave Generator")
ax.plot(t_LR_W,10-V_LR_W,c='r', ls='', marker='.',lw=1)
ax.set_xlabel("Time in seconds (s)")
ax.set_ylabel("Voltage (V)")
ax.grid()
ax.figure.savefig("Exponential Increase of the Potensial Over the Resistor"+
                  "For an LR Curcuit with the Wave Generator"+".png")


plt.show()
```

A.4 Impedance and Phase Plots

The Python code for this exercise is divided into two files. statslab.py file contains utility methods which we will be frequently using in this course. lab_7.py file contains the code which analyzes the data.

A.4.1 statslab.py

```
import numpy as np
import scipy.optimize as optim
import matplotlib.pyplot as plt


##############################################################################
# Utility Methods Library
#
# This file contains some utility method which are common to our data analysis.
# This library also contains customized plotting methods.
##############################################################################

# use bigger font size for plots
plt.rcParams.update({'font.size': 16})

def chi2(y_measure,y_predict,errors):
    """Calculate the chi squared value given a measurement with errors and
    prediction"""
    return np.sum( np.power(y_measure - y_predict, 2) / np.power(errors, 2) )

def chi2reduced(y_measure, y_predict, errors, number_of_parameters):
    """Calculate the reduced chi squared value given a measurement with errors
    and prediction, and knowing the number of parameters in the model."""
    return chi2(y_measure, y_predict, errors)/ \
            (y_measure.size - number_of_parameters)

def read_data(filename, skiprows=1, usecols=(0,1), delimiter=","):
    """Load give\n file as csv with given parameters,
    returns the unpacked values"""
    return np.loadtxt(filename,
                      skiprows=skiprows,
                      usecols=usecols,
                      delimiter=delimiter,
                      unpack=True)

def fit_data(model_func, xdata, ydata, yerrors, guess=None):
    """Utility function to call curve_fit given x and y data with errors"""
    popt, pcov = optim.curve_fit(model_func,
                                 xdata,
                                 ydata,
```

```
                                    absolute_sigma=True,
                                    sigma=yerrors,
                                    p0=guess)

    pstd = np.sqrt(np.diag(pcov))
    return popt, pstd

# y = ax+b
def linear_regression(xdata, ydata):
    """Simple linear regression model"""
    x_bar = np.average(xdata)
    y_bar = np.average(ydata)
    a_hat = np.sum( (xdata - x_bar) * (ydata - y_bar) ) / \
            np.sum( np.power((xdata - x_bar), 2) )
    b_hat = y_bar - a_hat  * x_bar
    return a_hat, b_hat

class plot_details:
    """Utility class to store information about plots"""
    def __init__(self, title):
        self.title = title
        self.x_log_scale = False
        self.y_log_scale = False

    def errorbar_legend(self, v):
        self.errorbar_legend = v
    def fitted_curve_legend(self, v):
        self.fitted_curve_legend = v
    def x_axis_label(self, v):
        self.x_axis_label = v
    def y_axis_label(self, v):
        self.y_axis_label = v
    def xdata(self, x):
        self.xdata = x
    def ydata(self, y):
        self.ydata = y
    def yerrors(self, y):
        self.yerrors = y
    def xdata_for_prediction(self, x):
        self.xdata_for_prediction = x
    def ydata_predicted(self, y):
        self.ydata_predicted = y
    def legend_position(self, p):
        self.legend_loc = p
    def chi2_reduced(self, c):
        self.chi2_reduced = c
    def set_x_log_scale(self, c):
        self.x_log_scale = c
    def set_y_log_scale(self, c):
        self.y_log_scale = c
```

```
def plot(plot_details, new_figure=True, error_plot=True):
    """Utility method to plot errorbar and line chart together,
    with given arguments"""
    if new_figure:
        fig  = plt.figure(figsize=(16, 10))
        fig.tight_layout()
        plt.style.use("seaborn-whitegrid")

    # plot the error bar chart
    if error_plot:
        plt.errorbar(plot_details.xdata,
                     plot_details.ydata,
                     yerr=plot_details.yerrors,
                     marker="o",
                     label=plot_details.errorbar_legend,
                     capsize=2,
                     ls="")

    # plot the fitted curve
    plt.plot(plot_details.xdata_for_prediction,
             plot_details.ydata_predicted,
             label=plot_details.fitted_curve_legend)

    # legend and title
    plt.title(plot_details.title)
    plt.xlabel(plot_details.x_axis_label)
    plt.ylabel(plot_details.y_axis_label)

    if plot_details.x_log_scale:
        plt.xscale("log")

    if plot_details.y_log_scale:
        plt.yscale("log")

    legend_pos = "upper left"
    if hasattr(plot_details, "legend_loc"):
        legend_pos = plot_details.legend_loc

    plt.legend(loc=legend_pos)
```

A.4.2 lab_7.py

```
#!/usr/bin/env python3
# @author: Pankaj
# -*- coding: utf-8 -*-

import math
import statslab as utils
import matplotlib.pyplot as plt
import numpy as np

# import the measured data from data files
datasets = [
    {
     "name": "AC Sine Wave (RC)",
     "file": "../data/rc_sine.csv",
     "R": 100000,
     "C": 0.022 * 10.0 ** (-6),
     "L": 0,
     "impedance_func": lambda w, R, C, L:
         np.sqrt(1.0 / (w * C) ** 2 + R ** 2)
    },
    {
     "name": "AC Sine Wave (LR)",
     "file": "../data/lr_sine.csv",
     "R": 100000,
     "C": 0,
     "L": 46*10**-3,
     "impedance_func": lambda w, R, C, L:
         np.sqrt( (w * L) ** 2 + R ** 2)
    },
    {
     "name": "AC Sine Wave (LCR)",
     "file": "../data/lcr_sine.csv",
     "R": 100000,
     "C": 0.022 * 10.0 ** (-6),
     "L": 46*10** (-3),
     "impedance_func": lambda w, R, C, L:
         np.sqrt( R ** 2 + np.power(w * L - 1.0 / (w * C), 2 ) )
    }
]


def analyze_data(data):
    print(data["name"])
    print("\tAnalyzing file %s" % data["file"])
    frequency, v_total, v_r, phase = utils.read_data(data["file"],
                            usecols=(0, 1, 2, 3),
                            skiprows=1)
```

```
    # frequency = omega / 2pi
    omega = 2*math.pi*frequency

    # z = v/v_r * R
    z_measured = v_total / v_r * data["R"]

    # z = sqrt((omega L - 1/omega C)^2 + R^2)
    z_theory = data["impedance_func"](omega, data["R"], data["C"], data["L"])

    fig = plt.figure(figsize=(16,10))
    fig.tight_layout()

    plt.subplot(2,2,1)
    plt.semilogx(frequency,z_measured)

    plt.xlabel("Frequency (Hz)")
    plt.ylabel("Z (Measured) $\Omega$")

    plt.subplot(2,2,2)
    plt.semilogx(frequency, z_theory)
    plt.xlabel("Frequency (Hz)")
    plt.ylabel("Z (Theoretical)  $\Omega$")

    plt.subplot(2,1,2)
    plt.semilogx(frequency, phase)
    plt.xlabel("Frequency (Hz)")
    plt.ylabel("Phase $^\circ$")
    plt.title(data["name"], y=-0.5)

    plt.savefig("%s.png" % data["name"], bbox_inches='tight')


for data in  datasets:
    analyze_data(data)
```

References

[1] Currents in LCR - currents-l-c-r.pdf (https://q.utoronto.ca/courses/235154/files/15436318/download?wrap=1).