Charge to Mass Ratio for Electron

PHY224 Fall 2021

Fredrik Dahl Bråten, Pankaj Patil

November 29, 2021

Abstract

In this variation of J. J. Thomson experiment, we aim to estimate the charge to mass ratio of an electron. As predicted by the theory, we observe the deflection of moving charge in a magnetic field. The electron is deflected in such a way that it forms a circular trajectory, for which the radius can be measured. This radius can then be used to measure the charge to mass ratio for an electron with the help of theoretical formulas relating the magnetic field to the radius of the orbit.

1 Introduction

A. Background Theory

Electron moving with velocity $\vec{v}$ through a magnetic field $\vec{B}$, experiences a force $\vec{F} = e\vec{v} \times \vec{V}$. For constant magnetic field and velocity perpendicular to it, the electron moves in circular orbit, such that $evB = m\frac{v^2}{r}$.

When electron is accelerated through potential V, we have $eV = \frac{1}{2}mv^2$, which when combined with former equation gives $\frac{1}{r} = \sqrt{\frac{e}{2m}}\frac{B}{\sqrt{V}}$.

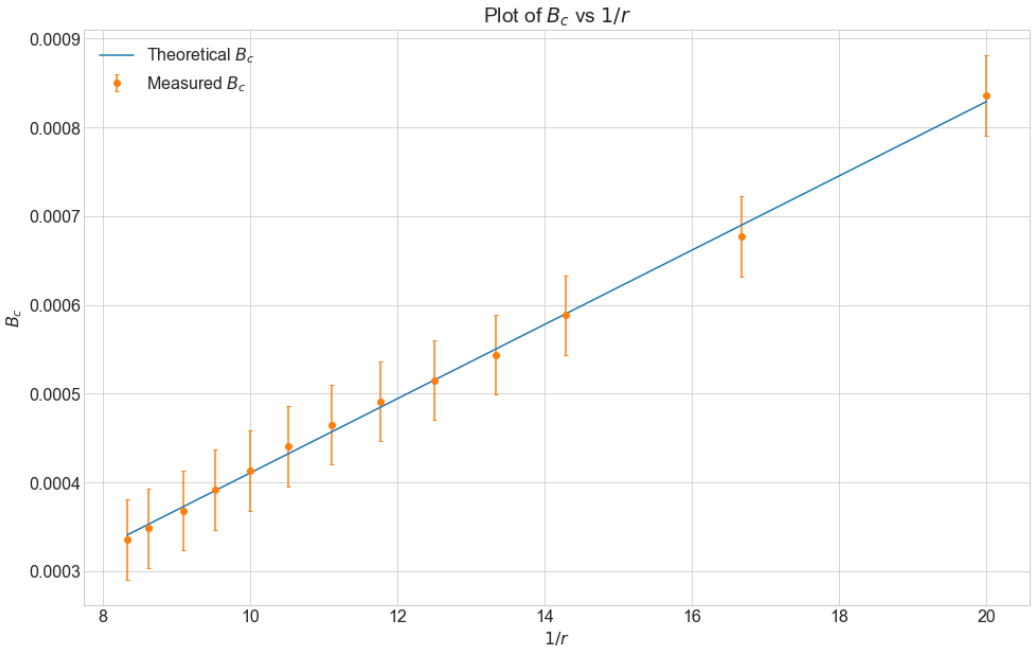
By measuring the potential and the radius of the orbit, we can determine the charge to mass ratio with above formula.

B. Materials and Methods

We followed the instructions given in the lab manual, and also the instruction given to us by the TA.

2 Results

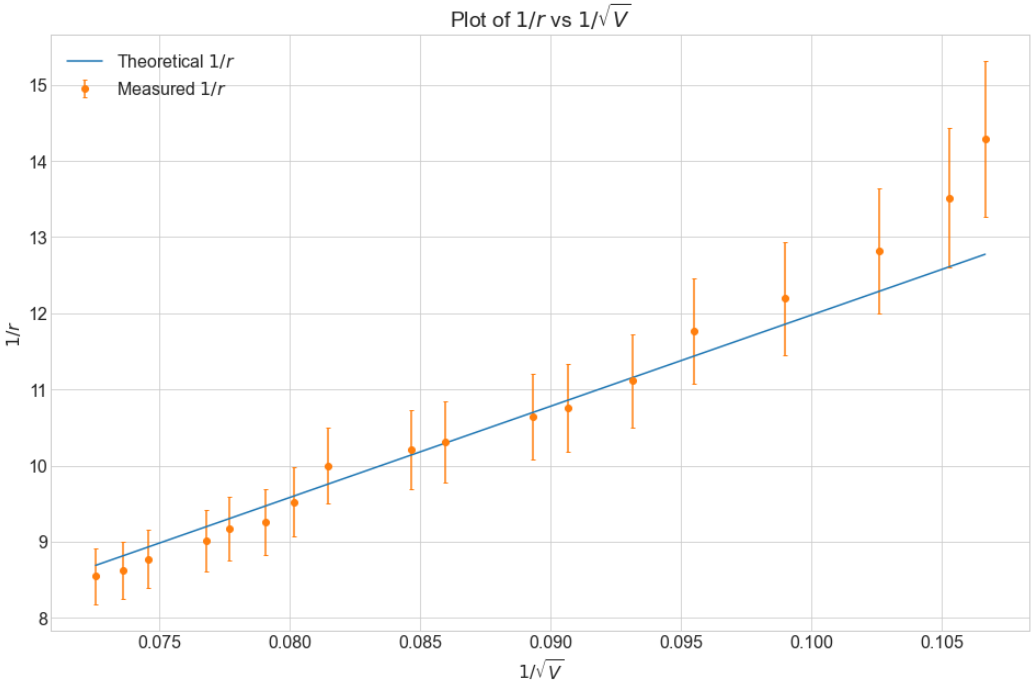
To compute the external magnetic field, we use $\frac{1}{r} = \sqrt{\frac{e}{2m}}\frac{1}{\sqrt{V}}(B_c + B_e) \implies B_c = (constant)\frac{1}{r} - B_e$. The constant factor is due to measurements done with constant voltage. The intercept of the fitted line gives $B_e = 0.00001 \pm 0.00005$ Tesla.



Fitting Equation $f(x) = ax + b$
$B_e = -b$

Figure 1: Computation of External Magnetic Field: B_c vs $1/r$

To compute the charge to mass ratio we use $\frac{1}{r} = \sqrt{\frac{e}{m}} k \frac{I - I_0}{\sqrt{V}}$. For this case we use the data obtained keeping current constant. The charge to mass ratio is give by $(\frac{slope}{k(I-I_0)})^2$. The charge to mass ratio obtained in our case is $-(1.61 \pm 0.04) \times 10^{11}\ C/kg$.



Fitting Equation $f(x) = ax$
$a = 119.77$

Figure 2: Computation of e/m: $1/r$ vs $1/\sqrt{V}$

3 Uncertainty

For the calculation of B_e, the uncertainty is given by the fitting function. Hence $\Delta B_e = \Delta(y - intercept) = \pm 0.00005$ Tesla.

For the computation of charge to mass ratio, we have $\frac{e}{m} = (\frac{slope}{k(I-I_0)})^2 \implies \Delta(\frac{e}{m}) = 2(slope)\frac{\Delta(slope)}{(k(I-I_0))^2}$. The uncertainty is then $\Delta(\frac{e}{m}) = 0.04 \times 10^{11}\ C/kg$.

4 Discussion

A Appendix

A.1 Python Code

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_8.py file contains the code which analyzes the data.

A.1.1 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
```

A.1.2 lab_8.py

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

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

# constants
mu_0 = 4*np.pi*10**(-7)
n = 75 # number of turn of the coil.
R = 0.15 # 15 cm radius of the coil.

# characteristic of coil dimension
k = 1/np.sqrt(2)*(4/5)**(3/2)*mu_0*n/R

# computation of B_e
# work with constant voltage data
r, measured_currents = utils.read_data("../../data/Changing_current.csv",
                usecols=(0, 1),
                skiprows=2)

r = r * 10 ** (-2) # to meters
r_1 = 1/r
B_c = (4/5)**(3/2)*mu_0*n/R*measured_currents

V_constant = 125.0 # Volts

# linear fitting equation
def model_function_Be(x, a, b):
    return  a*x + b

B_c_errors = np.ones_like(measured_currents) * (4/5)**(3/2)*mu_0*n/R * 0.1

popt, pstd = utils.fit_data(model_function_Be,
                            r_1,
                            B_c,
                            B_c_errors)

# get the y intercept
B_e = -popt[1]
print("External Magnetic Field B_e = %.5f +/- %.5f Tesla" % (B_e, pstd[1]))


# plot the predicted and measured data
fig = plt.figure(figsize=(16,10))
fig.tight_layout()
```

```
xdata = np.linspace(np.min(r_1),
                    np.max(r_1), 1000)
ydata = model_function_Be(xdata, popt[0], popt[1])
plt.plot(xdata, ydata, label="Theoretical $B_c$")
plt.xlabel("$1/r$")
plt.ylabel("$B_c$")
plt.title("Plot of $B_c$ vs $1/r$")

plt.errorbar(r_1,
             B_c,
             yerr=B_c_errors,
             marker="o",
             label="Measured $B_c$",
             capsize=2,
             ls="")

plt.legend()
plt.savefig("Coil B vs r_1.png", bbox_inches='tight')


# work with constant current data
r, measured_voltages = utils.read_data("../../data/Changing_voltage.csv",
                                       usecols=(0, 1),
                                       skiprows=2)

r = r * 10 ** (-2) # to meters
r_1 = 1/r
r_1_errors = np.ones_like(r_1) * 0.005 / (r ** 2)

I_constant = 0.964 # Volts
I_0 = B_e / k

# linear fitting equation
def model_function(x, a):
    return  a*x

popt, pstd = utils.fit_data(model_function,
                            1 / np.sqrt(measured_voltages),
                            r_1,
                            r_1_errors)

slope = popt[0]

print("Slope of the line = %.2f" % slope)

# plot the predicted and measured data
fig = plt.figure(figsize=(16,10))
fig.tight_layout()
```

```
xdata = np.linspace(np.min(1 / np.sqrt(measured_voltages)),
                    np.max(1 / np.sqrt(measured_voltages)), 1000)
ydata = model_function(xdata, slope)
plt.plot(xdata, ydata, label="Theoretical $1/r$")
plt.xlabel("$1/\sqrt{V}$")
plt.ylabel("$1/r$")
plt.title("Plot of $1/r$ vs $1/\sqrt{V}$")

plt.errorbar(1 / np.sqrt(measured_voltages),
             r_1,
             yerr=r_1_errors,
             marker="o",
             label="Measured $1/r$",
             capsize=2,
             ls="")

plt.legend()
plt.savefig("Charge To Mass Ratio.png", bbox_inches='tight')


charge_to_mass_ratio = (slope / (k * (I_constant - I_0) ) ) ** 2
charge_to_mass_ratio_error = 2 * slope * pstd[0]/ \
    (k * (I_constant - I_0) ) ** 2
print("Charge to Mass Ratio for Electron = -%.2e +/- %.2e C/kg"\
      % (charge_to_mass_ratio, charge_to_mass_ratio_error))
```
