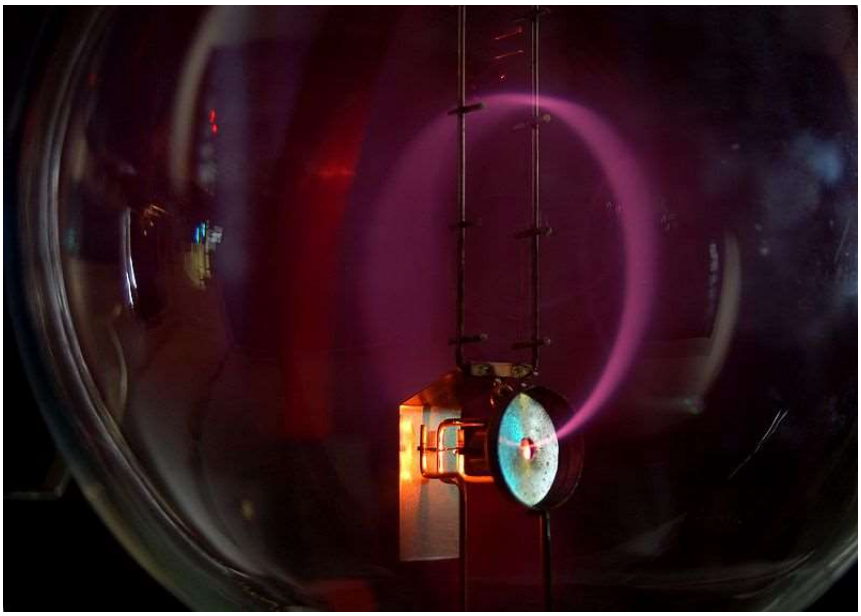
Worksheet 2 — E/M Ratio

William Thompson 10091404 12wt9@queensu.ca

Table of Contents:



1 – Preamble

1A – Modules

Import NumPy, Matplotlib, SymPy, and more

In [1]:
```
%pylab inline
import re
import sympy
from IPython.display import display, Markdown, Latex
import requests
import scipy.interpolate as interpolate
import scipy.stats as stats
from scipy.constants import physical_constants, mu_0
```

```
Populating the interactive namespace from numpy and matplotlib
```

1B – Units

Define units to convert measurements to SI

In [2]:
```
m = 1
cm = 1e-2*m
mm = 1e-3*m
V = 1
A = 1
T = 1
nT = 1e-9*T
```

2 – Functions

2A – Model

In [3]:
```
def B0(K, I):
    return K*I
```

In [4]:
```
def B(r, R, B0):
    return
```

In [5]:
```
def Kr(B, B0, K):
    return B/B0 * K
```

In [6]:
```
def B_T(Kr, I_s, I_l):
    return Kr * (I_s + I_l) / 2
```

In [7]:
```
def B_E(Kr, I_s, I_l):
    return Kr * (I_s - I_l) / 2
```

K_r expression:

Interpolate a function from the provided data table

In [8]:
```
# This is the data table provided in the experiment outline.
radius_vs_field_table = array([
    [0.0, 1],
    [0.1, 0.99996],
    [0.2, 0.99928],
    [0.3, 0.99621],
    [0.4, 0.98728],
    [0.5, 0.96663],
    [0.6, 0.92525],
    [0.7, 0.85121],
    [0.8, 0.73324],
    [0.9, 0.56991],
    [1.0, 0.38007]
])

# Do the interpolation
radius_vs_field_interp = interpolate.UnivariateSpline(
    # X-values
    radius_vs_field_table[:,0],
    # Y-values
     radius_vs_field_table[:,1]
)
def K_r(K, r, R):
    return radius_vs_field_interp(r/R) * K
```

In [9]:
```
def e_over_m(V, B, r):
    return 2*V/(B**2*r**2)
```

2B – Data Processing Functions

Subtract 1% from anode voltage readings:

In [10]:
```
def correct_anode_voltage(data):
    return data * [0.99, 1]
```

Convert a float into a latex exponential notation: **`latex_exp(2.3e-3)`** $\rightarrow 2.3 \times 10^{-3}$

In [11]:
```
def latex_exp(x, pres=2):
    exp = int(math.log10(abs(x)))
    mant = x / 10**exp * sign(x)
    return ('{:.'+str(pres)+'f} \\times 10^{{{:d}}}').format(mant, exp)
```

Get the strength of the magnetic field in Teslas using the NOAA website (http://www.ngdc.noaa.gov/geomag-web/#igrfwmm):

In [12]:
```
def get_B_E(year, month, day, lat, long):
    url = 'http://www.ngdc.noaa.gov/geomag-web/calculators/calculateIgrfwmm'
    response = requests.post(url, data={
        'browserRequest': 'true',
        'coordinateSystem': 'M',
        'dateStepSize': 1.0,
        'elevation': 1,
        'elevationUnits': 'K',
        'endDay': day,
        'endMonth': month,
        'endYear': year,
        'lat1': lat,
        'lat1Hemisphere': 'N',
        'lon1': long,
        'lon1Hemisphere': 'W',
        'model': 'WMM',
        'resultFormat': 'csv',
        'startDay': day,
        'startMonth': month,
        'startYear': year
    })
    csv = re.split('[,\n]', response.text)
    return float(csv[-2])*nT, float(csv[-10])*nT
```

In [13]:
```
def display_tabular_data(headers, data):
    output = '|'+'|'.join(str(h) for h in headers)+'|\n' +\
             '|'+'|'.join('-' for h in headers)+'|\n'
    output += '\n'.join('|'+'|'.join(str(v) for v in row)+'|' for row in data)
    return Markdown(output)
```

2C – Data Analysis Functions

In [14]:
```
def analyze_paired_data(diam, V, Il, Is):

    # Calculate Kr
    Kr = K_r(K, diam/2, R)

    # Eath's magnetic field
    Be = B_E(Kr, Is, Il)

    # Total field
    Bt = B_T(Kr, Is, Il)

    # Charge to mass ratio
    em = e_over_m(V, Bt, diam/2)

    return Be, Bt, em
```

3 – Data

3A – Readings

```
R = 15.4*cm
R_err = 5*mm

K = 7.73e-4 * T/A
K_err = 0.04e-4 *T/A

diamter_err = 1.5*mm

voltage_err = 0.5*V
amperage_err = 0.001*A

N = 130

# Store our results by diameter, and by pol
results = {
     8*cm:{},
    10*cm:{}
}

# 10cm radius, pol 1
results[10*cm]['pol1'] = array([
    [150.00, 0.966],
    [170.50, 1.020],
    [190.35, 1.068],
    [210.00, 1.161],
    [230.26, 1.196],
    [249.97, 1.275],
]) * [V, A] # Specify units

# 10cm radius, pol 2
results[10*cm]['pol2'] = array([
    [150.00, 1.109],
    [170.00, 1.157],
    [190.35, 1.221],
    [210.06, 1.312],
    [230.35, 1.335],
    [250.27, 1.402],
]) * [V, A] # Specify units

# 8cm radius, pol 1
results[8*cm]['pol1'] = array([
    [150.55, 1.176],
    [171.63, 1.302],
    [191.29, 1.350],
    [209.58, 1.434],
    [230.36, 1.493],
    [250.66, 1.557]
]) * [V, A] # Specify units

# 8cm radius, pol 2
results[8*cm]['pol2'] = array([
```

```
    [150.58, 1.360],
]) * [V, A] # Specify units
```

In tabular form:

10cm, pol1

In [16]:
```
display_tabular_data(
    ['Volts', 'Current'],
    results[10*cm]['pol1']
)
```

Out[16]:

Volts	Current
150.0	0.966
170.5	1.02
190.35	1.068
210.0	1.161
230.26	1.196
249.97	1.275

10cm, pol2

In [17]:
```
display_tabular_data(
    ['Volts', 'Current'],
    results[10*cm]['pol2']
)
```

Out[17]:

Volts	Current
150.0	1.109
170.0	1.157
190.35	1.221
210.06	1.312
230.35	1.335
250.27	1.402

In [18]:
```
display_tabular_data(
    ['Volts', 'Current'],
    results[8*cm]['pol1']
)
```

Out[18]:

Volts	Current
150.55	1.176
171.63	1.302
191.29	1.35
209.58	1.434
230.36	1.493
250.66	1.557

8cm, pol2

In [19]:
```
display_tabular_data(
    ['Volts', 'Current'],
    results[8*cm]['pol2']
)
```

Out[19]:

Volts	Current
250.14	1.702
230.57	1.634
210.63	1.578
190.7	1.498
170.12	1.38
150.58	1.36

3A – Accepted Values

In [20]:
```
(
    B_E_accepted,
    B_E_accepted_err
) = get_B_E(
    year=2016,
    month=2,
    day=3,
    lat=44.2253523,
    long=76.5009076
)
Markdown('NOAA reports that $B_E$ is locally {:.4e} $\pm$ {:.2e} T'.format(
    B_E_accepted,
    B_E_accepted_err
))
```

Out[20]: NOAA reports that B_E is locally 5.0522e-05 $\pm$ 1.65e-07 T

In [21]:
```
(
    E_M_ratio_accepted,
    _, # Skip
    E_M_ratio_accepted_err
) = physical_constants['electron charge to mass quotient']
E_M_ratio_accepted *= -1 # Make it positive instead of negative

Markdown('SciPy.constants reports that $e/m$ = {:.9e} $\pm$ {:.1e}'.format(
    E_M_ratio_accepted,
        E_M_ratio_accepted_err
))
```

Out[21]: SciPy.constants reports that e/m = 1.758820088e+11 $\pm$ 3.9e+03

4 – Analysis

4A – Correct for anode voltage drop

In [22]:
```
for radius, data_by_pol in results.items():
    for pol, data in data by pol.items():
```

4B – Calculate B_E, B_T, and e/m

Pair up data from pol1 and pol2 for each diameter

In [23]:
```
# View into results: results pol1 & pol2 for each diameter
results_iterator = [
    (diam, data_by_pol['pol2'], data_by_pol['pol1'])
    for diam, data_by_pol in results.items()
]

B_E_results = []
B_T_results = []
E_M_results = []

# Loop through each diameter, with pols paired.
for diam, data_l, data_s in results_iterator:

    # Select current from data_l, and data_s
    Il = data_l[:,1]
    Is = data_s[:,1]

    # Voltage is the average between the data sets
    V  = (data_s[:,0]+data_l[:,0])/2

    Be, Bt, em = analyze_paired_data(diam, V, Il, Is)

    # Append to lists
    B_E_results = np.concatenate((B_E_results, Be))
    B_T_results = np.concatenate((B_T_results, Bt))
    E_M_results = np.concatenate((E_M_results, em))
```

Now take the average of these results, and the error is the standard error on the mean:

In [24]:
```
B_E_averaged = -mean(B_E_results)
B_E_averaged_err = stats.sem(B_E_results, ddof=1)

B_T_averaged = mean(B_T_results)
B_T_averaged_err = stats.sem(B_T_results, ddof=1)

E M averaged = mean(E M results)
```

4C – Charge to Mass Ratio

Using the average of all the readings, we get that the charge-to-mass ratio e/m is:

In [25]:
```
Markdown("""
The calculated charge to mass ratio is: $e/m_{{calc}} = {em} \\pm {em_d}$\n
The accepted value is  $e/m_{{acc}} = {ema} \\pm {ema_d}$""".format(
    em    = latex_exp(E_M_averaged),
    em_d  = latex_exp(E_M_averaged_err,0),
    ema   = latex_exp(E_M_ratio_accepted),
    ema_d = latex_exp(E_M_ratio_accepted_err,0),
))
```

Out[25]: The calculated charge to mass ratio is: $e/m_{calc} = 1.92 \times 10^{11} \pm 2 \times 10^{9}$

The accepted value is $e/m_{acc} = 1.76 \times 10^{11} \pm 4 \times 10^{3}$

Which lies outside the uncertainty bounds.

In [31]:
```
E_M_percent_err = abs(E_M_averaged-E_M_ratio_accepted)/E_M_ratio_accepted * 100
Markdown(
    'The percent error relative to the accepted value is ${:.1f} \%$'
    .format(E_M_percent_err))
```

Out[31]: The percent error relative to the accepted value is 9.4%

This is a deviation from the expected result.

4D – Local Magnetic Field

In [27]:
```
Markdown("""
The calculated local magnetic field of the Earth is: $B_{{E_,calc}} = {be} \\pm
{be_d}$\n
The accepted value is  $B_{{E_,acc}} = {bea} \\pm {bea_d}$""".format(
    be   = latex_exp(B_E_averaged,3),
    be_d = latex_exp(B_E_averaged_err,1),
    bea   = latex_exp(B_E_accepted,3),
    bea_d = latex_exp(B_E_accepted_err,0),
))
```

Out[27]: The calculated local magnetic field of the Earth is: $B_{E,calc} = 0.544 \times 10^{-4} \pm 0.2 \times 10^{-4}$

The accepted value is $B_E \quad = 0\ 505 \times 10^{-4} \pm 0 \times 10^{-6}$

In [28]:
```
B_E_percent_err = abs(B_E_accepted-B_E_averaged)/B_E_accepted * 100
Markdown('The percent error in $B_E$ is: ${:.1f} \%$ '.format(B_E_percent_err))
```

Out[28]: The percent error in B_E is: 7.6%

4E – Derivation of Eq. (4)

The following is adapted from Physics Lab Online (http://dev.physicslab.org/document.aspx?doctype=3&filename=magnetism_biotsavartlaw2.xml):

We begin with the Bior-Savart Law:

$$d\vec{B} = \frac{\mu_0 I d\vec{l} \times \hat{r}}{4\pi r^2}$$

We are going to integrate this around each loop for each coil of wire. Our path around the coils will always be perpendicular to r, so the cross product becomes the magnitude of dl:

$$d\vec{B} = \frac{\mu_0 I d\vec{l}}{4\pi r^2} = \frac{\mu_0 I d\vec{l}}{4\pi(z^2+R^2)}$$

Since we are considering the central axis of the coils, it is clear that any horizontal contributions will cancel out due to symetry, do we only need to consider the vertical direction:

$$dB_x = \sin(\theta) dB = \frac{R}{r} dB = \frac{R}{\sqrt{z^2+R^2}} dB$$

$$dB_x = \frac{\mu_0 I d\vec{l}}{4\pi(z^2+R^2)^{3/2}}$$

Now we integrate around each coil:

$$\vec{B}_0 = \int_0^{2\pi n} \frac{\mu_0 I d\vec{l}}{4\pi(z^2+R^2)^{3/2}} \hat{z}$$

$$= \frac{2\pi n \mu_0 I}{4\pi(z^2+R^2)^{3/2}} \hat{z}$$

$$= \frac{n\mu_0 I}{2(z^2+R^2)^{3/2}} \hat{z}$$

Finally, we have two coils, so we can multiply this final answer by two to get the expected result:

$$\vec{B}_0 = \frac{n\mu_0 I}{(z^2+R^2)^{3/2}} \hat{z}$$

Which can be simplified to $B_0 = \frac{8n\mu_0 I}{5^{3/2} R} = KI$ and rearranged to find:

$$K = \frac{8n\mu_0}{5^{3/2} R}$$

In [29]:
```
K_expr = 8*N*mu_0/(5**1.5 * R)
K_expr_err = abs(8*N*mu_0/(5**1.5 * R**2)*R_err)
Latex('$K_{{expr}} = {} \pm {}$'.format(
    latex_exp(K_expr,3),
    latex_exp(K_expr_err,1)
))
```

Out[29]: $K_{expr} = 0.759 \times 10^{-3} \pm 0.2 \times 10^{-4}$

The manufacturer value for K is:

In [32]:
```
Latex('$K_{{expr}} = {} \pm {}$'.format(
    latex_exp(K,3),
    latex_exp(K_err,1)
))
```

Out[32]: $K_{expr} = 0.773 \times 10^{-3} \pm 0.4 \times 10^{-5}$

Which does fall within the uncertainties.

The relative error is:

In [33]:
```
rel_err = abs(K_expr-K)/K*100
Latex('${:.1f}\%$'.format(rel_err))
```

Out[33]: 1.8%