Charge to Mass Ratio

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

Read data

```
data = pd.read_csv("data.csv")
data
```

	V	I1	I2	I3	I4	I5
0	20	2.67	2.26	1.96	1.75	1.60
1	25	2.94	2.51	2.16	1.92	1.75
2	30	3.20	2.71	2.37	2.08	1.87
3	35	3.46	2.92	2.54	2.26	2.01
4	40	3.68	3.10	2.69	2.38	2.13
5	45	3.83	3.26	2.81	2.51	2.25

Calculate Helmholtz Field

```
currents = data.to_numpy()[:, 1:]
mu = 4*np.pi*1E-7
N = 72
R = 0.33
helmholtz_fields = 8*mu*N*currents/(np.sqrt(125)*R)
helmholtz_fields
```

```
array([[0.00052381, 0.00044338, 0.00038452, 0.00034332, 0.00031389],
       [0.00057678, 0.00049242, 0.00042376, 0.00037667, 0.00034332],
       [0.00062779, 0.00053166, 0.00046496, 0.00040806, 0.00036686],
```

```
       [0.0006788 , 0.00057286, 0.00049831, 0.00044338, 0.00039433],
       [0.00072196, 0.00060817, 0.00052773, 0.00046692, 0.00041787],
       [0.00075138, 0.00063956, 0.00055128, 0.00049242, 0.00044141]])
```

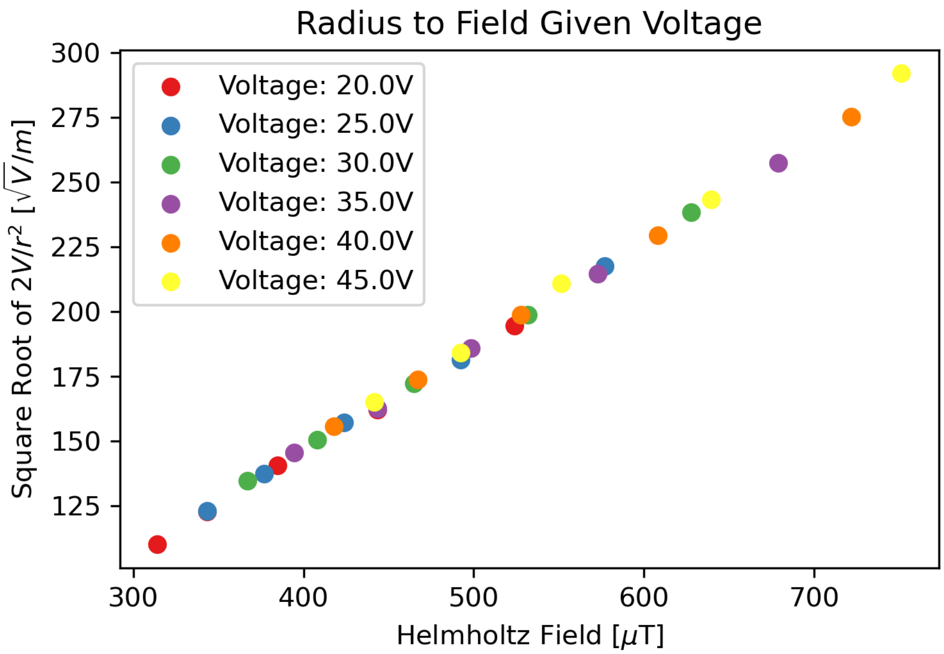
Diameters and voltages

```
diameters = np.array([0.065,0.078,0.090,0.103,0.115])

voltages = data.to_numpy()[:, 0]
voltages
```

```
array([20., 25., 30., 35., 40., 45.])
```

plot values

```
x = helmholtz_fields
y = np.sqrt((2*voltages[:, np.newaxis])/((diameters/2)**2))
slopes = np.zeros((6,2))
intercepts = np.zeros((6,2))

X = np.arange(0,900,100)

cmap = plt.get_cmap("Set1")
plt.figure()

for i in range(len(voltages)):
  plt.scatter(x[i]*1e6, y[i], color=cmap(i), label=f"Voltage: {voltages[i]}V")
  w, c = np.polyfit(x[i], y[i], 1, cov=True)
  e = np.sqrt(np.diag(c))
  slopes[i] = (w[0], e[0])
  intercepts[i] = (w[1], e[1])

plt.title("Radius to Field Given Voltage")
plt.xlabel("Helmholtz Field [$\mu$T]")
plt.ylabel("Square Root of $2V/r^2$ [$\sqrt{V}/m$]")
plt.legend()
plt.show()

for i in range(len(voltages)):
  print(f"{voltages[i]}V:")
  print(f" m = {slopes[i][0]:.3}+-{slopes[i][1]:.3}")
  print(f" b = {intercepts[i][0]:.3}+-{intercepts[i][1]:.3}\n")
```




```
20.0V:
 m = 4e+05+-6.1e+03
 b = -14.6+-2.49

25.0V:
 m = 4e+05+-7.82e+03
 b = -13.7+-3.52

30.0V:
 m = 3.97e+05+-4.31e+03
 b = -11.6+-2.11

35.0V:
 m = 3.96e+05+-4.94e+03
 b = -11.7+-2.61

40.0V:
 m = 3.94e+05+-2.95e+03
 b = -9.63+-1.65

45.0V:
 m = 4.08e+05+-7.01e+03
 b = -15.8+-4.1
```

To get the charge to mass ratio we do the following

$$m = \sqrt{\frac{e}{m_e}} \implies \frac{e}{m_e} = m^2$$

and the uncertainty is

$$\sigma_{e/m_e} = \frac{d\ e/m_e}{d\ m}\sigma_m = 2m\sigma_m$$

Then to calculate B_e we can do

$$B_e = \frac{b}{\sqrt{e/m}} = \frac{b}{m}$$

$$\sigma_{B_e} = \sqrt{(\frac{\partial B_e}{\partial m}\sigma_m)^2 + (\frac{\partial B_e}{\partial b}\sigma_m).3} = \sqrt{(\frac{\sigma_m}{m.3})^2 + \sigma_b.3}$$

and its uncertainty

```
intercepts[:, 0] = intercepts[:, 0]/slopes[:, 0]
intercepts[:, 1] = np.sqrt((slopes[:, 1]/slopes[:, 0]**2)**2 + intercepts[:, 1]**2)

slopes[:, 0] = slopes[:, 0]**2
slopes[:, 1] = slopes[:, 0]*slopes[:, 1]*2

for i in range(len(voltages)):
  print(f"{voltages[i]}V:")
  print(f" e/m = {slopes[i][0]:.3}+-{slopes[i][1]:.3}")
  print(f" B_e = {intercepts[i][0]:.3}+-{intercepts[i][1]:.3}\n")
```

```
20.0V:
 e/m = 1.6e+11+-1.95e+15
 B_e = -3.65e-05+-2.49

25.0V:
 e/m = 1.6e+11+-2.5e+15
 B_e = -3.43e-05+-3.52

30.0V:
 e/m = 1.58e+11+-1.36e+15
 B_e = -2.93e-05+-2.11
```

```
35.0V:
 e/m = 1.57e+11+-1.55e+15
 B_e = -2.95e-05+-2.61

40.0V:
 e/m = 1.55e+11+-9.18e+14
 B_e = -2.44e-05+-1.65

45.0V:
 e/m = 1.67e+11+-2.34e+15
 B_e = -3.88e-05+-4.1
```

Experimental Earth Field values yield:

```
earth_field = 8*mu*N*0.36/(np.sqrt(125)*R)
earth_field_error = 0.05
print(f"B_e = {earth_field:.3} +- {earth_field_error}T")
```

```
B_e = 7.06e-05 +- 0.05T
```

Both values agree but are not very precise, however the means of the fitted field line up better than the experimental field. The combined mean of a value is defined by

$$x_{\text{combined}} = \frac{\Sigma x_i/\sigma_i}{\Sigma 1/\sigma_i}, \sigma_{\text{combined}} = \frac{1}{\Sigma 1/\sigma_i}$$

This means that our combined fit would look like

```
slope, slope_err = (np.sum(slopes[:,0]/slopes[:,1])/np.sum(1/slopes[:,1]),1/np.sum(1/slopes[
intercept, intercept_err = (np.sum(intercepts[:,0]/intercepts[:,1])/np.sum(1/intercepts[:,1]
print(f" e/m = {slope:.3}+-{slope_err:.3}")
print(f" B_e = {intercept:.3}+-{intercept_err:.3}")
```

```
 e/m = 1.58e+11+-2.62e+14
 B_e = -3.09e-05+-0.418
```