In [1]:
```
from IPython.display import HTML

HTML('''<script>
code_show=true;
function code_toggle() {
 if (code_show){
 $('div.input').hide();
 } else {
 $('div.input').show();
 }
 code_show = !code_show
}
$( document ).ready(code_toggle);
</script>
The raw code for this IPython notebook is by default hidden for easier readin
g.
To toggle on/off the raw code, click <a href="javascript:code_toggle()">here</
a>.''')
```

Out[1]: The raw code for this IPython notebook is by default hidden for easier reading. To toggle on/off the raw code, click here.

James Drake

Section 1

Compton Scattering

Introduction

The goal of this lab is to investigate several effects of radiation from two radioactive sources, Cs-137 and Ba-133, which both radiate γ-rays. Radiation intensity is expected to follow the inverse square law and is thus proportional to $\frac{1}{r^2}$, which can be observed by varying the distance from a detector to a radioactive source. By placing sheets of material between the source and detector, attenuation can be observed. Assuming that the sheets are composed of the same material, the transmission probability is given by $T = e^{-\mu x}$, where x is the total thickness of the material and μ is a material-dependent attenuation coefficient. Furthermore, by scattering radiation from a target rod, Compton scattering can be observed, and the experimental differential cross section can be measured. From these quantities, the mass and classical radius of an electron can be found. This data can also be parametrized for a rod of unknown material in order to assess its chemical composition.

Procedure

This lab is split into four sections: inverse square law, attenuation, Compton scattering, and differential cross section.

Section 0: Calibration

Although calibration is not necessary for every section of this lab, I chose to start my analysis with this procedure. The data acquired in the lab was collected using a scintillation detector that was connected to a multichannel analyzer (MCA). The MCA readout sorts events into different channels, which correspond to a certain voltage signal. To convert from channel number to energy, it is assumed that the channels follow a linear relationship with energy such that $E = ac$, where E is the energy and c is the channel number. To find the coefficient a, one must fit a spectrum with known photopeak energies to the channels. This is done by selecting photopeaks that correspond to γ-ray emissions of known energy and fitting Gaussians to the peaks to find their centroids:

$$f(c) = Ae^{-\frac{1}{2}(\frac{c-\mu}{\sigma})^2}$$

Fit function for Gaussian peak. μ represents the centroid of the peak.

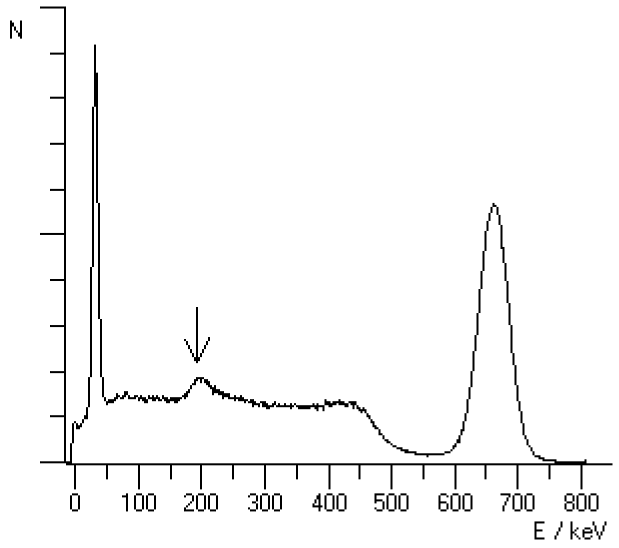

Figure 1. Emission spectrum for Cs-137. Credit to https://www.ld-didactic.de/software/524221en/Content/Appendix/Cs137.htm#:~:text=Caesium%2D137%20is%20a%20man,of%2

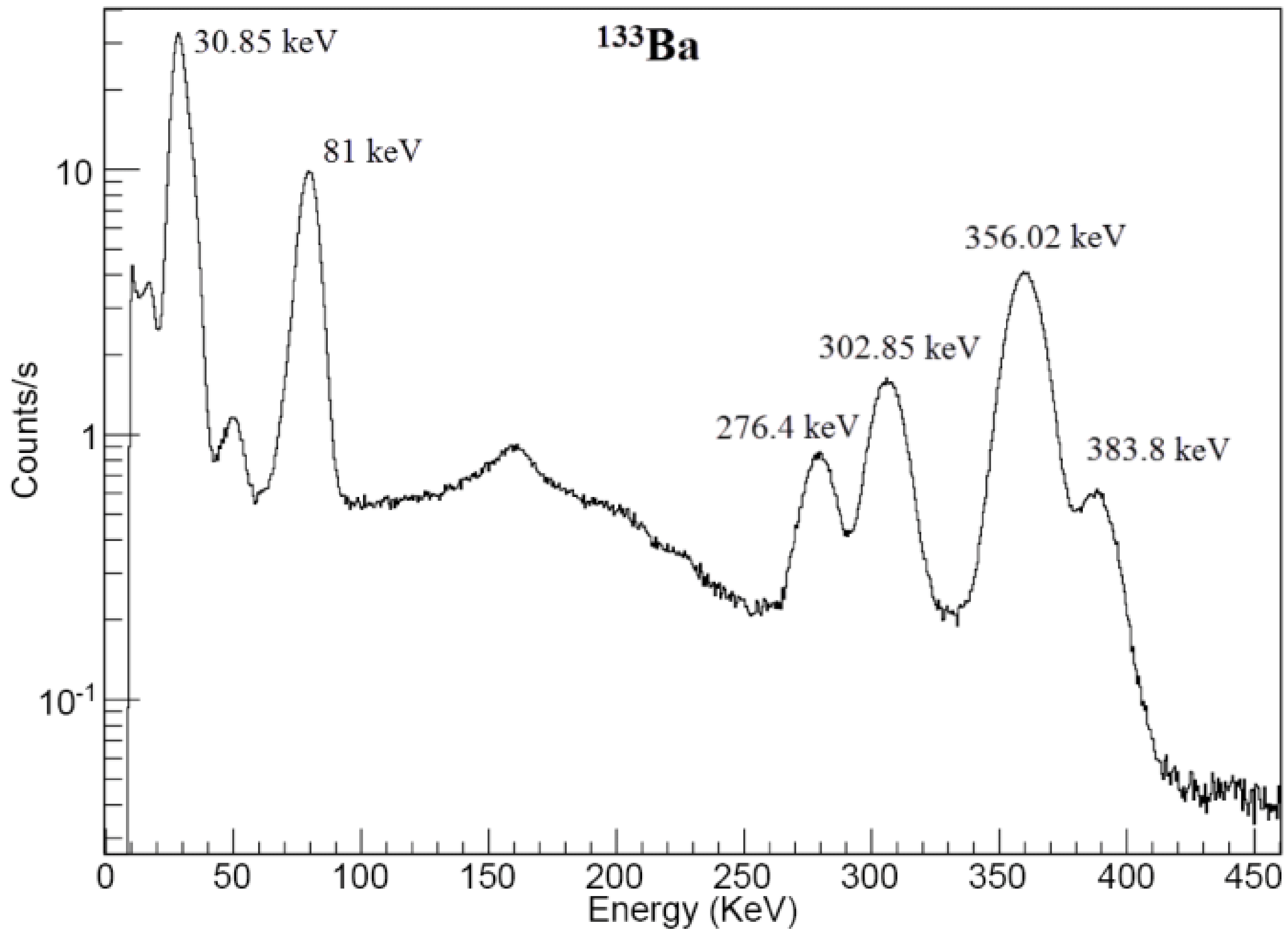

Figure 2. Emission spectrum for Ba-133. Credit to https://www.researchgate.net/figure/The-calibrated-spectrum-of-133-Ba_fig3_324492749

To calibrate the MCA channels, two spectra were used: one for Cs-137 (Figure 1), and one for Ba-133 (Figure 2). Cs-137 has one distinct photopeak at $661.7\ \mathrm{keV}$, while Ba-133 has three photopeaks at $30.85\ \mathrm{keV}$, $303\ \mathrm{keV}$, and $356\ \mathrm{keV}$. Although there were additional peaks present in the spectra pictured above, the four specified peaks were chosen due to their clear presence in the measured spectra (the rest were difficult to parse due to background events and detector resolution). After subtracting out the average background, each photopeak was fit to a Gaussian, and the centroid was associated with a photopeak energy. These four points were then used to generate a linear fit to find a, the conversion factor between channels and energy.

Section 1: Inverse Square Law

Radiation from a radioactive source decays in strength at a rate that is inversely proportional to the square of the distance:

$$I(r) = \frac{A}{r^2}$$

To observe this effect, a small sample of Ba-133 was placed in a disk holder and placed at different distances from the detector. To minimize background radiation from the Cs-137 source, the detector was aimed at a wall, 90° relative to the opening of the Cs-137 source (which was kept sealed for this section). Because the barium source was weakly radioactive, we prioritized measurements that were much closer to the detector rather than much farther away.

Two methods were used to analyze the inverse square law data. In both cases, a Gaussian was fit to the photopeak with the highest amplitude. For the first method, the height of this peak was used to fit to a function Ar^n, where A and n were free parameters. In the second method, the area was computed using the analytical form for the area under a Gaussian peak:

$$\int_a^b Ae^{-\frac{1}{2}(\frac{c-\mu}{\sigma})^2}\,dc = -\sqrt{\frac{\pi}{2}}A\sigma\left[erf(\frac{\mu-b}{\sqrt{2}\sigma}) - erf(\frac{\mu-a}{\sqrt{2}\sigma})\right]$$

In the equation above, erf is the error function. Again, these areas were fit to an arbitrary power function of r. In both cases, the value of n was compared to the expected value $n = -2$.

Section 2: Attenuation

Radiation incident on a different medium is subject to attenuation in that medium. The transmission probability for an individual γ-ray can be expressed as a decaying exponential:

$$T = e^{-\mu x}$$

Since μ depends on the material (and generally, the photon energy), it can be found by measuring the decrease in photopeak amplitude as a function of material thickness x. Experimentally, this was accomplished by measuring the emission spectrum of Cs-137 while varying the number of copper sheets between the detector and the source. The largest photopeak was fit to a Gaussian, which provided the amplitude of each peak. Each amplitude was then divided by the amplitude of the zero sheet fit and correlated with a thickness (product of the number of sheets used and the thickness of one sheet). From this, an exponential fit was generated and the attenuation coefficient of copper was found.

(Note: there is a discontinuity between the theory document and instructions document where the former specified aluminum sheets while the latter specified copper sheets. We used copper sheets to perform measurements in this section.)

Section 3: Compton Scattering

Photons incident on a massive particle scatter with a new energy given by the Compton scattering equation:

$$E' = \frac{E}{1 + (\frac{E}{mc^2})(1 - \cos\theta)}$$

In this section, radiation from the Cs-137 source is incident on a rod of either aluminum or an "unknown" material (the material composition is not necessary for this section). By varying the detector angle θ, the photopeak corresponding to incident γ-rays with $E = 661.7\ \mathrm{keV}$ will shift. After fitting a Gaussian to this peak, the centroid is converted from channels to energy, and thus provides the shifted photon energy E'. Since photons are scattering from electrons in the rods, fitting this data to the Compton scattering equation will yield the mass of an electron. For convenience, the mass is calculated using units of MeV/c^2.

Section 4: Differential Cross Section

The theoretical differential cross section is given by the following formula:

$$\frac{d\sigma}{d\Omega} = \frac{r^2}{2}\left(\frac{1+\cos^2\theta}{1+\alpha(1-\cos\theta)^2}\right)\left(1+\frac{\alpha^2(1-\cos\theta)^2}{(1+\cos^2\theta)(1+\alpha(1-\cos\theta))}\right)$$

The classical radius of the scattering target is r and the coefficient $\alpha = \frac{E_\gamma}{mc^2}$, where E_γ is the photopeak energy of Cs-137 $(661.7\mathrm{keV})$. The experimental differential cross section is calculated as follows:

$$\frac{d\sigma}{d\Omega} = \frac{\Sigma'_\gamma}{n_\theta I \Delta\Omega\epsilon}$$

Σ'_γ is the integral of the shifted photopeak, which was calculated using the same method outlined in Section 1. ϵ is the detector efficiency, which is a function of energy: $\epsilon = -0.5E(MeV) + 0.58$ and is accurate to the hundreth. n_θ is the number of electron scatterers in the material and is given by $n_\theta = \rho V N_A \Sigma w_i \frac{Z_i}{M_i}$, where ρ is the material density, V is the scattering volume, N_A is Avogadro's number w_i is the weight factor of an element, Z_i is its atomic number, and M_i is its molar mass. $\Delta\Omega$ is the solid angle of the detector, given by $\Delta\Omega = \frac{\pi(D/2)^2}{R_2^2}$, where D is the diameter of the scintillator (=5.08 cm) and R_2 is the distance from the center of the rod to the detector. I is the photon flux at the rod, and there are two approaches to calculating it. Approach 1 uses the following equation:

$$I_1 = \frac{A_0 f}{4\pi R_1^2}$$

Here, $A_0 = 10\ \mathrm{mCi}$ is the activity of the Cs-137 source, $f = 0.841$ is the fraction of decays that result in a photon with energy $661.7\ \mathrm{keV}$, and $R_1 = 31\ \mathrm{cm}$ is the distance from the source to the center of the rod. Approach 2 assumes that the inverse square law holds and uses the flux present at the detector measured at 0° with no rod to compute the flux at the rod:

$$I_d = \frac{A}{r_d^2}, r_d = R_1 + R_2$$

$$I_r = \frac{A}{r_r^2}, r_r = R_1$$

$$I_r = I_d\frac{r_d^2}{r_r^2} = I_d\frac{(R_1+R_2)^2}{R_1^2}$$

Using both methods, the experimental cross section can be plotted as a function of a normalized calculated cross section by computing the cross section as a function of θ for $r = 1$. The resulting plot is expected to be linear with a slope of r^2. For both approaches, this generates a linear fit that provides the classical radius of an electron.

To identify the material of the second rod, the normalized calculated cross section is multiplied by the newly found radius of the electron. This is plotted against a manipulated experimental cross section, which is simply equal to the differential cross section times n_θ. When plotted against the calculated theoretical cross section, the plot will again be linear; however, the slope will now be equal to n_θ. This is calculated for both approaches, then compared to several computed n_θ values for common machining materials, namely pure aluminum, pure copper, 6061 aluminum, 360 brass, and 304 stainless steel.

Analysis

Uncertainty

All counts were taken to have uncertainty of $\sqrt{N}$, where N is the number of counts. This was found from the following link: http://www.sprawls.org/ppmi2/STATS/ (http://www.sprawls.org/ppmi2/STATS/)

Channels were assumed to have no error.

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import odr
from scipy import signal
from scipy.special import erf
```

In [3]:
```
def red_chi_sq(data, fit, unc, p):
    #Computes reduced chi squared for a data set with p parameters
    chi = 0
    for i in range(len(data)):
        chi += ((data[i]-fit[i])/unc[i])**2
    return chi / (len(data)-p)

def slope(x,y):
    return (y[-1]-y[0])/(x[-1]-x[0])

def odr_fit(odr_data, odr_model, param_guess):
    fit = odr.ODR(odr_data, odr_model, beta0=param_guess)
    fit.run()
    while fit.output.info == 4:
        fit.restart(100)
    return fit

def gaussian(C,x):
    return C[0]*np.exp(-1/2*((x-C[1])/C[2])**2)

def origin_line(C,x):
    return C[0]*x

def power_eq(C,x):
    return C[0]*x**C[1]

def attenuation(C,x):
    return np.exp(-C[0]*x)

def compton(C,x):
    #Note: C[0] will have units of eV/c^2
    E = 661.7*10**3 #Cs-137 energy, eV
    return E/(1+(E/C[0])*(1-np.cos(x)))
```

Section 0: Calibration

In order to calibrate the MCA, we first measured three background measurements for each source (three facing the wall, three facing the Cs-137 source). These were averaged and subtracted from all future measurements. The spectra of Cs-137 and Ba-133 were obtained using the scintillation detector. All scans lasted for 120 seconds.

In [4]:
```
#Section 0: Channel calibration routine

figs0 = 2

header = 12
footer = 14

channels = np.arange(0,1024)

bk_cs1 = np.genfromtxt('background_facing1.Spe', delimiter='\n', skip_header=h
eader, skip_footer=footer)
bk_cs2 = np.genfromtxt('background_facing2.Spe', delimiter='\n', skip_header=h
eader, skip_footer=footer)
bk_cs3 = np.genfromtxt('background_facing3.Spe', delimiter='\n', skip_header=h
eader, skip_footer=footer)

bk_ba1 = np.genfromtxt('background_wall1.Spe', delimiter='\n', skip_header=hea
der, skip_footer=footer)
bk_ba2 = np.genfromtxt('background_wall2.Spe', delimiter='\n', skip_header=hea
der, skip_footer=footer)
bk_ba3 = np.genfromtxt('background_wall3.Spe', delimiter='\n', skip_header=hea
der, skip_footer=footer)

bk_cs = (bk_cs1+bk_cs2+bk_cs3)/3 #Average background for Cs-137
bk_ba = (bk_ba1+bk_ba2+bk_ba3)/3 #Average background for Ba-133
bk_cs_err = 1/3*(np.sqrt(bk_cs1)+np.sqrt(bk_cs2)+np.sqrt(bk_cs3))
bk_ba_err = 1/3*(np.sqrt(bk_ba1)+np.sqrt(bk_ba2)+np.sqrt(bk_ba3))

cs_meas = np.genfromtxt('cs-137_calibration.Spe', delimiter='\n', skip_header=
header, skip_footer=footer)
ba_meas = np.genfromtxt('ba-133_calibration18.5cm.Spe', delimiter='\n', skip_h
eader=header, skip_footer=footer)

cs_meas_err = np.sqrt(cs_meas)
ba_meas_err = np.sqrt(ba_meas)

cs_cal = cs_meas-bk_cs
cs_cal_err = cs_meas_err - bk_cs_err
ba_cal = ba_meas-bk_cs
ba_cal_err = ba_meas_err - bk_ba_err

figs0+=1
plt.plot(channels, cs_cal)
plt.title('Cs-137 Spectrum')
plt.xlabel('Channel Number\n\nFigure {}. Calibration spectrum of Cs-137.'.form
at(figs0))
plt.ylabel('Counts')
plt.show()

figs0+=1
plt.plot(channels, ba_cal)
plt.title('Ba-133 Spectrum')
plt.xlabel('Channel Number\n\nFigure {}. Calibration spectrum of Ba-133.'.form
at(figs0))
plt.ylabel('Counts')
plt.show()
```

```
cals = [cs_cal, ba_cal]
cals_err = [cs_cal_err, ba_cal_err]

peak_cents = []
peak_cents_err = []

gauss_model = odr.Model(gaussian)

for i in range(1):
    #Fits largest photopeak for Cs-137 measurement
    cent = np.where(cs_cal == max(cs_cal))[0][0]
    wid = signal.peak_widths(cs_cal, [cent])[0][0]
    amp = max(cs_cal)
    l = int(cent-wid*1.5)
    r = int(cent+wid*1.5)
    data = odr.RealData(x=channels[l:r], y=cs_cal[l:r], sy=cs_cal_err[l:r])
    fit = odr_fit(data, gauss_model, [amp, cent, wid])
    peak_cents.append(fit.output.beta[1])
    peak_cents_err.append(fit.output.sd_beta[1])

channels_dummy = np.copy(channels)
ba_cal_dummy = np.copy(ba_cal)
ba_cal_dummy_err = np.copy(ba_cal_err)

for i in range(3):
    #Fits 3 largest photopeaks for Ba-133 measurement
    cent = np.where(ba_cal_dummy == max(ba_cal_dummy))[0][0]
    wid = signal.peak_widths(ba_cal_dummy, [cent])[0][0]
    amp = max(ba_cal_dummy)
    l = int(cent-wid*1.5)
    r = int(cent+wid*1.5)
    data = odr.RealData(x=channels_dummy[l:r], y=ba_cal_dummy[l:r], sy=ba_cal_
dummy_err[l:r])
    fit = odr_fit(data, gauss_model, [amp, cent, wid])
    peak_cents.append(fit.output.beta[1])
    peak_cents_err.append(fit.output.sd_beta[1])
    ba_cal_dummy = np.copy(ba_cal_dummy[r:])
    ba_cal_dummy_err = np.copy(ba_cal_dummy_err[r:])
    channels_dummy = np.copy(channels_dummy[r:])


gamma_energies = [661700., 30850., 303000., 356000.]
energy_cal_data = odr.RealData(x=peak_cents, sx=peak_cents_err, y=gamma_energi
es)
oline_model = odr.Model(origin_line)
energy_cal = odr_fit(energy_cal_data, oline_model, [slope(peak_cents, gamma_en
ergies)])
energy_cal.output.pprint()
```

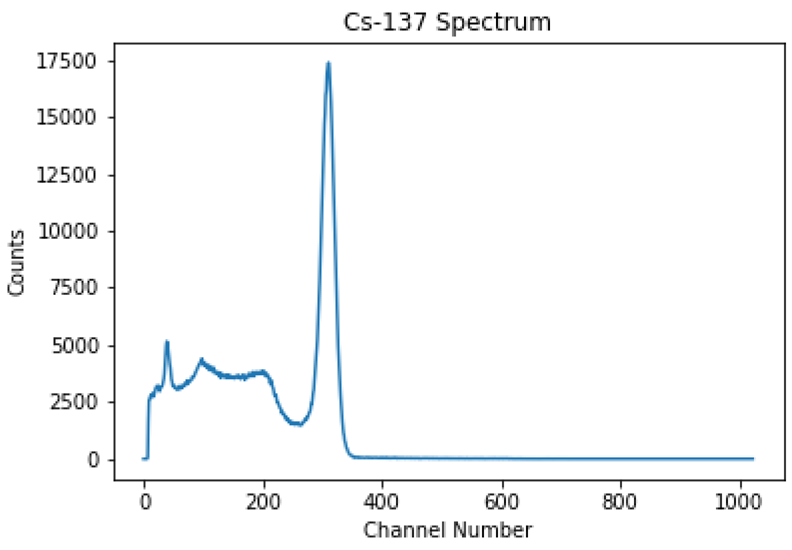

Figure 3. Calibration spectrum of Cs-137.

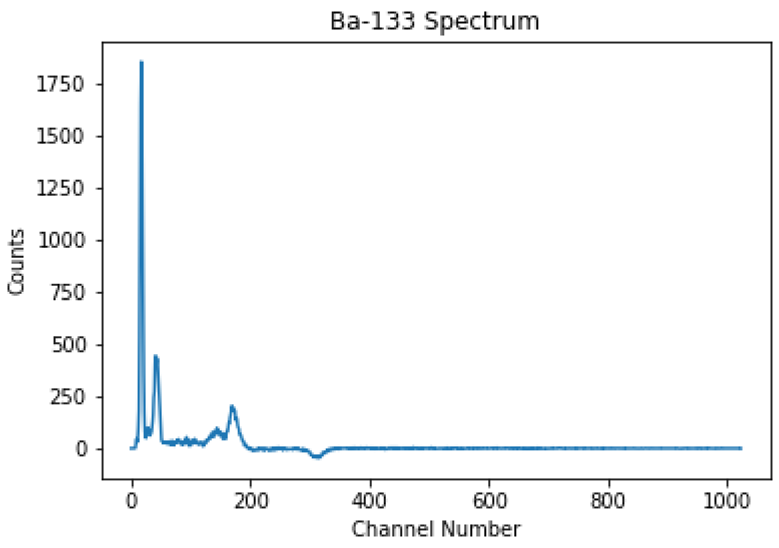

Figure 4. Calibration spectrum of Ba-133.

```
Beta: [2141.03213083]
Beta Std Error: [19.70179602]
Beta Covariance: [[4.93951311]]
Residual Variance: 78.58279912872557
Inverse Condition #: 1.0000000000000002
Reason(s) for Halting:
  Sum of squares convergence
```

Figures 3 and 4 show the spectra used to calibrate the MCA, after subtracting out the averaged background. The energy for each channel is given by the following:

$$E(c) = (2141.0 \pm 19.7)c$$

Though the uncertainty in this coefficient is low (relative error $\frac{\delta a}{a} = 0.0092$), it could be further improved by using additional peaks with known energies. However, in later sections of this lab, it becomes clear that this error is acceptable.

Section 1: Inverse Square Law

For this section, the detector was oriented at a 90° angle relative to the opening of the Cs-137 shielding to minimize background. All distances were measured with a ruler and thus have uncertainty $\pm 0.5\ \mathrm{mm}$.

To compute the error in the integral of each photopeak, the multivariable error approach is taken. Due to the complicated nature of the analytical form of the integral, the exact form of the error is omitted; however, it is calculated within the Gaussian area function.

In [5]:
```
# Section 1
figs = figs0

dist = [7.0, 8.5, 11.0, 12.0, 16.0, 31.0]
ba_inv_meas = []
for d in dist:
    ba_inv_meas.append(np.genfromtxt('ba-133_{}cm.Spe'.format(d), delimiter='
\n', skip_header=header, skip_footer=footer)-bk_ba)

dist.append(18.5)
dist= np.array(dist)/100
ba_inv_meas.append(ba_cal)
ba_inv_meas = np.array(ba_inv_meas)

def gaussian_area(fit, l,r):
    #Computes area and error in area, given a Gaussian fit and bounds of integ
ration
    params = fit.output.beta
    params_err = fit.output.sd_beta
    A = params[0]
    mu = params[1]
    sig = params[2]
    sA = params_err[0]
    smu = params_err[1]
    ssig = params_err[2]
    z1 = (mu-r)/(np.sqrt(2)*sig)
    z2 = (mu-l)/(np.sqrt(2)*sig)
    area = -np.sqrt(np.pi/2)*A*sig*(erf(z1)-erf(z2))
    #print(area)
    dsdA = -np.sqrt(np.pi/2)*sig*(erf(z1)-erf(z2))
    dsdsig = -np.sqrt(np.pi/2)*A*((erf(z1)-erf(z2)) - (z1*np.exp(-z1**2)-z2*np
.exp(-z2**2)))
    dsdmu = -np.sqrt(np.pi/2)*A*(1/np.sqrt(2)*(np.exp(-z1**2)-np.exp(-z2**2)))
    area_err = np.sqrt((dsdA*sA)**2+(dsdsig*ssig)**2+(dsdmu*smu)**2)
    return area, area_err

def max_peak_fit(xdata, ydata):
    #Finds maximum peak and does an odr fit
    cent = np.where(ydata==max(ydata))[0][0]
    height = max(ydata)
    out = signal.peak_widths(ydata, [cent])
    width = out[0][0]
    mult = 1.5
    l = int(max(0, cent-width*mult))
    r = int(min(len(xdata), cent+width*mult))
    #plt.plot(xdata[l:r], ydata[l:r])
    #plt.show()
    data = odr.RealData(x=xdata[l:r], y=ydata[l:r], sy=np.sqrt(ydata[l:r]))
    model = odr.Model(gaussian)
    fit = odr_fit(data, model,[height, cent, width])
    area, area_err = gaussian_area(fit,l,r)
    return fit, area, area_err

dist_err = np.ones_like(dist)*0.5/1000
heights = []
heights_err = []
```

```
areas = []
areas_err = []

for meas in ba_inv_meas:
    fit, a, ae= max_peak_fit(channels, meas)
    heights.append(fit.output.beta[0])
    heights_err.append(fit.output.sd_beta[0])
    areas.append(a)
    areas_err.append(ae)

inv_data = odr.RealData(x=dist, sx=dist_err, y=heights, sy=heights_err)
inv_model = odr.Model(power_eq)
inv_fit = odr_fit(inv_data, inv_model, [heights[0], -2])
inv_fit_chi = red_chi_sq(heights, power_eq(inv_fit.output.beta, dist), heights
_err, 2)

area_data = odr.RealData(x=dist, sx=dist_err, y=areas, sy=areas_err)
area_fit = odr_fit(area_data, inv_model, [areas[0], -2])
area_fit_chi = red_chi_sq(areas, power_eq(area_fit.output.beta, dist), areas_e
rr, 2)

figs += 1
plt.errorbar(x=dist, xerr=dist_err, y=heights, yerr=heights_err, fmt=',k', lab
el='Data')
plt.plot(sorted(dist), power_eq(inv_fit.output.beta, sorted(dist)), label='Fi
t, $n={:.2f} \pm {:.2f}$, $\chi^2={:.2f}$'\
         .format(inv_fit.output.beta[1], inv_fit.output.sd_beta[1], inv_fit_ch
i))
plt.title('Height Fit')
plt.xlabel('Distance (m)\n\nFigure {}. Inverse square law fit using peak heigh
ts.'.format(figs))
plt.ylabel('Peak Amplitude (counts)')
plt.legend()
plt.show()

figs += 1
plt.errorbar(x=dist, xerr=dist_err, y=areas, yerr=areas_err, fmt=',k', label=
'Data')
plt.plot(sorted(dist), power_eq(area_fit.output.beta, sorted(dist)), label='Fi
t, $n={:.2f} \pm {:.2f}$, $\chi^2={:.2f}$'\
         .format(area_fit.output.beta[1], area_fit.output.sd_beta[1],area_fit_
chi))
plt.title('Area Fit')
plt.xlabel('Distance (m)\n\nFigure {}. Inverse square law fit using peak area
s.'.format(figs))
plt.ylabel('Peak Area (counts)')
plt.legend()
plt.show()
```

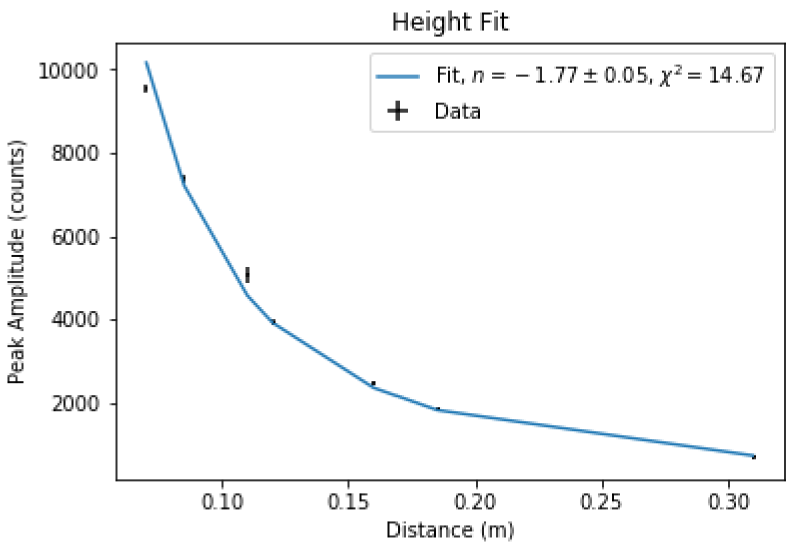

Figure 5. Inverse square law fit using peak heights.

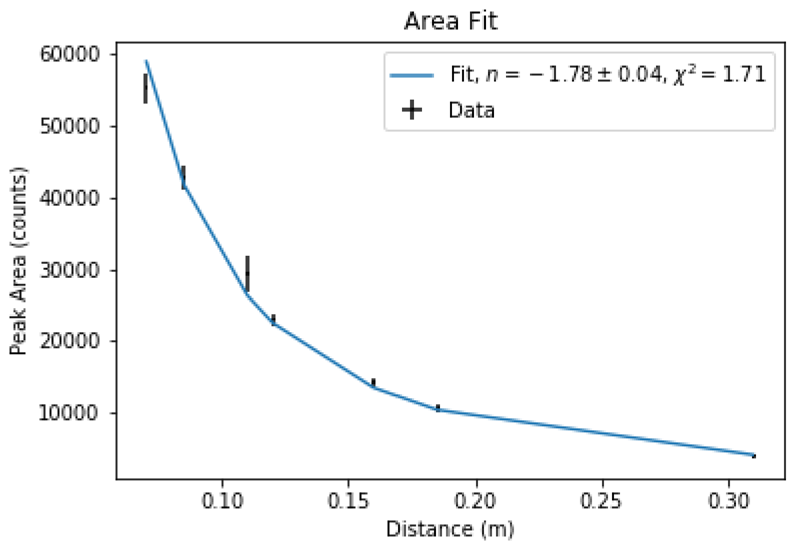

Figure 6. Inverse square law fit using peak areas.

Both approaches are in good agreement, with Figure 5's height fit producing an $n_h = -1.77 \pm 0.05, \tilde{\chi}^2 = 14.67$ and the area approach of Figure 6 yielding $n_a = -1.78 \pm 0.04, \tilde{\chi}^2 = 1.71$. Although the actual value for the inverse square law $n = -2$ is not within the uncertainty of either fit, the fit is still very close to theoretical predictions. The reduced chi squared values both support the quality of the fits as well. To improve on these measurements, it may be worth investigating decays at larger distances, as there is a significant measurement gap between $r = 0.16m$ and $r = 0.31m$. Additionally, further isolating the Ba-133 from the nearby Cs-137 could lead to a reduced background and thus cleaner data. Comparatively, the area fit produced a reduced chi squared that was an order of magnitude smaller than the height fit, so it seems that calculating the area of the peaks provides a more accurate representation of the inverse square behavior of the photopeaks (though both are clearly adequate for demonstrating this effect).

Section 2: Attenuation

First, a measurement of the Cs-137 spectrum was taken without any copper plates between the detector and the source. Then, after blocking the source with the lead brick, a sheet of copper was added to the holder and placed halfway between the source and the detector. In each trial, the brick was removed and data acquisition started once the brick rested on the table. Once the brick was put back in place, another sheet was added and this procedure was repeated.

The thickness of the copper sheets was assumed to be the same for each sheet, and the error on the digital calipers used to measure thickness was one-twentieth of a millimeter.

In [6]:
```
# Section 2

figs2 = figs

thickness = 0.00165
sheets = np.arange(0,10,1)
x_trans = sheets*thickness
x_trans_err = 1/20/1000*np.ones_like(x_trans)
trans_meas = []

for s in sheets:
    if s > 2 or s == 0:
        trans_meas.append(np.genfromtxt('cs-137_sheets{}.Spe'.format(s), delim
iter='\n', skip_header=header, skip_footer=footer)-bk_cs)
    elif s > 0:
        trans_meas.append(np.genfromtxt('cs-137_sheets{}v2.Spe'.format(s), del
imiter='\n', skip_header=header, skip_footer=footer)-bk_cs)

trans_meas = np.array(trans_meas)

I = []
I_err = []

for i in range(len(trans_meas)):
    fit,_,_ = max_peak_fit(channels, trans_meas[i])
    I.append(fit.output.beta[0])
    I_err.append(fit.output.sd_beta[0])

I = np.array(I)
T = I/I[0]
I_err = np.array(I_err)
T_err = np.sqrt((I/(I[0]**2)*I_err[0])**2+(1/I[0]*I_err)**2)


att_data = odr.RealData(x=x_trans, sx=x_trans_err, y=T, sy=T_err)
att_model = odr.Model(attenuation)
att_fit = odr_fit(att_data, att_model,[slope(x_trans, np.log(T))])
#att_fit.output.pprint()
att_chi = red_chi_sq(T, attenuation(att_fit.output.beta, x_trans), T_err ,1)

figs2 += 1
plt.errorbar(x=x_trans, xerr=x_trans_err, y=T, yerr =T_err, fmt='ok', label='D
ata')
plt.plot(x_trans, attenuation(att_fit.output.beta, x_trans), label='Fit, $\mu
 = {:.2f} \pm {:2f}, \chi^2={:.4f}$'\
         .format(att_fit.output.beta[0], att_fit.output.sd_beta[0],att_chi))
plt.title('Attenuation of Radiation')
plt.xlabel('Thickness (m)\n\nFigure {}. Attenuation of Cs-137 radiation throug
h copper plates.'.format(figs2))
plt.ylabel('Transmission Ratio')
plt.legend()
plt.show()
```

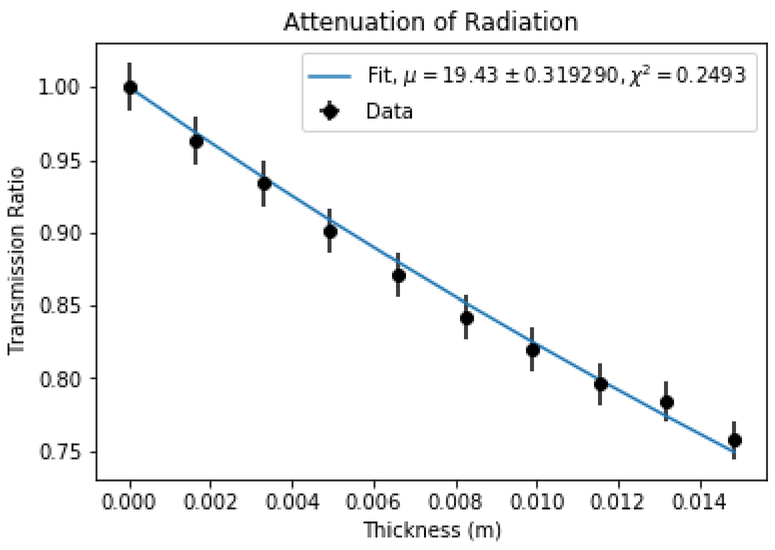

Figure 7. Attenuation of Cs-137 radiation through copper plates.

The data points in Figure 7 were generated by fitting the photopeak corresponding to an energy of $661.7\ \mathrm{keV}$, and thus the attenuation coefficient for copper at this energy is found to be $\mu = 19.43 \pm 0.32 \mathrm{m}^{-1}$. NIST lists the attenuation coefficient per density for $E = 600\ \mathrm{keV}$ in copper to be $\mu/\rho = 0.07625\ \mathrm{cm}^2/\mathrm{g}$ (https://physics.nist.gov/PhysRefData/XrayMassCoef/ElemTab/z29.html (https://physics.nist.gov/PhysRefData/XrayMassCoef/ElemTab/z29.html)); when multiplied by the density of copper $\rho_{Cu} = 8.96\ \mathrm{g\ cm}^{-3}$, we find $\mu = 0.6832 \mathrm{cm}^{-1} = 68.32\ \mathrm{m}^{-1}$. Thus, the measured μ is within the correct order of magnitude. Note that this literature comparison is between values for different photon energy, and a better comparison could be made with either an analytical form of μ/ρ or a computational fit using additional data from NIST.

Potential sources of error arise from the stacking method of the plates, which involved leaning the plates against each other and caused minor slanting and thus minor thickness differences, and material imperfections. Furthermore, the reduced chi squared value $\tilde{\chi}^2 = 0.25$ suggests that error was overestimated. To reduce error, longer scans should be taken, as longer scans observe more events and thus have lower relative error.

Section 3: Compton Scattering

For each rod, scattering measurements were taken at angles ranging from 20° to 80°, in increments of 10°. The error in the angle was equal to one degree. The analysis involved fitting a Gaussian to the shifted photopeaks, then converting its centroid to an energy value. These energies were then correlated with the Compton scattering equation and an ODR fit provided the mass of the electron in eV/c^2. Error in shifted energies were calculated using the error found in Section 0.

In [7]:
```
#Section 3

figs3 = figs2

def channel_to_ev(c, f):
    en = f.output.beta[0]*c
    en_err = f.output.sd_beta[0]*c
    return en, en_err

angles = np.arange(20, 90, 10)
energies, energies_err = channel_to_ev(channels, energy_cal)

al_counts = []
al_counts_err = []
al_fits = []
en_prime_al = []
en_prime_al_err = []

un_counts = []
un_counts_err = []
un_fits = []
en_prime_un = []
en_prime_un_err = []

for a in angles:
    back = np.genfromtxt('background_{}.Spe'.format(a), delimiter='\n', skip_h
eader=header, skip_footer=footer)
    meas = np.genfromtxt('cs-137_al{}.Spe'.format(a), delimiter='\n', skip_hea
der=header, skip_footer=footer)
    counts = meas-back
    counts_err = np.sqrt(meas)-np.sqrt(back)
    al_counts.append(counts)
    al_counts_err.append(counts_err)
    cent = np.where(counts == max(counts))[0][0]
    amp = max(counts)
    wid = signal.peak_widths(counts, [cent])[0][0]
    l = int(cent-wid)
    r = int(cent+wid)
    data = odr.RealData(x=channels[l:r], y=counts[l:r], sy=counts_err[l:r])
    fit = odr_fit(data, gauss_model, [amp,cent,wid])
    al_fits.append(fit)
    en_prime_al.append(fit.output.beta[1]*energy_cal.output.beta[0])
    en_prime_al_err.append(np.sqrt((energy_cal.output.beta[0]*fit.output.sd_be
ta[1])**2 \
                                    +(fit.output.beta[1]*energy_cal.output.sd_beta
[0])**2))

for a in angles:
    back = np.genfromtxt('background_{}.Spe'.format(a), delimiter='\n', skip_h
eader=header, skip_footer=footer)
    meas = np.genfromtxt('cs-137_cu{}.Spe'.format(a), delimiter='\n', skip_hea
der=header, skip_footer=footer)
    counts = meas-back
    counts_err = np.sqrt(meas)-np.sqrt(back)
    un_counts.append(counts)
    un_counts_err.append(counts_err)
    cent = np.where(counts == max(counts))[0][0]
    amp = max(counts)
    wid = signal.peak_widths(counts, [cent])[0][0]
    l = int(cent-wid)
    r = int(cent+wid)
    data = odr.RealData(x=channels[l:r], y=counts[l:r], sy=counts_err[l:r])
    fit = odr_fit(data, gauss_model, [amp,cent,wid])
    un_fits.append(fit)
    en_prime_un.append(fit.output.beta[1]*energy_cal.output.beta[0])
    en_prime_un_err.append(np.sqrt((energy_cal.output.beta[0]*fit.output.sd_be
ta[1])**2 \
                                +(fit.output.beta[1]*energy_cal.output.sd_beta
[0])**2))

angles = angles*np.pi/180
angles_err = np.ones_like(angles)*np.pi/180
en_prime_al = np.array(en_prime_al)
en_prime_al_err = np.array(en_prime_al_err)
en_prime_un = np.array(en_prime_un)
en_prime_un_err = np.array(en_prime_un_err)

compton_model = odr.Model(compton)

al_data = odr.RealData(x=angles, sx=angles_err, y=en_prime_al, sy=en_prime_al_
err)
al_fit = odr_fit(al_data, compton_model, [.511*10**6])
al_chi = red_chi_sq(en_prime_al, compton(al_fit.output.beta, angles), en_prime
_al_err, 1)

un_data = odr.RealData(x=angles, sx=angles_err, y=en_prime_un, sy=en_prime_un_
err)
un_fit = odr_fit(un_data, compton_model, [.511*10**6])
un_chi = red_chi_sq(en_prime_un, compton(un_fit.output.beta, angles), en_prime
_un_err, 1)

figs3 += 1
plt.errorbar(x=angles, xerr=angles_err, y=en_prime_al, yerr=en_prime_al_err, f
mt=',k', label='Data')
plt.plot(angles, compton(al_fit.output.beta, angles), label='Fit, $\chi^2={:.2
f}$'.format(al_chi))
plt.title('Compton Scattering for Aluminum Rod, $m_e = {:.3f} \pm {:.3f} MeV/c
^2$'\
          .format(al_fit.output.beta[0]*10**-6, al_fit.output.sd_beta[0]*10**-
6))
plt.xlabel('Scattering Angle (rad)'+\
           '\n\nFigure {}. Scattered photopeak energy as a function of the sca
ttering angle for aluminum.'.format(figs3))
plt.ylabel('Photopeak Energy (eV)')
plt.legend()
plt.show()

figs3 += 1
plt.errorbar(x=angles, xerr=angles_err, y=en_prime_un, yerr=en_prime_un_err, f
mt=',k', label='Data')
plt.plot(angles, compton(un_fit.output.beta, angles), label='Fit, $\chi^2={:.2
f}$'.format(un_chi))
plt.title('Compton Scattering for Unknown Rod, $m_e = {:.3f} \pm {:.3f} MeV/c^
2$'\
          .format(un_fit.output.beta[0]*10**-6, un_fit.output.sd_beta[0]*10**-
6))
plt.xlabel('Scattering Angle (rad)'+\
           '\n\nFigure {}. Scattered photopeak energy as a function of the sca
ttering angle for unknown material.'.format(figs3))
plt.ylabel('Photopeak Energy (eV)')
plt.legend()
plt.show()


global m_e #Used in Section 4, m_e is chosen as the closest calculated value t
o the literature value
m_e = un_fit.output.beta[0]
m_e_err = un_fit.output.sd_beta[0]
```

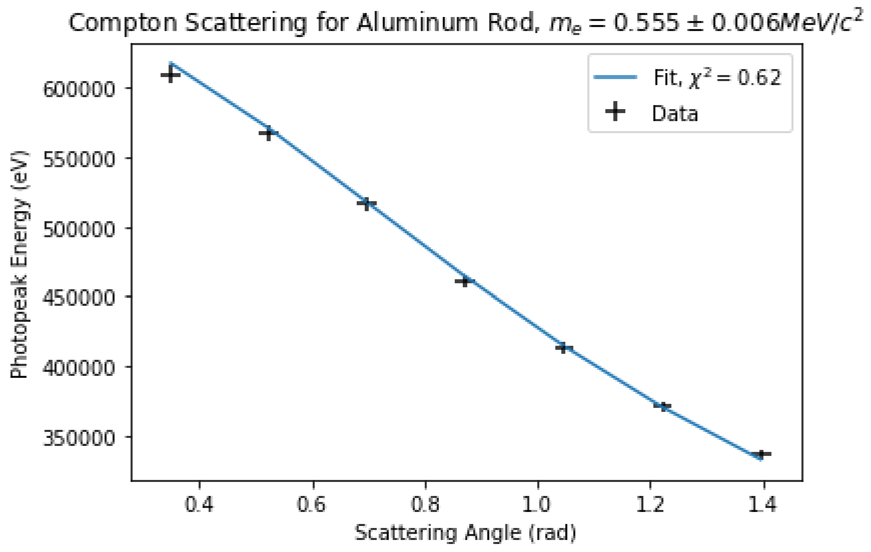

Figure 8. Scattered photopeak energy as a function of the scattering angle for aluminum.

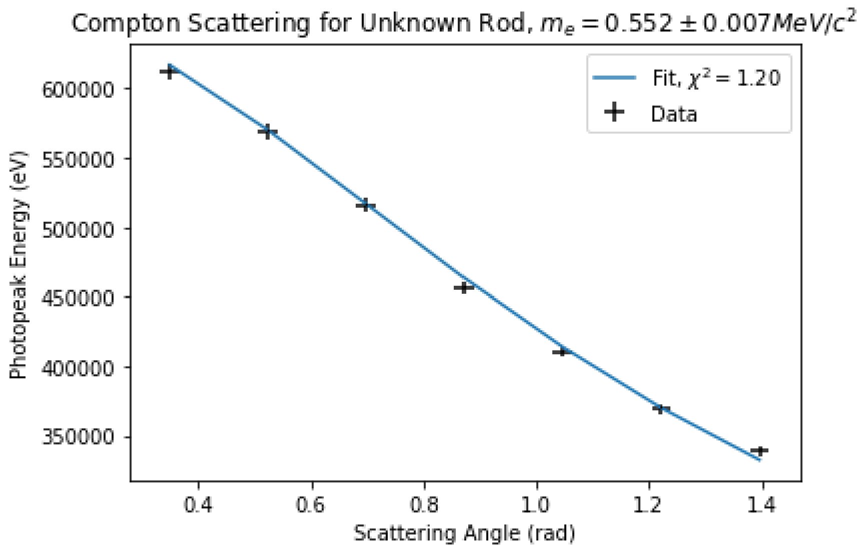

Figure 9. Scattered photopeak energy as a function of the scattering angle for unknown material.

The aluminum data in Figure 8 yields an electron mass $m_e = 0.555 \pm 0.006 \text{ MeV}/c^2$, and the unknown rod data in Figure 9 yields $m_e = 0.552 \pm 0.007 \text{ MeV}/c^2$. The textbook value for $m_e = 0.551 \text{ MeV}/c^2$, so both data sets correctly produce the mass of an electron within experimental uncertainty. Additionally, both fits have an excellent $\tilde{\chi}^2$, equal to 0.62 and 1.20 respectively. Because the unknown rod produced the closest measurement and had the best fit, I chose to use its resulting electron mass for Section 4.

Section 4: Differential Cross Section

The analysis for this section utilizes the same measurements as the previous section.

For the aluminum rod, the linear relationship between the experimental and calculated cross section is as follows:

$$\frac{d\sigma}{d\Omega}_{exp} = r_e^2 \frac{d\sigma'}{d\Omega}_{calc}, \frac{d\sigma'}{d\Omega} = \frac{1}{r_e^2}\frac{d\sigma}{d\Omega}$$

For the unknown rod, the linear relationship between the experimental and calculated cross section is as follows:

$$\frac{d\sigma'}{d\Omega}_{exp} = n_\theta \frac{d\sigma}{d\Omega}_{calc}, \frac{d\sigma'}{d\Omega}_{exp} = n_\theta \frac{d\sigma}{d\Omega}_{exp}$$

The error in the calculated cross section (as a function of θ and m_e) was found using multivariable error analysis, with the partial derivatives calculated using the central difference method. The error in the experimental cross section was found using multivariable error analysis, although its analytical form is much easier to compute:

$$\delta\frac{d\sigma}{d\Omega}_{exp} = \frac{d\sigma}{d\Omega}_{exp}\sqrt{(\frac{\delta\Sigma'_\gamma}{\Sigma'_\gamma})^2 + (\frac{\delta n_\theta}{n_\theta})^2 + (\frac{\delta I}{I})^2 + (\frac{\delta\Delta\Omega}{\Delta\Omega})^2 + (\frac{\delta\epsilon}{\epsilon})^2}$$

For Approach 1, $\delta I_1 = 0$; conversely, Approach 2's error is non-zero because of the uncertainty in the flux at the detector and distance between the detector and the rod.

All Σ'_γ values were computed using the Gaussian area function used in Section 1.

In [8]:
```
#Section 4

figs4 = figs3

eff_al = -0.5*en_prime_al/10**6+.58
eff_al_err = np.sqrt((-0.5*en_prime_al_err/10**6)**2 + (0.01)**2)

eff_un = -0.5*en_prime_un/10**6+.58
eff_un_err = np.sqrt((-0.5*en_prime_un_err/10**6)**2 + (0.01)**2)


r1 = .31
r2 = .242
d  = .0508
t = 120
N_A = 6.022*10**23
A0 = (10*10**-3)*(3.7*10**10)
f = 0.841

sol_ang = np.pi*(d/2)**2/(r2**2)
I1 = (A0*f)/(4*np.pi*r1**2)

al_sigs = []
al_sigs_err = []

al_sigs2 = []
al_sigs_err2 = []

for i in range(len(al_fits)):
    c = al_fits[i].output.beta[1]
    w = al_fits[i].output.beta[2]
    l = int(c-w)
    r = int(c+w)
    ga = gaussian_area(al_fits[i],l,r)
    al_sigs.append(ga[0])
    al_sigs_err.append(ga[1])
    #al_sigs2.append(np.sum(al_counts[i][l:r]))
    #al_sigs_err2.append(np.sum(al_counts_err[i][l:r]))

al_sigs = np.array(al_sigs)
al_sigs_err = np.array(al_sigs_err)

al_den = 146.5/((1.906/2)**2*np.pi*18.75) #g/cm^3
un_den = 479.7/((1.902/2)**2*np.pi*19.6)

al_vol = (1.902/2)**2*np.pi*2 #cm^3
un_vol = (1.902/2)**2*np.pi*2

al_ws = np.array([1])
al_zs = np.array([13])
al_ms = np.array([26.980])

al_n_th = al_den*al_vol*N_A*(np.sum(al_ws*al_zs/al_ms))

al_exp_diff = al_sigs/(al_n_th*I1*sol_ang*t*eff_al)
```

```
def differential_scattering(C,x):
    a = 661700/m_e
    return (C[0]**2/2)*((1+np.cos(x)**2)/((1+a*(1-np.cos(x)))**2))\
                *(1+(a**2*(1-np.cos(x))**2)/((1+np.cos(x)**2)*(1+a*(1-np.cos(x
)))))

def diff_th(C,m):
    a = 661700/m
    return (C[0]**2/2)*((1+np.cos(C[1])**2)/((1+a*(1-np.cos(C[1])))**2))\
                *(1+(a**2*(1-np.cos(C[1]))**2)/((1+np.cos(C[1])**2)*(1+a*(1-np
.cos(C[1])))))

def diff_err(C,th, th_err, m_err):
    dth = 10**-6
    dm = 10**-6
    dfdth = (differential_scattering(C, th+dth) - differential_scattering(C,th
-dth))/(2*dth)
    dfdm = (diff_th([C[0], th], m_e+dm) - diff_th([C[0], th], m_e-dm))/(2*dm)
    return np.sqrt((dfdth*th_err)**2 + (dfdm*m_err)**2)


'''
plt.plot(angles, differential_scattering([2.82*10**-15], angles))
plt.plot(angles, al_exp_diff)
plt.plot(angles, al_exp_diff2)
plt.show()
'''

al_x = differential_scattering([1], angles)
al_x_err = []
for angle in angles:
    al_x_err.append(diff_err([1], angle, angles_err[0], m_e_err))
al_x_err = np.array(al_x_err)

sol_ang_err = 2*sol_ang/r2*.0005
al_den_err = al_den*np.sqrt((0.05/146.5)**2+(2*0.001/20/1.902)**2)
al_n_th_err = al_n_th*np.sqrt((al_den_err/al_den)**2+(2*0.001/20/1.902)**2)
al_exp_diff_err = al_exp_diff*np.sqrt((al_sigs_err/al_sigs)**2 + (al_n_th_err/
al_n_th)**2 +\
                                     (sol_ang_err/sol_ang)**2 + (eff_al_err/ef
f_al)**2)

al_cross_data = odr.RealData(x=al_x, sx=al_x_err, y=al_exp_diff, sy=al_exp_dif
f_err)
al_cross_fit = odr_fit(al_cross_data, oline_model, [(2.82*10**-15)**2])

al_cross_chi1 = red_chi_sq(al_exp_diff, origin_line(al_cross_fit.output.beta,
al_x), al_exp_diff_err, 1)

r_e1 = al_cross_fit.output.beta[0]**0.5
r_e1_err = r_e1*(1/2*al_cross_fit.output.sd_beta[0]/al_cross_fit.output.beta[0
])

cs_0 = np.genfromtxt('cs-137_no_rod.Spe', delimiter='\n', skip_header=header,
skip_footer=footer) - bk_cs
cs_0_fit,cs_area,cs_area_err = max_peak_fit(channels, cs_0)
```

```
I2 = cs_area/(4*np.pi*(d/2)**2*t)*(r1+r2)**2/(r1**2)
I2_err = 1/(4*np.pi*(d/2)**2*t)*np.sqrt(((r1+r2)**2/(r2**2)*cs_area_err)**2+(c
s_area*2*(r1+r2)/(r1**2)*0.0005)**2)

al_exp_diff2 = np.copy(al_exp_diff)*I1/I2
al_exp_diff2_err =al_exp_diff2*np.sqrt((al_sigs_err/al_sigs)**2 + (al_n_th_err
/al_n_th)**2 +\
                                     (sol_ang_err/sol_ang)**2 + (eff_al_err/ef
f_al)**2+(I2_err/I2)**2)

al_cross_data2 = odr.RealData(x=al_x, sx=al_x_err, y=al_exp_diff2, sy=al_exp_d
iff2_err)
al_cross_fit2 = odr_fit(al_cross_data2, oline_model, [(2.82*10**-15)**2])

r_e2 = al_cross_fit2.output.beta[0]**0.5
r_e2_err = r_e2*(1/2*al_cross_fit2.output.sd_beta[0]/al_cross_fit2.output.beta
[0])

al_cross_chi2 = red_chi_sq(al_exp_diff2, origin_line(al_cross_fit2.output.beta
, al_x), al_exp_diff2_err, 1)

figs4 += 1
plt.errorbar(x=al_x, xerr=al_x_err, y=al_exp_diff, yerr=al_exp_diff_err, fmt=
'ok', label='Data')
plt.plot(al_x, np.copy(al_x)*r_e1**2, label='Fit 1, $\chi^2={:.2f}$'.format(al
_cross_chi1))
plt.title('Fit for Flux Method 1:\n'+ r'$r_e = {:.2f}$'.format(r_e1*10**15) \
          +r'$\times 10^{-15} \pm$' + r'${:.2f}$'.format(r_e1_err*10**15) \
          +r'$\times 10^{-15} m$' +'\n')
plt.xlabel(r'Scaled Cross Section $(\frac{1}{r^2} \frac{d\sigma}{d\Omega})$' +
\
          '\n\nFigure {}. Plot of calculated and experimentally measured diffe
rential cross sections using Approach 1.'\
           .format(figs4))
plt.ylabel(r'Measured Cross Section ($m^2$)')
plt.legend()
plt.show()

figs4 += 1
plt.errorbar(x=al_x, xerr=al_x_err, y=al_exp_diff2, yerr=al_exp_diff2_err, fmt
='ok', label='Data')
plt.plot(al_x, np.copy(al_x)*r_e2**2, label='Fit 2, $\chi^2={:.2f}$'.format(al
_cross_chi2))
plt.title('Fit for Flux Method 2:\n'+ r'$r_e = {:.2f}$'.format(r_e2*10**15) \
          +r'$\times 10^{-15} \pm$' + r'${:.2f}$'.format(r_e2_err*10**15) \
          +r'$\times 10^{-15} m$' +'\n')
plt.xlabel(r'Scaled Cross Section $(\frac{1}{r^2} \frac{d\sigma}{d\Omega})$' +
\
          '\n\nFigure {}. Plot of calculated and experimentally measured diffe
rential cross sections using Approach 2.'\
           .format(figs4))
plt.ylabel(r'Measured Cross Section ($m^2$)')
plt.legend()
plt.show()

un_x = np.copy(al_x)*r_e1**2
un_x_err = np.sqrt((r_e1**2*np.copy(al_x_err))**2 + (2*r_e1*np.copy(al_x)*r_e1
_err)**2)

un_sigs = []
un_sigs_err = []
un_sigs2 = []
un_sigs_err2 = []

for i in range(len(un_fits)):
    c = un_fits[i].output.beta[1]
    w = un_fits[i].output.beta[2]
    l = int(c-w)
    r = int(c+w)
    ga = gaussian_area(un_fits[i],l,r)
    un_sigs.append(ga[0])
    un_sigs_err.append(ga[1])
    un_sigs2.append(np.sum(un_counts[i][l:r]))
    un_sigs_err2.append(np.sum(un_counts_err[i][l:r]))

un_sigs = np.array(un_sigs)
un_sigs_err = np.array(un_sigs_err)

un_exp_diff = un_sigs/(I1*sol_ang*t*eff_un)
un_exp_diff_err = un_exp_diff*np.sqrt((un_sigs_err/un_sigs)**2 +\
                                     (sol_ang_err/sol_ang)**2 + (eff_un_err/ef
f_un)**2)

un_exp_diff2 = np.copy(un_exp_diff)*I1/I2
un_exp_diff2_err = un_exp_diff2*np.sqrt((un_sigs_err/un_sigs)**2 +\
                                     (sol_ang_err/sol_ang)**2 + (eff_al_err/ef
f_al)**2+(I2_err/I2)**2)

un_cross_data = odr.RealData(x=un_x, sx=un_x_err, y=un_exp_diff, sy=un_exp_dif
f_err)
un_cross_fit = odr_fit(un_cross_data, oline_model, [1])
un_cross_chi1 = red_chi_sq(un_exp_diff, origin_line(un_cross_fit.output.beta,
un_x), un_exp_diff_err, 1)

un_n_th1 = un_cross_fit.output.beta[0]
un_n_th1_err = un_cross_fit.output.sd_beta[0]

un_cross_data2 = odr.RealData(x=un_x, sx=un_x_err, y=un_exp_diff2, sy=un_exp_d
iff2_err)
un_cross_fit2 = odr_fit(un_cross_data2, oline_model, [un_fit.output.beta[0]])
un_cross_chi2 = red_chi_sq(un_exp_diff2, origin_line(un_cross_fit2.output.beta
, un_x), un_exp_diff2_err, 1)

un_n_th2= un_cross_fit2.output.beta[0]
un_n_th2_err = un_cross_fit2.output.sd_beta[0]

figs4 += 1
plt.errorbar(x=un_x, xerr=un_x_err, y=un_exp_diff, yerr=un_exp_diff_err, fmt=
'ok', label='Data')
plt.plot(un_x, origin_line(un_cross_fit.output.beta, un_x), label='Fit, $\chi^
2 = {:.2f}$'.format(un_cross_chi1))
plt.title(r'Fit for $n_\theta$ with Flux Method 1' + '\n' +\
           r'$n_\theta = {:.2f} \times'.format(un_n_th1/10**24) + r'10^{24} \p
m' + r'{:.2f}$'.format(un_n_th1_err/10**24) +\
         r'$\times 10^{24}$')
plt.legend()
plt.xlabel(r'Scaled Cross Section $(n_\theta \frac{d\sigma}{d\Omega})$' +\
          '\n\nFigure {}. Calculated and experimental cross sections, using el
ectron radius from Approach 1.'\
           .format(figs4))
plt.ylabel(r'Scaled Experimental Cross Section $(n_\theta \frac{d\sigma}{d\Ome
ga})$')
plt.show()

figs4 += 1
plt.errorbar(x=un_x, xerr=un_x_err, y=un_exp_diff2, yerr=un_exp_diff2_err, fmt
='ok', label='Data')
plt.plot(un_x, origin_line(un_cross_fit2.output.beta, un_x), label='Fit, $\chi
^2 = {:.2f}$'.format(un_cross_chi2))
plt.title(r'Fit for $n_\theta$ with Flux Method 2' + '\n' +\
           r'$n_\theta = {:.2f} \times'.format(un_n_th2/10**27) + r'10^{27} \p
m' + r'{:.2f}$'.format(un_n_th2_err/10**27) +\
         r'$\times 10^{27}$')
plt.xlabel(r'Scaled Cross Section $(n_\theta \frac{d\sigma}{d\Omega})$' +\
          '\n\nFigure {}. Calculated and experimental cross sections, using el
ectron radius from Approach 2.'\
           .format(figs4))
plt.ylabel(r'Scaled Experimental Cross Section $(n_\theta \frac{d\sigma}{d\Ome
ga})$')
plt.legend()
plt.show()

cu_ws = np.array([1])
cu_zs = np.array([29])
cu_ms = np.array([63.546])
cu_den = 8.96

brass_ws = np.array([.615, .035, .03, .355])
brass_zs = np.array([29, 26, 82, 30])
brass_ms = np.array([63.546, 55.845, 207.2, 65.38])
brass_den = 8.73

stl_ws = np.array([1-.18-.08-.008, .18,.08, .008])
stl_zs = np.array([26, 24, 28, 6])
stl_ms = np.array([55.845, 51.9961, 58.6934, 12.0107])
stl_den = 7.93

al_6061_ws = np.array([1-.012-.008-.007-.004-.0015-.0035-.0025-.0015, .012, .0
08,.007,.005,.0015,.0035,.0025,.0015])
al_6061_zs = np.array([13, 12, 14, 26, 29, 25, 24, 30, 22])
al_6061_ms = np.array([26.980, 24.3050, 28.0855, 55.845, 63.546, 24.3050, 51.9
961, 65.38, 47.867])
al_6061_den = 2.7

cu_n = cu_den*un_vol*N_A*np.sum(cu_ws*cu_zs/cu_ms)
brass_n = brass_den*un_vol*N_A*np.sum(brass_ws*brass_zs/brass_ms)
stl_n = stl_den*un_vol*N_A*np.sum(stl_ws*stl_zs/stl_ms)
al_6061_n = al_6061_den*un_vol*N_A*np.sum(al_6061_ws*al_6061_zs/al_6061_ms)
```

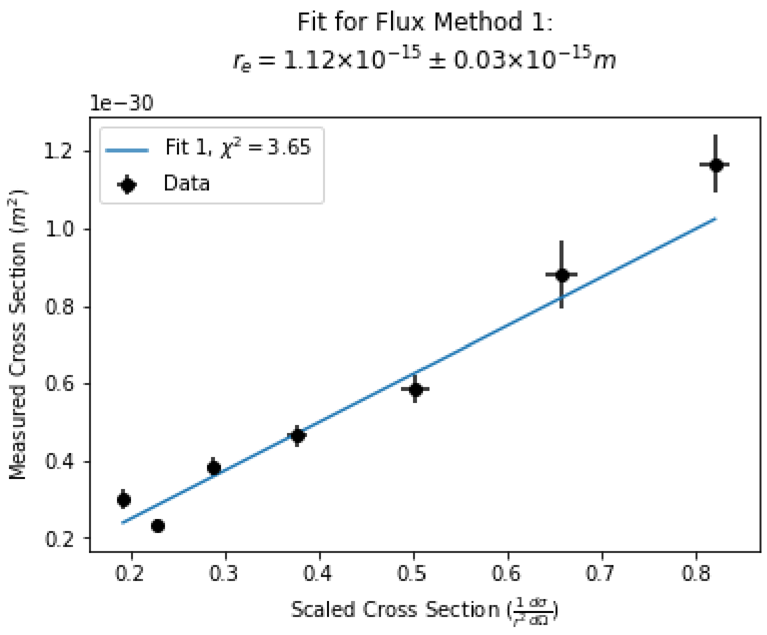

Figure 10. Plot of calculated and experimentally measured differential cross sections using Approach 1.

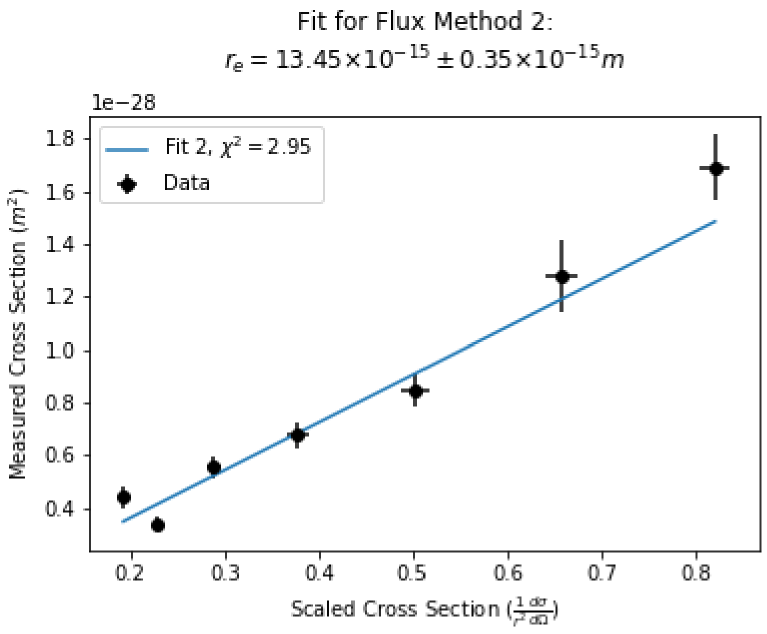

Figure 11. Plot of calculated and experimentally measured differential cross sections using Approach 2.

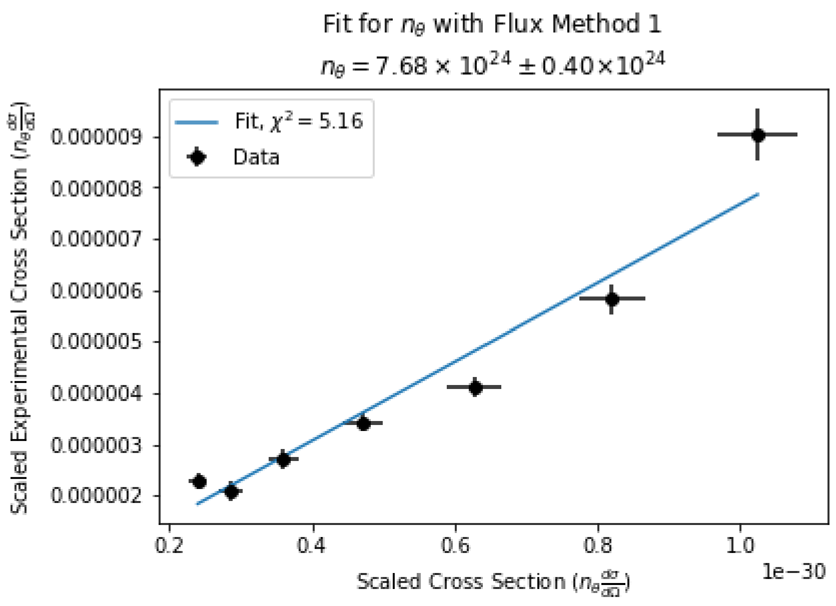

Figure 12. Calculated and experimental cross sections, using electron radius from Approach 1.

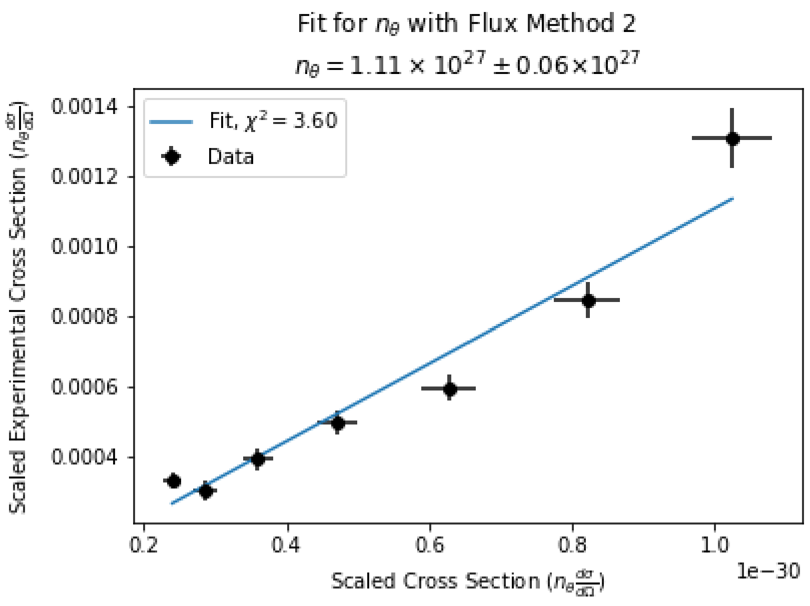

Figure 13. Calculated and experimental cross sections, using electron radius from Approach 2.

Figures 10 and 11 demonstrate a linear fit for finding the square of the classical electron radius and produce $r_e = 1.12 \times 10^{-15} \pm 0.03 \times 10^{-15}\ \mathrm{m}$ and $r_e = 13.45 \times 10^{-15} \pm 0.35 \times 10^{-15}\ \mathrm{m}$ respectively. Both have good $\tilde{\chi}^2$ values (3.65 and 3.49 respectively), meaning that the linear fit model is a good approach for the data. Given that the literature value for the classical radius of an electron is $2.82 \times 10^{-15}\ \mathrm{m}$, it is clear that Approach 1 produces a more accurate result, though both fits are still close to the correct order of magnitude and produce very nearly correct values.

Material	$n_\theta \times 10^{24}$
Unknown (Approach 1)	7.68 ± 0.40
Unknown (Approach 2)	1108.62 ± 50.66
Unknown (Average)	558.15 ± 28.51
Copper (Pure)	13.99
Brass (360)	14.09
Aluminum (Pure)	4.52
Aluminum (6061)	4.46
Stainless Steel (304)	12.64

Figure 14. List of n_θ for unknown material using Approaches 1 and 2, compared to several common machining materials

.

Figures 12 and 13 demonstrate the linear fits for n_θ using approaches 1 and 2 respectively, and Figure 14 compares the results of those fits to proposed values of n_θ for common machining materials. Originally, I expected the rod to be made of copper or bronze, and I leaned towards copper due to the dullness of the rod and its screw threads. Despite the good quality of the fits ($\tilde{\chi}^2 = 5.16, 3.60$ respectively), the data remains inconclusive, though Approach 1 yields a value that is somewhat close to those calculated for brass and copper.

Both approaches suffer from errors caused by differences between the actual and measured electron radius. This is especially prevalent in the second approach, where the radius is about five times larger than the actual value. This issue is exacerbated by the squaring of the radius: if the measured value $r_{e,meas} = r_{e,lit} - \delta r$, the square $r_{e,meas}^2 = r_{e,lit}^2 - 2r_{e,lit}\delta r + (\delta r)^2$, which gets larger as the difference between measured and literature value increases. Thus, this error could be a result of using an incorrect value of r_e, which is supported by the drastic increase in n_θ in Approach 2.

Considering that the unknown rod scattering data provided an excellent fit for Section 3, I do not think that the data is the source of the poor values of n_θ. It may instead suggest that the aluminum measurements should be improved so that they yield an electron radius that is closer to the actual value. Alternatively, both sets of data could be improved by taking longer measurements, as longer measurement times yield better statistics.

Conclusion

Below is a table summarizing important values found during this experiment:

Parameter	Literature	Measured
	Section 1	
$n_{heights}$	-2	-1.77 ± 0.05
n_{areas}	-2	-1.78 ± 0.04
	Section 2	
μ	$68.32\ \mathrm{m}^{-1}$	$19.43 \pm 0.32\ \mathrm{m}^{-1}$
	Section 3	
$m_e(\mathrm{Al})$	$0.511\ \mathrm{MeV}/c^2$	$0.555 \pm 0.006\ \mathrm{MeV}/c^2$
$m_e(\mathrm{Un})$	$0.511\ \mathrm{MeV}/c^2$	$0.552 \pm 0.007\ \mathrm{MeV}/c^2$
	Section 4	
$r_e(\mathrm{Al, Method\ 1})$	$2.82 \times 10^{-15}\ \mathrm{m}$	$1.12 \times 10^{-15} \pm 0.03 \times 10^{-15}\ \mathrm{m}$
$r_e(\mathrm{Al, Method\ 2})$	$2.82 \times 10^{-15}\ \mathrm{m}$	$13.45 \times 10^{-15} \pm 0.35 \times 10^{-15}\ \mathrm{m}$
$n_\theta(\mathrm{Un, Method\ 1})$	N/A	$7.68 \times 10^{24} \pm 0.40 \times 10^{24}$
$n_\theta(\mathrm{Un, Method\ 2})$	N/A	$1.11 \times 10^{27} \pm 0.06 \times 10^{27}$

Figure 15. Comparison of literature and measured values for all sections.

By varying the Ba-133 sample distance, the inverse square law was verified. The attenuation coefficient of copper was found for a given photopeak energy by placing copper sheets between Cs-137 and the detector. By scattering γ-rays from aluminum and (presumably) copper rods and applying the Compton scattering formula, the mass of an electron was found to great accuracy (for an undergraduate). Finally, using the aforementioned scattering measurements, the classical radius of the electron was found.

```
In [ ]:
```