Problem Set 8 problems: Only Coding Part Filled out; written portion on other pdf

Question 1: Drawing Feynmand Diagrams (40 points)

Learning objectives

In this question you will:

- Review the rules for drawing Feynman diagrams for QED processes
- Apply these rules to several processes of interest

The goal of this problem is to make you more comfortable with the language of Feynman diagrams. Although we won'd calculate the matrix elements explictly, we can learn a lot about the cross section by examing these diagrams. In each case below, draw the relevant Feynman diagrams. Label each incoming and outgoing leg with the particle species and momentum (call incoming particle momenta p_1 and p_2 and outgoing particle momenta p_3 and p_4. Using the rules shown on page 124 of Thomson or on page 13 of the notes for Lecture 15, mark each external line, internal propagator and vertex with the appropriate factor. make sure your external lines have arrows indicating whether the line represents a particle or an antiparticle. Thomson Figure 5.7 is an example of what your drawings should look like.

1a.

Draw the lowest order diagram for the elastic scattering process $e^+\mu^+ \to e^+\mu^+$

Write your answer here

1b.

Draw the two lowest order diagrams for the Compton scattering process $\gamma e^- \to \gamma e^-$ (Hint: both t-channel and s-channel propagators are possible)

Write your answer here

1c.

Draw the two lowest order diagrams for the process $e^+e^- \to \gamma\gamma$ (Hint: both t-channel and u-channel diagrams are possible)

Write your answer here

1d.

Draw the lowest order diagram for the process $e^+e^- \to u\overline{u}$ where u is an up quark. (We'll see at the end of the month that the quarks must turn into hadrons before we observe them in our detector. Never the less, quarks are Dirac particles so this is a perfectly reasonable process to calculate)

Write your answer here

Question 2: Invariant Form for Cross Sections (20 points)

Learning objectives

In this question you will:

- Review the results presented in class for the e^+e^- annihilation cross section
- Re-express the matrix element squared in a Lorentz invariant form

In class we discussed how to calculate the Feynman Diagrams and cross section for the process

$$e^+e^- \rightarrow \mu^+\mu^-$$

in the center of mass frame. In the frame the cross section for unpolarized scatttering can be written:

$$\frac{d\sigma}{d\Omega} = \frac{\alpha^2}{4s}\left(1+\cos^2\theta\right)$$

where α is the fine structure constant ($\frac{1}{137}$).

This expression came from multiplying the matrix element squared by a LIPS factor. The matrix element squared was:

$$\left\langle |\mathcal{M}|^2 \right\rangle = e^4\left(1+\cos^2\theta\right)$$

where the $<>$ is short hand for averaging over initial spin states and summing over final spin states. Since the phase space is Lorentz invariant, $\left\langle |\mathcal{M}|^2 \right\rangle$ must be as well to ensure that our final answer is invariant.

Show that we can rewrite the expression above in Lorentz invariant form

$$\left\langle |\mathcal{M}|^2 \right\rangle = 2e^4\left(\frac{t^+u^2}{s^2}\right)$$

where s, t amd u are the Mandelstam variables. Assume tha the masses of all particles can be neglected..

Question 3: Angular Distributions in e+e- --> mu+mu- (40 points)

Learning objectives

In this question you will:

- Study the angular distribution of muons produced in e+e- annihilation
- Demonstrate using data collected by the Babar experiment that the two muons are produced back-to-back in both theta and phi
- Assess quantitatively the accuracy of the calculation of the angular distribution by performing a chisquare calculation

3a.

Consider the process

$$e^+e^- \to \mu^+\mu^-$$

In QED, this occurs through the s-channel production of a virtual photon. If the energy is high enough so that the electron and muon masses are negligible, the matrix element squared can be written as:

$$|\mathcal{M}| = 2e^4\left(\frac{t^2+u^2}{s^2}\right)$$

where e is the electron charge and s, t and u are the Mandelstam variables.

Find the differential cross section $d\sigma/d\Omega$ in the center-of-mass frame (where Ω is $d^2/d\cos\theta d\phi$ for the outgoing μ^-). Express it in terms of the fine structure constant α, the Mandelstam variables and θ, where θ is the angle between the incoming e^- and the outgoing μ^-.

Write your answer here

3b.

If the electron beams are unpolarised, the matrix element cannot depend on ϕ, since the problem is azymuthally symmetric. Therefore $d\sigma/d\Omega$ depends only on $\cos\theta$. Integrate your cross section above over ϕ and then make a plot of the predicted angular distribution $d\sigma/d\cos\theta$.

Write your answer here

3c.

Now integrate over $\cos\theta$ to find the total cross section as function of s in units of nb

Write your answer here

3d.

The file mumu.dat contains data corresponding to an integrated luminosity of 74.674 pb^{-1} collected by the BaBar experiment at the SLAC B-factory. The file contains events from the process $e^+e^- \to \mu^+\mu^-(\gamma)$. (where ($\gamma$) means that the event selection may allow a low-energy photon). The photon emission can be treated as a small radiative correction, which does not modify the gross properties of the $e^+e^- \to \mu^+\mu^-$ process. The data were collected at a center-of-mass energy of10.539 GeV. Note, that this is below the B B̄ production threshold.

The following code reads these data file and puts the data into a form that can be easily used in python. (For people who prefer to use root, a mumu.root file is available in this directory)

In [213]:
```
import math
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# Parse the input file.
file = "mumu.dat"

# Inforamtion to offset needed to nterpret the info
# The order of variables is explained in the metadata at the top of th
e file
# The inital e- direction is the +z direction
#
#  isBCMuMu -- a boolean bit which provides a tighter (but cleaner) se
lection of di-muons.  You should require this bit to be 1
#  p1Mag -- Magnitude of the momentum of the mu- (in GeV)
#  p1CosTheta -- Cosine of thepolar angle of the mu-track
#  p1Phi -- Azimuthal angle of the mu- track
#  p1EmcCandEnergy -- Electromagnetic Calorimeter energy associated wi
th the highest-momentum track. For muons, this is expected to be non-z
ero, but small (<1 GeV)
#  p2Mag -- Magnitude of the momentum of the mu+ track (in GeV)
#  p2CosTheta -- Cosine of thepolar angle of the mu+ track
#  p2Phi -- Azimuthal angle of the mu+ track
#  p2EmcCandEnergy -- Electromagnetic Calorimeter energy associated wi
th the second highest-momentum track. For muons, this is expected to b
e non-zero, but small (<1 GeV)
#
inMeta = False
isBCMuMu = []
p1Mag = []
p1CosTheta = []
p1Phi = []
p1EmcCandEnergy = []
p2Mag = []
p2CosTheta = []
p2Phi = []
p2EmcCandEnergy = []

inMeta = True
for line in open(file,"r"):
    line = line.strip()
    info = line.split(",")
    if inMeta and ("<metadata>" in info[0]):
        inMeta = True
    elif inMeta and ("</metadata>" in info[0]):
        inMeta = False
    elif not inMeta:
        isBCMuMu.append(int(info[0]))
        p1Mag.append(float(info[1]))
        p1CosTheta.append(float(info[2]))
        p1Phi.append(float(info[3]))
        p1EmcCandEnergy.append(float(info[4]))
        p2Mag.append(float(info[5]))
        p2CosTheta.append(float(info[6]))
```

```
        p2Phi.append(float(info[7]))
        p2EmcCandEnergy.append(float(info[8]))
```

First, let's verify our statement that presence of the photon does not grossly effect the kinematics of these events. Make a scatter plot comparing the values of $\cos(\theta)$ for the two muon candidates in each event. Make a second scatter plot comparing the values of ϕ. Explain in words what these plots tell you. Make sure you require that the isBCMuMu bit is set to 1. In addition, to insure that both muons are in a part of the detector where they are well-measured, require $|\cos\theta| < 0.7485$ for each muon.

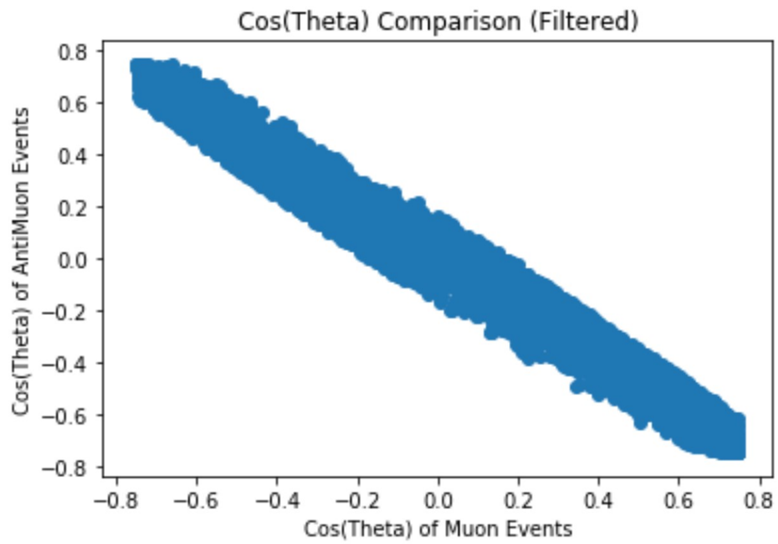
In [214]:

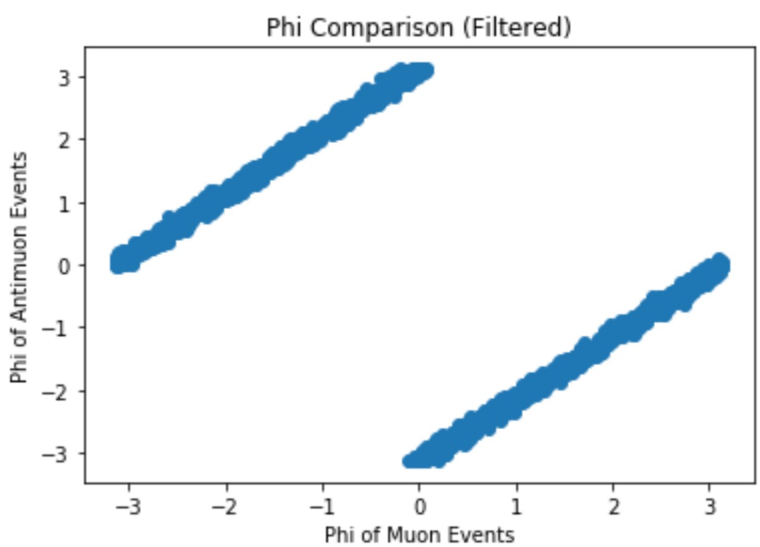
```
# typo change: particle 1 = negative muon, particle 2 = positive muon
(antimuon)
```

In [215]:

```
# require isBCMuMu == 1:

mumu_arr = np.array(isBCMuMu)

ones = np.argwhere(mumu_arr == 1)

filteredp1cos = np.array(p1CosTheta)[ones]
filteredp2cos = np.array(p2CosTheta)[ones]
filteredp1phi = np.array(p1Phi)[ones]
filteredp2phi = np.array(p2Phi)[ones]

# require |cos(theta)| < 0.7485

reqval = 0.7485

# f2 = 2nd filter
p1req = np.where(np.abs(filteredp1cos) < reqval)[0]
f2p1cos = filteredp1cos[p1req]

p2req = np.where(np.abs(filteredp2cos) < reqval)[0]
f2p2cos = filteredp2cos[p2req]
```

In [216]:
```
plt.figure()
plt.title("Cos(Theta) Comparison (Filtered)")
plt.scatter(f2p1cos, f2p2cos)
plt.xlabel("Cos(Theta) of Muon Events")
plt.ylabel("Cos(Theta) of AntiMuon Events")
plt.show()

plt.figure()
plt.title("Phi Comparison (Filtered)")
plt.scatter(filteredp1phi, filteredp2phi)
plt.xlabel("Phi of Muon Events")
plt.ylabel("Phi of Antimuon Events")
plt.show()
```





The $\cos(\theta)$ comparison plot shows that the $\cos(\theta)$'s of the muon and antimuon are negatively correlated. In other words, $\cos(\theta)$ of the antimuon becomes less positive as $\cos(\theta)$ of the muon becomes more positive. The ϕ comparison plot shows that there is positive correlation between the ϕ of the antimuon and muon but in seperate ranges. As ϕ_μ increases from $-\pi$ to 0, $\phi_{\bar{\mu}}$ increases from 0 to π. As ϕ_μ increases from 0 to π, $\phi_{\bar{\mu}}$ increases from $-\pi$ to 0.
These linear correlation plots tell us that the photon does not really affect the kinetmatics of the event because the resulting scattering angles and momenta are equal and opposite. Thus, we can safely assume treat the photon as a small radiative correction as we did.

3e.

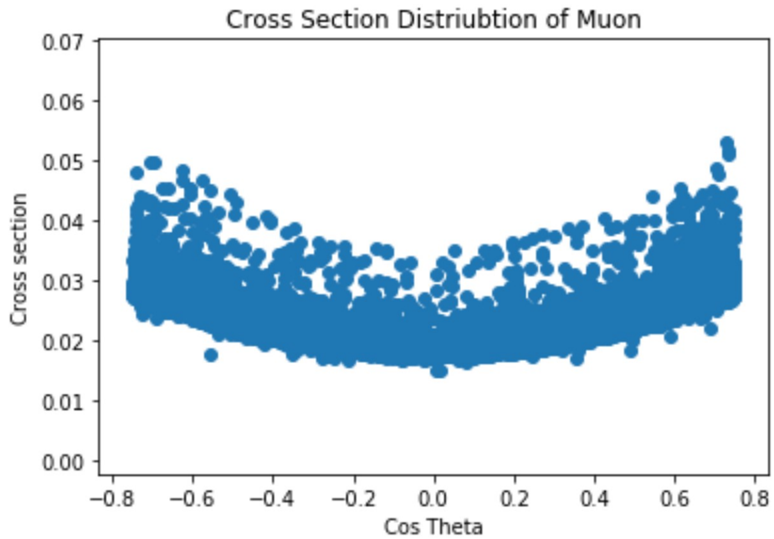
Because the electromagnetic interaction conserves parity, the angular distribution cannot contain any terms that are odd under parity inversion. Weak interactions, however, do not conserve parity. Since the Z boson has the same quantum numbers as the photon, diagrams involving the Z and the photon both contribute to the $e^+e^- \to \mu^+\mu^-$ cross section. The interference between these two diagrams introduces a term that violates parity. Plot the angular distribution for the μ^+ and for the μ^-. Is there evidence of parity violation in these plots?

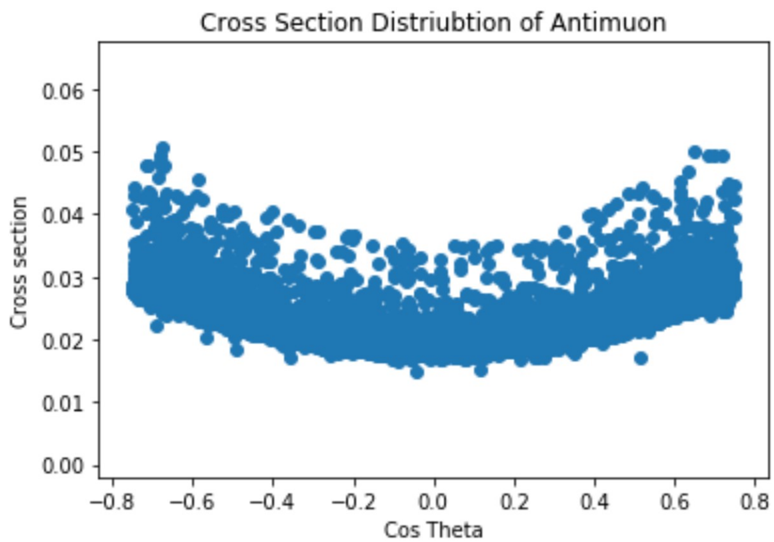
In [217]:
```
f1p1Mag = np.array(p1Mag)[ones]
f1p1Mag = f1p1Mag.reshape(len(f1p1Mag))
f1p2Mag = np.array(p2Mag)[ones]
f1p2Mag = f1p2Mag.reshape(len(f1p2Mag))
```

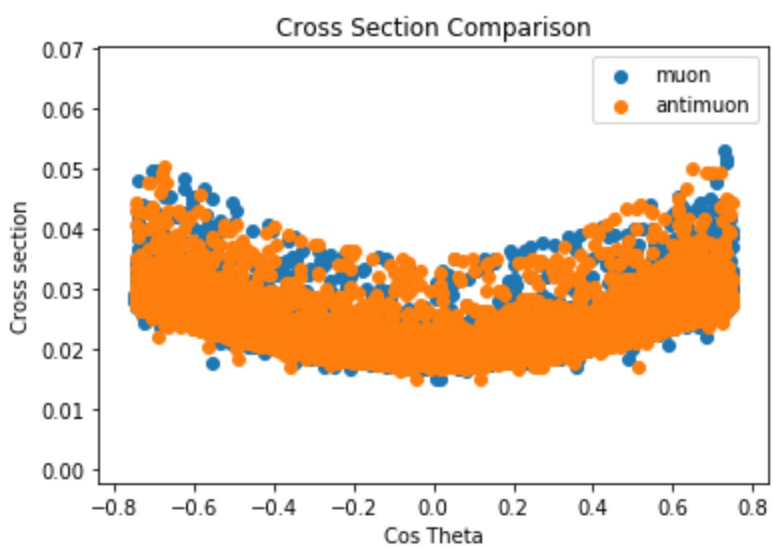
In [218]:
```
f2p1cos = f2p1cos.reshape(len(f2p1cos))
f2p2cos = f2p2cos.reshape(len(f2p2cos))
```

In [219]:
```
# Without the constant in front, but still considering s =  2*p1*p2
# low mass approx
s = 2*f1p1Mag*f1p2Mag
alpha = 1/137

factor = 2*np.pi*(alpha**2)/(4*s)

no_factor = False

if no_Denom:
    denom = 1

p1cross = (1 + (f2p1cos)**2)/denom
p2cross = (1 + (f2p2cos)**2)/denom
```

In [220]:
```
plt.figure()
plt.title("Cross Section Distriubtion of Muon")
plt.xlabel("Cos Theta")
plt.ylabel("Cross section")
plt.scatter(f2p1cos, p1cross)

plt.figure()
plt.title("Cross Section Distriubtion of Antimuon")
plt.xlabel("Cos Theta")
plt.ylabel("Cross section")
plt.scatter(f2p2cos, p2cross)

plt.figure()
plt.title("Cross Section Comparison")
plt.scatter(f2p1cos, p1cross, label = 'muon')
plt.scatter(f2p2cos, p2cross, label = 'antimuon')
plt.xlabel("Cos Theta")
plt.ylabel("Cross section")
plt.legend()
plt.show()
```







Based on these plots above, it appears that the cross section distribution of the muon is well aligned to the antimuon. There does not seem to be evidence for the parity violation from the cross section plots.

Physicists often characterize such parity violating effects in terms of a "forward-backward asymmetry":

$$A_{FB} = \left(\frac{N^{\mu -}_{\cos\theta > 0} - N^{\mu -}_{\cos\theta < 0}}{N^{\mu -}_{\cos\theta > 0} + N^{\mu -}_{\cos\theta < 0}} \right)$$

calculate A_{FB} and its uncertainty for the BaBar data (note: Because the data falls into one of two possible categories, the uncertainty follows a binomial distribution)

Fit the μ^- $\cos\theta$ distribution to the form

$$N = N_0 \left(1 + A\cos\theta + B\cos^2\theta\right)$$

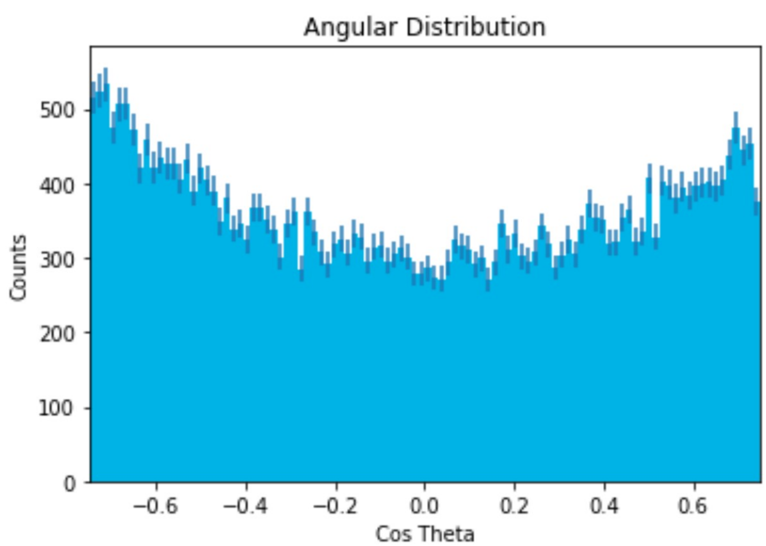
(make sure you only for for the range $|\cos\theta| < 0.7485$ Do your fitted values provide a value of A_{FB} that is consistent with the data?

In [221]:
```
pos = np.where(f2p1cos > 0)[0]
neg = np.where(f2p1cos < 0)[0]

pos_cos = f2p1cos[pos]
neg_cos = f2p1cos[neg]

N_pos = len(pos_cos)
N_neg = len(neg_cos)

analyticAfb = (N_pos - N_neg)/(N_pos + N_neg)
err_A = np.sqrt((((2*N_neg)/((N_pos+N_neg)**2))**2)*(N_pos) + (((2*N_p
os)/((N_pos + N_neg)**2))**2)*(N_neg))

print("Afb = ", analyticAfb, "+/-", err_A)
```

```
Afb =  -0.036204257687491304 +/- 0.005271768895359541
```

In [222]:
```
#Makes a histogram filled with the random numbers we generate
def plot_histogram(samples,xtitle,ytitle, title, nbins, limits):

    #Plot the histogram of the sampled data with nbins and a nice colo
r
    n, bins, patches =plt.hist(samples, bins=nbins, range=limits, colo
r=(0,0.7,0.9))  #Set the color using (r,g,b) values or
                                                                  #  u
se a built-in matplotlib color"""
    bincenters = 0.5*(bins[1:]+bins[:-1])
    errs = np.sqrt(n)

    plt.errorbar(bincenters, n, yerr=errs, fmt='none')
    #Add some axis labels and a descriptive title
    plt.xlabel(xtitle)
    plt.ylabel(ytitle)
    plt.title(title)

    #Get rid of the extra white space on the left/right edges (you can
delete these two lines without a problem)
    xmin, xmax, ymin, ymax = plt.axis()
    plt.axis([limits[0],limits[1],ymin,ymax])

    #Not necessarily needed in a Jupyter notebook, but it doesn't hurt
    plt.show()
    return n, bincenters, patches, errs
```

In [223]:
```
n, bincenters, patches, errs = plot_histogram(f2p1cos, "Cos Theta", "C
ounts", "Angular Distribution",
                                        nbins = 100,
                                        limits = [-reqval, reqval])
```

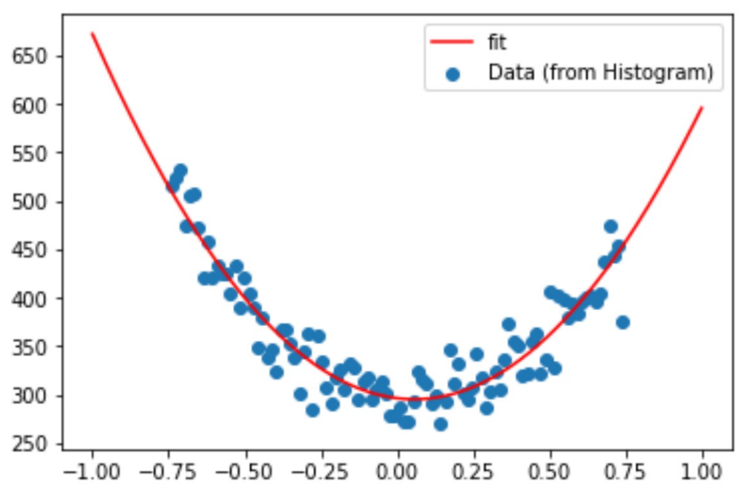


In [224]:
```
def polynomial(costheta, N_0, A, B):
    return N_0*(1 + A * costheta + B * (costheta **2))
```

In [225]:
```
fitparams, pcov = curve_fit(polynomial, bincenters, n)
errors = np.sqrt(np.diag(pcov))

N_0 = fitparams[0]
fitA = fitparams[1]
fitB = fitparams[2]

N_0err = errors[0]
fitA_err = errors[1]
fitB_err = errors[2]

print('Fitted N_0 = ', N_0, "+,-", N_0err)
print('Fitted A = ', fitA, "+,-", fitA_err)
print('Fitted B = ', fitB, "+,-", fitB_err)
```

```
Fitted N_0 =  296.25715798851445 +,- 3.421748403691694
Fitted A =  -0.12886021559621386 +,- 0.017879370772758776
Fitted B =  1.1404921896809936 +,- 0.05660963862871983
```

In [226]:
```
x = np.linspace(-1, 1, 1000)

plt.figure()
plt.plot(x, polynomial(x, N_0, fitA, fitB), 'r', label = 'fit')
plt.scatter(bincenters, n, label = 'Data (from Histogram)')
plt.legend()
plt.show()
```



A reduced χ^2 value near 1 indicates a good fit. Here, $\chi^2 > 1$, which suggests that the error might have be somewhat underestimated, but is still an overall good fit for the data.

To find A_{fb} we need to redefine it in terms of the fitted parameters A and B, as well as the bound "reqval" from the code (r in the equation below).

$$A_{fb} = \frac{\int_{-r}^{0} f(\cos(\theta))d\cos(\theta) - \int_{0}^{r} f(\cos(\theta))d\cos(\theta)}{\int_{-r}^{r} f(\cos(\theta))d\cos(\theta)}$$

where

$$\int_a^b f(\cos(theta)) = \int_a^b N_0[1 + A\cos(\theta) + B\cos^2(\theta)]d\cos(\theta)$$
$$\int_a^b f(\cos(theta)) = N_0[(b-a) + \frac{A}{2}(b^2 - a^2) + \frac{B}{3}(b^3 - a^3)]$$

Plugging in the above formula into A_{fb} and evaluating at the corresponding bounds given by r = 'reqval', we have:

$$A_{fb} = \frac{r^2 A}{2r + 2Br^3}$$

Finally having A_{fb} in terms of the fitted parameters.

In [228]:
```
# Finding A_fb:

Afb = (fitA * reqval**2)/(2*reqval + fitB*2*(reqval**3))
print("A_fb from fit = ", Afb)
```

```
A_fb from fit =  -0.029424658401102397
```

In [229]:
```
print("Upper Bound Afb from analytic formula = ", analyticAfb + err_A)
print("Lower Bound Afb from analytic formula = ", analyticAfb - err_A)
```

```
Upper Bound Afb from analytic formula =  -0.030932488792131764
Lower Bound Afb from analytic formula =  -0.04147602658285084
```

A_{fb} from the fit doesn't quite fit into the errorbar of the analytic A_{fb}, but it gets very close.

1f.

We can remove this parity violating term by taking the average of the angular distributions for the μ^+ and the μ^-:

$$\frac{d<N>}{d\cos\theta} = 0.5\left(\frac{dN^{\mu+}}{d\cos\theta} + \frac{dN^{\mu-}}{\cos\theta}\right)$$

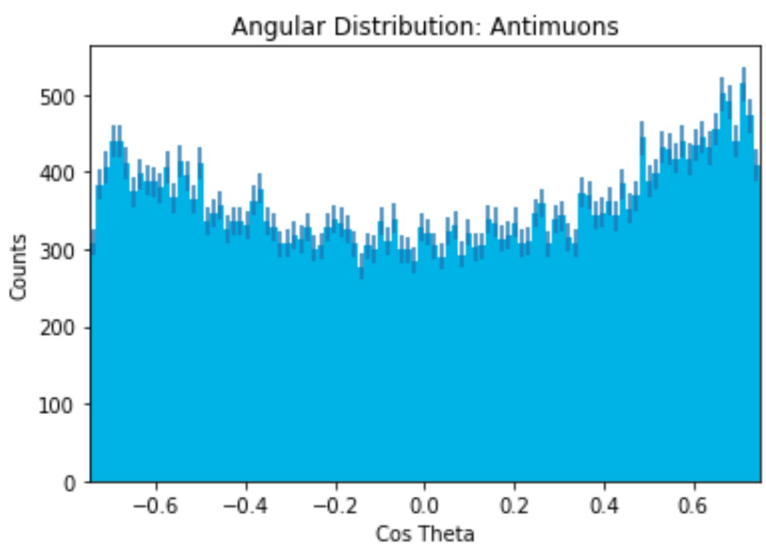
where $N^{\pm}$ are the number of μ^+ and μ^- respectively. Make a histogram of this distribution and fit the angular distribution to the form

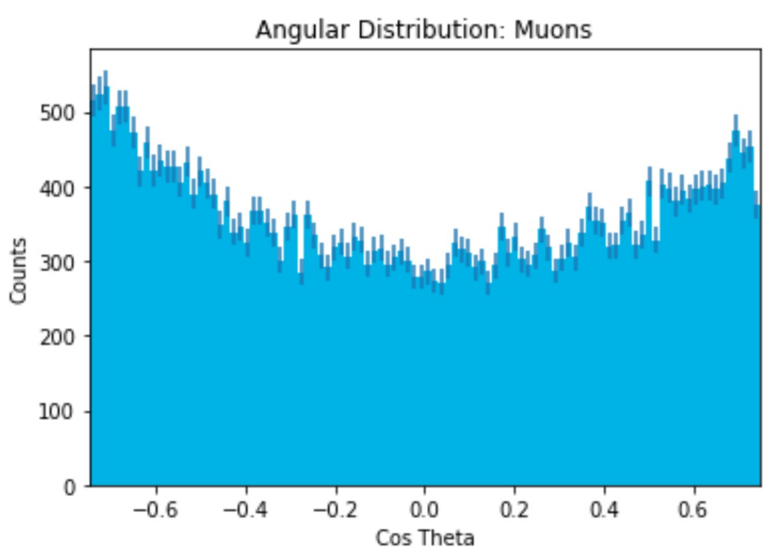
$$N = N_0\left(1 + B\cos^2\theta\right)$$

Is the fitted coefficient A consistent with your prediction?

Calculate the χ^2 between your histogram and your prediction. Turn the resulting χ^2 into a fit probability.

In [254]:
```
factor = 2 * np.pi * (alpha**2)/(4*s)
# muoncross = factor * (1 + f2p1cos**2)
# antimuoncross = factor * (1 + f2p2cos**2)

# overall = 0.5 * (muoncross + antimuoncross)

antiN, antiBins, apatches, anti_errs = plot_histogram(f2p2cos, "Cos Th
eta", "Counts", "Angular Distribution: Antimuons",
                                    nbins = 100,
                                    limits = [-reqval, reqval])

muonN, muonBins, mpatches, muon_errs = plot_histogram(f2p1cos, "Cos Th
eta", "Counts", "Angular Distribution: Muons",
                                    nbins = 100,
                                    limits = [-reqval, reqval])
```





In [255]:
```
# averaging out the distributions then:
average_dist = 0.5*(antiN + muonN)
#propagated error
err_avg_dist = np.sqrt(anti_errs**2 + muon_errs**2)
```

In [256]:
```
# same bins
print(muonBins == antiBins)
```

```
[ True  True  True  True  True  True  True  True  True  True  True  T
rue
  True  True  True  True  True  True  True  True  True  True  True  T
rue
  True  True  True  True  True  True  True  True  True  True  True  T
rue
  True  True  True  True  True  True  True  True  True  True  True  T
rue
  True  True  True  True  True  True  True  True  True  True  True  T
rue
  True  True  True  True  True  True  True  True  True  True  True  T
rue
  True  True  True  True  True  True  True  True  True  True  True  T
rue
  True  True  True  True  True  True  True  True  True  True  True  T
rue
  True  True  True  True]
```

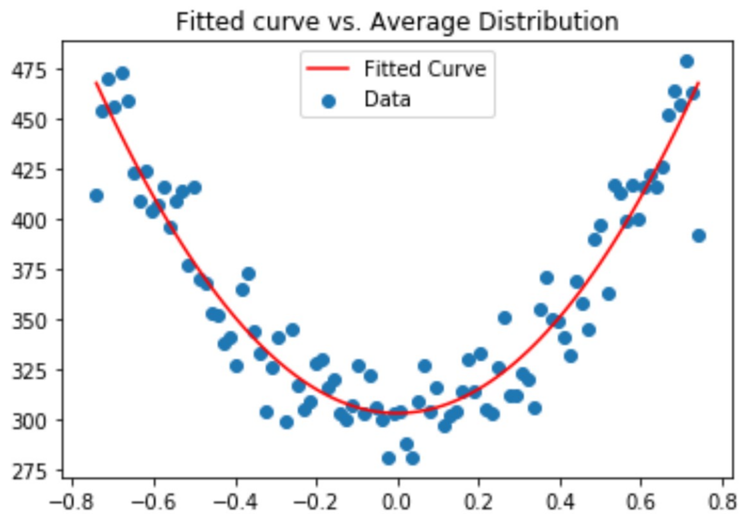
In [257]:
```
def justcos2(costheta, N_0, B):
    return N_0 * (1 + B * (costheta**2))
```

In [258]:
```
fp, matrix = curve_fit(justcos2, muonBins, average_dist)
```

In [259]:
```
errorsforavg = np.sqrt(np.diag(matrix))

avgN0 = fp[0]
avgB = fp[1]
errN0 = errorsforavg[0]
errB = errorsforavg[1]

print('From fit: ')
print('N_0 = ', avgN0, "+/-", errN0)
print('B = ', avgB, "+/-", errB)
```

```
From fit:
N_0 =  303.4207381207034 +/- 2.7320658008302954
B =  0.9871312575763619 +/- 0.04297923540171837
```

In [277]:
```
plt.figure()
plt.title("Fitted curve vs. Average Distribution")
plt.scatter(muonBins, average_dist, label = 'Data')
plt.plot(muonBins, justcos2(muonBins, avgN0, avgB), 'r', label = 'Fitt
ed Curve')
plt.legend()
plt.show()
```



In [276]:
```
average_dist
```

Out[276]:
```
array([412. , 454. , 470. , 456.5, 473.5, 459.5, 423.5, 409. , 424. ,
       404. , 407. , 416. , 396.5, 409.5, 414. , 377.5, 416. , 370.5,
       368. , 353. , 352.5, 338. , 341. , 327.5, 365. , 373.5, 344.5,
       333.5, 304. , 326.5, 341. , 299. , 345.5, 317.5, 305.5, 309.5,
       328. , 330. , 316. , 320. , 303. , 300. , 307. , 327. , 303. ,
       322.5, 306.5, 300. , 281.5, 303.5, 304. , 288.5, 281.5, 309. ,
       327.5, 304.5, 316.5, 297. , 302. , 304. , 314.5, 330.5, 314.5,
       333.5, 305. , 303.5, 326.5, 351. , 312.5, 312.5, 323.5, 320. ,
       306.5, 355. , 371.5, 350. , 349.5, 341. , 332. , 369.5, 358.5,
       345.5, 390. , 397.5, 363. , 417.5, 413.5, 399. , 417.5, 400. ,
       416. , 422.5, 416.5, 426. , 452.5, 464.5, 457.5, 479. , 463.5,
       392. ])
```

In [280]:
```
#computing chisquare
from scipy.stats import chisquare

chisq, pval = chisquare(f_obs = average_dist, f_exp = fittedvals)

print("Chisquare = ", chisq)
print("P-value = ", pval)
```

```
Chisquare =  85.40481727680621
P-value =  0.8330650505771748
```

In [281]:
```
### not sure if you can use reduced-chi square with non-linear fit

# counting_err = err_avg_dist
# fittedvals = justcos2(muonBins, avgN0, avgB)

# chi2 = 0
# for i in range(len(average_dist)):
#     chi2 += ((average_dist[i] - fittedvals[i])**2)/(counting_err[i]*
*2)

# df = len(fittedvals) - len(fp)
# reduced_chi2 = chi2/df
# print('Reduced Chi_2: ', reduced_chi2)
```

In [282]:
```
#analytic Afb (I tried to do with the same method in the previous prob
lem)
p = np.where(muonBins > 0)[0]
n = np.where(muonBins < 0)[0]
avgNpos = len(average_dist[p])
avgNneg = len(average_dist[n])

analyticAfbavg = (avgNpos - avgNneg)/(avgNpos + avgNneg)
errAavg = np.sqrt((((2*avgNneg)/((avgNpos+avgNneg)**2))**2)*(avgNpos)
+ (((2*avgNpos)/((avgNpos + avgNneg)**2))**2)*(avgNneg))
```

In [283]:
```
print(analyticAfbavg)
print(errAavg)
print("Can't use the same logic here, because 'average_dist' is a hist
ogram, not an actual dataset")
```

```
0.0
0.1
Can't use the same logic here, because 'average_dist' is a histogram,
not an actual dataset
```

In [271]:
```
# I can construct an A from the fitted parameter definition though. No
t that useful since I don't have a corresponding
# A from a dataset. I use the same integral definition for A_fb from b
efore, but use f(x) = N0(1 + B*x^2) instead
# same boundaries on the integrals. That resolves into the below formu
la for A
fittedAavg = -3/(avgB*(reqval**2))
print(fittedAavg)
```

```
-5.424537817704346
```