Monte Carlo Simulation for a New Proton Polarimeter

A thesis submitted in partial fulfillment of the requirement
for the degree of Bachelor of Science with Honors in
Physics from the College of William and Mary in Virginia,

by

Bradley M. DeBlois

Accepted for _____
(Honors)

_____
Advisor: Dr. C.F. Perdrisat

_____
Dr. Jeff Nelson

_____
Dr. Robert Lewis

_____
Dr. John Delos

Williamsburg, Virginia
May 2006

Abstract

An experiment in HallC of Jefferson lab will determine the ratio of the elastic electromagnetic form factors G_{E_p}/G_{M_p} of the proton by measuring the recoil proton polarization components. This will require characterizing the polarization of the recoil proton in the *ep* elastic scattering. The polarization of the proton is obtained by re-scattering the proton in a secondary target, the analyzer. A Monte Carlo simulation was written to produce distributions of the scattering angles, θ and ϕ, to be used to test the analysis code and study systematic asymmetry. The azimuthal angle distribution reveals the amount and direction of the proton polarization. A technique to remove the main source of asymmetry was tested. The background physics, properties of the simulation, resulting distributions, and methods for correcting systematic asymmetries are discussed.

Acknowledgements

Many thanks to...

My advisor, Professor Perdrisat, for the countless hours he spent with me. I was lucky to have an advisor willing to make time for me and explain things three times when I needed it.

Professor Brash of Christopher Newport University for answering emails on weekends and at night. His help on this project was essential.

The C-GEP group at Jefferson Lab for allowing me to take a small part in their ongoing experiment.

Contents

List of Figures

List of Tables

1 Introduction

The study of Nuclear Physics is dedicated to understanding the behavior of strongly interacting matter in terms of its basic constituents. A key component toward reaching this goal is the characterization of the structure of the nucleon. Important aspects of the structure of the nucleon are the form factors of the proton and neutron, $G_{E_p}, G_{M_p}, G_{E_n}$ and G_{M_n} since these form factors are directly related to the charge and current distributions inside the nucleon. Measurements of the form factors of nucleons and the nuclei were begun in the 1950's by R. Hofstadter [1]. The planned experiment measures the ratio of the elastic electromagnetic form factors, G_{E_p}/G_{M_p}. An experiment such as this relies heavily on computer simulation. To measure the ratio of the form factors, a great amount of data is collected and must be analyzed. An effective way of building and testing an analysis program is to simulate data that can be tested against the analysis program.

2 Form Factors

If the goal of a scattering experiment is to learn about the distribution of matter inside the nucleus, it is necessary to use a particle which will enter the nucleus. Electrons with wavelengths on the same order of magnitude as the nucleons, or smaller, are needed to probe the nucleus. This wavelength is governed by the de Broglie wavelength of the electron

$$\lambda = \frac{h}{p} \tag{1}$$

Therefore, these electrons will have small wavelengths and hence large momentum of the order 2 GeV for a desired wavelength of 0.1 fm. These fast electrons will interact with the nuclear matter through the electromagnetic field and yield information on the distribution of charge in the nucleus.

Information about the distribution of matter inside the nucleus is typically orga-

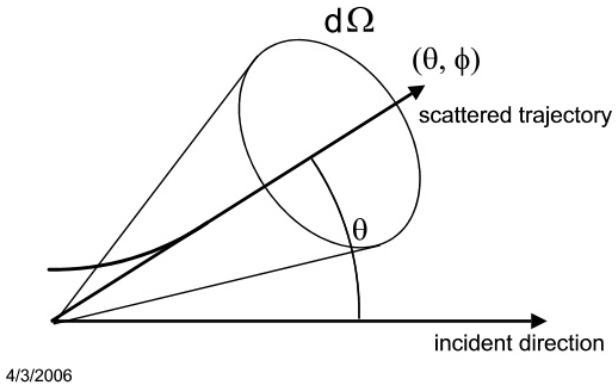

Figure 1: Classical cross section

nized into a quantity known as a differential scattering cross section, written $\sigma(\theta, \phi)$. Let (θ, ϕ) be the polar and azimuthal angles that specify the final direction of motion and $d\Omega$ be an infinitesimal solid angle around the ray at (θ, ϕ). Then the cross section is the number of particles scattered per second into $d\Omega$ divided by the incident flux [2]. In other words, the differential cross section is the area through which a particle can be scattered and is determined by the structure of the scattering element. Analytic forms of the cross section for particles scattered by a point charge can be obtained. For high energy electrons scattering from a point, the cross section is known as the Mott cross section and has the form [3]

$$\sigma_M(\theta) = \left(\frac{d\sigma}{d\Omega}\right)_{Mott} = \frac{\alpha^2}{4E_{beam}^2 sin^4\frac{\theta}{2}} \frac{E_e}{E_{beam}} cos^2\frac{\theta}{2} \tag{2}$$

However, scattering from the nucleus or a nucleon will not look like scattering from a point charge. Thus deviations from the cross section for a point charge reveal information regarding the nuclear structure, specifically the spacial distribution of charge. The equation above refers to the Mott scattering, σ_M. The actual cross

section will be the Mott cross section multiplied by a form factor

$$\sigma(\theta) = \sigma_M(\theta)[F(q)]^2. \tag{3}$$

The form factor F(q) is related to the charge density distribution $\rho(r)$ by a Fourier transform,

$$F(q) = \frac{4\pi}{q} \int \rho(r) sin(qr) r dr \tag{4}$$

This form factor is the quantity that represents the deviations from the point charge scattering and thus the quantity that reveals features of the nuclear structure.

For elastic scattering from the spin 1/2 proton, two form factors are required: F_1, the Dirac form factor, and F_2, the Pauli form factor [3]. The first, F_1, describes the non-point like nature of the electric and magnetic current distribution within the nucleus. The second, F_2, describes the non-point like nature of the distribution of the anomalous part of the magnetic moment. The proton does not have the magnetic moment of a Dirac particle, but is 2.79 times larger. The reason for this anomalous part is found in its complicated internal structure. These two form factors are related to the form factors of interest in this experiment by:

$$G_E = F_1 - \tau\kappa F_2 \tag{5}$$

$$G_M = F_1 + \kappa F_2 \tag{6}$$

where $\tau = Q^2/4M_p^2$, κ is the anomalous part of the magnetic moment (in nucleon magnetons $\frac{e\hbar}{2m_p}$) and M is the mass of the proton [3]. This experiment measures the ratio of the elastic electromagnetic form factors, G_{E_p}/G_{M_p}.

3 Polarization

A beam of nucleons is polarized if the spins are not randomly oriented but have a preferred orientation. Let n_+ represent the number of particles with a spin component

parallel to the preferred direction and n_- represent the number of particles with an antiparallel component, then the polarization is defined as

$$P = \frac{n_+ - n_-}{n_+ + n_-} \tag{7}$$

If the beam is not polarized then $n_+ - n_- = 0 = P$, and the scattering of the unpolarized beam will always be symmetric about the direction of incidence [4]. However, if a force acts preferentially on particles with a given spin orientation then a beam will become polarized. A force that depends on spin will create a left-right asymmetry.

To illustrate how asymmetry can occur, assume spin up protons are more likely to scatter to the left than to the right. Then when scattering N protons, N^+ spin up protons will go to the left and N^- will go to the right, with $N^+ > N$ and $N = N^+ + N^-$. The asymmetry in the scattering is defined as:

$$A = \frac{N_+ - N_-}{N_+ + N_-} \tag{8}$$

Now, of the N incident protons, $N^+ = \frac{N}{2}(1+\delta)$ will scatter to the left and $N^- = \frac{N}{2}(1-\delta)$ to the right so that in the absence of a spin dependence, $N^+ = N^- = N/2$; hence δ characterizes the spin dependence of the scattering probability. For an incoming spin up proton

$$A = \frac{N_+ - N_-}{N_+ + N_-} = \frac{1 + \delta - (1 - \delta)}{1 + \delta + 1 - \delta} = +\delta \tag{9}$$

And if the incoming proton is spin down the asymmetry becomes

$$A = \frac{N_+ - N_-}{N_+ + N_-} = \frac{1 - \delta - 1 - \delta}{1 - \delta + 1 + \delta} = -\delta \tag{10}$$

Therefore, measuring the asymmetry directly determines the spin dependence of the scattering probability and produces opposite values for spin up and spin down protons.

The main cause of this spin dependence is the coupling of the spin and angular momentum, $\vec{s}$ and $\vec{L}$.

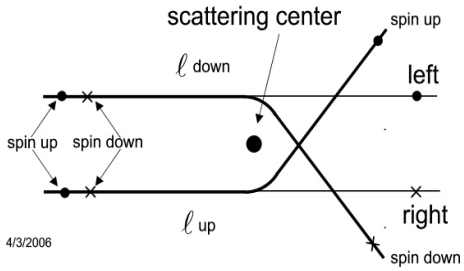

Figure 2: Spin-orbit Asymmetry

Figure 2 shows two nucleons with a given spin orientation, up or down. It also shows the direction of the angular momentum. As the nucleons approach the scattering center, they are scattered if the spin and angular momentum are in the same direction and not scattered if they are antiparallel. This phenomenon is given by the simplified yet illuminating equation

$$P(\theta) = \frac{1}{2}\left(1 + \frac{2\vec{L}\cdot\vec{s}}{|\vec{L}|}\right) \tag{11}$$

where P is the scattering probability, $\vec{L}$ is the angular momentum of the projectile relative to the target center $\vec{L} = b\vec{v}$ where b is the impact parameter, and $\vec{s}$ is the spin of the projectile. This gives $P = 1$ for $\vec{L}$ parallel to $\vec{s}$ and $P = 0$ for $\vec{L}$ antiparallel to $\vec{s}$.

4 Polarization Transfer Method

This experiment makes use of polarization and asymmetry to measure the form factor ratio, G_{E_p}/G_{M_p}. The G_{E_p}/G_{M_p} form factor ratio can be obtained from the recoil proton polarization transfer coefficients of the $\vec{e}p \rightarrow e\vec{p}$ reaction. In this reaction, the polarization of the electrons is transferred to the recoil proton with two non zero components, P_t perpendicular to, and P_l parallel to the proton momentum in the scattering plane. The polarizations are:

$$I_0 P_n = 0 \tag{12}$$

$$I_0 P_t = -2\sqrt{\tau(1+\tau)} G_{Ep} G_{Mp} tan\frac{\theta_e}{2} \tag{13}$$

$$I_0 P_l = \frac{1}{M_p}(E_{beam} + E_e)\sqrt{\tau(1+\tau)} G_{Mp}^2 tan^2\frac{\theta_e}{2} \tag{14}$$

Together these equations give:

$$\frac{G_{E_p}}{G_{M_p}} = -\frac{P_t}{P_l}\frac{E_{beam} + E_e}{2M_p}\tan\frac{\theta_e}{2} \tag{15}$$

In this way, the form factor ratio can be calculated from a single measurement of the two recoil polarization components P_t and P_l, perpendicular and parallel to the proton momentum at the hydrogen target [5].

5 The New Focal Plane Polarimeter

5.1 Basic Operation of a Polarimeter

A polarimeter is an apparatus used to reconstruct the tracks of scattered protons which can then be used to calculate the polarization of the protons. A polarimeter has two main components, the analyzers and the detectors. Typically there are two detectors, usually wire chambers which will be described in some detail in the next section, followed by an analyzer block, followed by two more detectors. The initial set of detectors defines the path of the incident proton. The proton is then scattered at some angle by the analyzer block. The second set of detectors defines the path of the scattered proton [6]. The incident and scattered paths reveal the scattering angles from which polarization can be calculated. Refer to Figure 3.

5.2 The New Focal Plane Polarimeter

In order to collect as much useful data as possible, the new polarimeter will use two sets of the analyzer/detector combination. This experiment will measure the polarization transferred to the recoil proton with a polarized electron beam scattered

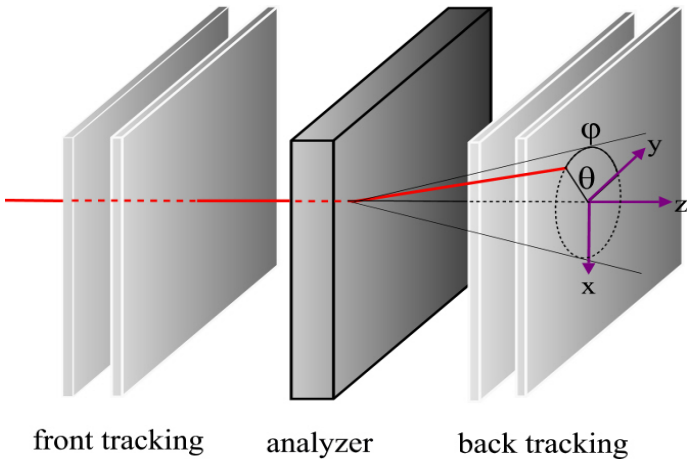

Figure 3: Conceptual Polarimeter

by an unpolarized hydrogen target. The new focal plane polarimeter collects data on the recoil protons. A magnetic field directs the recoil proton around a 25 degree bend and through two focal plane drift chambers, sandwiching the focal plane where protons with the same momentum are focused to a single point. Downstream from the focal plane the protons enter the polarimeter and data is collected on tens of millions of events.

The incident track of the proton is defined by the focal plane detectors. The proton is then scattered by the first analyzer. The outgoing particle is observed by two detectors which define the scattered track. In addition, these chambers also define the incident track for the second analyzer. The proton is then scattered by a second analyzer and a second set of detectors defines the final scattered track. The detectors are wire chambers built specifically for this experiment and will be discussed later. The analyzers are blocks made of CH_2.

The benefit of having consecutive analyzer/detector combinations is that more protons will be scattered and thus more data can be collected. Due to the high energy of the incoming protons, about forty percent of them will not encounter any nuclear scattering when traveling through the analyzer. By having two analyzers, the protons have two opportunities to scatter.

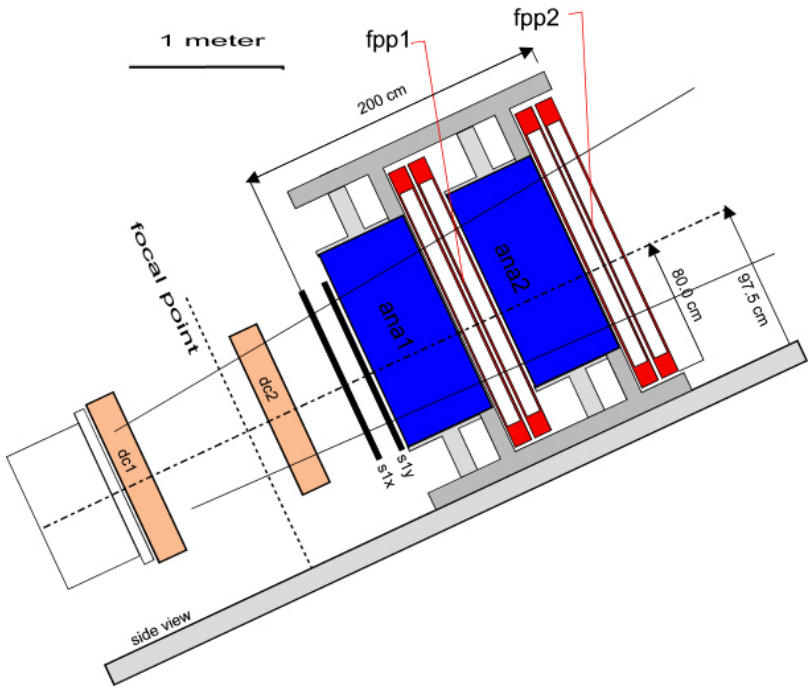

Figure 4: The New Polarimeter

6 Drift Chambers

6.1 Basic Operation

Drift chambers are the primary tracking detectors used in high energy scattering experiments. Drift chambers are wire chambers divided up into drift cells. A drift cell has a sense wire in the center (anode, at ground potential), and field wires left and right at negative potential, as well as cathode planes above and below and also at negative potential, as illustrated in Figure 5. An ionizable gaseous mixture fills the drift cell.

As a particle traverses the drift cell it ionizes atoms of the gas and the resulting electrons drift toward the sense wire. In some cases, the electrons have enough energy to ionize gas molecules as well. As a result, an avalanche of ionization electrons is produced. The particle also hits a scintillation counter which covers the entire area of the chamber. When the electron hits the scintillator it starts a timer, the timer is

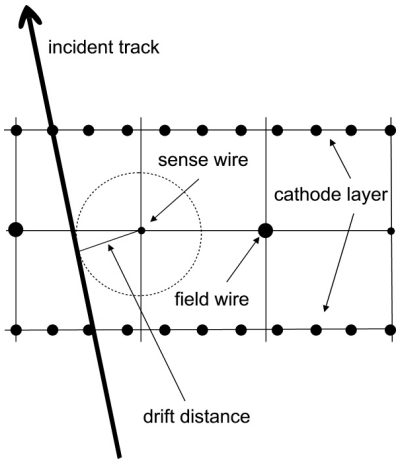

Figure 5: Drift Cell

stopped when the drifting electrons reach the sense wire [7]. In this manner the drift time is obtained. Figure 5 shows the drift cell. The dotted circle represents points that will have equal drift times. Assuming we know the drift velocity and the arrival time of the particle, the distance from the origin of the electrons and the sense wire, the drift distance, is

$$x = \int_{t_1}^{t_0} u dt, \tag{16}$$

where $t_1 - t_0$ defines the drift time and u is the drift velocity.

6.2 Wire Chambers for the New Polarimeter

In this particular GEP-III Hall C experiment, a set of large wire chambers was built and delivered from Russia. The new focal plane polarimeter will make use of four wire chambers. Each wire chamber has three separate wire planes designed in the manner explained above. However, each wire plane has a different orientation. One plane is at +45 degrees, the center plane is at zero degrees, and the third wire plane is at -45 degrees relative to the dispersive direction of the spectrometer, the (x) direction. Using three wire planes in different orientations provides three drift times and thus three drift distances per chamber. This allows for a reconstruction of the path of the

particle that traveled through the chamber. The sense wires are spaced 2.0 cm apart with a field wire in between. The wire planes are 1.6 cm apart with a cathode layer in between. The wires of the cathode layer are 0.3 cm apart. Thus the drift cell is 2.0 cm by 1.6 cm. The sense wires are 30 μm gold plated Tungsten and the field wires are 100 μm Beryllium and Bronze alloy.

7 The Simulation

When performing an experiment that will collect an immense amount of data, it is crucial to have a mechanism for predicting the results of the experiment so that one can gain experience with the experimental data before actually running the experiment. Computer simulations are the preferred method for simulating experimental events. Computer methods can produce simulated data, that with careful consideration of physical properties, can look identical to the experimental data. Thus the simulated data can be used as if it were real data with the benefit of knowing all of the properties of the data ahead of time.

Using the physics simulation package GEANT, written specifically for high energy scattering experiments, real results of the collision of the electrons at the target can be simulated. This sends protons down the path and into the polarimeter. If GEANT is used to code the properties of the polarimeter then drift times can be calculated. These drift times can then be passed to the analysis package, which will receive drift times from the actual experiment. The goal of the analysis package is to reconstruct the tracks of the protons based on the drift times. After the analysis package does this for the simulated drift times, the reconstructed tracks can be compared to the known simulated tracks and thus the correctness of the analysis package can be determined.

A similar simulation was written for a prior Jefferson Lab experiment in Hall A. The code for that experiment needed to be updated to meet the needs of the new

polarimeter.

7.1 Coding the Polarimeter Geometry

The first step in updating the old simulation was to model the new polarimeter. The GEANT simulation package is written for high energy experiments and makes defining a package of detectors fairly easy. Defining the polarimeter includes creating all elements in Figure 4.

Two identical focal plane wire chambers are needed, two scintillators, two analyzing blocks, four wire chambers, the air space in between each element and a room to house the entire apparatus. GEANT allowed for easy creation of these elements. Table 1 shows the dimensions of each component. Each component was positioned

Component	x (cm)	y (cm)	z (cm)
focal plane wire chambers	122	81	17
scintillators	122	81	1
analyzers	142	108	56
fpp wire chambers	166	134	11

Table 1: Polarimeter Dimensions

at the correct z-position with z=0 at the focal plane. Air volumes with z-dimensions that exactly fill the space between the detector elements were also added. The wire chambers are defined as boxes of air since air behaves similarly to actual gas of the chambers, Ar-ethane. Developing the geometry of the wire chambers will be discussed when calculating drift distances is discussed. The analyzers are defined as boxes of CH_2. Events are simulated at the target through GEANT kinematics simulation code and the recoil protons appear at the polarimeter at z=0 or the focal plane. The protons travel through the polarimeter and scatter at the analyzers. The scattering occurs as a result of the interaction between the proton and the nuclei of the carbon

and hydrogen in the CH_2 analyzers and the amount and types of scattering will be discussed in depth later.

As the proton moves through the polarimeter, the simulation records the x,y,z position of the particle every time it enters or exits one of the components. Accessing this information enables one to build the tracks of the protons, calculate drift distances, and calculate scattering angles. Calculating and organizing this information produces several important distributions that will ultimately be used to calculate polarization. Examining these important distributions is critical to ensure the accuracy of the simulation. The calculation and results of these distributions will be discussed at length in the following sections.

7.2 Drift Distance

Recall that the drift time is the time it takes the electron avalanche created by an incoming proton to reach the sense wire and the drift distance is the distance traveled. The drift time is an extremely important quantity since it will be the data collected in the actual experiment. The process for producing simulated drift times is to use geometry to calculate drift distances which can then be converted to drift times.

Calculating the drift distance begins with defining the incident track and calculating where it hits each wire plane. From the code, the x, y, and z coordinates are recorded each time the track enters or exits an element of the polarimeter. Define the entry and exit points of the chamber as $P_1(x_1, y_1, z_1)$ and $P_2(x_2, y_2, z_2)$ respectively.

Think of these two points as defining a line in three dimensions with canonical equations

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} = \frac{z - z_1}{z_2 - z_1}. \tag{17}$$

The z position of a given wire plane is known so solving for x and y is simple

$$x_{plane} = \frac{(z_{plane} - z_1)(x_2 - x_1)}{z_2 - z_1} + x_1 \tag{18}$$

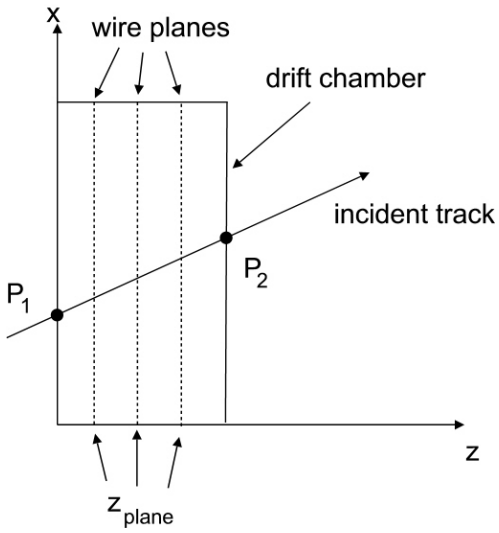

Figure 6: Drift Geometry

$$y_{plane} = \frac{(z_{plane} - z_1)(y_2 - y_1)}{z_2 - z_1} + y_1 \tag{19}$$

where x_{plane} and y_{plane} are the (x, y) coordinates of the incident track at the given wire plane.

After calculating where the incident track hit the wire plane, the next step is to figure out which wire is closest to the track since that is the wire which will register a hit. This is done through careful examination of the geometry of the wire planes. Each wire of a wire plane is given a number and placed at an appropriate position. An algorithm was written by Dr. Edward Brash, of Christopher Newport University, to determine which wire number is closest to the calculated (x, y) point of the incident track at the wire plane. Next, the drift distance, the shortest distance between the track and the wire, is calculated. This is another three dimensional geometry problem involving two skew lines, the track and the wire. The desired quantity is the closest distance between the two lines. Dr. Brash also wrote an algorithm for calculating this distance, the code can be found in the appendix. In this manner drift distances were calculated and their distributions observed.

Figure 7 shows a sample distribution of drift distances for one of the wire planes

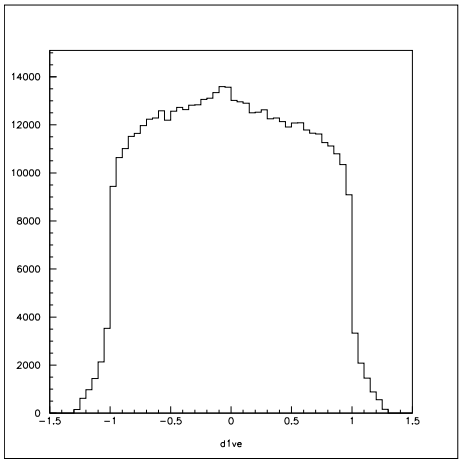

Figure 7: Drift Distance for one wire plane

in the first chamber. Drift distances close to 0 cm are more likely and note the drop off as distance increases. The largest possible drift distance given the dimensions of the drift cell is approximately 1.3 cm which the distribution also shows.

7.3 Scattering Angles

The relevant variables will be analyzed by examining the spherical scattering angles θ and ϕ after scattering in each of the analyzer blocks. These angles are the polar and azimuthal angles between the incident track and the scattered track. The method for obtaining these angles is as follows: calculate the cartesian angles for the incident track. Build a rotation matrix from those angles. Rotate the scattered track by the rotation matrix. Now the scattered track is in the same coordinate system as the incident track and the spherical angles are the polar and azimuthal angles [8].

Consider the incident track $\mathbf{f}$: θ_f is the angle between the projection of the track onto the xz plane and the z-axis, ϕ_f is the angle between the projection of the track onto the yz plane and the z-axis, and ψ_f is the angle between the projection of the track onto the yz plane and the track. These three angles represent the cartesian angles.

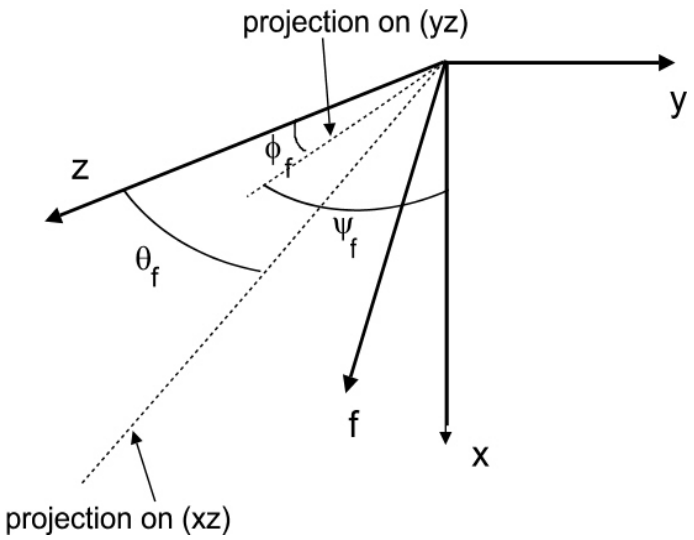

Figure 8: Cartesian Angles

Using small angle approximations

$$\theta_f = x/z \tag{20}$$

$$\phi_f = y/z \tag{21}$$

where x, y, and z are given by the direction vector of the incident track. Also

$$\tan\psi_f = \tan\theta_f \cos\phi_f \tag{22}$$

Now the rotation should be broken into two steps, a rotation of the yz plane around the x-axis by an angle θ_f and the second a rotation by an angle ψ_f. This rotation sets the incident track along the z direction. The new projection of the scattered track **r** is now [8]:

$$\begin{bmatrix} r'_x \\ r'_y \\ r'_z \end{bmatrix} = \begin{bmatrix} \cos\psi_f & 0 & -\sin\psi_f \\ 0 & 1 & 0 \\ \sin\psi_f & 0 & \cos\psi_f \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_f & -\sin\phi_f \\ 0 & \sin\phi_f & \cos\phi_f \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \tag{23}$$

We can now calculate the spherical angles θ and ϕ

$$\theta = \tan^{-1}\left(\frac{\sqrt{r'_{x^2} + r'_{y^2}}}{r'_z}\right) \tag{24}$$

$$\phi = \tan^{-1}\left(\frac{r'_x}{r'_y}\right) \tag{25}$$

The values of θ and ϕ for each scattered event at each analyzer is collected. They are placed into a histogram so that the θ and ϕ distribution for each analyzer can be examined. Figures 9 and 10 are examples of the distributions.

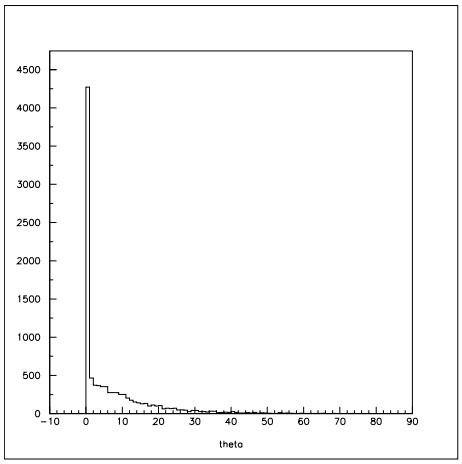

Figure 9: Theta distribution

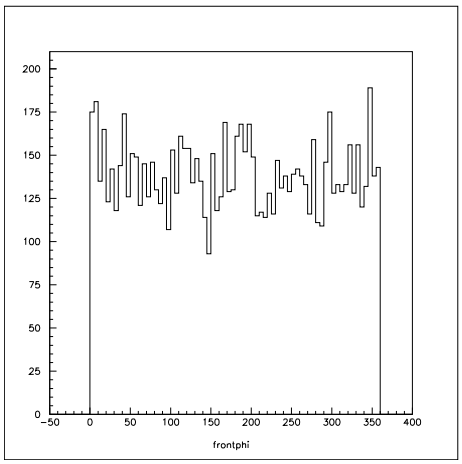

Figure 10: Phi distribution

There is a lot to be learned about the simulation from the θ and ϕ distributions it produces. In general, θ determines the momentum transfer. The azimuthal angle, ϕ

is a particularly important angle since its distribution will determine the polarization. It is also a very important quantity that helps to refine and correct the simulation.

8 Analyzing the Data

Building an appropriate simulation requires an iterative process. A basic simulation is written, run, and the results are analyzed. Based on the initial results, the simulation is refined and made more comprehensive. For this particular experiment the θ and ϕ distributions were repeatedly analyzed and the simulation was corrected accordingly.

8.1 Single vs Multiple track Events

As charged and heavy particles like protons travel through matter, they undergo multiple Coulomb scattering in the electric fields of the atoms/molecules, and strong nuclear interaction with the nuclei/nucleons of these atoms/molecules (as well as much weaker interaction with the charges of the nuclei/nucleons). Every proton traversing the analyzer undergoes multiple Coulomb scattering; if nothing else happens, they appear in the Coulomb peak at zero degrees, with a width inversely proportional to their momentum, and proportional to the square root of the thickness of material. If they do undergo nuclear scattering, then they will be scattered as some finite angle, and Coulomb scattering will only slightly smear their θ and ϕ distribution. Initial θ distributions showed that only about thirty percent of the events were experiencing nuclear scattering and produced a single track. This number was much less than the expected value of about sixty percent. Notice in Figure 11 that the Coulomb peak contains a high percentage of the events and not enough scattered tracks appear in the tail. In fact, there are only events scattered by a few degrees.

The expected sixty percent scattering is calculated from the probability of survival, $e^{-x/\lambda}$ where λ is the mean free path in the scattering block and $x = \rho d$ where ρ is

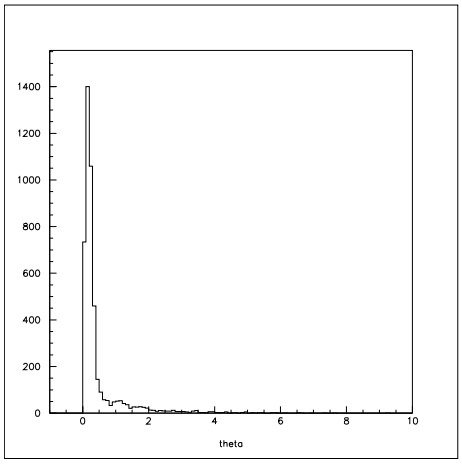

Figure 11: Theta Distribution for Single Tracks

the density of the analyzer and d is the thickness. For $CH_2, \lambda = 57g/cm^2$ and $x = 0.94 * 55 = 51.7$ so that $e^{-51.7/57} \approx 0.40$. If forty percent are expected to survive then a total of sixty percent should interact.

It became immediately apparent that limiting the simulation to only single track events was not physically correct and producing incomplete results. At high energy the proton-nucleus interaction is highly inelastic and the scattering results in one or several particles being created as in $pp \rightarrow pn\pi^+$, $pp \rightarrow pp\pi^+\pi^-$ etc. Also, the incident proton may scatter from another nucleon before exiting the analyzer. In fact a single incident proton could produce a chain of scattering within the analyzer. This effect is defined as scattering multiplicity, such that if the incident proton results in two particles after scattering then it has multiplicity two, three particles after scattering results in multiplicity three and so on. Figure 12 shows one manner in which a multiplicity of three could occur.

In GEANT, tracking the multiple scattered tracks is simply a physical setting, thus multiple tracks are easily observed. The difficulty lies in selecting only the scattered particles of interest. This was accomplished by tracking only the particles that experienced the smallest scattering angle θ. Adding multiple tracks provided the data shown in Table 2 for the first chamber with 10,000 events where the first column

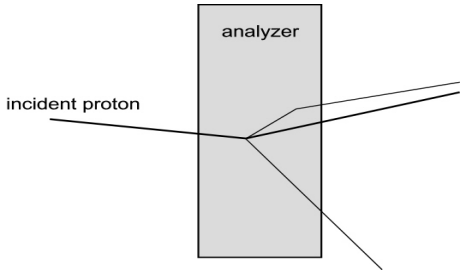

Figure 12: Multiplicity of 3

multiplicity	# scattered	% scattered
1	1913	19
2	1367	14
3	984	10
higher	1354	14

Table 2: Front Analyzer

is the multiplicity, the second column is the number of events that were scattered with the given multiplicity and the third column is the percentage of the 10,000 events that were scattered with the given multiplicity. Adding the percentages, 57 percent of events were scattered, very close to the expected value of sixty percent.

For the second chamber, only events which experienced no nuclear scattering in the first chamber are considered, for this particular run 4600 events are considered as incident particles to be analyzed by the second chamber. Table 3 shows the result from these 4600 events.

multiplicity	# scattered	% scattered
1	772	18
2	523	12
3	394	9
higher	523	12

Table 3: Rear Analyzer

Adding up the third column shows that when including scattering multiplicity, about 51 percent of the events experienced nuclear scattering again close to the expected amount. Thus in each analyzer, when multiple tracks were allowed, the number of scattered events was close to the expected value of sixty percent. Figure 13 shows the θ distributions for the first two multiplicities on the same plot as the full θ distribution. The distributions have been set so that the Y axis is a logarithmic scale so that it is easier to see the difference in the number of events in the Coulomb peak and tail. The highest plot, represented by the full line, is the full theta distribution. Notice how each consecutive multiplicity contains fewer and fewer events.

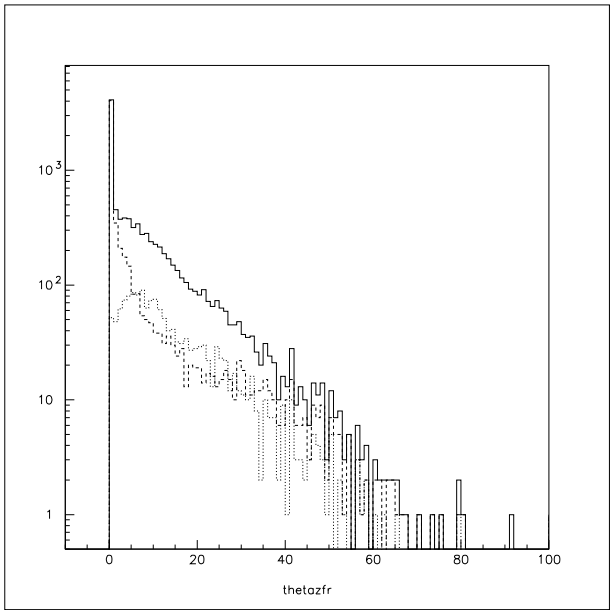

Figure 13: Theta of Varying Multiplicity

When multiple tracks are included, the θ distribution looks much nicer. Compare Figures 14 and 15. Notice that there are many more events in the tail of the multitrack distribution (Figure 15) which are the events of interest. Each distribution is taken from 10000 events recorded from the front analyzer.

Now that the correct number of analyzable events have been produced, attention is turned to the ϕ distribution.

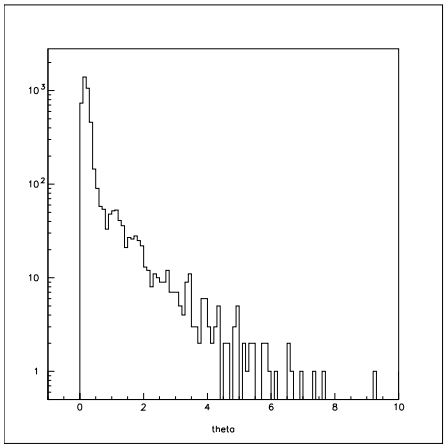

Figure 14: Theta Distribution for Single Tracks

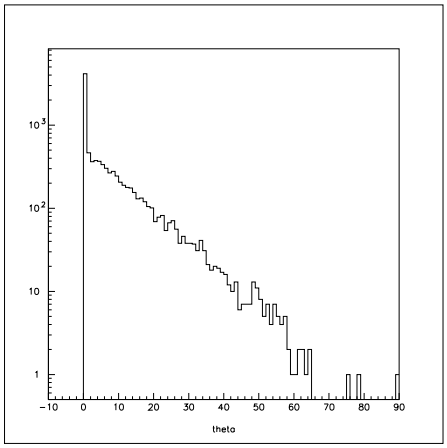

Figure 15: Theta Distribution for Multiple Tracks

8.2 The Azimuthal Angle and Asymmetry

The result of scattering in the polarimeter is an azimuthal asymmetry in the scattering distribution. Analyzing this asymmetry provides information on the polarization of the proton.

The azimuthal angular distribution of the number of particles scattered, N, is:

$$N = N_0(1 + A_y(\theta)\vec{P}^{fpp} \cdot \vec{n}) \tag{26}$$

where N_0 is the number of particles scattered in the absence of polarization, $\overrightarrow{P}^{fpp}$ is the proton polarization vector at the polarimeter, A_y is the analyzing power of the analyzing reaction, and $\hat{n}$ is a unit vector normal to the scattering plane, $\hat{n} = \hat{k} \times \hat{k'}/|\hat{k} \times \hat{k'}|$ where $\hat{k}$ and $\hat{k'}$, are the unit vectors in the direction of the incident and scattered proton [5].

Now, P^{fpp} has three components, longitudinal, normal, and transverse. It is important to notice how the components of the polarization change from the target to the polarimeter. The proton spin precesses as it travels from the target resulting in new polarization components at the polarimeter as shown in Figure 16.

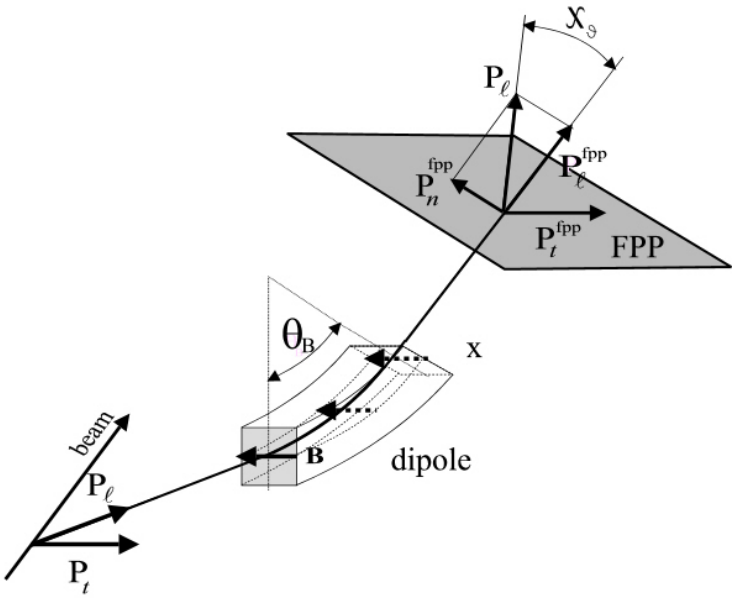

Figure 16: Spin Precession

The transverse polarization component P_t is parallel to the magnetic field and does not undergo any precession. The longitudinal polarization component P_l is perpendicular to the magnetic field and precesses with angle χ_θ.

In terms of the polarization components at the polarimeter, $\vec{P_l} \cdot \hat{n} = 0$ since the longitudinal polarization is perpendicular to the normal vector, $\vec{P_n} \cdot \hat{n} = cos(\phi)$ and $\vec{P_t} \cdot \hat{n} = sin(\phi)$. This produces an azimuthal distribution which, for spin-1/2 particles is of the general form:

$$N(\theta, \phi) = N_0(\theta)[1 + A_y(\theta)(P_t sin(\phi) - P_n cos(\phi))] \tag{27}$$

where θ and ϕ are the polar and azimuthal scattering angles, $N(\theta, \phi)$ and $N_0(\theta)$ are the number of particles recorded after the analyzer for the polarized and unpolarized beam, respectively; $A_y(\theta)$ is the analyzing power which depends on the material of the analyzer, the energy of the particle, and the polar angle.

It is useful to convert the azimuthal distribution to a detection probability distri-

bution:

$$f^{\pm}(\theta,\phi) = \frac{\epsilon(\theta,\phi)}{2\pi}(1 \pm A_y(P_t sin(\phi) - P_n cos(\phi)) \tag{28}$$

where the $\pm$ refers to the sign of the beam helicity and $\epsilon(\theta,\phi)$ is the efficiency of the polarimeter. It is clear that the distributions are sinusoidal and out of phase. Adding the distributions will show instrumental asymmetries and subtracting the distributions removes helicity independent effects.

8.3 The Azimuthal Angle and the Cone Test

In an attempt to ensure that the simulation is correct, unpolarized protons are scattered in the polarimeter. Without polarization there is no preferred direction and should therefore be no asymmetry or structure in the azimuthal, ϕ, distribution. However, when the simulation is run with high statistics, 20 million events, a clear structure is observed. See Figure 17.

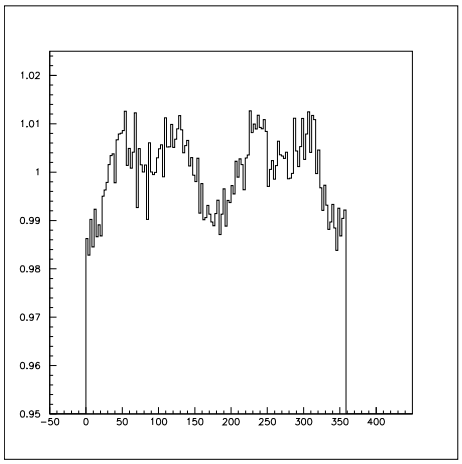

Figure 17: Phi Distribution

It is important to understand this structure. Think of this distribution as a flat distribution with large dips at 0 and π and smaller dips at $\pi/2$ and $3\pi/2$. The dips represent some sort of 'missing' data at those points. Now consider the back

of the second detector. Project the incident proton to the chamber. Now, without polarization, scattering is equally likely on a circle around the incident projection, meaning any azimuthal angle is equally likely. The circle radius is defined by the polar angle, θ. Consider what happens when the incident projection is near one edge of the detector, say the right side which corresponds to 0 radians. Each ϕ is equally likely but only the ones to the left side of the incident projection will be seen by the detector. There are an equal number scattered to the right of the incident point but these will not be recorded by the detector. See Figure 18. These unrecorded points are the 'missing' data that create the dips in the distribution.

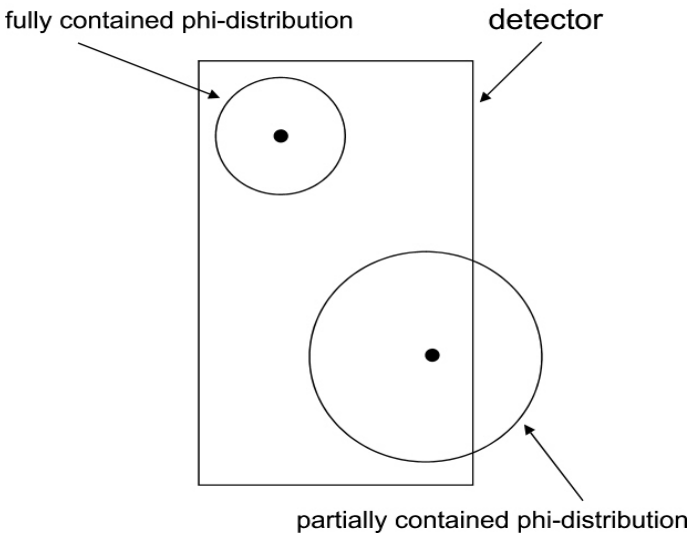

Figure 18: Cone Test Logic

To correct for this effect, there must be a test on the data that only accepts events for which a full ϕ distribution is possible. A cone test is defined with this sort of test. The cone has its origin at the point of interaction and opens to a circle at the back of the second detector defined by θ. Using the geometry of this cone, events for which a full phi distribution is not possible are eliminated. This effectively shaves off the peaks of the original ϕ distribution.

The process begins by locating the projection of the incident proton on the back of the chamber. The x and y position of this projection is defined by the following

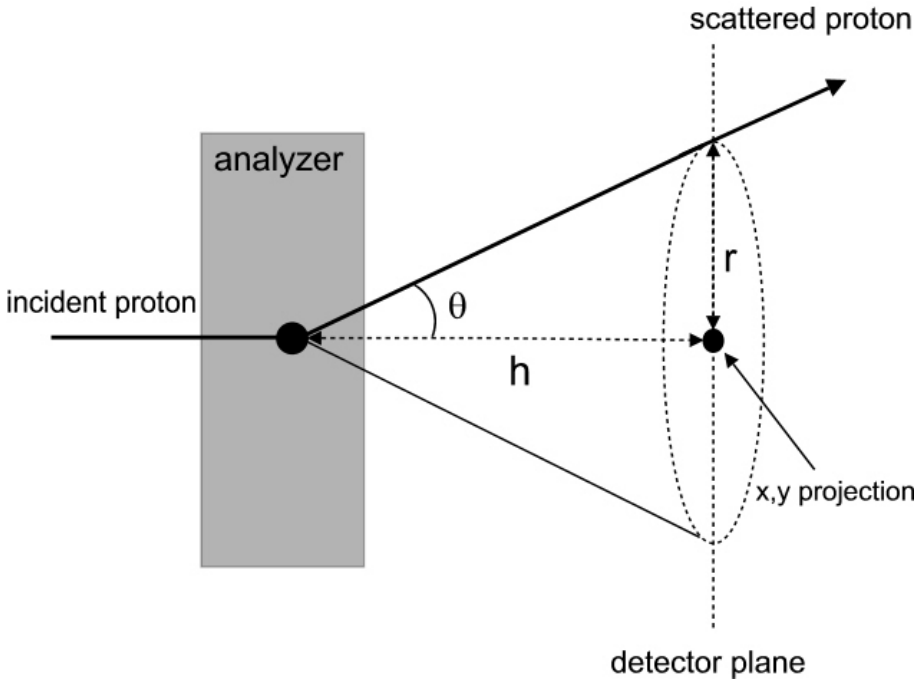

Figure 19: Cone Test Geometry

equations:

$$x_{proj} = \frac{(z - z_1)(x_2 - x_1)}{z_2 - z_1} + x_1 \tag{29}$$

$$y_{proj} = \frac{(z - z_1)(y_2 - y_1)}{z_2 - z_1} + y_1 \tag{30}$$

where (x_1, y_1, z_1) and (x_2, y_2, z_2) are points on the incident track and z is the z position of the back of the detector.

The height of the cone is just the distance between the point of interaction and the projection of the incident track onto the back of the detector.

$$h = \sqrt{(x_{proj} - x)^2 + (y_{proj} - y)^2 + (z_{plane} - z)^2} \tag{31}$$

where (x,y,z) is the point of interaction.

The radius of the cone is easily computed

$$r = h \cdot tan(\theta) \tag{32}$$

The circle at the back of the detector defined by this cone must be totally contained within the detector. To check this, add the radius to x and y coordinates of the projected point of the incident particle in the plus and minus direction. If this number is outside the dimension of the detector then the event is not accepted, otherwise it is accepted. Adding the cone test reduced the structure considerably as shown in Figure 20. The first plot is the final distribution after the cone test was applied, the second

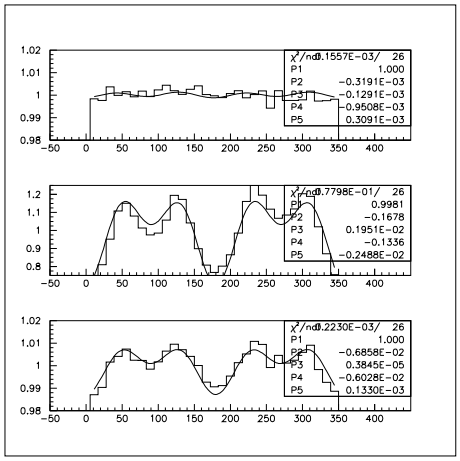

Figure 20: Phi Distributions-Cone Test

plot is the distribution of the ϕ values of the rejected events, and the final plot is the original ϕ distribution without a cone test. After the cone test the ϕ distribution has a much smaller structure than it did before the cone test.

To see which events are being rejected refer to Figure 21. The full line represents all θ values and is the usual logarithmic distribution of θ. The dotted line shows those events which were rejected by the cone test. As expected, most of the events rejected had fairly high θ values which correspond to larger cones.

As this paper was being written, a Fourier analysis of the data was performed and a structure still observed. In addition, Professor Brash of Christopher Newport

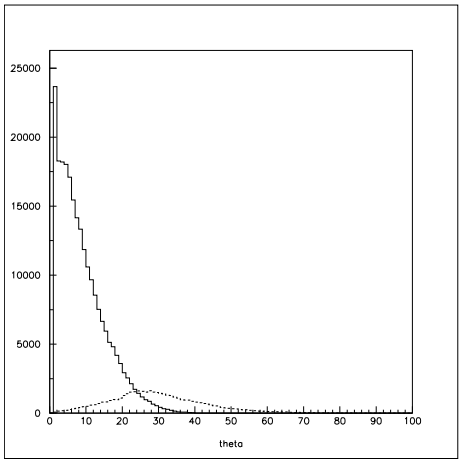

Figure 21: Theta Distributions: Full and Rejected

University wrote a second, and improved cone test that again decreased the structure of the ϕ distribution. Figure 22 shows both cone tests: the left column represents the improved cone test and the right column represents the cone test described above. The first row is the final distribution after the cone test was applied, the second plot is the distribution of the ϕ values of the rejected events, and the final plot is the original ϕ distribution without a cone test. The Fourier parameters are shown in the upper right hand corner.

Table 4 summarizes the results.

	No Cone Test	Circular Cone Test	Elliptical Cone Test
$cos(2\phi)$	-0.00686	-0.00158	-0.00032
$sin(2\phi)$	+0.00000	-0.00003	-0.00013
$cos(4\phi)$	-0.00603	-0.00090	-0.00095
$sin(4\phi)$	+0.00013	+0.00032	-0.00031
RMS Deviation	0.00913	0.00185	0.00106
Improvement	-	4.9	8.6

Table 4: Cone Test Results, the statistical uncertainty on all numbers is about $3 \times 10^{-}4$

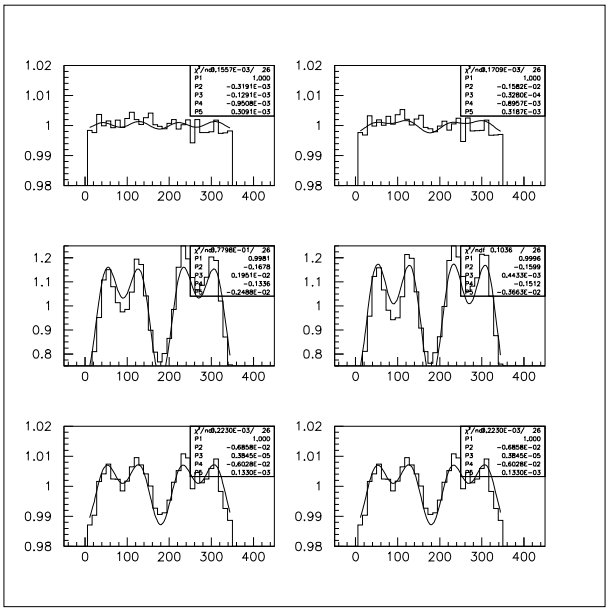

Figure 22: Comparing Cone Tests

The numbers in Table 4 are the coefficients of function

$$\frac{N(\phi)}{N_0} = 1 + Acos(2\phi) + Bsin(2\phi) + Ccos(4\phi) + Dsin(4\phi) \tag{33}$$

that fits the ϕ distribution, and the RMS Deviation is

$$RMS = \sqrt{A^2 + B^2 + C^2 + D^2}. \tag{34}$$

As Table 4 shows, the elliptical cone test decreased the structure. The improvement was made by dropping the earlier assumption that all tracks were perpendicular to the detector. For the earlier cone test, the azimuthal distribution is obtained by essentially 'cutting' the cone with the detector plane. When the tracks are perpendicular to the detector plane the result is a circle and hence the so called circular cone test. However, this is a bad assumption. Tracks are usually not perpendicular and cutting the cone with the detector plane creates an ellipse and hence the so called elliptical cone test.

In the circular cone test, the radius was the variable of interest to ensure that the azimuthal distribution was fully contained. In the elliptical cone test this is no longer the case. As Figure 23 shows, it is actually x_1 and x_2 that must be taken into

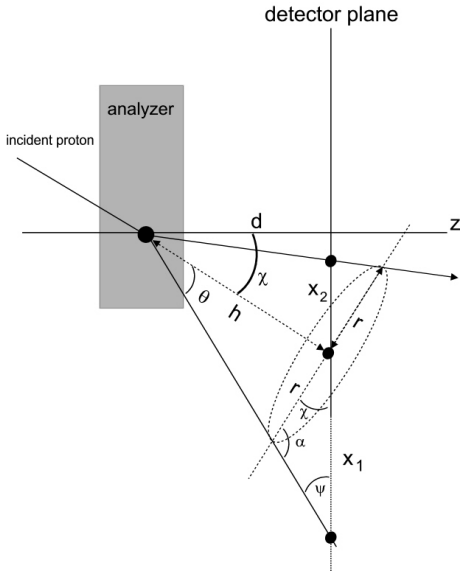

Figure 23: Elliptical Cone Test Geometry

account. When including this geometrical phenomenon, the azimuthal distribution gets better, the magnitudes of the distribution fit are all smaller than 10^{-3} which is acceptably low.

9 Conclusion

A focal plane polarimeter has been constructed for use in an experiment to obtain the ratio of the elastic electromagnetic form factors G_{E_p}/G_{M_p} by measuring the recoil proton polarization components. While the polarimeter is still in a testing phase, a computer simulation was written and described above. It produces data and predicts the results of the future true data collection. Analyzing the polar scattering angle distribution revealed that multiple track events must be taken into account. The initial distribution of the azimuthal angle revealed a systematic asymmetry. The main source of this asymmetry was removed by a technique called the cone test, which rejects events for which a full azimuthal distribution is not measurable.

There is still a great deal of information to be learned by studying the results

of the simulation. A main area that has not been studied has to do with the drift times. The electric field inside the drift chamber is not uniform and therefore the drift velocity is not uniform within a drift cell. The effects of this variable electric field must be included in the code and analyzed. Also, the drift time and drift distance are related mathematically by a fairly complicated relationship that must be accurately established.

References

[1] R. Hofstadter, Rev. Mod. Phys. 28, 214 (1956)

[2] M.A. Preston, *Physics of the Nucleus.* Addison-Wesley Publishing, Reading, Massachusetts, 1962.

[3] C.F. Perdrisat, V. Punjabi, and M. Vanderhaeghen, "'Nucleon Electromagnetic form Factors,"' in preparation (2006).

[4] L.R.B. Elton, *Introductory Nuclear Theory.* Second ed. W.B. Saunders Company, Philadelphia 1966.

[5] V. Punjabi, C.F. Perdrisat and Hall A Collaboration, "Proton elastic form factor ratios to $Q^2 = 3.5 GeV^2$ by polarization transfer," Phys. Rev. C 71 055202 (2005)

[6] C.F. Perdrisat, "Polarimetry in the Few GeV Region," Oct 26, 1994, unpublished.

[7] W.R. Leo, *Techniques for Nuclear and Particle Physics Experiments.* Second Revised ed. Springer-Verlag, 1994.

[8] O. Gayou, "Proton Form Factors, Measurement of the Proton Form Factors Ratio by Recoil Polarimetry," Ph.D. Thesis, College of William and Mary, unpublished. (2002)

[9] Stewart, James, *Multivariable Calculus.* 4th ed. Pacific Grove: Gary W. Ostedt, 1999.

A Relevant Code

A.1 Source Code for the Geometry

This is part of the routine that defines all of the components of the polarimeter.

```
        data hch1_size/75.00,60.00,8.5345/
data aira_size/75.00,60.00,32.131/
data hch2_size/75.00,60.00,8.5345/
data airb_size/60.96,40.64,12.387/
data sci1_size/60.96,40.64,0.50/
data airc_size/60.96,40.64,9.297/
        data sci2_size/60.96,40.64,0.50/
data aird_size/71.12,53.975,14.114/
        data anl1_size/71.12,53.975,27.94/
data aire_size/83.0,67.0,2.542/
data fch1_size/83.0,67.0,5.60/
data airf_size/83.0,67.0,5.00/
data fch2_size/83.0,67.0,5.60/
data airg_size/71.12,53.975,2.277/
        data anl2_size/71.12,53.975,27.94/
data airh_size/83.0,67.0,2.282/
data fch3_size/83.0,67.0,5.60/
        data airi_size/83.0,67.0,5.00/
data fch4_size/83.0,67.0,5.60/
c
        zhch1=-51.562
zaira=zhch1+hch1_size(3)+aira_size(3)
zhch2=zaira+aira_size(3)+hch2_size(3)
        zairb=zhch2+hch2_size(3)+airb_size(3)
        zsci1=zairb+airb_size(3)+sci1_size(3)
        zairc=zsci1+sci1_size(3)+airc_size(3)
        zsci2=zairc+airc_size(3)+sci2_size(3)
        zaird=zsci2+sci2_size(3)+aird_size(3)
        zanl1=zaird+aird_size(3)+anl1_size(3)
        zaire=zanl1+anl1_size(3)+aire_size(3)
        zfch1=zaire+aire_size(3)+fch1_size(3)
        zairf=zfch1+fch1_size(3)+airf_size(3)
        zfch2=zairf+airf_size(3)+fch2_size(3)
        zairg=zfch2+fch2_size(3)+airg_size(3)-0.1
        zanl2=zairg+airg_size(3)+anl2_size(3)
        zairh=zanl2+anl2_size(3)+airh_size(3)
        zfch3=zairh+airh_size(3)+fch3_size(3)
        zairi=zfch3+fch3_size(3)+airi_size(3)
        zfch4=zairi+airi_size(3)+fch4_size(3)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  the mother volume will be the HALL volume, it is filled with air
c
      write(6,*)'defining volumes now'
      write(6,*)'defining HALL now'
      call gsvolu ( 'HALL', 'BOX ', nair, HALL_size, ndim, iv_HALL )
c
      if ( iv_HALL.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_HALL=',iv_HALL
          write ( 6,* ) '        HALL geometry setup failed'
          stop
      end if
c
c  we do not see the HALL volume in the pictures
```

```
c
      call gsatt ( 'HALL', 'SEEN', 0 )
c
c  define the null rotation
c
      write(6,*)'defining null rot now'
      irot = 1
      call gsrotm(irot,nul_rot(1),nul_rot(2),nul_rot(3),
     x                 nul_rot(4),nul_rot(5),nul_rot(6) )
      irotnull = irot
c
c  define the rotation necessary to position the x-straws
c
      write(6,*)'defining rotx now'
      irotx = 2
      call gsrotm(irotx,rotx(1),rotx(2),rotx(3),
     x                 rotx(4),rotx(5),rotx(6) )
c
c
c  define the rotation necessary to position the u-straws
c
      write(6,*)'defining rotu now'
      irotu = 3
      call gsrotm(irotu,rotu(1),rotu(2),rotu(3),
     x                 rotu(4),rotu(5),rotu(6) )
c
c
c  define the rotation necessary to position the v-straws
c
      write(6,*)'defining rotv now'
      irotv = 4
      call gsrotm(irotv,rotv(1),rotv(2),rotv(3),
     x                 rotv(4),rotv(5),rotv(6) )
c
c
      write(6,*)'defining hch1 volume now'
      call gsvolu ( 'hch1', 'BOX ', nt_air,
     x                  hch1_size, ndim, iv_targ )
c
      if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        hch1 geometry setup failed'
          stop
      end if
c
c
      write(6,*)'defining aira volume now'
      call gsvolu ( 'aira', 'BOX ', nt_air,
     x                  aira_size, ndim, iv_targ )
c
      if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        aira geometry setup failed'
          stop
      end if
c
c
      write(6,*)'defining hch2 volume now'
      call gsvolu ( 'hch2', 'BOX ', nt_air,
     x                  hch2_size, ndim, iv_targ )
c
      if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        hch2 geometry setup failed'
          stop
      end if
c
c
```

```
      write(6,*)'defining airb volume now'
      call gsvolu ( 'airb', 'BOX ', nt_air,
     x                  airb_size, ndim, iv_targ )
c
      if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        airb geometry setup failed'
          stop
      end if
c
c
      write(6,*)'defining scintillator now'
      call gsvolu ( 'sci1', 'BOX ', nt_sci,
     x                  sci1_size, ndim, iv_targ )
c
      write(6,*)iv_targ
      if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        sci1 geometry setup failed'
          stop
      end if
c
c
      write(6,*)'defining airc volume now'
      call gsvolu ( 'airc', 'BOX ', nt_air,
     x                  airc_size, ndim, iv_targ )
c
      if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        airc geometry setup failed'
          stop
      end if
c
c
      write(6,*)'defining the second scintillator now'
      call gsvolu ( 'sci2', 'BOX ', nt_sci,
     x                  sci2_size, ndim, iv_targ )
c
      write(6,*)iv_targ
      if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        sci2 geometry setup failed'
          stop
      end if
c
c
      write(6,*)'defining aird volume now'
      call gsvolu ( 'aird', 'BOX ', nt_air,
     x                  aird_size, ndim, iv_targ )
c
      if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        aird geometry setup failed'
          stop
      end if
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      write(6,*)'DEFINING THE FIRST ANALYZER NOW!'
      call gsvolu ( 'anl1', 'BOX ', nt_analyz,
     x                  anl1_size, ndim, iv_targ )
c
      write(6,*)nt_analyz,anl1_size,iv_targ
c
      if ( iv_targ.le.0 ) then
```

```
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        analyzer geometry setup failed'
          stop
        end if
c
c
          write(*,*)'analyzer 1 thickness =       ',anl1_size(3)*2.0
          write(*,*)'nt_sci =             ',nt_sci
          write(*,*)'nt_ch2 =             ',nt_ch2
          write(*,*)'nt_analyz =          ',nt_analyz
c
c
        write(6,*)'defining aire volume now'
        call gsvolu ( 'aire', 'BOX ', nt_air,
     x                  aire_size, ndim, iv_targ )
c
        if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        aire geometry setup failed'
          stop
        end if
c
c
        write(6,*)'defining fch1 volume now'
        call gsvolu ( 'fch1', 'BOX ', nt_air,
     x                  fch1_size, ndim, iv_targ )
c
        if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        fch1 geometry setup failed'
          stop
        end if
c
c
        write(6,*)'defining airf volume now'
        call gsvolu ( 'airf', 'BOX ', nt_air,
     x                  airf_size, ndim, iv_targ )
c
        if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        airf geometry setup failed'
          stop
        end if
c
c
        write(6,*)'defining fch2 volume now'
        call gsvolu ( 'fch2', 'BOX ', nt_air,
     x                  fch2_size, ndim, iv_targ )
c
        if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        fch2 geometry setup failed'
          stop
        end if
c
c
        write(6,*)'defining airg volume now'
        call gsvolu ( 'airg', 'BOX ', nt_air,
     x                  airg_size, ndim, iv_targ )
c
        if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        airg geometry setup failed'
          stop
        end if
```

```
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      write(6,*)'DEFINING THE SECOND ANALYZER NOW!'
      call gsvolu ( 'anl2', 'BOX ', nt_analyz,
     x                   anl2_size, ndim, iv_targ )
c
      write(6,*)nt_analyz,anl2_size,iv_targ
c
      if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        anl2 geometry setup failed'
          stop
      end if
c
c
        write(*,*)'analyzer 2 thickness =       ',anl2_size(3)*2.0
        write(*,*)'nt_sci =             ',nt_sci
        write(*,*)'nt_ch2 =             ',nt_ch2
        write(*,*)'nt_analyz =          ',nt_analyz
c
c
      write(6,*)'defining airh volume now'
      call gsvolu ( 'airh', 'BOX ', nt_air,
     x                   airh_size, ndim, iv_targ )
c
      if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        airh geometry setup failed'
          stop
      end if
c
c
      write(6,*)'defining fch3 volume now'
      call gsvolu ( 'fch3', 'BOX ', nt_air,
     x                   fch3_size, ndim, iv_targ )
c
      if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        fch3 geometry setup failed'
          stop
      end if
c
c
      write(6,*)'defining airi volume now'
      call gsvolu ( 'airi', 'BOX ', nt_air,
     x                   airi_size, ndim, iv_targ )
c
      if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        airi geometry setup failed'
          stop
      end if
c
c
      write(6,*)'defining fch4 volume now'
      call gsvolu ( 'fch4', 'BOX ', nt_air,
     x                   fch4_size, ndim, iv_targ )
c
      if ( iv_targ.le.0 ) then
          write ( 6,* ) ' ugeom: error, iv_targ=',iv_targ
          write ( 6,* ) '        fch4 geometry setup failed'
          stop
      end if
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
```

```
c
c
c  make the target "SEEN" in the pictures and give it COLOr number 3
c  (probably green) and then position it inside the mother volume
c
      call gsatt ( 'aira', 'SEEN', 1 )
      call gsatt ( 'aira', 'COLO', 1 )
      call gspos ( 'aira', 1, 'HALL', 0.,0.,real(zaira),irot,'only' )
      call gsatt ( 'airb', 'SEEN', 1 )
      call gsatt ( 'airb', 'COLO', 1 )
      call gspos ( 'airb', 1, 'HALL', 0.,0.,real(zairb),irot,'only' )
      call gsatt ( 'airc', 'SEEN', 1 )
      call gsatt ( 'airc', 'COLO', 1 )
      call gspos ( 'airc', 1, 'HALL', 0.,0.,real(zairc),irot,'only' )
      call gsatt ( 'aird', 'SEEN', 1 )
      call gsatt ( 'aird', 'COLO', 1 )
      call gspos ( 'aird', 1, 'HALL', -5.08,0.,real(zaird),irot,'only' )
      call gsatt ( 'aire', 'SEEN', 1 )
      call gsatt ( 'aire', 'COLO', 1 )
      call gspos ( 'aire', 1, 'HALL',-5.08,0.,real(zaire),irot,'only' )
      call gsatt ( 'airf', 'SEEN', 1 )
      call gsatt ( 'airf', 'COLO', 1 )
      call gspos ( 'airf', 1, 'HALL',-5.08,0.,real(zairf),irot,'only' )
      call gsatt ( 'airg', 'SEEN', 1 )
      call gsatt ( 'airg', 'COLO', 1 )
      call gspos ( 'airg', 1, 'HALL',-5.08,0.,real(zairg),irot,'only' )
      call gsatt ( 'airh', 'SEEN', 1 )
      call gsatt ( 'airh', 'COLO', 1 )
      call gspos ( 'airh', 1, 'HALL',-5.08,0.,real(zairh),irot,'only' )
      call gsatt ( 'airi', 'SEEN', 1 )
      call gsatt ( 'airi', 'COLO', 1 )
      call gspos ( 'airi', 1, 'HALL',-5.08,0.,real(zairi),irot,'only' )
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      call gsatt ( 'anl1', 'SEEN', 1 )
      call gsatt ( 'anl1', 'COLO', 4 )
      call gspos ( 'anl1', 1, 'HALL', -5.08,0.,real(zanl1),irot,'only' )
      call gsatt ( 'anl2', 'SEEN', 1 )
      call gsatt ( 'anl2', 'COLO', 4 )
      call gspos ( 'anl2', 1, 'HALL', -5.08,0.,real(zanl2),irot,'only' )
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      call gsatt ( 'hch1', 'SEEN', 1 )
      call gsatt ( 'hch1', 'COLO', 2 )
      call gspos ( 'hch1', 1, 'HALL', 0.,0.,real(zhch1),irot,'only' )
      call gsatt ( 'hch2', 'SEEN', 1 )
      call gsatt ( 'hch2', 'COLO', 2 )
      call gspos ( 'hch2', 1, 'HALL', 0.,0.,real(zhch2),irot,'only' )
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      call gsatt ( 'fch1', 'SEEN', 1 )
      call gsatt ( 'fch1', 'COLO', 5 )
      call gspos ( 'fch1', 1, 'HALL', -5.08,0.,real(zfch1),irot,'only' )
      call gsatt ( 'fch2', 'SEEN', 1 )
      call gsatt ( 'fch2', 'COLO', 5 )
      call gspos ( 'fch2', 1, 'HALL', -5.08,0.,real(zfch2),irot,'only' )
```

```
      call gsatt ( 'fch3', 'SEEN', 1 )
      call gsatt ( 'fch3', 'COLO', 5 )
      call gspos ( 'fch3', 1, 'HALL', -5.08,0.,real(zfch3),irot,'only' )
      call gsatt ( 'fch4', 'SEEN', 1 )
      call gsatt ( 'fch4', 'COLO', 5 )
      call gspos ( 'fch4', 1, 'HALL', -5.08,0.,real(zfch4),irot,'only' )
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      call gsatt ( 'sci1', 'SEEN', 1 )
      call gsatt ( 'sci1', 'COLO', 3 )
      call gspos ( 'sci1', 1, 'HALL', 0.,0.,real(zsci1),irot,'only' )
      call gsatt ( 'sci2', 'SEEN', 1 )
      call gsatt ( 'sci2', 'COLO', 3 )
      call gspos ( 'sci2', 1, 'HALL', 0.,0.,real(zsci2),irot,'only' )
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      write(6,*) 'everything is positioned'
```

A.2 Source Code for Calculations

This is part of the file that is executed for each event. All of the important variables: drift distances, θ, ϕ are calculated. Code for the cone tests is also here.

```
      subroutine get_wire_numbers(xa,ya,za,xb,yb,zb,nhu,nhx,nhv,
     %     nu1,nx1,nv1,nu2,nx2,nv2)
      implicit none
      real*8 xa,ya,za,xb,yb,zb
      integer*4 nu1,nx1,nv1
      integer*4 nu2,nx2,nv2
      integer*4 nhu,nhx,nhv
      real*8 zc,zu,zx,zv,zt,xp,yp,xu,xx,xv,yu,yx,yv,uw,xw,vw
      real*8 anu,anx,anv
      nu2=0
      nx2=0
      nv2=0
c
c We have the (x,y,z) coordinates of the entrance (a) and exit (b) points
c of the track.  We can use this information to calculate the wire numbers that
c were hit in each plane.
c
      zc=(zb-za)/2.0+za
      zu=zc-1.60
      zx=zc
      zv=zc+1.60
      zt=(zb-za)
      xp=(xb-xa)/zt
      yp=(yb-ya)/zt
c
c Project to the FRONT of the "cell" associated with each plane.
c
      xu=xa+xp*(zu-za-0.8)
      yu=ya+yp*(zu-za-0.8)
      xx=xa+xp*(zx-za-0.8)
      yx=ya+yp*(zx-za-0.8)
      xv=xa+xp*(zv-za-0.8)
      yv=ya+yp*(zv-za-0.8)
c
c      xu=xa
c      yu=ya
c      xx=xa
c      yx=ya
c      xv=xa
c      yv=ya
c
      uw=(xu+yu)/sqrt(2.0)
      xw=xx
      vw=(-xv+yv)/sqrt(2.0)
c
c      write(*,*)'********************'
c      write(*,*)'A: ',xa,ya,za
c      write(*,*)'B: ',xb,yb,zb
c      write(*,*)'U: ',xu,yu,zu
c      write(*,*)'X: ',xx,yx,zx
c      write(*,*)'V: ',xv,yv,zv
c      write(*,*)'W: ',uw,xw,vw
c      write(*,*)'********************'
c
      anu=(-uw-3.592+104.0)/2.0
      anx=(-xw-5.080+84.0)/2.0
      anv=(vw-3.592+104.0)/2.0
c
c      write(*,*)'Wires: ',anu,anx,anv
```

```
c      write(*,*)'********************'
       nu1=anu
       nv1=anv
       nx1=anx
       if((anu-nu1).ge.0.500)nu1=nu1+1
       if((anx-nx1).ge.0.500)nx1=nx1+1
       if((anv-nv1).ge.0.500)nv1=nv1+1
c      write(*,*)'using front of chamber: ',nu1,nx1,nv1
c
c Now project to the BACK of the "cell" associated with each plane.
c
       xu=xa+xp*(zu-za+0.8)
       yu=ya+yp*(zu-za+0.8)
       xx=xa+xp*(zx-za+0.8)
       yx=ya+yp*(zx-za+0.8)
       xv=xa+xp*(zv-za+0.8)
       yv=ya+yp*(zv-za+0.8)
c
c      xu=xb
c      yu=yb
c      xx=xb
c      yx=yb
c      xv=xb
c      yv=yb
c
       uw=(xu+yu)/sqrt(2.0)
       xw=xx
       vw=(-xv+yv)/sqrt(2.0)
c
c      write(*,*)'********************'
c      write(*,*)'A: ',xa,ya,za
c      write(*,*)'B: ',xb,yb,zb
c      write(*,*)'U: ',xu,yu,zu
c      write(*,*)'X: ',xx,yx,zx
c      write(*,*)'V: ',xv,yv,zv
c      write(*,*)'W: ',uw,xw,vw
c      write(*,*)'********************'
c
       anu=(-uw-3.592+104.0)/2.0
       anx=(-xw-5.080+84.0)/2.0
       anv=(vw-3.592+104.0)/2.0
c
c      write(*,*)'Wires: ',anu,anx,anv
c      write(*,*)'********************'
       nu2=anu
       nv2=anv
       nx2=anx
       if((anu-nu2).ge.0.500)nu2=nu2+1
       if((anx-nx2).ge.0.500)nx2=nx2+1
       if((anv-nv2).ge.0.500)nv2=nv2+1
       nhu=1
       nhx=1
       nhv=1
       if(nu1.ne.nu2)nhu=2
       if(nx1.ne.nx2)nhx=2
       if(nv1.ne.nv2)nhv=2
       return
       end
       subroutine get_drift_distance_ejb(xa,ya,za,xb,yb,zb,
      &           nu,nx,nv,distu,distx,distv,idflag)
c
c Author: Ed Brash - December 15th, 2005
c Yet another attempt at a full drift distance calculation
c
       implicit none
```

```
c
      real*8 xa,ya,za,xb,yb,zb
      real*8 distu,distx,distv,xp,yp,zc
      integer*4 nu,nx,nv
      real*8 uw,xw,vw
      real*8 zt,xt,yt
      real*8 xu,yu,zu
      real*8 xv,yv,zv
      real*8 xx,yx,zx
      logical idflag
c
c     the direction of each of the lines
c     vect1=track   vectu=u wire
      real*8 vect1(1:3), vectu(1:3)
      real*8 vectx(1:3), vectv(1:3)
c     the difference vector between the defining points
      real*8 du(1:3),dx(1:3),dv(1:3)
c
c     the normal vector to both lines
      real*8 normu(1:3)
      real*8 normx(1:3)
      real*8 normv(1:3)
      real*8 normumag,normxmag,normvmag
c
c     the coefficients of the distance vector
      real*8 dvectu(1:4)
      real*8 dvectx(1:4)
      real*8 dvectv(1:4)
c
      idflag=.false.
      vect1(1)=xb-xa
      vect1(2)=yb-ya
      vect1(3)=zb-za
c
      zu=(zb+za)/2.0-1.60
      zv=(zb+za)/2.0+1.60
      zx=(zb+za)/2.0
c
c     use line number to calculate distance relative to
c     wire plane, then convert to x and y
c      write(*,*)'nu nx nv',nu,nx,nv
      uw=-2.0*nu-3.592+104.0
      xw=-2.0*nx-5.080+84.0
      vw=2.0*nv+3.592-104.0
      xu=uw/sqrt(2.0)
      yu=uw/sqrt(2.0)
      xx=xw
      yx=0
      xv=-vw/sqrt(2.0)
      yv=vw/sqrt(2.0)
c      write(*,*)'uw xu yu zu',uw,xu,yu,zu
c      write(*,*)'xw xx yx zx',xw,xx,yx,zx
c      write(*,*)'vw xv yv zv',vw,xv,yv,zv
c
c     define direction vector for wires, will be the same
c     for each wire in a given plane, and is known
c     for each plane
      vectu(1)=1.0/sqrt(2.0)
      vectu(2)=-1.0/sqrt(2.0)
      vectu(3)=0.0
      vectx(1)=0.0
      vectx(2)=1.0
      vectx(3)=0.0
      vectv(1)=1.0/sqrt(2.0)
```

```
      vectv(2)=1.0/sqrt(2.0)
      vectv(3)=0.0
c
c      write(*,*)'distance calculations .....'
c      write(*,*)xa,ya,za
c      write(*,*)vect1(1),vect1(2),vect1(3)
c      write(*,*)xu,yu,zu
c      write(*,*)vectu(1),vectu(2),vectu(3)
c      write(*,*)xx,yx,zx
c      write(*,*)vectx(1),vectx(2),vectx(3)
c      write(*,*)xv,yv,zv
c      write(*,*)vectv(1),vectv(2),vectv(3)
c      write(*,*)'distance calculations .....'
c
c     cross product
      normu(1)=vect1(2)*vectu(3)-vect1(3)*vectu(2)
      normu(2)=vect1(3)*vectu(1)-vect1(1)*vectu(3)
      normu(3)=vect1(1)*vectu(2)-vect1(2)*vectu(1)
      normx(1)=vect1(2)*vectx(3)-vect1(3)*vectx(2)
      normx(2)=vect1(3)*vectx(1)-vect1(1)*vectx(3)
      normx(3)=vect1(1)*vectx(2)-vect1(2)*vectx(1)
      normv(1)=vect1(2)*vectv(3)-vect1(3)*vectv(2)
      normv(2)=vect1(3)*vectv(1)-vect1(1)*vectv(3)
      normv(3)=vect1(1)*vectv(2)-vect1(2)*vectv(1)
c      write(*,*)normu(1),normu(2),normu(3)
c
      normumag=sqrt(normu(1)**2+normu(2)**2+normu(3)**2)
      normxmag=sqrt(normx(1)**2+normx(2)**2+normx(3)**2)
      normvmag=sqrt(normv(1)**2+normv(2)**2+normv(3)**2)
      normu(1)=normu(1)/normumag
      normu(2)=normu(2)/normumag
      normu(3)=normu(3)/normumag
      normx(1)=normx(1)/normxmag
      normx(2)=normx(2)/normxmag
      normx(3)=normx(3)/normxmag
      normv(1)=normv(1)/normvmag
      normv(2)=normv(2)/normvmag
      normv(3)=normv(3)/normvmag
c      write(*,*)normumag
c
      du(1)=xa-xu
      du(2)=ya-yu
      du(3)=za-zu
      dx(1)=xa-xx
      dx(2)=ya-yx
      dx(3)=za-zx
      dv(1)=xa-xv
      dv(2)=ya-yv
      dv(3)=za-zv
c
c
c     distance formula
      distu=du(1)*normu(1)+du(2)*normu(2)+du(3)*normu(3)
      distx=dx(1)*normx(1)+dx(2)*normx(2)+dx(3)*normx(3)
      distv=dv(1)*normv(1)+dv(2)*normv(2)+dv(3)*normv(3)
c
      if(distu.gt.1.0.or.distx.gt.1.0.or.distv.gt.1.0)
     &            idflag=.true.
      if(distu.gt.1.28.or.distx.gt.1.28.or.distv.gt.1.28)then
        write(*,*)'Problem Child !!!'
        write(*,*)'distance calculations .....'
```

```
        write(*,*)xa,ya,za
c        write(*,*)vect1(1),vect1(2),vect1(3)
        write(*,*)xu,yu,zu
c        write(*,*)vectu(1),vectu(2),vectu(3)
        write(*,*)xx,yx,zx
c        write(*,*)vectx(1),vectx(2),vectx(3)
        write(*,*)xv,yv,zv
c        write(*,*)vectv(1),vectv(2),vectv(3)
c        write(*,*)'normalization factors'
c        write(*,*)normu(1),normu(2),normu(3)
c        write(*,*)normumag
c        write(*,*)normx(1),normx(2),normx(3)
c        write(*,*)normxmag
c        write(*,*)normv(1),normv(2),normv(3)
c        write(*,*)normvmag
c        write(*,*)'Drift distance: ',distu, distx, distv
      endif
c
      return
      end
      subroutine calc_theta_phi(xin1,yin1,zin1,xin2,yin2,zin2,
     &     xsc1,ysc1,zsc1,xsc2,ysc2,zsc2,
     &     theta,phi)
c
      implicit none
      include 'fpp_local.h'
      include 'geant_local.h'
c
      real*8 xin1,yin1,zin1,xin2,yin2,zin2
      real*8 xsc1,ysc1,zsc1,xsc2,ysc2,zsc2,theta,phi
      real*8 ftheta, fphi, fpsi
      real*8 lin,lout,theta_ejb,phi_ejb
c
      real invect(1:3)
      real scvect(1:3)
      real scvect2(1:3)
      real in(1:3)
      real out(1:3)
      real scat(1:3)
c
      invect(1)=xin2-xin1
      invect(2)=yin2-yin1
      invect(3)=zin2-zin1
      scvect(1)=xsc2-xsc1
      scvect(2)=ysc2-ysc1
      scvect(3)=zsc2-zsc1
c      write(*,*)'INCOMING: ',invect(1),invect(2),invect(3)
c      write(*,*)'SCATTERED: ',scvect(1),scvect(2),scvect(3)
c
c EJB calculation of theta and phi
c
      in(1)=invect(1)/invect(3)
      in(2)=invect(2)/invect(3)
      in(3)=invect(3)/invect(3)
      out(1)=scvect(1)/scvect(3)
      out(2)=scvect(2)/scvect(3)
      out(3)=scvect(3)/scvect(3)
      lin=sqrt(in(1)**2+in(2)**2+in(3)**2)
      lout=sqrt(out(1)**2+out(2)**2+out(3)**2)
      scat(1)=out(1)-in(1)
      scat(2)=out(2)-in(2)
      scat(3)=out(3)
      x_ejb=scat(1)
```

```
      y_ejb=scat(2)
      z_ejb=scat(3)
      if(scat(1).ge.0.0.and.scat(2).gt.0.0)then
         phi_ejb=atan(scat(1)/scat(2))*57.2957795
      else if(scat(1).ge.0.0.and.scat(2).lt.0.0)then
         phi_ejb=atan(scat(1)/scat(2))*57.2957795+180.00
      else if(scat(1).le.0.0.and.scat(2).lt.0.0)then
         phi_ejb=atan(scat(1)/scat(2))*57.2957795+180.00
      else if(scat(1).le.0.0.and.scat(2).gt.0.0)then
         phi_ejb=atan(scat(1)/scat(2))*57.2957795+360.00
      endif
c
      theta_ejb=acos((in(1)*out(1)+in(2)*out(2)+in(3)*out(3))/
     &     (lin*lout))*57.2957795
c      write(*,*)'EJB Incoming Vector = ',in(1),in(2),in(3)
c      write(*,*)'EJB Outgoing Vector = ',out(1),out(2),out(3)
c      write(*,*)'EJB Scattered Vector = ',scat(1),scat(2),scat(3)
c      write(*,*)'EJB Thetas = ',theta_ejb,phi_ejb
c
c end EJB calculation
c
      ftheta=acos(invect(3)/sqrt(invect(1)**2+invect(3)**2))
      fphi=acos(invect(3)/sqrt(invect(2)**2+invect(3)**2))
      fpsi=acos(sqrt(invect(2)**2+invect(3)**2)/sqrt(invect(1)**2
     &     +invect(2)**2+invect(3)**2))
c
c      write(*,*)'ftheta, fphi, fpsi',
c     &    ftheta*57.296,fphi*57.296,fpsi*57.296
      scvect2(1)=scvect(1)*cos(fpsi)-sin(fpsi)*(scvect(2)*sin(fphi)
     &     +scvect(3)*cos(fphi))
      scvect2(2)=scvect(2)*cos(fphi)-scvect(3)*sin(fphi)
      scvect2(3)=scvect(1)*sin(fpsi)+cos(fpsi)*(scvect(2)*sin(fphi)
     &     +scvect(3)*cos(fphi))
c
c      write(*,*)'SCATTERED 2: ',scvect2(1),scvect2(2),scvect2(3)
      theta=atan(sqrt(scvect2(1)**2+scvect2(2)**2)/scvect2(3))*57.2957795
      phi=atan(scvect2(1)/scvect2(2))*57.2957795
      if (scvect2(1).lt.0.0.and.scvect2(2).gt.0.0)
     &  phi=phi+360.00
      if (scvect2(1).lt.0.0.and.scvect2(2).lt.0.0)
     &  phi=phi+180.00
      if (scvect2(1).gt.0.0.and.scvect2(2).lt.0.0)
     &  phi=phi+180.00
c      write(*,*)'Theta,phi =',theta,phi
      theta=theta_ejb
      phi=phi_ejb
c
      return
      end
      subroutine calc_zclose_sclose(x0f,y0f,tphif,tthetaf,
     &            x0r,y0r,tphir,tthetar,zclose,sclose)
      real*8 x0f,y0f,tphif,tthetaf
      real*8 x0r,y0r,tphir,tthetar
      real*8 zclose,sclose
      real*8 term1,term2,term3,term4,term5,term6
      real*8 rbig
      rbig=1.0e15
      term1=(x0r-x0f)*(tphir-tphif)
      term2=(y0r-y0f)*(tthetar-tthetaf)
      term3=(tphir-tphif)**2
```

```
      term4=(tthetar-tthetaf)**2
      if((term3+term4).ne.0) then
         zclose=-(term1+term2)/(term3+term4)
      else
         zclose=rbig
      endif
      term5=(x0r-x0f+(tphir-tphif)*zclose)
      term6=(y0r-y0f+(tthetar-tthetaf)*zclose)
      sclose=sqrt(term5**2+term6**2)
      return
      end
      subroutine cone_test(xin1,yin1,zin1,xin2,yin2,zin2,
     &     theta,phi,zclose,zplane)
      implicit none
      include 'fpp_local.h'
      include 'geant_local.h'
      real*8 xin1,yin1,zin1,xin2,yin2,zin2
      real*8 theta,phi,zclose,zplane,h,r
      real*8 xproj,yproj,xclose,yclose
      real*8 xlong,xshort
      iconef=1
c
c     Calculate the incident particle projection on the
c     back of the chamber using two points to define
c     the incident track and the z pos of the plane
c
      xproj=((zplane-zin1)*(xin2-xin1))/(zin2-zin1)+xin1
      yproj=((zplane-zin1)*(yin2-yin1))/(zin2-zin1)+yin1
      write(*,*)zplane,xin1,yin1,zin1,xin2,yin2,zin2
      write(*,*)'xproj: ',xproj
      write(*,*)'yproj: ',yproj
c
c     Calculate the x and y of the scattering point given
c     the z of the scattering point
c
      xclose=((zclose-zin1)*(xin2-xin1))/(zin2-zin1)+xin1
      yclose=((zclose-zin1)*(yin2-yin1))/(zin2-zin1)+yin1
      write(*,*)'xclose: ',xclose
      write(*,*)'yclose: ',yclose
      write(*,*)'zclose: ',zclose
      write(*,*)'zplane: ',zplane
      write(*,*)'theta: ',theta
c
c     Calculate the height of the cone
      h=sqrt((xproj-xclose)**2+(yproj-yclose)**2+(zplane-zclose)**2)
c
c     Calculate the radius of the cone
      r=h*tan(theta*0.0174532925)
c
      write(*,*)'height of cone= ',h
      write(*,*)'radius of cone= ',r
c
c     Check to see if any part of the distribution is
c     outside of the chamber dimensions
c
      if((xproj+r)>(41.5-5.08))then
          write(*,*)'over x'
          iconef=0
      endif
      if((xproj-r)<(-41.5-5.08))then
          write(*,*)'under x'
          iconef=0
      endif
      if((yproj+r)>33.5)then
```

```
            write(*,*)'over y '
            iconef=0
        endif
        if((yproj-r)<-33.5)then
            write(*,*)'under y'
            iconef=0
        endif
c
        return
        end
```