Effect of Hyperons on Pion Asymmetries Measured in the Q_{weak} Experiment

A thesis submitted in partial fulfillment of the requirement
for the degree of Bachelor of Science in Physics from
The College of William and Mary

by

Marcus Sundar Starman

Accepted for HONORS
(Honors or no-Honors)

Dr. Wouter Deconinck, Director

Gina L. Hoatson
Dr. Gina Hoatson, Physics

Dr. Joshua Erlich, Physics

Lawrence Leemis
Dr. Lawrence Leemis, Mathematics

Williamsburg, VA
May 5, 2015

Effect of Hyperons on Pion Asymmetries Measured in the Q_{weak} Experiment

Marcus Sundar Starman

May 5, 2015

Abstract

Experimental nuclear physics has largely been concerned with testing the Standard Model of particle physics. The Q_{weak} experiment was designed to measure the weak interaction force of the proton. The experiment was also used to make asymmetry measurements on pions produced by inelastic collisions. The asymmetry measured in the experiment for pions was larger than expected at lower momentums given the kinematic conditions. A possible cause for the larger asymmetry is the production of hyperons, which decay into pions that can be detected. This idea was investigated, and an acceptance function for these pions was formulated. The acceptance function took the form of a four dimensional vector structure based on the probability of the pions being detected based on their initial kinetic energy, θ angle, ϕ angle, and z position.

1 Introduction

The Standard Model of particle physics was formulated in the 1970s. This model is the governing theory for the different types of particles and their interactions. The model provides for a small number of particles that are the building blocks of all matter; six leptons and six quarks. Leptons are the electron, muon, and tauon, and each has a corresponding neutrino. Quarks are up and down, charm and strange, and top and bottom. These basic particles make up other subatomic particles, such as two up quarks and a down quark make a proton, and two down quarks and an up quark make a neutron, as seen in Figure 1.

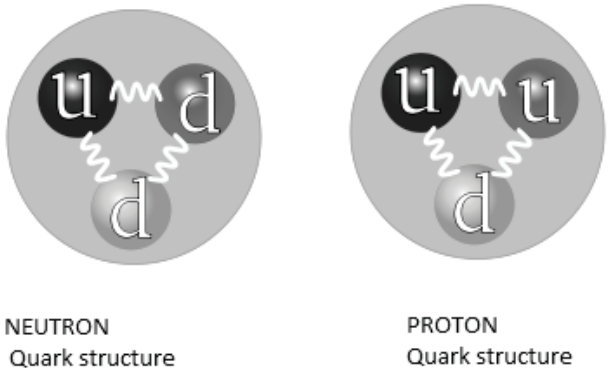


Figure 1: A graphic of the quark structure of the proton and the neutron, showing the quark flavors present in each particle.

The Standard Model describes the interactions of these basic particles, and the particles that they construct. There are four types of fundamental interactions; strong, electromagnetic, weak, and gravitational [4]. The strong interaction binds quarks together in protons and neutrons, and binds protons and neutrons together in the nuclei; these interactions are mediated by gluons. Electromagnetic interactions bind electrons with nuclei, and dominate the intermolecular forces in liquids and solids; these interactions are mediated by photon

exchange. Weak interactions are responsible for radioactive decay, including β-decay, or the emission by a proton or neutron of an electron and neutrino, and the decay of one particle into another; the mediators are the W and Z bosons. The weak interaction is parity dependent, which means it is dependent upon the spins of the interacting particles, specifically because it violates parity. This is in contrast to the electromagnetic interaction, which can be dependent on spin states but conserves parity symmetry. Gravitational interactions exist between all particles, but on the scale of particle physics it is dominated by the other interactions, because gravitational interactions are mass dependant; the graviton is thought to be the mediator.

Q_{weak} was an experiment designed to directly measure the weak interaction of the proton through elastic electron scattering [3]. This experiment was conducted in Hall C at the Thomas Jefferson National Accelerator Facility. The setup consisted of a liquid hydrogen (LH_2) target, collimators, a toroidal magnet, and a ring of eight detectors. An incoming electron beam from the accelerator would scatter off of the protons in the LH_2 target. The electron beam was run at 1.16 GeV, and could be set to a helicity of $+1$ or -1, setting a spin for the incoming electron to parallel or anti-parallel. The electrons would scatter elastically off of the protons, and then went through a series of collimators and were redirected by a magnetic field before being picked up by a ring of eight detectors, consisting of eight Cherenkov bars with a Photomultiplier Tube (PMT) on either end of the bar [6] (see Figure 2). The quartz Cherenkov bars are a scintillating material, meaning that when particles such as pions interact with the material, photons in the form of Cherenkov radiation are released. When a photon reaches the PMT, it causes an electron to be knocked loose, and the signal exponentially increases as it moves through the PMT, generating a large signal at the end.

Using the condition that photoelectrons be detected in both PMTs for a detector for a single event, I was able to filter out useful events from noise. Q_{weak} was able to measure the weak

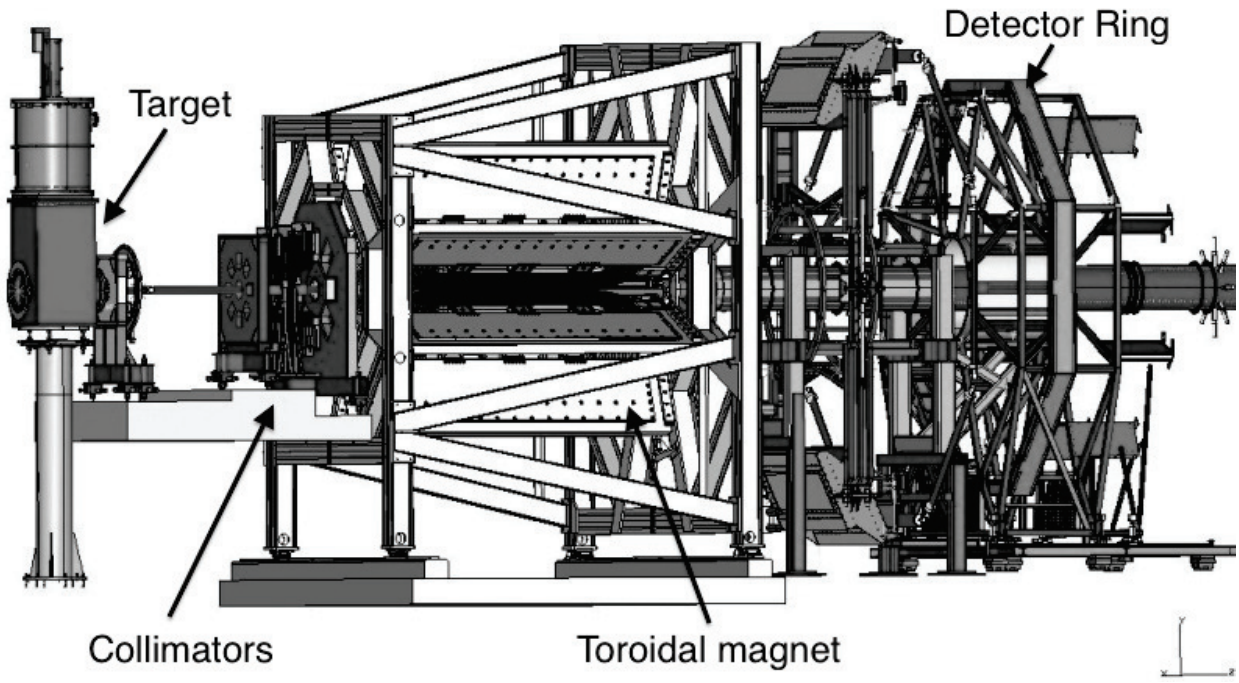


Figure 2: The set up for the Q_{weak} experiment. The liquid hydrogen target, collimators, magnet, and detectors are labeled [5].

interaction force of the proton to a greater accuracy than previous experiments, and was also the first experiment to make this measurement directly. The experiment functioned as a Standard Model test by detecting electrons that scattered elastically off of protons. 'Elastic' scattering means that momentum and kinetic energy are conserved in the scattering interaction. Protons and electrons interact through both the electromagnetic interaction and the weak interaction. The electromagnetic interaction is parity conserving; parity is the idea that when particles interact, the interaction does not depend on the spin state of the particles. Specifically, if we switch from a left-handed to a right-handed coordinate system, the spin

vector of the particle stays the same, as depicted in Figure 3. However, the weak interaction violates parity. To pick out the weak interaction from the magnetic interaction, the helicity of the incoming beam of electrons was switched between -1 and $+1$, polarized along the z axis. Between the two helicities, the differences in the scattering could only be caused by the weak interaction.

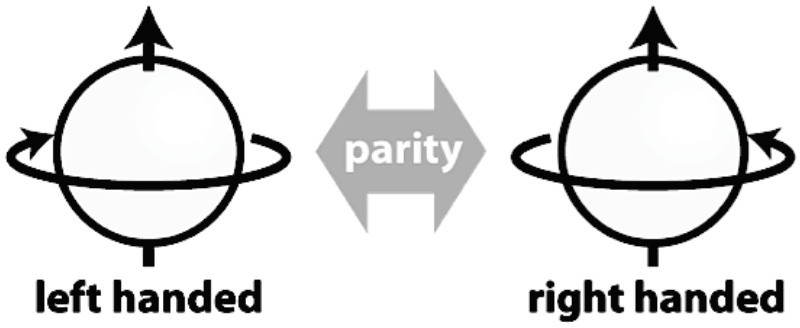


Figure 3: Depiction of parity conservation when switching from a left to a right-handed coordinate system. The vertical arrow is the angular momentum vector, representing the spin of the particle.

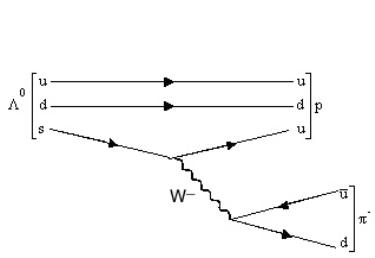
Q_{weak} was also run with a beam energy of 3.35 GeV, to measure an asymmetry more closely related to proton structure. This causes the electrons to exhibit inelastic scattering off of the protons, where kinetic energy is not conserved, causing some of the energy to go into pion creation. The experiment was set to detect pions, and when the momentum cutoff for detection was lowered by a factor of three, large pion asymmetries were measured by the detectors. The asymmetry can be calculated by looking at the small differences in the momentum distribution between the two helicities. The asymmetry is calculated using:

$$A = \left(\frac{\rho_{\text{helicity}+1} - \rho_{\text{helicity}-1}}{\rho_{\text{helicity}+1} + \rho_{\text{helicity}-1}} \right), \tag{1}$$

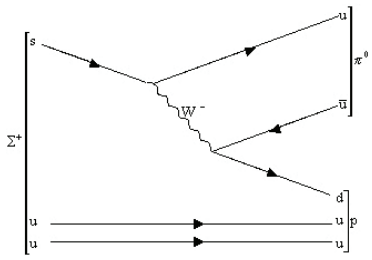
where ρ is the momentum distribution. This equation gives the asymmetry because the electron-proton electromagnetic interaction is the same for both helicities, so any momentum difference would have to be due to the weak interaction. To actually find the asymmetry, I used histograms of the momentum distribution and applied Equation 1. During the experiment, the strength of the magnetic field was varied to include pions of different energies. The correlation between the pion's momentum and the magnetic field current is that higher energy particles require a stronger magnetic field for the particle's path to be bent to the detectors, while this will push lower energy particles outside of the ring of detectors. Alternatively, lower energy particles require a weaker magnetic field for detection, and this will allow higher energy particles to continue almost straight through the experiment, in the middle of the detector ring.

Depending on the energy of the incoming beam, various other particles can be produced, including hyperons. Hyperons are a type of particle consisting of three quarks, characterized by having at least one strange quark. The lowest energy hyperons, Λ hyperons, go through spontaneously weak decay into a proton and a π^-, or a neutron and a π^0. Λ hyperons have a lifetime on the order of 10^{-10} seconds, through the decay of the strange quark into an up or down quark (Figure 4). The momentum of these pions is dependent upon the helicity of the incoming beam, as well as the spin state of the individual hyperon. Hyperons are self analyzing, meaning that the spin angular momentum directly affects the momentum vector of the decay particles. These factors have the potential to cause large discrepancies in the pion asymmetry with the production of only a few hyperons, because the pions are only affected by the weak interaction.

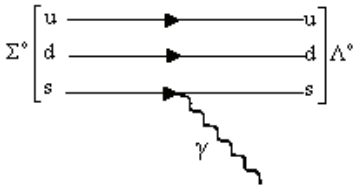
My research attempted to explain the large pion asymmetry at lower momentums through



(a) Λ^0 hyperon decay to a proton and a π^-.



(b) Σ^+ hyperon decay to a proton and a π^0.



(c) Σ^0 hyperon decay to a Λ^0 hyperon and a photon.

Figure 4: Feynman diagrams of the weak decay of various hyperons [1].

the production of hyperons that decay into pions that are detected and affect the asymmetry measurements. Once I established that hyperons could produce the asymmetries seen in the Q_{weak} data, I worked on creating an acceptance function. The acceptance function used the probability of a pion being detected based on its position, energy, and polar angles, and combined that with the hyperon cross-section that would produce that particular pion. By combining the probability of a pion being detected with the probability of that pion being produced, we will be able to determine the effect of hyperons on the Q_{weak}'s auxiliary results. I am organizing the acceptance function in the form of a look-up table of vectors contained within other vectors.

2 Experiment

I used a hyperon generator code written in C++ by Dr. Konrad Aniol of California State University, Los Angeles, to conduct my verification for the potential impact of hyperons on Q_{weak} measurements [2]. His program simulated the production of hyperons using given input parameters; including the type of hyperon to be produced, the incoming beam energy, constants β and γ which are defined by the beam energy, and the radiation length of the target. The output file would then go into another program which simulated the decay of the hyperons into pions, and I specifically chose π^-. I only simulated π^- because only negatively charged particles are directed into the detectors. This output file had the x, y, and z components of the position and momentum, as well as the cross-section of the hyperon that produced that pion. These output files were generated for the three lowest-energy hyperons, lambda (Λ^0), sigma plus (Σ^+), and sigma zero (Σ^0). Dr. Aniol's code had to be edited to have the correct experiment parameters for Q_{weak}, and different variables had to be activated for the different types of hyperons and for the changing beam helicity. Plots were created using ROOT, a data analysis framework that uses scripts or macros written in C++. My scripts imported the output files, created a tree to hold the data, then used the tree and histogram manipulation to produce the final asymmetry plot. The plots for Λ^0 hyperons are contained in Figure 5, plots for Σ^+ hyperons are contained in Figure 6, and plots for Σ^0 hyperons are contained in Figure 7. The large error bars seen at higher energies for all three hyperon asymmetry plots are due to low statistics at those energies, as evidenced by the momentum distribution plots.

I used the server farm at Jefferson Laboratory to run simulations based on Q_{weak} exper-

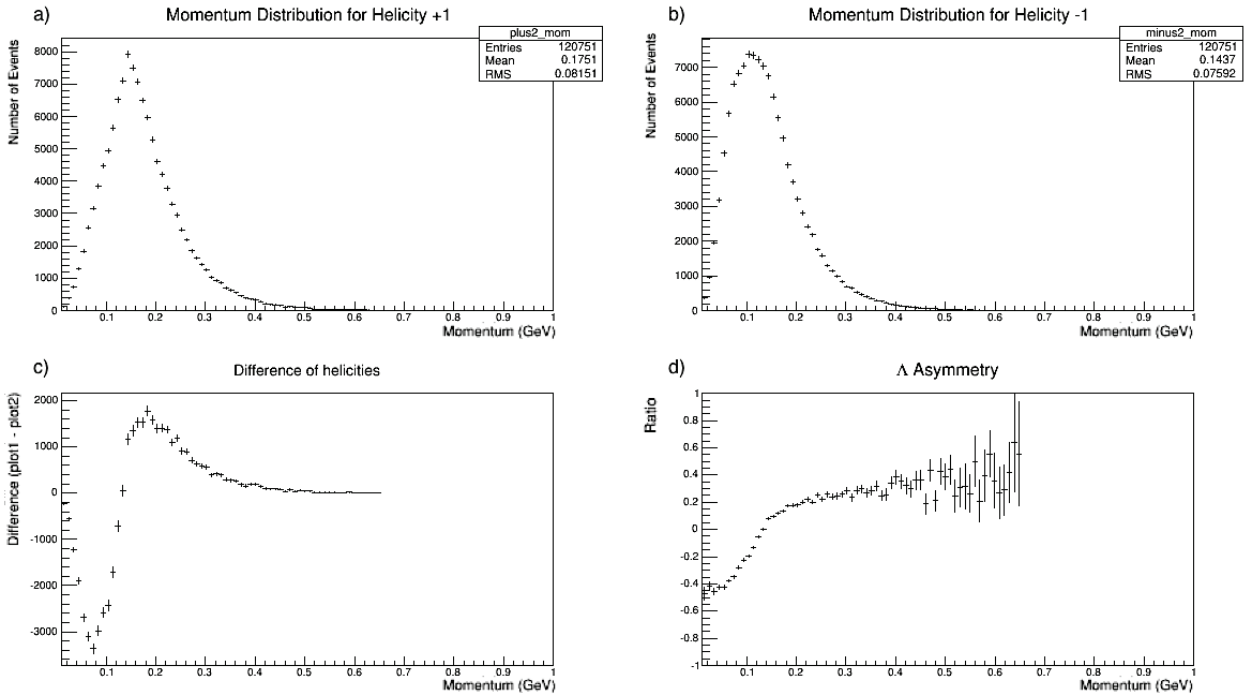


Figure 5: Plots made using pions produced by Λ^0 hyperons. Plots a) and b) are the momentum distribution of the pions generated by Dr. Aniol's code using the Q_{weak} parameters when the helicities of the incoming beam are $+1$ and -1 respectively. Plot c) shows the difference in the momentum histograms, which was an intermediary step towards the final plot, plot d), which shows the asymmetry as calculated using Equation 1. The large error bars in plot d) of the asymmetry at higher energies are due to low statistics at those energies, which can be confirmed by looking at plots a) and b) of the momentum distributions.

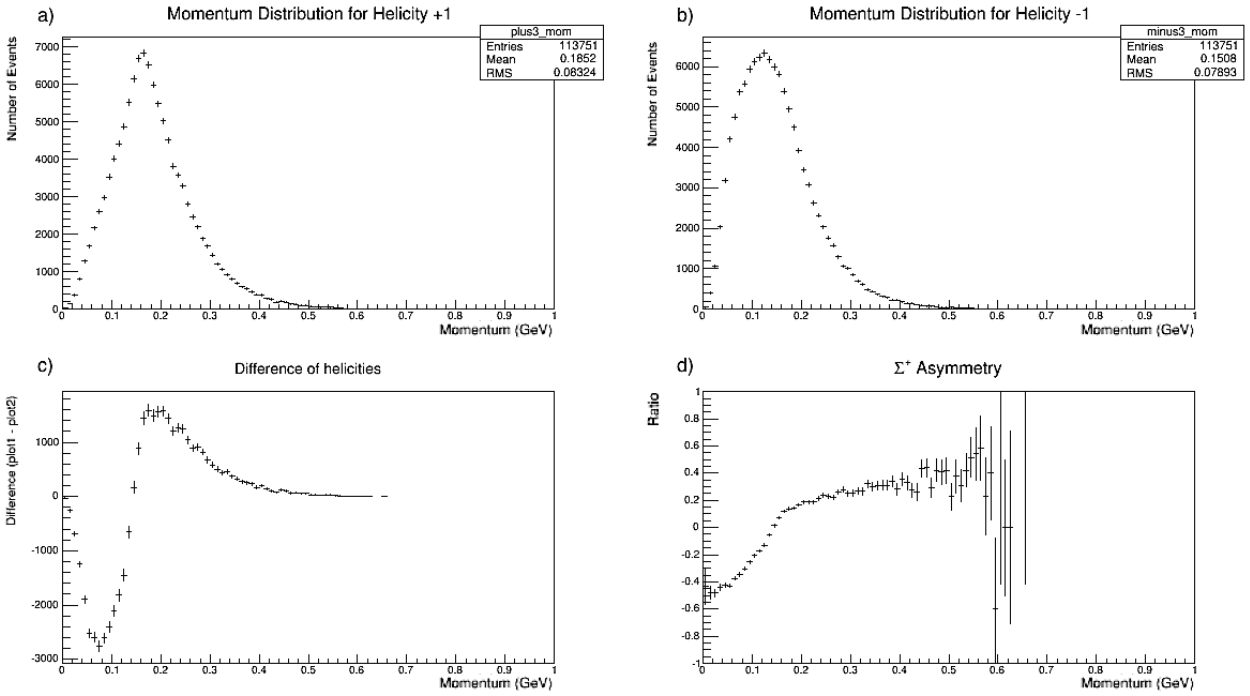


Figure 6: Plots made using pions produced by Σ^+ hyperons. Plots a) and b) are the momentum distribution of the pions generated by Dr. Aniol's code using the Q_{weak} parameters when the helicities of the incoming beam are $+1$ and -1 respectively. Plot c) shows the difference in the momentum histograms, which was an intermediary step towards the final plot, which shows the asymmetry as calculated using Equation 1. The large error bars in plot d) of the asymmetry at higher energies are due to low statistics at those energies, which can be confirmed by looking at the plots a) and b) of the momentum distributions.

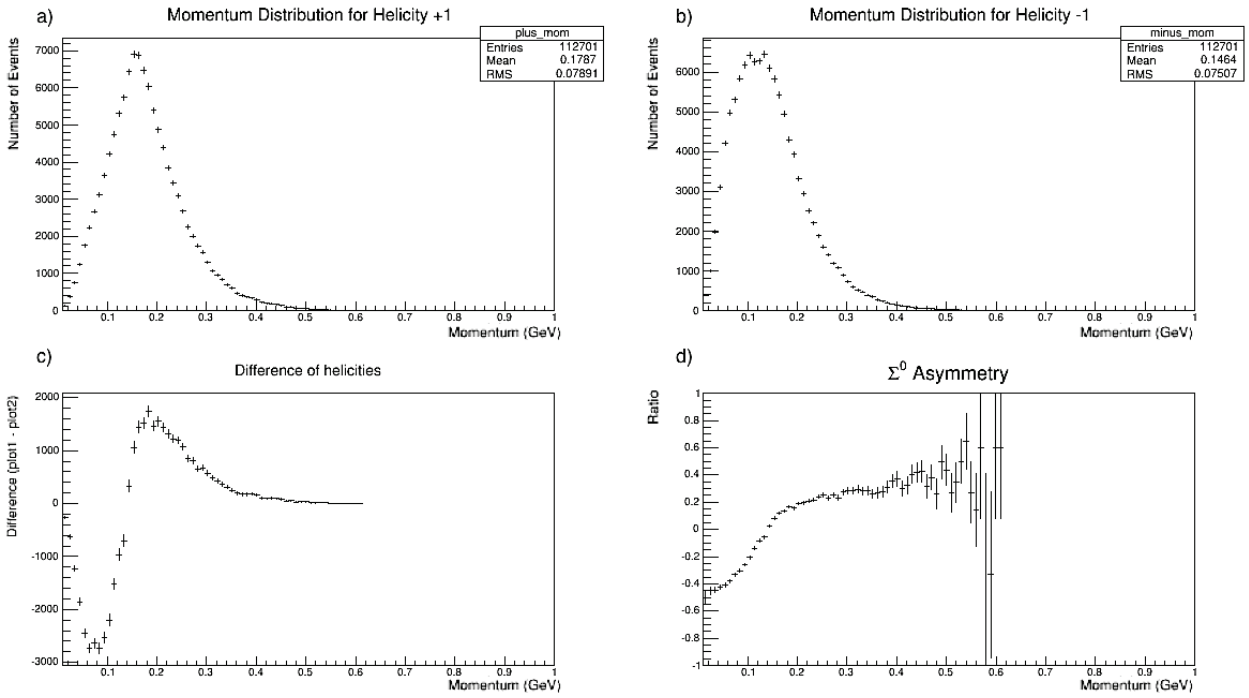
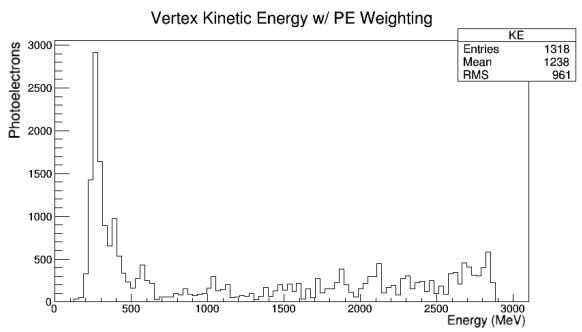


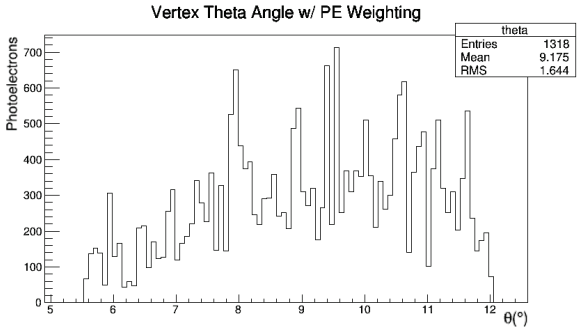
Figure 7: Plots made using pions produced by Σ^0 hyperons. Plots a) and b) are the momentum distribution of the pions generated by Dr. Aniol's code using the Q_{weak} parameters when the helicities of the incoming beam are $+1$ and -1 respectively. Plot c) shows the difference in the momentum histograms, which was an intermediary step towards the final plot, plot d), which shows the asymmetry as calculated using Equation 1. The large error bars in plot d) of the asymmetry at higher energies are due to low statistics at those energies, which can be confirmed by looking at the plots a) and b) of the momentum distributions.

imental parameters. The simulation was set up to generate pions inside the liquid hydrogen target over a given range of energies and angles. Initially, I simulated events over an energy range (E') of 200 to 3000 MeV and an angle range (θ) of 5.5 to 12 degrees. The energy range was chosen because pions have a rest mass of 140 MeV, so 200 MeV is a good minimum for creating pions with a low kinetic energy, while the upper limit of 3000 MeV corresponds to the incoming beam energy of 3350 MeV. The θ range was chosen because it was thought to cover most of the range of pions that would make it through the collimators. We call the process of simulating pions 'throwing' because pions are simulated randomly within the specified ranges, so the simulation 'throws the dice' over those ranges to determine exactly what pion is simulated. Pions were thrown evenly over $\cos(\theta)$, instead of θ, because if we threw evenly in θ, there would be too many pions simulated at small angles. Throwing flat in $\cos(\theta)$ keeps the distribution isotropic. The current in the toroidal magnet was set to 3000 A for these simulations, based on the asymmetry plots from Dr. Aniol's modified hyperon generator. The magnetic current operating range for the Q_{weak} experiment was 1000 to 9000 A, so 3000 A is well within the operating range. Those plots showed strong asymmetries for pions in the 200 to 400 MeV range, and a magnetic field of 3000 A directs pions in that energy range into the detectors. For the first script, I had the simulation record event information when the event was detected by the Cherenkov detectors. For an event to count as detected, it had to create photoelectrons that were registered by the PMTs. The plots in Figure 8 showed a peak around 300 MeV as expected, but the θ distribution indicated that the simulated range was not large enough; the tail behavior seemed to be cut off at 12°.

The acceptance function was intended to be a function of four variables, using the pion's initial energy, z position, θ angle, and ϕ angle, taking the probability of a pion being de-



(a) Kinetic energy distribution weighted by the number of photoelectrons produced by each event. The peak around 300 MeV is clearly visible.



(b) θ distribution weighted by the number of photoelectrons produced by each event. Near the edges, especially at 12°, there appears to be a sharp cutoff.

Figure 8: Plots from a simulation that only recorded event information if the event was detected by the Cherenkov detectors (meaning a photoelectron was registered in the PMTs). Additional cuts were used based on photoelectrons being registered in both PMTs of a single Cherenkov bar, and weighted based on the total number of photoelectrons for the event, indicating how strong of a signal the event generated.

tected and combining that with the probability of the pion being created form a hyperon decay. With the information contained in the four variables, any pions generated could be accounted for, and the probability of the pion being accepted by the detectors quantified. Accepted pions were defined as the pion events that generated at least one photoelectron in each detector. The probability of a pion being accepted was calculated by taking the ratio of the accepted pions to the total number of simulated pions. The probability of a pion being created from a hyperon incorporated the hyperon cross section (probability of hyperon existing) along with Dr. Aniol's decay function for the pion (specifically π^-) that is produced

by the hyperon. Multiplying the two probabilities creates the acceptance function, which gives the total probability that a pion produced by a hyperon is detected and impacts the results of the Q_{weak} experiment. We focused on π^- because only negatively charged particles moving through the magnetic field would be directed into the detectors.

The next set of simulation runs used a larger θ range, from 4.5 to 14.5 degrees. All thrown events were saved to the ROOT file to facilitate construction of the acceptance function, because we needed the accepted pions as well as all of the simulated pions. I only simulated over one detector octant, because the results can be extrapolated to the entire detector ring. We needed the number of accepted pions divided by the number of thrown pions to determine the probability that a pion with certain origin variables is registered by the Cherenkov detectors. When determining which pions were accepted, I put a cut on the detected pions forcing an event to have generated a photoelectron in both PMTs for the detector. A 'cut' omits events that do not satisfy the given condition. Pions should cause more than one photoelectron to be created in the detector, and requiring a photoelectron in both detectors for a single event means the photoelectrons were not noise or dark current, but actually a result of a pion. Dark current is the phenomenon in PMTs where there is always a small current, even without incident photons. Through the cutting process, I obtained a histogram for the pions that were accepted, and divided by a histogram of all thrown pions to generate a histogram showing the probability of a pion begin accepted. Histograms were made for the initial kinetic energy, the initial θ angle, the initial ϕ angle, and the initial position along z, as seen in Figure 9. Because the simulation only generated pions over one detector, the plot for the azimuthal angle ϕ only covers the detector, with $\phi = 0$ being the middle of the detector bar.

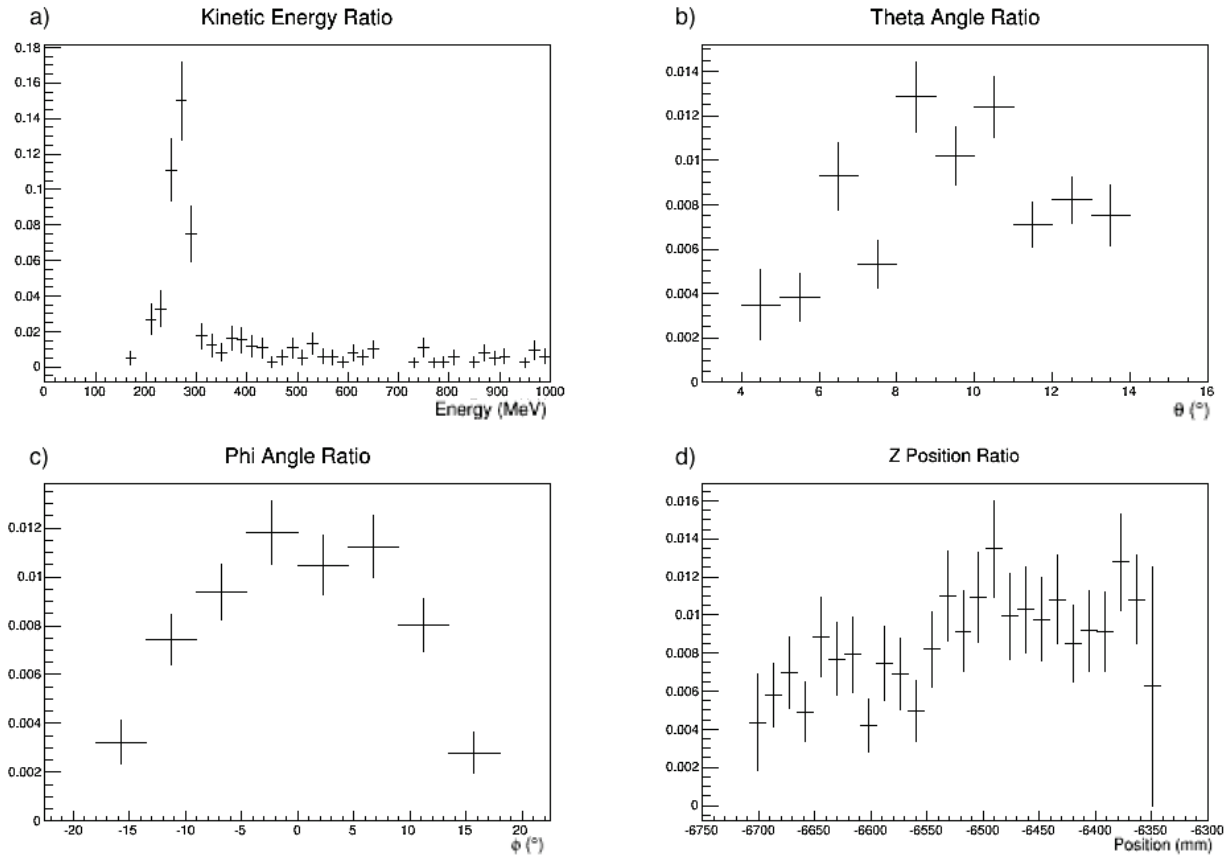


Figure 9: Plots of the acceptance ratios for pions based on the pion's initial kinetic energy (plot a), initial θ angle (plot b), initial ϕ angle (plot c), or initial z position (plot d). The ratio was calculated by using the histogram of the pions that made it to the detector (as determined by those events that had at least one photoelectron in both of the PMTs for the same event) and dividing it by the histogram of all of the simulated pions. The error bars are large for these histograms due to a low number of simulated events causing low statistics.

Again, the peak around 300 MeV is very clear, though it seems to be closer to 280 MeV. All of the plots in Figure 9 have an issue with low statistics, as evidenced by the large error bars. Running a simulation with 100,000 events results in the simulating generating 50,000 pions within the target. Out of these 50,000 pions, only 433 caused photoelectrons in both of the PMTs and are regarded as a detected event. To raise the statistics to an acceptable level, 1,000 simulations were run with 100,000 events in each simulation. Concatenating the simulations gave about 50 million simulated pions, and we could proceed with the acceptance function given the statistical improvement. The number of pions necessary for statistical significance was approximated based on the number and size of desired histogram bins for each of the four variables.

Similar plots to Figure 9 were made for the 50 million simulated pions, as seen in Figure 10. High statistical significance can be seen from the small error bars on the histogram bins. These simulations finally produced appropriately small uncertainties to continue with creating an acceptance function.

The acceptance function took the form of vectors within vectors, with a total of four vectors. By arranging the function this way, it works like a look-up table for simulated pions. From Dr. Aniol's code, the hyperon decay script outputs the initial position and momentum of the pion in x, y, and z components, and the cross-section for the pion production. From the momentum components, the initial kinetic energy, θ angle, and ϕ angle can be calculated for the pion. With the four variables defined, each pion can be put through the four-dimensional table to compute the probability of the pion being detected.

The outermost vector used the initial kinetic energy bins, then nested in the kinetic energy vector were initial θ angle bins, and nested inside the θ vectors were the initial

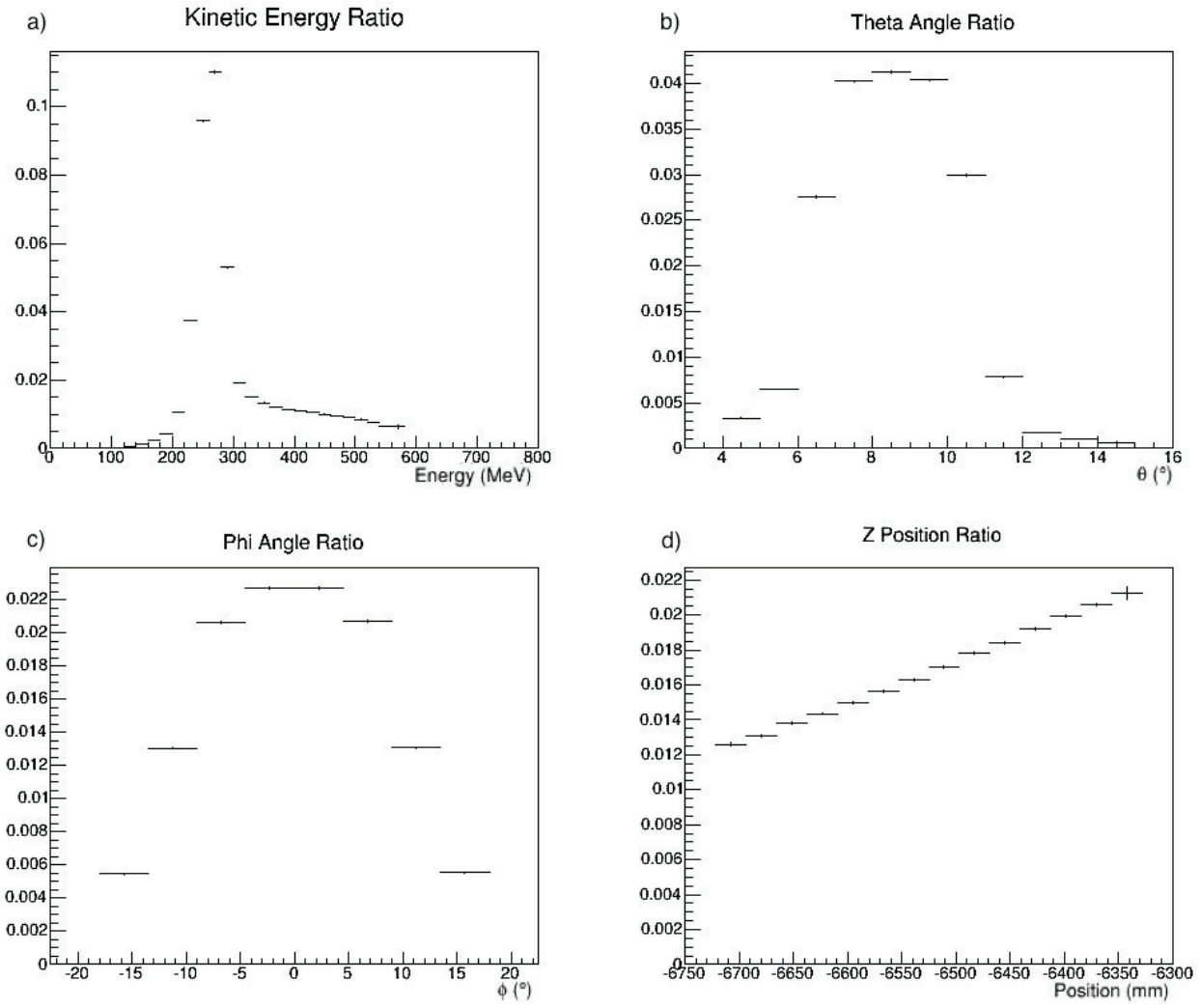


Figure 10: Plots of the acceptance ratios for pions based on the pion's initial kinetic energy (plot a), initial θ angle (plot b), initial ϕ angle (plot c), or initial z position (plot d). The ratio was calculated by using the histogram of the pions that made it to the detector (as determined by those events that had at least one photoelectron in both of the PMTs for the same event) and dividing it by the histogram of all of the simulated pions (approximately 50 million) for each variable separately.

ϕ angle vectors, and finally nested within the ϕ vectors were the initial z position vectors. The contents of the innermost vector are the probabilities of a pion being accepted based on the number of thrown pions based on the specific kinetic energy, θ angle, ϕ angle, and z position bin that the individual entry is nested. I used 25 bins at 20 MeV steps for the kinetic energy, 10 bins at 1° steps for the θ angle, 10 bins at 4.5° steps for the ϕ angle, and 14 bins at 25 mm steps for the z position. There is no good graphical representation of this four-dimensional vector structure. The best graphical concept of the structure is a three-dimensional coordinate system with the x, y, and z axes corresponding to kinetic energy (25 vertices), θ angle (10 vertices), and ϕ angle (10 vertices) respectively, and the intersections of the bins for the three variables containing the probability of acceptance. This three-dimensional structure would then be repeated 14 times for each z position bin. Some example output of the bins with non-zero probabilities is shown in Figure 11, to give a better idea of the set up of the vector structure. It is possible to see the nesting of the z position vector within the ϕ angle vector, nested within the θ angle vector, nested within the kinetic energy vector. The probabilities in the far left column are accurate probabilities from the 50 million simulated events. The example range I chose gives some of the highest probabilities for pion acceptance, as predicted by the plots from Figure 10.

0.3898	from 260 to 280 MeV,	8 to 9 degrees,	-4.5 to 0 degrees,	-6700 to -6675 mm
0.3968	from 260 to 280 MeV,	8 to 9 degrees,	-4.5 to 0 degrees,	-6675 to -6650 mm
0.3989	from 260 to 280 MeV,	8 to 9 degrees,	-4.5 to 0 degrees,	-6650 to -6625 mm
0.3776	from 260 to 280 MeV,	8 to 9 degrees,	-4.5 to 0 degrees,	-6625 to -6600 mm
0.3823	from 260 to 280 MeV,	8 to 9 degrees,	-4.5 to 0 degrees,	-6600 to -6575 mm
0.3956	from 260 to 280 MeV,	8 to 9 degrees,	-4.5 to 0 degrees,	-6575 to -6550 mm
0.3972	from 260 to 280 MeV,	8 to 9 degrees,	-4.5 to 0 degrees,	-6550 to -6525 mm
0.3725	from 260 to 280 MeV,	8 to 9 degrees,	-4.5 to 0 degrees,	-6525 to -6500 mm
0.3891	from 260 to 280 MeV,	8 to 9 degrees,	-4.5 to 0 degrees,	-6500 to -6475 mm
0.4118	from 260 to 280 MeV,	8 to 9 degrees,	-4.5 to 0 degrees,	-6475 to -6450 mm
0.4255	from 260 to 280 MeV,	8 to 9 degrees,	-4.5 to 0 degrees,	-6450 to -6425 mm
0.4244	from 260 to 280 MeV,	8 to 9 degrees,	-4.5 to 0 degrees,	-6425 to -6400 mm
0.454	from 260 to 280 MeV,	8 to 9 degrees,	-4.5 to 0 degrees,	-6400 to -6375 mm
0.4169	from 260 to 280 MeV,	8 to 9 degrees,	-4.5 to 0 degrees,	-6375 to -6350 mm
0.353	from 260 to 280 MeV,	8 to 9 degrees,	0 to 4.5 degrees,	-6700 to -6675 mm
0.3657	from 260 to 280 MeV,	8 to 9 degrees,	0 to 4.5 degrees,	-6675 to -6650 mm
0.3843	from 260 to 280 MeV,	8 to 9 degrees,	0 to 4.5 degrees,	-6650 to -6625 mm
0.3965	from 260 to 280 MeV,	8 to 9 degrees,	0 to 4.5 degrees,	-6625 to -6600 mm
0.3979	from 260 to 280 MeV,	8 to 9 degrees,	0 to 4.5 degrees,	-6600 to -6575 mm
0.3934	from 260 to 280 MeV,	8 to 9 degrees,	0 to 4.5 degrees,	-6575 to -6550 mm
0.424	from 260 to 280 MeV,	8 to 9 degrees,	0 to 4.5 degrees,	-6550 to -6525 mm
0.4114	from 260 to 280 MeV,	8 to 9 degrees,	0 to 4.5 degrees,	-6525 to -6500 mm
0.4045	from 260 to 280 MeV,	8 to 9 degrees,	0 to 4.5 degrees,	-6500 to -6475 mm
0.4063	from 260 to 280 MeV,	8 to 9 degrees,	0 to 4.5 degrees,	-6475 to -6450 mm
0.4284	from 260 to 280 MeV,	8 to 9 degrees,	0 to 4.5 degrees,	-6450 to -6425 mm
0.4349	from 260 to 280 MeV,	8 to 9 degrees,	0 to 4.5 degrees,	-6425 to -6400 mm
0.4341	from 260 to 280 MeV,	8 to 9 degrees,	0 to 4.5 degrees,	-6400 to -6375 mm
0.4406	from 260 to 280 MeV,	8 to 9 degrees,	0 to 4.5 degrees,	-6375 to -6350 mm

Figure 11: Example of how the four dimensional vector structure is organized. The nesting of the vectors (z position within ϕ angle, ϕ angle within θ angle, θ angle within kinetic energy) can be seen. The probability of pion acceptance is contained in the far left column. These probabilities are accurate for the simulations, calculated using 50 million events. The range was chosen to show bins with some of the highest acceptance probabilities.

3 Analysis

From the final asymmetry plots, it is clear that hyperons can cause a large affect on the asymmetry. At lower momentums (<350 MeV) strong asymmetries are seen, and the hyperon asymmetry plots have small error bars in this range indicating good statistics. At higher momentums there are not enough simulated pions, so the asymmetry varies from point to point due to low statistics, as indicated by the large error bars. A pion with a momentum of 280 MeV corresponds to the magnetic field produced by a 3000 A current, which is well within the parameters of the Q_{weak} experiment.

Simulations were run with the Q_{weak} experiment parameters, utilizing ranges from the asymmetry study. From the initial plots, a peak for the kinetic energy of the pions is seen around 300 MeV. Based on the 3000 A magnetic current used in the simulation, that peak is expected. There is an unusually large tail at higher energies, but that could be caused by noise from higher energy pions triggering photoelectrons in the detectors. Higher energy pions generate more photoelectrons as they pass through the Cherenkov bars, and cause a larger signal in the PMTs.

Using a simulation in which all of the events were saved made it possible to determine the ratio of accepted pions. Multiple simulations, totaling approximately 50 million events, were needed to achieve a statistical significance. The acceptance plots show high ratios around kinetic energies from 240 to 280 MeV, with a peak probability of 0.11 from 260 to 280 MeV, and θ angles of 7 to 10 degrees with a peak probability of 0.041 from 8 to 9 degrees. The ϕ angle plot shows a higher acceptance at the middle of the detector bar with a peak probability of 0.0225 from -5 to 5 degrees, which drops evenly as the angle

moves to the extremities. This ϕ distribution makes sense based on the collimators and the toroidal magnet set up, and considering that the simulation only ran over one octant. The z dependance appears to be linear, increasing through the target from a probability of 0.0125 where the electron beam first enters to 0.021 at the end of the target.

Based on the ranges of the peak probability for each of the four variables, the corresponding probability in the acceptance function look up table is 0.4406. This means that 44% of the pions produced in that specific bin are detected by the experiment.

4 Conclusion

Using code that generates and decays hyperons, it was possible to calculate the pion asymmetry caused by hyperons. The plots show that it is plausible that the production of a few hyperons could have caused the pion asymmetry measured by Q_{weak} to be off, possibly on the order of thousands, as seen in the actual data.

A ratio for pion acceptance has been generated with small error bars, and the probability for a pion being detected has been formed into a four-dimensional vector structure. Acceptance for pions produced for Dr. Aniol's hyperon decay simulation can be calculated using the vector structure within the variable ranges defined within the vector structure.

The vector structure now needs to be expanded over pion simulations with different magnetic field currents, as all of my simulations ran at 3000 A. Simulations also need to be conducted over all eight detector octants, to examine the possibility of octant dependance for pion acceptance. It is possible that the acceptance ratios change slightly in each detector octant.

A similar acceptance function will be used with the MOLLER experiment, but will have to be changed to reflect MOLLER experimental parameters. MOLLER will attempt to measure the asymmetry in electron-electron (Møller) scattering. MOLLER will also be conducted at Jefferson Laboratory. This will require a more robust acceptance function, which will need to be developed.

5 Acknowledgements

I would like to thank Dr. Wouter Deconinck for being my research advisor, and allowing me to work in his lab. The experience gained has been invaluable to my college experience. I would also like to Dr. Mark Dalton for being a mentor throughout this project, and helping to give direction and guidance to my work.

I would also like to thank Dr. Konrad Aniol for the use of his hyperon generator simulation and hyperon decay simulation, which were integral in developing the acceptance function for pions produced by hyperons in the Q_{weak} experiment.

Many thanks to Ms. Valerie Gray, Mr. James Dowd, and Mr. Kurtis Bartlett for their assistance with C++ and ROOT, and navigating the server farm at Jefferson Laboratory.

Thank you to Dr. Joshua Erlich, Dr. Lawrence Leemis, Dr. Gina Hoatson, and Dr. Wouter Deconinck for agreeing to be my examining committee.

This work was supported in part by the National Science Foundation under Grant Nos. PHY-1206053.

References

[1] High School Teachers at CERN. *Examples of particle interactions described in terms of Feynman vertices* CERN. Accessed March 2015. http://teachers.web.cern.ch/teachers/archiv/HST2002/feynman/examples.htm.

[2] Aniol, Konrad. *Index of /∼aniol/moller/scripts* Accessed September 2014. https://userweb.jlab.org/∼aniol/moller/scripts/.

[3] Mammei, Juliette M. *Parity-Violating Elastic Electron Nucleon Scattering: Measurement of the Strange Quark Content of the Nucleon and Towards a Measurement of the Weak Charge of the Proton* 2010: Virginia Polytechnic Institute and State University.

[4] Perkins, Donald H. *Introduction to High Energy Physics* 2000: Cambridge University Press, N.Y.

[5] Rajotte, Jean-Francois. *Measurement of the Proton's Weak Charge and the Qweak Experiment* 2011: Qweak Collaboration. JLAB-PHY-11-1447.

[6] Wang, Peiqing. *A Measurement of the Proton's Weak Charge Using an Integration Cerenkov Detector System* 2011: University of Manitoba (Canada).

A Simulation Macro

The macro that was run on Jefferson Laboratory's server farm to simulate the 1000 runs of 100,000 events. The seeds, name, and job ID were set in the script that made the actual submission. As can be seen in the code, simulations were only made in octant 5, and setting the ReactionType to 88 only simulated pions.

```
#Programmer: Marcus Starman
#Purpose: The macro file for finding the acceptance function for pions
#
#Date: 11-11-2014

/control/execute myQweakConfiguration.mac
/control/execute macros/noTracking.mac

/random/setSavingFlag 1
/random/setDirectoryName random/%name%_%jobid%
/random/setSeeds %seedA% %seedB%

#decide what to track (3-track everything)
/TrackingAction/TrackingFlag 3

#set reaction region and type, and octants to throw in
/EventGen/SelectReactionRegion 1
/EventGen/SelectReactionType 88
/EventGen/SelectOctant 5

#set particle to pi minus
/EventGen/SetThetaMin 4.5 degree
/EventGen/SetThetaMax 15  degree
/EventGen/SetEPrimeMin 150 MeV
/EventGen/SetEPrimeMax 700 MeV
/PrimaryEvent/SetParticleType pi-

/EventGen/SetBeamEnergy 3.35 GeV
/MagneticField/SetActualCurrent 3000 A

/Trigger/Disable cer
/Trigger/Enable all
/Trigger/Show
```

```
#set the rootfile name
/Analysis/RootFileName rootfiles/%name%_%jobid%_workdir/%name%_%jobid%.root

#number of events
/run/beamOn %nevents%
```

B Ratio Plot Constructor

The code that was run in ROOT to create and divide the histograms to make the acceptance ratio plots.

```
/*****************************
Written by: Marcus Starman

This code creates histograms from the 1000 ROOT files that were simulated.
Histograms are for kinetic energy, theta angle, phi angle, and z position.
*****************************/
#include <iostream>
#include <cmath>
#include <vector>

#include <TROOT.h>
#include <TChain.h>
#include <TFile.h>
#include <TTree.h>
#include <TH1.h>
#include <TH2.h>
#include <TStyle.h>
#include <TCanvas.h>

#include "QweakSimUserMainEvent.hh"

void Func()
{
  //Create TChain and chain together all of the necessary ROOT files.
  TChain *chain = new TChain("QweakSimG4_Tree");
  chain->Add("/lustre/expphy/cache/mss/home/mstarman/rootfiles/hyperon_accept\
  /hyperon_accept_*.root");

  QweakSimUserMainEvent *event = 0;
  chain->SetBranchAddress("QweakSimUserMainEvent",&event);

  //Define histograms. Those ending in "a" are accepted events, those ending
  //in "t" are thrown events.
  TH1F *ka = new TH1F("ka","ka",40,0,800);
  TH1F *ta = new TH1F("ta","ta",13,3,16);
  TH1F *pa = new TH1F("pa","pa",10,-22.5,22.5);
  TH1F *za = new TH1F("za","za",16,-6750,-6300);
```

```
TH1F *kt = new TH1F("kt","kt",40,0,800);
TH1F *tt = new TH1F("tt","tt",13,3,16);
TH1F *pt = new TH1F("pt","pt",10,-22.5,22.5);
TH1F *zt = new TH1F("zt","zt",16,-6750,-6300);

//Series of loops to fill histograms with events.
Int_t entries = chain->GetEntries();
for (Int_t i=0; i < entries; i++){
  chain->GetEntry(i);
  if (event->Cerenkov.PMT.NbOfHits>0) {
  if (event->Cerenkov.PMT.PMTLeftNbOfPEs[5]>0 &&
    event->Cerenkov.PMT.PMTRightNbOfPEs[5]>0) {

    ka->Fill(event->Primary.OriginVertexKineticEnergy);
    ta->Fill(event->Primary.OriginVertexThetaAngle);
    za->Fill(event->Primary.OriginVertexPositionZ);

  if (event->Primary.OriginVertexPhiAngle>0) {
    pa->Fill(event->Primary.OriginVertexPhiAngle+(-180));
  }else if (event->Primary.OriginVertexPhiAngle<0) {
    pa->Fill(event->Primary.OriginVertexPhiAngle+180);
  }
  }
}

kt->Fill(event->Primary.OriginVertexKineticEnergy);
tt->Fill(event->Primary.OriginVertexThetaAngle);
zt->Fill(event->Primary.OriginVertexPositionZ);

if (event->Primary.OriginVertexPhiAngle>0) {
  pt->Fill(event->Primary.OriginVertexPhiAngle+(-180));
}else if (event->Primary.OriginVertexPhiAngle<0) {
  pt->Fill(event->Primary.OriginVertexPhiAngle+180);
}
}

//Defining additional histograms for the division to get a ratio.
TH1F *k = (TH1F*)ka->Clone("k");
TH1F *t = (TH1F*)ta->Clone("t");
TH1F *p = (TH1F*)pa->Clone("p");
TH1F *z = (TH1F*)za->Clone("z");

k->Sumw2();
k->Divide(kt);
k->SetTitle("Kinetic Energy Ratio;Energy (MeV)");
```

```
k->SetStats(0);

t->Sumw2();
t->Divide(tt);
t->SetTitle("Theta Ratio;#theta (#circ)");
t->SetStats(0);

p->Sumw2();
p->Divide(pt);
p->SetTitle("Phi Ratio;#phi (#circ)");
p->SetStats(0);

z->Sumw2();
z->Divide(zt);
z->SetTitle("Z Ratio;Position (mm)");
z->SetStats(0);

//Creating TCanvases and drawing histograms with error bars.
TCanvas *c1 = new TCanvas("c1","c1",1500,1000);
c1->Divide(4,2);
c1->cd(1);
ka->SetTitle("Kinetic Energy Accepted;Energy (MeV)");
ka->Draw("e");
c1->cd(2);
ta->SetTitle("Theta Accepted;#theta (#circ)");
ta->SetStats(0);
ta->Draw("e");
c1->cd(3);
pa->SetTitle("Phi Accepted;#phi (#circ)");
pa->SetStats(0);
pa->Draw("e");
c1->cd(4);
za->SetTitle("Z Accepted;Position (mm)");
za->SetStats(0);
za->Draw("e");
c1->cd(5);
kt->SetTitle("Kinetic Energy Thrown;Energy (MeV)");
kt->Draw("e");
c1->cd(6);
tt->SetTitle("Theta Thrown;#theta (#circ)");
tt->SetStats(0);
tt->Draw("e");
c1->cd(7);
pt->SetTitle("Phi Thrown;#phi (#circ)");
pt->SetStats(0);
```

```
  pt->Draw("e");
  c1->cd(8);
  zt->SetTitle("Z Thrown;Position (mm)");
  zt->SetStats(0);
  zt->Draw("e");

  TCanvas *c2 = new TCanvas("c2","c2",1000,1000);
  c2->Divide(2,2);
  c2->cd(1);
  k->Draw("e");
  c2->cd(2);
  t->Draw("e");
  c2->cd(3);
  p->Draw("e");
  c2->cd(4);
  z->Draw("e");

}
```

C Four-dimensional Vector Structure

Code to create the four-dimensional vector structure that becomes the look-up table for pions. Currently, it outputs the non-zero probabilities out of the 35000 unique entries.

```
/******************
written by: Marcus Starman

This code constructs a four-dimensional vector structure that acts as a
look-up table for pions in the Q_weak Experiment. The table is meant to
be combined with the output of Dr. Aniol's hyperon decay function.
******************/
#include <iostream>
#include <cmath>
#include <vector>
#include <iomanip>

#include <TROOT.h>
#include <TChain.h>
#include <TFile.h>
#include <TTree.h>
#include <TH1.h>
#include <TH2.h>
#include <TStyle.h>
#include <TCanvas.h>

#include "QweakSimUserMainEvent.hh"

void VectorFuncShort()
{
  TChain *chain = new TChain("QweakSimG4_Tree");
  chain->Add("/lustre/expphy/cache/mss/home/mstarman/rootfiles/hyperon_accept\
  /hyperon_accept_*.root");

  QweakSimUserMainEvent *event = 0;
  chain->SetBranchAddress("QweakSimUserMainEvent",&event);

  //Define constants
  Int_t entries = chain->GetEntries();

  static const Int_t LENGTH_KE = 25;
  static const Int_t LENGTH_THETA = 10;
  static const Int_t LENGTH_PHI = 10;
```

```
  static const Int_t LENGTH_Z = 14;

  Int_t PhiAngle = 0;
  Int_t index_KE = 0;
  Int_t index_theta = 0;
  Int_t index_phi = 0;
  Int_t index_z = 0;

  //Define vectors -> [KE][theta][phi][z]
  std::vector<std::vector<std::vector<std::vector<Double_t> > > > ACCEPT;
  std::vector<std::vector<std::vector<std::vector<Double_t> > > > THROWN;
  std::vector<std::vector<std::vector<std::vector<Double_t> > > > PROB;

  //Resizing vectors to proper length
  ACCEPT.resize(LENGTH_KE); //[KE]
  THROWN.resize(LENGTH_KE);
  PROB.resize(LENGTH_KE);

  for (Int_t i_KE = 0; i_KE < LENGTH_KE; i_KE++) {
    ACCEPT[i_KE].resize(LENGTH_THETA); //[KE][theta]
    THROWN[i_KE].resize(LENGTH_THETA);
    PROB[i_KE].resize(LENGTH_THETA);

    for (Int_t i_theta = 0; i_theta < LENGTH_THETA; i_theta++) {
      ACCEPT[i_KE][i_theta].resize(LENGTH_PHI); //[KE][theta][phi]
      THROWN[i_KE][i_theta].resize(LENGTH_PHI);
      PROB[i_KE][i_theta].resize(LENGTH_PHI);

      for (Int_t i_phi = 0; i_phi < LENGTH_PHI; i_phi++) {
ACCEPT[i_KE][i_theta][i_phi].resize(LENGTH_Z,0); //[KE][theta][phi][z]
THROWN[i_KE][i_theta][i_phi].resize(LENGTH_Z,0);
PROB[i_KE][i_theta][i_phi].resize(LENGTH_Z,0);
      }
    }
  }

  for (Int_t i=0; i < entries; i++){ //Loop over all entries in the TChain
    chain->GetEntry(i);
    //Fit phi to one detector bar. This works for MD5,
    //because the center of the detector is at 180 degrees.
    if (event->Primary.OriginVertexPhiAngle > 0) {
      PhiAngle = event->Primary.OriginVertexPhiAngle+(-180);
    }else if (event->Primary.OriginVertexPhiAngle < 0) {
      PhiAngle = event->Primary.OriginVertexPhiAngle+180;
    }
```

```
    //Begin the if/if else/else statements to determine correct variable bins.
      //Determine kinetic energy index
if (event->Primary.OriginVertexKineticEnergy > 100 &&
  event->Primary.OriginVertexKineticEnergy < 120) {
  index_KE = 0;
}else if (event->Primary.OriginVertexKineticEnergy > 120 &&
  event->Primary.OriginVertexKineticEnergy < 140) {
  index_KE = 1;
}else if (event->Primary.OriginVertexKineticEnergy > 140 &&
  event->Primary.OriginVertexKineticEnergy < 160) {
  index_KE = 2;
}else if (event->Primary.OriginVertexKineticEnergy > 160 &&
  event->Primary.OriginVertexKineticEnergy < 180) {
  index_KE = 3;
}else if (event->Primary.OriginVertexKineticEnergy > 180 &&
  event->Primary.OriginVertexKineticEnergy < 200) {
  index_KE = 4;
}else if (event->Primary.OriginVertexKineticEnergy > 200 &&
  event->Primary.OriginVertexKineticEnergy < 220) {
  index_KE = 5;
}else if (event->Primary.OriginVertexKineticEnergy > 220 &&
  event->Primary.OriginVertexKineticEnergy < 240) {
  index_KE = 6;
}else if (event->Primary.OriginVertexKineticEnergy > 240 &&
  event->Primary.OriginVertexKineticEnergy < 260) {
  index_KE = 7;
}else if (event->Primary.OriginVertexKineticEnergy > 260 &&
  event->Primary.OriginVertexKineticEnergy < 280) {
  index_KE = 8;
}else if (event->Primary.OriginVertexKineticEnergy > 280 &&
  event->Primary.OriginVertexKineticEnergy < 300) {
  index_KE = 9;
}else if (event->Primary.OriginVertexKineticEnergy > 300 &&
  event->Primary.OriginVertexKineticEnergy < 320) {
  index_KE = 10;
}else if (event->Primary.OriginVertexKineticEnergy > 320 &&
  event->Primary.OriginVertexKineticEnergy < 340) {
  index_KE = 11;
}else if (event->Primary.OriginVertexKineticEnergy > 340 &&
  event->Primary.OriginVertexKineticEnergy < 360) {
  index_KE = 12;
}else if (event->Primary.OriginVertexKineticEnergy > 360 &&
  event->Primary.OriginVertexKineticEnergy < 380) {
  index_KE = 13;
```

```
}else if (event->Primary.OriginVertexKineticEnergy > 380 &&
  event->Primary.OriginVertexKineticEnergy < 400) {
  index_KE = 14;
}else if (event->Primary.OriginVertexKineticEnergy > 400 &&
  event->Primary.OriginVertexKineticEnergy < 420) {
  index_KE = 15;
}else if (event->Primary.OriginVertexKineticEnergy > 420 &&
  event->Primary.OriginVertexKineticEnergy < 440) {
  index_KE = 16;
}else if (event->Primary.OriginVertexKineticEnergy > 440 &&
  event->Primary.OriginVertexKineticEnergy < 460) {
  index_KE = 17;
}else if (event->Primary.OriginVertexKineticEnergy > 460 &&
  event->Primary.OriginVertexKineticEnergy < 480) {
  index_KE = 18;
}else if (event->Primary.OriginVertexKineticEnergy > 480 &&
  event->Primary.OriginVertexKineticEnergy < 500) {
  index_KE = 19;
}else if (event->Primary.OriginVertexKineticEnergy > 500 &&
  event->Primary.OriginVertexKineticEnergy < 520) {
  index_KE = 20;
}else if (event->Primary.OriginVertexKineticEnergy > 520 &&
  event->Primary.OriginVertexKineticEnergy < 540) {
  index_KE = 21;
}else if (event->Primary.OriginVertexKineticEnergy > 540 &&
  event->Primary.OriginVertexKineticEnergy < 560) {
  index_KE = 22;
}else if (event->Primary.OriginVertexKineticEnergy > 560 &&
  event->Primary.OriginVertexKineticEnergy < 580) {
  index_KE = 23;
}else if (event->Primary.OriginVertexKineticEnergy > 580 &&
  event->Primary.OriginVertexKineticEnergy < 600) {
  index_KE = 24;
}else {
  index_KE = 99; //garbage value
}

if (index_KE != 99) { //Determine theta index
  if (event->Primary.OriginVertexThetaAngle > 4 &&
    event->Primary.OriginVertexThetaAngle < 5) {
    index_theta = 0;
  }else if (event->Primary.OriginVertexThetaAngle > 5 &&
    event->Primary.OriginVertexThetaAngle < 6) {
    index_theta = 1;
  }else if (event->Primary.OriginVertexThetaAngle > 6 &&
```

```
  event->Primary.OriginVertexThetaAngle < 7) {
  index_theta = 2;
}else if (event->Primary.OriginVertexThetaAngle > 7 &&
  event->Primary.OriginVertexThetaAngle < 8) {
  index_theta = 3;
}else if (event->Primary.OriginVertexThetaAngle > 8 &&
  event->Primary.OriginVertexThetaAngle < 9) {
  index_theta = 4;
}else if (event->Primary.OriginVertexThetaAngle > 9 &&
  event->Primary.OriginVertexThetaAngle < 10) {
  index_theta = 5;
}else if (event->Primary.OriginVertexThetaAngle > 10 &&
  event->Primary.OriginVertexThetaAngle < 11) {
  index_theta = 6;
}else if (event->Primary.OriginVertexThetaAngle > 11 &&
  event->Primary.OriginVertexThetaAngle < 12) {
  index_theta = 7;
}else if (event->Primary.OriginVertexThetaAngle > 12 &&
  event->Primary.OriginVertexThetaAngle < 13) {
  index_theta = 8;
}else if (event->Primary.OriginVertexThetaAngle > 13 &&
  event->Primary.OriginVertexThetaAngle < 14) {
  index_theta = 9;
}else {
  index_theta = 99; //garbage value
}

if (index_theta != 99) { //Determine phi index
  if (PhiAngle > -22.5 && PhiAngle < -18){
    index_phi = 0;
  }else if (PhiAngle > -18 && PhiAngle < -13.5){
    index_phi = 1;
  }else if (PhiAngle > -13.5 && PhiAngle < -9){
    index_phi = 2;
  }else if (PhiAngle > -9 && PhiAngle < -4.5){
    index_phi = 3;
  }else if (PhiAngle > -4.5 && PhiAngle < 0){
    index_phi = 4;
  }else if (PhiAngle > 0 && PhiAngle < 4.5){
    index_phi = 5;
  }else if (PhiAngle > 4.5 && PhiAngle < 9){
    index_phi = 6;
  }else if (PhiAngle > 9 && PhiAngle < 13.5){
    index_phi = 7;
  }else if (PhiAngle > 13.5 && PhiAngle < 18){
```

```
      index_phi = 8;
    }else if (PhiAngle > 18 && PhiAngle < 22.5){
      index_phi = 9;
    }else {
      index_phi = 99; //garbage value
    }

    if (index_phi != 99) { //Determine z index
      if (event->Primary.OriginVertexPositionZ > -6700 &&
event->Primary.OriginVertexPositionZ < -6675) {
index_z = 0;
      }else if (event->Primary.OriginVertexPositionZ > -6675 &&
event->Primary.OriginVertexPositionZ < -6650) {
index_z = 1;
      }else if (event->Primary.OriginVertexPositionZ > -6650 &&
event->Primary.OriginVertexPositionZ < -6625) {
index_z = 2;
      }else if (event->Primary.OriginVertexPositionZ > -6625 &&
event->Primary.OriginVertexPositionZ < -6600) {
index_z = 3;
      }else if (event->Primary.OriginVertexPositionZ > -6600 &&
event->Primary.OriginVertexPositionZ < -6575) {
index_z = 4;
      }else if (event->Primary.OriginVertexPositionZ > -6575 &&
event->Primary.OriginVertexPositionZ < -6550) {
index_z = 5;
      }else if (event->Primary.OriginVertexPositionZ > -6550 &&
event->Primary.OriginVertexPositionZ < -6525) {
index_z = 6;
      }else if (event->Primary.OriginVertexPositionZ > -6525 &&
event->Primary.OriginVertexPositionZ < -6500) {
index_z = 7;
      }else if (event->Primary.OriginVertexPositionZ > -6500 &&
event->Primary.OriginVertexPositionZ < -6475) {
index_z = 8;
      }else if (event->Primary.OriginVertexPositionZ > -6475 &&
event->Primary.OriginVertexPositionZ < -6450) {
index_z = 9;
      }else if (event->Primary.OriginVertexPositionZ > -6450 &&
event->Primary.OriginVertexPositionZ < -6425) {
index_z = 10;
      }else if (event->Primary.OriginVertexPositionZ > -6425 &&
event->Primary.OriginVertexPositionZ < -6400) {
index_z = 11;
      }else if (event->Primary.OriginVertexPositionZ > -6400 &&
```

```
event->Primary.OriginVertexPositionZ < -6375) {
index_z = 12;
      }else if (event->Primary.OriginVertexPositionZ > -6375 &&
event->Primary.OriginVertexPositionZ < -6350) {
index_z = 13;
      }else {
index_z = 99;
      }
    }
  }
}
//Only fill vectors with "good" events
if (index_KE != 99 && index_theta != 99 && index_phi != 99 && index_z != 99) {
  THROWN[index_KE][index_theta][index_phi][index_z]++;

  if (event->Cerenkov.PMT.NbOfHits>0) {
    if (event->Cerenkov.PMT.PMTLeftNbOfPEs[5]>0 &&
      event->Cerenkov.PMT.PMTRightNbOfPEs[5]>0) { //Accepted pions
      ACCEPT[index_KE][index_theta][index_phi][index_z]++;
    }
  }
}
    }
  //Fill the PROB vector by dividing ACCEPT by THROWN by element
  for (Int_t i_KE = 0; i_KE < LENGTH_KE; i_KE++) {
    for (Int_t i_theta = 0; i_theta < LENGTH_THETA; i_theta++) {
      for (Int_t i_phi = 0; i_phi < LENGTH_PHI; i_phi++) {
for (Int_t i_z = 0; i_z < LENGTH_Z; i_z++) {
  PROB[i_KE][i_theta][i_phi][i_z] = ACCEPT[i_KE][i_theta][i_phi][i_z] /
  THROWN[i_KE][i_theta][i_phi][i_z];
  if (ACCEPT[i_KE][i_theta][i_phi][i_z] == 0){
    PROB[i_KE][i_theta][i_phi][i_z] = 0;
  }
}
      }
    }
  }

  //Display contents of the PROB vector that aren't zero.
  for (Int_t i_KE = 0; i_KE < LENGTH_KE; i_KE++) {
    for (Int_t i_theta = 0; i_theta < LENGTH_THETA; i_theta++) {
      for (Int_t i_phi = 0; i_phi < LENGTH_PHI; i_phi++) {
for (Int_t i_z = 0; i_z < LENGTH_Z; i_z++) {
  if (PROB[i_KE][i_theta][i_phi][i_z] != 0) {
    cout << std::setprecision(4) << PROB[i_KE][i_theta][i_phi][i_z]
```

```
    << "\tfrom " << 100 + i_KE*20 << " to " << 100 + (i_KE + 1)*20
    << " MeV,\t" << 4 + i_theta*1 << " to " << 4 + (i_theta + 1)*1
    << " degrees,     \t" << -22.5 + i_phi*4.5 << " to "
    << -22.5 + (i_phi+1)*4.5 << " degrees,\t" << -6700 + i_z*25
    << " to " << -6700 + (i_z + 1)*25 << " mm" << endl;
  }
}
      }
    }
  }

}
```