PARTICLE-ATMOSPHERE INTERACTION[1]

NM Supercomputing Challenge
Final Report, March 23, 2010

Team 84 McCurdy High

Team Members:

Dennis Trujillo
Oliver Galvan
Brandon Ricci

Project Mentors:

John Pretz
Brenda Dingus
Philip Sanchez

1 Amended from 'Gamma-Muonic Flux Supernovae Correlation Model'

2. Executive Summary

As a result of cosmic radiation interaction with the atmosphere we find that a series of phenomena known as extensive air showers results from the decay of these particles when they make contact with the gases and other elements our atmosphere is composed of. Because each particle decays differently depending on the strength of the forces that bind it we find that air showers produced by different particles produce a different array of daughter particles which if unstable continually break up into other subatomic particles. Because of such we can determine what kind of particle each air shower is produced by based on the pattern of decay left as it approaches ground level and the energy of the particle causing the air shower.

The purpose of our project is to determine the difference between gamma and electromagnetic air showers. We are going to solve this problem with the help of the CORSIKA computational model. The CORSIKA model itself is a Monte Carlo system, that is to say, that it randomly chooses numbers for its plotting. We want to evaluate the distribution of particles at seven thousand feet and first interaction relative to density that the GNUplots produce. All graphs are plotted on a Cartesian coordinate system with the center of the interaction as the origin. If a particle is a proton there will be much more energy towards the center of the interaction. If a particle is a gamma then it will not have as much particles and they will be more spread out towards the center of the interaction. Another means of determining the differences is by evaluating the different daughter particles that each interaction produces. For example if a particle is a proton it will produce muons. If it the particle is gamma then it will produce solely electromagnetic daughter particles. We are not going to launch these results once, but more like a thousand times in order to gain a more accurate understanding of the characteristics associated with air shower events.

3. Introduction

Extensive air shower simulation is a area of study which has great potential relative to the understanding of particle decay and development as evidenced by findings related to particle decay and daughter particle development as a result of air showers resulting from interactions similar to those evaluated within our study. Such studies can result in information relative to the stability of certain particles, the decay rate of these particles, and the creation of daughter particles. Essentially the evaluation of particle interaction within nature is comparable to the work done at CERN with the LHC although with real interactions being taken into consideration. Therefore we find that the study and simulation of extensive air showers is a vital and cheap means of evaluating particle characteristics and the forces which drive the universe as a whole.

The primary goal of several institutions around the country and throughout the world is particle characterization as a result of computer simulation and physical experimentation in order to produce a more full understanding of the components which account for all matter and likewise hold the key to understanding why the universe exists in the means in which it does. These simulations and experiments are integral to one another; each exists to prove or test the other. We find that code can be written according to theory but the true test of its worth can only be found through experimentation, and similarly experiments need to be designed according to certain standards which can be determined through computer simulation.

A companion approach involving the use of both physical experimentation and computer simulation can provide the best means of evaluation. A large scale analysis of data produced by proven code and the implementation of physical experimentation could potentially lead to the association of certain before unknown characteristics with certain particles, a cheap means of particle study yielding the same results as something as complex as the LHC, and potentially provide information relative to why the universe exists as it does and perhaps lead to the discovery of before unknown particles.

3.1 Problem Statement

How can raw data be accurately used to potentially identify the parent particle causing an extensive air shower event, i.e. Hadronic vs. electromagnetic in origin? How can these observations be used to further our knowledge of particle behavior and characteristics?

3.2 Procedure Overview

Our study begins with the acquisition of raw data produced by a series of simulations validated by code for data analysis. Such data is produced by the CORSIKA version 6900 program, a Monte Carlo simulation package which simulates the interaction and break up of certain particles and light nuclei. This data contains a series of parameters relative to particle decay and interaction, including parent particle type, daughter particle types, and energy at ground level interactions of 7000 ft in elevation.

Using this data we then compare the number of particles produced, the types of daughter particles, and energies at ground level to the parent of the original interaction. These are then plotted on a curve in order to determine the fundamental differences between particle interactions and establish traits common to each parent as it decays.

This means of distinguishing particles is commonly used and can potentially be used to classify and determine before unknown characteristics of particles as they decay and interact. We have meshed the CORSIKA version 6900 software with our own coding in order to write a program which can take raw data from any source, in terms of daughter particle types, total energy in GeV or TeV, and number of particles produced and provide a fairly guess as to what particle initiated the interaction in question.

4. Background Research

4.1 General Particle Information

Particles responsible for extensive air showers must have a source. Stars, supernova, black holes, and similar objects produce radiation. This radiation comes in the form of particles. Different particles decay faster than others depending on how stable they are. All particles decay at different rates depending on how stable they are and the rate of velocity with which they collide with an object or material in space. All particles have different charges and masses. Relative to our project we are looking at protons, muons, and gamma particles. For example less stable particles such as pions and kaons produce a cascade faster as a result of but they are not relative to our project.

A cascade occurs when particles decay producing more and more particles; a chain reaction occurs. The explosion produces masses of particles. Cascades produce particles exponentially. These cascades occur 24 hours a day 7 days a week. Cascade simulation is important because it is the main focus of our project and it provides a means of studying particle interaction cheaply.

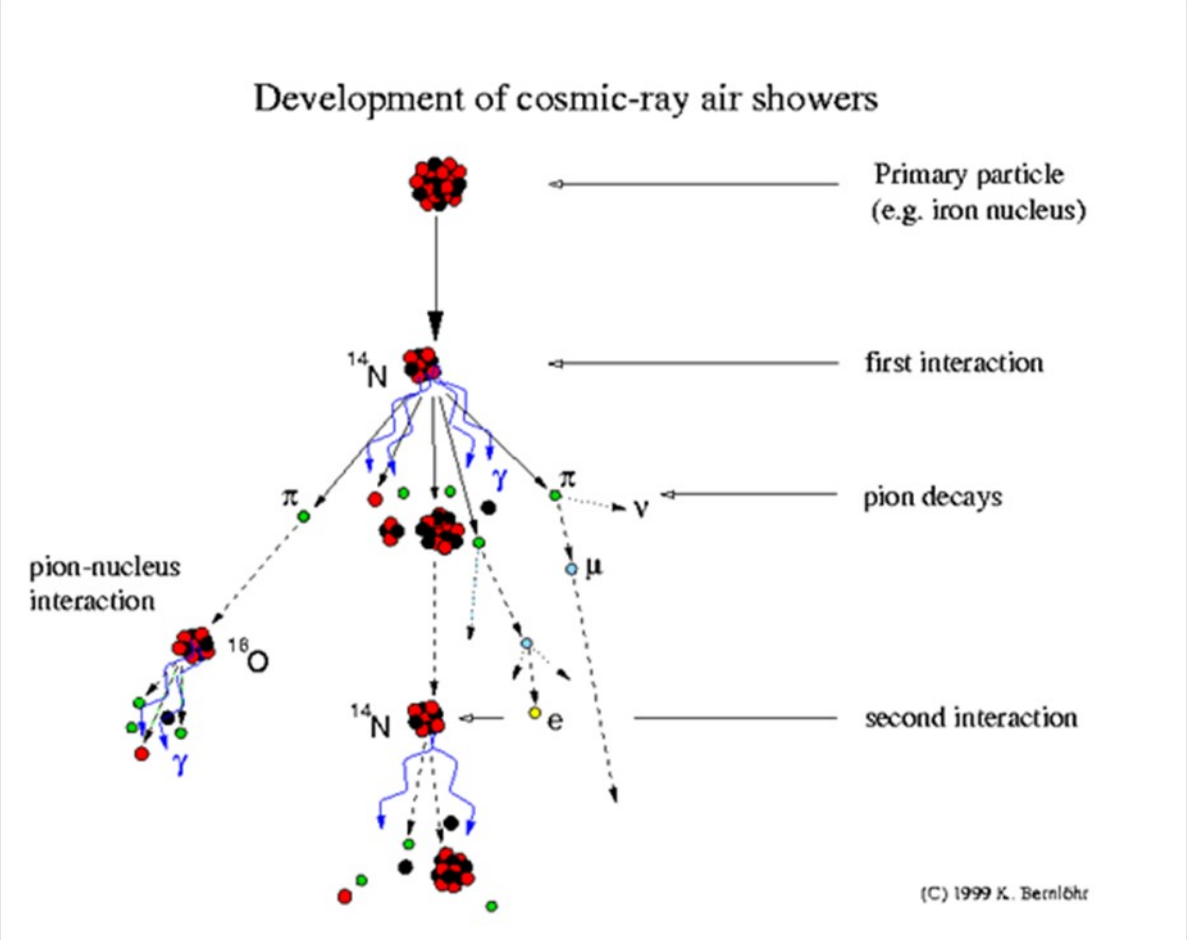
Since protons have a charge and a mass they are affected by the gravitational fields of planets, supernovas, black holes and objects in space of which either hold a strong enough gravitational pull or magnetic field to interact with the particle's t Hadronic vs rajectory. These influences alter the path of the particles when they reach the earth. Particles will decay if they encounter an object at a high enough velocity no matter how stable a particle is. This includes protons. Even though they are very stable they will still decay. We are aware that the magnetosphere and material interaction does play a role in our CORSIKA program, and have set out to identify and understand the corresponding equations. When protons hit the magnetosphere, the proton particles will be repelled, since both protons and the magnetosphere have different charges. If the proton has a high enough velocity it will break up and decay through. When they hit the earth the cascade the parent particle produces can be seen as

containing several characteristics common of the parent which produced it.

Gammas are the opposite from protons; they do not have a charge and a mass. So they are not affected by gravitational pulls of objects in space. They will also produce daughter particles but that is unlikely.

We are going to gather the attributes of both Hadronic and electromagnetic air shower interactions and with the help of our computational model determine the differences between these two groups of particle interactions.

[2]



4.2 Equations Used

The following equations are integral to calculating the outcome of interactions between particles

2 http://lpsc.in2p3.fr/TPsubat/m2/cosmic-rays.jpg

and a material and are essential to understanding the interaction between the atmosphere and a particle. These equations describe the phenomena accurately and are essential to describing several particle characteristics.

Bethe-Block Stopping Formula

$$dEi = \frac{\lambda z^2}{\beta} \kappa 1 (\ln(\Upsilon - 1) - \beta + \kappa 2) = \frac{\lambda \Upsilon^2}{\Upsilon^2 - 1} \kappa 1 (\ln(\Upsilon^2 - 1) - \beta + \kappa 2)$$

β = v/c, γ is its Lorentz factor, z is the charge of the ionizing particle in units of e. κ1 = 0.153287 MeV g −1 cm2 and κ2 = 9.386417 are derived from values for dry air

Deflection in Earth's Magnetic Field

$$\alpha \approx lZ \frac{[\vec{p} X \vec{B}]}{p^2}$$ Hadronic vs

This is a description of particle deflection in Earth's magnetic field, where l = length, z = charge, B vector = magnetic field vector, p vector = particle momentum

Time of Flight Hadronic vs

$$dt = \frac{l}{c \beta ave}$$

At the first interaction of the primary in the atmosphere, the timing of the shower is started. The time interval dt is the time elapsed as the particle moves along its path; dt is calculated by dividing the path length l by the average particle velocity, where B ave is the arithmetic mean of the particle at the beginning and end of the trajectory.

Mean Path

$$\lambda tni = m air / \sigma tni$$

Describes the interaction between muons and a material, in this case air. The mean free path for these interactions is given by the equation above where m air = 14.54 is the average atomic weight of air in g/mol and λ tni is given in g/cm^2.

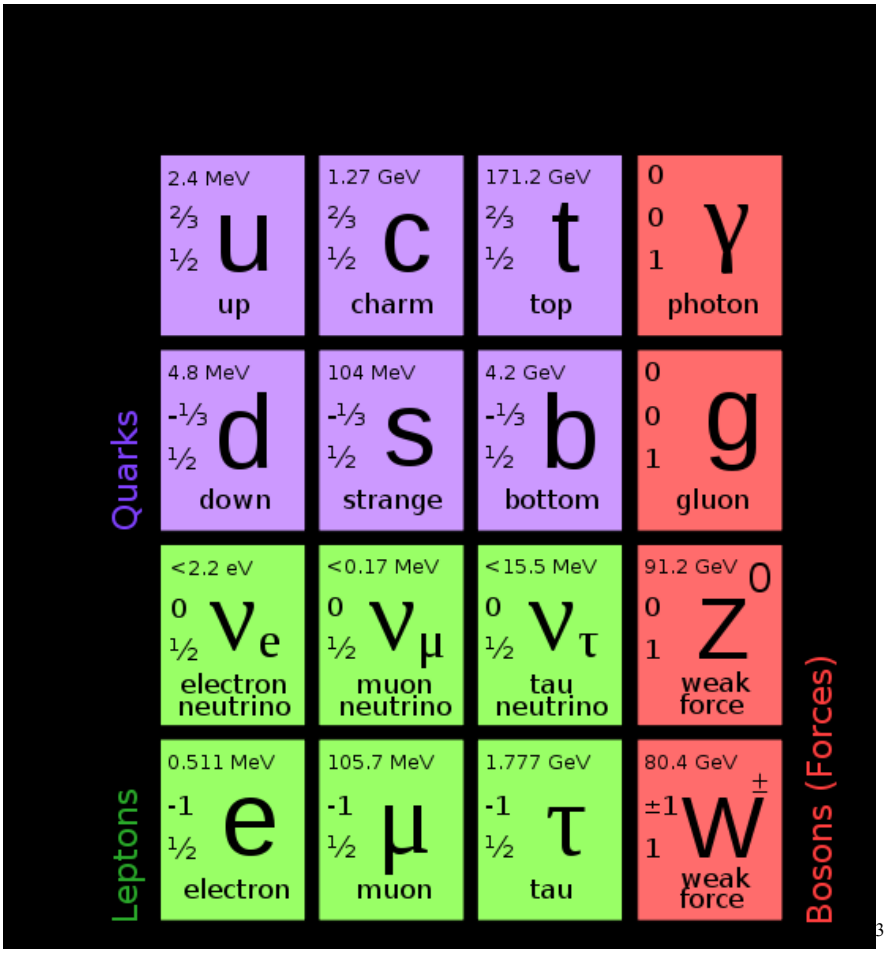
Probability of Material Traverse

$$P tni(\lambda) = \frac{1}{\lambda tni} e^{-\lambda/\lambda}$$

Describes the probability of a muon to traverse an atmospheric layer of thickness λ without corresponding interaction.

4.3 Air Shower Evaluation

We find air showers to be reliant on the particles that produce them; the particle that produces an interaction will inevitably affect what happens during this process. Therefore we find that an interaction which contains a gamma as a parent will decay differently than a proton or other particle, and as a result produce a different numbers of daughter particles as opposed to a proton shower produced under the same conditions. Likewise these particles will differ in terms of final ground level energy, and daughter particle types.



4.4 Hadronic VS. Gamma Particles

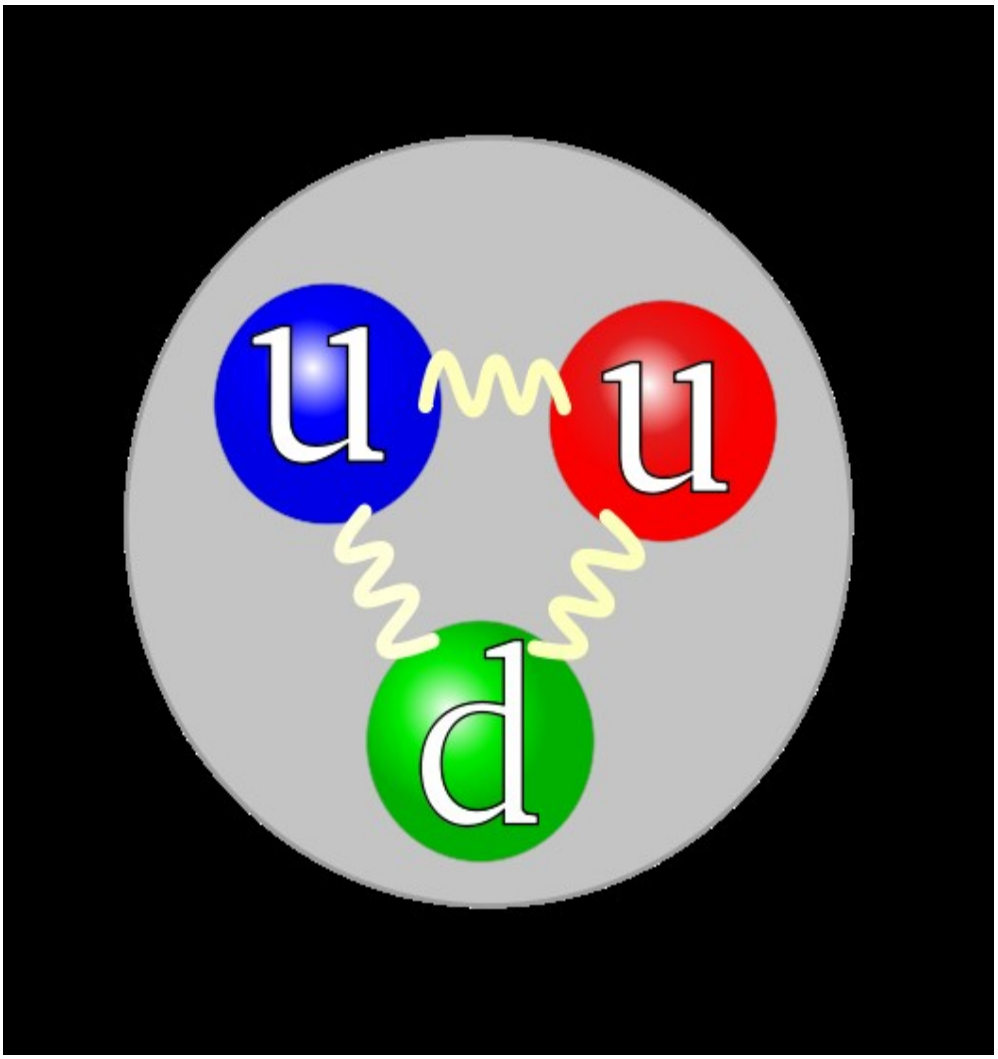
In terms of our project we are solely identifying the differences between Hadronic air shower events and electromagnetic or gamma induced events. These particles have a series of major fundamental differences in terms of the physical qualities of the individual particles themselves and in terms of the air showers they produce, as evidenced by our work. Some of the fundamental differences between protons, the parent of Hadronic showers, and gamma particles can be found in variances in charge, i.e. protons have a charge of 1+e, mass of $1.672621637(83)\times10^{-27}$ and a mean lifetime of $>2.1\times10^{29}$yr. Gamma particles however have no mass, no charge, and are stable with an indefinite lifetime (see figure above). Likewise we find that air shower events produced by differing particles

3 Standard Model of Particle Physics

decay differently as a result of their stability and makeup.

[4]Model of Proton

We find protons to be composed of one down and two up quarks as evidenced by the model below.



4 http://en.wikipedia.org/wiki/File:Quark_structure_proton.svg

5. Procedures

<u>5.1 CORSIKA Simulation</u>

The computational model we are implementing in our project is written in the FORTRAN and C programming languages and is known as CORSIKA version 6900. CORSIKA is a Monte Carlo program, meaning it sets random values for simulation, hence it looks at a wide array of simulations and interactions at different energies and altitudes. Because of such we find that CORSIKA is non-limiting and useful for a wide variety of terms. Besides imaging gamma and proton showers, which are the focus of our project, CORSIKA also simulates air showers created by other subatomic particles, nuclei of certain elements, and photons.

```
 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A

   000      000     0000       0000    00   0      0      0
  0   0    0   0    0    0    0    0   00   0    0       0 0
 0        0     0   0     0   0        00   0  0        0   0
 0        0     0   0    0     0000    00   00         0     0
 0        0     0   0000           0   00   0  0       0000000
  0   0    0   0    0   0     0    0   00   0    0     0     0
   000      000     0     0    0000    00   0      0   0     0

 COSMIC RAY SIMULATION FOR KASCADE

 A PROGRAM TO SIMULATE EXTENSIVE AIR SHOWERS IN ATMOSPHERE

 BASED ON A PROGRAM OF P.K.F. GRIEDER, UNIVERSITY BERN, SWITZERLAND
 QGSJET MODEL ACCORDING TO N.N. KALMYKOV AND S.S. OSTAPCHENKO, MSU, MOSCOW, RUSS
IA
 HDPM MODEL ACCORDING TO J.N. CAPDEVIELLE, COLLEGE DE FRANCE, PARIS, FRANCE
 GHEISHA ROUTINES ACCORDING TO H. FESEFELDT, RWTH AACHEN, GERMANY
 EGS4 ACCORDING TO W.R. NELSON, H. HIRAYAMA, D.W.O. ROGERS, SLAC, STANFORD, USA
 NKG FORMULAS FOR FAST SIMULATION OF EL.MAG. PARTICLES

 REFERENCES: D. HECK, J.KNAPP, J.N. CAPDEVIELLE, G. SCHATZ, T. THOUW,
             REPORT FZKA 6019 (1998)
             SEE ALSO WEB PAGE  http://www-ik.fzk.de/corsika/


             INSTITUT FUER KERNPHYSIK
             FORSCHUNGSZENTRUM KARLSRUHE
             POSTFACH 3640
             D-76021 KARLSRUHE
             GERMANY

 IN CASE OF PROBLEMS CONTACT:   Dr. Tanguy Pierog
                     e-mail:   tanguy.pierog@ik.fzk.de
                        FAX:   (49) 7247-82-4075
                      PHONE:   (49) 7247-82-8134
   OR                      :   Dr. Dieter Heck
                     e-mail:   dieter.heck@ik.fzk.de
                        FAX:   (49) 7247-82-4075
                      PHONE:   (49) 7247-82-3777
   AND SEND YOUR LIST-FILE BY E-MAIL

 NUMBER OF VERSION :  6.900
```

[5]

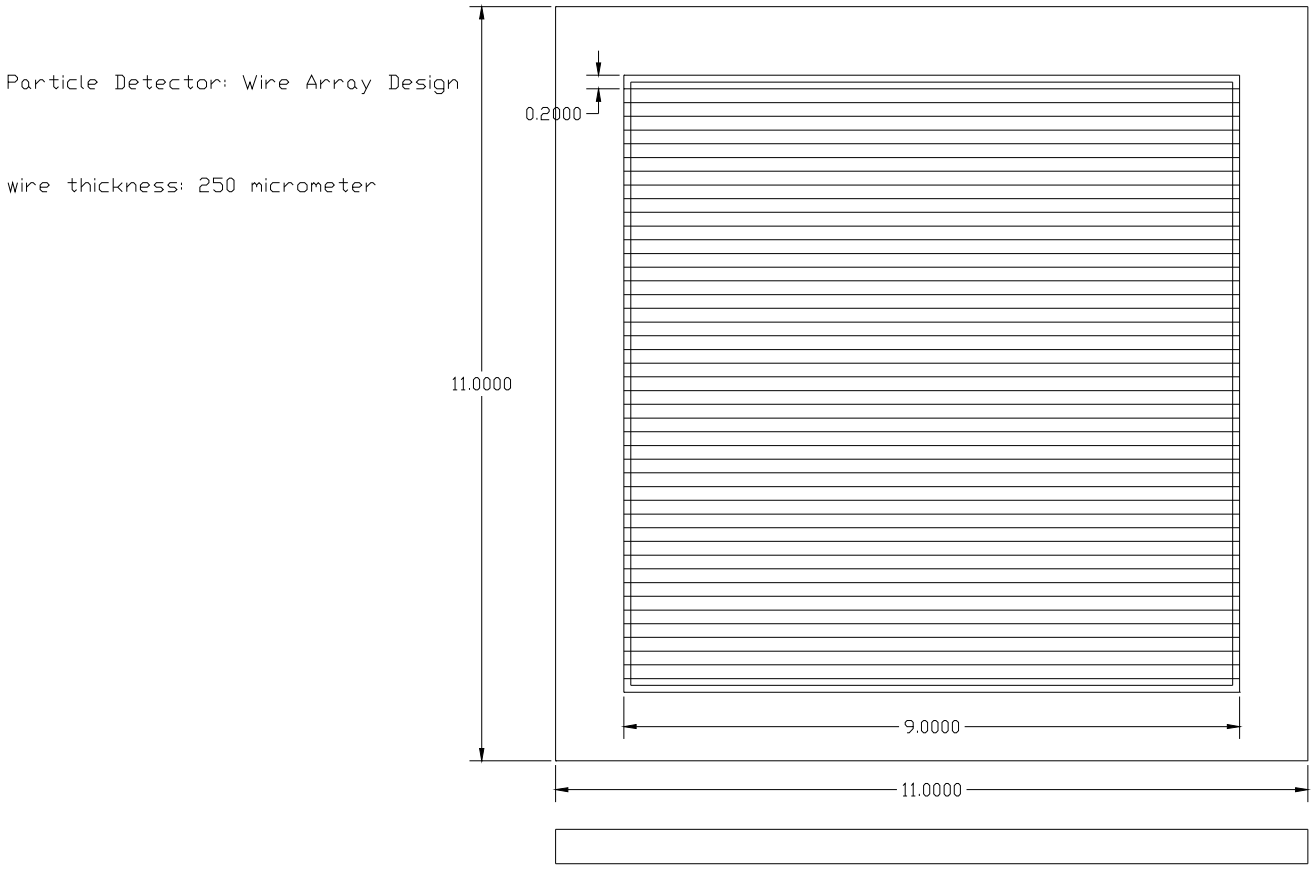
<u>5.2 Physical Model</u>

Our physical model is based on a wire array detector known as a Geiger Muller detector and consists of several arrays of thin wire, approximately 250 micrometers in diameter, which is laid out

5 CORSIKA start page

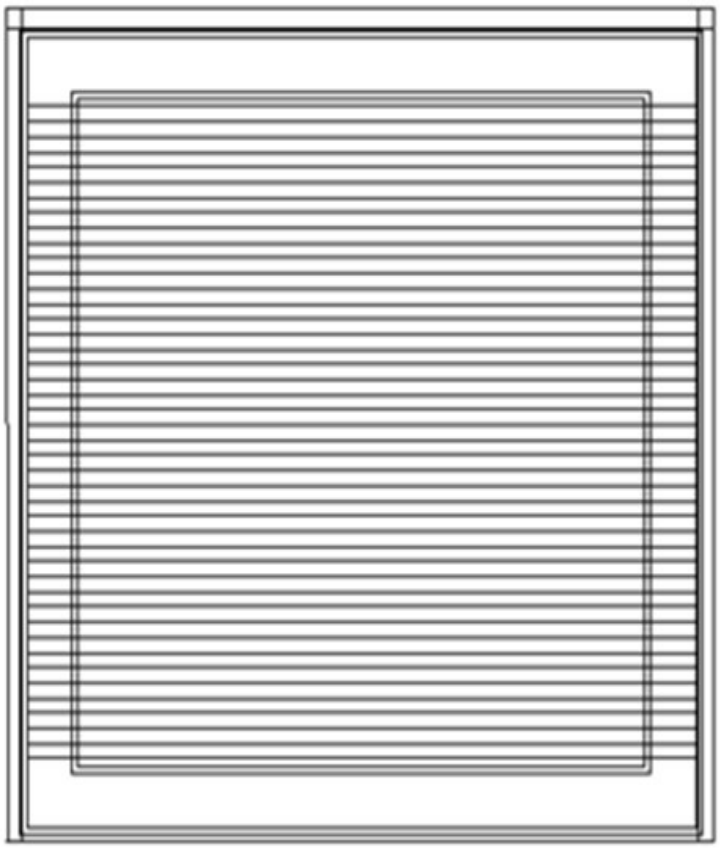
and charged with high voltage. These detectors are assembled in an array and are designed to have several layers of wire arrays. The physical properties of these detectors allow particles with a charge to be counted because when these particles encounter the array they interact with the voltage carried through the sense wires and change the overall voltage running through the system, which initiates our counter to consider this small change as a particle interaction. We must allow for arrays to be layered in order to effectively determine a particle's origin. The reason for such lies in the fact that background radiation interacts with the detector and can cause changes in voltage within one array although if data from layered arrays is considered we can consider background radiation ruled out. Such phenomenon exists because we find that background radiation has a low implicit velocity and charge, therefore it should cause a change in only one array although we find that particles resulting from cosmic rays and their daughter particles hold an inherit high charge and velocity and should likewise pass through multiple detectors easily. With the combination of a semi large array of particles we should be able to determine the pattern produced by several real air showers and compare them to data found through our simulations in CORSIKA and our original coding to determine the parent particle which initiated the shower.

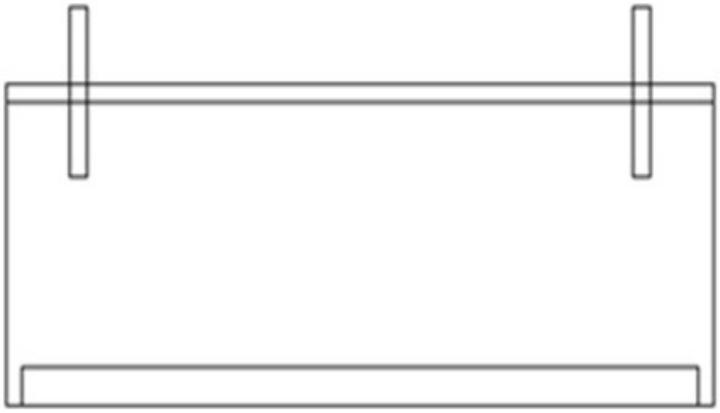
At this point we have not been able to run our detector due to problems associated with the particle counter and power supply. This side project is still in progress and we intend to complete and run it in order to possibly compare data collected from this with our own coding and the CORSIKA models.



6 Original particle detector dsesign

Array Housing-Particle Detector Array





6. Analysis of Methods

One of the methods we are going to use involves the evaluation the series of plots and graphs that were produced by CORSIKA and our own coding. We are going to use these graphs as a means of determining the differences between Hadronic and Electromagnetic air showers. Through our research we have discovered that the methods we have employed, evaluating daughter particle distances from the center, and evaluating daughter particle types produced are effective ways of determining the differences of Hadronic and Electromagnetic air showers.

In one of our graphs we are comparing the distance of daughter particles from the center of the interaction from both proton and gamma showers. We intend to determine similar qualities between similar particles in order to produce a means of particle identification.

In another one of our graphs we are comparing the daughter particle spread of both Hadronic and Electromagnetic showers. This method helped us determine the different types of daughter particles that will be produced between both Hadronic and Electromagnetic showers in one interaction.

In our plots we are comparing the images of the Hadronic and Electromagnetic showers. Some visual differences we need to analysis in the plots are high amounts of energy or low amounts of energy, if the energy is more compressed or more spread out, and weather or not the interaction produces daughter particles or not.

If the air shower is produce by particles that have a mass and a charge then visually the plot will reveal high amounts of energy, the energy will be more compressed, and there will daughter particles like muons present. If the air shower is produced by particles without a mass and a charge then there will be lower amounts of energy, the energy will be more spread out, and there will be no daughter particles.

7. Sample Results

7.1 Sample Data

The following data has been produced as a result of our own coding and the COSIKA software and represents a fragment of the data used for our study.

7.2 Data from Distcalc.cpp

The following data excerpt represents particle distance versus bin number produced by the Distcalc.cpp program. This data is visualized in the 'Particle Distance from Center' plots following.

0 2670

1 2527

2 2054

3 1710

4 1363

5 1060

6 898

7 740

8 550

9 498

10 437

11 377

12 294

13 276

14 244

15 207

16 188

17 165

18 153

19 129

20 132

21 108

22 86

23 96

24 73

25 74

26 73

27 66

28 71

29 43

30 50

31 55

32 50

33 38

34 46

35 41

36 25

37 31

38 37

39 22

7.3 Data from PID-Distributions.cpp

The following data represents the number of daughter particles versus daughter particle types produced by our PID-Distributions.cpp program, and is visualized directly below in the 'Daughter Particle Spread' plots.

Gamma Parent

0 637

1 430627

2 18410

3 33565

5 55

6 61

Proton Parent

0 366

1 755891

2 37362

3 64554

5 7493

6 7302

7.4 Daughter Particle Spread

The following graph represents daughter particle types produced by a pool of gamma and proton interactions, focusing on particle types 1-6. We find that for the given data set that there is a subtle

although noticeable difference between the daughter particles produced by both gamma and proton parents between the number of type five and six particles. This difference therefore provides a possible means of particle identification and differentiation.

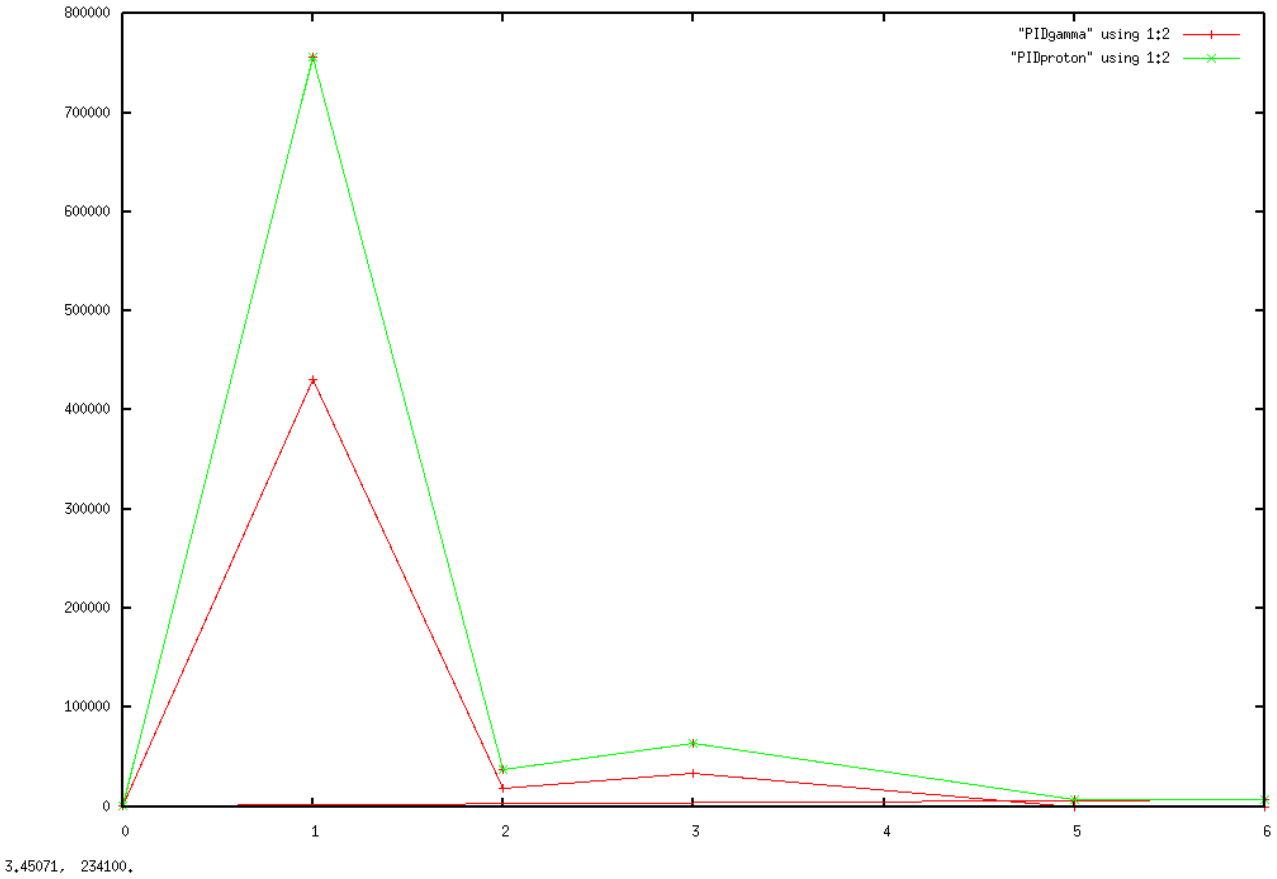
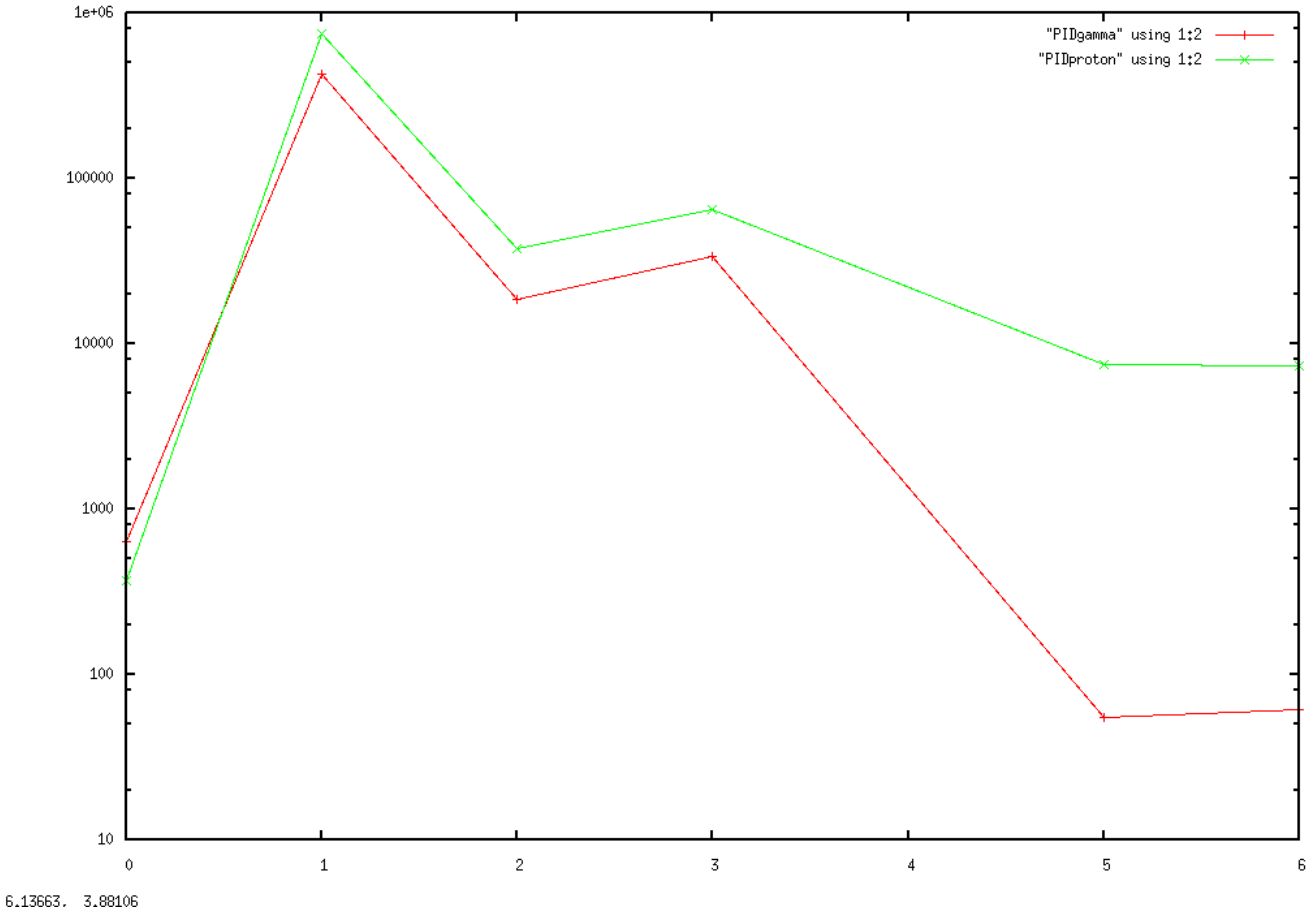


Image plotted with Log Scale

In the following plot the same data set used above is plotted with a log scale function in order to make more subtle differences stand out. As a result we find that daughter particle types produced by both gamma and proton parents is similar, although through use of the log scale function the differences between particle types five and six, the muon particle and antiparticle, μ±, is made more clear.



7.5 Particle Distance from Center

In the following images we find a representation of the individual particle distances from the

center plotted as a function of number of particles versus particle distance. On the image portrayed directly below we find that the distances of particles, i.e muons, produced by different parent particles is very close regardless of the fact that they are produced by different parents. As a result we can label particle distance from the center as a fairly inaccurate means of distinguishing the parent of an interaction, but a valuable means of differentiating between daughter particle types produced.

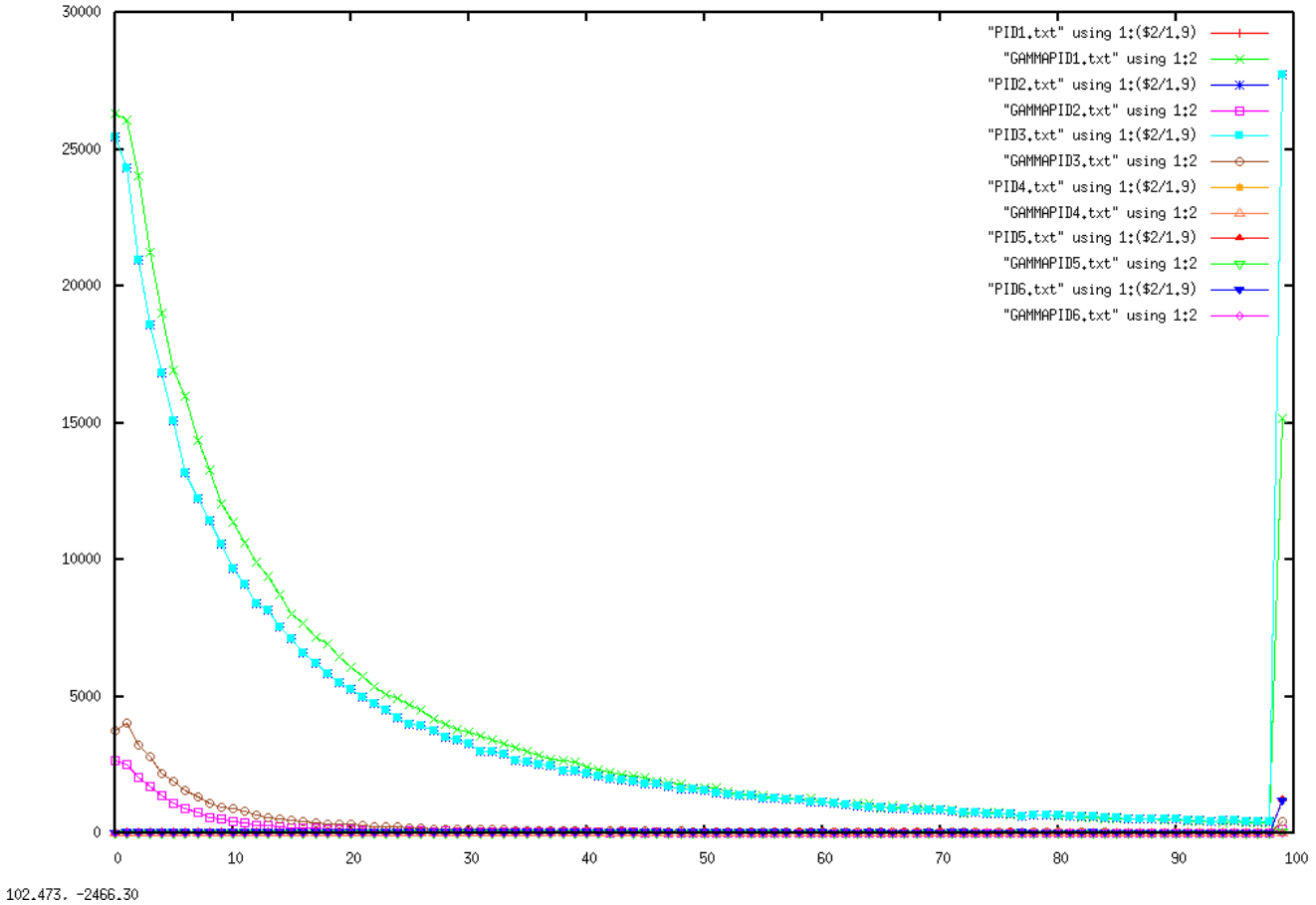
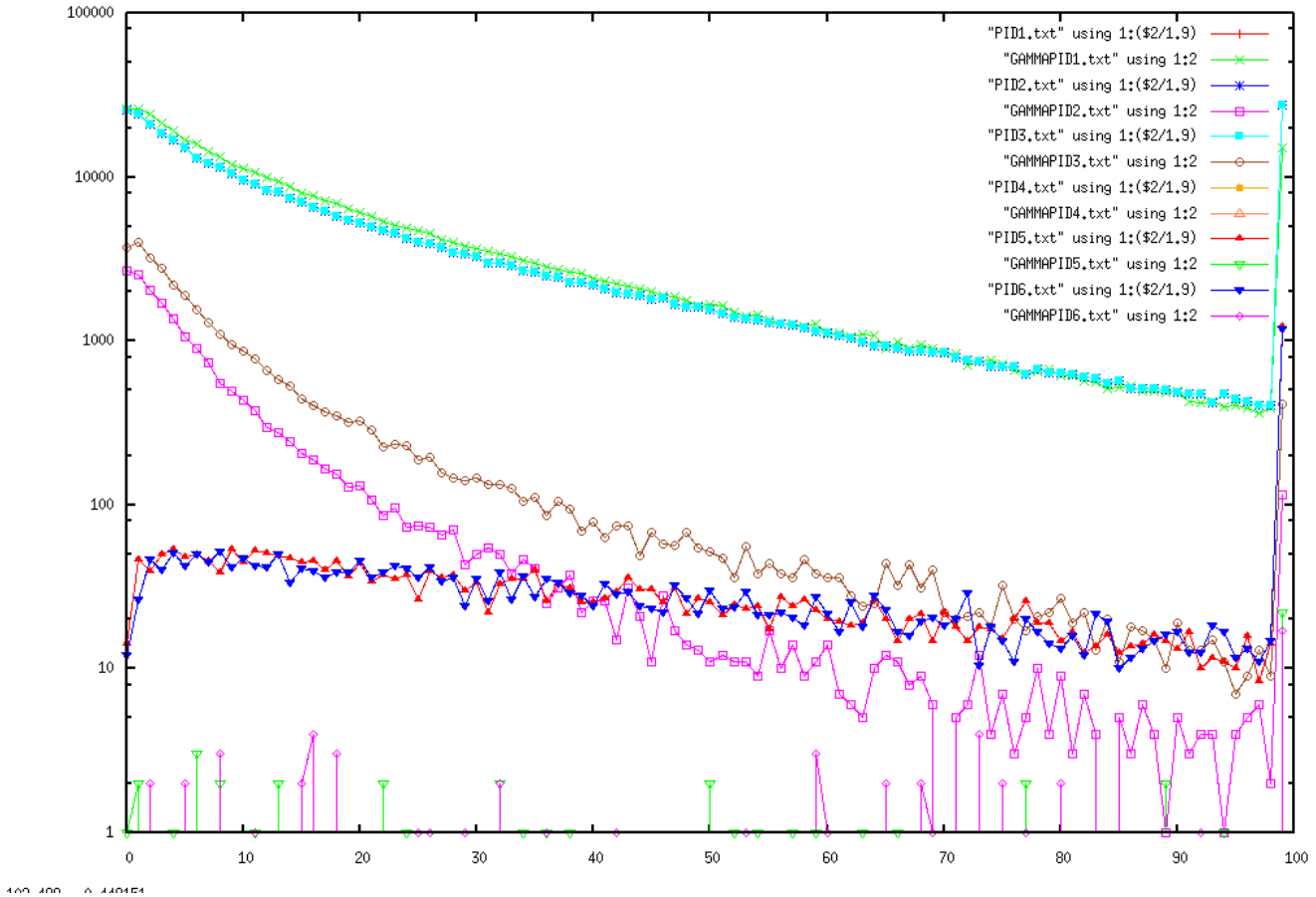


Image Plotted with Log scale Function

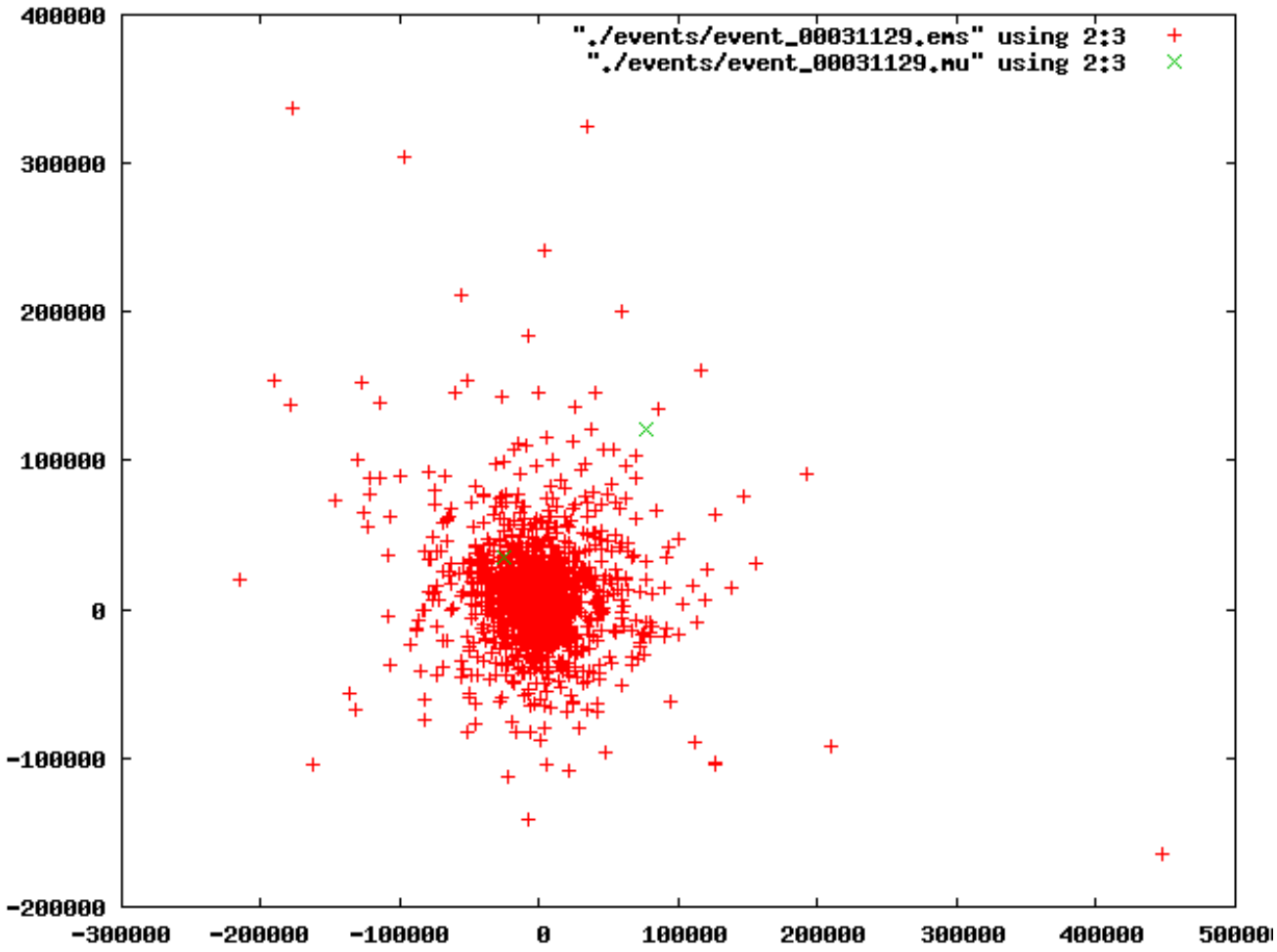
The plot below represents the same data table as above although with an added log scale function used to make subtle differences between our plotted data easier to interpret. Likewise the

linear regression of each particle type versus distance from center is made more clear.



7.6 Proton Shower Images

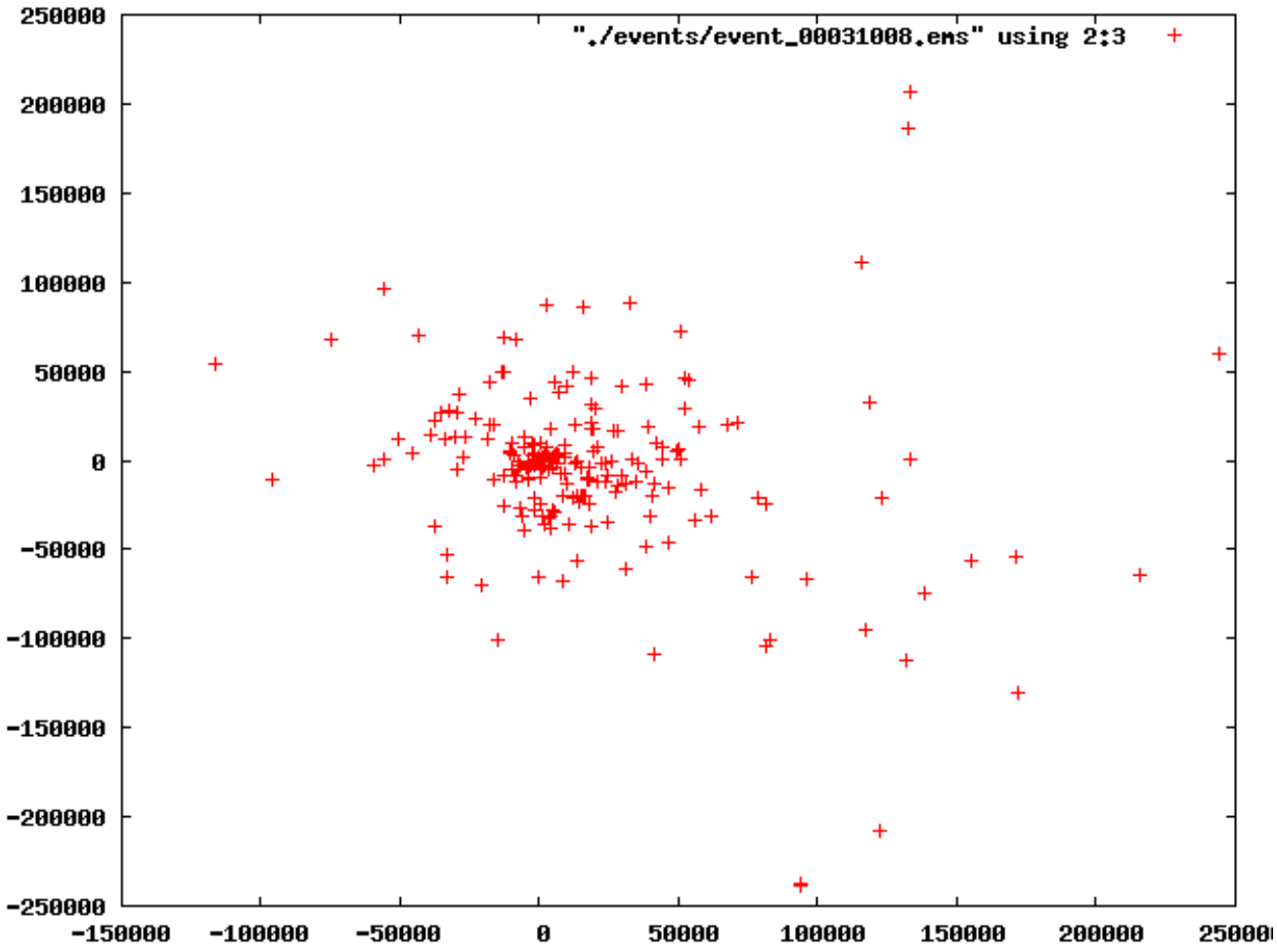
Following are a series of proton plots produced in CORSIKA and plotted in GNUplot, each represents an individual shower with differing energies. Through evaluating these plots we can begin to see differences between different interactions, i.e. muon production in proton showers[7], in the case of the following images.[8]

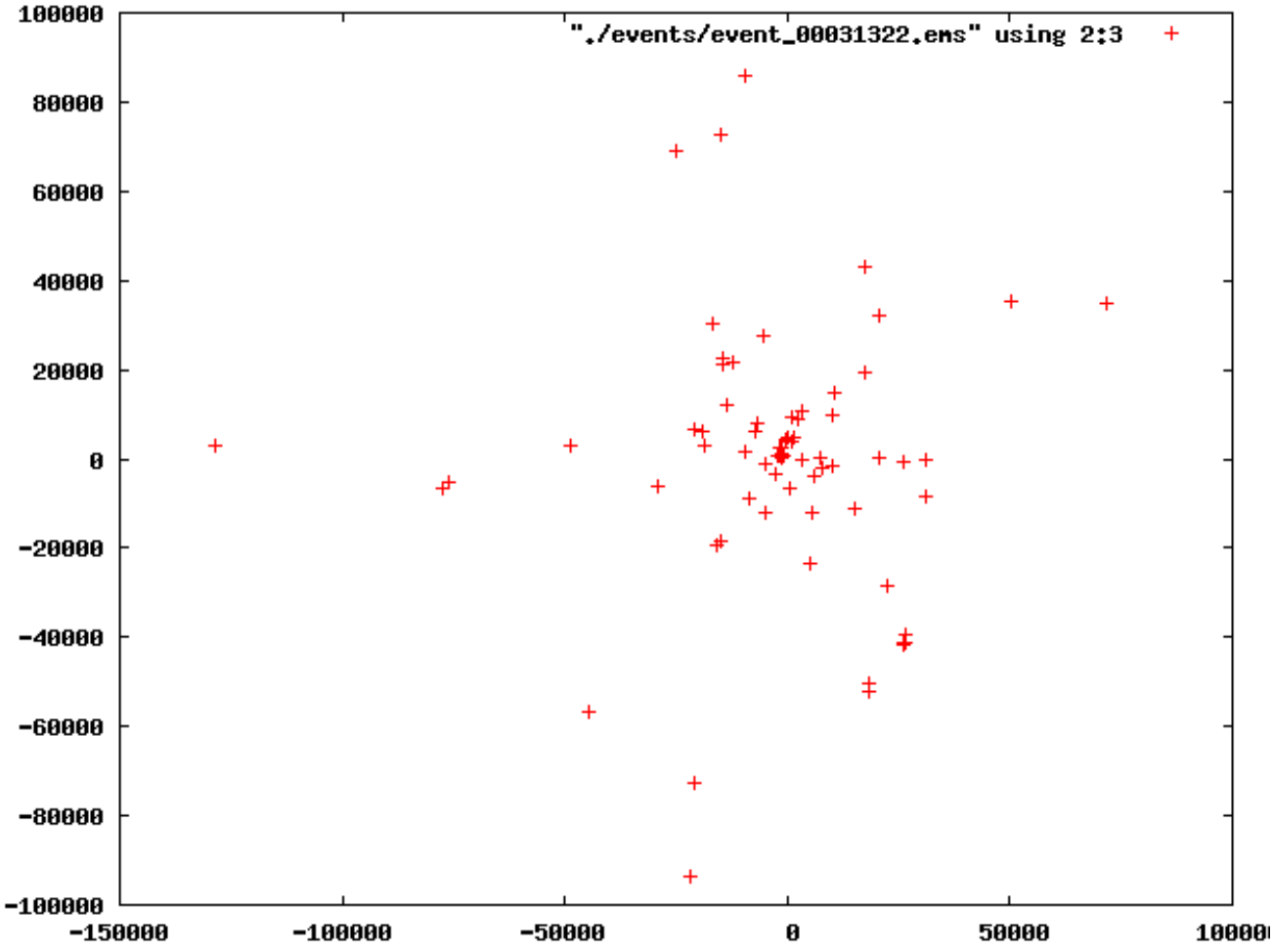


7 Muon particles represented in green in plots.
8 All GNUplot images original product of data produced by CORSIKA v. 6900

7.7 Gamma Shower Images

The following images contain data relative to gamma particle interactions; once again we find some visual differences between gamma and proton induced interactions. Note in the following images the lack of muons and overall difference in number of daughter particles.




"./events/event_00031322.ems" using 2:3
100000
80000
60000
40000
20000
0
-20000
-40000
-60000
-80000
-100000
-150000
-100000
-50000
0
50000

8. *Future Work*

This project is still in progress. In the future we would like to:

- Input data and actually determine weather the interaction is a proton or a gamma. Right now we only have characteristics; we do no actually have solid conclusions that the interaction we are looking at is a proton or a gamma

- Optimize our coding to produce images in our GNUplots that will be easier and more efficient to analyze

- If possible we would like to feed our coding with real world data and compare both the real world data and data from the CORSIKA source code to determine if results are the same in both situations.

9. Conclusion

With our research we have discovered that our coding and the CORSIKA source code will help us develop plots and graphs that will distinguish between characteristics of both Hadronic and Electromagnetic air showers. Some characteristics we have discovered in our plots and graphs are energy amounts, the spread and compression of the energy amount, distance away form the center, and daughter particles. With the characteristics that we have gathered from our research we will be able to determine the differences between Hadronic and Electromagnetic air showers. We have found that the most significant original achievement made as a result of our project is relative to determining specific characteristics of different air shower events.

10. Acknowledgments

In consideration of support received from external sources and people we would like to accredit the success of this project to:

- John Pretz
- Brenda Dingus
- Michelle Thomsen
- Philip Sanchez

11. Bibliography

10.1 Internet Resources

Carlson, Shawn, “Counting Particles from space”. Scientific American February 2001: 1-2

Wales, Jimmy. "Statistics". Wikimedia Foundation, Inc.. April 2010 <http://en.wikipedia.org/wiki/Statistics#Statistical_methods>.

Pierog, Tanguey. "CORSIKA an Air Shower Simulation Program". Karlsruhe Institute of Technology . April 2010 <http://www-ik.fzk.de/CORSIKA/>.

Wales, Jimmy. "Bethe Formula". Wikimedia Foundation, Inc.. April 2010 <http://en.wikipedia.org/wiki/Bethe_formula>.

Wales, Jimmy. "Lorentz Factor". Wikimedia Foundation, Inc.. April 2010 <http://en.wikipedia.org/wiki/Lorentz_factor>.

Wales, Jimmy. "Muon". Wikimedia Foundation, Inc.. April 2010 <http://en.wikipedia.org/wiki/Muons>.

Wales, Jimmy. "Particle Physics". Wikimedia Foundation, Inc.. April 2010 <http://en.wikipedia.org/wiki/Particle_physics#Subatomic_partilces>.

Wales, Jimmy. "List of Particles". Wikimedia Foundation, Inc.. April 2010 <http://en.wikipedia.org/wiki/List_of_subatomic_particles>.

12. Appendix

In the following sections we provide a guide to the code, its functions and some notes relative to material incorporated into our coding.

12.1 Particle Identification Table

Particle identifications			
Identification	Particle	Identification	Particle
1	γ	51	ρ°
2	e^+	52	ρ^+
3	e^-	53	ρ^-
		54	Δ^{++}
5	μ^+	55	Δ^+
6	μ^-	56	Δ°
7	π°	57	Δ^-
8	π^+	58	$\overline{\Delta}^{--}$
9	π^-	59	$\overline{\Delta}^{-}$
10	K_L°	60	$\overline{\Delta}^\circ$
11	K^+	61	$\overline{\Delta}^+$
12	K^-	62	$K^{*\circ}$
13	n	63	K^{*+}
14	p	64	K^{*-}
15	$\overline{p}$	65	$\overline{K^*}^\circ$
16	K_S°	66	ν_e
17	η	67	$\overline{\nu}_e$
18	Λ	68	ν_μ
19	Σ^+	69	$\overline{\nu}_\mu$
20	Σ°		
21	Σ^-	71	$\eta \rightarrow \gamma\gamma$
22	Ξ°	72	$\eta \rightarrow 3\pi^\circ$
23	Ξ^-	73	$\eta \rightarrow \pi^+\pi^-\pi^\circ$
24	Ω^-	74	$\eta \rightarrow \pi^+\pi^-\gamma$
25	$\overline{n}$	75	μ^+ add. info.
26	$\overline{\Lambda}$	76	μ^- add. info.
27	$\overline{\Sigma}^-$		
28	$\overline{\Sigma}^\circ$		
29	$\overline{\Sigma}^+$		
30	$\overline{\Xi}^\circ$		
31	$\overline{\Xi}^+$		
32	$\overline{\Omega}^+$		
50	ω		

Table 4: Particle identifications as used in CORSIKA (to be continued).

85

Particle identifications (continued)			
Identification	Particle	Identification	Particle
116	D°	149	$\overline{\Lambda_c}$
117	D^{+}	150	$\overline{\Xi_c}$
118	$\overline{D}$	151	$\overline{\Xi_c}^{\circ}$
119	$\overline{D}^{\circ}$	152	$\overline{\Sigma_c}$
120	D_s^{+}	153	$\overline{\Sigma_c}$
121	$\overline{D_s}$	154	$\overline{\Sigma_c}^{\circ}$
122	η_c	155	$\overline{\Xi_c'}$
123	$D^{*\circ}$	156	$\overline{\Xi_c'}^{\circ}$
124	D^{*+}	157	$\overline{\Omega_c}^{\circ}$
125	$\overline{D^*}$		
126	$\overline{D^*}^{\circ}$	161	Σ_c^{*++}
127	D_s^{*+}	162	Σ_c^{*+}
128	$\overline{D_s^*}$	163	$\Sigma_c^{*\circ}$
130	J/ψ	171	$\overline{\Sigma_c^*}$
131	τ^{+}	172	$\overline{\Sigma_c^*}$
132	τ^{-}	173	$\overline{\Sigma_c^*}^{\circ}$
133	ν_{τ}		
134	$\overline{\nu}_{\tau}$		
137	Λ_c^{+}		
138	Ξ_c^{+}		
139	Ξ_c°		
140	Σ_c^{++}		
141	Σ_c^{+}		
142	Σ_c°		
143	$\Xi_c'^{+}$		
144	$\Xi_c'^{\circ}$		
145	Ω_c°		
$A \times 100 + Z$	nucleus of Z protons and $A - Z$ neutrons ($2 \leq A \leq 59$)		
9900	Cherenkov photons on particle output file		

Table 4: (continued) Particle identifications as used in CORSIKA.

12.2 Data Read Code[9]

The data read code basically reads data produced by CORSIKA from a file and cleans it up. We are taking raw heterogeneous numbers and making sense of them so that they can be used for evaluation.

Make file

```
all:

    g++ -I . -c read.cpp

    g++ -I . -c milCORSIKAInterface.cpp

    g++ read.o milCORSIKAInterface.o -o read -lz




clean:

    rm -f *.o

    rm -f read
```

MilCORSIKA.cpp

```
// variables to print out: particle id space, x, y, new line


#include <iostream>

#include <cmath>

#include <cstdlib>

#include <fstream>

using namespace std;
```

9 Data read code written by John Pretz, alterations made by Dennis Trujillo and Oliver Galvan

```
int main()

{
const char* filename = "infile"; //name of file to read
ifstream infile (filename);                    //create input stream object




if (!infile)

{
cout << endl << "Failed to open file" << filename;
return 1;
}



double pid = 0;
double posx = 0;
double posy = 0;

while (!infile.eof())
{
```

//need to import data, coordinates in Cartesian coordinate system of particles produced from each event

PID-Distributions.cpp

infile >> pid;

infile >> posx;

infile >> posy;

cout<< pid<<""<<posx<<""<<posy<< endl;

}

//Run a for loop to determine distance of each particle produced to center

for (double posx = 0; posx <= 0; posx ++)

{

//using 3D distance equation, applied to each coordinate relative to center

//d= SQRT [(delta x)^2 + (delta y)^2 + (delta z)^2]

double DIcent = 0;

//DIcent = sqrt (pow (posx), 2) + sqrt (pow (posy), 2);

cout<< DIcent <<endl;

}

//need to write 'for' statement with 'if' statement within to determine particle types and count

//need to count total number of daughter particles produced

//print to file

}

MilCORSIKA.hh

#ifndef __milCORSIKAInterface_hh__

#define __milCORSIKAInterface_hh__

#include <fstream>

#include <vector>

#include <stdio.h>

#include <string>

#include <iostream>

#include <zlib.h>

#include <list>

#include <algorithm>

#include <fstream>

using namespace std;

```
///Holds details about one particle from the CORSIKA shower
typedef struct {
   int      id;  ///<  CORSIKA particle ID
   double px;  ///< px in GeV
   double py;  ///< py in GeV
   double pz;  ///< pz in GeV
   double t0;  ///< time in ns
   double x0;  ///<X position in cm
   double y0;  ///<Y position in cm




//         ofstream outfile("~Dennis/Desktop/outfile");
//{
//if (1 % 5 == 0)

//outfile << end1;
//outfile << setw(10) << *(px + i);
//}



} CParticle;
```

```
///Gets information from CORSIKA file and holds some other nice variables
class milCORSIKAInterface
{

private:
  milCORSIKAInterface(int nsub=21, int lsub=273);
  static milCORSIKAInterface * theCORSIKAFile;
  ~milCORSIKAInterface();

public:
  static inline milCORSIKAInterface & GetCORSIKAFile();
  static inline milCORSIKAInterface * GetCORSIKAFilePtr();

    void OpenInputFile(const string filename);
    int NextEvent();

    void PrintRunHeader();
    void PrintRunEnd();

    double *GetRunHeader()   { return fRunHeader;}
    double *GetRunEnd()      { return fRunEnd;}
    double *GetEventHeader() { return fEventHeader;}
    double *GetEventEnd()    { return fEventEnd;}
```

inline void SetOutputFile(string f) {filename=f;} ///<Stores the output file location

inline string GetOutputFile() {return filename;} ///<Returns the output file location

typedef std::vector<CParticle> CParticleList;

typedef CParticleList::const_iterator CIterator;

CParticleList GetCPlist() {return CPlist;} ///<Gets the information for all shower particles reaching the ground

// Here is where we changed the code, currently trying to print list CPlist, example below didn't work.

//{

//int particles;

//cout << particles;

//}

private:

CParticleList CPlist; ///<Hold information of all shower particles from CORSIKA file

int RecordType(float f); ///< Check the data type acoording to the first word of the sub-block.

// Return value:

// 1: run header

// 2: event header

```
    //   3: end of event
    //   4: end of run
   int ReadBuff();       ///< Read one record from disk file
   int GetSubblock();   ///< get one sub block from the data buff

   void FillRunHeader(float *buff);
   void FillRunEnd(float *buff);
   void FillEventHeader(float *buff);
   void FillEventEnd(float *buff);

   void FillEvent(float *buff); ///< stores the particle information from one sub-block


//   std::ifstream inputFile;      ///< input stream
   gzFile inputFile;

   int   NumberOfSubblock;  ///< number of sub-blocks in a record (21)
   int   LengthOfSubblock;  ///< length of a sub-block (27words)
   int   RecordLength;      ///< length of the record in words
   int   BuffLength;        ///< data buffer length in bytes
   int   BuffPos;           ///< current position in data buff
   float* databuff;         ///< data buff to store one record
   float *buff;             ///< a pointer to a sub-block in data buff
   int   BuffRead;          ///< flag if a record in read
```

```
    double fRunHeader[273];   ///< run header information
    double fRunEnd[273];      ///< end of run information
    double fEventHeader[273]; ///< event header block
    double fEventEnd[273];    ///< event end block
    CParticle cp;
    string filename;

};


milCORSIKAInterface *milCORSIKAInterface::GetCORSIKAFilePtr()
{
  if (theCORSIKAFile == NULL)
    theCORSIKAFile= new milCORSIKAInterface();
  return theCORSIKAFile;
}

milCORSIKAInterface &milCORSIKAInterface::GetCORSIKAFile()
{
  if (theCORSIKAFile == NULL)
    theCORSIKAFile= new milCORSIKAInterface();
  return *theCORSIKAFile;
```

```
}


#endif
```

Read.cpp

```
#include <milCORSIKAInterface.hh>

#include <iostream>
#include <cmath>
#include <cstdlib>

using namespace std;

int main(int argc,char** argv)
{
  if(argc != 2)
    {
      cout<<"Gotta gimme a file"<<endl;
      exit(1);
    }

  string infile = argv[1];
```

```
milCORSIKAInterface& file = milCORSIKAInterface::GetCORSIKAFile();
file.PrintRunHeader();
file.OpenInputFile(infile);
unsigned eventNumber = 0;
while(!file.NextEvent())
  {
    eventNumber++;
    milCORSIKAInterface::CParticleList particles = file.GetCPlist();
    double* header = file.GetEventHeader();

    int npart = 0;
    for(unsigned i = 0 ; i < particles.size() ; i++){
      int level = particles[i].id%10;
      if(level == 1)
        npart++;
    }

    if(npart > 10) //particles.size() > 0)
      {
        printf("%d particle type\n",(int)header[2]);
        printf("%0.2f total energy in GeV\n",header[3]);
        printf("%zd particles hitting the ground\n",particles.size());
```

```
int n_1 = 0;
int n_2 = 0;
int n_3 = 0;

char muOutfileName[1024];
sprintf(muOutfileName,"events/event_%08d.mu",eventNumber);

char emsOutfileName[1024];
sprintf(emsOutfileName,"events/event_%08d.ems",eventNumber);

FILE* muOutfile = fopen(muOutfileName,"w");

FILE* emsOutfile = fopen(emsOutfileName,"w");

for(unsigned i = 0 ; i < particles.size() ; i++)
  {
    int pid = particles[i].id/1000;
    int level = particles[i].id%10;
    double energy = sqrt(pow(particles[i].px,2) +
                         pow(particles[i].py,2) +
                         pow(particles[i].pz,2));
    //            printf("pid:%d level:%d energy:%f\n",pid,level,energy);
```

```
    if(level == 1)
     {
      n_1++;
      if(pid == 5 or pid == 6)
       fprintf(muOutfile,"%d %f %f\n",pid,particles[i].x0,particles[i].y0);
      if(pid == 1 or pid == 2 or pid == 3)
       fprintf(emsOutfile,"%d %f %f\n",pid,particles[i].x0,particles[i].y0);
     }
    if(level == 2)
     {
      n_2++;



     }
    if(level == 3)
     {
      n_3++;
     }

  }
 fclose(muOutfile);
 fclose(emsOutfile);
 printf("n_1:%d n_2:%d n_3:%d\n",n_1,n_2,n_3);
 printf("-----------------------------------\n");
```

```
        }


    }
}
```

12.3 Particle Determination Code

The particle determination code uses a series of parameters to read data and calculate the distance of daughter particles from the center of the interaction in terms of a Cartesian coordinate system. This data is then placed in a series of bins and graphed with a log scale to provide for better difference evaluation between particle interactions.

Distcalc.cpp

```
//DISTCALC: calculate distance from center


// variables to print out: particle id space, x, y, new line


#include <iostream>
#include <cmath>
#include <cstdlib>
#include <fstream>
#include <map>
using namespace std;
```

```
int main(int argc,char** argv)

{



int histogram[100];

for(unsigned i = 0 ; i < 100 ; i++){
  histogram[i] = 0;
}



for(int argnumber = 1 ; argnumber < argc ; argnumber++){

//if(argc != 2){
          //cout<<"akkkkk.  gimme one and only one input file"<<endl;
//       return 1;
//}

const char* filename = argv[argnumber]; //name of file to read
ifstream infile (filename);                 //create input stream object
```

```
if (!infile)

{
cout << endl << "Failed to open file" << filename;
return 1;
}


double pid = 0;
double posx = 0;
double posy = 0;


while (!infile.eof())
{

//need to import data, coordinates in cartesian coordinate system of particles produced from each event


infile >> pid;
infile >> posx;
```

```
infile >> posy;

double DIcent = 0;
if(pid == 4){
DIcent = sqrt (pow (posx, 2) + pow (posy, 2));
//cout<<DIcent<<endl;

int histogrambin = DIcent / 1000;
if(histogrambin < 0)
  histogrambin = 0;
if(histogrambin > 99)
  histogrambin = 99;

histogram[histogrambin] = histogram[histogrambin] + 1;
}
//if(DIcent > 0 && DIcent < 1000){
//  histogram[0] = histogram[1] + 1;
//}

//particlecounts[pid] = particlecounts[pid] + 1;

//cout<< pid<<" "<<posx<<" "<<posy<< endl;
}
```

```
//Run a for loop to determine distance of each particle produced to center

//for (double i = 0; i <= 0; i ++)
//{
//using 3D distance equation, applied to each coordinate relative to center
//d= SQRT [(delta x)^2 + (delta y)^2 + (delta z)^2]




//cout<< DIcent <<endl;
//}

//need to write 'for' statement with 'if' statement within to determine particle types and count

//for (double i = 0; i <=0; lnum ++)
//{

//if pid = 2
//{
//ofstream file;
//int i = 0;
//float x = 100;
```

```
//file<< "PID\t

//need to count total number of daughter particles produced

//print to file



}

for(unsigned i = 0 ; i < 100 ; i++){
  cout<<i<<" "<<histogram[i]<<endl;
}


}
```

PID-Distributions.cpp

The PID-Distribution code reads data produced by CORSIKA and sorted by the read and milCORSIKA files in order to separate the daughter particles produced by each type of interaction into singular bins. This data is then graphed in GNUplot with an added log scale to help identify subtle differences between interactions.

```
//PID-Distributions: calculate spread of PID type distribution

// variables to print out: particle id space, x, y, new line
```

```
#include <iostream>
#include <cmath>
#include <cstdlib>
#include <fstream>
#include <map>
using namespace std;

int main(int argc,char** argv)

{
map<int,int> particlecounts;


for(int argnumber = 1 ; argnumber < argc ; argnumber++){

//if(argc != 2){
          //cout<<"akkkkk.  gimme one and only one input file"<<endl;
//        return 1;
//}

const char* filename = argv[argnumber]; //name of file to read
ifstream infile (filename);                 //create input stream object
```

```
if (!infile)

{
cout << endl << "Failed to open file" << filename;
return 1;
}


double pid = 0;
double posx = 0;
double posy = 0;


while (!infile.eof())
{

//need to import data, coordinates in cartesian coordinate system of particles produced from each event


infile >> pid;
```

```
infile >> posx;

infile >> posy;


particlecounts[pid] = particlecounts[pid] + 1;


//cout<< pid<<" "<<posx<<" "<<posy<< endl;

}


//Run a for loop to determine distance of each particle produced to center


//for (double i = 0; i <= 0; i ++)

//{

//using 3D distance equation, applied to each coordinate relative to center

//d= SQRT [(delta x)^2 + (delta y)^2 + (delta z)^2]


//double DIcent = 0;


//DIcent = sqrt (pow (posx), 2) + sqrt (pow (posy), 2);

//cout<< DIcent <<endl;

//}


//need to write 'for' statement with 'if' statement within to determine particle types and count
```

```
//for (double i = 0; i <=0; lnum ++)

//{


//if pid = 2

//{

//ofstream file;

//int i = 0;

//float x = 100;

//file<< "PID\t


//need to count total number of daughter particles produced


//print to file



}


for(map<int,int>::const_iterator iter = particlecounts.begin() ; iter != particlecounts.end() ; iter++){

  cout<<iter->first<<" "<<iter->second<<endl;

}


}
```