Carnegie Mellon University
MELLON COLLEGE OF SCIENCE

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN THE FIELD OF PHYSICS

TITLE: "Determination of an Energy Determination Algorithm and the Optimized Detector Design for an Ultrahigh-Energy Cosmic Neutrino Experiment"

PRESENTED BY: Zhen Tang

ACCEPTED BY THE DEPARTMENT OF PHYSICS

James Russ	6/2/14
JAMES RUSS, CHAIR PROFESSOR	DATE
Stephen Garoff	6/2/14
STEPHEN GAROFF, DEPT HEAD	DATE

APPROVED BY THE COLLEGE COUNCIL

FRED GILMAN, DEAN DATE

Development of an Energy Determination Algorithm and the Optimized Detector Design for an Ultrahigh-Energy Cosmic Neutrino Experiment

by

Zhen Tang

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

at

Carnegie Mellon University

Department of Physics

Pittsburgh, Pennsylvania

Advised by Professor James S. Russ

May 29, 2014

Abstract

In this thesis, we discuss the optimization procedure for the TAUWER (TAU shoWER) experiment, which is designed to detect showers generated by Earth-skimming neutrinos. Monte Carlo Simulations are done through CORSIKA (COsmic Ray SImulations for KAscade) software to provide us with detailed information about hit patterns on the detector array from these showers. We use this to determine the trigger conditions, rates, and optimal detector layout. We also use machine learning classification methods to generate classifiers to assign the energy scale for observed showers.

Acknowledgments

I would like to express my profound gratitude to my advisor Prof. James Russ. It is my honor to be his Ph.D. student. Jim not only teaches me on high energy physics with his tremendous knowledge and incredible wisdom, but also provides me an example in the future with his great personality. I will always remember and follow his instructions in the future research, and treat other people nicely just like he treats me.

I would also like to thank my committee, Prof. Cosma Shalizi, Prof. Hy Trac and Prof. Helmut Vogel. Thank you for your comments on my thesis and great questions on my defense, especially for Prof. Helmut Vogel's detailed comments.

I thank Prof. Manfred Paulini. He is always very helpful in both research and normal life.

I thank Chunlei Liu, Haiyun Lu, Menglei Sun, Aristotle Calamba, Benjamin Carlson and all other friends who help me a lot in my research.

Finally, thanks to my family. Your constant support and encouragement really means a lot to me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Cosmic neutrinos

Experimental results from cosmic ray experiments as of now have covered energy region of cosmic rays from below 10^9 eV to several 10^{20} eV as shown in $Fig.1.1$. This spectrum can be described by a power law with two kinks. The first one is a "knee" at about 3 PeV (1 PeV = 10^{15} eV), where the curve steepens from $E^{-2.7}$ to E^{-3}. And the second one is an "ankle" at about 3 EeV (1 EeV = 10^{18} eV), where the curve starts to flatten. Although it is well accepted that the cosmic rays below the knee are from galactic sources such as supernova remnants[23], and cosmic rays above the ankle come from extra-galactic sources, as indicated by the high level of isotropy[4], yet we have no clue about the origins of these ultra-high-energy cosmic rays (UHECRs) above the knee, e.g., the locations and compositions of them. At the same time, according to the "GZK cutoff" effect[42], proposed by Greisen, Zatsepin and Kuzmin, cosmic rays (protons or heavier nuclei) with energies in excess of 40 EeV cannot travel for longer than 50 Mpc, due to the interaction with the cosmic microwave background (CMB) radiation[30], which contradicts the previous ideas that these UHECRs are from extra-galactic sources. The experimental evidence remains

unclear both about the nature of UHECRs and their acceleration mechanism and source.

Relativistic shock acceleration is one possible acceleration mechanism for UHECRs. Protons can accumulate very high energy through the rebounding between plasma shock waves over billions of years. Gamma-ray bursts (GRBs) and active galactic nuclei (AGN)[20] are two possible ways to generate UHECRs by this mechanism. However, many theoretical difficulties still remain unexplained and limited solid evidence has been found so far to indicate a specific source. Hence new experiments should be designed and appropriate probes need to be selected to provide us with information about these greatest mysteries of modern physics[12][33], e.g., the mechanism of how cosmic rays can obtain such high energies.

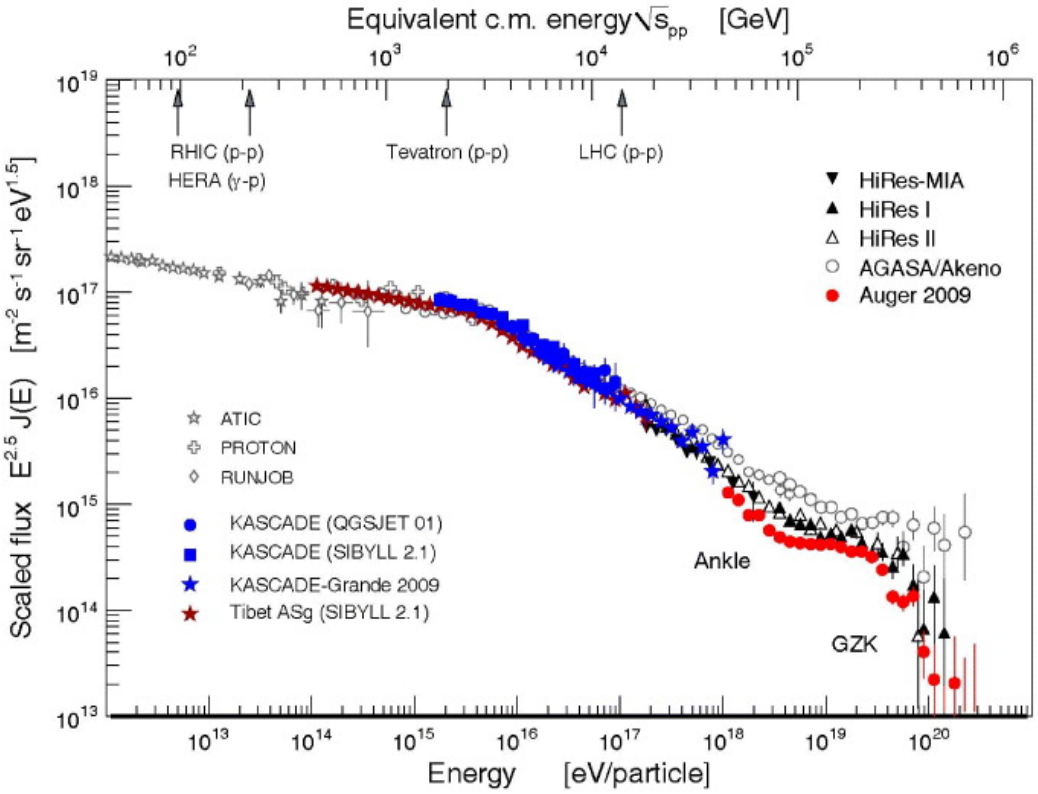


Figure 1.1: Cosmic ray energy spectra of various experiments[3]. Different experiments provide similar spectrum for energies lower than the GZK limit, but large discrepancy exists above the GZK limit

Cosmic neutrinos have long been proposed as an ideal probe to explore the above questions for several reasons. Firstly, they are charge neutral and hence they can travel in straight line without being deflected by the magnetic fields. Secondly, they interact very weakly with ambient matter and radiation and therefore can travel across cosmological distances to the Earth[40]. As we will see in the following part, very close relations are exist beween UHECRs and these neutrinos. Great progress can be made in understanding the origin of UHECR if we can obtain the information like the energy spectrum, arrival direction and flavor ratios of the ultra-high-energy (UHE) cosmic neutrinos[27].

The cosmic neutrino spectrum, from one model of UHECR production, generated by proton sources undergoing relativistic shock acceleration up to a maximum energy of 200 EeV, is shown in $Fig.\ 1.2$[34], with source power spectrum $\alpha = 2.4$, where α is defined as Flux$\propto E^{-\alpha}$. UHE protons can interact through photopion production via Δ resonance: $p\gamma \rightarrow \Delta(1232) \rightarrow \pi^+ n$. Both of the secondary particles can decay to neutrinos: pion decay $\pi^+ \rightarrow \mu^+\nu_\mu \rightarrow e^+\nu_e\bar{\nu}_\mu\nu_\mu$ and neutron decay $n \rightarrow pe\bar{\nu}_e$. The neutrinos from pion decay inherit on average $0.05E_p$ while neutrinos from neutron decay acquire only $3 \times 10^{-4}E_p$[34]. Based on this fact we can derive that $[E_\nu^2 d\Phi_\nu/dE]^{CMB,n-decay}_{E_\nu=6\times10^{15}eV} = 2 \times 10^{-3}[E_\nu^2 d\Phi_\nu/dE]^{CMB,\pi-decay}_{E_\nu=10^{18}eV}$, which is in accordance with the ν (CMB only) curve in $Fig.\ 1.2$. The Δ resonance production peaks when $m_\Delta^2 = m_p^2 + 2E_pE_\gamma(1 - cos\theta)$, where θ is the angle between the proton and γ momenta in the lab frame, which means $E_p(\Delta) \simeq 1.6\times10^{17}/(E_\gamma/eV)$. When the protons propagate through CMB where $E_\gamma \sim 0.7\times10^{-3}eV$, we can have the proton energy at the Δ resonance peak $E_p(\Delta) \simeq 2.3 \times 10^{20}eV$, which reveals the fact that the interaction between protons with energies in excess of 6×10^{19} eV and CMB is the main cause of EeV neutrino spectrum. Experimental data from Auger[13] provides us the upper limit of diffuse neutrino flux at EeV energies for all three flavors $E_\nu^2 d\Phi_\nu/dE < 3 \times 10^{-7}GeVcm^{-2}sr^{-1}s^{-1}$. The measurements of the diffuse photon background by Fermi LAT experiments lower the amount of low energy photons in CMB and

hence further lower the upper limit to $E_\nu^2 d\Phi_\nu/dE < 5 \times 10^{-8} GeV cm^{-2} s^{-1} sr^{-1}$. With the relationship equation between flux of pion decay and neutron decay we have just derived, we can have the upper bound flux of PeV neutrinos from CMB neutron decay as $10^{-10} GeV cm^{-2} s^{-1} sr^{-1}$. This is too small to be the major source of PeV neutrinos, based on the observation of two PeV neutrinos by IceCube[1], as shown in $Fig.\ 1.2$. Since the UV/optical/IR radiation backgrounds will provide much higher energy photons compared to CMB, the value of corresponding $E_p(\Delta)$ will be much lower than in the CMB case, based on the relation $E_p(\Delta) \simeq 1.6 \times 10^{17}/(E_\gamma/eV)$. Numerical simulations show that interaction between UV/optical/IR photons and protons with energy below 10^{18} eV is the main cause of the PeV scale neutrino flux. The experiment result from IceCube indicates this flux for all neutrino flavors is $3.6 \times 10^{-8} GeV cm^{-2} sr^{-1} s^{-1}$. However, due to the wide Δ resonance peak, wide thermal spectrum of CMB photons and redshifts, there are still considerable neutrino flux expected between the two peaks ($\sim$ PeV and EeV), as seen in $Fig.\ 1.2$, looking at the neutrino flux curve between $10^{15} eV \sim 10^{18} eV$.

The situation will change if the sources are nuclei instead of protons. The cosmic ray spectrum with respect to Fe sources is shown in $Fig.\ 1.3$, where the maximum energy of Fe is 5200 EeV with source spectrum $\alpha = 2.0$. Photopion processes between nuclei and CMB photons will be suppressed since this process will require energies of nuclei above $\sim\ AE_{GZK}$, where A is the mass number of nucleus, which leads to a negligible diffuse nuclear flux. Yet in an UV/optical/IR background, photopion processes can produce PeV neutrinos with nuclei energies $\sim\ 20AE_\nu$. But this energy is much higher than the scenario for protons, and hence makes the protons the dominant source for PeV neutrino peak. On the other hand, the photodisintegration process will also impact the neutrino spectrum with nuclear sources. Only the CMB mode will be considered since neutrinos produced in UV/optical/IR radiation backgrounds via this process will have energies below 10^{14} eV. In photodisintegration, because $Z \sim A/2$ for typical cosmic nu-

clei, about half of the secondary nucleons are neutrons. Hence the upper limit of PeV neutrino flux from secondary neutron decay can be derived as $[E_\nu d\Phi_\nu/dE]^{CMB,n-dec}_{E_\nu=10^{15}eV} \simeq 2\times10^{-4}[E^2 d\Phi_{CR}/dE]_{E=2.5\times10^{18}eV} < 10^{-11}GeVcm^{-2}s^{-1}sr^{-1}$, while the upper limit flux with proton sources is about $10^{-9}GeVcm^{-2}s^{-1}sr^{-1}$. Consequently, the neutrino flux with nuclear sources is highly suppressed compared to that with proton sources.

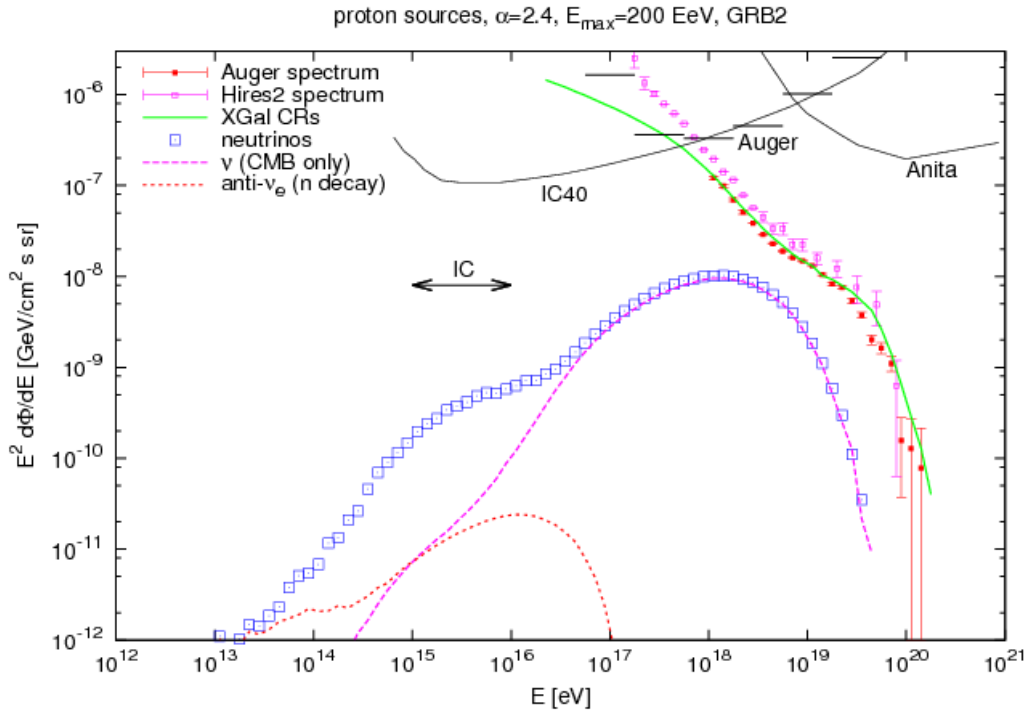


Figure 1.2: Cosmic neutrino spectrum with proton sources. The blue curve with label "neutrinos" is the result of Monte Carlo simulation as described in the text. The estimations of neutrino flux from IceCube[2], Auger[5] and Anita[22] are also shown in this plot. The estimation from IceCube is based on the two PeV neutrino events they observed

Both two scenarios have the same initial ratio among three flavors as $\nu_e : \nu_\mu : \nu_\tau = 1 : 2 : 0$. Due to oscillations with maximal mixing along cosmic distances[6], the final ratio of neutrinos when they arriving the Earth will become $\nu_e : \nu_\mu : \nu_\tau = 1 : 1 : 1$. Identification of UHE ν_τs will bring us pure information about the distribution and energy spectrum of cosmic ray sources, since the atmospheric neutrino background does not extend into the UHE regime.

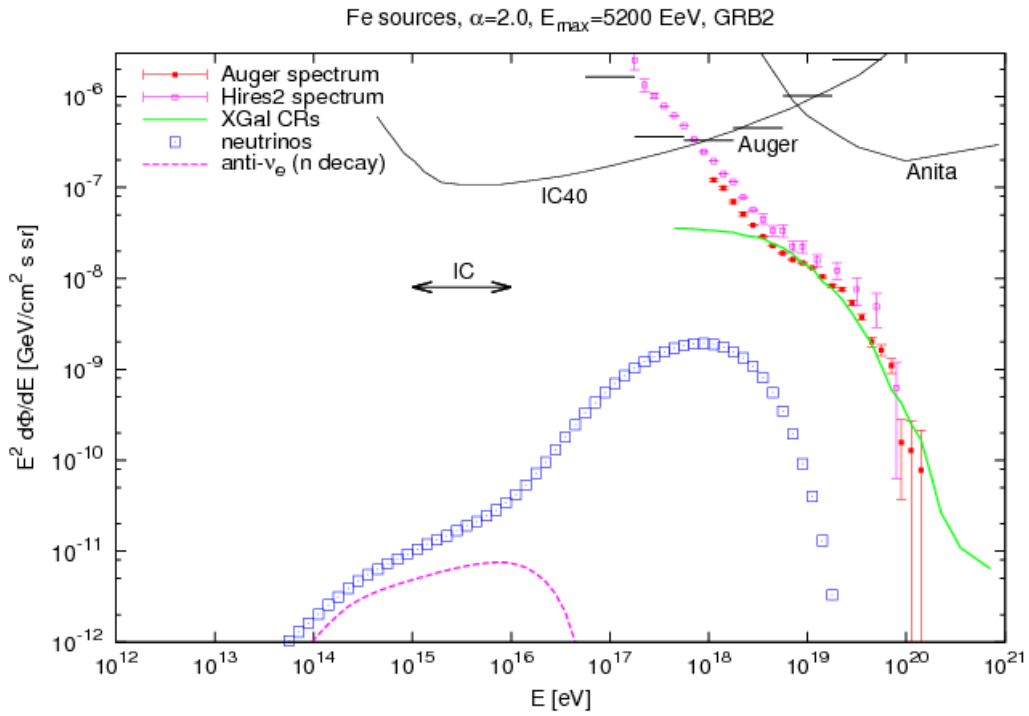


Figure 1.3: Cosmic neutrino spectrum with Fe sources

1.2 TAUWER physics

As we have discussed in the last section, understanding the shape and amplitude of neutrino spectrum at ultra-high energy region can help us answer the most interesting questions in astroparticle physics, such as the acceleration mechanism for UHECRs and the origin of the highest energy objects in the universe. Several neutrino experiments are currently running, e.g., IceCube, ANITA, ARIANNA, and ARA. IceCube is trying to detect the signals from upward-moving neutrinos with near 180° zenith angle, which requires the neutrinos and their interaction products to travel the whole earth. This requirement limits the upper bound of the neutrino energy at 10 PeV. The ANITA experiment is using a ballon-borne radio Cerenkov detector to detect the neutrino signals summed over all incident directions. It has a lower bound for neutrino energy at 5 EeV. The subsequent ice cap experiments ARA[8] and ARIANNA[10] at the South Pole, which are also pursuing the idea of radio Cerenkov

detection, relying on long propagation distances for radio signals in ice to cover a very large sensitive volume compared to IceCube or TAUWER. Their electronic sensitivity decreases this lower bound to 1 EeV.

TAUWER is designed to detect hadronic air showers generated by Earth-skimming ν_τ at the energy scale from 10 PeV to 200 PeV, which covers the energy regime between the cutoff of IceCube sensitivity due to finite detection volume and the threshold for radio Cerenkov detection in ARA37 or ARIANNA. If an Earth-skimming ν_τ interacts with the rock via weak interaction charged current channel as shown in $Eq.\ 1.1$, a τ lepton will be generated in the direction very close to its parent neutrino. This τ will inherit about 80% of ν_τ energy and decay after it escape into the air, via all hadronic modes. The secondary particles including pions and photons can develop into showers. Such showers will be the targets we want to detect. More details about this process will be analyzed in Chapter 2.

$$\nu_\tau N \rightarrow \tau X \tag{1.1}$$

Neutrinos at small zenith angle with the energy regime we are interested in will rarely interact in the low-density material of the atmosphere and hence remain invisible to ground detectors. Upward-moving neutrinos in the 10-200 PeV energy range with zenith angles near 180°cannot penetrate the whole Earth to be observed by, e.g., IceCube.

1.2.1 TAUWER detector station and array

Our proposed detector array consists of 1600 detector stations covering 2.5 km^2, with the distance between each two stations at 40 meters, as shown in the right part of $Fig.\ 1.4$. This design is based on the detector optimization, to be described in Chapter 3. The whole array will be put on a steep mountain slope with an inclined angle about 30° to the horizontal plane, facing another large mountain on the opposite site. The purpose of the steep slope

is to provide elevation for rows of detector stations. The detector plane for each station will be oriented approximately normal to the τ shower axis, in the range of 0° to 10° to the horizontal plane, as shown in $Fig.\ 1.5$. The two mountains can suppress the background signals from horizontal cosmic air showers, and the way we put these detectors can maximize the signals from the τ air showers, since the detector plane will be perpendicular to the shower axis. Each detector station includes two 20×40 cm scintillators, parallel to each other and separated by 1.6 meter as shown in the left part of $Fig.\ 1.4$. This design can provide us information about time of flight (TOF), which can help to distinguish the directions of the incident showers.

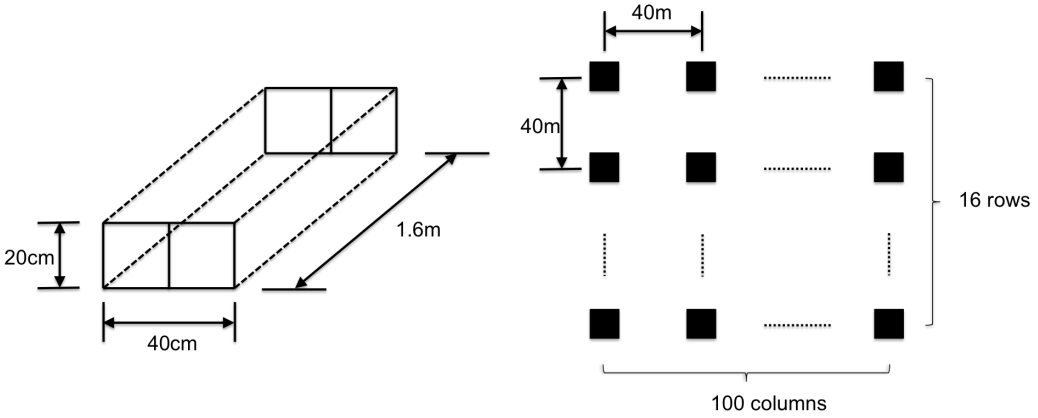


Figure 1.4: The left side shows the geometry of single TAUWER detector station and the right side provides a view of whole array

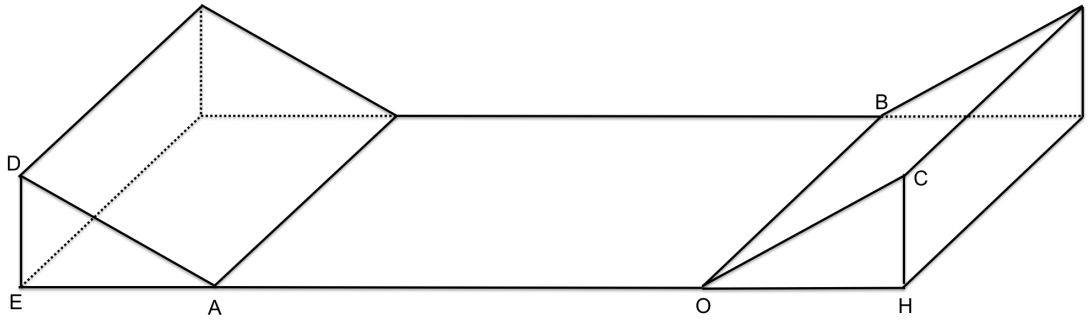


Figure 1.5: Geometry requirements for detector array

1.2.2 KIT test

A test has been done using the Kaskade EAS array at the Karlsruhe Institute of Technology (KIT) for two years to show the response of the TAUWER detector stations to extensive air showers (EAS) events, since this is one of the main sources of background. Eight stations were used, with four at the corners of a square 20m and the others at the corners of a square 40m. The detector was triggered if the signal was five times the mean noise level. Triggered events were time-matched to EAS events detected in the full Kaskade array. This experimental result shows that each TAUWER station will rarely have more than one hit per EAS event, which matches the result from CORSIKA simulations for the vertical air showers quite well. Meanwhile, as we will see more details in Chapter 3, the hit patterns for the τ air showers we are trying to detect are quite different from that of EAS events. For τ showers, the detector near the intersection point of shower axis and the inclined plane where we put the detector array on will always have many hits. Therefore, the EAS events can be effectively excluded as the background noise from the signals of target showers we want to detect.

In this thesis, we will firstly describe the whole process about how an Earth-skimming neutrino transfers to a detectable shower, and give the estimation of this process (Chapter 2). Then the simulation results using the shower simulation package CORSIKA will be described and analyzed in Chapter 3. Based on the simulation results we predict the hit pattern distribution for each shower. We use classification methods to generate energy classifiers, which can be used to decide the energy scale for showers in data, based on their hit patterns (Chapter 4). Meanwhile, we would also like to introduce some of the key computing techniques we have used for our research, especially the experience about supercomputing system (Chapter5). The future work and conclusions will be discussed at the last two chapters.

Chapter 2

Shower Evolution

Our goal is to detect earth-skimming neutrinos. Hence we would like to analyze the process of τ neutrinos going through the shell of earth, interacting with the rock and generating a τ lepton. If the τ lives long enough to escape from the Earth and decays via a hadronic decay mode (64% probable), then the high energy decay particles will develop air showers in the atmosphere. These showers will propagate along the τ direction and can be detected by our detector array. Five of the most significant decay modes of τ lepton are listed as follows,

$$\tau^- \to \pi^- \, \pi^0 \, \nu_\tau \quad (25.52 \pm 0.09\%) \tag{2.1}$$

$$\tau^- \to \pi^- \, \nu_\tau \quad (10.83 \pm 0.06\%) \tag{2.2}$$

$$\tau^- \to \pi^- \, \pi^0 \, \pi^0 \, \nu_\tau \quad (9.30 \pm 0.11\%) \tag{2.3}$$

$$\tau^- \to \pi^+ \, \pi^- \, \pi^- \, \nu_\tau \quad (8.99 \pm 0.06\%) \tag{2.4}$$

$$\tau^- \to \pi^+ \, \pi^0 \, \pi^- \, \pi^- \, \nu_\tau \quad (2.70 \pm 0.08\%) \tag{2.5}$$

Our detector stations will primarily detect electrons and muons, which are the final long-lived particles from the hadronic decay of τ lepton. Photons play a small role, as will be described later. The number of detected particles depends on the initial energy of the

shower. Hence we can neglect the difference between the decay modes and only focus on the energy goes into each shower.

Meanwhile, two other neutrinos types, ν_e and ν_μ, can also interact with the Earth shell. However, both electrons and muons generated via CC interactions cannot produce hadronic air showers to be detected by the array. Hence we only consider the ν_τ case.

In this section, we will introduce models related to each step we just described. The probabilities for neutrino's charged-current interaction with the Earth shell and the decay of the generated τ lepton will be estimated. By combining them we can derive the probabilities of detecting showers generated from τ neutrinos with different energies and incident angles. The simulation results from CORSIKA will also be displayed at the end of this section and comparisons will be made between these results and our analysis.

2.1 Interaction of neutrinos in the Earth shell

2.1.1 Charged-current interaction of neutrino

For our project, we are interested in a ν_τ's interaction with a nucleus in the Earth shell through the charged-current interaction, since we will need the produced τ lepton to generate a shower for detection. The charged-current interaction can be represented as,

$$\nu_\tau N \rightarrow \tau X \tag{2.6}$$

For a neutrino energy between 10 PeV and 200 PeV, the standard model cross section for a neutrino charged-current interaction with a nuclear is, according to CTEQ4-DIS[21] or

$Ref.$ [36].

$$\sigma_{CC} = 5.53 \times 10^{-36}\text{cm}^2(E_\nu/1\text{GeV})^{0.363} \tag{2.7}$$

The charged-current interaction length can be derived as,

$$L_{int} = \frac{1}{\sigma_{CC} N_A \rho_{rock}/A} \tag{2.8}$$

where $N_A = 6.022 \times 10^{23} mol^{-1}$ is the Avogadro's constant, $\rho_{rock} = 2.65 g/cm^3$ is the density of the rock (the main material of earth shell), and $A = 1 g/mol$ is the molar mass of nucleon. Because of the short range of the weak interaction, each nucleon in the target molecule adds its cross section incoherently.

Then the interaction length can be expressed as,

$$\begin{aligned} L_{int} &= \frac{1}{5.53 \times 10^{-36} \times (E_\nu/1\text{GeV})^{0.363} \times 6.02 \times 10^{23} \times 2.65} cm \\ &= 7.524 \times 10^3 \times (\frac{1PeV}{E_\nu})^{0.363} km \end{aligned} \tag{2.9}$$

From $Eq.$ 2.9 we can see that the charged-current interaction length of τ neutrino is about 3262 km for 10 PeV and 1099 km for 200 PeV. Older other sources with different cross section values, such as [16], agree with the energy dependence and are within a factor of $2^{\pm 1}$ in total cross section.

The neutrino's neutral-current interaction cross section is 2.4 smaller than that of charged-current interaction, which means the interaction length of neutrino through neutral-current channel is about 2.4 times greater than that of charged-current channel. Hence we can conclude that with a very high probability there is no neutral-current interaction before the charged-current interaction. Therefore, the neutrino will almost have the same energy as it

just enters the Earth shell when it interacts through a charged-current channel.

Assume the trajectory of a neutrino in the earth is shown in $Fig.\ 2.1$. Point A is the incident point of ν_τ. At point C, the neutrino interacts with rock via charged-current channel and generates a τ lepton. The τ keeps going without decaying until it get out of the Earth shell through point B.

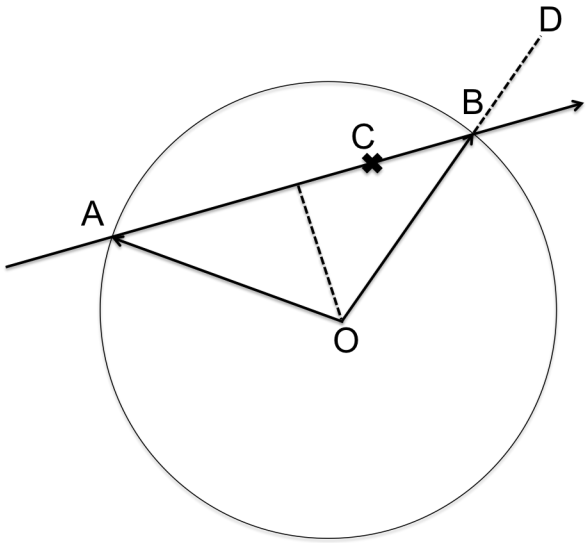


Figure 2.1: Trace of neutrino in the earth shell

We assume the distance between point A and point B is L, the distance between point C and point B is x, then the distance between point A and point C is L-x. The probability of ν_τ goes through point A to point C and decay at C can be shown as follows, where we use dx to represent the short distance where the charged-current interaction happens.

$$P_{int} = e^{-(L-x)/L_{int}} \frac{dx}{L_{int}} \tag{2.10}$$

Lepton inelasticity y is defined as follows,

$$y = \frac{E_\nu - E_\tau^i}{E_\nu} \tag{2.11}$$

where E_ν is the energy of ν_τ before the interaction, E_τ^i is the energy of τ lepton when it is generated.

Meanwhile, the moving direction of generated τ lepton generated will have very small intersection angle between that of the ν_τ since the angle $\Delta\theta \propto \Delta P_T/E_\tau$ and ΔP_T is in the order of W boson mass. Hence we consider they are travelling along the same direction.

$< y >= 0.2$ is the average inelasticity for the energy scale we are interested in according to [16][19][21]. Hence we have,

$$E_\tau^i = 0.8E_\nu \tag{2.12}$$

2.1.2 Survival of τ lepton in the Earth shell

Now we need to consider the energy loss of τ lepton when it travels along point C to point B. The generally used equation to describe the average energy loss of τ is shown as follows[16][28],

$$< \frac{dE_\tau}{dx} >= -(\alpha + \beta E_\tau)\rho_{rock} \tag{2.13}$$

where $\alpha \simeq 2 \times 10^{-3} GeV\ cm^2/g$ represents the ionization energy loss[37], and β represents energy loss from Bremsstrahlung, pair production and photonuclear interaction. Within the high energy scale from 10^7 GeV to 2×10^8 GeV where we are interested in, $\beta E_\tau \simeq 6 \times 10^{-7} E_\tau cm^2/g$ is much larger than α and hence the ionization energy loss can be neglected compared to radiative energy loss.

For analytical purposes, one approximation is made as follows:

$$< \frac{dE_\tau}{dx} > \simeq \frac{dE_\tau}{dx} \tag{2.14}$$

This assumption is made because when radiation processes dominate the energy loss, the energy loss distribution is nearly symmetric about its mean value, in contrast to the asymmetric high-loss tail that characterizes ionization-dominated energy loss.

Therefore we have the energy loss formula for the τ as follows,

$$\frac{dE_\tau}{dx} \simeq -\beta\rho E_\tau \tag{2.15}$$

where we use ρ to represent ρ_{rock} for notational simplicity.

The parameter β can depend on E_τ. For $Ref.$ [16], three parameterizations were studied,

$$\text{(1) } E_\tau = E_\tau^i e^{-\beta\rho x},\ \beta = constant \tag{2.16}$$

$$\text{(2) } E_\tau = \frac{E_\tau^i \beta_\tau e^{-\beta_\tau \rho x}}{\beta_\tau + \gamma_\tau E_\tau^i (1 - e^{-\beta_\tau \rho x})}$$

$$\beta = \beta_\tau + \gamma_\tau E_\tau \tag{2.17}$$

$$\text{(3) } E_\tau = exp\Big(-\frac{\beta_0}{\beta_1}(1 - e^{\beta_1 \rho x}) + ln(E_\tau^i / E_0) e^{-\beta_1 \rho x} \Big) E_0$$

$$\beta = \beta_0 + \beta_1 ln(E/E_0) \tag{2.18}$$

where x is the distance the τ lepton travelled.

For solution (1), we have $\beta = 0.85 \times 10^{-6} cm^2/g$. This solution is appropriate for $E_\nu = 10^{10}$ GeV but does not perform well for lower energies, hence we will not use it. For solution (2), we have $\beta_\tau = 0.71 \times 10^{-6} cm^2/g$ and $\gamma_\tau = 0.35 \times 10^{-18} cm^2/(g\ GeV)$[9]. Solution (3) is derived by adding a logarithmic energy dependence and we have $\beta_0 = 1.2 \times 10^{-6} cm^2/g$, $\beta_1 = 1.6 \times 10^{-7} cm^2/g$ and $E_0 = 10^{10} GeV$[18].

Monte Carlo simulation results show that the solution (3) fits best over our energy range of interest, as indicated in Fig. 6 in [16]. Hence we will use solution (3) in the following part of this section.

One problem we need to consider here is the role of energy loss in determining the maximum traveling distance of the τ lepton in rock, X_{th}. To detect the hadronic shower generated from the τ lepton, we have to impose a lower bound on the τ energy when the τ exits from the Earth shell.

Based on the simulation results, we take $E_{th} = 10$ PeV. If the threshold energy is lower than 10 PeV, very few tracks in the shower will be detected by our array. Thus lower energy events are of no use for our analysis. Details will be described in the next Chapter.

From $Eq.\ 2.18$, we have,

$$E_\tau(x) = exp\big(-\frac{\beta_0}{\beta_1}(1 - e^{\beta_1 \rho x}) + ln(E^i_\tau/E_0)e^{-\beta_1 \rho x}\big)E_0 \tag{2.19}$$

which shows how the τ energy decreases when it travels. We define the maximum traveling distance X_{th} as the distance the τ lepton has travelled before its energy drops below the energy threshold E_{th}, which can be represented by the following equation,

$$E_{th} = E_\tau(X_{th}) \tag{2.20}$$

Combining $Eq.\ 2.19$ and $Eq.\ 2.20$ we can obtain the numerical solution of X_{th} by solving the following equation,

$$E_{th} = exp\big(-\frac{\beta_0}{\beta_1}(1 - e^{\beta_1 \rho X_{th}}) + ln(E^i_\tau/E_0)e^{-\beta_1 \rho X_{th}}\big)E_0 \tag{2.21}$$

If X_{th} is less than the distance between point C and point B, we can directly conclude that the probability to detect the corresponding shower generated by this τ lepton is zero.

$Fig.$ 2.2 shows the relationship between the initial energy of τ lepton and its maximum traveling distance in rock. Based on this plot, we would like to set the upper limit of x to be 50 km, which means that the neutrino must interact with the earth shell through charged-current channel and turn out into a τ lepton within 50 km of the surface of the Earth.

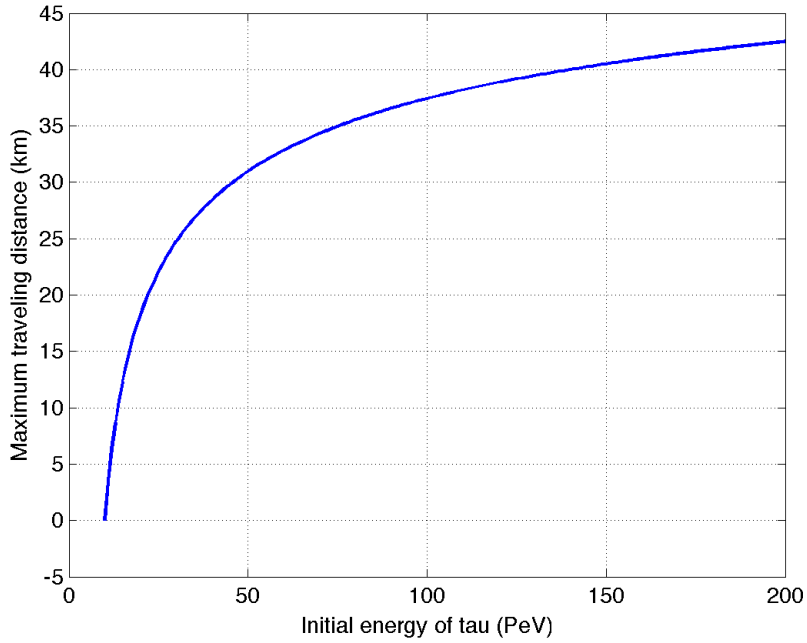


Figure 2.2: The range of τ in the rock with respect to its initial energy. This range is defined as the average distance that τ will travel before its kinetic energy has been reduced to the threshold energy by the average energy loss

However, most τ leptons cannot survive for such a long distance because of the τ lifetime. The τ decay length can be represented as follows,

$$L_d = \gamma c t_0 \simeq 5 \times 10^{-8} \frac{E_\tau}{1GeV} km \tag{2.22}$$

According to $Eq.$ 2.22, we can see that for τ with energy 10^7 GeV, the decay length is 0.5km, and hence only when the neutrino interaction happens very close to the Earth's surface can the generated τ lepton survive into the air, or else the τ will decay into a

shower and this shower will be absorbed immediately by the rock and cannot be detected by our detector stations. For τ with larger energy such as 2×10^8 GeV, the decay length is about 10 km, and we can allow a much larger distance region for distance between point C and point B in $Fig.\ 2.1$ compared to the low energy case.

The survival probability of the τ with respect to the distance it travelled can be expressed as follows,

$$\frac{dP_{surv}}{dx} = -\frac{P_{surv}}{ct_0 E_\tau / m_\tau} \tag{2.23}$$

where t_0 is the lifetime of τ lepton.

Since $E_\tau = E_\tau(x)$, we can have the following equation,

$$\frac{dP_{surv}}{P_{surv}} = -\frac{-dx}{ct_0 E_\tau(x) / m_\tau} = -f(x)dx$$
$$\text{where } f(x) \equiv \frac{1}{ct_0 E_\tau(x) / m_\tau} \tag{2.24}$$

From $Eq.\ 2.24$ we can derive the solution of survival probability as follows, Since $E_\tau = E_\tau(x)$, we can have the following equation,

$$P_{surv} = N exp\{-\int_0^x f(z)dz\} \tag{2.25}$$

where N is the normalization coefficient.

If the traveling distance of τ lepton is zero, then we have the survival probability of unity,

which can be used to obtain N as follows,

$$P_{surv}|_{x=0} = 1$$

$$\Rightarrow \ N = 1$$

$$\Rightarrow \ P_{surv} = exp\{-\int_0^x f(z)dz\} = exp\{-\int_0^x \frac{1}{ct_0 E_\tau(z)/m_\tau}dz\}$$

$$\text{where } E_\tau(z) = exp\{-\frac{\beta_0}{\beta_1} + (\frac{\beta_0}{\beta_1} + ln(E_\tau^i/E_0))e^{-\beta_1\rho z}\}E_0 \tag{2.26}$$

where $t_0 = 2.906 \times 10^{-13}$ s is the mean lifetime of τ lepton, and $m_\tau = 1776.82$ MeV is the mass of τ. $Fig.$ 2.3 shows the survival probabilities of τ leptons with several typical initial energies versus the distance they passed.

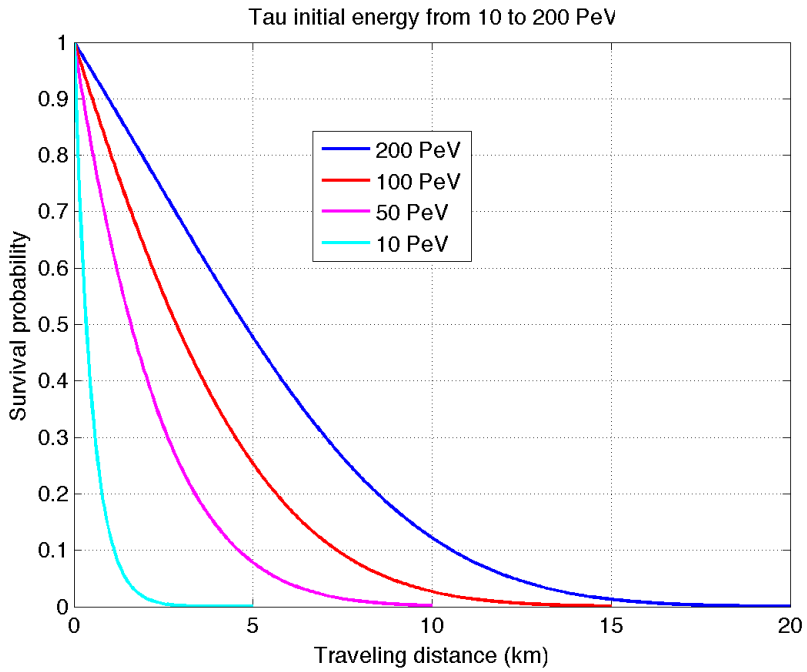


Figure 2.3: The survival probability of τ with initial energy from 10 to 200 PeV versus the distance it passed

Combining $Eq.$ 2.26 with $Eq.$ 2.10 which represents the probability that ν_τ travels through point A to point C and interacts with rock through charged-current channel at point C, we

can have the final probability to have τ lepton escapes from the Earth,

$$P_{eject} = \int P_{int} P_{surv}$$
$$= \int_0^L dx e^{-(L-x)/L_{int}} \frac{1}{L_{int}} exp\{-\int_0^x \frac{1}{ct_0 E_\tau(z)/m_\tau} dz\} \tag{2.27}$$

Assume $\angle$ABD in $Fig.\ 2.1$ is θ, then the distance between neutrino incident point A and τ escape point B is,

$$L = 2\pi R \cos(\pi - \theta) \tag{2.28}$$

where $R = 6371$km is the radius of the Earth.

We calculate the escape probability here with the incident angle θ from 91 to 98 degree. The result is listed in $Table\ 2.1$. The reason why the probabilities are so small is the term dx/L_{int}. The interaction length of neutrino is greater than 1000 km for the energy scale we are interested in while the effective integral path of dx is less than 50 km due to the short decay length of τ lepton.

Table 2.1: Escape probability with respect to θ and ν_τ energy in units of 10^{-4}

θ(degree) \ Energy (PeV)	10	20	50	100	200
91	1.2	2.8	8.5	18	35
92	1.1	2.6	7.5	15	29
93	1.0	2.4	6.7	13	23
94	0.97	2.2	5.9	11	19
95	0.91	2.0	5.2	10	16
96	0.85	1.8	4.6	8	13
97	0.79	1.7	4.1	7	10
98	0.74	1.5	3.6	6	9

The long neutrino interaction length and finite size of active detection volume combine to make the detection probability per neutrino low in all astrophysical neutrino experiments,

not just this one. The active volume for this proposal is several times larger than that for IceCube, for example.

2.2 τ lepton in the air

If the τ lepton decays in the Earth's shell, the secondary particles would be absorbed by rock immediately and therefore cannot be detected by us. Therefore we require the τ lepton to decay after it escapes into the air. We use CORSIKA to simulate the generation of showers from τ leptons after they escape into the air with different energies from 10 to 200 PeV. As we show in Chapter 3, the τ has to decay far enough in front of the detector array to allow a hadron shower to develop. This will reduce the probabilities in $Table$ 2.1 somewhat, but not dramatically.

Chapter 3

Detector Optimization and Simulation

Neutrino interactions are very rare and require a very high detection efficiency per event. For hadronic air showers, the detection centers on the high density of upward-moving particles in a small region around the shower axis. EAS events develop from interactions in the upper layers of the earth's atmosphere. The showers develop at high altitudes, and shower tracks that reach the earth's surface are largely muons. The electromagnetic component has ranged out due to ionization losses. The EAS events have low particle density and move downward. The TAUWER detector array has to exploit those differences to be sensitive to neutrino interaction events and insensitive to the much-more prevalent EAS background rate in the 10-200 PeV range, which is about 10^5 times larger according to $Fig.1.1$ and $Fig.1.2$.

In TAUWER the detection efficiency will be of order 100% if a τ shower core passes close to several TAUWER detectors, so that one or more stations detects a large number of particles within a 20ns time window. In this section we study shower development characteristics in this energy range and determine the detector arrangement needed and the shower development length L that will set up the conditions for 100% detection.

3.1 Simulation process

After the τ lepton escapes from the Earth, the simulation is started. The first step is to have the τ decay. As we have already discussed at the beginning of chapter 2, there are five major hadronic decay modes for high energy τ leptons. We will use the TAUOLA package with a specific decay mode, shower distance and τ energy to generate the hadron energies and angles in the τ rest frame, and then boost them to the lab frame. The generated particles will include pions and photons with known energies and directions. The next step of simulation will be finished by CORSIKA.

CORSIKA[31] is a Monte Carlo simulation program widely used in astrophysics to generate air showers. Different geometry conditions can be set in this program and the energy scale for cosmic radiation is from 0.3 PeV to 1000 PeV. For our case, we will focus on the models CORSIKA uses for high energy hadronic interactions since our interested energy scale is from 10 PeV to 200 PeV. Six models are available in CORSIKA: DPMJET[29], HDPM[24], QGSJET01[29], SIBYLL[7], VENUS[41] and NEXUS[15]. The comparisons between these models can be found at [32][17]. However, since there is no experimental evidence to show the correctness of any of these models until now, all the six models remain to be theoretical guesses. In our case, we use the default model QGSJET01 to do the simulation.

Each high energy pion or photon starts its own shower and CORSIKA follows the development until the shower has reached the detector plane defined in the program. Here, the detector plane is an inclined plane indicated in $Fig.\ 3.1$. All the shower particles from the pions and photons in the τ decay are summed to make the CORSIKA event file. Since we need to define the decay geometry for CORSIKA to do simulations, sampling points along the decay distribution need to be selected. Each sampling point will have a weight from the actual decay distribution, and then we can approximate the integral decay distribution by a

finite set of sampling points. The decay distances we selected are 2.5 km, 3 km, 5 km, 7.5 km, 10 km and 15 km in front of the detector plane. More details will be discussed in the following part to see why only these distances are selected.

The simulation is done by our team members at Roma for all five main decay modes of τ leptons. Due to the limitation of computing resources, only several energies of τ leptons are simulated, they are 10 PeV, 20 PeV, 50 PeV, 100 PeV and 200 PeV.

3.2 Simulation data

The ASCII data files we obtained can be directly read. The first two lines give us the incident angle and energy of τ leptons, followed by lines of track information. Each line gives the information of one track arriving at the detector plane in the following format,

Track number, PID, PP, Px, Py, Pz, X, Y, Z, T, weight (3.1)

The PID is the particle ID. PP is the total momentum of each track, while Px, Py and Pz are the momentum along x, y, z direction which are defined in $Fig.\ 3.1$. X, Y and Z are coordinates of the interaction point between the track and the inclined plane. T provides us the time stamp information. The weight number here means we use "THIN" option in CORSIKA, which is defined in [31]. By applying the "THIN" option, only part of the secondary particles with energy greater than a limit we set will be followed in the simulation. For showers with energy greater than 10 PeV, this procedure can reduce the computing time dramatically. Meanwhile, the size of generated data files will be greatly compressed, which reduces the load for transferring these data from Roma to us. For example, the data file corresponding to 100 PeV at 5 km distance for 3π mode is 5.1 GB. If the "THIN" option is not applied, then the size will increase to about 84 GB. With the

weight number coming from "THIN" option, we can retrieve the full shower based on the algorithm we will explain in the section "Obtain unthinned tracks".

3.2.1 Pre-selection

Before we do anything to the data file we obtain from CORSIKA, a pre-selection according to the particle ID is done. Many lines come from tracks we are not interested in. Hence, if we can get rid of them at the beginning, it is much more convenient for the following steps. Very few pions survive to reach the detector array. If they don't get to the plane, they are not in the file. Photons count rarely because they deposit a very small fraction of their energy in our scintillator detectors. A single photon rarely makes a coincidence, because that requires the product of two small probabilities. Therefore, only electrons, muons and their corresponding anti-particles will be selected for further analysis.

Such selection saves us time in running the following unthin step and significantly reduces the size of the generated unthinned files.

The time spent on this pre-selection step is negligible compared to the other steps. Each data file usually contains 100 showers. We split the 100 showers as well as the pre-selection by perl codes named "divide.pl", which can be found in Appendix A.1.

3.2.2 Obtain unthinned tracks

After the pre-selection, we have significantly decreased the size of the data file. But these files cannot be directly used because of the weight number at the end of each line in the data file. As we have just discussed, these weight numbers can be used to retrieve the whole shower based on the unthin algorithm we will introduce now. $Fig.\ 3.1$ describes the geometry of showers and inclined plane.

Point A is the τ decay point of the shower, with y and z coordinates fixed. To calculate the x

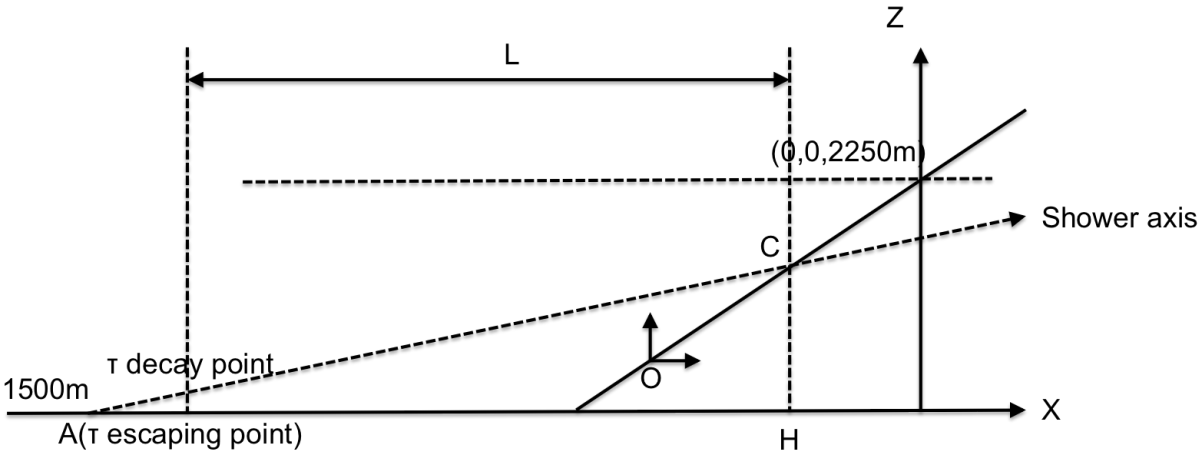


Figure 3.1: Shower center and inclined plane

.

coordinate of A, we need to read out the incident angle θ from the data file for each shower, which is uniformly distributed between 2.0° to 2.5°. The normal plane is perpendicular to the mean shower axis which has an angle 2.5° to the horizontal plane. By construction the mean shower axis always passes a fixed point (0, 0, 2250). The coordinates of A in meters are,

$$A_x = -750/tan(\theta)$$
$$A_y = 0, A_z = 1500 \tag{3.2}$$

Point C is the intersection point of shower axis and the inclined plane. Since we know the coordinate of A and incident angle θ, we can easily get the coordinate of C by compare the equations for the line AC and the inclined plane.

The equation for AC is,

$$z - z_a = tan(\theta)(x - x_a) \tag{3.3}$$

and that for the inclined plane is,

$$z - z_o = tan(\alpha)(x - x_o) \tag{3.4}$$

As L changes, the origin of the detection plane moves as shown in $Table\ 3.1$, where the L is defined as the mean distance between τ decay point and C.

Table 3.1: Origin of the detection plane with respect to L

L(km)	Coordinates of O
2.5	(-16700,0,1600)
3	(-15500,0,1640)
5	(-14000,0,1690)
7.5	(-11500,0,1790)
10	(-9000,0,1900)
15	(-4000,0,2100)

Therefore, the C's coordinate is,

$$\begin{aligned} x_c &= (-x_a tan(\theta) + x_o tan(\alpha) + z_a - z_o)/(tan(\alpha) - tan(\theta)) \\ y_c &= 0 \\ z_c &= tan(\theta)(x_c - x_a) + z_a \end{aligned} \tag{3.5}$$

Take one track as an example to show how we do the unthin step. We can have the coordinates of the intersection point between this track and the inclined plane from the data file, as well as its direction. Define that intersection point as P, and the direction of this track as $\vec{n}_p$. Then we can have,

$$\begin{aligned} P &= (x_1, y_1, z_1) \\ \vec{n}_p &= (p_x, p_y, p_z) \end{aligned} \tag{3.6}$$

There exists a point F on AC, such that PF is perpendicular to AC. Hence F is the intersection point of AC and the plane which contains P and is perpendicular to AC. The plane's equation is,

$$(x - x_1) + tan(\theta)(z - z_1) = 0 \tag{3.7}$$

Combined with the form of AC, we will have,

$$x_f = \frac{x_a tan^2(\theta) + z_1 tan(\theta) + x_1}{tan^2(\theta) + 1}$$

$$y_f = 0$$

$$z_f = z_1 - (x_f - x_1)/tan(\theta)$$

$$F = (x_f, y_f, z_f) \tag{3.8}$$

The unit direction vector of AC is

$$\vec{n}_{ac} = (C - A).Unit() \tag{3.9}$$

The unit direction vector of FP is

$$\vec{n}_{fp} = (P - F).Unit() \tag{3.10}$$

Then we can get the tangential unit direction vector of circle (around F, radius=$|FP|$),

$$\vec{n}_t = \vec{n}_{ac}.Cross(\vec{n}_{fp}) \tag{3.11}$$

The number of tracks we regenerate is determined by the weight number. Basically, we just do the loop for floor(weight) times. Within each loop, assume the point we generated

on the circle is G. The following steps will give us the way to obtain G coordinate in the normal frame (FM0).

First, generate a random angle ϕ uniformly between 0 and 2π. In the frame (FM1) where F is the origin and the circle plane is the y-z plane, we can have the coordinate of G without losing generality,

$$
\begin{aligned}
x_g &= 0 \\
y_g &= |FP|cos(\phi) \\
z_g &= |FP|sin(\phi) \\
G &= (x_g, y_g, z_g)
\end{aligned} \tag{3.12}
$$

Second, do the clockwise rotation around y-axis by θ to let the FM1 become FM2, such that the x, y and z axis are parallel to the x, y and z axis in FM0,

$$G.RotateY(-theta) \tag{3.13}$$

Third, move the origin of FM2 to the origin of FM0,

$$G = G + F \tag{3.14}$$

Now the G coordinate we get is the point the new track goes through. We still need to find the direction of the new track, which is the momentum direction $\vec{n}_g$. And to keep the same style as the original file, the coordinate of K will also be provided, which is the intersection point of the new track and the inclined plane.

The momentum values of the new track along normal($\vec{n}_{1g} = \vec{n}_{ac}$), radial($\vec{n}_{2g} = \vec{n}_{fg}$) and tangential($\vec{n}_{3g} = \vec{n}_{1g} \times \vec{n}_{2g}$) directions should be unchanged. These values could be ob-

tained by projecting the original momentum vector($\vec{n}_p$) into the corresponding three directions of the original track,

$$
\begin{aligned}
p_n &= \vec{n}_p \cdot \vec{n}_{1p} \\
p_r &= \vec{n}_p \cdot \vec{n}_{2p} \\
p_t &= \vec{n}_p \cdot \vec{n}_{3p}
\end{aligned}
\tag{3.15}
$$

Now could get the momentum vector of the new track,

$$
\begin{aligned}
&\vec{n}_g = p_n \vec{n}_{1g} + p_r \vec{n}_{2g} + p_t \vec{n}_{3g} \\
&\Rightarrow p_{xg} = \vec{n}_g(0), p_{yg} = \vec{n}_g(1), p_{zg} = \vec{n}_g(2)
\end{aligned}
\tag{3.16}
$$

The new track is along this direction $\vec{n}_g$ and passing through G. We could get the equation of the new track,

$$
\frac{x - x_g}{p_{xg}} = \frac{y - y_g}{p_{yg}} = \frac{z - z_g}{p_{zg}} \tag{3.17}
$$

Compared to the equation of the inclined plane, we will have the intersection point of the new track with the inclined plane, K.

Until now, we have developed the steps to obtain the new track's momentum vector and the corresponding intersection point K's coordinate. The algorithm tackling with unthinned files has been finished. And the codes "regenerate.cpp" can also be found in the Appendix A.2.

We have also done some tests to see if our algorithm is proceeded in the right way. The distribution of intersection points on the plane perpendicular to the shower center should be circularly symmetric. $Fig.\ 3.2$ is the test result, and we can see that the distribution here

fulfills the requirement.

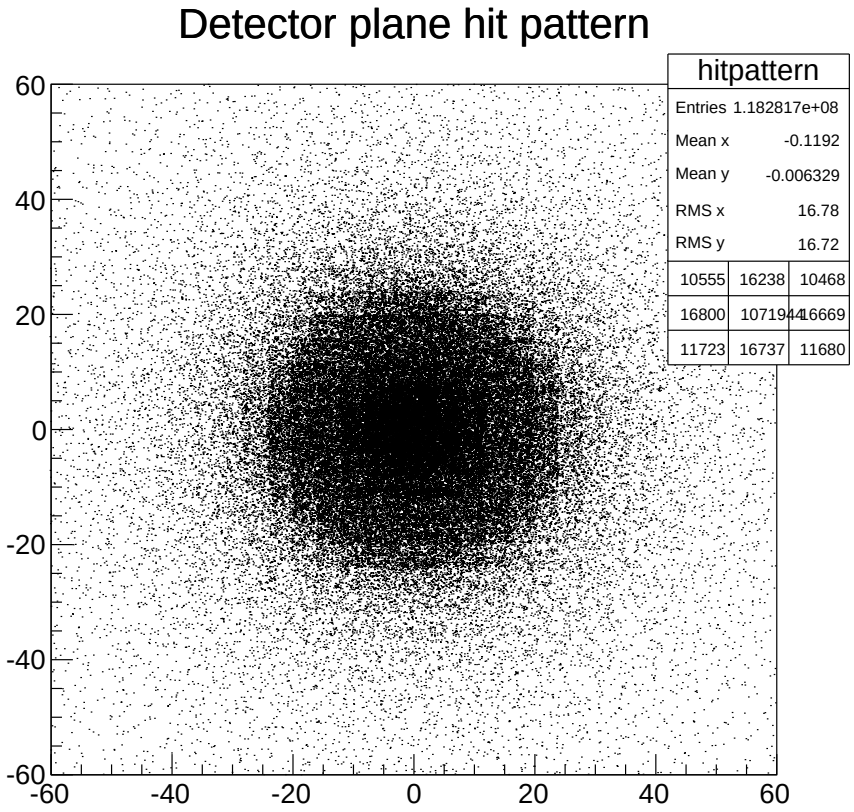


Figure 3.2: Distribution of intersection points on the normal plane

.

3.3 Detector array

3.3.1 Optimization considerations for TAUWER detector array

The neutrino flux in the ultra-high energy region is quite low and the probabilities of Earth-skimming neutrinos to generate τ leptons and escape into the air are also very small according to $Table$ 2.1. We would like to optimize the detector layout to ensure that no matter at what incident angle or position in the array area the shower axis has, the shower will

be detected. This limits the distance between two neighbor stations of the detector array. On the other hand, the array cost depends on the number of detector stations, so we cannot make the spacing too small. The ability to locate the shower core, defined by the electron cluster, independent of where the shower axis intersects the detector plane is also an important consideration to optimize our array. The following analyze of the simulation showers in this section will drive us the optimization conditions of our detector array. We use the simulation data under $\pi^-\pi^-\pi^+$ mode with 100 PeV energy to explain.

3.3.2 Hit patterns of showers on the transverse plane

To show the characteristic of shower development with respect to different shower development length L, we need to obtain the coordinates of hit points between tracks in the shower and the center detector plane, which is defined as the shower transverse plane passing through through C as shown in $Fig.\ 3.1$. The algorithm is shown as follows.

Firstly, read in a line from the unthinned data file, which provides us the particle type, momentum and intersection point's coordinates with the detector plane.

Secondly, according to the momentum vector and the intersection point coordinate, we can calculate the equation of this track. Combining to the equation of the normal plane, we can have the intersection point's coordinates of this track and the normal plane passing through point C.

The following part describes the coding details of this algorithm.

The information we obtained from the data file includes the momentum of a track (p_x, p_y, p_z), and the coordinates of the intersection points between this track and the inclined plane $(x1, y1, z1)$. The track's equation is,

$$\frac{x - x_1}{p_x} = \frac{y - y_1}{p_y} = \frac{z - z_1}{p_z}$$

The equation of center detector plane is

$$n_{ac}(0)(x - x_c) + n_{ac}(2)(z - z_c) = 0$$

where n_{ac} is the unit direction vector of AC while A is the τ decay point and C is the intersection point between shower center and the inclined plane. (x_c, y_c, z_c) are the coordinates of C.

Now we can obtain the coordinates of intersection point between tracks and the center detector plane in the standard frame.

$$hit_x = (p_z * x1/p_x + n_{ac}(0) * x_c/n_{ac}(2) + z_c - z1)/(p_z/p_x + n_{ac}(0)/n_{ac}(2))$$
$$hit_y = p_y * (hit_x - x1)/p_x + y1$$
$$hit_z = p_z * (hit_x - x1)/p_x + z1$$

Based on the coordinates of hit points on the transverse plane, we can derive the position distribution of particles on the transverse plane perpendicular to the shower center, as shown in $Fig.\ 3.3$ and $Fig.\ 3.4$. From these plots we can see, muons spread much broader than electrons on the transverse plane. Electrons are produced by EM processes, in which the characteristic transverse momentum transfer per interaction is of the order of 1 MeV (electron mass). Therefore, the EM processes remain concentrated near the shower axis, with some broadening when photons come from π^0 production deeper in the shower and not in the earliest few interactions. This feature of the high density electron core is independent of shower development length L and is the crucial feature for TAUWER trigger design. Muons, on the other hand, are produced only from decays of shower pions, where the characteristic transverse momentum is around 140 MeV (pion mass). Therefore, the muon spectrum has a much wider angular spread than the electron spectrum because of the

different angular characteristics of the EM interaction and the strong interaction.

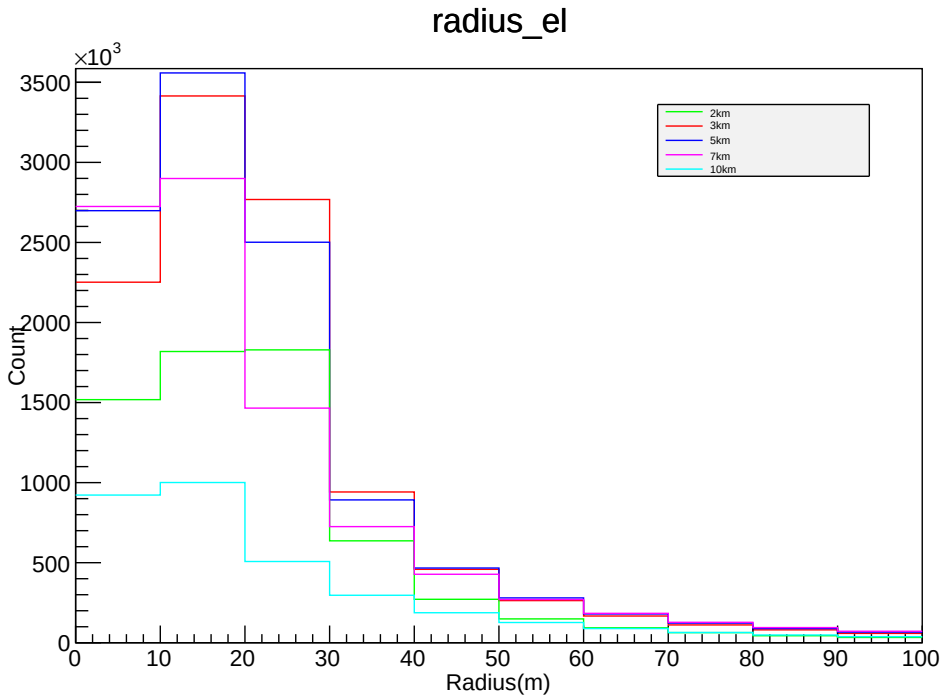


Figure 3.3: Position distribution of electrons on transverse plane per shower for pipipi mode at 100 PeV

3.3.3 Geometry of TAUWER detector array

As the electron radius plot ($Fig.$ 3.3) shows, independent of energy and shower development length, the electrons cluster within a circle of radius 40 meter about the centroid. Therefore, this detector array with less than 40 meter spacing will always have at least one detector that sees the high-density electron core, which makes the acceptance of candidate showers close to 1. The spacing more than 40 meter will make our detector array lose this characteristic. Meanwhile, as discussed in the beginning of this section, if we decrease the grid distance, the cost will go up if we want the detector array to cover the same area. Based on the trade-off between the acceptance of the array and the cost, we determine the grid spacing to be 40 meter. The funding of 1.5-2 million dollars can afford 1600 detector

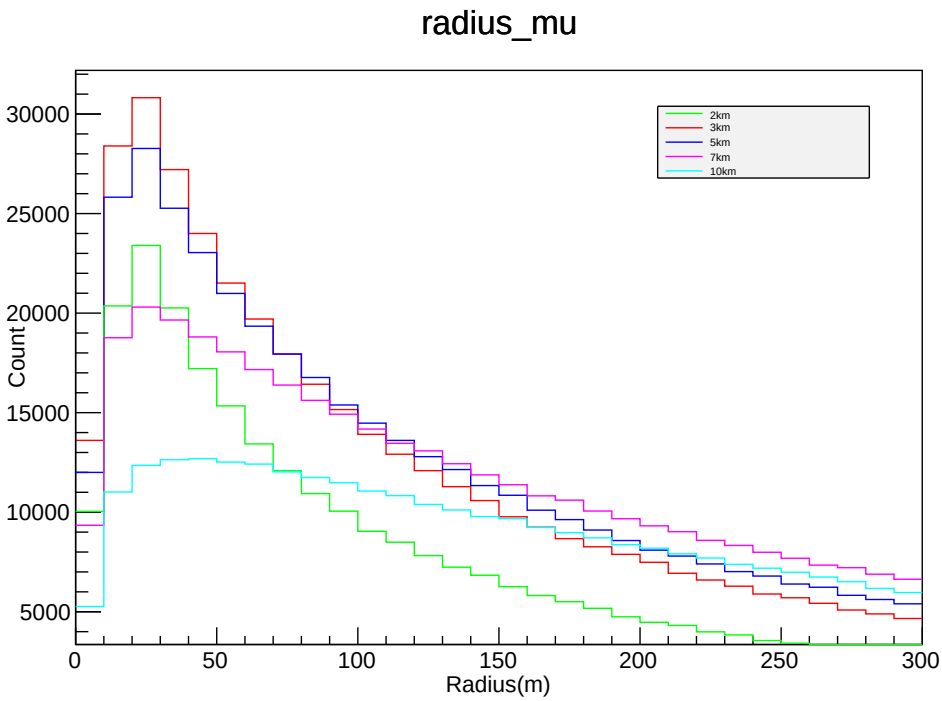


Figure 3.4: Position distribution of muons on transverse plane per shower for pipipi mode at 100 PeV

stations, which could cover 2.5 km^2 area. We need to make sure the array have at least about 600 meter along the vertical direction to cover the electron cores from showers with vertical angle from 91° to 98°. Therefore we make the detector array along each vertical line have 16 stations and each horizontal line have 100 stations. And our detector array can be easily expanded if we can have more funding in the future. The detector stations are placed facing the direction perpendicular to the average direction of shower axis, which has 2.5 degree with the horizontal plane. Trigger of the array is based on the situation that upward-moving neutrino showers will have at least one detector with a large number of hits within a 20 ns time window due to the electron charge cluster near the shower axis. We require large energy deposit in two layers to eliminate cases of a single vertical air shower giving a track that deposits a large amount of energy in a single counter because it travels a long distance in the scintillator. The KIT test runs show that this happens at the μHz rate in EAS events.

3.3.4 Simulation for detector array

In this part, we will estimate the detection efficiency of the array and correlations among detector stations under variant conditions such as different position shifts and distances between τ decay point and the inclined plane, based on the unthinned data files we obtained from last section. Although we have 1600 detector stations covering 2.5 km^2, the simulation is done within an area of 1 km^2 which consists an 16×40 array, since few tracks can hit the region outside of this area around the shower centroid.

We will calculate the coordinates of hit points between tracks and the detector planes for two layers per station. If either of the layers of each station is hit by a track, we will consider this track as a detected track. Otherwise, the track is missed by our detectors.

We will repeat the similar process for 640 times related to all detector elements for each

track since we only analyze the 16×40 array. After scanning all tracks from the unthinned data file, we will have the results such as the number of tracks detected by each detector element. According to these numbers, we can calculate the correlation between detectors or some other interesting information we need.

The following part describes the details about judging a track is detected or not.

As we have already explained, the track's equation is,

$$\frac{x - x_1}{p_x} = \frac{y - y_1}{p_y} = \frac{z - z_1}{p_z}$$

The equation of detector plane is

$$n_{ac}(0)(x - det_x) + n_{ac}(2)(z - det_z) = 0$$

where $det = (det_x, det_y, det_z)$ are the coordinates detector plane going through and can be calculated as follows. There are two layers of detectors, hence we will have $16 * 40 * 2 = 1280$ det coordinates. The side geometry of one detector station is shown in $Fig.\ 3.5$.

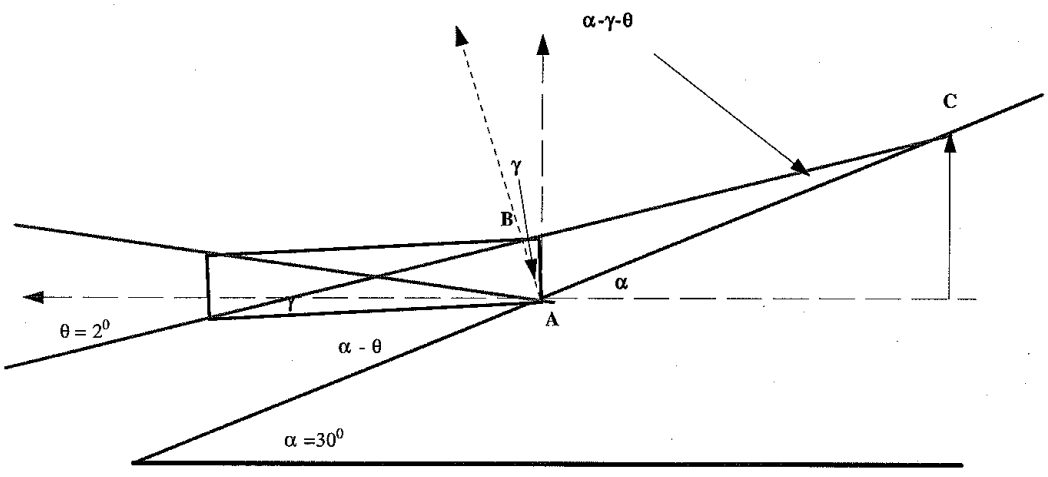


Figure 3.5: Shower center and inclined plane

.

Here we assume the distance between two detector stations is $distance = 40m$ along x and z axis in the standard frame. Points(det) of the first layer are on the inclined plane, evenly distributed along x and z axis by the distance of 40 meters and centered around the mean position of C. The coordinates of "mean" C is calculated under $\theta = 2.5°$.

$$det1_z[i * 40 + j] = z_c + distance/2 - 8 * distance + i * distance$$
$$det1_x[i * 40 + j] = (det1_z[i * 40 + j] - z_o)/tan(\alpha) + x_o$$
$$det1_y[i * 40 + j] = distance * j - 20 * distance + distance/2$$

The coordinates of second layer's det can be derived from the first layer's det,

$$det2_x[i * 40 + j] = det1_x[i * 40 + j] - d * cos(\theta)$$
$$det2_y[i * 40 + j] = det1_y[i * 40 + j]$$
$$det2_z[i * 40 + j] = det1_z[i * 40 + j] - d * sin(\theta)$$

where $d = 0.2/tan(\gamma) = 1.6m$ is the perpendicular distance between two detector layers, and $\gamma = 7 * Pi/180$, which is shown in $Fig.\ 3.5$.

Now we can obtain the coordinates of intersection points(hit1 and hit2) between tracks and detector elements(including two layers) in the standard frame.

$$hit_x = (p_z * x1/p_x + n_{ac}(0) * det_x/n_{ac}(2) + det_z - z1)/(p_z/p_x + n_{ac}(0)/n_{ac}(2))$$
$$hit_y = p_y * (hit_x - x1)/p_x + y1$$
$$hit_z = p_z * (hit_x - x1)/p_x + z1$$

We need to judge if the intersection points are located within the detector region. But it is difficult to do that in the standard frame. We would like to transfer the coordinates into

frame where det point is the origin, x-axis is along the opposite direction of standard y-axis and y-axis is parallel to the intersection line between detector plane and standard x-z plane.

$hit = hit - det$

$hit.RotateY(\theta)$

$newhit_x = hit_y, \quad newhit_y = hit_z$

Now if $newhit_x \in (-0.2, 0.2)$ and $newhit_y \in (0, 0.2)$ for hit1 or hit2, we claim this track is detected.

3.3.5 Information derived from the simulations

The information we can obtain from the detector array are hitnumbers from each detector station, and the corresponding time stamps of shower tracks. The data we use from Monte Carlo simulation are able to provide us precisely how many particles each detector station captures. But in the real experiment, we cut the hitnumbers for each station at four, since our scintillator detectors may saturate after four hits. All the counts derived from the simulation greater than four will be considered as four to mimic the real experiment better. The showers we plan to detect are electron-rich, and the majority of the electrons will travel in the directions very close to the shower center. Muons come from π-meson decays and have a much broader angular spread and lower density. Many fewer muons will hit TAUWER stations. The stations around the shower center will have much greater hit numbers compared to the other stations if we use the simulation data directly, even if we limit effective hit numbers to let them be less or equal than four. After checking many hit patterns, we find that under most scenarios, only one muon can hit a given station. The station with maximum hit number is considered to be the station closest to the shower cen-

ter, and the region containing this station and the other eight neighbor stations around it is described as electron core region. Based on simulation results of analysis of shower development, most of the electron hits will be located within this region, and all the hits outside will be considered as muon hits.

The relation between the detection efficiency and the distances between the τ decay point and inclined plane will also be examined to find out the detecting range of our array. $Table$ 3.2 provides us the relationship between particles being detected per shower and the distance. When we make the table, the outliers, defined as the 8 smallest and 8 largest counts from the 100, are removed to obtain a better estimate of the deviation from average behavior. This policy will be applied to all the tables we make. But we also attach the information of statistics without cuts in the plots corresponding to each table, to illuminate the fluctuation.

From $Table$ 3.2, we can see that the number of particles detected increases as the shower development length L increases from 2.5 km to 5 km (for electrons) or 7.5 km (for muons) and then decreases after that. When the distance is greater than 10 km, the number of detected particles shrinks to a very small number, as shown in the 15 km case. This phenomenon matches our expectation. Shower maximum occurs around 6 interaction lengths in any material for the energies we are interested in. For electrons in air, that distance is about 5 km and that is the reason why the 5 km case gives us largest number of detected electrons. For muons in air, since they do not have a radiative component to their energy loss, the peak will show up a little further along. Meanwhile, all tracks undergo ionization energy loss of about 1.2 GeV/km in air. For both electrons and muons, after 10 km the average track is too soft to generate more tracks, so the shower attenuates by ionization loss and dies out. This means that at 15 km there are few electrons and muons detected for a 100 PeV event. For distances smaller than 2.5 km, we see that the variance is so big that no effective information can be extracted to do further analysis. Hence from now on, we

will focus on the shower development distances between 2.5 km and 10 km.

Table 3.2: Number of particles detected per shower vs. Distance with energy 100 PeV

Distance(km)	Electron	Muon
2.5	366.4(346.7)	21.9(13.3)
3	648.7(458.7)	36.2(12.8)
5	696.7(349.1)	36.1(13.1)
7.5	486.6(247.4)	34.2(9.9)
10	186.8(127.5)	25.0(7.7)
15	0.11(0.31)	1.0(0.9)

$Fig.$ 3.6 to $Fig.$ 3.7 show us the details of hit detector numbers for each shower development length L, which are corresponding to $Table$ 3.3.

Table 3.3: Hit detector number per shower vs. Distance

Distance(km)	2.5	3	5	7.5	10
Electron	29.0(16.4)	43.5(15.2)	49.4(11.6)	51.0(12.4)	33.7(12.9)
Muon	17.0(10.0)	27.8(9.2)	28.7(9.6)	29.6(7.6)	23.2(6.8)

Even though photons will not contribute to coincidence counting, independent photons in the high density EM region near the shower core may well interact in the front and back scintillators within the 20ns trigger time window. This will increase the number of counts for the stations at the core of the shower. If the number of hits in these stations is already above the saturation limit, there will be no impact from the added photon hits. For showers with fluctuations toward low numbers of electrons, the photon hits will help the overall trigger efficiency somewhat. The effect is not large and we choose to ignore it in the following calculations.

Figure 3.6: Number of detectors hit by electrons. We add 0 for 2km case, 100 for 3.5km case, 200 for 5km case, 300 for 7.5km case and 400 for 10km case to put the five curves in one plot

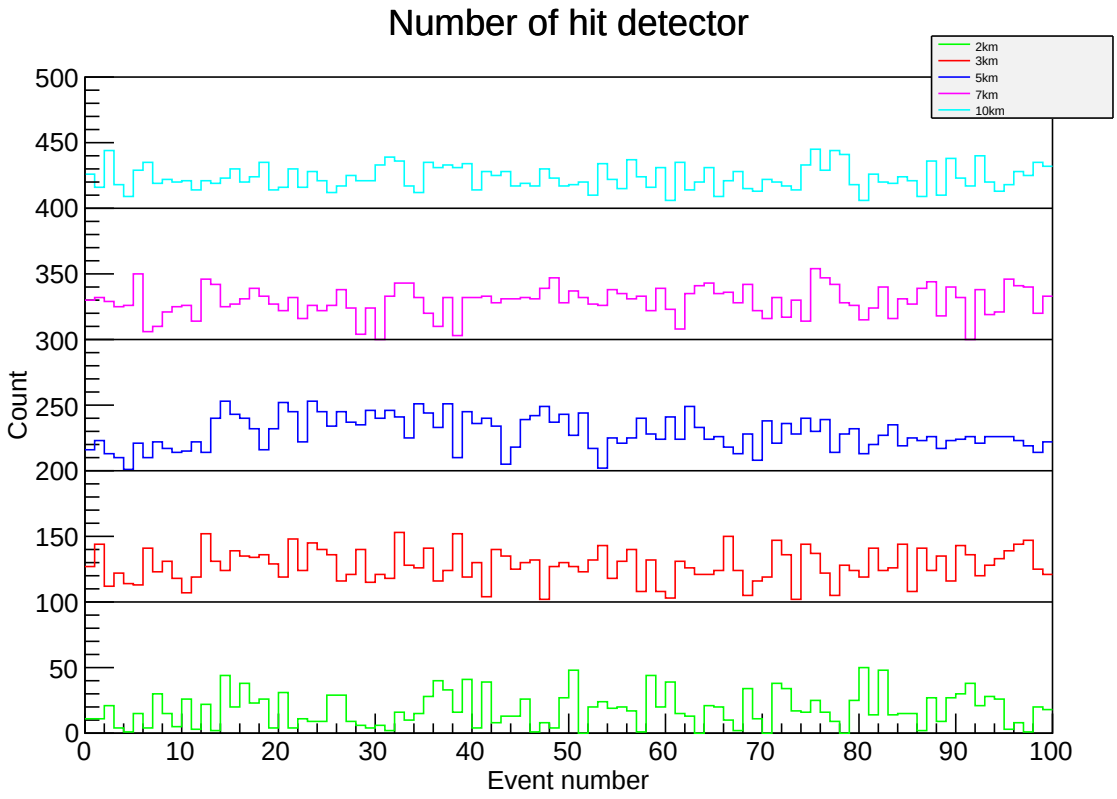


Figure 3.7: Number of detectors hit by Muons. We add 0 for 2km case, 100 for 3.5km case, 200 for 5km case, 300 for 7.5km case and 400 for 10km case to put the five curves in one plot

3.3.6 Solid angle of incident ν_τ

We need the solid angle to estimate event rates of the detector array. The solid angle is calculated as follows,

$$4\pi \times \frac{\theta_V}{180^\circ} \times \frac{\theta_H}{360^\circ} \tag{3.18}$$

θ_V is defined as the effective vertical angle coverage. $\theta_V = 98^\circ - 91^\circ = 7^\circ$ in our case, and we need to use the angle-average probability for a specific energy from $Table$ 2.1. We choose a conservative vertical angle range for the array. If a suitable site is located, we can use zenith angles less than 90 degrees and accept events that interact in the shield mountain indicated in $Fig.$ 1.5. θ_H is defined as the horizontal angle coverage. With a horizontal rotation test as shown in the next section, we can find a range of angle, that when the shower is horizontally rotated within this range the detection efficiency of the array will not be deteriorated.

3.3.7 Rotation Test

We want to find the detector efficiency for showers with different incident horizontal angles. The easiest way to do this is rotating the particle tracks included in the data file by different angles around the designated axis, instead of rotating the whole detector array. Since the algorithm of detecting a track has already been introduced in the previous section, only the rotation part will be described here.

We rotate tracks along the "rotateaxis" by five different angles, from 0 degree to 20 degree with step distance 5 degree. And the "rotateaxis" is on the x-z plane, perpendicular to the shower axis AC, and passing through point C, which is the intersection point of the shower axis and the inclined plane. This algorithm can be achieved by the following codes, and

$Fig.$ 3.8 and $Fig.$ 3.9 show us the test results.

$rotateaxis = \vec{n}_{AC}$

$rotateaxis.RotateY(Pi/2);$

$\vec{x} = \vec{x} - \vec{c}$

$\vec{x}.Rotate(rotateangle, rotateaxis)$

$\vec{x} = \vec{x} + \vec{c}$

Here $\vec{x}$ representing the coordinates of the intersection points between tracks and the inclined plane, and $\vec{c}$ denotes the coordinates of point C. $\vec{n}_{AC}$ is the unit direction of the shower axis AC.

For electrons, the number of hit detectors will increase when we increase the horizontal rotation angle within the region between 0° and 70°. The number of hit detectors will keep increasing after 70° but this trend will stop at 90° due to physics cuts of the mountain geometry condition. For muons, the number of hit detectors are almost independent of the horizontal angle between 0° and 70°. But we can see the trend that this number will decrease after 70°.

The event rates of TAUWER experiment can be estimated. E^{-2} neutrino spectrum is used here according to IceCube experiment[1]. The following equation gives the flux of one neutrino flavor,

$$E^2 d\phi_{\nu_\tau}/dE = 1.2 \times 10^{-8} GeV cm^{-2} sr^{-1} s^{-1} \tag{3.19}$$

Based on the KIT test lasting for two years, the operating efficiency of our stations is above 90%, and hence we use 2×10^7 s as the time for an operating year.

According to the rotation test, we can at least have horizontal angle in the range from -

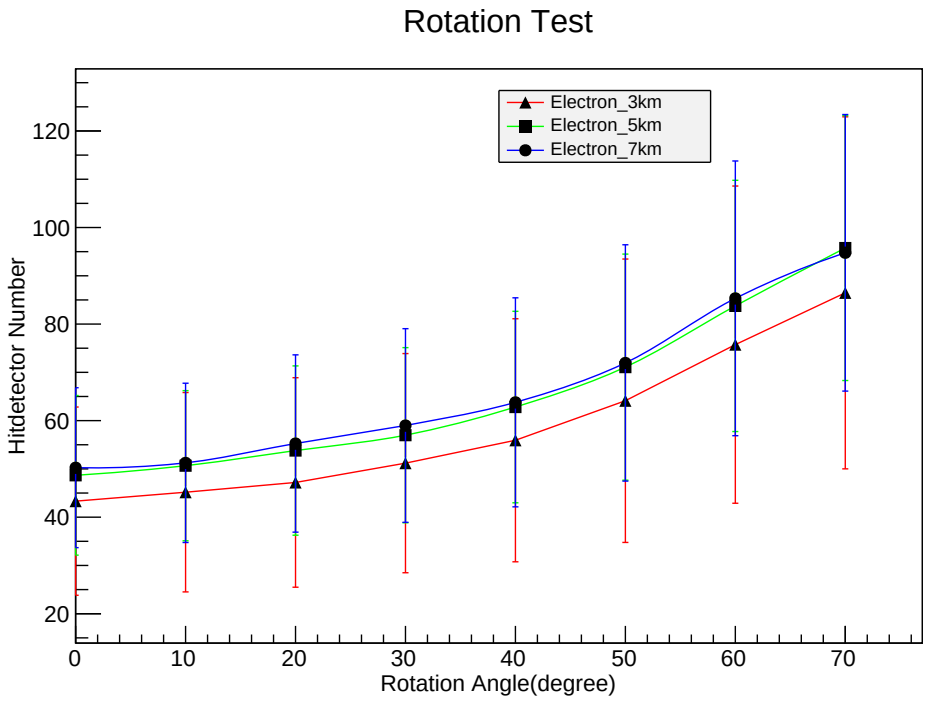


Figure 3.8: Number of hit detectors by electrons

.

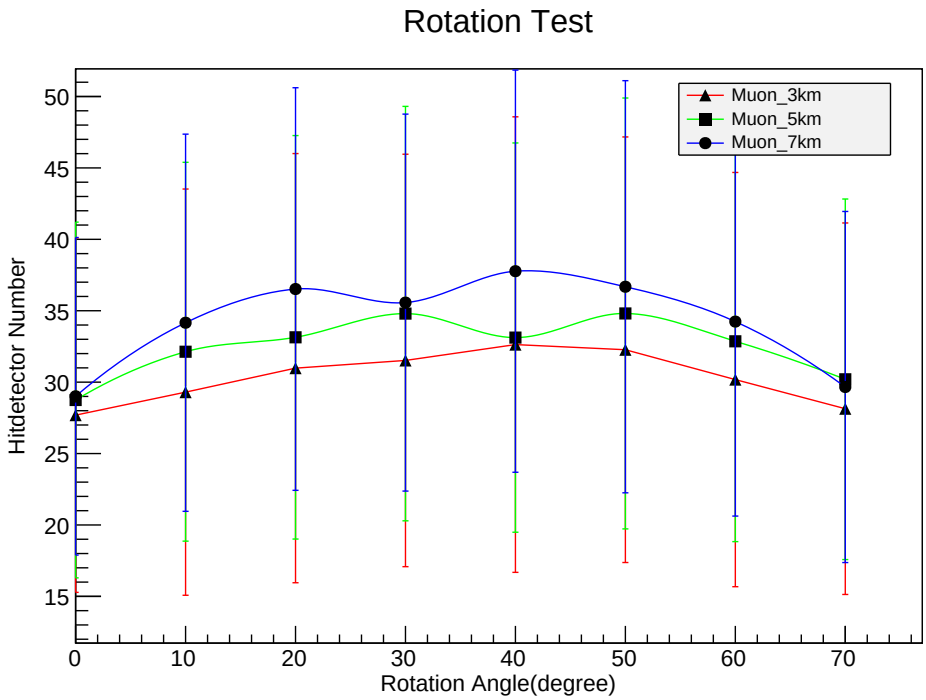


Figure 3.9: Number of hit detectors by muons

.

70° to +70° to keep the acceptance rate at 1. Meanwhile, we have calculated the escape probability of τ with respect to vertical angles between 91° to 98°. Hence the solid angle is $4\pi \times (7/180) \times (140/360) = 0.19$ sr.

We can take the mean escape probability with respect to a specific energy from 10 PeV to 200 PeV to calculate the event rates, where the mean is taken by averaging the results from 91° to 98°. The mean escape probability for each neutrino energy is shown in $Table.\ 3.4$, which can be derived from $Table.\ 2.1$ in Chapter 2.

Table 3.4: Mean escape probability with respect to ν_τ energy in units of 10^{-4}

Energy (PeV)	Probability	Energy (PeV)	Probability
10	1	110	12
20	2	120	13
30	3	130	14
40	5	140	15
50	6	150	16
60	7	160	16
70	8	170	17
80	9	180	18
90	10	190	19
100	11	200	19

And the event rates can be estimated by summation of rates from each energy bin, which can be represented as follows,

$$\sum \phi_{\nu_\tau} \times \Delta_E \times (0.19\text{sr}) \times (2 \times 10^7\text{s}) \times (2.5\text{km}^2) \times (\text{mean escape probability}) \qquad (3.20)$$

where $\Delta_E = 10$ PeV is the energy bin width. The event rates we estimate of our detector array is 0.434 eV/yr within the energy region between 10 PeV and 200 PeV.

The bounds on the flight distance L require that the τ decay point has to be at least 2.5 km away from the detector array and less than 12.5 km. This lowers the expected rate by about a factor of 2, but the exact value depends on the final site geometry. An estimate of

0.2 eV/yr is a good working number, using the IceCube flux estimate. If there are other sources of UHE neutrinos, e.g., AGN sources whose emission peaks in the 50 PeV range, then the TAUWER rate can be substantially higher.

Chapter 4

Event Classification

One of the main purposes of the TAUWER experiment is to measure the energy of each detected high energy Earth-skimming ν_τ. Several pieces of information can be provided by our detector array. Showers from neutrinos with different energies should bring us different hit patterns, which are reflected by hitnumbers of our detector stations. At the same time, time stamps of hit tracks in each shower will also be recorded.

First we try to derive the energy of a shower based on its hit pattern. However, there is no available theory to achieve that. We tried to set up a relation between the mean hitnumber of each shower and its energy by polynomial regression, but the large variance of mean hitnumber for showers with same energy did not allow this approach to succeed. The main difficulty is the wide range of L that is possible for any event. We cannot constrain L just from the hit pattern. Hence, the job of direct derivation of shower energy needs to be modified, which will be introduced in detail in this section.

In this thesis the term *classification* means the assignment of an energy to a given τ lepton shower candidate based on a statistical comparison of the characteristics of the data event to those of a selected set of characteristics of simulated events at a set of energy calibration points. The energy is assigned to the calibration point that gives the best description of the

data characteristics based on a set of classification algorithms to be described here. The statistical nature of classification means that the energy of a given event is not determined precisely but rather is discretized by the calibration energy set. The ultimate resolution of the energy assignment is limited by the fluctuations and will be described by the relative probabilities of being assigned to different calibration points in the set for a data event of true energy E_τ.

4.1 Data characteristics

The data from simulation is summed over L and then divided into two parts, one for the electron core region and the other for the muon region, which is discussed in last chapter. As we have already explained, approximate methods need to be introduced to derive the energy. We will discretize the energies, and after that, each sub-range can be considered as a class. Based on simulation there are systematic differences on hit patterns from different classes. For example, showers with energy 100 PeV should have more hits than showers with energy 10 PeV. If the two energies are considered as two classes, classification can be done based on the data features extracted from their hit patterns. One can sub-divide the classification range into smaller energy intervals. In this chapter we will also use a multi-class method, using three sampling points, in order to illustrate the procedure.

4.2 Classification methods

There are many classification methods available, such as naive Bayes or decision tree. Each of them will have their own assumptions and applicable conditions. Two famous methods will be introduced here, logistic regression and support vector machine (SVM). In this section, we will briefly introduce the two methods and make comparisons between them to

choose the best classification method for our problem. Details of the methods can be found in [25][14][35].

4.2.1 Notation

In this section, we use $\boldsymbol{x}$ to represent an event and y to represent the label. Label 0 and 1 is used for a two-class problem and label 1, 2 and 3 is used for a three-class problem. Assume we have N events and each event has K features, then we can express i^{th} event as $\boldsymbol{x}_i^T = (x_i^{(1)}, x_i^{(2)}, ..., x_i^{(K)})$, where $i = 1, 2, ..., N$. For a two-class problem, we have two labels for y from $\{0, 1\}$. For a multi-class problem (assume we have three classes), then y can be selected from $\{1, 2, 3\}$.

4.2.2 Logistic regression

Logistic regression is a discriminative classifier by using log loss, which is widely applied in both academia and industry. We first consider classification problems with two classes. There is one assumption made in two-class logistic regression about the probability of a event to be classified into class 1 under the observation of its feature $\boldsymbol{x}$, which makes it a discriminative classification method,

$$P(y = 1|\boldsymbol{x}) = \frac{1}{1 + e^{-\theta^T \boldsymbol{x}}} \equiv \mu(\boldsymbol{x}) \tag{4.1}$$

where $\theta^T = (\theta_1, \theta_2, ..., \theta_K)$ describes the weights for each feature who will be derived after classification and $\theta^T \boldsymbol{x} = \theta_1 x_1 + \theta_2 x_2 + ... + \theta_K x_k$. The ratio between $\mu(\boldsymbol{x})$ and not $\mu(\boldsymbol{x})$ is called "odds" if $0 < \mu(\boldsymbol{x}) < 1$ for a binary event. Since the term *log odds* means $log(\mu(\boldsymbol{x})/(1 - \mu(\boldsymbol{x}))) = \theta^T \boldsymbol{x}$ according to $Eq.\ 4.1$, our assumption is equivalent to the assumption that the log odds between the classes of logistic regression is linear.

We immediately have,

$$P(y|\boldsymbol{x}) = \mu(\boldsymbol{x})^y(1-\mu(\boldsymbol{x}))^{1-y} \tag{4.2}$$

With this assumption, we can have the likelihood function,

$$\begin{aligned} l(\theta) &= \sum_{i=1}^{n} lnP(y_i|\boldsymbol{x}_i;\theta) \\ &= \sum_{i} y_i ln\mu(\boldsymbol{x}_i) + (1-y_i)ln(1-\mu(\boldsymbol{x}_i)) \end{aligned} \tag{4.3}$$

And then the parameter vector θ can be estimated by maximize this log conditional likelihood,

$$\hat{\theta} = \arg\max_{\theta} l(\theta)^1 \tag{4.4}$$

Many methods can be used to derive the value of $\hat{\theta}$, such as gradient ascent, Newton-Raphson method, and iterative reweighted least squares (IRLS). Details of these methods can be found in [26].

We can use the value of $\hat{\theta}$ to predict the class of new events for two-class problem,

$$\hat{y} = \begin{cases} 1, & \hat{p}(y=1|\boldsymbol{x}) > 0.5 \\ 0, & otherwise \end{cases} \tag{4.5}$$

[1] arg max gives the parameter value where the following function can achieve its maximum

which can be simplified as,

$$\hat{y} = \begin{cases} 1, & \hat{\theta}^T \boldsymbol{x} > 0 \\ 0, & otherwise \end{cases} \tag{4.6}$$

The decision boundary divides events into two labels: 0 and 1, which are the energy assignments in our case. Hence $\hat{\theta}^T \boldsymbol{x}$ can be considered as the decision boundary for logistic regression, which is linear due to the strong assumption we made. We may need to worry about this point since in the real world. Most of the cases will need non-linear decision boundaries, and the assumption of logistic regression therefore does not hold. However, the way we use logistic regression is to do classification. The probability for a given event to be assigned label 0 or 1 is determined by plugging the estimated $\hat{\theta}$ into $Eq.\ 4.1$. The energy label will still be correct if the probability of the right label is greater than the wrong label, even though the assumption about having a linear decision boundary does not hold. Because of that, logistic regression is still widely used even when we are not sure if the decision boundary is linear. In our case, we apply logistic regression as a baseline method and compare it to the SVM method. But it must be pointed out that if the decision boundary is nonlinear, systematic errors will exist in the classifier derived from logistic regression. And gross nonlinearity in the decision boundary could smear the classifier severely, which gives the limitation of the logistic regression method.

Logistic regression can be extended to deal with multi-class cases. If we have K labels 1,2,3,...K. We need to modify the assumption as follows,

$$P(y = i|\boldsymbol{x}) = \begin{cases} \frac{exp(\theta_i \boldsymbol{x})}{1+\sum_{k=1}^{K-1} exp(\theta_k \boldsymbol{x})}, & i = 1, 2, ...K-1 \\ \frac{1}{1+\sum_{k=1}^{K-1} exp(\theta_k \boldsymbol{x})}, & i = K \end{cases} \tag{4.7}$$

we can estimate the parameters $\{\theta_i\}_{i=1}^{K-1}$, pick up the label $\hat{y}$ as follows,

$$\hat{y} = \arg\max_i P(y = i|\boldsymbol{x}) \tag{4.8}$$

and do classification following the same procedures as two-class cases we just discussed.

4.2.3 Support Vector Machine (SVM)

SVM is a method with outstanding classification capability which is widely used in statistical and machine learning fields[14][38]. The key idea in SVM is to maximize the margin between different classes of samples, where the margin is defined as distances between closest samples to the decision boundary, as shown in $Fig.\ 4.1$. The margin can be represented as follows,

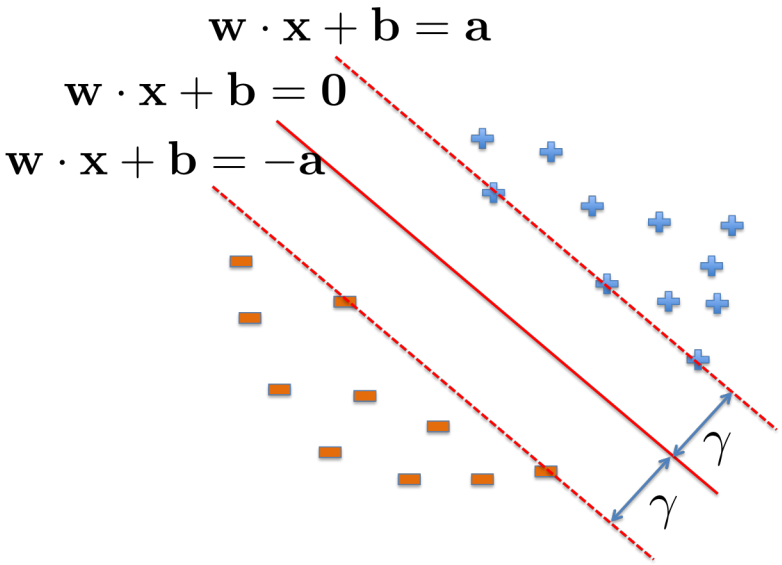


Figure 4.1: Decision boundary and margin of SVM

$$Margin = 2\gamma = 2a/||\mathbf{w}|| \tag{4.9}$$

Now we can have the objective function of SVM to maximize the margin,

$$\min_{\mathbf{w},b} \mathbf{w} \cdot \mathbf{w}$$

$$\text{such that } (\mathbf{w} \cdot \boldsymbol{x}_j) y_j \geq 1 \tag{4.10}$$

The above algorithm is called a hard margin SVM and will provide us a linear decision boundary. However, if the two-class data we have can not be separated by a linear decision boundary as shown in $Fig.\ 4.2$, the hard margin algorithm will fail since it never allow any mis-classification.

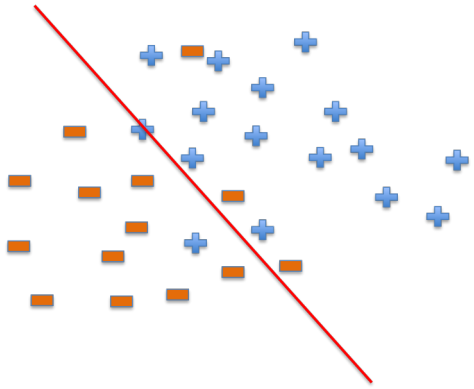


Figure 4.2: No available linear decision boundary

The way we fix that is by adding a loss function to the objective function. The loss function can be considered to be a penalty term. Instead of forbidding the mis-classification in the hard margin case, we want to limit the happening of mis-classification by adding penalty term for each mis-classified event. This is called a soft margin problem.

There are several available loss functions. We pick three of them, shown in $Fig.\ 4.3$. The mathematical form is as follows, where we have y as the true value and $\hat{y}$ as the predicted value (the continuous value from the classification).

1, zero-one loss: $I(y = \hat{y})$.

2, hinge loss: $max(0, 1 - y\hat{y})$.

3, log loss: $log(1 + exp(-y\hat{y}))$.

The most popular one is zero-one loss. This loss function is very straightforward. We assign 0 to the rightly classified events and assign 1 to the mis-classified events. However, this loss function gives mis-classified events with the same penalty without considering how far they are away from the decision boundary. Also the zero-one loss is more difficult to implement from the optimization angle compared to continuous loss functions. Hence, we introduce loss functions like hinge loss and log loss.

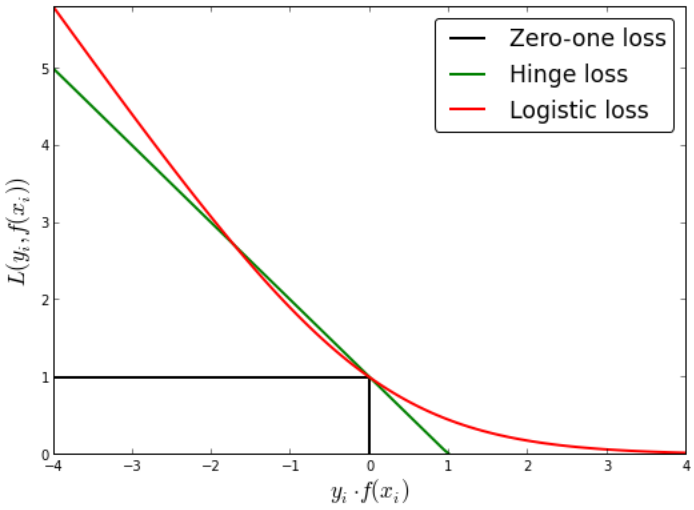


Figure 4.3: Three popular loss functions (made by Fabian Pedregosa, loss functions for ordinal regression, http://fa.bianp.net/blog/2013/loss-functions-for-ordinal-regression/)

We use the hinge loss for SVM, and the objective function can be modified as follows,

$$\min_{\mathbf{w},b} \mathbf{w} \cdot \mathbf{w} + C \sum_j \xi_j$$

$$\text{such that } (\mathbf{w} \cdot \boldsymbol{x}_j) y_j \geq 1 - \xi_j \; and \; \xi_j \geq 0 \; \forall j \tag{4.11}$$

This objective function actually demonstrates the difference between SVM and logistic regression. If we modify $Eq.4.11$ slightly as follows,

$$\min_{\mathbf{w},b} \mathbf{w} \cdot \mathbf{w} + C \sum_j \max\left(0, 1 - (\mathbf{w} \cdot \boldsymbol{x}_j) y_j\right) \tag{4.12}$$

which shows that hinge loss is used for SVM. Log loss is used for logistic regression shown as follows,

$$\min_{\mathbf{w},b} \mathbf{w} \cdot \mathbf{w} + C \sum_j log(1 + exp(-(\mathbf{w} \cdot \boldsymbol{x}_j) y_j)) \tag{4.13}$$

SVM can also be used to tackle multi-class cases. The objective function for K-class SVM with hinge loss is as follows,

$$\min_{\mathbf{w},b} \sum_{k=1}^{K} \mathbf{w}^{(k)} \cdot \mathbf{w}^{(k)} + C \sum_j \sum_{t \neq k} \xi_j^{(k)}$$

$$\text{such that } \mathbf{w}^{(k)} \cdot \boldsymbol{x}_j^{(k)} \geq \mathbf{w}^{(t)} \cdot \boldsymbol{x}_j^{(t)} + 1 - \xi_j^{(t)} \; and \; \xi_j^{(t)} \geq 0 \; \forall t \neq k, \forall j \tag{4.14}$$

$Eq.$ 4.11 is called as primal problem. By introducing Lagrange multipliers, $Eq.$ 4.11 is

equivalent to the following problem,

$$\min_{\mathbf{w},b} \max_{\alpha,\mu} \mathcal{L}(\mathbf{w}, b, \alpha, \mu)$$
$$where\ \mathcal{L} = \frac{1}{2}\mathbf{w} \cdot \mathbf{w} + C\sum_j \xi_j - \sum_j \alpha_j((\mathbf{w} \cdot x_j + b)y_j - 1 + \xi_j) - \sum_j \mu_j \xi_j \tag{4.15}$$

For SVM, Karush-Kuhn-Tucker conditions[11] hold and we can generate the dual problem which is equivalent to the prime one simply by changing the order of minimization and maximization in $Eq.\ 4.15$,

$$\max_{\alpha,\mu} \min_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b, \alpha, \mu) \tag{4.16}$$

α can be derived from the following equation which is an optimization problem named as quadratic programming (QP) and many available packages can be directly applied to solve this problem.

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j$$
$$\text{such that } \sum_i \alpha_i y_i = 0\ and\ 0 \le \alpha_i \le C \tag{4.17}$$

Then **w** and b can be derived as,

$$\mathbf{w} = \sum_i \alpha_i y_i \boldsymbol{x}_i$$
$$b = y_j - \mathbf{w} \cdot \boldsymbol{x}_j\ \forall j \text{ such that } 0 < \alpha_j \le C \tag{4.18}$$

Until now, the differences between logistic regression and SVM are just their loss functions. In the reality, most of the decision boundaries between classes are not linear. Although sometimes a linear decision boundary can give us acceptable classification accuracy when

true boundary is nonlinear, we still hope there would be improvement by generating a nonlinear decision boundary. This can be easily implemented for SVM by using kernel tricks. A slight modification to $Eq.\ 4.17$ and $Eq.\ 4.18$ as follows can bring us the nonlinear boundary we want,

$$\max_{\alpha} \sum_{i} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j)$$

$$\text{such that } \sum_{i} \alpha_i y_i = 0 \; and \; 0 \leq \alpha_i \leq C$$

$$\mathbf{w} = \sum_{i} \alpha_i y_i \phi(\boldsymbol{x}_i)$$

$$b = y_j - \mathbf{w} \cdot \phi(\boldsymbol{x}_j) \; \forall j \text{ such that } 0 < \alpha_j \leq C \tag{4.19}$$

where $\phi(\boldsymbol{x})$ is the function to map the data point $\boldsymbol{x}$ into high dimension feature space. But we never need its explicit form, since the dot product of two such points, $\phi(\boldsymbol{x}_i)$ and $\phi(\boldsymbol{x}_j)$, can be computed fastly using the kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

There are several kernel functions available. We will use three of the most popular ones. They are,

1, Radial kernel: $\exp(-\gamma|\boldsymbol{x}_i - \boldsymbol{x}_j|^2)$

2, Polynomial kernel: $(\gamma \boldsymbol{x}_i \cdot \boldsymbol{x}_j + a)^b$

3, Sigmoid kernel: $tanh(\gamma \boldsymbol{x}_i \cdot \boldsymbol{x}_j + a)$

4.3 Energy Identification

For energy identification, the time stamp information of tracks is not very helpful. We will only use the hitnumbers from 640 detector stations to do the classification. As we have discussed in section "Data characteristics", our scintillator detectors are not very sensitive

to detect hitnumbers greater than four, but are sensitive enough to tell the hitnumbers under four. Hence we will divide these hitnumbers into three kinds: no hit, hitnumbers between one and three, and hitnumbers larger than three.

4.3.1 Classification of events with 20 PeV and 100 PeV

The sample we use in this section consists of 376 events with 20 PeV and 500 events with 100 PeV. These events are grouped from 2km, 3km, 5km, 7km and 10km cases without any rotation.

Feature extraction

For each event's hit pattern, eight features can be extracted, which are listed as follows,

- Number of detectors with one hit in the electron region
- Number of detectors with two hits in the electron region
- Number of detectors with three hits in the electron region
- Number of detectors with hitnumber>3 in the electron region
- Number of detectors with one hit outside the electron region
- Number of detectors with two hits outside the electron region
- Number of detectors with three hits outside the electron region
- Number of detectors with hitnumber>3 outside the electron region

All the eight features are independent. We have tried to extract more features related to the geometry characteristics of the hit pattern, such as the moments. But these features do not bring us any benefits in the classification. We hope more useful features can be extracted in the future.

Data normalization

We perform data normalization before we apply any training algorithms on them. Otherwise the different scales of data in the feature space will bias the training methods.

We calculate the mean and standard deviation for each feature column, and then the data normalization can be done as follows,

$$x_i = (x_i - mean_k)/std_k,\ i = 1, 2, ...N,\ k = 1, 2, ...K \tag{4.20}$$

where N is the number of events, and K is the total number of features in each event.

And now the data on each feature column have mean at zero and standard deviation at one.

Feature selection

Lasso[39] is a widely used method to do feature selection. And we will use lasso together with logistic regression to do feature selection for our data. We use cross-validation to implement this algorithm as follows,

1, The labeled samples are randomly divided into 10 segments with approximately equal size.

2, For the i^{th} loop (i = 1,2,...10) we use the i^{th} segment as the test set, and the other nine segments as the training set. We then apply logistic regression with lasso to do the feature selection and record the selected features.

3, Features which are selected at least one time will be kept for future use.

$Fig.\ 4.4$ describes the number of times being selected for each feature, which shows all eight features we extracted from the data are selected.

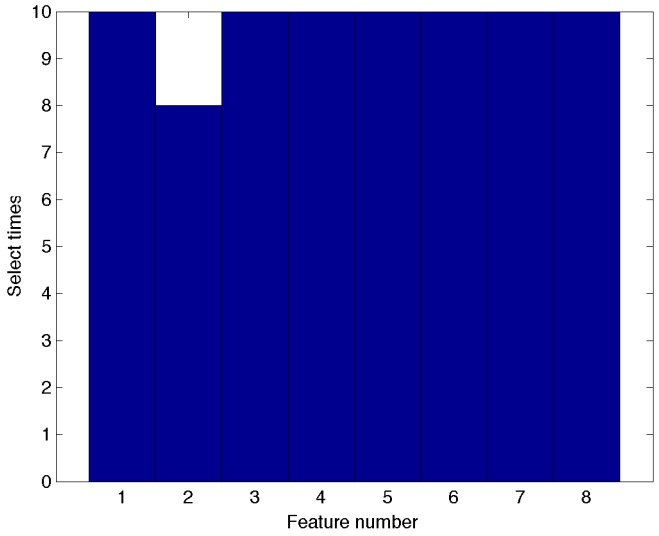


Figure 4.4: Feature selection by logistic regression with lasso

Selection of classification methods

One common comparison metric is classification accuracy. For each method, we will apply the same steps to calculate the accuracies as follows,

1, Divide the labeled data into 10 segments.

2, For the $i^{th}(i = 1, 2, ...10)$ loop, choose the i^{th} segment as the test set, and the remaining nine segments as the training set. Use different methods to do classification.

3, Record the number of mis-classified events for each iteration as $mc_i, i = 1, 2, ...10$, and assume the number of all labeled events is tc, then the accuracy of this classification method is 1-$\sum_i mc_i/tc$.

However, we would like to know more information from the classification besides the accuracy according to our objective, and hence cannot compare these classification methods only based on the accuracies. More details need to be reviewed.

Several corresponding concepts will be considered in the comparison. They are,

True positive: Positive event which is classified as positive.

False positive: Negative event which is classified as positive.

False negative: Positive event who is classified as negative.

True negative: Negative event who is classified as negative.

False negative is the counterpart of true positive and true negative is the counterpart of false positive. We can derive all of the above numbers if we know true positive and false positive, as well as the accuracy, which can be easily calculated if we know the event numbers.

Receiver operating characteristic (ROC) curves can be used to provide this information all together. The ROC curve describes the relationship between the true positive rate and the false positive rate at various discrimination thresholds for the classification method. The accuracy of the classification method only shows one cutpoint on the ROC curve. A classifier with higher accuracy does not necessarily mean better performance, since the high accuracy may come because of the discrimination threshold we choose. With a different discrimination threshold, the result may change dramatically. Hence a more comprehensive standard to compare these classification methods needs to be provided. We can use the area under the curve of ROC (AUC) as the new standard. Methods with largest AUC will be considered as the winner. To achieve this goal, we only need to modify the previous algorithm slightly as follows,

1, Divide the labeled data into 10 segments.

2, In the i^{th} loop, we choose the i^{th} segment as the test datasets, and the remaining nine segments will be the training datasets. In the training, we use different discrimination thresholds. In the previous algorithm, when the probability of being positive for one event is greater than 0.5, we classify this event as positive sample. However, to make the ROC curve, we can choose a group of threshold values from 0 to 1. For each threshold we choose, the number of misclassified negative events will be recorded as false positives fp_i, and the number of positive events who are classified correctly will be recorded as true positive as tp_i.

3, Calculate the true positive rate and false positive rate for each discrimination threshold and draw ROC curve with these points.

Classification by logistic regression

We use logistic regression with lasso as the baseline method to do the classification. The classification accuracy for two energy scales is shown in $Table$ 4.1

Table 4.1: Energy identification accuracy by logistic regression for two-class

Real \ Predicted	20 PeV	100 PeV
20 PeV	0.86(0.06)	0.14(0.06)
100 PeV	0.19(0.05)	0.81(0.05)

The classifier is trained by two energy scales, 20 PeV and 100 PeV. We would also like to see the classification ability of this classifier on the middle energy events, i.e., events with 50 PeV. And the result is shown in $Fig.$ 4.5, from which we can see that 58.0% of 50 PeV events are classified into 100 PeV and the leftover 42.0% are classified into 20 PeV.

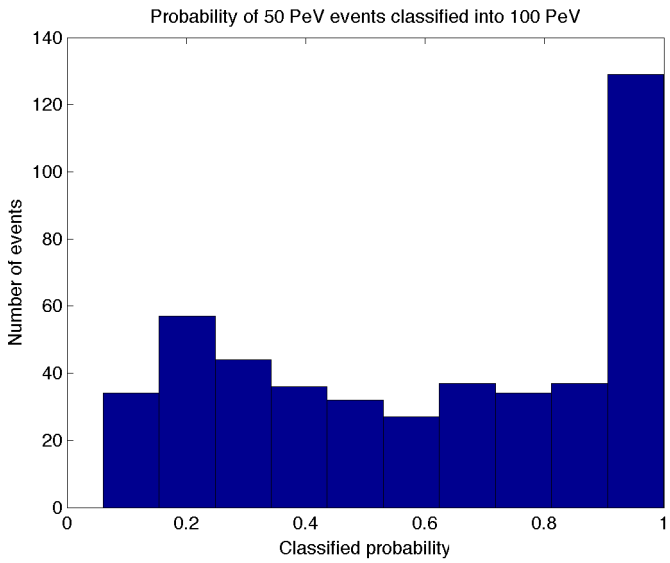


Figure 4.5: The probability of 50 PeV events classified into 100 PeV by logistic regression

Classification by SVM

As we have discussed before, SVM with three kernel functions will be used on our datasets and comparisons will be made among them to decide which kernel is the most appropriate choice for our case.

Again, we use cross-validation to select the coefficients in SVM objective function. Take radial kernel case as an example, C and γ need to be selected. We scan C from 0.1, 1, 10, 100 and γ from 0.01, 0.1, 1, 10. The classification accuracy are listed in $Table.\ 4.2$ for each pair of C and γ. We can conclude that $C = 10$ and $\gamma = 0.1$ will be used for SVM with radial kernel.

Table 4.2: Use cross-validation to select coefficients for SVM with radial kernel

C $\setminus$ γ	0.01	0.1	1	10
0.1	0.830	0.836	0.832	0.718
1	0.830	0.843	0.818	0.731
10	0.840	0.849	0.788	0.735
100	0.843	0.837	0.751	0.722

Similarly, we can choose the coefficients for polynomial kernel and sigmoid kernel. For polynomial kernel, we have $C = 10$, $\gamma = 1$, $a = 0$ and $b = 3$. For sigmoid kernel, we have $C = 100$, $\gamma = 0.01$ and $a = -0.3$.

A comparison between the three kernel function are made through ROC curves. The algorithm to obtain the ROC plots are shown as follows, which will be used to generate all the ROC curves in this section.

- Split all data points into 10 segments. For each run, use one segment as testing set and the other nine segments as training set
- For each run, we can obtain (fpr,tpr) pairs with fixed numbers of fpr, where tpr stands for true positive rate and fpr stands for false positive rate.

- After 10 runs, we will have 10 tprs for each fpr. Obtain the mean of these 10 tprs as mtpr, and standard error as stpr. Then the final ROC curves will be drawn by points (fpr, mtpr) and stpr will be used as error bars.

The ROC curves related to three kernel functions are shown in $Fig.\ 4.6$ and $Fig.\ 4.7$. From the plot we can conclude that kernel function is not sensitive to the classification in our case and we will pick up radial kernel as this kernel usually gives us best result.

One phenomenon can be seen from each ROC curve, that there will be larger error bars at the low false positive rate side, and this phenomenon holds for all the ROC curves we draw in this paper. On the very high false positive rate side, almost all events will be classified as positive no matter which segment we select as the testing set, and hence the error bars there are quite small. On the other side, we do not have that restriction. And 10 testing sets can give us very different true positive rates leading to larger error bars.

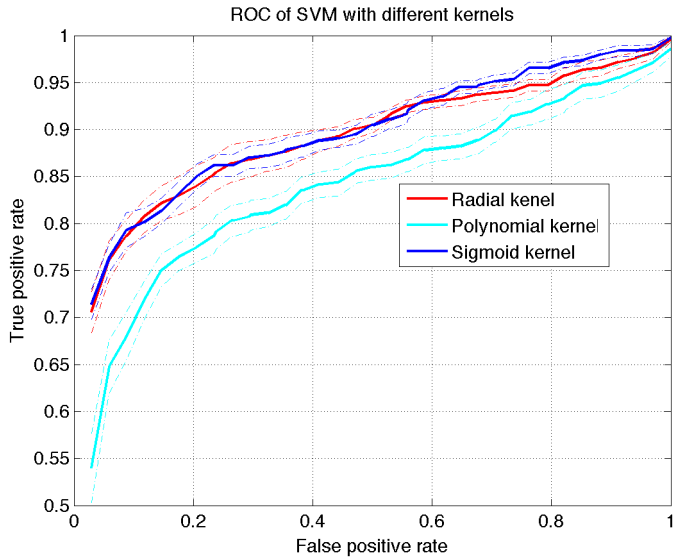


Figure 4.6: ROC curves of SVM with radial kernel, polynomial kernel and sigmoid kernel

The classification accuracy of SVM with radial kernel is listed in $Table\ 4.3$.

According to this table, we cannot see any advantage brought by SVM. But the accuracy is calculated when we set the decision threshold at 0.5, which means when the possibility

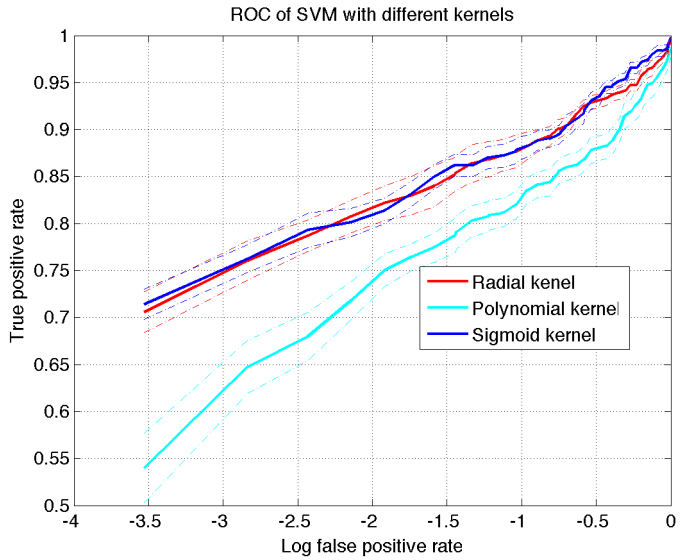


Figure 4.7: Semilog ROC curves of SVM with radial kernel, polynomial kernel and sigmoid kernel

Table 4.3: Energy identification accuracy by SVM for two-class

Real \ Predicted	20 PeV	100 PeV
20 PeV	0.88(0.04)	0.12(0.04)
100 PeV	0.22(0.06)	0.78(0.06)

of one event to be classified into 100 PeV is greater than 0.5, then we will classified this event with energy 100 PeV. However, this threshold can be modified due to our needs. For example, if we prefer larger true positive rate for 100 PeV, we can decrease the threshold to make more events classified into this energy scale. To make a better comparison, we still need to take a look at the ROC curves of logistic regression and SVM, which are shown in $Fig.\ 4.8$ and $Fig.\ 4.9$. However, still no statistically important difference can be seen from the two plots. Although generally SVM performs better than logistic regression, it is not the case for our data sets. In the following part of this chapter, we will provide classification results from both two methods.

As the logistic regression case, we will use the SVM classifier generated from 20 PeV and 100 PeV events to test 50 PeV events. And the probability of these events classified into

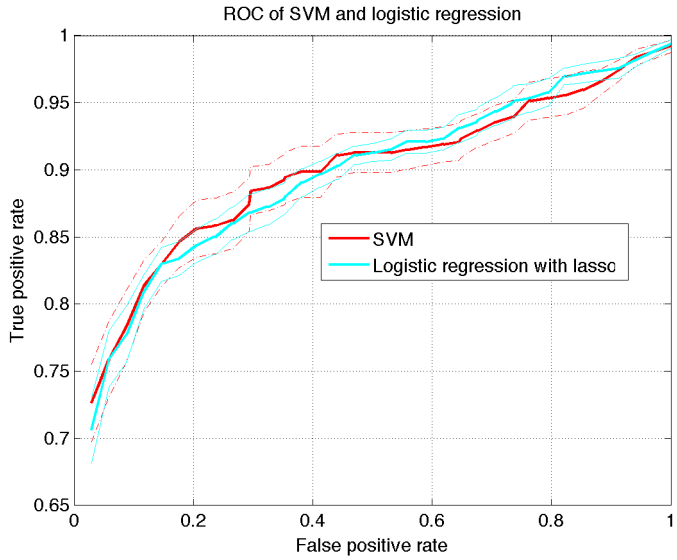


Figure 4.8: ROC curves of SVM and logistic regression with lasso

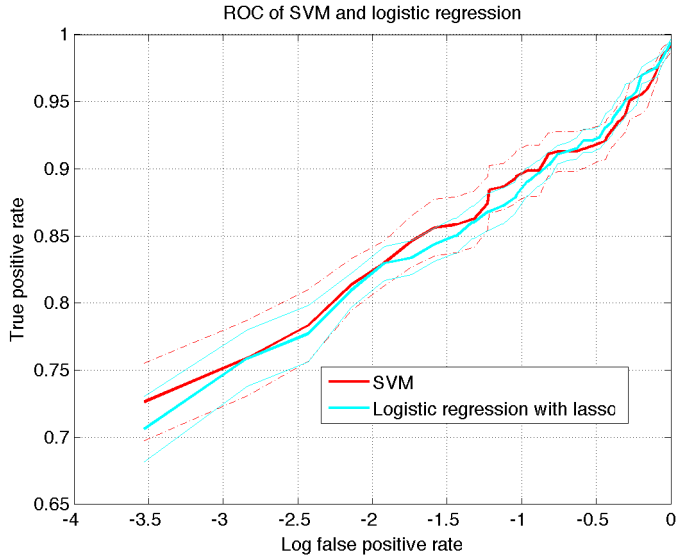


Figure 4.9: Semilog ROC curves of SVM and logistic regression with lasso

100 PeV is shown in $Fig.$ 4.10, from which we can see that 67.0% of 50 PeV events are classified into 100 PeV and the leftover 33.0% are classified into 20 PeV. From which we can see that SVM would like to classify more 50 PeV events into 100 PeV compared to logistic regression.

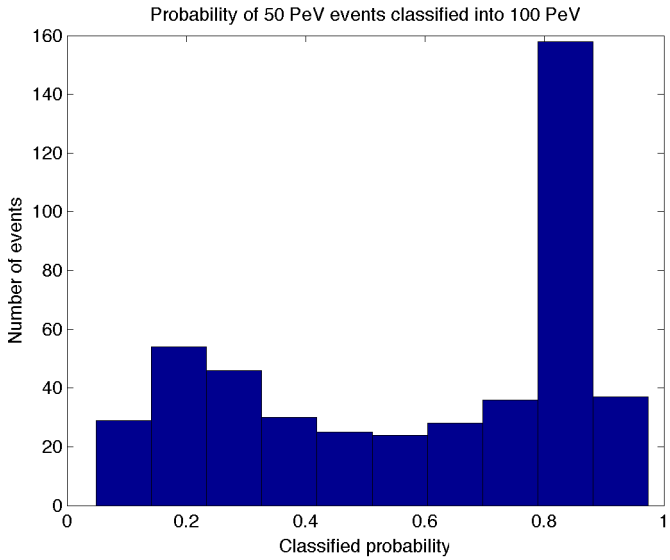


Figure 4.10: The probability of 50 PeV events classified into 100 PeV by SVM

Classification for multiple energy scales

In this section, we use logistic regression with lasso and SVM with radial kernel to classify three energy scales, 20 PeV, 50 PeV and 100 PeV. We merge all different distance scales (2km, 3km, 5km,7km and 10km) for each energy scale.

The classification result of logistic regression is shown in $Table\ 4.4$ and that of SVM is shown in $Table\ 4.5$. The algorithm to generate the results is as follows,

- Given data: events with 20 PeV, 50 PeV and 100 PeV
- To generate the result for 20 PeV, we split the events with 20 PeV into 10 segments. For each run, combine nine segments with all events in the energy of 50 PeV and 100 PeV as the training set, and the leftover one segment as the testing set.
- Use corresponding classification method and training set to train the classifier and then do the test using the testing set. For each run, we have the probability of events with 20 PeV to be classified into all three energy scales. Since we have 10 runs, we

will get 10 numbers for each energy scale. The mean will be the final classification probability and the number within the parentheses is the standard deviation of the 10 numbers.

- repeat the same procedures for 50 PeV and 100 PeV

Table 4.4: Energy identification accuracy by logistic regression for three-class

Test vs predicted	20 PeV	50 PeV	100 PeV
20 PeV	0.59(0.09)	0.40(0.09)	0.01(0.01)
50 PeV	0.28(0.05)	0.39(0.07)	0.33(0.08)
100 PeV	0.13(0.04)	0.27(0.05)	0.60(0.07)

Table 4.5: Energy identification accuracy by SVM for three-class

Test vs predicted	20 PeV	50 PeV	100 PeV
20 PeV	0.60(0.09)	0.38(0.10)	0.01(0.02)
50 PeV	0.32(0.08)	0.42(0.07)	0.25(0.04)
100 PeV	0.12(0.04)	0.33(0.06)	0.55(0.06)

4.4 Conclusion

In this section, two widely used classification methods, logistic regression and SVM are applied to identify the energy regime of ν_τs based on the hit patterns from their corresponding τ air showers. A comparison between the two methods is done via the ROC curves, which shows SVM does not outperform logistic regression in our case. One possible reason is that the events we use in the classification are a mixture of events with five different shower development lengths. Both two methods show strong abilities to distinguish events at the energy of 20 PeV and 100 PeV. Three-class classification is also done to give us a picture about how well we can achieve with our classifiers to identify events into smaller energy intervals.

Chapter 5

Future Work

5.1 Different decay modes

As we have already discussed in Chapter 2, we have five main decay mode of τ lepton. However, limited by the time, we only analyze the $\pi\pi\pi$ mode in detail. Although we have claimed that it should be the energy goes into the shower matters instead of the decay mode, we would still like to verify this viewpoint by analyzing the other four decay modes in the future.

5.2 Classification

5.2.1 Feature extraction

For each shower, we extract 8 features as described in Chapter 4. However, such extraction lost some geometric information of the hit pattern. In the future, we will try to find out more useful features to see if they could help improving the quality of classification.

5.2.2 Resolution of energy estimator

Until now, we have discussed the difficulties encountered when we use the hit patterns of a shower to decide its energy. The way to deal with that is to discretize the energy scale into several bins, and use classification methods to decide the specific energy bin each shower will fall in. The bin width decides the energy resolution of the classifier. For now, we can have a high accuracy in distinguish the events with 20 PeV and 100 PeV as shown in $Table$ **??**. But limited by the computing powers, we only did simulations for several energies. In the future, we will run more samples to cover more energy scales, say, all the energies from 10 PeV to 200 PeV with the step size at 10 PeV. Then we can get the energy resolution by finding out the two closest energies who can generate us a non-trivial classifier, which has the classification accuracy greater than 0.6 (the trivial classifier has the mean accuracy at 0.5).

5.3 Location of detector array

The geometry requirements for our detector array is shown in $Fig.\ 1.5$. The left and right side's objects stand for two mountain peaks while the middle part represents a valley. The right side's plane OBC is the position we will mount our detector stations on, and the left side's mountain peak is used to suppress the environment noise.

The angles of $\angle COH$ and $\angle DAE$ are all about 30 degree. The distance between two mountains AO is about 6 km and the length of valley OB should be at least 2.5 km. The height of both peaks CH and DE should be at least 1 km.

It is believed that the possible location candidates within the US can be found in Utah or Colorado states. And hence we download the geometry data from USGS website, which provide us the elevation information the land grid for the two states. After we download

the geometry data in the format of ".img", a software named ERDAS need to be used to transfer the ".img" files into ".grd" files which is similar to ".txt" and can be read by text editors. The detailed explanation of these steps can be found in Appendix B.

Each dataset stands for one area region with highest and lowest elevation provided. Since we will need the mountain peak with height at about 1 km, all the regions with the difference between highest and lowest elevation less than 1 km will be firstly excluded. And then we will calculate the inclined angle of two neighbor grid based on their elevations together with longitudinal and latitudinal degrees. We will need this inclination to last at 30 degree for at least 1.7 km, while the inclination on the perpendicular direction lasts at about 0 degree for at least 2.5km. This is the requirement from the mountain peak and valley length. After we have found all the candidate peaks, we have to pick up the pairs of peaks such that their distance is about 6 km to fulfill the requirement from valley width. And finally we will select the location manually from all selected pairs of mountain peaks.

Chapter 6

Conclusion

In this thesis, we have discussed some physics background about ultra-high energy neutrinos from extra-galactic sources. Many theories have been developed to explain but they are all guesses at this energy scale due to the lacking of experimental observations. Hence ultra-high energy neutrino experiments are of great necessity. The IceCube experiment has been operating for more than eight years and has observed twenty-eight neutrino candidates for extra-galactic neutrinos. But only two events are above 1 PeV. The history of cosmic ray physics emphasizes the importance of having independent experiments to measure important quantities like the extra-galactic neutrino spectrum above 1 PeV. Therefore, TAUWER project, with a different mechanism to detect ultra-high energy neutrinos, has been introduced. Our study of simulation events shows that TAUWER can be optimized to have sensitivity above 10 PeV, and the event rates we estimated show that our experiment has more counts per year than IceCube produces at lower energy. It is also explained that TAUWER array is modular in square kilometer array sizes and can be expanded to improve sensitivity with relatively low cost. Classification methods from machine learning field are applied here to estimate the energy of an individual neutrino interaction event, based on the shower statistics. Because the energy scale is logarithmic, even the present estimator is

good enough to give an indication of the spectral behavior of observed events. Since the accuracy of the present energy estimator is limited by the available statistics of the training and evaluation samples, the estimator can be made more precise by having higher sample populations that enable additional factors to be included in the estimation algorithm, employing the same methods we used in this thesis.

Appendices

Appendix A

Codes to analyze CORSIKA data sets

A.1 The code to do pre-selection and divide each data file shower by shower

Perl is good at texture manipulation because of its strong "regular expression" function. We will exclude all the lines include "nan' and then select lines with PIDs represent electron, muon and their anti-particles. The codes "divide.pl" is shown as follows,

```
#! /usr/bin/perl -w
use strict;
my $count = 0;
my $flag = 0;
open ZENITH, '>', "zenith_100evt.dat";
while(<>){
chomp;
if(/nan/){
print "nan\n";
next;
}
my @lines = split;
if(/1e\+08/){
$count++;
```

```
print $count,"th event starts\n";
print ZENITH $lines[4],"\n";
open OUTPUTFILE, '>', "shower$count.dat";
$flag = 1;
}
if($flag == 1){
print OUTPUTFILE $_,"\n" if(($lines[1]==2) || ($lines[1]==3) || ($lines[1]==5) || ($lines[1]==6));
}
if(/ -999 /){
print $count,"th event ends\n";
close OUTPUTFILE;
$flag = 0;
last if ($count == 100);
}
}
close ZENITH;
```

A.2 The code to unthin the dataset

```
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <fstream>
#include "time.h"
#include "omp.h"
#include "TMath.h"
#include "TFile.h"
#include "TVector3.h"
#include "TRandom3.h"
using namespace std;
const Double_t Pi = 3.14159265359;

Int_t main(){
clock_t evstart,evend;
Double_t timeregion = 0;
evstart = clock();
Int_t num = 0;
Double_t alpha = 30*TMath::Pi()/180;
Double_t theta[100] = {0};
ifstream anglefile;
```

```
anglefile.open("zenith_100evt.dat");
while(!anglefile.eof()){
anglefile >> theta[num];
theta[num] = (theta[num]*180/TMath::Pi()-90)*Pi/180;
//cout<<num<<" "<<theta[num]<<endl;
num++;
}
anglefile.close();

//TVector3 o(-16700,0,1600); //for 2.5km;
//TVector3 o(-15500,0,1640); //for 3km;
TVector3 o(-14000,0,1690); //for 5km;
//TVector3 o(-11500,0,1790); //for 7.5km;
//TVector3 o(-9000,0,1900); //for 10km;
//TVector3 o(-4000,0,2100); //for 15km;
num = 0;

#pragma omp parallel for private(num)
for(int iter=0;iter<100;iter++){
num = 0;
Int_t threadnum = omp_get_thread_num();
TVector3 a;
a(0) = -750/tan(theta[iter]);
a(1) = 0;
a(2) = 1500;
TVector3 c;
c(0)=(-a(0)*tan(theta[iter])+o(0)*tan(alpha)+a(2)-o(2))/(tan(alpha)-tan(theta[iter]));
c(1) = 0;
c(2) = tan(theta[iter])*(c(0)-a(0))+a(2);
TVector3 n1 = c-a;
n1=n1.Unit();
TVector3 n2;

ifstream infile;
ofstream o_file;
TRandom3 r1(0);
Int_t ipart = 0;
Int_t iflag = 0;
Double_t pp = 0, px = 0, py = 0, pz = 0;
Double_t x1 = 0, y1 = 0, z1 = 0, t1 = 0;
Double_t wei = 0;
Double_t angle = 0;
```

```
Double_t xf = 0, yf = 0, zf = 0;
Double_t xg = 0, yg = 0, zg = 0;
Double_t r = 0;
char infilename[100] = {0};
char outfilename[100] = {0};
sprintf(infilename,"shower%d.dat",iter+1);
sprintf(outfilename,"shower%d_unthinned.dat",iter+1);
infile.open(infilename,ifstream::in);
o_file.open(outfilename);
while(!infile.eof()){
infile>>ipart>>iflag>>pp>>px>>py>>pz>>x1>>y1>>z1>>t1>>wei;
if(infile.fail()) break;
num++;
wei = TMath::Nint(wei);
if(wei < 1) continue;
x1=x1/100;
y1=y1/100;
z1=z1/100;
if(iflag == 2 —— iflag == 3 —— iflag == 5 —— iflag == 6 —— iflag == 14 —— iflag
== 15){
xf = ((c(2)-a(2))*c(0)/(c(0)-a(0))+x1/tan(theta[iter])+z1-c(2))/((c(2)-a(2))/(c(0)-a(0))+1/tan(theta[iter]));
yf = 0;
zf = z1-(xf-x1)/tan(theta[iter]);
r = sqrt((x1-xf)*(x1-xf)+y1*y1+(z1-zf)*(z1-zf));
TVector3 f(xf,yf,zf);
TVector3 p(x1,y1,z1);
TVector3 mom(px,py,pz);
n2=p-f;
n2=n2.Unit();
TVector3 n3 = n1.Cross(n2);
n3=n3.Unit();
Double_t a1 = mom*n1;
Double_t a2 = mom*n2;
Double_t a3 = mom*n3;
o_file<<iflag<<" "<<pp<<" "<<px<<" "<<py<<" "<<pz<<" "<<x1<<" "<<y1<<"
"<<z1<<" "<<t1<<endl;
Double_t beta1 = pp*1000;//find the speed of the track
if(iflag == 2 —— iflag == 3) beta1 = beta1/sqrt(beta1*beta1+0.511*0.511);
if(iflag == 5 —— iflag == 6) beta1 = beta1/sqrt(beta1*beta1+105.7*105.7);
if(iflag == 14 —— iflag == 15) beta1 = beta1/sqrt(beta1*beta1+938.3*938.3);
for(Int_t np=0;np<wei-1;np++){
angle = r1.Uniform(0,2*Pi);
```

```
xg = 0;
yg = r*cos(angle);
zg = r*sin(angle);
TVector3 g(xg,yg,zg);
g.RotateY(-theta[iter]);
xg = g(0);
yg = g(1);
zg = g(2);
g += f;
xg = g(0);
yg = g(1);
zg = g(2);
TVector3 n4 = g-f;
n4 = n4.Unit();
TVector3 n5 = n1.Cross(n4);
TVector3 momg = a1*n1+a2*n4+a3*n5;
momg=momg.Unit()*pp;
Double_t xk = 0, yk = 0, zk = 0;
Double_t p1 = 0, p2 = 0, p3 = 0;
p1 = momg(0);
p2 = momg(1);
p3 = momg(2);
xk = ((p3*xg)/p1+o(2)-o(0)*tan(alpha)-zg)/(p3/p1-tan(alpha));
yk = p2*(xk-xg)/p1+yg;
zk = p3*(xk-xg)/p1+zg;
//time difference
Double_t delta_dis = sqrt((xg-xk)*(xg-xk)+(yg-yk)*(yg-yk)+(zg-zk)*(zg-zk));
Double_t speed = 3.0*pow(10.0,8.0)*beta1;
Double_t delta_time = delta_dis/(speed);
if(xk<xg)
delta_time = -delta_time;
Double_t newt1 = t1+delta_time;
o_file<<iflag<<" "<<pp<<" "<<p1<<" "<<p2<<" "<<p3<<" "<<xk<<" "<<yk<<"
"<<zk<<" "<<newt1<<endl;
}
}
}
o_file.close();
infile.close();
}
evend = clock();
timeregion = (Double_t)(evend-evstart);
```

```
return 0;
}
```

A.3 The code to analyze showers

As discussed in Chapter 6, we would like to provide the two versions of code to analyze hit patters of showers. The first version is names as PC version, which reads the data files line by line directly from the hard drive. Hence it has low memory requirement and can be run on local machines. The second version is named as blacklight version, which firstly read all the data files into memory and then scan the files stored in memory instead of hard drive. The blacklight version requires large memory and can save a great amount of time on complexity systems such as Pittsburgh Supercomputing Center, but it will not helpful on local machines.

For simplicity, we only provide the blacklight version as follows, and the PC version can be obtained by replacing the buffer read code as discussed in Chapter 6.

```
#include <cstdio>
#include <cstdlib>
#include <streambuf>
#include <string>
#include <iostream>
#include <fstream>
#include <time.h>
#include "omp.h"
#include "TMath.h"
#include "TFile.h"
#include "TRandom3.h"
#include "TVector3.h"
#include "TH2.h"
#include "TCanvas.h"
#include "TStyle.h"
```

```
using namespace std;

const Double_t Pi = 3.14159265358979312;
const Int_t nshift = 1;
const Double_t mydistance = 40;

struct membuf : std::streambuf {
membuf(char* begin, char* end) {
this->setg(begin, begin, end);
}
};

Int_t shiftcount(Double_t x1, Double_t y1, Double_t x2, Double_t y2, Double_t pid, Int_t
i, Int_t j, Int_t k, Int_t& num_hit, Int_t (&det_tot)[1][640], Int_t (&detel)[1][640], Int_t
(&detmu)[1][640], Int_t (&detpr)[1][640], Int_t (&detot)[1][640]){
Int_t num = 0;
Int_t flag = 0;
Double_t x = 0, y = 0;
if( (( (x1>=-0.2)&&(x1<=0.2) ) && ( (y1>=0)&&(y1<=0.2) )) || (( (x2>=-0.2)&&(x2<=0.2)
) && ( (y2>=0)&&(y2<=0.2) )) ){
if(pid==2 || pid==3 || pid==5 || pid==6 || pid==14 || pid==15){
det_tot[k][i*40+j]++;
num_hit++;
if(pid==2 || pid==3) detel[k][i*40+j]++;
else if(pid==5 || pid==6) detmu[k][i*40+j]++;
else if(pid==14 || pid==15) detpr[k][i*40+j]++;
else {
detot[k][i*40+j]++;
cout≪"strange pid: "≪pid≪endl;
}
if(( (x1>=-0.2)&&(x1<=0.2) ) && ( (y1>=0)&&(y1<=0.2) )) flag += 1;
if(( (x2>=-0.2)&&(x2<=0.2) ) && ( (y2>=0)&&(y2<=0.2) )) flag += 2;
}
}
return flag;
}

Int_t main(int argc, char* argv[]){
if(argc!=5){
cout≪"Format Error, Format should be: ./detectors 2km 0 5 100"≪endl;
return -1;
}
```

```
for (int i = 0; i < argc; ++i) {
std::cout ≪ argv[i] ≪ std::endl;
}
const Int_t showernum = (const Int_t)atof(argv[4]);

Double_t angleshift = atof(argv[2])*TMath::Pi()/180;
TRandom3 r(0);
Double_t alpha = 30*TMath::Pi()/180;
Double_t meantheta = 2.25*TMath::Pi()/180;

Double_t *theta = new[showernum] Double_t;
Int_t num=0;
ifstream anglefile;
string infilepath1 = argv[1];
char infilename1[100] = {0};
sprintf(infilename1,"pipipi_%s_data/zenith_100evt.dat",infilepath1.c_str());
anglefile.open(infilename1);
while(!anglefile.eof()){
anglefile≫theta[num];
theta[num] = (theta[num]*180/TMath::Pi()-90)*Pi/180;
num++;
}
anglefile.close();

Int_t i=0, j=0, k=0;

TVector3 o;
if(!strcmp(argv[1],"2km"))
o.SetXYZ(-16700,0,1600);
else if(!strcmp(argv[1],"3km"))
o.SetXYZ(-15500,0,1640);
else if(!strcmp(argv[1],"5km"))
o.SetXYZ(-14000,0,1690);
else if(!strcmp(argv[1],"7km"))
o.SetXYZ(-11500,0,1790);
else if(!strcmp(argv[1],"10km"))
o.SetXYZ(-9000,0,1900);
else if(!strcmp(argv[1],"15km"))
o.SetXYZ(-4000,0,2100);
else{
cout≪"ERROR!"≪endl;
cout≪argv[1]≪endl;
```

```
return -1;
}
TVector3 meana;
meana(0) = -750/tan(meantheta);
meana(1) = 0;
meana(2) = 1500;
TVector3 meanc;
meanc(0)=(-meana(0)*tan(meantheta)+o(0)*tan(alpha)+meana(2)-o(2))/(tan(alpha)-tan(meantheta));
meanc(1) = 0;
meanc(2) = tan(meantheta)*(meanc(0)-meana(0))+meana(2);
TVector3 meann = meanc-meana;
Double_t d0 = 0.2/tan(7*Pi/180);
//Calculate the coodinate of each pair of detectors;
Double_t det1x[640] = {0};
Double_t det1y[640] = {0};
Double_t det1z[640] = {0};
Double_t det2x[640] = {0};
Double_t det2y[640] = {0};
Double_t det2z[640] = {0};
for(i=0;i<16;i++){
for(j=0;j<40;j++){
det1z[i*40+j] = meanc(2)+mydistance/2-8*mydistance+i*mydistance;
det1x[i*40+j] = (det1z[i*40+j]-o(2))/(tan(alpha))+o(0);
det1y[i*40+j] = mydistance*j-20*mydistance+mydistance/2;
det2x[i*40+j] = det1x[i*40+j]-d0*cos(meantheta);
det2y[i*40+j] = det1y[i*40+j];
det2z[i*40+j] = det1z[i*40+j]-d0*sin(meantheta);
}
}


#pragma omp parallel for private(num, i, j, k) schedule(dynamic,1)
for(Int_t iter=0; iter<showernum;iter++){
TVector3 a;
a(0) = -750/tan(theta[iter]);
a(1) = 0;
a(2) = 1500;
TVector3 c;
c(0)=(-a(0)*tan(theta[iter])+o(0)*tan(alpha)+a(2)-o(2))/(tan(alpha)-tan(theta[iter]));
c(1) = 0;
c(2) = tan(theta[iter])*(c(0)-a(0))+a(2);
TVector3 n = c-a;
n=n.Unit();
```

```
TVector3 direcshift = n;
direcshift.RotateY(-TMath::Pi()/2);


Int_t det_tot[nshift][640] = {0};
Int_t detel[nshift][640] = {0};
Int_t detmu[nshift][640] = {0};
Int_t detpr[nshift][640] = {0};
Int_t detot[nshift][640] = {0};
Int_t flag = 0;
char infilename[100] = {0};
char inclinehit[100] = {0};
char outfiletime[100] = {0};
ofstream o_file;
ofstream o_file_time;
Int_t threadnum = omp_get_thread_num();
sprintf(infilename,"pipipi_%s_data/shower%d_unthinned.dat",infilepath1.c_str(),iter+1);
sprintf(inclinehit,"shower%d_incline_hit.dat",iter+1);
sprintf(outfiletime,"shower%d_time_hit.dat",iter+1);
filebuf *pbuf;
ifstream filestr;
long size;
filestr.open(infilename,ios::binary);
pbuf = filestr.rdbuf();
size = pbuf->pubseekoff(0,ios::end,ios::in);
pbuf->pubseekpos(0,ios::in);
char *buffer = new[size] char;
pbuf->sgetn(buffer,size);
filestr.close();
membuf sbuf(buffer,buffer+size);
istream infile(&sbuf);
o_file.open(inclinehit);
o_file_time.open(outfiletime);
Double_t pid = 0;
Double_t pp = 0, px = 0, py = 0, pz = 0;
Double_t x1 = 0, y1 = 0, z1 = 0, t1 = 0;
Double_t x2 = 0, y2 = 0, z2 = 0;
TVector3 incline1;
TVector3 hit1, hit2;
Int_t num_hit = 0;
while(!infile.eof()){
num++;
infile≫pid≫pp≫px≫py≫pz≫x1≫y1≫z1≫t1;
```

```
if(infile.fail()) break;
if(t1<0) continue;
Double_t tmpx = x1+atof(argv[3]);
Double_t tmpy = y1;
Double_t tmpz = z1;

x1 = (o(2)+pz*tmpx/px-o(0)*tan(alpha)-tmpz)/(pz/px-tan(alpha));
y1 = py*(x1-tmpx)/px+tmpy;
z1 = pz*(x1-tmpx)/px+tmpz;

TVector3 xangleshift(x1,y1,z1);
TVector3 pangleshift(px,py,pz);
xangleshift = xangleshift-c;
xangleshift.Rotate(angleshift,direcshift);
pangleshift.Rotate(angleshift,direcshift);
xangleshift = xangleshift+c;
px = pangleshift(0);
py = pangleshift(1);
pz = pangleshift(2);
x1 = xangleshift(0);
y1 = xangleshift(1);
z1 = xangleshift(2);
if(px == 0) continue;
for(i=0;i<16;i++){
hit1(0) = (pz*x1/px+n(0)*det1x[i*40]/n(2)+det1z[i*40]-z1)/(pz/px+n(0)/n(2));
hit1(1) = py*(hit1(0)-x1)/px+y1;
hit1(2) = pz*(hit1(0)-x1)/px+z1;
hit2(0) = (pz*x1/px+n(0)*det2x[i*40]/n(2)+det2z[i*40]-z1)/(pz/px+n(0)/n(2));
hit2(1) = py*(hit2(0)-x1)/px+y1;
hit2(2) = pz*(hit2(0)-x1)/px+z1;

Double_t tmp1_x,tmp1_y,tmp1_z,tmp2_x,tmp2_y,tmp2_z;
for(j=0;j<40;j++){
tmp1_y = hit1(1)-det1y[i*40+j];
tmp1_z = hit1(2)-det1z[i*40+j];
tmp2_y = hit2(1)-det2y[i*40+j];
tmp2_z = hit2(2)-det2z[i*40+j];
for(k=0;k<nshift;k++){
flag = shiftcount(tmp1_y,tmp1_z/cos(theta[iter]),tmp2_y,tmp2_z/cos(theta[iter]),pid,i,j,k,num_hit,
det_tot, detel,detmu,detpr,detot);
if(flag > 0){
incline1(0) = x1;
```

```
incline1(1) = y1;
incline1(2) = z1;
incline1 = incline1-c;
incline1.RotateY(-60*Pi/180);
o_file≪pid≪" "≪incline1(0)≪" "≪incline1(1)≪" "≪incline1(2)≪'\n';
o_file_time≪pid≪" "≪i≪" "≪j≪" "≪" "≪flag≪" "≪det1x[i*40+j]≪" "≪det1y[i*40+j]≪"
"≪det1z[i*40+j]≪" "≪det2x[i*40+j]≪" "≪det2y[i*40+j]≪" "≪det2z[i*40+j]≪" "≪pp≪"
"≪px≪" "≪py≪" "≪pz≪" "≪x1≪" "≪y1≪" "≪z1≪" "≪t1≪'\n';
flag = 0;
}
}//end nshift
}//end j
}//end i
}//end while
o_file.close();
o_file_time.close();


ofstream o_file_1;
ofstream o_file_2;
ofstream o_file_3;
ofstream o_file_4;
char detectorhit[40] = {0};
char detectorhit_el[40] = {0};
char detectorhit_mu[40] = {0};
char detectorhit_pr[40] = {0};
char detectorhit_ot[40] = {0};
sprintf(detectorhit,"shower%d_detectors_hit.dat",iter+1);
sprintf(detectorhit_el,"shower%d_detectors_el_hit.dat",iter+1);
sprintf(detectorhit_mu,"shower%d_detectors_mu_hit.dat",iter+1);
sprintf(detectorhit_pr,"shower%d_detectors_pr_hit.dat",iter+1);
sprintf(detectorhit_ot,"shower%d_detectors_ot_hit.dat",iter+1);
o_file.open(detectorhit);
o_file_1.open(detectorhit_el);
o_file_2.open(detectorhit_mu);
o_file_3.open(detectorhit_pr);
o_file_4.open(detectorhit_ot);
for(k=0;k<nshift;k++){
for(i=0;i<16;i++){
for(j=0;j<40;j++){
o_file≪det_tot[k][40*i+j]≪" ";
o_file_1≪detel[k][40*i+j]≪" ";
o_file_2≪detmu[k][40*i+j]≪" ";
```

```
o_file_3≪detpr[k][40*i+j]≪” ”;
o_file_4≪detot[k][40*i+j]≪” ”;
}
}
o_file≪’\n’;
o_file_1≪’\n’;
o_file_2≪’\n’;
o_file_3≪’\n’;
o_file_4≪’\n’;
}
o_file.close();
o_file_1.close();
o_file_2.close();
o_file_3.close();
o_file_4.close();

char detectorplot[40] = {0};

for(k=0;k<nshift;k++){
for(i=0;i<640;i++){
det_tot[k][i] = 0;
detel[k][i] = 0;
detmu[k][i] = 0;
detpr[k][i] = 0;
detot[k][i] = 0;
}
}
num = 0;
num_hit = 0;
flag = 0;
delete[] buffer;
cout≪”New Shower”≪endl;
}
delete[] theta;
return 0;
}
```

And we have run a test job for both versions on blacklight. For PC version, the run time is,

Elapsed Time : 2406 Seconds

User CPU Time : 1488.6680 Seconds

System CPU Time : 28943.3830 Seconds

Maximum Core Memory Used : 27.8672 Mbytes

Characters Read : 94.7131 Mbytes

Characters Written : 7903.5594 Mbytes

Logical I/O Read Requests : 12734

Logical I/O Write Requests : 104824829

CPU Delay : 1200.4635 Seconds

Block I/O Delay : 105.2895 Seconds

For blacklight version, the run time is,

Elapsed Time : 804 Seconds

User CPU Time : 4166.7640 Seconds

System CPU Time : 6203.0470 Seconds

Maximum Core Memory Used : 121.6680 Mbytes

Characters Read : 94.5451 Mbytes

Characters Written : 7903.4014 Mbytes

Logical I/O Read Requests : 672

Logical I/O Write Requests : 1011579

CPU Delay : 15.8707 Seconds

Block I/O Delay : 11.6889 Seconds

From the two results we can see that running speed of blacklight version is largely improved compared to PC version due to the significant decrease of I/O times.

The script we use for RAMDISK mechanism is,

```
#! /bin/csh
#PBS -l walltime_min=10:50:00
#PBS -l walltime_max=12:00:00
#PBS -l ncpus=256
#PBS -j oe
#PBS -q batch
set echo
set EXE=detectors
set BINDIR=/usr/users/6/tminuit/TAUWER/job_03_15_2014
set RESULT=/brashear/tminuit/inclined_30degree/2E17_40m
set WORKDIR=$SCRATCH_RAMDISK
setenv OMP_NUM_THREADS PBS_$NCPUS
cd $WORKDIR
cp $BINDIR/$EXE .
mkdir 0degree 10degree 20degree 30degree 40degree 50degree 60degree 70degree
ja
omplace -nt $OMP_NUM_THREADS ./$EXE 10km 0 0 80 /brashear/tminuit/unthin/2E17
0degree &
omplace -nt $OMP_NUM_THREADS ./$EXE 10km 10 0 80 /brashear/tminuit/unthin/2E17
10degree &
omplace -nt $OMP_NUM_THREADS ./$EXE 10km 20 0 80 /brashear/tminuit/unthin/2E17
20degree &
omplace -nt $OMP_NUM_THREADS ./$EXE 10km 30 0 80 /brashear/tminuit/unthin/2E17
30degree &
omplace -nt $OMP_NUM_THREADS ./$EXE 10km 40 0 80 /brashear/tminuit/unthin/2E17
40degree &
omplace -nt $OMP_NUM_THREADS ./$EXE 10km 50 0 80 /brashear/tminuit/unthin/2E17
50degree &
omplace -nt $OMP_NUM_THREADS ./$EXE 10km 60 0 80 /brashear/tminuit/unthin/2E17
60degree &
omplace -nt $OMP_NUM_THREADS ./$EXE 10km 70 0 80 /brashear/tminuit/unthin/2E17
70degree &
wait
ja -cshlt
mkdir pipipi$_$10km
mv *degree pipipi_10km/.
mv pipipi_10km $RESULT/.
unset echo
```

And the time report is shown as follows, which tells us the RAMDISK will bring us about

85% improvement.

Elapsed Time : 117 Seconds

User CPU Time : 1212.9080 Seconds

System CPU Time : 10.4760 Seconds

Maximum Core Memory Used : 123.1328 Mbytes

Maximum Virtual Memory Used : 634.5312 Mbytes

Characters Read : 94.3445 Mbytes

Characters Written : 7903.4140 Mbytes

Logical I/O Read Requests : 134

Logical I/O Write Requests : 1011615

CPU Delay : 0.3368 Seconds

Block I/O Delay : 0.0000 Seconds

Appendix B

Steps to obtain readable geometry datasets

B.1 Download geometry data

We have selected Colorado and Utah states as the region to find ideal position for our detector array. Elevation maps about the two states are downloaded from USGS website. The following steps show the details about that and the same methods can be used to other states if we decide to enlarge our search region in the future.

- Go to national map viewer and select the interested region.
- Choose “Elevation” and click next. Then download the 1/3 arc-second .img files.

B.2 Tranfer .img to .grd via ERDAS

This process is under the instruction of Debaleena Majumdar. Since the downloaded img files cannot be directly used by programs such as C++ or python, we need to transfer these

files into grd files by ERDAS and then do future analysis. The steps to generate grd files are listed as follows,

- Create a 2D Viewer by File→New→2D View. And load img file into ERDAS via this 2D Viewer by File→Open→Raster Layer.
- Go to File→Export Data. And select export format as GRD (Surfer:ASCII/Binary). Click save, and a new window will come out. Choose the ASCII GRD File option, and click OK. Then the grd file will be generated.

Appendix C

Computing techniques

In this chapter, we will discuss the computing techniques we applied in our research, such as multi-thread programming, and ramdisk in supercomputer center.

C.1 Multithreading programming

Nowadays, most of the desktops have used CPUs with multi-cores, and each core can run one to two threads. Take the cpu I use as an example. It is i7-3770k produced by Intel, with 4 cores in it and each core can handle two threads. Hence this cpu can run eight parallel jobs at the same time, which could largely shrink the computing time.

The ideal speed should be eight times faster. However, several drawbacks could reduce the speed of multithreading. One major point is, the intermediate variable within each thread may need to communicate each other which is very time consuming. The running time is usually decided by the slowest thread, since all the other threads need to wait for it no matter how fast they finish their own jobs. If we can reduce the communication between different threads and assign approximately equal load for each thread, these drawbacks can be largely suppressed.

Our jobs fortunately fulfill the requirements to suppress the drawbacks of multithreading. For each energy scale at specific decay mode, we will generate about 100 showers from CORSIKA, and the main information we want to obtain from them is to find out the hit pattern of each shower, which requires minimal communication between the analysis of two showers. At the same time, since the showers are simulated from the same initial condition, many of them will have similar shower size in general even though big variance between showers can exist. Based on that, we can see that the multithreading will be of great help in our analysis.

There are several popular API (application programming interface) available to implement the multithreading such as OpenMP, MPI and pthreads. We apply OpenMP in our jobs. And it can be done simply by adding the following short code in front of the for loop where we analysis the showers one by one.

#pragma omp parallel for private(num, i, j, k) (C.1)

Where the private means that the variables num, i, j and k used in each thread are totally different even though they are in the same names.

Another improvement can be made through dynamic mechanism by modifying the previous code as follows,

#pragma omp parallel for private(num, i, j, k) schedule(dynamic,1) (C.2)

In the previous case shown in ($C.1$), the computer will assign each shower a specific thread number before they are analyzed. Since we have eight threads available for i7-3770k CPU and 100 shower waiting for run, the computer will assign shower $i + 8j$ to thread i, where $i = 1, 2, ...8$, $j = 0, 1, ...$ and $i + 8j \leq 100$. We find a problem when running in this static

mode. Some threads have to run many large showers and could be really slow while the others threads are just waiting there since the shower assigned to them are very small due to the variance in the 100 showers. If we apply the dynamic mechanism as shown in ($C.2$), such situation can be diminished. The computer will only assign shower 1 to shower 8 to thread 1 to thread 8 correspondingly in the beginning, then the next shower will be assigned dynamically to the thread who just finishes the previous analysis job.

$Table.\ C.1$ gives us the run time of above three mechanisms on analyzing 100 showers with 100 PeV, where "Single" means no multithreading is applied, "Static" means static schedule with eight threads and "Dynamic" means dynamic schedule for eight threads. From the table we can see that the applying of multithreading gives us running speed about

Table C.1: Run time of three mechanisms to show the benefit of dynamic multithreading

Mechanism	Run time
Single	666m58.216s
Static	217m32.958s
Dynamic	174m23.583s

three times faster and the dynamic schedule provides 20% improvement compared to the static schedule.

C.2 Pittsburgh Supercomputing Center (PSC)

To speed up the analysis, we need the blacklight system at PSC. The code we run on blacklight is the same as the code we run on my desktop. But in blacklight, we can apply for much more cores and each core can have one thread. Therefore the ideal running speed should be much faster. The unit available in blacklight is called blade, and each blade includes 16 cores and 128 GB memory. We are required to apply for at least one blade or more, which provides a huge amount of memory. For example, if we apply for two blades,

then we could have 32 threads and 256 GB memory while the typical size of 100 showers is less than 100 GB. This characteristic will be made use of and the details will be explained in this section.

At first, we use the code from my desktop directly to run on blacklight. However, the speed is much lower than we expected. From the running report we find that it is the large amount of I/O result in the slow speed. Therefore we need to modify the code to reduce the I/O times.

The PC version code can read in dataset line by line directly from the hard driver where these data files are stored, which is shown as follows,

```
ifstream infile;
infile.open(infilename);
while(!infile.eof()){
   infile≫pid≫pp≫px≫py≫pz≫x1≫y1≫z1≫t1;
   ...
 }
```
(C.3)

The modification we made is to read in all data files into the memory before we start to analyze them. And then scan the file stored in memory line by line. One requirement is of course, there are enough memeory to store all these data files, and we do have that on

blacklight as discussed in the first paragraph. The code to do that is shown as follows,

```
filebuf *pbuf;
ifstream filestr;
long size;
filestr.open(infilename,ios::binary);
pbuf = filestr.rdbuf();
size = pbuf→pubseekoff(0,ios::end,ios::in);
pbuf→pubseekpos(0,ios::in);
char *buffer = new[size] char;
pbuf→sgetn(buffer,size);
filestr.close();
membuf sbuf(buffer,buffer+size);
istream infile(&sbuf);
```
(C.4)

where the infilename is the dataset we will read into the memory. And now we can easily read the variables line by line from memory just as the while loop shown in (C.3).

The experiment shows that this procedure can largely decrease the running time on blacklight. However, the two versions of codes spend very similar time on my local desktop. The reason for that is on local machine, the contents read from the hard drive is time-consuming but the number of I/O does not matter. On the other side, for computers with huge complexity system and large amount of nodes such as blacklight, the number of I/O time does matter. Hence in the future if we want to run the code on our own desktop, we do not have to use the buffer-reading, which means low memory size is required. But if we want to use blacklight, it is better to use the buffer-reading version which can reduce the time three

times in our case.

Another important improvement is suggested by the expert from PSC, David O'Neal, which is not directly related to our codes but the blacklight system. A technique called RAMDISK can be applied. The main idea is to make use of the available memory to form a "disk", which will be used to store all the results generated by our code. However, this ramdisk will disappear when our blade allocation ends, so we have to copy all the files on the ramdisk to normal hard disk before the job ends. The advantage for RAMDISK is because the Memory I/O speed is usually 100 times faster than that of the hard disk. As well as we have enough memory, this technique could help greatly. A comparison of running time based on a test job is listed in $Table.\ C.2$, where "PC" means the version we run on local machine, "blacklight" means the version we use buffer-reading, and "RAMDISK" represents the combination of buffer-reading and RAMDISK technique. And the last one has achieved 20 times improvement.

Table C.2: Run time of three mechanisms to show the benefit of RAMDISK

Mechanism	Run time (s)
PC	2406
Blacklight	804
RAMDISK	117

The complete code of blacklight version and the time analysis report can be found in Appendix A.3.

C.3 More about blacklight

We have discussed a lot of benefits brought by the supercomputing center in last chapter. However, one big drawback of blacklight must be pointed out here. In the script we have attached in the Appendix to apply for the allocation of computing units, we also need to

request the time we need to use. This time will including the running time of the program and the leftover jobs such as copy the generated files from ramdisk to hard drive. If the real time exceeds the time we requested, the whole job will be abandoned with no intermediate results saved. Therefore, a precise estimation of running time is needed. However, different jobs will need different running time. For examples, showers with different energies could result in a significant difference in the size of data files. Such phenomenon largely increases the difficulty to estimate the running time. For now, I use a relatively large time request, which also leads to some waste. In the future, better estimation way should be developed to solve this problem.

Appendix D

Previous analysis work: Fitting CDF Upsilon Mass Data

The Υ mesons consist of bound states with a b quark and an anti-b-quark. They have orbital angular momentum at zero, spin angular momentum at 1 and negative parity. Strong force is the binding force of the Υ system, and hence we can use the Υ mesons to learn the characteristics of strong interactions, which can contribute to the understanding of Quantum Chromodynamics (QCD).

The CDF detector at the Fermilab Tevatron, using events generated in $p\bar{p}$ interactions at 1.96 TeV, can provide us the largest samples for three Υ states: the ground state Υ(1S), and two stable radial excitations Υ(2S) and Υ(3S). The Υ mesons can decay into a pair of $\mu^+\mu^-$ by annihilation of the quark and antiquark. We can therefore obtain the mass distribution of the Υ mesons by calculating the invariant mass of the muon pair. However, many other ways can bring us the muon pair. Hence there is background noise in the mass distribution of the Υ mesons. Meanwhile, the signal peaks corresponding to the three Υ states cannot be fit simply by one Gaussian distribution due to apparatus resolution, kinematic selection cuts, and quantum electrodynamic radiative effects. This Appendix describes a study to use

the combination of different functional forms to fit the mass distribution curves of three Υ states from Monte Carlo simulation. After that, we can have the function forms to describe both the mass distribution of three Υ mesons, as well as the background noise.

D.1 Types of fitting functions

To determine the Υ(nS) yields in the various cosθ^* bins for data, we rely on fitting fixed signal shapes to the distribution, along with a polynomial background. We determine the signal shapes by fitting Monte Carlo events for each Υ(nS) state after processing through CdfSim and Production. Trigger effects are developed from data and are emulated in the BSTntuple step. The resulting mass distributions are binned into histograms and are fit by a smooth functional form.

We studied many different fitting functions and their combinations to fit the upsilon mass distribution. The basic shapes are Gaussian (G), Crystal Ball (CB), Modified Crystal Ball (MCB) or Johnson SU function. The functional forms used and the paramters involved are shown in the following equations.

$$PDF_G = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{D.1}$$

$$PDF_{JSU} = \frac{\delta}{\lambda\sqrt{2\pi}\sqrt{\left(\frac{x-\xi}{\lambda}\right)^2+1}} \exp\left(-1/2\left(\gamma+\delta\log\left(\left(\frac{x-\xi}{\lambda}\right)+\sqrt{\left(\frac{x-\xi}{\lambda}\right)^2+1}\right)\right)^2\right) \tag{D.2}$$

$$PDF_{CB} = \begin{cases} A \exp\left(-\frac{(x-Et)^2}{2\sigma^2}\right) & \frac{x-Et}{\sigma} > -\alpha \\ \\ A(\frac{n}{\alpha})^n \frac{\exp(-\alpha^2/2)}{(\frac{Et-x}{\sigma}+\frac{n}{\alpha}-\alpha)^n} & \frac{x-Et}{\sigma} \leq -\alpha \end{cases} \tag{D.3}$$

$$PDF_{MCB} = \begin{cases} A\left(\exp\left(-\frac{(x-Et)^2}{2\sigma^2}\right) + R\exp\left(-\frac{(x-Et)^2}{2(B\sigma)^2}\right)\right) & \frac{x-Et}{\sigma} > -\alpha \\ \\ A(\frac{n}{\alpha})^n \frac{\exp(-\alpha^2/2)}{(\frac{Et-x}{\sigma}+\frac{n}{\alpha}-\alpha)^n} & \frac{x-Et}{\sigma} \leq -\alpha \end{cases} \tag{D.4}$$

Single function shapes did not fit well at all, so we used combinations of functions: 2G, 3G, JSU+G, JSU+2G, CB+G, JSU+CB. We chose JSU+CB as our fitting function, because it alone can fit all the data distributions successfully with no fit status complaints from Minuit and with suitable χ^2 values. The other options all led to MINUIT fit errors, either an inaccurate error matrix or an error matrix that is not positive-definite. The fit parameters for each p_T and cosθ^* bin for each state are given in Tables 1-3. The label for each column is $(p_T bin)\_(cos\theta^* bin)$. The column at left gives the name of the fit variable as it appears in the CB (first four entries) or JSU function definition. There are a pair of numbers for each entry: the fit value and its uncertainty. The mass centroids for the CB (E_t) and JSU (ϵ) are fixed in all fits (zero uncertainty) to values determined by letting these parameters float, then fixing them to the mean values for the ensemble of distributions. The uncertainties in these parameters are always very small when they float.

To illustrate the difference that the tail function of the CB makes, Fig. 1 shows successful fits results using two different fitting functions JSU+G and JSU+CB for Υ(1S) events gen-

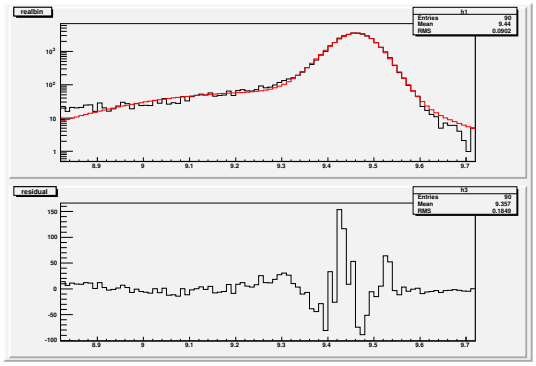
erated with $\mid y \mid < 0.8$ and $3 < p_T < 4$ GeV/c. The histogram data are the same in both plots, all events which survive production and cut selection, integrated over decay angles. The JSU+CB choice has a much better χ^2 value for the fit, in accord with our observation above that it alone behaves well for all the mass fits in p_T and cosθ^* bins.

In order to prepare for a study of possible mass shifts and width scale factors between data and Monte Carlo, we also made fits to data integrated over all decay angles in all p_T bins for the three Υ(nS) states.

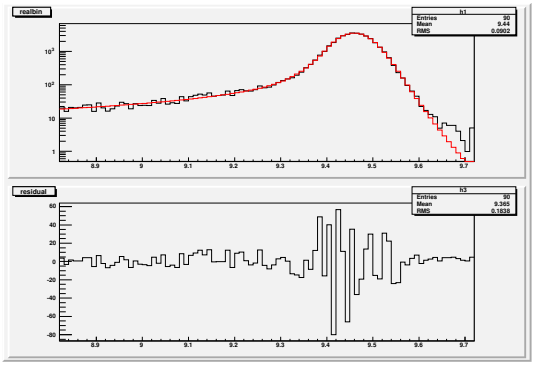
For the 1S and 3S cases the JSU+CB function gave a good fit to the integrated data as well as to the angle-binned data, as shown in the example of Fig. 1. However, for 2S case, we observed indication of a shape discrepancy in the deviation between the Monte Carlo distribution and the bin integral of the JSU+CB fit function. The effect is subtle, but to ensure good behavior in the fit function for scale factor studies we defined a modified crystal ball function (MCB) for the 2S case by introducing a second Gaussian with the same centroid but different width and normalization. In Figure 2 one sees that using a JSU+MCB function improves the agreement between the MC distribution and the fit compared to JSU+CB. The difference is small but noticeable for these statistics. The parameterization of the MCB is given in Eq.(4). The forms for both JSU+CB and JSU+MCB functions are well suited for mass shift and scale factor studies, described in the body of this note. Applying the MCB to the 1S and 3S distributions did not produce a significant improvement in the fit χ^2. The parameters for the fit functions for the angle-integrated distributions are given in Table 4 and the plots are shown in Figs. 3-5 for all three states in all p_T bins.

The mass fits in individual cosθ^* bins for each p_T bin of the 1S data are shown in Fig. 3. For the 2S, they are in Fig. 4. For the 3S they are in Fig. 5.[1].

[1]In Minuit all bins are used. In evaluating the fit we calculate χ^2 only for bins with population content



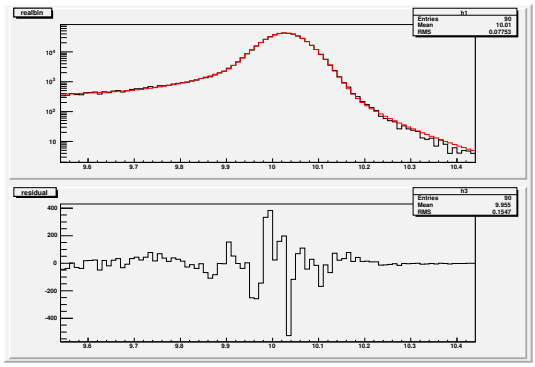
(a) JSU+G(Fit Status = 3, chisq/dof = 121.026/55)



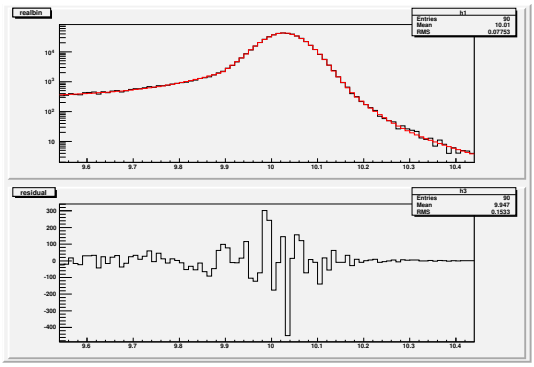
(b) JSU+CB(Fit Status = 3, chisq/dof = 52.4816/55)

Figure D.1: 1S_01 pt bin

>25. In all cases the peak bin population exceeds 6000 events/bin.



(a) JSU+CB(Fit Status = 3, chisq/dof = 143.987/58)



(b) JSU+MCB(Fit Status = 3, chisq/dof = 89.9629/58)

Figure D.2: 2S_0 pt bin

D.2 Results

The following tables describe fitting results for 1S, 2S and 3S states of upsilon. We apply JSU+CB for 1S and 3S, and JSU+MCB for 2S.

TABLE 1: Υ(1S)

1S_10MeV	0_0	0_1	0_2	0_3	0_4	0_5	0_6	0_7	0_8	0_9
Et	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455
	0	0	0	0	0	0	0	0	0	0
sigma	0.0425089	0.038882	0.0411788	0.0413963	0.0401254	0.0409778	0.0402457	0.0426179	0.0396393	0.0330786
	0.00126944	0.00163081	0.00112434	0.00058379	0.00092477	0.00059822	0.00064427	0.00167435	0.00057791	0.00400483
n	0.861583	1.11066	0.970489	0.992449	1.02875	0.942269	1.00726	0.840479	1.11715	0.87432
	0.0609162	0.0657177	0.0579229	0.0708235	0.0713722	0.0690353	0.0712068	0.090559	0.089384	0.186826
alpha	1.80875	1.52522	1.73533	1.85772	1.75655	1.87728	1.82102	1.88804	1.79899	1.3277
	0.0787825	0.100161	0.10093	0.0419161	0.066118	0.0436722	0.0465132	0.112882	0.0439253	0.303074
gamma	0.209741	0.17329	0.181144	0.114666	0.145696	0.0738587	0.0386369	0.274812	-0.226844	0.247094
	0.0298975	0.0293473	0.0471431	0.0688773	0.0450568	0.0883749	0.122295	0.0355258	0.37183	0.0278401
delta	1.65888	2.19359	1.86026	1.2992	1.64155	1.29727	1.47136	1.54277	1.06891	4.05382
	0.228158	0.322157	0.346737	0.164842	0.205134	0.18809	0.249385	0.270192	0.183726	1.83465
lamda	0.0666344	0.091333	0.0745187	0.0483414	0.0647098	0.04908	0.0545291	0.0565496	0.0340228	0.169468
	0.0116191	0.015732	0.0165737	0.00844095	0.00997748	0.00973175	0.0125165	0.0105802	0.0110062	0.0803961
epsilon	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706
	0	0	0	0	0	0	0	0	0	0
a_crystal	0.582759	0.511462	0.590063	0.71461	0.607444	0.766248	0.75842	0.587379	0.868392	0.30193
	0.0746312	0.070998	0.106369	0.0661638	0.0730565	0.0601129	0.078395	0.0910653	0.0634969	0.0926775
chisq	43.1838	52.4812	62.2024	49.827	62.9288	57.743	54.8929	47.2233	50.0251	27.2671
Bin	70	62	65	64	59	60	52	46	44	31
MaximumX	9.46025	9.46001	9.46015	9.46116	9.46074	9.46037	9.46059	9.46029	9.46141	9.45865

1S_10MeV	1_0	1_1	1_2	1_3	1_4	1_5	1_6	1_7	1_8	1_9
Et	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455
	0	0	0	0	0	0	0	0	0	0
sigma	0.040174	0.0389165	0.0438454	0.0399275	0.0429724	0.0413121	0.0412828	0.0404976	0.0482938	0.0394459
	0.00226208	0.00128043	0.00091725	0.00169598	0.00067608	0.00080105	0.00068322	0.00076347	0.00388304	0.00362961
n	0.905437	1.15547	0.876667	1.01777	0.888587	0.855163	0.99855	1.20957	1.35879	0.799432
	0.0777192	0.0662775	0.061799	0.0759942	0.0660951	0.0639603	0.0923418	0.0810603	0.316553	0.233109
alpha	1.64451	1.57014	1.85484	1.71813	1.92059	1.88388	1.83699	1.7	1.25579	1.67212
	0.119183	0.0728748	0.0655761	0.0915831	0.0460591	0.0523314	0.0547661	0.0543494	0.493496	0.241329
gamma	0.180743	0.120256	0.156702	0.171382	0.100292	0.123245	0.0776814	0.0664577	0.294484	0.207197
	0.0258955	0.0392595	0.0573976	0.0381013	0.0590983	0.0630558	0.105968	0.121055	0.020046	0.036907
delta	2.0417	2.33786	1.50931	1.90047	1.25923	1.51175	1.16821	1.62493	3.24239	2.09103
	0.351509	0.404935	0.228227	0.375672	0.144472	0.226763	0.219515	0.303637	0.792767	0.33765
lamda	0.0878081	0.103291	0.0562999	0.0795927	0.045121	0.0583368	0.0443831	0.0629225	0.122184	0.0830137
	0.018036	0.0205184	0.0108161	0.0193004	0.00689335	0.0114213	0.0110989	0.0154411	0.033538	0.015425
epsilon	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706
	0	0	0	0	0	0	0	0	0	0
a_crystal	0.475808	0.584463	0.64982	0.58577	0.709483	0.686241	0.746865	0.728207	0.346219	0.337793
	0.07215	0.0663696	0.0846443	0.0786017	0.0527841	0.0741538	0.077284	0.0990346	0.18831	0.12045
chisq	47.2102	49.5312	45.4496	55.6895	72.8489	58.2478	57.9303	26.6631	29.9171	31.8278
Bin	63	58	59	55	56	58	60	50	46	32
MaximumX	9.46024	9.45937	9.46164	9.45961	9.46233	9.46068	9.46107	9.46026	9.45958	9.46155

1S_10MeV	2_0	2_1	2_2	2_3	2_4	2_5	2_6	2_7	3_0	3_1	3_2
Et	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455
	0	0	0	0	0	0	0	0	0	0	0
sigma	0.0414979	0.0425255	0.0412956	0.0408522	0.0417938	0.0420081	0.0415767	0.0416398	0.0407285	0.0387104	0.043403
	0.00087274	0.001224	0.00141349	0.00164165	0.00071563	0.00111608	0.00078165	0.00061252	0.00118149	0.0014798	0.00090965
n	0.989245	0.896928	0.927184	0.995937	0.915738	0.952756	1.15554	1.23295	0.945176	0.978141	0.920011
	0.0562755	0.0550725	0.0571853	0.0620911	0.0639685	0.0594537	0.0680852	0.0807066	0.0739491	0.0727919	0.0868944
alpha	1.77749	1.74713	1.7389	1.72594	1.87539	1.72696	1.7164	1.7947	1.76551	1.66463	1.84656
	0.0462831	0.0704808	0.0788231	0.0718905	0.0466196	0.0627307	0.0490549	0.0422128	0.0622802	0.0732943	0.055589
gamma	0.0908545	0.17352	0.165311	0.143574	0.121356	0.161103	0.0682772	-0.0242764	0.0920377	0.114042	0.110444
	0.0509381	0.0265954	0.0295157	0.0336733	0.0512056	0.0333947	0.0738749	0.0911325	0.0424114	0.0337292	0.0509844
delta	1.6115	1.6996	1.79554	1.94261	1.47048	1.67836	1.546	1.33214	2.02119	2.47875	1.38262
	0.204416	0.181998	0.270491	0.362114	0.18279	0.169253	0.210584	0.13121	0.307784	0.490395	0.157581
lamda	0.0674308	0.0707555	0.0766364	0.0838888	0.0597945	0.0671744	0.0627251	0.0489618	0.0931732	0.116825	0.0586413
	0.0110231	0.00965653	0.0141754	0.0196409	0.00953748	0.00870737	0.0111477	0.00712784	0.0167642	0.0262024	0.00870731
epsilon	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706
	0	0	0	0	0	0	0	0	0	0	0
a_crystal	0.666838	0.542945	0.575115	0.602333	0.68652	0.572042	0.692524	0.733765	0.604021	0.553561	0.646821
	0.0612401	0.0650349	0.0679236	0.0606921	0.0608609	0.0622679	0.0727168	0.0534158	0.0619466	0.0600742	0.0644066
chisq	89.5586	69.3082	70.9565	69.0584	60.8943	69.1303	77.6858	64.4182	44.1457	61.3344	59.8081
Bin	76	78	76	72	73	72	65	54	59	62	59
MaximumX	9.46041	9.46086	9.46003	9.45968	9.4603	9.46112	9.46072	9.46244	9.45966	9.45907	9.46111

1S_10MeV	3_3	3_4	3_5	3_6	3_7	4_0	4_1	4_2	4_3	4_4	4_5
Et	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455
	0	0	0	0	0	0	0	0	0	0	0
sigma	0.0404472	0.043002	0.0446679	0.0433679	0.039491	0.0417605	0.0417328	0.0466099	0.0477213	0.0433253	0.0461749
	0.00222502	0.00082186	0.0007922	0.00065904	0.00133104	0.00185918	0.00136481	0.00068395	0.00120013	0.00184838	0.00117393
n	0.978555	0.99126	0.962447	1.10602	2.22503	0.886358	0.973029	0.84052	0.845139	0.843975	0.789224
	0.084351	0.0865602	0.0853293	0.0934322	0.343751	0.076685	0.0940518	0.0825284	0.0863962	0.0867226	0.0883743
alpha	1.6946	1.83486	1.84253	1.85859	1.57165	1.58532	1.75044	1.943	1.85656	1.86744	1.91108
	0.11837	0.0525572	0.0539255	0.0459701	0.0797359	0.11673	0.0673033	0.0504429	0.0897816	0.0866809	0.0812861
gamma	0.137485	0.0884843	0.112091	-0.239487	-1.05843	0.174465	0.133917	0.116386	0.199393	0.114983	0.163022
	0.0349153	0.0854788	0.0748618	0.277783	0.489768	0.0236774	0.0374613	0.0633035	0.0359642	0.0467579	0.0524176
delta	2.0555	1.43021	1.23502	1.04398	1.72715	2.23903	2.04479	1.13765	1.37889	1.93093	1.52864
	0.460569	0.236289	0.185608	0.148922	0.225981	0.292558	0.348228	0.150916	0.208022	0.455675	0.271676
lamda	0.0936983	0.0589543	0.0458519	0.0336161	0.0494647	0.10437	0.10246	0.0449803	0.055045	0.0901303	0.064269
	0.0243854	0.0127196	0.00952631	0.00956644	0.0145751	0.0153832	0.0210574	0.00828558	0.0105861	0.0257863	0.0141506
epsilon	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706
	0	0	0	0	0	0	0	0	0	0	0
a_crystal	0.540164	0.717204	0.704235	0.838502	0.854984	0.42422	0.574157	0.724868	0.590515	0.60312	0.62834
	0.0836566	0.0822053	0.0750216	0.0588319	0.0246468	0.0702804	0.0636903	0.0616237	0.0874225	0.082911	0.0985203
chisq	74.2719	41.5369	37.9468	38.9658	34.8621	86.2724	56.9153	51.2187	57.2525	48.319	76.2578
Bin	56	56	49	47	32	58	54	58	53	51	49
MaximumX	9.45975	9.46042	9.46235	9.46413	9.463	9.46012	9.45875	9.46195	9.46185	9.45978	9.46089

1S_10MeV	4_6	4_7	5_0	5_1	5_2	5_3	5_4	5_5	5_6	5_7	6_0
Et	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455
	0	0	0	0	0	0	0	0	0	0	0
sigma	0.0441602	0.0308923	0.088509	0.0441242	0.0518952	0.0466031	0.101048	0.0439797	0.0612643	0.0458825	0.0812936
	0.00160097	0.00957508	0.0187178	0.00422741	0.0139128	0.00324452	0.0220763	0.0369955	0.00396949	0.00220055	0.0173389
n	0.835066	20	0.946546	0.266817	0.513344	0.332216	3.83901	20	0.494242	2.76062	20
	0.122254	10.0795	1.22942	0.178837	0.258233	0.156424	13.626	17.2274	0.16168	2.00847	13.2744
alpha	1.79286	0.270219	0.452263	1.86304	1.15021	2.07558	0.32246	0.0443491	1.89493	1.57005	0.189383
	0.0897331	0.106194	0.615994	0.292677	0.710669	0.160758	0.356062	0.0504218	0.159659	0.183719	0.114807
gamma	0.170282	0.107396	0.290075	0.211418	0.236316	0.199835	0.246961	0.24055	0.221869	-1.0392	0.272783
	0.0382437	0.0374258	0.0148193	0.0262212	0.0254725	0.0435356	0.0145997	0.0218604	0.0420741	1.88291	0.0171436
delta	1.6643	3.93777	2.43909	2.16502	3.21357	1.87209	5.01373	2.91111	3.0871	1.94433	3.11011
	0.213938	1.57342	0.338046	0.326244	0.704182	0.363418	2.65416	0.485324	1.8077	2.58085	0.666023
lamda	0.0761513	0.172832	0.111375	0.1031	0.153067	0.0912441	0.235195	0.141372	0.110249	0.0895032	0.150369
	0.0116681	0.0704085	0.0150267	0.0168875	0.0345624	0.0200569	0.123673	0.0247834	0.064469	0.156692	0.0319208
epsilon	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706
	0	0	0	0	0	0	0	0	0	0	0
a_crystal	0.524995	0.150677	0.137408	0.271162	0.194672	0.489013	0.154814	0.106238	0.587867	0.896126	0.121904
	0.0852674	0.0377494	0.0440962	0.106247	0.0896036	0.125395	0.0351125	0.0121335	0.12701	0.0884158	0.0189622
chisq	37.5855	38.6585	33.3664	36.1091	28.0814	29.7396	35.114	41.4951	32.8656	21.4503	30.9156
Bin	43	26	33	32	29	32	30	29	28	21	28
MaximumX	9.46043	9.46416	9.45919	9.46064	9.45982	9.45959	9.45941	9.45999	9.46187	9.45824	9.45859

1S_10MeV	6_1	6_2	6_3	6_4	6_5	6_6	6_7	7_0	7_1	7_2	7_3
Et	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455
	0	0	0	0	0	0	0	0	0	0	0
sigma	0.0500843	0.0473122	0.05366	0.0553469	0.0414716	0.0252683	0.0531346	0.0536802	0.0806845	0.0565015	0.0551185
	0.0038774	0.00997728	0.00179501	0.00113224	0.0190859	0.00668657	0.00278102	0.00191535	0.0185396	0.00279337	0.00612665
n	0.254132	0.442937	0.856061	0.590918	20	1.44231	1.48375	0.339814	0.7886	0.80375	0.380445
	0.181816	0.349175	0.184889	0.156511	19.9655	0.965243	0.658283	0.187795	0.787176	0.219516	0.191674
alpha	2.10952	1.75985	1.91518	2.10021	0.0895978	0.356265	1.59674	2.28101	0.564918	1.74689	1.73643
	0.21112	0.358802	0.105679	0.0918492	0.0750264	0.271351	0.178783	0.190882	0.884003	0.21351	0.284869
gamma	0.205859	0.186309	0.0116132	0.00940167	0.234307	0.151884	-0.991244	0.164203	0.243178	0.208237	0.224816
	0.0439862	0.0430521	0.252564	0.166214	0.021184	0.0268375	0.623858	0.0456266	0.01379	0.0274585	0.020313
delta	2.80945	2.05243	141.961	1.27875	2.47493	82.3024	0.745743	1.74797	3.4668	3.95237	2.39954
	0.780168	0.545326	109.702	0.769118	0.410986	100.869	0.285505	0.490011	0.679135	1.76376	0.300317
lamda	0.144515	0.103774	5.19223	0.030205	0.125746	4.40917	0.0138241	0.10465	0.184076	0.230308	0.138261
	0.041814	0.0325946	17.2689	0.0212566	0.0216406	10.668	0.010725	0.0312013	0.0359463	0.105332	0.0198335
epsilon	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706
	0	0	0	0	0	0	0	0	0	0	0
a_crystal	0.468368	0.288717	0.86457	0.897538	0.10222	0.183232	0.880582	0.63477	0.134417	0.482081	0.262744
	0.171229	0.157212	0.0878045	0.040957	0.0170581	0.0371368	0.0292779	0.119271	0.0585757	0.153894	0.107126
chisq	27.8596	16.9807	31.6659	26.5808	51.6933	37.7721	16.9159	47.6453	76.1845	46.2103	63.2609
Bin	28	28	29	28	29	27	23	37	38	37	37
MaximumX	9.45858	9.46104	9.46019	9.46603	9.46011	9.45914	9.47467	9.45803	9.45859	9.45726	9.45856

1S_10MeV	7_4	7_5	7_6	7_7
Et	9.455	9.455	9.455	9.455
	0	0	0	0
sigma	0.0540902	0.0465043	0.0545578	0.033615
	0.00225972	0.0149628	0.00950304	0.0293988
n	0.379633	0.123568	0.404124	20
	0.205791	0.269655	0.372404	12.8994
alpha	2.18521	1.10022	1.3547	0.0555888
	0.148024	0.58995	0.539208	0.0606488
gamma	0.16584	0.187535	0.211028	0.180957
	0.0429816	0.0195727	0.0223925	0.0216329
delta	1.86547	2.34236	2.54852	2.72467
	0.409605	0.356756	0.428265	0.516696
lamda	0.115183	0.136231	0.156432	0.166169
	0.0279253	0.0230079	0.0296599	0.0322726
epsilon	9.4706	9.4706	9.4706	9.4706
	0	0	0	0
a_crystal	0.573633	0.11169	0.175914	0.100379
	0.135028	0.0442043	0.0992047	0.0162621
chisq	52.0618	56.1031	58.1132	44.3912
Bin	39	38	36	35
MaximumX	9.45814	9.46089	9.45871	9.46028

TABLE 2: $\Upsilon(2S)$

2S_10MeV	0_0	0_1	0_2	0_3	0_4	0_5	0_6	0_7	0_8	1_0	1_1
Et	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03
	0	0	0	0	0	0	0	0	0	0	0
sigma	0.0431504	0.0432583	0.042693	0.0421808	0.0410493	0.0415879	0.0405585	0.0425495	0.0409204	0.0434407	0.0431602
	1.08E-05	0.00048331	1.31E-05	0.00069226	0.00049241	0.00040443	0.00072954	0.00048739	0.00083853	1.00E-05	0.00057227
n	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0
alpha	3.08486	3.14191	3.1376	3.03658	3.28431	3.17897	3.12401	3.01052	3.04843	3.33112	3.55249
	0.0652814	0.0336269	0.0010585	0.0451666	0.0480095	0.0385632	0.0450216	0.0388398	0.057985	0.347671	0.103784
gamma	0.721057	0.672115	0.738222	0.617866	1.21048	0.680024	0.837054	0.657496	0.913361	0.973997	1.33515
	0.125162	0.0685898	0.080589	0.101773	0.223911	0.0578043	0.161941	0.0905023	0.154038	0.318561	0.342127
delta	0.922577	0.838586	0.88047	1.01796	0.855318	0.797609	1.00998	1.03499	1.06919	0.746449	0.780589
	0.0779458	0.0428174	0.0356364	0.0869509	0.066512	0.0425889	0.110115	0.0807621	0.166276	0.151663	0.0843903
lamda	0.0418537	0.0379772	0.0418092	0.0504143	0.034387	0.0356949	0.0482565	0.0476648	0.0483178	0.0373486	0.0332723
	0.00518048	0.00322987	0.00315991	0.00667706	0.00406436	0.00314064	0.00769146	0.00577663	0.0109204	0.00968982	0.00643178
epsilon	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366
	0	0	0	0	0	0	0	0	0	0	0
a_crystal	0.75096	0.721855	0.760121	0.736834	0.827955	0.725569	0.772188	0.718573	0.782115	0.810193	0.84919
	0.0350431	0.0251671	0.0238486	0.0443406	0.0232446	0.0216751	0.0419369	0.0367593	0.0339819	0.0364213	0.0257633
R	0.0243345	0.0274591	0.0173022	0.0105012	0.0566179	0.0718431	0.0486294	0.00219844	0.0226634	0.0319623	0.0468311
	0.0100142	0.0184357	0.00842074	0.0124123	0.0182458	0.0369737	0.0224605	0.00428987	0.0174003	0.0127925	0.0209108
B	1.9522	1.82155	2.13385	2.37521	1.83659	1.50599	1.8942	5.16021	2.20262	1.93124	1.92406
	0.234752	0.328857	0.352253	0.631185	0.120744	0.17096	0.190528	10.0786	0.508634	0.285338	0.210999
chisq	59.223	57.7239	57.2987	54.0267	63.7416	48.2564	59.1487	46.7767	58.4144	53.8196	60.2948
Bin	65	67	65	66	65	66	64	62	62	62	60
MaximumX	10.0269	10.0274	10.0272	10.0274	10.0263	10.0277	10.0267	10.0268	10.0262	10.0272	10.0266

2S_10MeV	1_2	1_3	1_4	1_5	1_6	1_7	2_0	2_1	2_2	2_3	2_4
Et	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03
	0	0	0	0	0	0	0	0	0	0	0
sigma	0.0418625	0.0429914	0.0420672	0.0427183	0.0431847	0.0425095	0.0434346	0.0453571	0.0452802	0.0450949	0.043571
	0.00044902	9.90E-06	0.0009254	1.84E-05	0.00076384	0.00067497	0.00020699	0.00087167	0.00084124	5.40E-05	0.00062946
n	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0
alpha	3.19383	3.17454	3.16023	3.19775	3.00017	3.0406	3.06919	3.17587	3.40708	2.90117	2.9072
	0.0678918	0.118007	0.0911642	0.126462	0.0486054	0.0504282	0.0103858	0.098211	0.114423	0.00287788	0.01857
gamma	0.768813	0.681697	0.900831	0.685969	0.50618	0.517031	1.10256	0.695462	0.888483	0.447341	0.466528
	0.100244	0.144755	0.25716	0.170709	0.0661273	0.0593448	0.330332	0.100564	0.177325	0.0362107	0.0409053
delta	0.781919	0.878994	0.983544	0.857586	1.12112	1.07561	1.19533	0.731127	0.715348	1.18817	1.17296
	0.0659165	0.10181	0.191243	0.126295	0.117863	0.109199	0.147628	0.0677753	0.0596822	0.0654867	0.095376
lamda	0.0357842	0.0467606	0.0539645	0.0455764	0.0561059	0.0555572	0.0618291	0.0338096	0.0377763	0.0624564	0.0630654
	0.00485128	0.00673328	0.0144005	0.00885	0.0086559	0.00820871	0.00722297	0.00445398	0.00533186	0.00528045	0.00764481
epsilon	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366
	0	0	0	0	0	0	0	0	0	0	0
a_crystal	0.751167	0.745189	0.82038	0.777508	0.67211	0.690941	0.838462	0.723249	0.789	0.642309	0.646018
	0.0297709	0.0438397	0.0461808	0.0466616	0.0435313	0.0356982	0.0395553	0.031449	0.0355093	0.0287949	0.0325668
R	0.193946	0.012702	0.0590167	0.0288638	0.00096211	0.00085157	0.0667499	0.0648581	0.0391613	0.0126164	0.0676609
	0.0906709	0.0142409	0.0395968	0.0272069	0.00209307	0.00144768	0.0129331	0.0544544	0.0234724	0.0199079	0.0555263
B	1.42652	2.20256	1.74629	1.77387	19.9964	19.9997	1.90925	1.55705	1.95206	1.90318	1.49915
	0.124092	0.931889	0.287496	0.600385	12.6888	13.4752	0.101303	0.263776	0.278198	0.768768	0.22686
chisq	41.055	51.6413	69.0625	44.2405	67.3314	65.2356	72.6467	72.4215	63.1134	58.8979	90.0363
Bin	63	61	61	57	59	56	65	65	64	67	65
MaximumX	10.0272	10.0275	10.0274	10.0278	10.0273	10.0276	10.0266	10.0276	10.0272	10.0273	10.0272

2S_10MeV	2_5	2_6	2_7	3_0	3_1	3_2	3_3	3_4	3_5	3_6	3_7
Et	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03	10.03
	0	0	0	0	0	0	0	0	0	0	0
sigma	0.0449813	0.0445685	0.0426519	0.0493109	0.0509833	0.0526345	0.0518424	0.0505866	0.0502951	0.0486928	0.0543039
	0.00059694	6.85E-06	0.00139896	0.00043835	6.91E-05	0.00070015	0.00044135	0.00013703	0.00074098	0.00077447	8.49E-06
n	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0
alpha	3.45043	3.118	3.05095	3.58567	3.33339	3.0547	2.93578	2.96419	2.94387	2.83088	2.92472
	0.104736	0.00027074	0.0785518	0.0563691	0.00466063	0.0783245	0.0157195	0.0044835	0.0395919	0.0182104	0.0003197
gamma	0.996248	0.663236	0.451967	1.60253	0.89035	0.545657	0.645563	0.624237	0.45176	0.435629	1.95655
	0.245627	0.0978163	0.0928704	0.273508	0.229974	0.068322	0.097571	0.17969	0.0654716	0.0517512	0.450649
delta	0.733418	0.913502	1.06894	0.720789	0.745497	0.891769	90.484	1.81101	1.59957	116.793	107.351
	0.0652852	0.0566886	0.157108	0.064171	0.0656017	0.118092	126.29	0.429801	0.272253	141.649	153.676
lamda	0.0365413	0.0473141	0.0605859	0.0297523	0.0521833	0.0533204	8.72966	0.151316	0.113981	9.73996	7.89166
	0.00543114	0.00534386	0.0134318	0.00588505	0.00817993	0.00960858	12.6347	0.0445616	0.0265722	14.0528	13.9124
epsilon	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366	10.0366
	0	0	0	0	0	0	0	0	0	0	0
a_crystal	0.809794	0.754223	0.723454	0.874101	0.841967	0.732612	0.878065	0.816483	0.719901	0.722587	0.953678
	0.0373696	0.0344399	0.064346	0.0123589	0.0322211	0.029238	0.0162424	0.0555961	0.0521205	0.0319562	0.00788406
R	0.0325894	0.0305214	0.139406	0.0571149	0.0420094	0.00275824	0.00446565	0.0401591	0.00676674	0.00861737	0.0154991
	0.0164171	0.015327	0.137089	0.0118522	0.0113236	0.00151894	0.00123538	0.0205826	0.00223266	0.0041935	0.00535289
B	2.18864	2.01228	1.44835	2.14223	2.06186	20	20	2.04125	19.9993	18.8192	4.23468
	0.354937	0.335782	0.267828	0.125858	0.222769	9.74546	10.5285	0.24061	13.631	15.6603	2.22824
chisq	74.1556	67.6876	54.3778	201.689	160.385	243.752	153.181	251.178	224.806	266.49	147.41
Bin	60	58	44	68	67	66	65	63	62	62	45
MaximumX	10.0269	10.0273	10.0282	10.0257	10.0276	10.0272	10.0289	10.028	10.0273	10.0278	10.0295

TABLE 3: $\Upsilon(3S)$

3S_10MeV	0_0	0_1	0_2	0_3	0_4	0_5	0_6	0_7	1_0	1_1	1_2
Et	10.24	10.24	10.24	10.24	10.24	10.24	10.24	10.24	10.24	10.24	10.24
	0	0	0	0	0	0	0	0	0	0	0
sigma	0.131138	0.123773	0.117747	0.00747787	0.104525	0.101703	0.132745	0.0936172	0.135424	0.121338	0.144584
	0.0074282	0.0109735	0.0107926	0.00276833	0.0152727	0.0186231	0.00928483	0.0191259	0.0146831	0.0120812	0.00986487
n	0.656961	0.903889	2.74543	1.78474	1.06476	1.29653	0.52392	0.798513	0.715752	0.718482	0.348573
	0.213463	0.351665	1.79132	0.599531	0.383493	0.449781	0.278413	0.231222	0.397996	0.287417	0.188325
alpha	0.95651	0.788834	0.598201	0.0441861	0.79143	0.674293	1.09177	0.743729	0.781603	0.801022	1.04583
	0.120243	0.154453	0.12412	0.0187703	0.181047	0.162843	0.19587	0.137441	0.195729	0.146754	0.158934
gamma	0.139911	0.134121	0.148165	0.167034	0.144831	0.145093	0.150428	0.13673	0.137796	0.1254	0.137644
	0.00732875	0.00777999	0.00779153	0.00662949	0.00873285	0.00934391	0.0081943	0.0113123	0.00956701	0.00956207	0.00897168
delta	3.09886	2.72556	2.94553	2.9253	2.93727	3.04273	2.82449	2.70925	2.85894	2.60603	2.83108
	0.255701	0.203882	0.24794	0.219043	0.235309	0.264345	0.226452	0.217971	0.275097	0.204525	0.246906
lamda	0.140281	0.123301	0.131565	0.132121	0.131287	0.135472	0.122516	0.118729	0.132387	0.119841	0.130703
	0.0120068	0.00950341	0.0113665	0.0103171	0.0108288	0.012077	0.0102117	0.0100492	0.0131705	0.00985965	0.011916
epsilon	10.365	10.365	10.365	10.365	10.365	10.365	10.365	10.365	10.365	10.365	10.365
	0	0	0	0	0	0	0	0	0	0	0
a_crystal	0.107136	0.100943	0.0985144	0.0623782	0.0958815	0.0950568	0.104128	0.0981458	0.0984659	0.097258	0.0997721
	0.00356066	0.00470151	0.00482795	0.00212745	0.0055398	0.00631323	0.00421651	0.00593366	0.00524442	0.00469695	0.0041718
chisq	53.0012	64.0797	58.6867	52.5547	65.6288	58.6813	52.0823	51.5601	61.846	61.4986	72.3201
Bin	61	60	59	58	60	57	51	47	49	50	49
MaximumX	10.359	10.3594	10.3588	10.3582	10.3588	10.3588	10.359	10.3594	10.3591	10.3597	10.3591

3S_10MeV	1_3	1_4	1_5	1_6	1_7	2_0	2_1	2_2	2_3	2_4	2_5
Et	10.24	10.24	10.24	10.24	10.24	10.24	10.24	10.24	10.24	10.24	10.24
	0	0	0	0	0	0	0	0	0	0	0
sigma	0.1355	0.115475	0.138047	0.098614	0.101724	0.136706	0.142415	0.13303	0.137282	0.138962	0.136929
	0.00777053	0.0124587	0.0170835	0.01426	0.0186551	0.00625259	0.00647423	0.00678506	0.00733995	0.00667998	0.00806886
n	0.316243	0.660078	1.0448	0.519395	0.529886	1.03578	1.76539	0.62725	1.70662	0.823569	1.03411
	0.16239	0.255489	0.850135	0.16561	0.280767	0.406632	0.929997	0.1752	0.712712	0.289151	0.425813
alpha	1.21663	0.921276	0.66769	0.885918	1.02479	0.810448	0.714581	0.945742	0.739111	0.891361	0.786458
	0.133004	0.140823	0.235871	0.137982	0.233176	0.132018	0.125731	0.0988468	0.104586	0.1206	0.132844
gamma	0.125387	0.125095	0.148638	0.129253	0.114696	0.141993	0.142035	0.130291	0.136363	0.130804	0.151513
	0.00907716	0.010588	0.0101297	0.0112898	0.0119003	0.0063107	0.00621479	0.00665381	0.00663305	0.00671126	0.00715125
delta	3.93795	2.59407	2.45138	2.75859	3.18967	2.90908	2.77575	3.04516	2.65219	3.03632	3.00422
	0.599715	0.204897	0.218375	0.246443	0.349977	0.213286	0.191898	0.221001	0.166505	0.232159	0.241001
lamda	0.18012	0.11594	0.110163	0.124767	0.143213	0.13995	0.131425	0.147019	0.126408	0.14595	0.142411
	0.0282423	0.00962088	0.0101196	0.0116444	0.016266	0.0105831	0.0093502	0.0110422	0.00822789	0.0115364	0.0117616
epsilon	10.365	10.365	10.365	10.365	10.365	10.365	10.365	10.365	10.365	10.365	10.365
	0	0	0	0	0	0	0	0	0	0	0
a_crystal	0.10957	0.101056	0.0993347	0.102279	0.101364	0.105068	0.107198	0.104678	0.103087	0.104965	0.102113
	0.00393668	0.00473124	0.00706751	0.00492753	0.00620767	0.00368526	0.00388922	0.00315267	0.00371206	0.0034395	0.00382723
chisq	52.4905	44.2343	45.0259	38.2215	51.7069	73.2499	71.1438	87.1666	69.6056	82.9732	74.3465
Bin	46	45	47	46	42	75	76	74	73	73	69
MaximumX	10.3593	10.3598	10.3591	10.3594	10.3599	10.3586	10.3588	10.359	10.359	10.359	10.3583

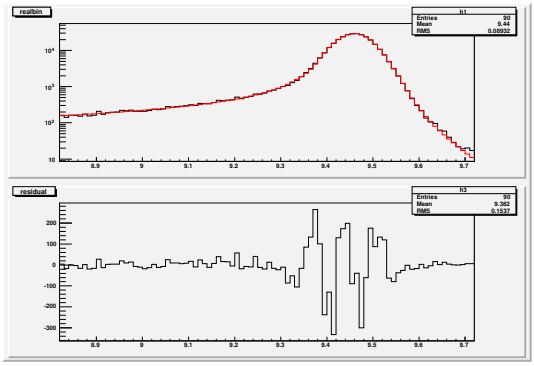
3S_10MeV	2_6	2_7	3_0	3_1	3_2	3_3	3_4	3_5	3_6	3_7
Et	10.24	10.24	10.24	10.24	10.24	10.24	10.24	10.24	10.24	10.24
	0	0	0	0	0	0	0	0	0	0
sigma	0.105113	0.12569	0.160193	0.170715	0.157542	0.163827	0.099753	0.179572	0.136496	0.186843
	0.0124535	0.00898073	0.00705072	0.00920554	0.00618268	0.00983351	0.0137841	0.00647636	0.0149218	0.0127427
n	0.703579	0.598036	2.08632	0.700243	0.189268	7.84182	0.701684	0.00378975	0.549241	4.82E-08
	0.175133	0.216442	1.46607	0.357988	0.134133	12.9053	0.185409	0.0983485	0.241934	0.04303
alpha	0.905751	1.10469	0.698835	0.918181	1.32613	0.586246	0.889396	1.50944	0.755058	1.40911
	0.111604	0.12599	0.123473	0.14828	0.134311	0.121162	0.12945	0.112427	0.158471	0.0804943
gamma	0.117731	0.109965	0.0997866	0.113455	0.0904699	0.11033	0.0467575	0.120002	0.094372	0.114558
	0.00954408	0.00836495	0.00651669	0.00669588	0.00681714	0.00707409	0.0098548	0.00745061	0.00897148	0.009693
delta	2.5706	2.82842	3.84161	3.48808	3.17137	2.78256	2.62309	2.94113	2.55212	2.68685
	0.158173	0.213022	0.464436	0.350287	0.245917	0.21695	0.152266	0.241429	0.186486	0.236648
lamda	0.120244	0.133273	0.21284	0.189919	0.173007	0.151436	0.143189	0.161603	0.141386	0.154544
	0.00776173	0.0105033	0.0262163	0.0195051	0.0139014	0.0120662	0.00875042	0.0137328	0.0106717	0.0141384
epsilon	10.365	10.365	10.365	10.365	10.365	10.365	10.365	10.365	10.365	10.365
	0	0	0	0	0	0	0	0	0	0
a_crystal	0.102245	0.0909992	0.108912	0.103496	0.11232	0.104106	0.094818	0.123254	0.0878408	0.108632
	0.00420835	0.00346863	0.00387648	0.00353897	0.0033016	0.00461743	0.00429607	0.00376248	0.00529634	0.00489767
chisq	67.8845	58.3913	256.797	243.691	231.161	238.57	190.803	189.973	165.869	121.668
Bin	63	51	76	73	76	69	69	71	56	51
MaximumX	10.3598	10.3601	10.3595	10.359	10.3601	10.3594	10.3621	10.3588	10.3601	10.3589

TABLE 4: Parameters for angle-integrated pt bins

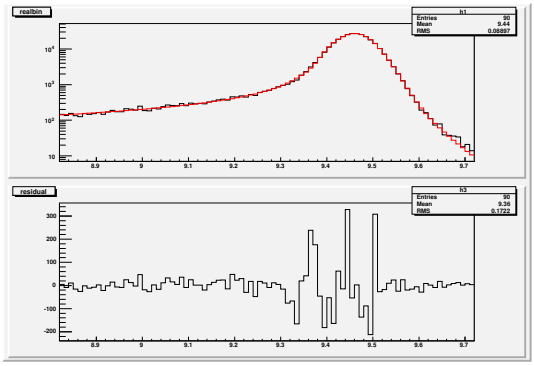
1S_10MeV	pt_0	pt_1	pt_2	pt_3	pt_4	pt_5	pt_6	pt_7
Et	9.455	9.455	9.455	9.455	9.455	9.455	9.455	9.455
	0	0	0	0	0	0	0	0
sigma	0.0411922	0.0417284	0.0420343	0.0425716	0.0455928	0.0831717	0.0863095	0.0564572
	0.00023169	0.0002686	0.00026501	0.00038451	0.00045105	0.00320212	0.00340639	0.00113931
n	0.961239	0.996901	1.01112	0.994826	0.915354	2.19683	3.28123	0.976207
	0.0231309	0.0238562	0.0213303	0.0298568	0.0317881	0.764054	1.71469	0.0604198
alpha	1.83529	1.80615	1.78561	1.79183	1.8387	0.528178	0.455861	1.84212
	0.0147152	0.0157288	0.0144254	0.0193263	0.0224807	0.13589	0.113523	0.0495863
gamma	0.125697	0.10986	0.102297	0.0856344	0.125295	0.257087	0.253507	0.145859
	0.0197401	0.0203579	0.0159247	0.0207283	0.0183133	0.0053341	0.00575364	0.0188484
delta	1.4022	1.4434	1.51871	1.5596	1.45923	3.23262	3.18451	1.80295
	0.0478093	0.0520427	0.044157	0.0604725	0.0656859	0.292419	0.31949	0.185264
lamda	0.0533452	0.0557802	0.0609806	0.0665943	0.0630328	0.148433	0.155406	0.10746
	0.00258642	0.00286524	0.00247154	0.00354603	0.00395261	0.0130847	0.0153146	0.0135122
epsilon	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706	9.4706
	0	0	0	0	0	0	0	0
a_crystal	0.702284	0.687711	0.653872	0.652677	0.639265	0.155396	0.146831	0.610775
	0.020533	0.021865	0.0190031	0.0239443	0.0272383	0.0179159	0.0165612	0.0510626
chisq	118.329	145.669	152.771	110.114	118.636	72.2151	77.9992	128.419
Bin	78	78	77	80	82	83	80	83
MaximumX	9.46073	9.46103	9.46115	9.46078	9.46109	9.45973	9.45925	9.45885

2S_10MeV	pt_0	pt_1	pt_2	pt_3
Et	10.03	10.03	10.03	10.03
	0	0	0	0
sigma	0.0418525	0.0424482	0.0439852	0.0495952
	4.12E-06	4.39E-06	3.91E-06	4.41E-06
n	0	0	0	0
	0	0	0	0
alpha	3.0814	3.14908	3.07781	2.89692
	0.00014987	0.00018104	0.00015891	0.00026493
gamma	0.690706	0.682025	0.661013	0.442158
	0.0149185	0.0154209	0.0184667	0.0137433
delta	0.928447	0.882559	0.92875	1.39973
	0.0130275	0.019512	0.0223082	0.0699951
lamda	0.0435133	0.045313	0.0500297	0.101467
	0.00110476	0.00188723	0.00216033	0.00743588
epsilon	10.0366	10.0366	10.0366	10.0366
	0	0	0	0
a_crystal	0.738343	0.75891	0.754037	0.732545
	0.00592092	0.00661158	0.00787237	0.0103607
R	0.0374741	0.065833	0.0779581	0.0614813
	0.00484135	0.0121667	0.0103158	0.0131848
B	1.75212	1.52956	1.63025	1.59119
	0.0650247	0.0643942	0.0542262	0.068585
chisq	89.9629	89.6683	106.403	186.721
Bin	66	66	67	71
MaximumX	10.0271	10.0275	10.0273	10.0277

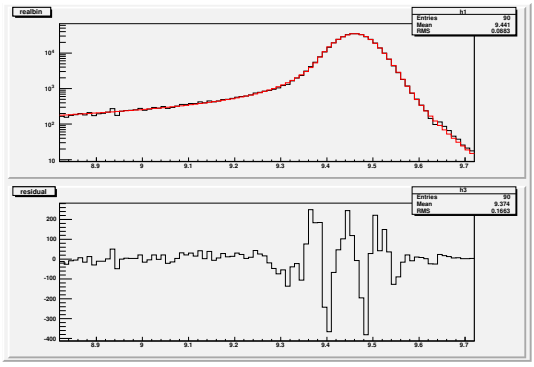
3S_10MeV	pt_0	pt_1	pt_2	pt_3
Et	10.24	10.24	10.24	10.24
	0	0	0	0
sigma	0.133286	0.135607	0.143584	0.161836
	0.0015994	0.00212532	0.00145849	0.00166898
n	0.97296	1.03573	0.921574	1.26647
	0.120702	0.167906	0.117788	0.238968
alpha	0.855101	0.853179	0.885276	0.798778
	0.0406802	0.0521789	0.0403668	0.046505
gamma	0.151357	0.130448	0.135035	0.0982616
	0.00239141	0.00286726	0.002232	0.00245412
delta	3.07041	2.83961	2.92058	3.01238
	0.0859653	0.0858293	0.0715056	0.0876093
lamda	0.136308	0.128986	0.139006	0.166132
	0.00396264	0.00406134	0.00353769	0.00498217
epsilon	10.365	10.365	10.365	10.365
	0	0	0	0
a_crystal	0.106717	0.102195	0.105698	0.109904
	0.00115661	0.00143291	0.0011236	0.00139285
chisq	123.479	99.5111	119.033	269.086
Bin	82	83	83	87
MaximumX	10.3587	10.3595	10.359	10.3598



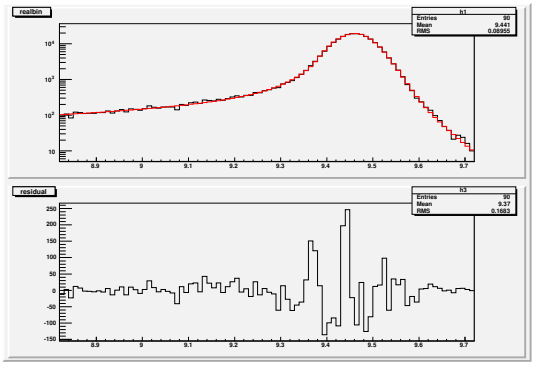
(a) 1S_0(Fit Status = 3, chisq/dof =118.329 /71)



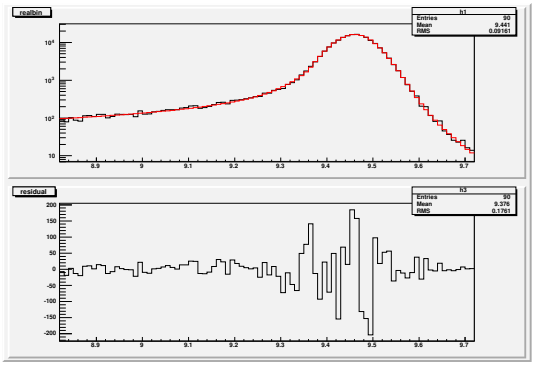
(b) 1S_1(Fit Status = 3, chisq/dof =145.669 /71)



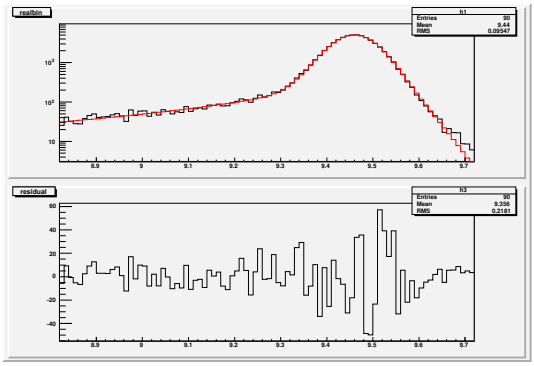
(c) 1S_2(Fit Status = 3, chisq/dof = 152.771/70)



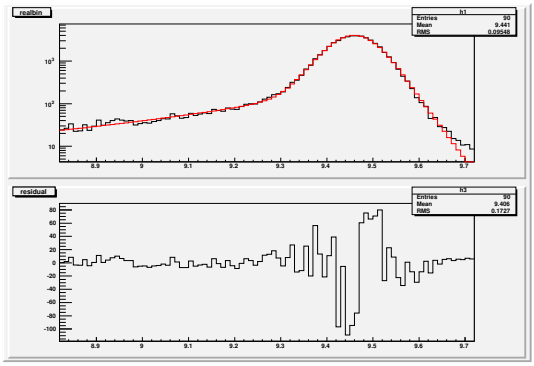
(d) 1S_3(Fit Status = 3, chisq/dof = 110.114/73)



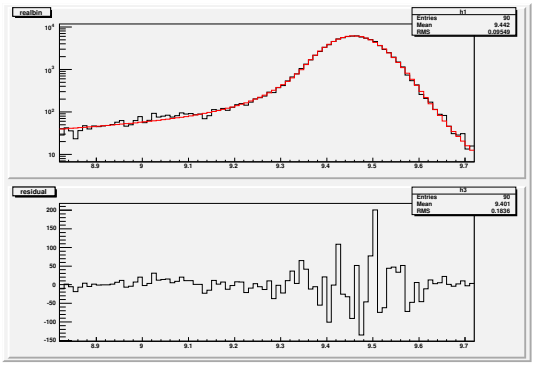
(e) 1S_4(Fit Status = 3, chisq/dof = 118.636/75)



(f) 1S_5(Fit Status = 3, chisq/dof = 72.2151/76)

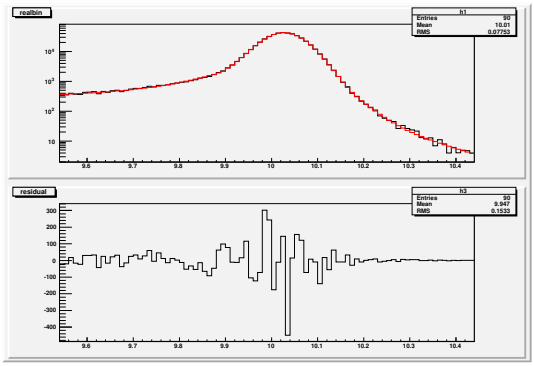


(g) 1S_6(Fit Status = 3, chisq/dof = 77.9992/73)

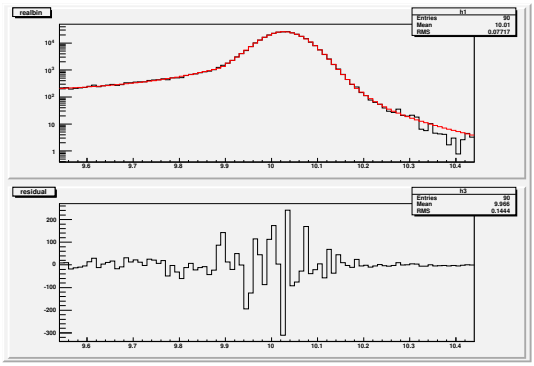


(h) 1S_7(Fit Status = 3, chisq/dof = 128.419/76)

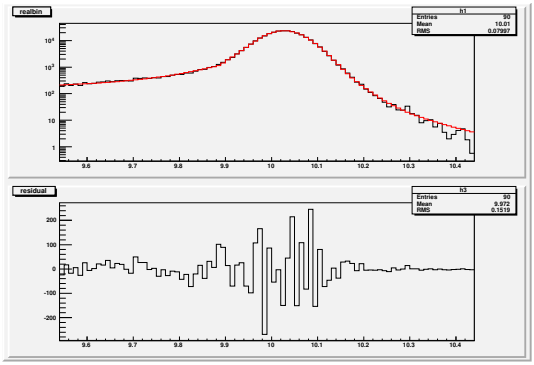
Figure D.3: Angle-integrated 1S pt bins



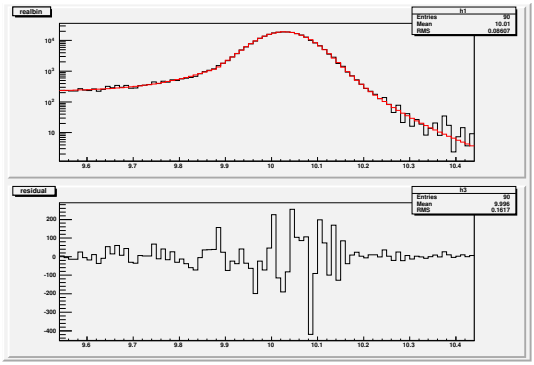
(a) 2S_0(Fit Status = 3, chisq/dof = 89.9629/58)



(b) 2S_1(Fit Status = 3, chisq/dof = 89.6683/58)

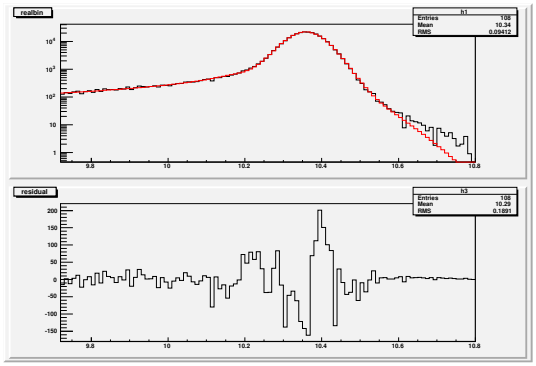


(c) 2S_2(Fit Status = 3, chisq/dof = 106.403/59)

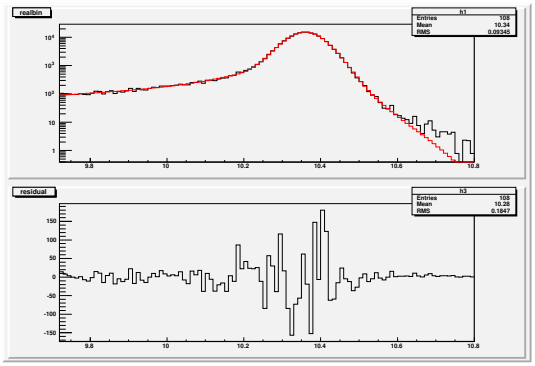


(d) 2S_3(Fit Status = 3, chisq/dof = 186.721/63)

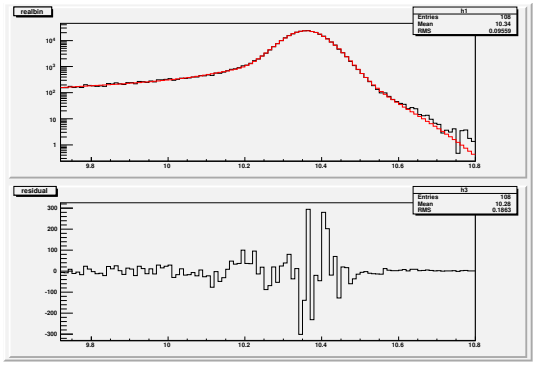
Figure D.4: Angle-integrated 2S pt bins



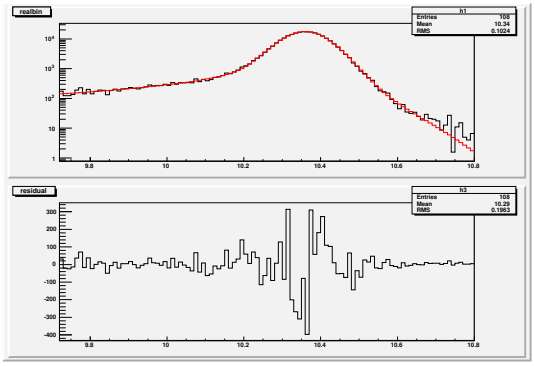
(a) 3S_0(Fit Status = 3, chisq/dof = 123.479/75)



(b) 3S_1(Fit Status = 3, chisq/dof = 99.5111/76)



(c) 3S_2(Fit Status = 3, chisq/dof = 119.033/76)



(d) 3S_3(Fit Status = 3, chisq/dof = 269.086/80)

Figure D.5: Angle-integrated 3S pt bins

Bibliography

[1] Mark Gerald Aartsen, R Abbasi, Y Abdou, M Ackermann, J Adams, JA Aguilar, M Ahlers, D Altmann, J Auffenberg, X Bai, et al. First observation of pev-energy neutrinos with icecube. *Physical Review Letters*, 111(2):021103, 2013.

[2] R Abbasi, Yasser Abdou, T Abu-Zayyad, J Adams, JA Aguilar, M Ahlers, K Andeen, J Auffenberg, X Bai, M Baker, et al. Constraints on the extremely-high energy cosmic neutrino flux with the icecube 2008-2009 data. *Physical Review D*, 83(9):092003, 2011.

[3] J Abraham, P Abreu, M Aglietta, EJ Ahn, D Allard, J Allen, J Alvarez-Muñiz, M Ambrosio, L Anchordoqui, S Andringa, et al. Measurement of the energy spectrum of cosmic rays above 10¡ sup¿ 18¡/sup¿ ev using the pierre auger observatory. *Physics Letters B*, 685(4):239–246, 2010.

[4] J Abraham, M Aglietta, C Aguirre, D Allard, I Allekotte, P Allison, C Alvarez, J Alvarez-Muniz, M Ambrosio, L Anchordoqui, et al. Anisotropy studies around the galactic centre at eev energies with the auger observatory. *Astroparticle Physics*, 27(4):244–253, 2007.

[5] Pedro Abreu, M Aglietta, M Ahlers, EJ Ahn, IFM Albuquerque, D Allard, I Allekotte, J Allen, P Allison, A Almela, et al. Search for point-like sources of ultra-high energy

neutrinos at the pierre auger observatory and improved limit on the diffuse flux of tau neutrinos. *The Astrophysical Journal Letters*, 755(1):L4, 2012.

[6] DV Ahluwalia, CA Ortiz, and GZ Adunas. Robust flavor equalization of cosmic neutrino flux by quasi bi-maximal mixing. *arXiv preprint hep-ph/0006092*, 2000.

[7] Eun-Joo Ahn, Ralph Engel, Thomas K Gaisser, Paolo Lipari, and Todor Stanev. Cosmic ray interaction event generator sibyll 2.1. *Physical Review D*, 80(9):094003, 2009.

[8] Patrick Allison, Jan Auffenberg, Robert Bard, JJ Beatty, DZ Besson, S Böser, C Chen, Pisin Chen, Amy Connolly, Jonathan Davies, et al. Design and initial performance of the askaryan radio array prototype eev neutrino detector at the south pole. *Astroparticle physics*, 35(7):457–477, 2012.

[9] C Aramo, A Insolia, A Leonardi, G Miele, L Perrone, O Pisanti, and DV Semikoz. Earth-skimming uhe tau neutrinos at the fluorescence detector of pierre auger observatory. *Astroparticle Physics*, 23(1):65–77, 2005.

[10] Steven W Barwick. Arianna: A new concept for uhe neutrino detection. In *Journal of Physics: Conference Series*, volume 60, page 276. IOP Publishing, 2007.

[11] Kristin P Bennett and Erin J Bredensteiner. Duality and geometry in svm classifiers. In *ICML*, pages 57–64, 2000.

[12] Pasquale Blasi. Theoretical challenges in acceleration and transport of ultra high energy cosmic rays: A review. *arXiv preprint arXiv:1208.1682*, 2012.

[13] Pierre Auger Collaboration. Ultrahigh energy neutrinos at the pierre auger observatory. *arXiv preprint arXiv:1304.1630*, 2013.

[14] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[15] HJ Drescher, M Hladik, S Ostapchenko, T Pierog, and Klaus Werner. Parton-based gribov–regge theory. *Physics Reports*, 350(2):93–289, 2001.

[16] Sharada Iyer Dutta, Yiwen Huang, and Mary Hall Reno. Tau neutrino propagation and tau energy loss. *Physical Review D*, 72(1):013005, 2005.

[17] Ralph Engel, Dieter Heck, and Tanguy Pierog. Extensive air showers and hadronic interactions at high energy. *Annual Review of Nuclear and Particle Science*, 61:467–489, 2011.

[18] Daniele Fargion. Discovering ultra-high-energy neutrinos through horizontal and upward τ air showers: evidence in terrestrial gamma flashes? *The Astrophysical Journal*, 570(2):909, 2002.

[19] George M Frichter, Douglas W McKay, and John P Ralston. Prediction for the ultra-high energy neutrino-nucleon cross section from new structure function data at small x. *Physical review letters*, 74(9):1508, 1995.

[20] Thomas K Gaisser and Todor Stanev. Neutrinos and cosmic rays. *Astroparticle Physics*, 2012.

[21] Raj Gandhi, Chris Quigg, Mary Hall Reno, and Ina Sarcevic. Neutrino interactions at ultrahigh energies. *Physical Review D*, 58(9):093009, 1998.

[22] PW Gorham, P Allison, BM Baughman, JJ Beatty, K Belov, DZ Besson, S Bevan, WR Binns, C Chen, P Chen, et al. Observational constraints on the ultrahigh energy cosmic neutrino flux from the second flight of the anita experiment. *Physical Review D*, 82(2):022004, 2010.

[23] Francis Halzen and Spencer R Klein. Astronomy and astrophysics with neutrinos. *Physics Today*, 61(5):29, 2008.

[24] Dieter Heck, J Knapp, JN Capdevielle, G Schatz, T Thouw, et al. *CORSIKA: A Monte Carlo code to simulate extensive air showers*, volume 6019. FZKA, 1998.

[25] David W Hosmer Jr and Stanley Lemeshow. *Applied logistic regression*. John Wiley & Sons, 2004.

[26] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*. Wiley. com, 2013.

[27] M Iori, A Sergi, D Fargion, M Gallinaro, and M Kaya. Study of a detector array for upward tau air-showers. *arXiv preprint astro-ph/0602108*, 2006.

[28] J Jones, I Mocioiu, MH Reno, and I Sarcevic. Tracing very high energy neutrinos from cosmological distances in ice. *Physical Review D*, 69(3):033004, 2004.

[29] NN Kalmykov, SS Ostapchenko, and AI Pavlov. Quark-gluon-string model and eas simulation problems at ultra-high energies. *Nuclear Physics B-Proceedings Supplements*, 52(3):17–28, 1997.

[30] Spencer R Klein et al. Arianna: A radio detector array for cosmic neutrinos on the ross ice shelf. *arXiv preprint arXiv:1207.3846*, 2012.

[31] Johannes Knapp and Dieter Heck. *Extensive Air Shower Simulation with CORSIKA: A User's Manual*. Kernforschungszentrum Karlsruhe, 1993.

[32] Johannes Knapp, Dieter Heck, and Gerd Schatz. *Comparison of hadronic interaction models used in air shower simulations and of their influence on shower development and observables*, volume 5828. FZKA, 1996.

[33] AV Olinto. Ultra high energy cosmic rays: the theoretical challenge. *Physics Reports*, 333:329–348, 2000.

[34] Esteban Roulet, Guenter Sigl, Arjen van Vliet, and Silvia Mollerach. Pev neutrinos from the propagation of ultra-high energy cosmic rays. *Journal of Cosmology and Astroparticle Physics*, 2013(01):028, 2013.

[35] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical Programming*, 127(1):3–30, 2011.

[36] Günter Sigl. Cosmic rays and neutrino interactions beyond the standard model. *Nuclear Physics B-Proceedings Supplements*, 87(1):439–441, 2000.

[37] RM Sternheimer, MJ Berger, and Stephen M Seltzer. Density effect for the ionization loss of charged particles in various substances. *Atomic Data and Nuclear Data Tables*, 30(2):261–271, 1984.

[38] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[39] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[40] V Van Elewyck. High-energy neutrino astronomy: Status and prospects for cosmic-ray physics. *arXiv preprint arXiv:1209.3425*, 2012.

[41] Klaus Werner. Strings, pomerons and the venus model of hadronic interactions at ultrarelativistic energies. *Physics Reports*, 232(2):87–299, 1993.

[42] Georgi T Zatsepin and Vadem A Kuz'min. Upper limit of the spectrum of cosmic rays. *JETP Lett.(USSR)(Engl. Transl.)*, 4, 1966.