Spring 2003

Study of energy sampling weights in the DØ detector using multiparameter fitting method

Qun Yu
Louisiana Tech University

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

STUDY OF ENERGY SAMPLING WEIGHTS IN THE DØ DETECTOR USING MULTIPARAMETER FITTING METHOD

by

Qun Yu, B.S.

A Dissertation Presented in Partial Fulfillment
Of the Requirements for the Degree
Doctor of Philosophy

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

May 2003

LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

May 14,2003
Date

We hereby recommend that the thesis/dissertation prepared under our supervision by Qun Yu entitled Study of Energy Sampling Weights in the D0 Detector Using Multiparameter Fitting Method be accepted in partial fulfillment of the requirements for the Degree of Ph.D in Computational Analysis and Modeling

Supervisor of Thesis/Dissertation Research

Head of Department

Department

Recommendation concurred in:

Advisory Committee

Approved:
Director of Graduate Studies

Approved:
Dean of the Graduate School

Dean of the College

GS Form 13
(4/02)

ABSTRACT

The DØ calorimeter at Fermilab is a sampling calorimeter measuring the energy of particles produced in high energy proton-antiproton collisions. A set of accurate sampling weights is of significant importance to DØ research activity. The objective of this work was to obtain a set of optimized sampling weights for the DØ central calorimeter, the Inter-Cryostat Detector (ICD), the Central Calorimeter Massless Gap (CCMG), and the End Calorimeter Massless Gap (ECMG).

The foundation of the optimization method is that, in high energy physics, the ratio of energy E and the corresponding momentum P of a particle is approximately 1, in units where speed of light is c=1. The energy distributions in different layers of the calorimeter is different, there are also differences among different calorimeter channels. A computational model, based on those differences, was formulated to calculate the energy E, and the momentum P was obtained through the detectors' precision tracker measurements. The Chi-square minimization was performed between E/P and 1 using the minimization software package "TMiuit" from the European Organization for Nuclear Research (CERN).

The result of this minimization was a set of energy sampling weights for electrons, which verifies that the existing sampling weights for the CC region are optimized. For ICD, CCMG and ECMG regions, more work will be needed due to the lack of data in those regions.

APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Thesis/Dissertation. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Thesis/Dissertation. Further, any portions of the Thesis/Dissertation used in books, papers, and other works must be appropriately referenced to this Thesis/Dissertation.

Finally, the author of this Thesis/Dissertation reserves the right to publish freely, in the literature, at any time, any or all portions of this Thesis/Dissertation.

Author _____

Date 05/15/2003

GS Form 14
(4/03)

TABLE OF CONTENTS

LIST OF TABLES

PAGE

LIST OF FIGURES

PAGE

ACKNOWLEDGMENTS

I would like to express my deep, sincere gratitude to my advisor, Dr. Lee Sawyer, and my former advisor, Dr. Zeno Greenwood, for their detailed advice, continuous guidance, and encouragement without which I can never finish this dissertation. I also appreciate the time and support of other advisory committee members: Dr. Neeti Parashar, Dr. George Butler, and Dr. Summet Dua.

I dedicate this dissertation to my parents and my wife for their ever-lasting encouragement, love and patience.

CHAPTER ONE

INTRODUCTION

1.1 The Standard Model

Particle physics is the study of the basic elements of matter and the forces acting among them. It aims to determine the fundamental laws that control the make-up of matter and the physical universe. Particle physics seeks to answer two questions:

1. What are the fundamental building blocks from which all matter is made?
2. What are the interactions between them that govern how they combine and decay?

The Standard Model is a picture that describes how different elementary particles are organized and how they interact with each other and with different forces. The elementary particles are split into two families, namely the quarks and the leptons. Each of these families consists of six particles, and each family splits into three generations, with the first generation being the lightest, and the third the heaviest. Furthermore, there are four different force-carrying particles which lead to the interactions between particles. The following table describes the structure of the Standard Model.

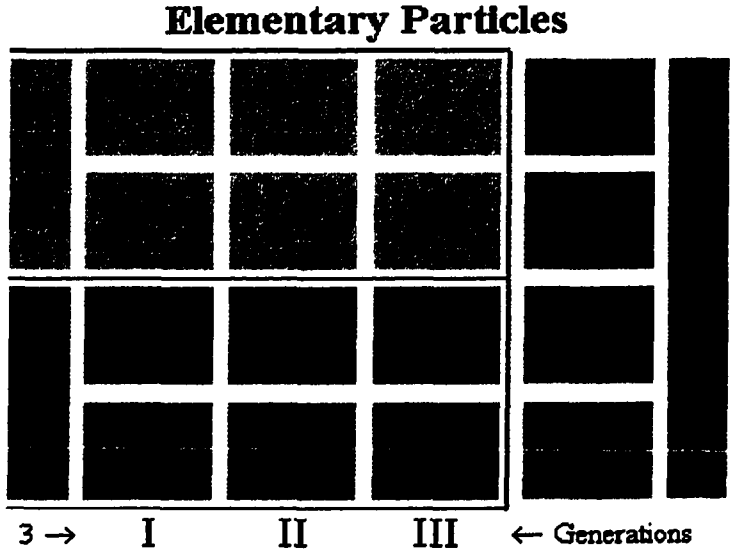


Figure 1.Standard Model of Particles

It is known that every elementary particle has a corresponding particle called anti-particle with opposite electrical charge and magnetic moment [1]. There are six different types of Quarks called "flavors": up, down, charmed, strange, top, and bottom, each of which has three different kinds of charge. Each type of quark has its corresponding anti-quark, thus giving us a total number of 36 quarks. Leptons also come in six flavors: electron, muon tau, electron neutrino, muon neutrino, and tau neutrino. Together with their corresponding anti-particles, there are a total of 12 leptons.

Quarks have not been found to exist by themselves; they are always arranged to form certain composite particles called "hadrons" [2]. There are two types of hadrons: baryons, which are composed of three quarks, and mesons which are made up of a quark and an anti-quark.

Unlike quarks, leptons can exist by themselves. Among them, the electron is a stable elementary particle which has a negative charge of 1.602e-19 coulombs and a rest mass of 9.11e-31kg, and a spin of ½ [3]; The muon is an unstable particle with a mass 207

times that of an electron and a mean lifetime of about 2.2 x 10^{-6} [4]; The neutrino is a stable uncharged fundamental particle with no mass (or extremely light) and with a spin of ½. It has a very weak interaction with matter[5].

The Standard Model of particles not only provides a framework to explain both the low energy process (chemistry and nuclear physics) and the high energy process (sub-nuclear physics), it is also used to predict new particles. Charm, bottom, top quarks, tau neutrino, W, Z boson and gluon were predicted by the Standard Model before they were observed experimentally [6].

To better understand the sub-nuclear structure, we need some means to "look" at the target particles. Microscope will not work because the wavelength is longer than the size of particles; instead, physicists use particles to observe particles. Quantum mechanics tells us that particles also possess wave-like properties and that the wavelength shrinks as the energy grows. The higher of particle energy, the smaller structure we can "see". Thus, we need very high energy particles (or waves) to resolve very small structures.

With higher energy we can obtain more detailed information of the particle structure. When two charged particles come closer, they bounce off from each other due to the electric field, and the closer they approach to each other, the more detailed information of the structure can be indicated by the deflection [7]. High energy also allows new particles to be produced.

Accelerators are built and used for the purpose of generating high energy, high speed particles such as protons and antiprotons for the research of high energy physics. Accelerators usually utilize the electric field to speed up positively charged particles in the direction of the field, and negatively charged particles in the opposite direction of

the field. High energy particles collide inside a device called a collider. The Accelerator-Collider is the basic method to produce sub-nuclear particles and (maybe) new particles. The after-collision information is obtained through complex detectors. The part of the detector used to measure the energies of the particles produced in the collision is called a calorimeter. Calorimetry plays a very important role in the particle physics research.

1.2 Calorimetry Principles

When a high-energy electron passes through a material with a high atomic number, the primary mechanism by which it loses energy is that a charged particle interacts with the Coulomb field around a nucleus and emits an energetic photon. A high-energy photon, on the other hand, will interact predominately via pair production, in which a photon converts into an electron-positron pair in the vicinity of a nucleus. The particles emitted in these interactions can themselves undergo the same procedure. Thus, an energetic electron or photon passing through a dense material will result in a shower of secondary electrons, positrons, and photons. This process is called an electromagnetic shower. The shower will continue to develop until all the secondary particles have sufficiently low energies that other energy loss mechanisms (mostly ionization) become important. The rate at which an incident particle loses energy is a constant of the material, and is usually specified as a radiation length: X_0.

$$\frac{dE}{E} = \frac{dX}{X_0}$$

Hadronic particles also cause showers, but the mechanism by which the shower is generated is different from that of electromagnetic showers. Hadrons lose energy primarily through inelastic collisions with atomic nuclei. These collisions produce secondary hadrons, which can in turn undergo inelastic collisions, and the process continues until all particles have either been stopped by ionization losses or absorbed by nuclear processes. The size of hadronic shower is given by the nuclear interaction length λ for the material. Hadronic showers are much more extended in space than electromagnetic showers of similar energy [8].

After the collision of electron and positrons (or other particle collision), a detector is needed to track and identify the particles produced by the collision. Through calorimetry, electrons and photons can be identified and measured, Jets can be identified and measured, and missing transverse energy E_T can be determined [9]. A calorimeter is a composite detector using a total absorption of particles to measure the energy and position of incident particles or jets. In the process of absorption, showers are generated by cascades of interactions.In the course of showering, eventually, most of the incident particle energy will be converted into heat. The calorimeter utilizes the characteristic interactions between the incident particles and matter(e.g. atomic excitation, ionization)to generate a detectable effect. Calorimetry is also the only practicable way to measure neutral particles among the secondaries produced in a high-energy collision.

Calorimeters are usually composed of different parts, custom-built for optimal performance on different incident particles. Each calorimeter is made of multiple individual cells, over whose volume the absorbed energy is integrated; cells are aligned to form towers typically along the direction of the incident particle. The analysis of cells

and towers allows one to measure lateral and longitudinal shower profiles; hence, their arrangement is optimized for this purpose, and usually changes orientation in different angular regions. Typically, incident electromagnetic particles, electrons and gammas for example,are fully absorbed in the electromagnetic calorimeter, which is made of the first layers of a composite calorimeter; its construction takes advantage of the comparatively short and concentrated electromagnetic shower shape to measure energy and position with optimal precision for these particles. Electromagnetic showers have a shape that fluctuates within comparatively narrow limits; Incident hadrons, on the other hand, may start their showering in the electromagnetic calorimeter, but will nearly always be absorbed fully only in later layers, i.e. in the hadronic calorimeter , built precisely for their containment. Hadronic showers have a widely fluctuating shape.

Calorimeters can also provide signatures for particles that are not absorbed: muons and neutrinos. Muons do not shower in matter, but their charge leaves an ionization signal, which can be identified in a calorimeter if the particle is sufficiently isolated, and then can be associated to a track detected in tracking devices inside the calorimeter. Neutrinos, on the other hand, leave no signal in a calorimeter, but their existence can sometimes be inferred from energy conservation: in a hermetically closed calorimeter, at least a single sufficiently energetic neutrino, or an unbalanced group of neutrinos, can be "observed" by forming a vector sum of all measured momenta, taking the observed energy in each calorimeter cell along the direction from the interaction point to the cell. The precision of such measurements, usually limited to the transverse direction, requires minimal leakage of energy in all directions, hence a major challenge for designing a practical calorimeter [10].

1.3 Sampling Calorimetry

From the construction point of view, the calorimeters fall into two categories: the homogeneous calorimeter and the heterogeneous calorimeter. For homogeneous detectors, the same medium is used both to cause the shower development and to detect the generated particles. In Heterogeneous, or sampling calorimeters, the medium responsible for the showering is separate from active material used in detection. The nuclear interaction length of active detector material is so large that calorimeters have to be sampling devices for the reasons of cost and compactness. A typical calorimeter therefore contains alternate layers of absorber (e.g. iron or copper) and detector (e.g. plastics cintillators, proportional counters, or liquid ionization chambers). In sampling calorimeters the functions of particle absorption and active signal readout are separated. This allows optimal choice of absorber materials and a certain freedom in signal treatment. Heterogeneous calorimeters are mostly built as a sandwich structure, with sheets of heavy-material absorber (e.g. lead, iron, uranium) alternating with layers of active material (e.g. liquid or solid scintillators, or proportional counters). Only the fraction of the shower energy absorbed in the active material is measured. Since hadron calorimeters need considerable depth and width to create and absorb the shower, they are necessarily of the sampling calorimeter type.

In practice, most calorimeters are combined electromagnetic and hadronic detectors. Because the nuclear interaction length is so much bigger than the radiation length, most hadrons pass through the electromagnetic front compartment and interact in the hadronic part behind. The figure below illustrates a typical sampling calorimeter.

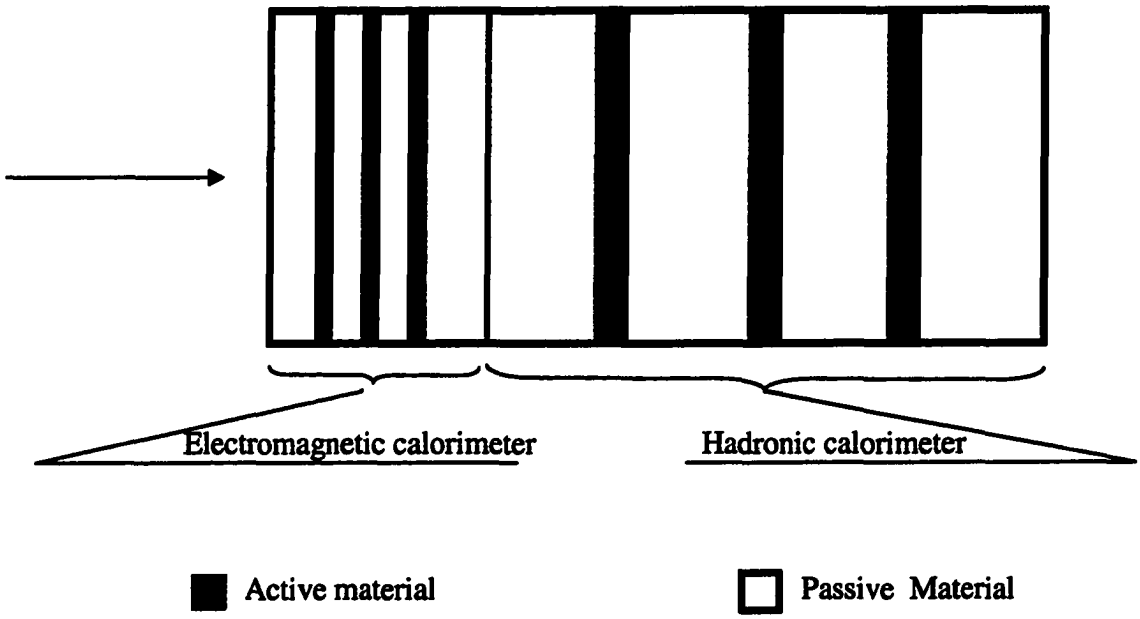


Figure 2.Illustration of a sampling calorimeter

The nature of sampling calorimetry requires that we need to do reconstruction to get the information on energy deposited in the whole calorimeter.

1.4 Energy Reconstruction

The raw event data output from the detector electronic readout are in the form of ADC counts, and these quantities these quantities themselves are not very interesting, they must be converted to physical objects such as electrons, photons etc. The process of turning the raw detector data into descriptions of objects such as leptons and jets is called reconstruction, and is carried out by certain computer programs.

The particle reconstruction process can be divided into three major phases [11]:

1. **Hit finding:** in this phase the raw data is unpacked and converted into 'hits' (i.e., energy deposits in calorimeter cells, or pulses on tracking chamber wires) of definite energy and spatial location. For the calorimeter, hit finding consists primarily of converting the energy deposited in each cell from ADC counts to GeV.

This conversion ultimately comes from test beam measurements, in which the response of calorimeter modules to beams of known energy are measured.

2. **Tracking and clustering:** during this phase, hits that are spatially close together are joined together to produce 'clusters' in the calorimeter and 'tracks' in the tracking chambers.
3. **Particle identification:** during this phase information from all parts of the detector is combined to produce a collection of objects which are candidates for being jets, electrons, or muons. The criteria used for identifying these candidates are deliberately made quite loose so that they have high efficiency, but so there will also be a large background. When performing an analysis, one will typically make much tighter cuts on the reconstructed objects.

After the particle reconstruction, we can utilize the reconstructed data samples, such as Z samples, to apply more strict selection cuts to select target particles, elections, for example. Once we have "pure" data samples, we can reconstruct the calorimeter energy by calculating the summation of weighted layer energies.

1.5 Application of calorimetry in high-energy physics

High energy physics, also called particle physics, is the study of the basic elements of matter and the forces acting among them. It aims to determine the fundamental laws that control the make-up of matter and the physical universe.

In order to do experiments in particle physics, it is necessary to give these particles incredibly high amounts of energy. There are a couple of reasons why we need high energy particles. The first reason is that the energy resolution in a particle detector is proportional to the energy of incident particles; the second reason we need high energy

particles is because we are interested in creating new particles. The accelerator is the basic tool to create these high energy particles, because it allows us to create the particle collisions that we want to study from. Accelerators work by accelerating charged particles using electric fields. There are two basic types of accelerators used in physics: linear and non-linear accelerators. A linear accelerator accelerates particles in a straight line, particles are injected at one end of the tube and are subjected an electric field, they are accelerated down the length of the tube, and come out with a higher energy. The longer the accelerator, the more energy the particle will have. An accelerator in California, SLAC (Stanford Linear Accelerator), is over 4 km long and can accelerate particles to energies of 50 GeV.

There are two types of collisions utilized in the linear colliders: one is for stationary target experiments, where particles are accelerated in the tube and then shot onto the target; another type is the linear collider, where instead of one beam of particles being accelerated, two beams of opposite charge are accelerated side by side. They then enter a magnetic field, and since they have opposite charge, they bend in opposite directions in an arc, where they then collide. Since beams are smashing into each other, they collide with a greater energy then they would with just one beam. The more energy in the collision, the better it is for studying the particles, and for creating new particles.

In circular accelerators, particles are accelerated along a circular path using electric and magnetic fields. They go around and around until the desired energy is achieved, at which point they are made to collide. Like linear accelerators, there are two types of collisions utilized, the first type is called the fixed target collision. The second type is the circular collider, which involves two oppositely charged particles being accelerated in

opposite directions around the collider and then made to collide when they have enough energy. Because two moving beams collide with each other, much higher energy collisions are made to occur.

A problem with circular accelerators is that as the particles go around the loop, they radiate away a great deal of energy. A large percentage of the energy the accelerator gives the particles goes to compensate the lost energy. The amount of energy loss depends on the mass of the accelerated particle, and how sharp the curve is. The heavier the particle, the less energy it radiates away; thus, these types of accelerators are more suited for use with protons and antiprotons. Also, the bigger the circumference of the accelerator, the less lost energy; therefore, accelerators with large circumferences are better.

The collision of high energy particles gives us an opportunity to study after-collision physics. So how do we obtain the after-collision information? This is where the calorimetry comes to existence. The diagram on the next page is a cross section of a simplified particle detector, from the diagram we can see that calorimeters play a very important roll in the detector.

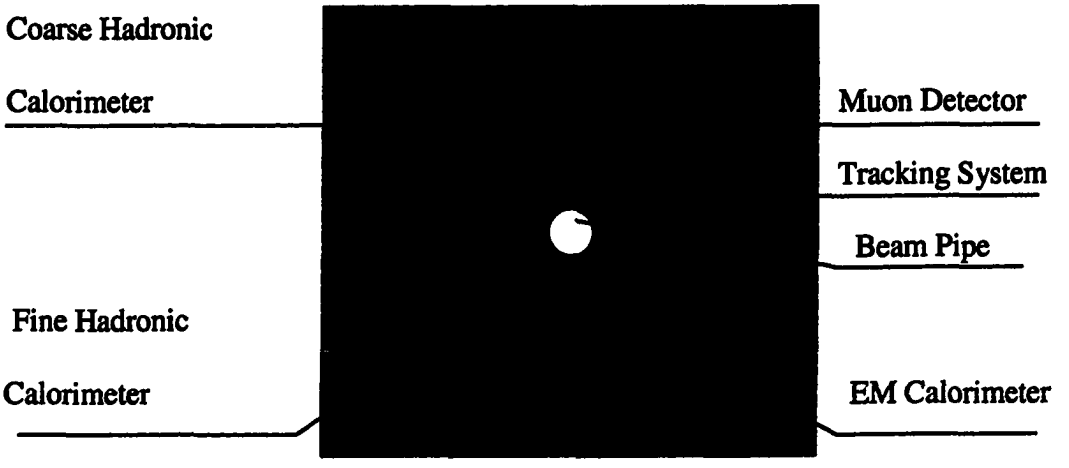


Figure 3.Cross Section of a detector

Broadly speaking, calorimeters are devices to measure the total energy of particles. Various techniques can therefore be applied, depending on the detailed properties of the particles of interest, and in general, no one device will be optimal for all types of particles. The two broadest subclasses of calorimeters in high energy physics are the electromagnetic calorimeters used primarily for photons and electrons, and the hadronic calorimeters used for most charged mesons and baryons. Most of these types of calorimeters operate by absorbing and thereby measuring a significant amount of the incoming particle's energy directly. Calorimeters, along with the track systems in the detector, provide us means to identify specific particles and their momentum and energy and therefore allow us to "see" what happened inside the detector.

1.6 Overview of Remaining Chapters

Chapter two is dedicated to the description of D0 experiment and D0 detector.

* Description of Fermilab Tevatron

* Description of D0 Detector

* Detailed information on D0 Calorimeter

-- Construction

-- Readout

-- Calibration

-- Inter-cryostat Detector

* Detailed information on D0 Central Pre-shower Detector

-- Construction

-- Readout

-- Calibration

Chapter three sets up a model for the calorimeter. Through the model, the energy of the calorimeter is reconstructed.

Chapter four discusses the data reduction and analysis.

In Chapter five, a likelihood fitting method is used to find a set of sampling weights for a central calorimeter and calorimeter ICD CCMG and ECMG regions.

In chapter six, Monte Carlo simulation of $Z \rightarrow e^+e^-$ event is discussed. It includes events generation, particle reconstruction, and simulation of energy resolution. The electron energy resolution from simulation is compared with that from real data samples.

-- Electromagnetic sampling weights obtained from Monte Carlo simulation.

-- Electromagnetic sampling weights obtained from $Z \rightarrow e^+e^-$ data sample.

-- Discussion of improved $Z \rightarrow e^+e^-$ sample purity using CPS information.

Conclusion

Appendix A --- Glossary

Appendix B --- Louisiana Tech University ICD group

Appendix C --- Program list

References

CHAPTER TWO

THE DØ EXPERIMENT

2.1 Fermilab Tevatron

The Fermilab Tevatron is the highest energy accelerator in the world. It became operational in 1983. It has a circumference of approximately 4 miles. The Tevatron accepts protons and antiprotons from the Main Injector and accelerates them from 150 GeV to 980 GeV. It uses cryogenically cooled magnets to accelerate the beam.

The Tevatron is the key element for Fermilab to achieve its mission. In fixed-target operation, the Tevatron accelerates protons to approximately 800 GeV and directs them onto a target of specially selected material such as lead, copper, beryllium, or even hydrogen. Materials are chosen because of the specific particles they tend to produce. Tevatron can also perform the collider operation. In a collider, beams are accelerated to 900 GeV, and when colliding, a proton and an antiproton annihilate each other and give all of their energy to new particles, allowing the creation of heavy but slow particles. The total energy of the collision equals aproximately 1.8 TeV. The Tevatron can be considered an excellent quark-producing machine. With its help, the CDF (The Collider Detector Facility) and DZero experiments discovered the top quark in 1995. With the

Run II upgrade, Tevatron has an increased level of luminosity. Luminosity is a measurement of particle interaction, specifically the chance that a proton will collide with an antiproton, the higher the luminosity,the greater the chance of quark production. With at least 20 times as many collisions as in Run I, Run II of Tevatron is promising in at least three areas:

• Thousands of top quarks, instead of the hundred or so in Run I, will be produced. That means scientists at Fermilab will achieve a new level of detail in studying the massive elementary particle discovered in 1995. It also means more precise measurements of the top and of the W boson (one of the force-carriers in the Standard Model) and this might point the way toward the discovery of the Higgs boson.

• New b physics. As the world highest energy accelerator, the Tevatron is thus uniquely suited to investigate instances of CP violation (anomalies between matter and antimatter) exclusive to the bottom quark.

• New particle discoveries. Increasing Tevatron luminosity also increases the possibility of creating higher-energy collisions that could produce particles heavier than the top, opening the door to super-symmetry. The theory of super-symmetry proposes that every particle in the Standard Model has one or two super-partner particles, which are much heavier and have a different spin. If the theory is right, then Run II has a chance to discover one or two of these heavier particles, provided that they are not too heavy.

2.2 DØ Detector

Fermilab has the world's highest energy particle accelerator Tevatron. It accelerates proton and antiproton beams to 980 GeV before they collide with each other. The collision points are surrounded by arrays of instrumentation called detectors. The DØ

detector is a large, multipurpose detector for studying $p\bar{p}$ collisions which has been operating at the Fermilab Tevatron since 1992.

Detectors for colliding beam experiments are composed of many different particle-detection devices, each with their own strengths and weaknesses. The general layout, however, is dictated by the physics of how particles interact with matter. Closest to the interaction point are the tracking detectors, which are devices designed to measure the three-dimensional trajectories of particles passing through them. Often, the tracking detectors are immersed in a magnetic field; this permits a determination of the momentum of the charged particles via a measurement of their bending radius. Surrounding the tracking detectors is typically a calorimeter; this is a device which measures the energy of particles which hit it. A calorimeter should be 'thick' so that it will absorb all the energy of incident particles; conversely, tracking detectors should contain as little material as possible so as to minimize multiple scattering and losses prior to the calorimeter. The DØ detector follows the general plan outlined above.

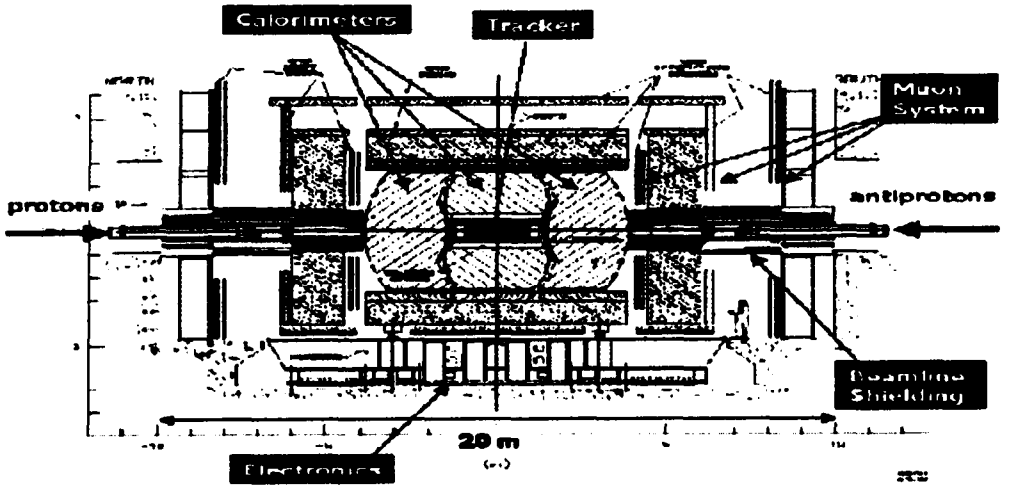


Figure 4.Cross section of DØ Detector

Figure 4 is a side view of the DØ Run II detector. It is about thirty feet tall and fifty feet long and consists of four major parts.

2.2.1 The DØ Coordinate System

The DØ uses a right-handed coordinate system with the z axis along the beam line, and the positive direction is coincided with photon beam. As shown in Figure 5, the positive x is defined to be pointing radially outward from the center of the beam line, and the positive y pointing vertically upward. The positive z is orthogonal to both x and y such that the system is right-handed. The positive z direction is also called "south".

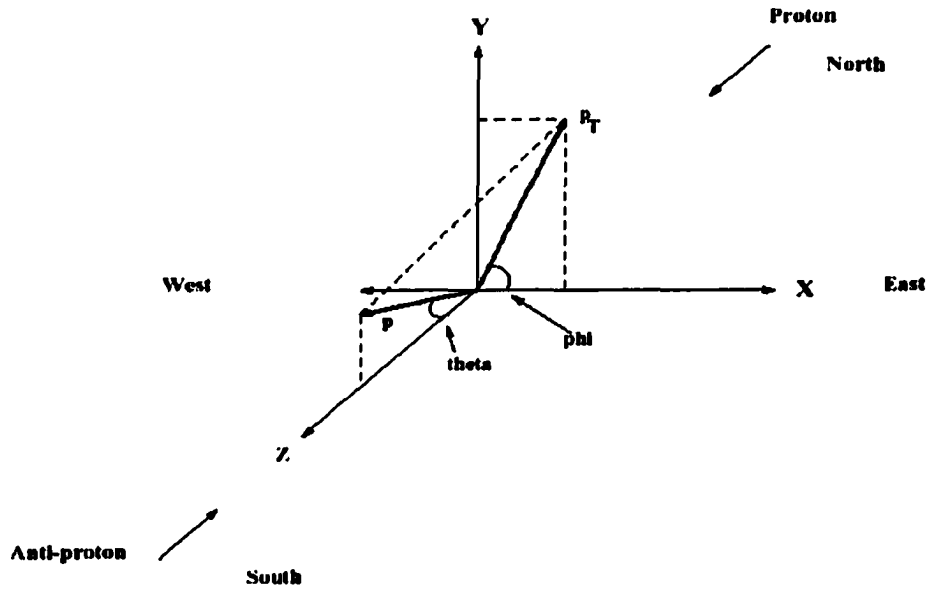


Figure 5.The DØ Coordinate System

A spherical coordinate system is also frequently used. The polar angle theta (θ) and the azimuthal angle ϕ are defined in a standard way for the spherical coordinate: θ is measured with respect to the positive z direction and ϕ is defined as the angle around the positive x axis with $\phi=0$ being the positive x direction and $\phi=\pi/2$ being the

positive y direction. It is convenient to introduce a function η of the polar angle θ called psudorapidity:

$$\eta = -\ln(\tan\frac{\theta}{2})$$

The reason to use the (η,ϕ) coordinate is that, in high energy circumstances, η approaches the true rapidity y of the particle:

$$y = \frac{1}{2}\ln\left(\frac{E+p_z}{E-p_z}\right)$$

Rapidity is a useful variable because of its invariance under a longitudinal Lorenz boost. P_T in the Figure 5 is called transverse momentum and is defined to be the projection of $\vec{p}$, momentum vector, on the $x-y$ plane:

$$P_T = |p| * \sin(\theta)$$

2.2.2 Tracking System

The point where the beams collide is surrounded by "tracking detectors" to record the tracks (trajectories) of the high energy particles produced in the collision. The measurements closest to the collision are made using silicon detectors. These are flat wafers of silicon chip material. They give very precise information, but they are expensive, so they are put closest to the beam where they don't have to cover so much area. The information from the silicon detector can be used to identify b-quarks. Outside the silicon, D0 has an outer tracker made using scintillating fibers, which produce photons of light when a particle passes through. The whole tracker is immersed in a

powerful magnetic field so the particle tracks are curved; from the curvature we can deduce their momentum.

2.2.3 Central and Forward Preshower Detectors

Between the outer surface of the solenoid and inner radius of the the central calorimeter is the Central Preshower Detector, the main function of which is for electron identification. The detector consists of three layers of fine-grained scintillator strips.

The DØ Forward Preshower Detector (FPS) is also designed to enhance the electron identification capability. The two FPS detectors will cover the pseudorapidity range $1.4<|\eta|<2.5$, with one detector mounted on the inner face of each of the End Calorimeter (EC) cryostats.

2.2.4 Calorimeter

Outside the tracker is a dense absorber to capture particles and measure their energies. This is called a calorimeter. It uses uranium metal in liquefied argon; the uranium causes particles to interact and lose energy, and the argon detects the interactions and gives an electrical signal that we can measure. The calorimeter is described in detail later.

2.2.5 Muon System

The outermost layer of the detector detects muons. Muons are unstable particles but they live long enough to leave the detector. High energy muons are quite rare and a good sign of interesting collisions. Unlike most common particles, they do not get absorbed in the calorimeter, so by putting particle detectors outside it, we can identify muons. Because the muon system has to surround all of the rest of the detector, it ends up being very large, and it is the first thing that you see when looking at DØ.

2.2.6 Trigger System

Proton-antiproton collisions happen inside the detector 2.5 million times every second. We cannot record all those events; at most, perhaps 20 events per second can be stored on computer tape. The trigger is the system of fast electronics and computers to decide, in real time, whether an event is interesting enough to be recorded.

2.3 DØ Calorimeter

2.3.1 Construction

The DØ calorimeter is a sampling calorimeter, with liquid argon (LAr) as the ionization medium. The primary absorber material is depleted uranium, with copper and stainless steel used in the outer regions. Since uranium is very dense, the calorimeter is relatively compact.

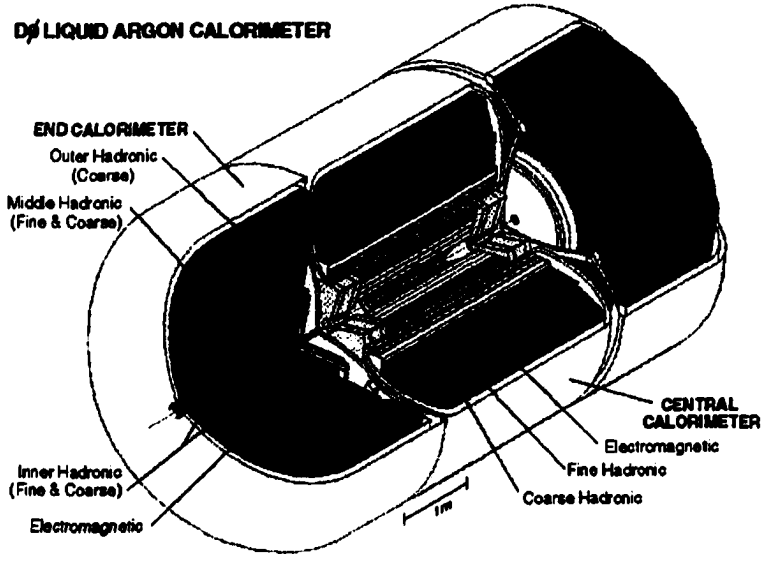


Figure 6.D0 Calorimeter

An overview of the calorimeter is shown in the figure 6 above. Since it uses LAr, it must sit inside of a cryostat in order to keep the argon cold. In order to facilitate assembly

and allow access to the central detectors, the calorimeter is divided into three major parts, each in its own cryostat: the central calorimeter (CC) and the two end calorimeters (EC). The central calorimeter provides coverage over a pseudo-rapidity of about 1.2. It is roughly toroidal, and consists of three concentric layers of modules. The inner layer consists of 32 electromagnetic (EM) modules, which are thick enough to contain most electromagnetic showers. The middle layer consists of 16 fine hadronic (FH) modules, which measure the showers due to hadronic particles. The final layer consists of 16 coarse hadronic (CH) modules, which measure any leakage of energy out of the FH layer, and which also serve to reduce any leakage out of the back of the calorimeter into the muon system.

The two sections of the end calorimeter provide coverage on each side of the CC from the pseudorapidity of about 1.3 to about 4. The EC is composed of three concentric layers of modules. Like the CC, the modules are divided into electromagnetic and fine and coarse hadronic types; however, the geometry is rather different. The center of the EC consists of a disc-shaped electromagnetic module, backed by the cylindrical fine and coarse inner hadronic modules. Arranged in a ring around this central core are the fine and coarse middle hadronic modules, and around them is a final ring of coarse outer hadronic modules.

In both the CC and EC, the area in (η,ϕ) space covered by a typical readout cell is 0.1x0.1. However, in the third layer of the EM modules, where electromagnetic showers typically deposit the bulk of their energy, the readout cells have areas of 0.05 x 0.05. In addition, cells with $|\eta| > 3.2$ have a ϕ size of 0.2 and are somewhat larger in η as well. See figure 7 below for the illustration of calorimeter segmentation.

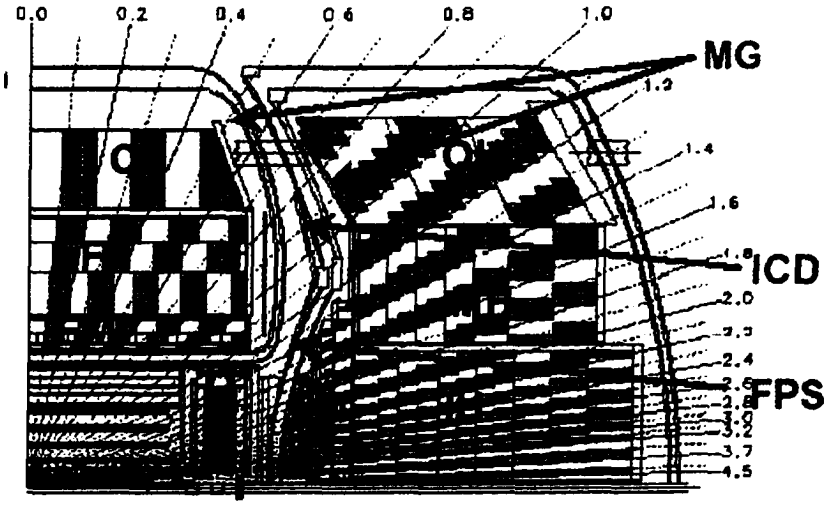


Figure 7.Calorimeter Segmentation

The whole calorimeter is divided into a large number of modules, each of which consists of a stack of interleaved absorber plates and signal boards. The following figure shows a cross-sectional schematic view of a section of this stack. The absorber plates are separated from the signal boards by a LAr-filled gap of 2.3 mm. The signal boards consist of a copper pad sandwiched between two 0.5 mm thick pieces of G10. The outer surfaces of these boards are coated with a resistive epoxy coating. During operation, the absorber plates are grounded, while a positive voltage of 2.0 -- 2.5 kV is applied to the resistive coatings. As a shower develops in the calorimeter, charged particles crossing the LAr gap leave a trail of ionization. The liberated electrons are collected on the signal board after a drift time on the order of 450 ns, and induce a signal on the copper pad via capacitive coupling. In order to measure the transverse positions of showers, the readout pads are subdivided into smaller cells.

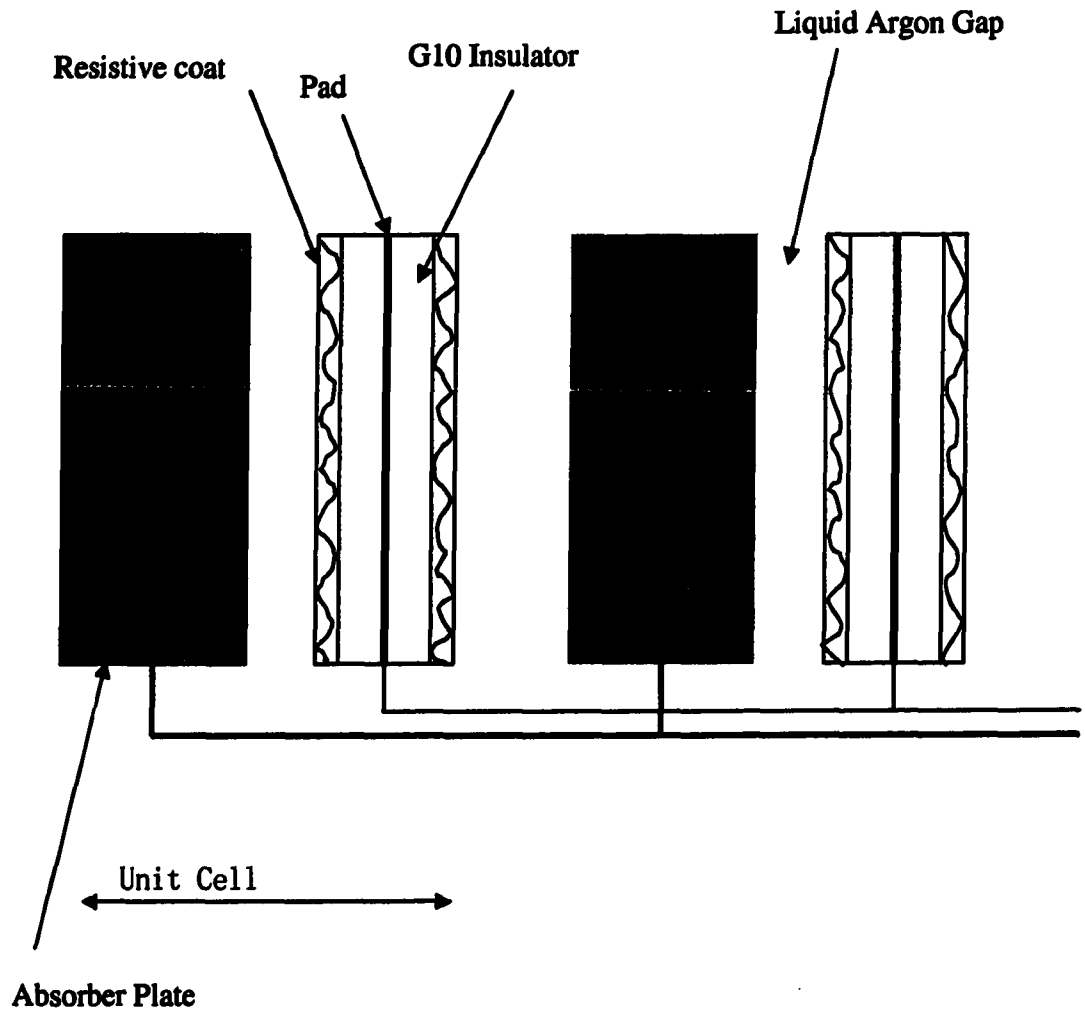


Figure 8.Calorimeter Cell illustration

2.3.2 Readout

Each calorimeter cell consists of a liquid argon cap between an absorber plate and a G10 board. The G10 board has a high-resistivity coating to which a potential is applied with respect to the absorber plate to create the drift electric field. Particles transversing the gap produce ionized trail of electrons and ions. In the electric field the electrons drift towards the G10coating, producing a current. The current induces image charge on a

copper pad etched on the G10 board under the resistive coat. A readout cell is formed from many pads grouped together.

Signals from calorimeter modules are brought to the four cryostat feed-through ports by specially fabricated 30Ω coax cables. These cables are connected to multilayer printed circuit feed-through boards located above the liquid level in each feed-through port. Eight T-shaped, 27-layer feed-through boards penetrate each port and provide a re-ordering of signals between the module-oriented input side and the $\eta - \phi$ order appropriate to analysis on the output side.

Short cables connect the output from the feed-through boards to charge-sensitive hybrid preamplifiers mounted in four enclosures on the surface of each cryostat, near the ports. The picture of a preamp is shown below.

Figure 9.Picture of a preamplifier

The preamps are integrating circuits that convert the image charges produced by the calorimeter cells to voltages that are proportional to the input charge. According to the pattern of detector capacitance, the calorimeter cells are classified into 14 categories[1]; therefore, 14 species of preamps are needed to compensate for the detector capacitance in

the electromagnetic layers, haydronic layers, and the central calorimeter and end-cap calorimeter cells.

Output signals from the preamplifiers are transported on twisted pair cable to base line subtractor (BLS) shaping and sampling hybrid circuit; the aim of the BLS is to subtract the signal baseline before an interaction from a signal just after it in order to get the exact signal amplitude. Input signals are integrated (430 ns) and differentiated ($33\,\mu$s). Signals from portions of logical cells that straddle the CC-EC boundary are merged at the BLS input. At the input to the BLS, a portion of the signal is extracted with about 100 ns rise time and added into trigger towers of $\Delta\eta = \Delta\phi = 0.2$ for use in event selection. The main signals are sampled just before s beam-crossing and 2.2 μ s after; the difference is provided as a dc voltage proportional to the collected charge. Two storage capacitors for each channel allow double buffering at the analog level. Fast baseline restoration occurs in a few μ s so as to minimize the effects of event pile-up.

Depending on signal size, the BLS outputs can be amplified by 1 or by 8 so as to reduce the dynamic range requirements of the subsequent digitization. A bit is set to record the chosen amplification; there is special provision for forcing a specific amplification (or both) for calibration purposes. The BLS outputs are multiplexed 16-fold onto the crate backplane, and each channel is sent in serial time slices over 50m twisted pair cables from the detector platform to the MCH.

The 24-channel 12-bit ADC circuits in the MCH, together with the x1 or x8 amplifier allows an overall dynamic range of 2^{15}. Each time-slice of each channel is digitized in about 10 μ s, yielding a total digitization time of 160 μ s for 384 signals. Gain parameters are set so that about 3.75 MeV of deposited energy corresponds to one least

count. Minimum ionizing particles deposit between 8(EM1) and 90(FH1) MeV in the layers of the calorimeters. Channels which have an absolute difference of signal and pedestal below an adjustable threshold can be suppressed from the readout buffer.

Both single-channel random noise (electronics and uranium radioactivity) and multi-channel coherent noise have been measured in beam tests and in the D0 Hall. The single channel noise can be characterized as 2000 + 3100xC (nF) electrons. For many channels, the total random noise varies with square root of N, where N is the number of channels included; the total coherent noise varies, typically as N. A useful measure of coherent noise is the number of channels that can be summed before all the coherent noise exceeds the random noise. In the DØ environment, this number is typically greater than 5000 channels[6]. The diagram on the next page is a simplified calorimeter data flow path.

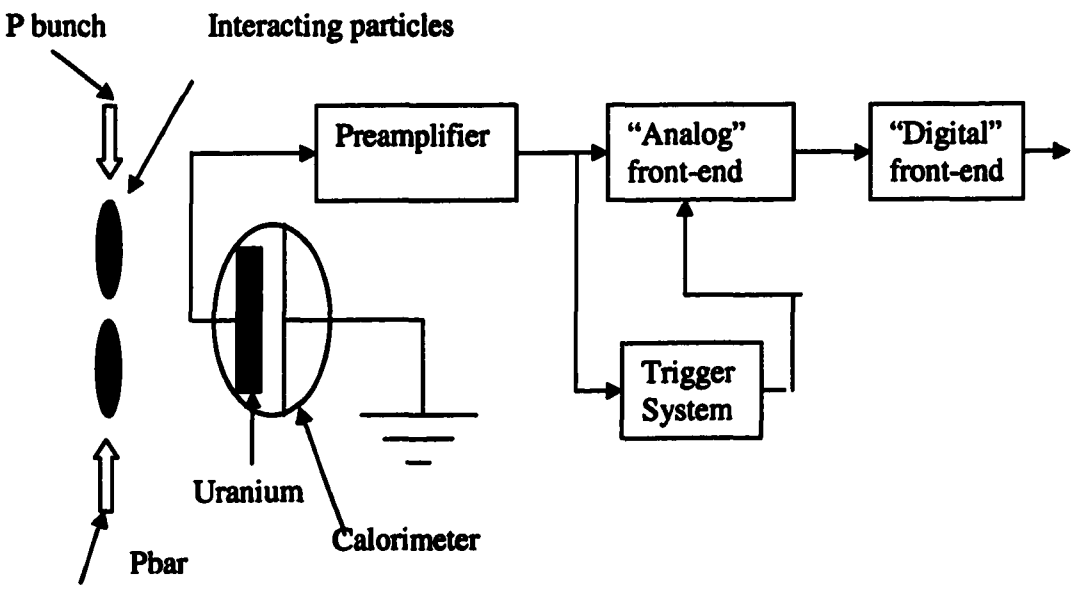


Figure 10.Calorimeter Data Flow Path

2.3.3 Calibration

Hardware of calibration system:

The DO Calorimeter detector is made up of many cells sandwiched between uranium plates. This sandwich is immersed in liquid argon and has an electric field applied that causes any free ions to be collected on the cell. These cells are connected in towers that extend away from the interaction point. As particles pass through the detector, they liberate ions that collect on detector cells. The charge on these cells is integrated by Charge Sensitive Preamplifiers, producing a proportional voltage. The output of each preamp is shaped, digitized, and stored if a trigger occurs. The mapped cell information is used to reconstruct particle paths and energy deposits.

The calorimeter itself is divided into three main sections, the central calorimeter and the north and south end-cap calorimeters. Each section contains four Preamp Boxes and each box contains 4608 preamps. There is a calibration system associated with each Preamp Box for a total of 12 systems. During a calibration run, the signal from each system is directed to one of the 32 positions in the Preamp Box, stimulating 144 preamp channels. The position is incremented until all preamps have been stimulated.

The calibration system is composed of three units. (See the figure below)

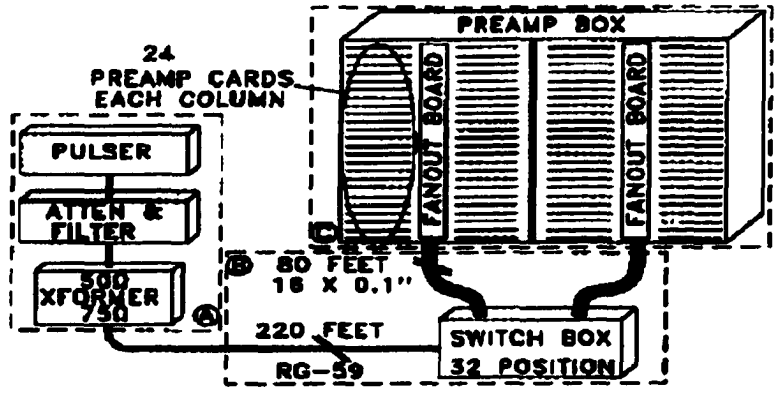


Figure 11.Calorimeter Calibration System

A. Pulser, attenuator, filter, transformer.

B. Distribution switch box, transmission lines.

C. Fan-out board, preamp board

Calibration is accomplished by injecting charges into the front end of each preamp hybrid. A timed high voltage pulse is converted to a current source by a 500KQ resistor at the preamp input. An often used method of injecting charge is to connect a capacitor, which is charged to some known voltage, to the input of the preamp. The preamp output voltage can swing to 5 volts and the integrating capacitor is 5 pico-farad. The calibration signal, therefore, must supply a charge of 25 pico-coulombs for full scale.

$$q = CV = (5x10^{-12})5 = 25 \text{ pico-coulombs}$$

The three units are described below.

Part A:

Pulser

The pulse is formed by charging a transmission line3 to a predetermined voltage. A DAC power supply is connected to the line through a limiting resistor RLIMIT and a BLOCKING diode. See the block diagram for the pulser below.

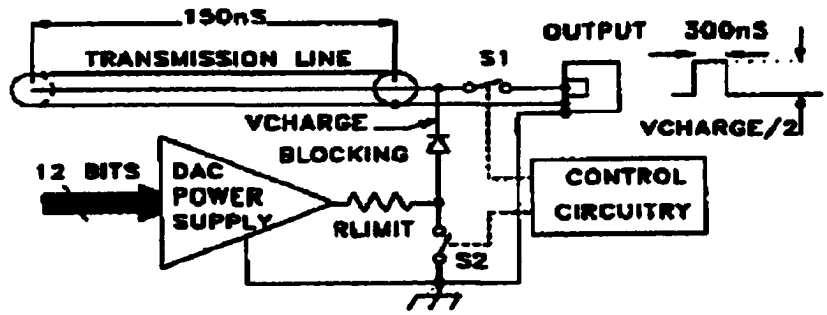


Figure 12.Pulser Block Diagram

The switches S1 and S2 are opened and the cable charges to the DAC voltage setting. When the Pulser is triggered the CONTROL CIRCUIRY immediately closes S1 and S2.

The switch S1 connects the output of the pulser to the distribution system. Switch S2 removes the charging voltage so the Silicon Controlled Rectifiers turn off. The result is a well-defined pulse of fixed width, with variable amplitude controlled by a 12 bit DAC.

Attenuator

The amplitude of the pulser's output is coarsely controlled by a 0-120 db attenuator. The attenuation factor can be remotely set in 10 db increments, allowing coverage of the entire dynamic range of the electronics. This unit takes 100 milliseconds to configure a new setting but is not changed often.

Filter

The output of the pulser is a rectangular pulse with rise and fall times in the 10ns range. Due to physical constraints of the distribution of the pulse, the final lines that feed the preamps (4 cards; 6 hybrids/card) cannot be terminated into the characteristic impedance. Therefore, some frequency limits are imposed to help control reflections. The final lines, which are not terminated, are short, compared to wavelength.

Transformer

The transformer provides three functions.

First, it inverts the polarity of the signal sent to the preamp hybrids. The initial design of the pulser was for preamps that required a positive signal. However, during development it was established that the opposite polarity preamp (that is a negative voltage pulse) should be used. Since pulser construction was completed with a positive voltage pulse, it was decided that a transformer could provide the inversion with no other rework of the system.

The transformer provides DC isolation between the Preamp Box ground, which is mounted on the detector, and the Moving Counting House ground, 150 feet away.

Finally, it matches the output impedance of the pulser / Attenuator / Filter (50Ω) with the distribution system and fan-out (75Ω).

Part B:

Transmission line

Each transmission path is electrically equal. All cables are matched for length and attenuation and each path in the fan-out board is made to be of equal length.

Delivering the calibration signal to the preamps requires a low loss system. Initially, a 50Ω system was developed, but it was found there was excessive signal degradation. A shift was made to use a 75Ω system with the largest cable possible.

Switch

The pulse must be distributed to all of the electronic channels in the system. Each Preamp Box has 32 distinct paths that the pulse can be channeled to (16 in each half of the Preamp Box). The Switch Box is simply a 32 position coaxial switch.

Mercury wetted reed relays are used to provide the switching to the particular path. They provide a very repeatable and reliable connection to each position selected.

Part C

Fan-out

The purpose of the fan-out is to distribute the pulsed lines, which emanate from the Switch Box, over the entire Preamp Box. It is also where the signal is terminated into its characteristic impedance (75Ω). Each Switch Box drives two fan-out boards that reside in each half of the Preamp Box between two columns of preamp cards. The final path of the pulse signal is via short cables connecting the fan-out board to the preamp board. One pulse position (1 of 32) will cause six areas of the preamp box electronics RI be 'hit'

simultaneously. A total of 144 channels of electronics are pulsed at any one position. The final conversion of the pulse from a voltage to a current signal is carried out on the preamp card. The input of each preamp has the high value resistor (500kΩ) that converts the voltage signal into a current signal.

The content of calibration:

Linearity: The linearity of the calorimeter electronics and the pulser system is verified with a comparison of the electronics response (in ADC counts) vs. the pulse height (in DAC units) for each gain path(x1 and x8). The corresponding fitted slope, a_1 and a_8, are of the order of 0.25 ADC/DAC and the ratio $g = a_8 / a_1 \approx 1$, which leads to 1 ADC corresponding roughly to 1 MeV.

Non-linearity: The non-linearity at low energy has been observed and the cause can be traced back to the Switched Capacitor Arrays, which are storing the analog calorimeter signal until the Level 1 and Level 2 trigger decisions have been taken. For the ADC to energy conversion a universal correction function has been derived at Fermilab. After applying this correction function, the residues are better than ±5 ADC over the whole range for both gain paths.

Gain: For all calorimeter channels, the gain calibration factors have been determined from the deviation of the slope ADC/DAC from its ideal value of 0.25. The factors obtained show that there is a dependency with respect to the pre-amp type and the capacitance of a given cell. The dispersion of the factors for electromagnetic channels is about 5%, and 10% for hadronic channels. There are also systematic shifts of the slope values for different pre-amp types.

- inter-calibration In order to take into account variations between the different

calorimeter modules or inhomogeneous distributions of non-instrumented material in front of the calorimeter, the uniformity of the calorimeter response in ϕ is evaluated using the number of events and the energy deposited in each of the 64-φ modules covering one ring of $\eta = 0.1$.

2.4 DØ Preshower

Electrons play an indispensable role in high P_T physics at the Tevatron collider. The success of the Tevatron Run II physics program depends on whether the detector can efficiently identify electrons and muons both on-line and off-line with an acceptable level at high luminosity. Given a limited trigger bandwidth, additional background rejection is needed for lepton triggers at the high luminosity from Run II. The DØ central pre-shower detector is designed to improve electron identification and to help the electromagnetic energy resolution otherwise degraded by the presence of the solenoid. The cylindrically shaped scintillating detector with axial and stereo views is installed between the calorimeter and the solenoid. The pre-shower detector will provide an additional means of distinguishing electrons from the background.

2.4.1 Construction

The DØ detector is a compact detector and it has only a 5.0 cm radical space between the solenoid and the calorimeter. With this limitation and cost consideration, a scintillator based detector was chosen to become the pre-shower detector. The pre-shower detector consists of a tapered lead radiator plus three layers of finely segmented scintillating strips with wavelength-shifting (WLS) fiber readout. The three layers of strips are arranged in z-u-v views. The cylindrically shaped detector is placed directly outside the solenoid and

inside the D0 central calorimeter cryostat bone at a radius of about 72 cm. The solenoid consists of about one radiation length(X_0) of material at normal incidence. The lead radiator is tapered along the beam direction so the coil plus lead yields an approximately $2X_0$ total thickness for all particle trajectories. The following figures are the end view and side view of the central pre-shower detector (CPS).

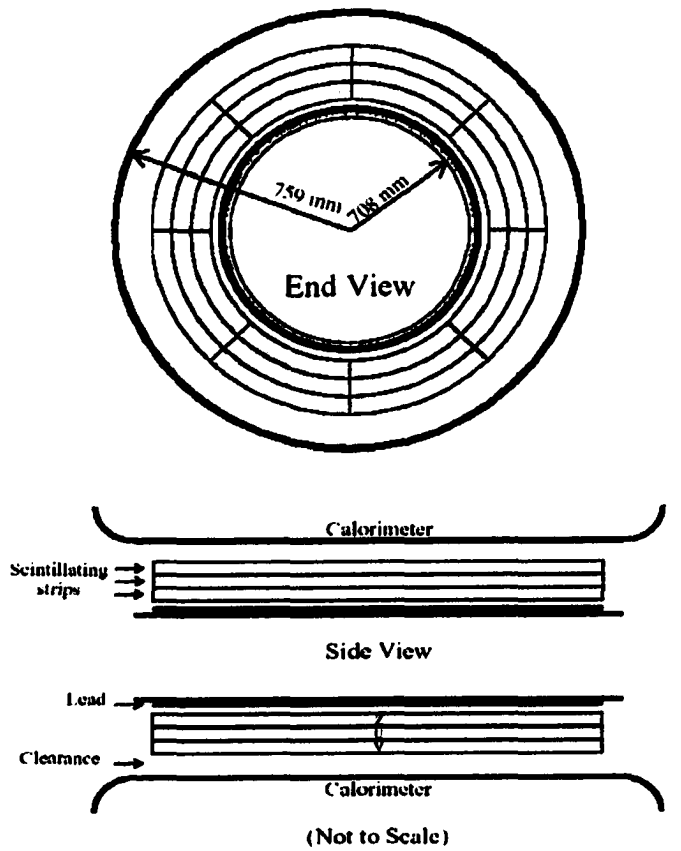


Figure 13.Schematic end and side view of the central pre-shower detector

The space geometry for each layer of scintillators is described in Table 1 below:

Table 1 Basic geometry of the central pre-shower detector

Items	Detector Space (Radii)(mm)	Clearance (Radii)(mm)	Length (mm)	stereo Angle (degree)
Solenoid	--707.8	--	2730.0	--
Lead	707.8-714.3	--	vary	--
Air gap	714.3-718.0	--	--	--
Axial layer	718.0-725.5	725.5-729.0	2730.0	0.0°
U stereo layer	729.0-736.5	736.5-740.0	2730.0	22.98°
V stereo layer	740.0-747.5	747.5-759.0	2730.0	23.29°
Calorimeter	759.0--	--	--	--

The pre-radiator of the pre-shower detector is composed of the solenoid and the tapered lead. To make a constant 2 radiation length for all particle trajectories, a tapered lead is installed on the solenoid, and the lead is stepped with the step size approximately equal to the beam size along the z direction.

To facilitate the construction, each layer of the detector is made into 8 octants, each 273cm long. The WLS fibers are split at z=0 and are connected to the clear fibers at both ends of each octant. The fiber splitting at z=0 effectively has the occupancy for each channel and therefore make the detector less vulnerable to high rates. The clear fibers from either end of octants are grouped together to form one bundle. There are a total of 24 octants and 48 clear-fiber bundles for the central pre-shower detector.

The information from the axial strips of the pre-shower detector will be combined with the fiber tracker information at level-1. Since fibers are not split for the fiber tracker, the

signals from the two WLS fibers of the same axial strip of the pre-shower detector will have to be combined electronically before a match with the tracker can be made.

Since 80 trigger sectors in φ are used in the fiber tracker, the pre-shower detector is also subdivided into 80 sectors, with 10 sectors in each octant. In addition, there are 16 strips in one sector (2x160 readout channels per octant) resulting in a total of 7680 readout channels for the detector. The fiber-fiber connectors are designed to match the segmentation of the detector with 16 channels per connector.

2.4.2 Readout

Scintillation light produced by showers will be collected by the WLS fibers and piped through clear fibers to the photo-detectors outside the magnet. A typical length for the clear fibers is 8 – 11 meters. The WLS and clear fiber are joined by connectors at the end of each fiber. Visible Light Photon Counters (VLPC) are used as photo-detectors. This solid state device is capable of high-rate single photon counting with a quantum efficiency greater than 70%. Typically, the VLPC has a gain of about 10^4. Before the VLPC signals are sent to SVX-II chips for amplification and digitization, they will be split into two channels each by special chips designed for the pre-shower to allow for fast trigger pick-off and to effectively extend the dynamic range of the readout system. To facilitate the Level-1 electron trigger, the readout for the axial layer is integrated with the fiber tracker readout.

Figure 14.The extruded triangular scintillator strip

2.4.3 Calibration

As with all detectors, in order to have physical significance, one must convert a signal to real world units. In order to do this, one must excite the detector with a known, stable source and observe the signal. Calibration of the detector will be done in two steps. Light emitting diodes (LED) will be utilized to provide a quick calibration on-line.

The WLS fibers will be excited by blue LED's at the ends of the detector. By comparing the one and two photoelectron peaks from the LED light, the relative calibration of the detector elements can be performed. An absolute calibration can be done by two methods. One method is to use the minimum ionizing particles. Monte Carlo simulation shows that the pre-shower detector will be showered by the non-interacting charged pions. The peak corresponding to the dE/dx energy loss of the minimum ionization particles is very clearly visible; therefore, the detector can be calibrated using MIT energy loss if the readout system is capable of detecting the MIP peak. The other method is to use the ϕ symmetry of physics events. No known physics will produce energy flow asymmetry in ϕ for unpolarized PP_{bar} beams; and that implies that any non-uniformity of the detector response can be calibrated by comparing the energy distributions for each channel.

CHAPTER THREE

MODEL FOR THE CALCULATION OF CALORIMETER ENERGY

3.1 Energy of A Single Channel

We mentioned in Chapter 2 that the DØ calorimeter is a sampling calorimeter; it has 17 layers, each of which is subdivided in eta and phi. One such subdivision corresponds to a single read-out electronics pathway, and is called a channel. The DØ calorimeter has 55000 channels, corresponding to 0.1x0.1 in $\eta - \phi$ for most of the calorimeter. Electrons usually develop showers within electromagnetic layer and deposit energy in EM layer 1 through EM layer 4 and FH1, fine hadronic layer 1.

Energy deposited in the calorimeter is recorded by the detector electronics as ADC counts. To calculate the calorimeter energy, we need to convert the ADC counts into energy. If we denote the ADC count for channel k in layer i as A_{ik}, then the energy for this channel will be

$$E_{ik} = C_i * A_{ik} .$$

where C_i is the conversion coefficient for the i_{th} layer. Here, the conversion coefficients for all channels in a layer are the same because all cells in a specific layer have the same physical dimensions, such as width and thickness, and all channels have the

same electronic setup along the electronic read-out pathway. For example, all channels have the same capacitance and same preamp gain, etc. Figure 15 illustrates the channel-layer structure.

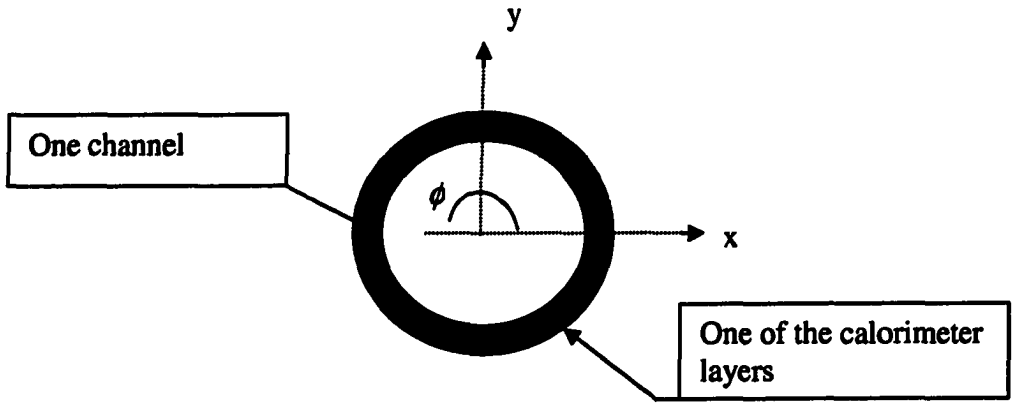
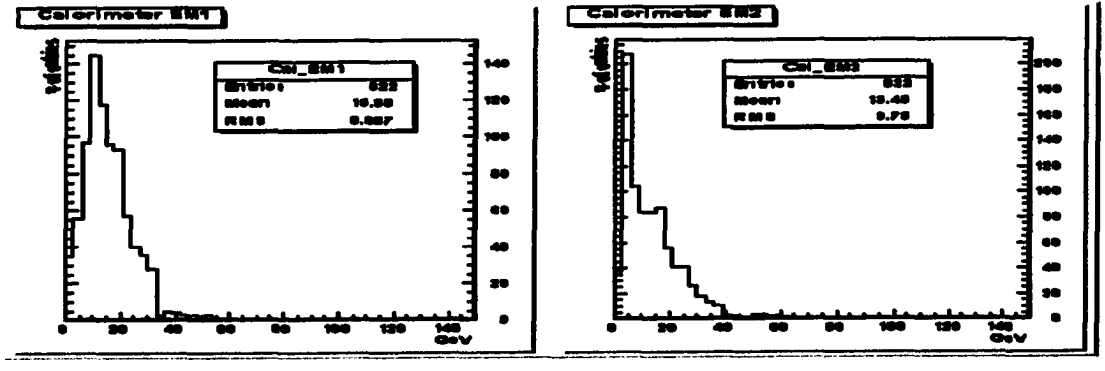


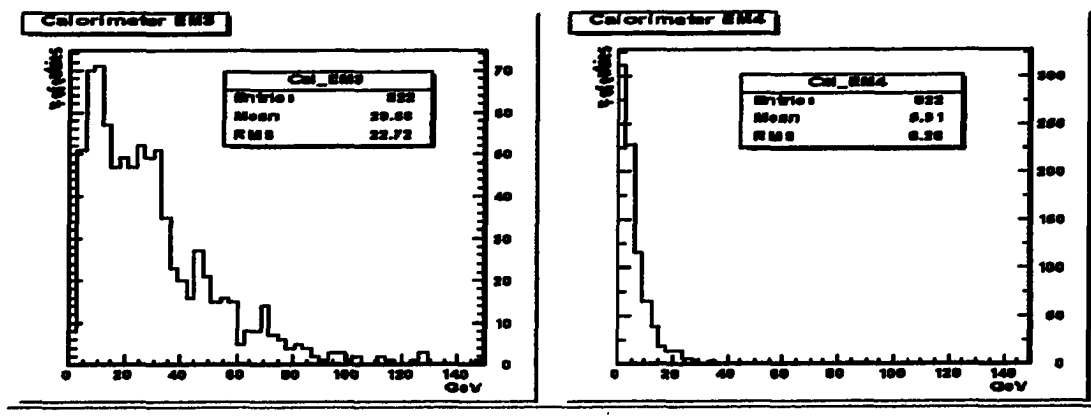
Figure 15.Channel-Layer illustration

3.2 Weighted Single Channel Energy

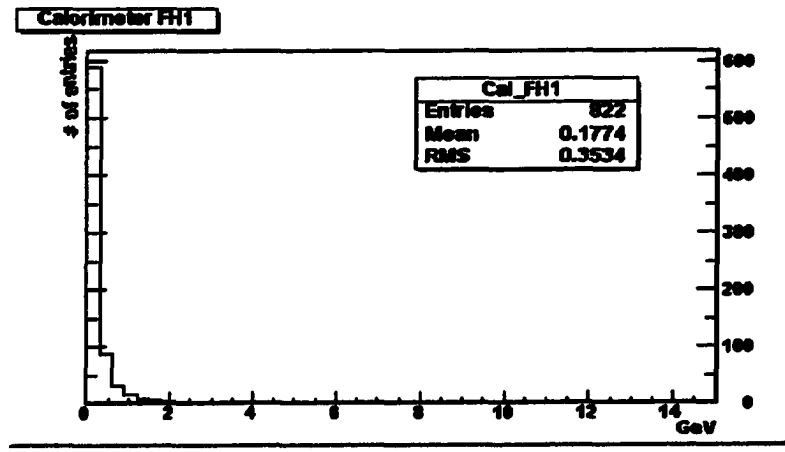
Because particles deposit different amount energy among different layers, the energy deposition is not evenly distributed. For electrons, most of the energy is deposited in electromagnetic layers or EM1 – EM4 layers, although a fraction is deposited in the FH1 layer. Among EM layers, EM3 usually has the biggest portion of the energy deposited by electrons. The Figure 16 "Electron Energy Distribution" illustrates the fact.



EM1 EM2



EM3 EM4



FH1

Figure 16.Electron Energy Distribution

The mean of these layered energy distribution for a sample of 60 GeV electrons is: 15.38 GeV, 13.43 GeV, 29.66GeV, 5.91 GeV, 0.18 GeV for EM1 through EM4 and FH1 layer respectively.

If we take into consideration the weights of the energy deposition by different layers, the single channel energy formula becomes

$$E_{ik} = C_i * A_{ik} * W_i .$$

where W_i is the sampling weight for the i_{th} layer. Again, all the cells in a specific layer have the same sampling weight because they all have the same amount of material and sampling layers in front of it, and all cells in the layer have the same response on average because of the reason mentioned earlier(capacitance, gain etc.).

From the expression of single channel energy, the energy deposition for the i_{th} layer can be expressed as

$$E_i = \sum_k C_i * A_{ik} * W_i$$

E_i --- Energy deposited in the i_{th} layer.

C_i --- Conversion coefficient for the i_{th} layer.

A_{ik} --- ADC counts for the channel k in the i_{th} layer.

k --- number of channels.

3.3 Final Formula

Due to the fact that the energy deposition is different from channel to channel, we need a factor to reflect this difference. ε_{ik} is the correction factor in the following formula

$$E = \sum_i \sum_k C_i * W_i * A_{ik} * \varepsilon_{ik}$$

In the formula, k stands for the number of channels per layer, and i is the number of layers. Thus, the summation over k gives us the total energy of all channels of layer i, and the summation over i results in the total energy for all layers, the calorimeter energy.

3.4 Data Reduction Framework

The goal of the data reduction is to find a set of data samples which are suitable for the energy reconstruction of different layers, including the ICR region. The following diagram illustrates the schematic to conduct the data reduction.

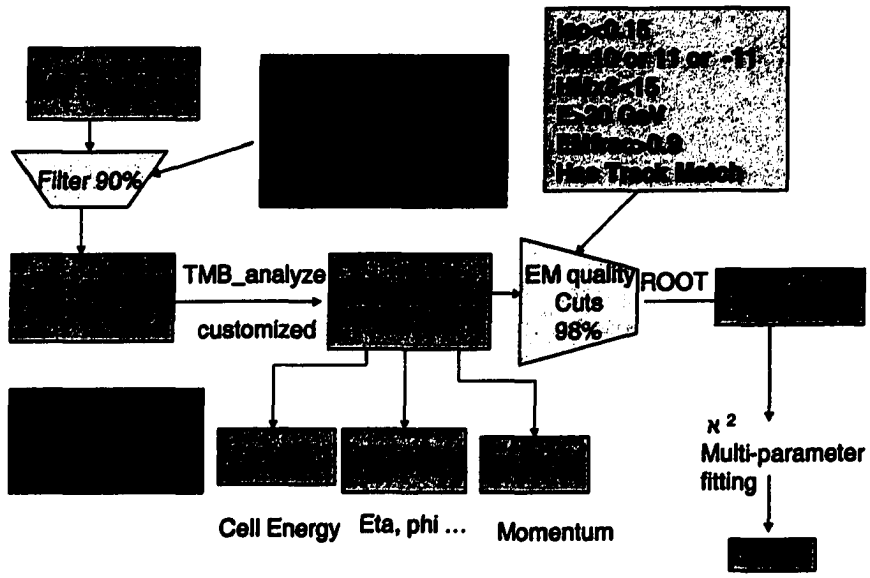


Figure 17.Data Reduction Schematic

First, all the original data, no matter whether they are from the DØ data acquisition system or created by D0 simulation program Monte Carlo, are called raw data. These data contain all the information about the detector response to the incident events. These raw data are then processed by DØRECO, a software package which is used to perform particle and jet reconstruction. After going through this step and basic selection standard for EM objects, about 90% of raw data are filtered out and the outputs are stored in SAM (explained in next section) in different formats, among them Thumbnail.

Thumbnail is a compressed data format and can not be used directly; this is where another software package, TMB_analyze, comes in. It takes a Thumbnail file as input and generates a Thumbnail ROOT tree which contains particle information (CAL, EMPART_S, TRACK etc.) and can be used by a user's analysis program. In order to do the analysis, further customized selection cuts(quality cuts)for qualified electrons are

necessary; these customized selection standards make certain requirement on iso, id, HMx8, E, EMfrac, and track match. The next chapter "Data Ruduction and Analysis", provides detailed information about these parameters. The result of quality cuts is a data sample ready to be used with ROOT software package for analysis.

3.4.1 Data Retrieval

SAM, stands for "Sequential Access Model", is a database system created at Fermilab to handle all DØ data in a manner that allows data browsing and retrieval. Users can make request for datasets (data collections) stored on SAM; they can also store their own data files on SAM. SAM provides a worldwide system of shareable computing and storage resources [12].

EDM stands for Event Data Model. A package is a group of closely related classes and class utilities. For DØ experiment, the EDM contains three packages: identifiers, edm, and rcp.

- The **identifiers** package is used to identify various objects in (or related to) an event.
- The **edm** package is the home for the fundamental classes of the EDM: *Event*, *AbsChunk*, *TKey*, the selectors, and *THandle*. These are the classes from which one inherits to extend the event data portion of the EDM. The **edm** package also includes the class template *Link*, and its associated classes.
- The package **rcp** contains the classes *RCP* and *RCPManager*, which provide a way for programs to read run-control data from external text files.

Chunks are the smallest unit of persistence of event data; they are also the

mechanism by which reconstruction packages communicate their results to each other. Chunks contain the following information:

1. Bookkeeping information
   - The information of chunks used to create this chunk.
   - The description of the algorithm(s) used to create this chunk.
   - The description of the calibration constants, alignment constants, or other non-event-specific information used to create this chunk.
2. Typing information
   - The name of the class for an instance of the class.
   - This name must be identical to the one returned by instances of the class.
3. ASCII representation of itself (for use in simple debugging) which can be printed to a given output stream.

Data are originally from two sources: data recorded during the online data taking or simulated events produced with the DØ Monte Carlo (MC) program. These physics data are called Raw Data Chunk (RDC). Each RDC contains the raw detector signals and is the primary input to DØRECO, the DØ offline reconstruction program. DØRECO is designed to produce two output formats which can be used for further physics analyses. The Data Summary Tape (DST) contains all information necessary to perform any physics analysis [13], and is designed to be 150 Kb per event. The Thumbnail (TMB) contains a summary of the DST, and is designed to be 15 Kb per event. The RECO output will be stored in SAM database.

Since this analysis uses data in Thumbnail format, it is necessary to provide some basic information on it. The Thumbnail data format was created to deal with the huge amount of data to be stored for the D0 Run II. The fundamental idea is to avoid storing redundant information and keep relevant information in the most compact way [14]. A typical Thumbnail file contains the following chunks:

- the ThumbNailChunk,
- the HistoryChunks, providing information history,
- the TMBTriggerChunk, providing a minimal information about triggers,
- the MCKineChunk (only for Monte-Carlo files).

The main content of a Thumbnail file is the ThumbNailChunk. The physics data chunks from RECO program are compressed into this chunk and stored on disks, so users can not use it before it is decompressed back into physics chunks. The following chart describes the data flow with Thumbnail.

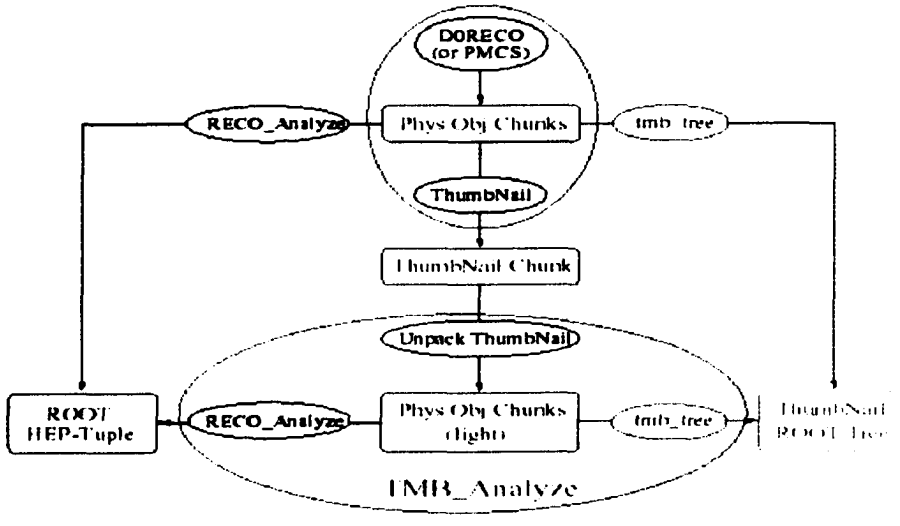


Figure 18.Generation and use of ThumbNail

The starting point is the D0RECO output, Physics Object Chunks. Information from these trunks is compressed and packed in a ThumbNailChunk by a software package

called ThumbNail. Data in this format must be unpacked by a software package called UnpackThumbNail before they can be used for analysis purposes; the output will be reconstructed Physics Object Chunks. Finally, an interface program called TMB_Analyze is designed for users to perform unpacking and to choose to produce either ROOT-HEP-Tuple or ThumbNail ROOT Tree by modifying certain rcp switches.

3.4.2 Data Storage Model

Why ROOT tree?

From the last section we know that users can produce either Root-Tuple or ThumbNail root tree. ROOT tree data format is used for the analysis. The reason is that it has some critical advantages over HEP-ROOT tuples:

- Information is stored in TCloneArray [14], which enables efficient storage and retrieval.
- The same C++ class used for creating the tree has to be used in analyzing it.
- C++ pointers can be used to link different quantities stored in the tree.
- New features of ROOT does not support ROOT-tuple format.

Content of a ROOT tree

A ROOT tree is generated from thumbnail files through TMBAnalyze program.

The following table [15] lists the branches of a ROOT tree and the physics objects corresponding to that branch.

Table 2 List of branches

Object	Branch	File	cvs package
Vertex	Vrts	TMBVrts.hpp(.cpp)	
ChargedParticle	Trks	TMBTrks.hpp(.cpp)	
	IsoTrks	TMBIsoTrks.hpp(.cpp)	
EMparticle	Emcl	TMBEmcl.hpp(.cpp)	
MuonParticle	Muon	TMBMuon.hpp(.cpp)	
Tau	Taus	TMBTaus.hpp(.cpp)	tmb_tree
Jet	Jets	TMBJets.hpp(.cpp)	
MissingET	Met	TMBMet.hpp(.cpp)	
FPSRecoCluster	PS	TMBFps.hpp(.cpp)	
CPS3DCluster	CPS	TMBCps.hpp(.cpp)	
Global info	Glob	TMBGlob.hpp(.cpp)	
Triggers fired	Trig	TMBTrig.hpp(.cpp)	
MCParticle	MCpart	TMBMCpart.hpp(.cpp)	mc_analyze
MCVertex	MCvtx	TMBMCvtx.hpp(.cpp)	
LinkedPhysObj	TRefs	TMBTRefs.hpp(.cpp)	tmb_analyze

The first column gives the names of physics objects, the second column lists the names for the corresponding objects, and the third column gives the name of files that implement those physics objects and the tree.

If a branch contains a single entry per event (for example MissingET),then this branch is represented by a simple class. If a branch contains multiple entries (for example several jets in one event), then this branch is represented by a TClonesArray. If there are several Physics Chunks (for example jets reconstructed using several algorithms)for one type object, only one branch is created and a word is added to each reconstructed object as a label indicating the algorithm used. Objects stored in the tree are linked by TRef pointers or a TRefArray array of pointers [16]. These links are illustrated in Figure 19.

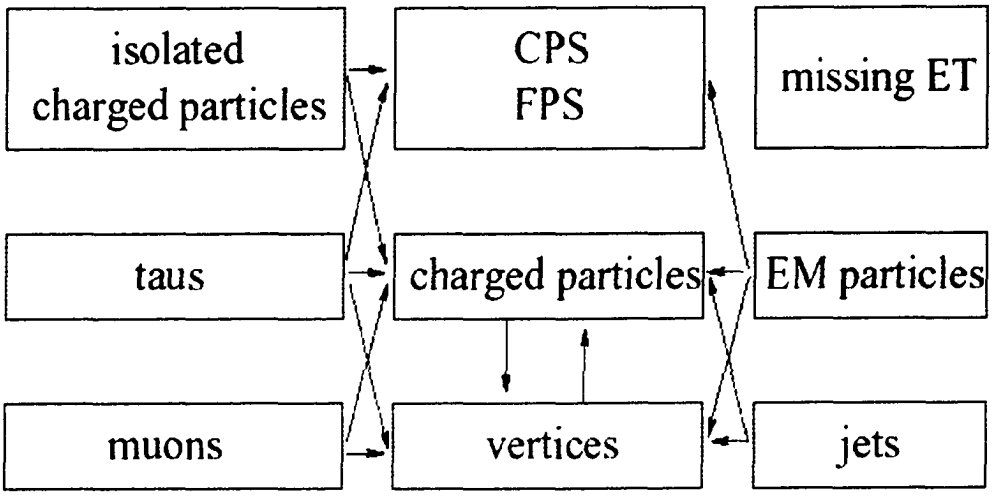


Figure 19.Links between Objects

Arrows in Figure 19 should be interpreted this way: the arrow pointing to B from A indicates B is the information source for A.

3.4.3 ThumbNail ROOT tree generation

In order to optimize the sampling weights for the calorimeter, TMB ROOT trees are needed with the following information

- CAL (calorimeter information, i.e. cell energy)
- EMPART_S(EM clusters using "simple cone algorithm")
- Track information (track momentum)

The interface software package tmb_analyze is used to produce the ThumbNail ROOT tree from ThumbNail data files. The following procedure generates a ROOT tree from the input ThumbNail file.

1> add the necessary packages:

>> setup D0RunII p13.06.01

>> setup d0tools

>> setup d0cvs

>> newrel -t p13.06.01 p130601

>> cd p130601

>> addpkg em_util v00-02-74

>> addpkg emcandidate v00-01-03

>> gmake em_util.all EMUTIL_SHARED=1

>> gmake emcandidate.all EMUTIL_SHARED=1

then copy to the working directory from lib/Linux2.4-KCC_4_0

libem_util.so and libemcandidate.so

>> cp ./lib/Linux2.4-KCC_4_0/.so .*

next, link directory "macro" in the local release to current directory.

>> ln –s /d0dist/dist/releases/p13.06.01/tmb_analyze/macros/macros .

>> cp macros/MakeTMBTreeClasses_so.C .

Edit MakeTMBTreeClasses_so.C to ensure it loads libem_util.so and then

>> d0setwa

>> root

>> root[] .x MakeTMBTreeClasses_so.C

>> root[] .q

after this, there should be a MakeTMBTreeClasses_so in the current directory.

Now that the framework is set up, the next step is

2> generate ROOT tree "tmb_tree.root

First, download the thumbnail file from SAM using script get_file.py,

then copy the necessary rcp file to the current direcory

>>cp /d0dist/dist/releases/p13.06.01/tmb_analyze/rcp/runTMBTreeMaker.rcp .

then create a text file list which contains the absolute

address of the thumbnail file on the disk. This is the input file. Then create a macro to run rund0exe (mine is "run"). In "run" the -name parameter specifies the directory which will hold the ouptut file tmb_tree.root, so named by default. One may specify the number of events you want to run by changing the value of parameter –num. If you don't use -num, rund0exe will run over all the events.

Now, we are ready to produce tmb_tree.root

>> chmod +x run

>> ./run

The following is an image of the structure of tmb_tree.root when opened in a TBrowser, which is especially designed for ROOT files.

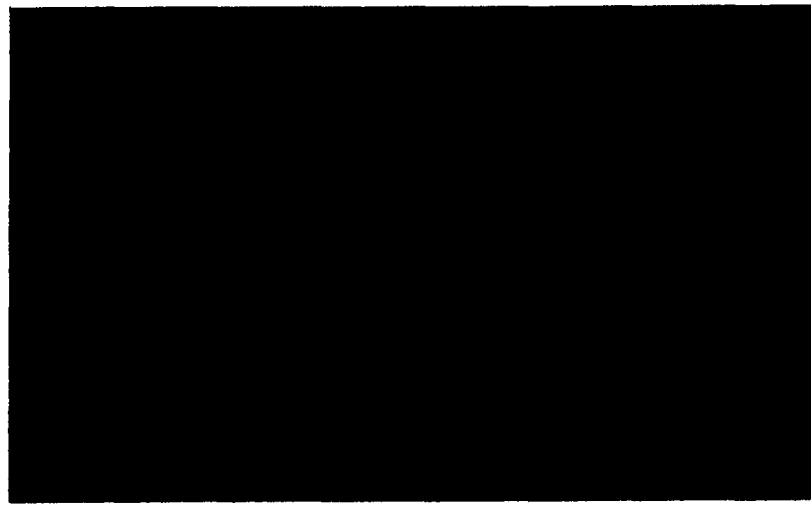


Figure 20.Structureof tmb_tree.root taken from TBrowser's window

The Thumbnail ROOT tree contains branches and leaves. The left half window displays all the branches for each of which there is a associate C++ class; The right half window contains all the leaves corresponding to that branch, and these leaves are data members in that particular C++ class.

CHAPTER FOUR

DATA REDUCTION AND ANALYSIS

4.1 E/P

In high energy physics, the relationship between the total energy E, the momentum p and the rest mass m of a particle is

$$E^2 = p^2c^2 + m^2c^4$$

It is convenient to measure momentum in units of GeV/c and mass in units of GeV/c^2 (1 eV/c^2 = 1.78 * 10^{-36}kg). Charges are measured in terms of e, the electronic charge (e = 1.6 * 10^{-19}C). Energy is measured in GeV, 1 GeV = 10^9eV. It is a convention that we let c = 1 in high energy physics, then E, p and m all have the same measurement unit (GeV) and the formula is reduced to $E^2 = p^2 + m^2$. We know that the rest mass of an electron is 0.000511GeV, a very, very small number compared to the energy of a particle (45GeV, for example); thus, $E^2 \approx p^2$ or $\frac{E}{p} \approx 1$.

Data samples are from data sets defined on SAM; these data have been processed with DØ Run II product P13.05.00. Table 3 on the next page lists the information of these data.

To determine the sampling weights, we use Drell-Yan events. In particle physics, events with two electrons and little jet activity are typical of a class of physics final states called Drell-Yan events. $Z \dashrightarrow e^+e^-$ decay is a Drell-Yan event; it will produce a final state with two high P_T electrons which are highly isolated.

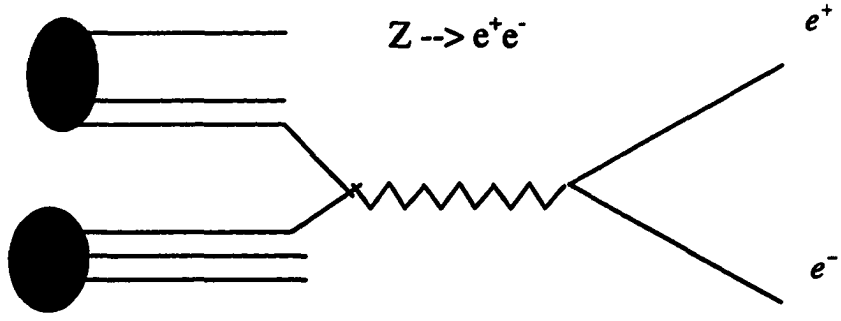


Figure 21.Drell-Yan Events

All file names in the table 3 should be prefixed by "WZskim-emStream-" and suffixed by ".raw_p13.05.00".

Table 3 Data sample

File name	Run number	Event number	Total events
20021222-001143	169236-169260	2532420-523772	42697
20021222-002907	169236-160260	5054701-2169242	43564
20021221-114027	169687-169708	6250108-5853750	41024
20021221-115845	165008-169247	16042947-13848972	44316
20021221-183224	165239-169172	94069883-5539925	45309
20021221-185419	165241-169172	98730188-3302270	45166
20021221-191841	169171-169172	374549-6180242	40409
20021221-193921	169171-169265	1888397-34207451	41013

20021221-201615	169192-169201	7056699-9259365	42465
20021221-203350	169183-169224	13396359-917658	43563
20021221-211034	169190-169236	20603246-3988468	44710
20021221-212921	169192-169248	25785120-18256883	45532
20021221-214750	169200-169246	288480-6276806	40678
20021221-220612	169200-169261	1821591-7702448	41205
20021221-222420	169200-169260	3504352-2394781	41662
20021221-224143	169201-169248	6068719-22761772	42498
20021221-233546	169224-169260	4458634-2540010	45220
20021221-235351	169226-169260	8987703-2601280	42995

The source of data samples which contain Drell-Yan events is WZ group defined SAM dataset wzskim1305.em [17]. In this section, only a fraction of wzskim1305.em samples are used to illustrate the electron selection cuts.

The first step to selecting good electrons is to identify the electromagnetic objects (EM objects) among all possible particle objects from detector data, based on a set of EM quality cuts.

4.2 EM Quality Cuts – Select EM Candidate

The detector electronic system records all necessary information associated with each event. From this information, we can rebuild the charged particle tracks and reconstruct the vertex for that event. Along with other information such as the EM cluster, we can construct an EM candidate and, at the same time, decide whether this candidate is a valid

one. The two graphs below present calorimeter energy distribution before and after quality cuts. The total entry is 357045. After quality cuts, a lot of background information was removed, leaving only EM candidates left, this is especially clear in the 15-40GeV area. The total entry for the EM candidate is 62654. This EM quality cuts reduced the size of data by 82.45%.

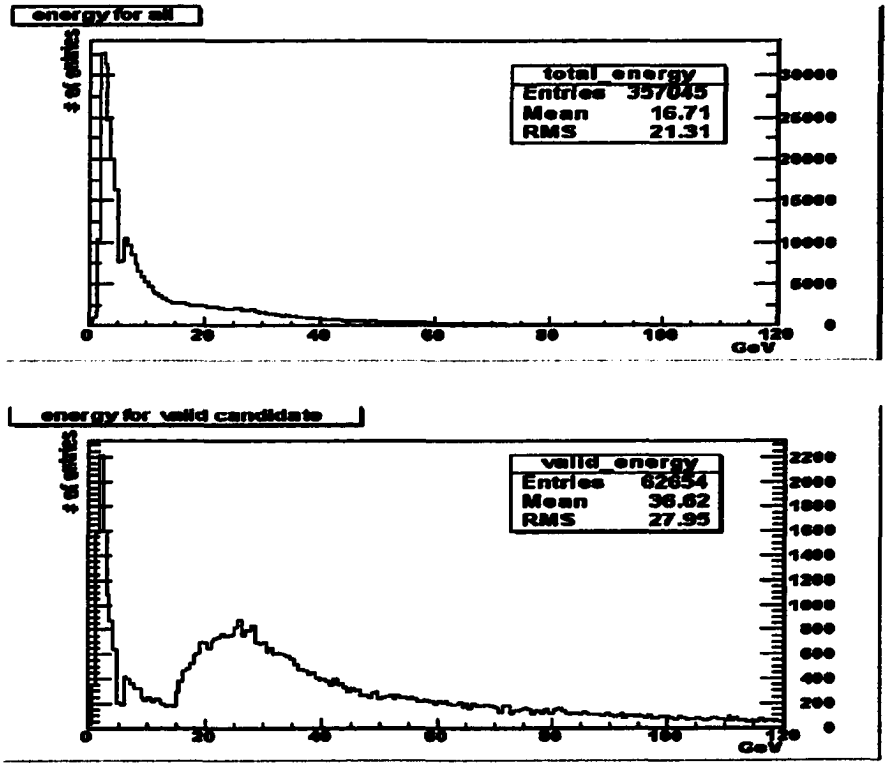


Figure 22.Calorimeter Energy Distribution

Upper plot --- energy distribution for all event entry

Bottom plot --- energy distribution for EM objects

If we compare the statistical data of the above distribution, we find that the mean energy of EM objects is much higher than that of total energy distribution. This fact gives us the information that the background has much lower average energy, and we may use this information to make energy cuts to help select EM objects.

4.3 Good Electron Cuts – Select Good Electrons

After performing EM quality cuts, we can make further, stricter cuts on EM objects to select good electrons. The following electron selection cuts are customized cuts based on EM-ID group's standard selection cuts [18].

- iso < 0.15;
- id = 10 or 11 or -11;
- HMx8 < 15;
- E > 20 GeV;
- EMfrac > 0.9;
- Has Track Match.

The following section is dedicated to the explanation of these cuts [19].

iso --- iso = isolation for cluster selection: $\frac{EisoTot - EisoCore}{EisoCore}$,

where *EisoTot* is overall (EM + hadronic) tower energy in a (η, ϕ) circle of radius = 0.4 centered on the highest pT tower; *EisoCore* is the EM tower energy in a (η, ϕ) circle of radius = 0.2 centered on highest pT tower.

id --- EMparticle id.

Abs(id) = 11 indicates:

- ▷ Final Cluster seed = Initial Cluster; pass EMfraction, pT and isolation criteria; has an associated SEMparticle.
- ▷ Final Cluster seed = Initial Cluster; pass isolation criterium; has an associated SEMparticle; doesn't pass EMfraction and pT criteria.

id = 10 indicates:

▷ Final Cluster seed = Initial Cluster; pass EMfraction, pT and isolation criteria; doesn't have an associated SEMparticle.

HMx8 --- A Fisher determinant of a 8x8 matrix which characterizes the shape of the EM particle showers.

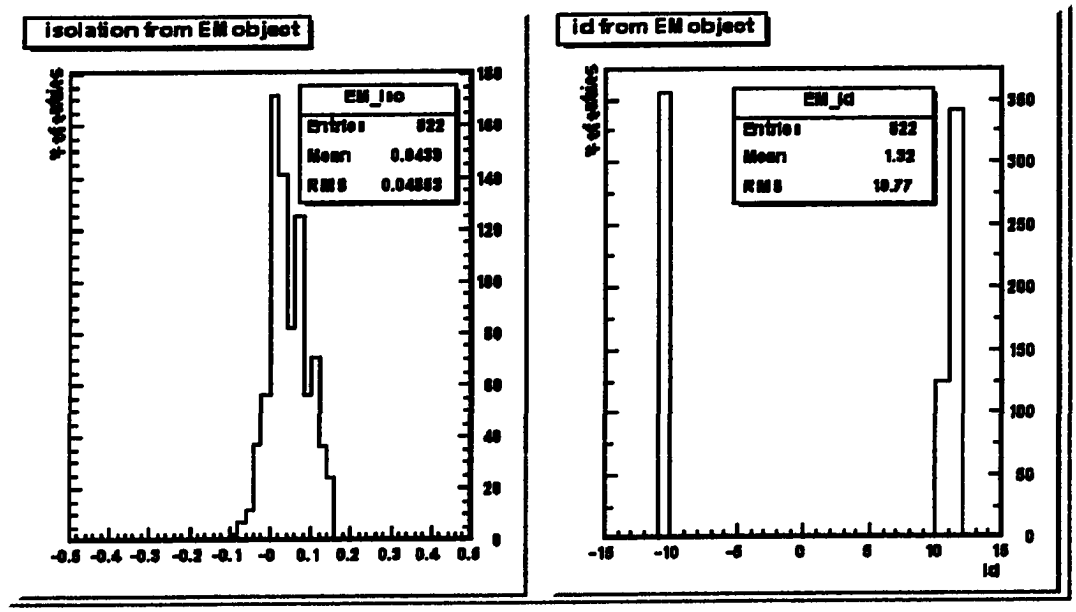
E --- Calorimeter energy.

EMfrac --- cluster EMfraction.

$$= \frac{PS+EM1+EM2+EM3+EM4}{PS+EM1+EM2+EM3+EM4+HA} = \frac{\sum_{i=0}^{4} floorE(i)}{HA+\sum_{i=0}^{4} floorE(i)}$$

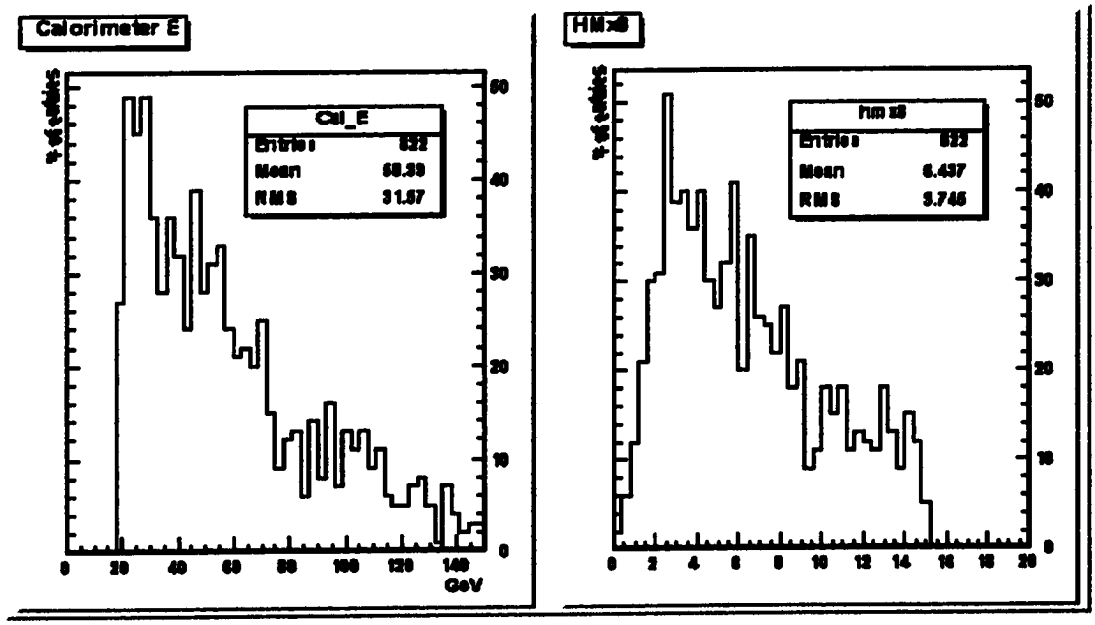
Has Track Match --- The EM particles develop tracks when they go through the electromagnetic Field, and one of the tracks will match the EM cluster so that we can identify the cluster associated with the EM particle.

After we applied those electron selection cuts to the valid EM candidates we obtained through EM quality cuts, we got 822 electrons. That is a $1 - \frac{882}{62654} = 98.6\%$ reduction in data size. Figures 23-25 provide the plots for these selection cuts after applying them to valid EM candidate sample.



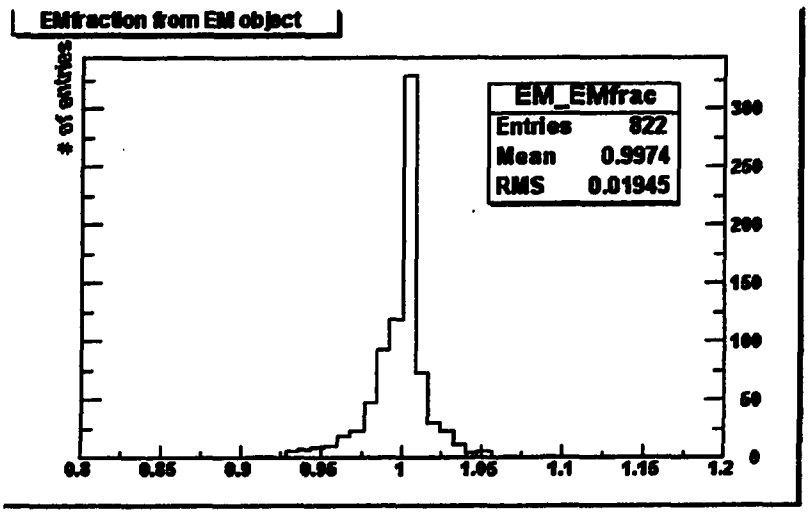
iso id

Figure 23.Electron selection cuts on iso and id



E HMx8

Figure 24.Electron selection cuts on E and HMx8



EMfrac

Figure 25.Electron selection cuts on EMfrac

4.4 Multi-parameter Fitting

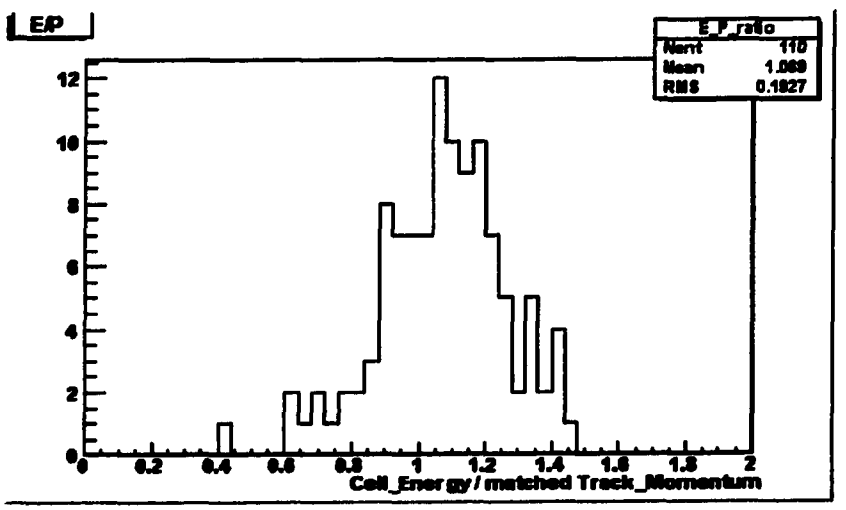


Figure 26.E over P

Figure 26 is drawn from a small sample of good electron data; the above plot shows that E–P ratio is approximately one.

In the expression $\frac{E}{p}$, p is the particle momentum, and since the detector's tracking system provides very good independent measurement for this parameter, it can be easily,

precisely determined. E is the energy the particle deposits in the calorimeter, so we can construct the energy using the model we established earlier: $\mathbf{E = \sum_i \sum_k C_i * W_i * A_{ik} * \varepsilon_{ik}}$. So E is the summation of all energies from EM1—EM4 (electromagnetic layers), FH1 (Fine Hadronic layer 1), ICD, CCMG (Central Calorimeter Massless Gap) and ECMG (End Calorimeter Massless Gap) with the sampling weights unknown.

Sampling weights are determined through the procedure of the χ^2 minimization. The formula is

$$\chi^2 = \sum \frac{(\frac{E}{p} - 1)^2}{\sigma^2}, \quad \text{where} \quad \frac{\sigma^2}{(E/p)^2} = \frac{\sigma^2(E)}{E^2} + \frac{\sigma^2(p)}{p^2} \quad \text{with} \quad \sigma(E) = \frac{0.15}{\sqrt{E}},$$

$\sigma(p) = 0.02 + 0.002p$ [9].

In the Chi-Square formula, only the sampling weights are unknown and thus, the minimization will result in a set of sampling weights.

The minimization procedure and result will be presented in next chapter.

CHAPTER FIVE

FITTING AND RESULTS

5.1 Fitting Procedure

TMinuit class

Fitting within ROOT framework is based on the TMinuit class. The ancestor of the TMinuit class, MINUIT, is a software package written in FORTRAN by Fred James [20], which has been converted to a C++ class called TMinuit.

The TMinuit class acts on a multiparameter FORTRAN function to which one must give the generic name FCN. There are two ways to set the FCN function: the function FCN is defined via the TMinuit SetFCN member function whenever the "histogram.Fit" command is invoked, users can also define their own FCN function to fit their needs. A user-defined FCN was used in this document. Once FCN is defined, it is the task of TMinuit to find those values of the parameters which give the lowest value of chi-square.

For demonstration purposes the following example illustrates how to use TMinuit to do chis-quare minimization.

Example: suppose we have a set of measurement y_i with errors σ_i corresponding to n data points x_i. Our goal is to find a function $f(x;a_1,a_2, ...,a_m)$ that best fit the y_i measurements. Here f depends on both x and a set of m parameters known as "fit

parameters". A typical solution to find the fit parameters a_j (j=1 to m) is known as the *Least Square Method.*

$$S = \sum_{i=1}^{n} [\frac{y_i - f(x_i; a_j)}{\sigma_i}]^2$$

The function S is related to another function called Chi-square:

$$\chi^2 = \sum_{i=1}^{n} (\frac{x_i - \mu_i}{\sigma_i})^2$$

They are often referred to interchangeably. We form a sum over all measurements that is proportional to the difference between the data points y_i and the predicted value of y for that value of x. Then TMinuit will vary the parameters a_j until S reaches a minimum. The data is given by the following table; the goal is to find the parameter a,b in $f(x) = ax+b$ that best fits the y values.

Table 4 Data table

x	0	1	2	3	4	5
y	0.92	4.15	9.78	14.46	17.26	21.9
σ	0.5	1.0	0.75	1.25	1.0	1.5

The FCN function *fcn*() and the function to do fitting are listed below.

FCN:

```
Float_t y[6],x[6],errory[6];
//==========================
void fcn(Int_t &npar, Double_t *gin, Double_t &f, Double_t *par, Int_t iflag)
{
```

```
const Int_t n = 6;
Int_t i;
//calculate chisquare
Double_t chisq = 0;
Double_t delta;
for (i=0;i<n; i++) {
   delta  = (y[i]-func(x[i],par))/errory[i];
   chisq += delta*delta;
  }
 f = chisq;
}

Double_t func(float x,Double_t *par)
{
 Double_t value= par[0]*x + par[1];
 return value;
}
```

FIT function

```
void lfit()
{// The errors on y values
      errory[0]=0.5;
      errory[1]=1.0;
      errory[2]=0.75;
```

```
errory[3]=1.25;
errory[4]=1.0;
errory[5]=1.5; // the x values
x[0]=0;
x[1]=1;
x[2]=2;
x[3]=3;
x[4]=4;
x[5]=5; // the y values
y[0]=0.92;
y[1]=4.15;
y[2]=9.78;
y[3]=14.46;
y[4]=17.26;
y[5]=21.9;
TMinuit *gMinuit = new TMinuit(3); //initialize TMinuit
with a maximum of 3 parameters
gMinuit->SetFCN(fcn);
Double_t arglist[10];
Int_t ierflg = 0;
arglist[0] = 1;
gMinuit->mnexcm("SET ERR", arglist ,1,ierflg);
// Set starting values and step sizes for parameters
```

```
  static Double_t vstart[2] = {4,2};
  static Double_t step[2] = {0.1,0.1};
  gMinuit->mnparm(0, "a", vstart[0], step[0], 0,0,ierflg);
  gMinuit->mnparm(1, "b", vstart[1], step[1], 0,0,ierflg);
// Now ready for minimization step
  arglist[0] = 500;
  arglist[1] = 1.;
  gMinuit->mnexcm("MIGRAD", arglist ,2,ierflg);
// Print results
  Double_t amin,edm,errdef;
  Int_t nvpar,nparx,icstat;
  gMinuit->mnstat(amin,edm,errdef,nvpar,nparx,icstat);
  gMinuit->mnprin(3,amin);
}
```

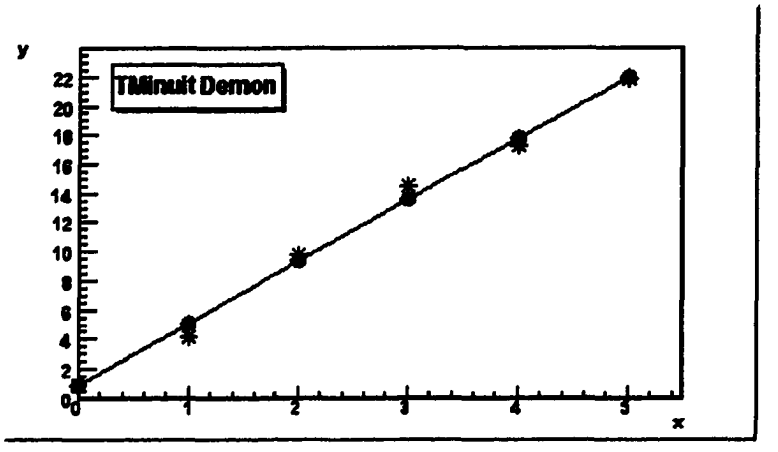
After running Ifit() in ROOT, we have the optimized values for a and b.

$$a=4.22694, \qquad b=0.879203.$$

Then the fitting function is

$$f(x) = 4.22694x+0.879203$$

In Figure 27 a plot of the data points (x_i, y_i) and the fitting function $f(x)$ demonstrates the functionality of TMinuit.



* data points (x_i, y_i)

$f(x)$

Figure 27.TMinuit Demo: Data Points vs. Fitting Function

5.2 Fitting Results

Sampling weights for Central Calorimeter

Both detector real data and MC simulation data are used in the calculation of sampling weights. The detector data used for the determination of the sampling weights for the CC region is from the SAM dataset definition "wzskim1305.em" which was defined by WZ group [21]. A total number of 774589 events were processed and about 14000 good electrons [22] were selected as CC electrons.

After feeding the electron information into a TMinuit embedded C++ macro, the output CC sampling weights are listed in the following table:

Table 5 Sampling weights for Central Calorimeter

EM1	EM2	EM3	EM4	FH1
0.55	1.267	0.677	2.42	0.022

Since the detector data already contained the sampling weights information when they were reconstructed, the sampling weights in the table are relative values. That means if the old sampling weights are perfect, then all the new sampling weights in the table would be unity or equal to one. A value less than one indicates that the original sampling weight for that layer is over weighted; on the other hand, if a sampling weight value is greater than one, it means the original sampling weight for that layer is under weighted.

The related computer programs and macros are listed in appendix B.

For Monte Carlo simulation data, SAM data set "p13-05-certification-z-zee" was used for the same C++ macro. The sampling weights for CC region using Monte Carlo data are listed in table 6, table 7 is the sampling weights for all the layers using the same set of Monte Carlo data.

Table 6 Sampling weights for CC from Monte Carlo data

EM1	EM2	EM3	EM4	FH1
0.903	0.651	1.286	0.725	5.395

Table 7 Sampling weights for all layers from Monte Carlo data

EM1	EM2	EM3	EM4	FH1	ICD	CCMG	ECMG
0.827812	1.03701	1.20386	0.79085	5.11584	17.4084	2.41062	12.9962

A discussion will be given in the last chapter, conclusion.

CHAPTER SIX

MONTE CARLO SIMULATION

6.1 Monte Carlo Simulation

Monte Carlo (MC) at Fermilab is a set of algorithms and programs which fully simulate the response of the DØ detector. MC simulation involves several different steps, each of which has a software package dedicated to it. The minimum steps to generate a root-tuple* for physics analysis are listed below [23].

Generator->D0GSTAR->D0Sim->D0RECO->RECO_ANALYZE

MC events from a generator need to be processed first by d0gstar and then by d0sim to make them look like real raw data events. D0RECO rebuilds both MC and Raw Data into events for analysis. RECO_ANALYZE runs on RECO output to make root-tuples for physics analysis.

Generator

MC events are generated using event generators. There are several different generators for different simulation purposes; PYTHIA and ISAJET are two popular ones. PYTHIA

is a program for the generation of high energy physics events; it describes the collisions between the fundamental particles such as e^+, e^-, p and $\bar{p}$ in various combinations.

ISAJET is a Monte Carlo program which simulates $p-p$, $\bar{p}-p$, and $e-e$ interactions at high energies.

D0GSTAR

D0gstar stands for **D0GEANT** Simulation of the Total Apparatus Response. The purpose of this simulation package is to offer a central facility for Monte Carlo studies of the D0 detector with different configurations. It allows a non-specialist to have a full GEANT simulation of the apparatus with a simple interface, at the same time providing the specialist with a full description of all the sub-detectors [23]. Figure 28 – Figure 30 are plots of the detector generated by D0GEANT using single particle electron, muon and pion, respectively.

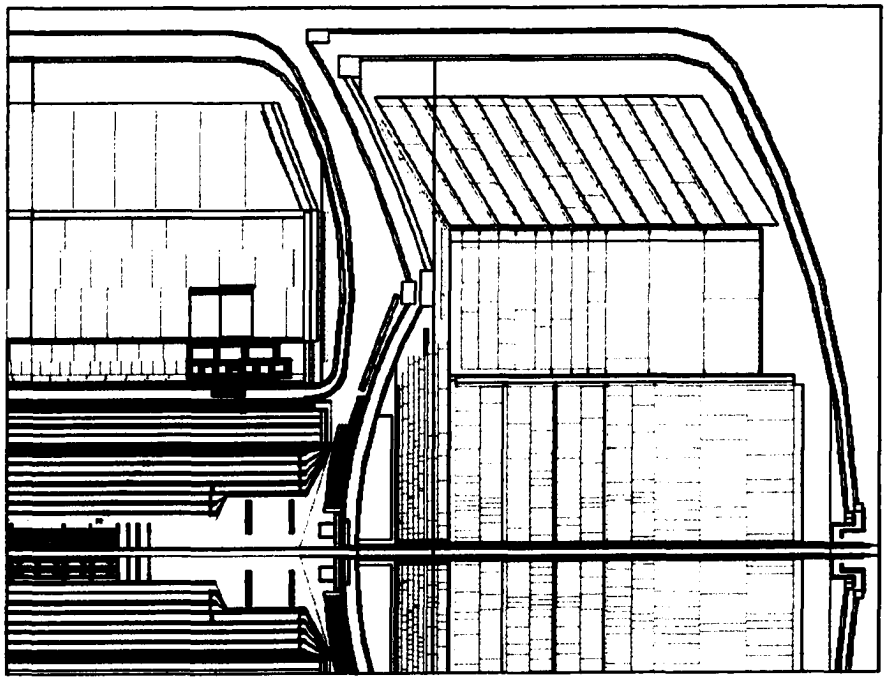


Figure 28(1).Detector plot generated by GEANT—electron CC region

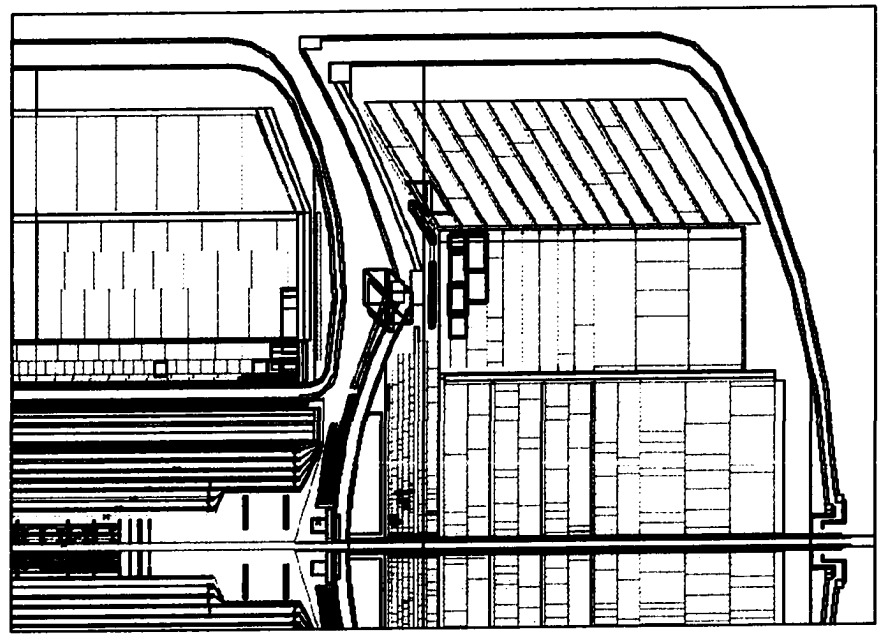


Figure 28(2).Detector plot generated by GEANT—electron CC, ICD and EC region

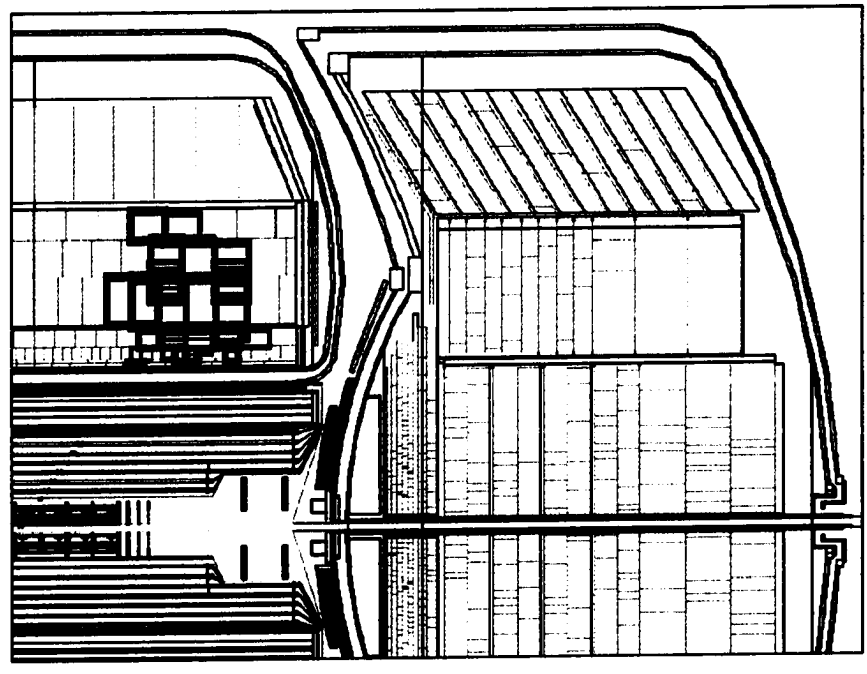


Figure 29(1).Detector plot generated by GEANT---Pion CC region

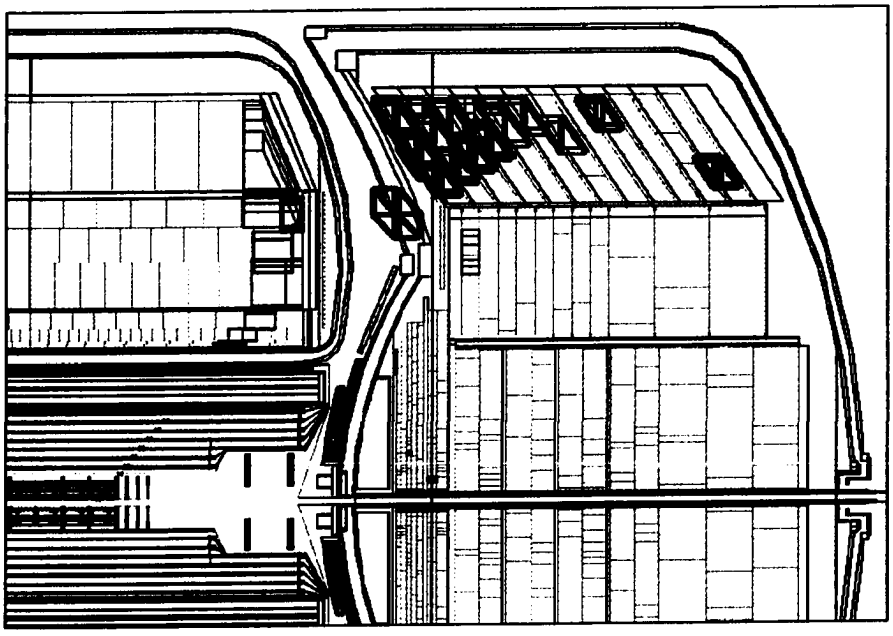


Figure 29(2).Detector plot generated by GEANT---pion CC, ICD and EC region

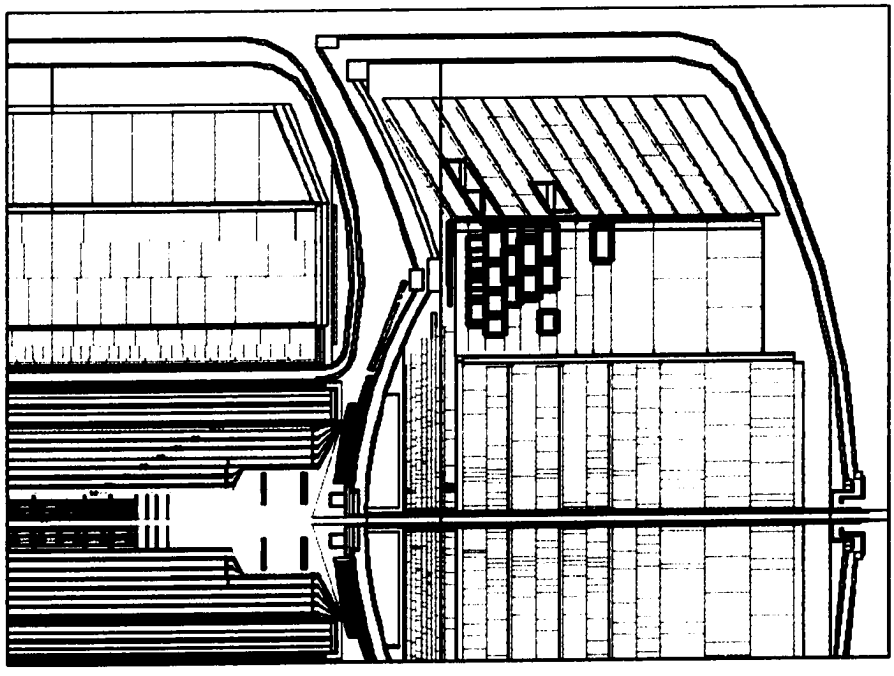


Figure 30.Detector plot generated by GEANT---pion EC region

D0Sim

D0sim takes D0gstar's output as input and does the digitization for each sub-detector and the raw data simulation. The outputs of D0sim are files that are suitable for the input of next step, d0reco. D0Sim performs the following functions:

Merge hard scatter and minbias events

Add calorimeter pileup from previous events

Make L1CalTTowerChunk for L1 simulation

Add calorimeter noise

Add SMT noise and inefficiencies

Add CFT noise and inefficiencies

Add Muon noise and inefficiencies

Make a RawDataChunk

Drop all D0GStar chunks

Monte carlo program structure

D0GSTAR generated detector plot

D0RECO

D0RECO is D0's official offline reconstruction program. It is responsible for converting Raw Data into Physics Objects usable by all D0 physics analyses. It can process both real and simulated raw data, and produces output in the form of DSPACK files containing EDM Chunks (described in the next chapter). D0RECO consists of many separate software packages. The program performs the following logical steps:

1> unpack the raw data,

2> reconstruct detector-specific clusters / hits, applying calibration / alignment constants,

3> find global tracks candidates,

4> find primary and secondary vertex candidates,

5> find Physics Object candidates (ChargedParticle, EM, TAU, etc.).

RECO_ANALYZE

RECO_ANALYZE takes the DSPACK files, the output of D0RECO program, as input and produce single root-tuple (or tree) with separate branches associated with the various components of D0RECO. These ROOT-tuples are ready to be used by all D0 users. Like D0RECO, RECO_ANALYZE also consists of many separate software packages associated with the different branches on the resultant ROOT tree.

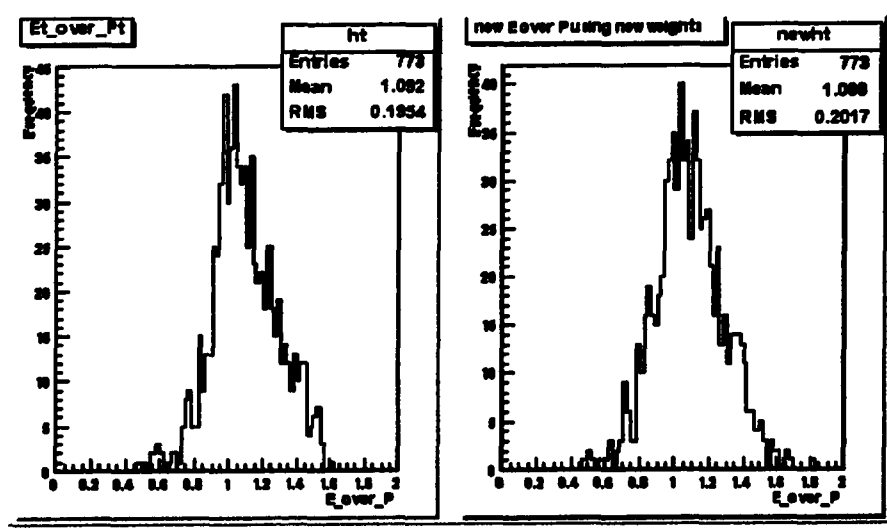
CHAPTER SEVEN

CONCLUSIONS

In summary, this work studied the different parts of the DØ detector calorimeter. A model was set up to calculate the calorimeter energy and a multiparameter fitting formula was developed to optimize the sampling weights of calorimeter. Both real data collected from DØ detector and data generated from the Monte Carlo simulation of the detector were used to calculate the energy sampling weights for the EM calorimeter (EM1 through EM4 and FH1). The reconstructed energy E was calculated based on the energy reconstructon model, $E = \sum_i \sum_k C_i * W_i * A_{ik} * \varepsilon_{ik}$. New sampling weights were obtained for each set of data by conducting the Chi-square minimization according to the formula $\chi^2 = \frac{(\frac{E}{P} - 1)^2}{\sigma^2}$. To evaluate the new sampling weights, the E/P plots have been redrawn using the new sampling weights and existing sampling weights. The conclusion is that the energy reconstructed using new sampling weights match very well the energy reconstructed using existing sampling weight for electrons in both real data and Monte Carlo data. This verified, for the first time using real detector data, that existing sampling weights are optimized. Monte Carlo data was also used to calculate the sampling weights for ICD, CCMG, and ECMG regions. The sampling weights are very different from those for CC region and those existing sampling weights for ICD, CCMG and ECMG region,

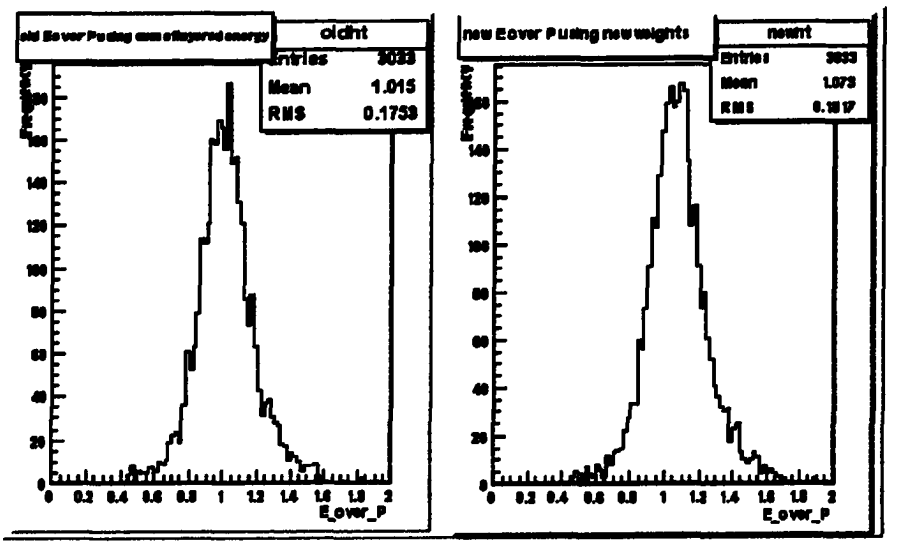
but currently there is no means to evaluate them due to the difficulty of obtaining real data (electron) from those regions.

Figure 31 plots the E/P ratio for the CC region using both existing sampling weights and new sampling weights obtained from real data; Figure 32 plots the E/P ratio for CC, ICD, CCMG and ECMG regions using the existing sampling weights and new sampling weights obtained from Monte Carlo data. By comparing the plots using existing sampling weight with those using new sampling weights, we notice that the mean and RMS from new sampling weights are very close to their counterparts using the existing sampling weights. Although the E/P ratios are very close to each other, the new sampling weights for the CC region using the real data are quite different from those existing ones. The reason for this discrepancy is that more than one minimization scheme leads to the same optimized results.



Existing sampling weights New sampling weights

Figure 31.E over P for real data (CC region)



Existing sampling weights New sampling weights

Figure 32.E over P for Monte Carlo data

Future work includes using additional detector information (central preshower information, for example) to further purify the electrons. By doing so, we can improve the electron efficiency in our data sample and, at the same time, reduce the background noise and thus obtain a better set of sampling weights. Another objective is to redo the multiparameter optimization using real data from ICD region once they are available. Some other work is also needed to re-identify the electrons in ICD region which have been mistaken as taus or jets.

APPENDIX A

GLOSSARY

ADC ----Analog-to-Digital-Conversion. It is an electronic process in which a continuously variable signal is converted, without altering its essential content into a multi-level digital signal.

BLS --- Base Line Subtractor. It is an electronics system sitting between pre-amps and the ADCs in the calorimetry electronics. The output signals from pre-amps have a long falling time so that an output signal will overlap with the tail of the previous signal, the function of BLS is to setup a threshold to Remove the overlap.

Calorimeter --- A device in particle physics to measure the energy deposited in it by particles.

CERN --- CERN is the European Organization for Nuclear Research, the world's largest particle physics centre. CERN explores what matter is made of, and what forces hold it together.

CFT ---Central Fiber Tracker

CPS --- The short term for Central Pre-shower Detector. It is a device to track and identify electrons and photons in the central region of D0 detector.

D0GSTAR --D0 Geant Simulation of the Total Apparatus Response. It is a FORTRAN-based program which can simulate the D0 detector Response based on GEANT (below).

D0sim --- ---A Program that simulate the D0 electronics. It takes D0GSTAR output as input and does the digitization for each detector, pileup and raw data formatting.

DSPACK --It is a shared memory based client/server data manager that allows Independent processes to create and access shared objects which can be defined by the programmer as C-like data structures at run time.

Electron --- The lightest elementary particle, it has one negative charge and a spin of ½. It is a stable particle.

EM calorimeter --- Electromagnetic Calorimeter. Calorimeters are usually composed of different parts. EM Calorimeter is the part through which the energy of incident Electromagnetic particles is fully absorbed.

em_util --- -A software package that helps construct and analyze the EM candidate objects.

Eta (η) ---- Eta is defined as ln ($\tan\theta/2$), where θ is the angle of cluster track with

Respect to the beam direction (+z).

Fermilab --It is a particle physics national laboratory of Department of Energy. One of the major high energy physics labs in the world. It has highest energy accelerator in the world.

GEANT ---Detector simulation program developed at CERN.

Hadron --- Composite particle made of quarks and gluons. Hadrons have no strong charge but they participate in residual strong interaction due to the strong charges of their component particles.

ICD ------- Short term for Inter Cryostat Detector, a layer of calorimetry between the D0 cryostats.

ICR --- -----Short term for Inter-Cryostat Region, it refers to the region of $0.8 \leq |\eta| \leq 1.5$, where the central and end cryostats overlap.

ISAJET --- A Monte Carlo program which simulates pp, $p\bar{p}$, e^+e^- interactions at high energies. ISAJET is based on perturbative QCD plus phenomenological models for parton and beam jet fragmentation.

Likelihood fitting --- A scientific method to estimate the model parameters underlying the observed data. The principal is to find parameters that make the observed data most likely.

mc_runjob --Python scripts designed to aid in running MC and other executables.

MINUIT --- A software tool to find the minimum value of a multi-parameter function and analyze the shape of the function around the minimum. The principle application is foreseen for statistical analysis, working on chisquare or log-likelihood functions, to compute the best-fit parameter values and uncertainties, including correlations between the parameters.

Monte Carlo simulation --- Simulations of physics processes (particle collisions, detector responses.) based on sampling random numbers.

Muon ------ An unstable second-generation lepton, having a mass 207 times that of an electron and a mean life time of 2.2e-6.

Neutrino --- A stable fundamental particle with no mass (or very small mass) and no charge. It has a spin of ½. There are six different types of neutrinos.

Phi (ϕ) --- --Angle of the projection of a cluster (or track) in the plane perpendicular to the beam direction.

Photon --- A particle with zero rest mass. It is a carrier particle of the electromagnetic depending on their frequencies, photons have different names such as visible lights, x-rays, and gamma-rays.

Pion ------An elementary particle classified as a meson. It exists in three forms: neutral, positively charged and negatively charged. The charged pions Decay into muons and neutrinos; the neutral pion decays into two gamma-ray photons.

PMCS --- ----Parameterized Monte Carlo Simulation is a fast Monte Carlo simulation Program. PMCS is a factor of 2000 faster than D0GSTAR.

Proton --- One of hadrons. It has a basic structure of two up quarks and one down quark. It has one positive charge.

PYTHIA -- A computer program for the generation of high energy physics events. i.e. for the description of collisions at high energies between elementary particles such as e+, e-, p and pbar in various mbinations.

RECO --- --D0 offline reconstruction program. It is responsible for reconstructing objects that are used to perform all DØ physics analyses. It is a CPU intensive program that processes either collider events recorded during online data taking or simulated events produced with the DØ Monte Carlo (MC) program.

RECO_ANALYZE ---It takes as input the *DSPACK* files (with all of the associated reconstructed *Chunks*) generated by *RECO*, and produces a single ROOT "Ntuple" (Tree), with separate branches associated with the various components of *RECO*.

ROOT --- A scientific program developed at CERN. It is widely used by major high energy physics and nuclear laboratories around the world. Its programming interface is based on C++.

ROOT-tuple --- It is a data structure or data format designed to hold the output data of RECO_ANALYZE. A ROOT-tuple (ROOT file) contains many branches each of which has a associated software package by which it is generated. ROOT-tuples can be read with a specially designed browser and can be analyzed by ROOT macros.

SMT ---Silicon Microvertex Tracker

Tevatron ----high energy particle accelerator at Fermi National Accelerator laboratory.

Theta (θ) ---Angle between the cluster (or track) and the beam direction (+z).

Transverse Energy (E_t) --- Energy in the direction perpendicular to the beam direction.

For very high energy particles, $E \approx PC$ and $E_t \approx P_t$ in units where $C = 1$.

Transverse Momentum (P_t) --- Track momentum in the direction perpendicular to the beam direction .

APPENDIX B

ICD PROJECT AT LOUISIANA TECH UNIVERSITY

Intercryostat Detector (ICD) is a scintillator-based layer of Calorimetry of D0 Detector. It improves energy measurement in the intercryostat region between etas of 1.1 to 1.4, see figure below.

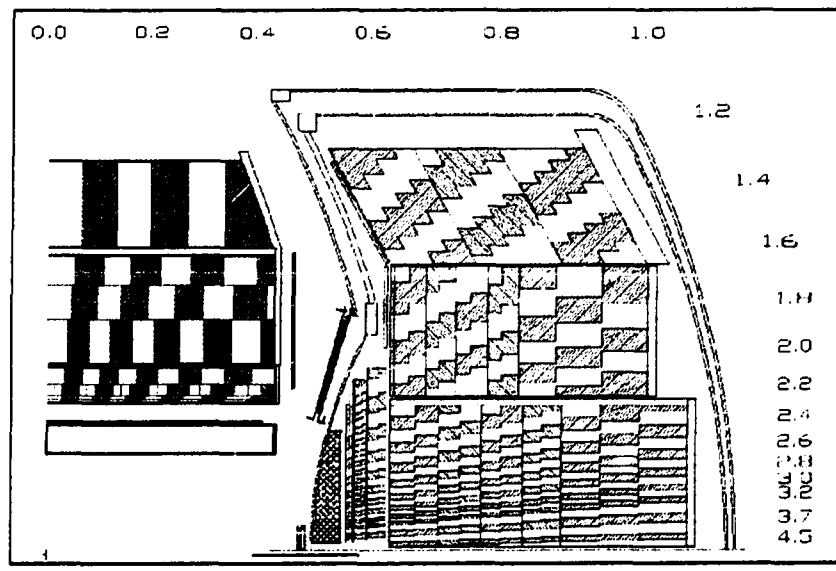


Figure 33.ICD covers the region in eta from 1.1 to 1.4

Light signals are converted into an electrical signal, amplified and shaped, and passed into the experiment readout. It is incorporated into the calorimetry of the D0 experiment at the Fermi National Accelerator Lab's Tevatron collider. ICD plays an important role in D0 calorimetry, both in term of measuring the energy of jets as well as in the calculation of missing transverse energy E_T. The combination of a rapidly changing material profile and extra uninstrumented material in the intermediate regions of the calorimeters leads to reduced sampling of showers, and hence a degradation of energy measurements for this region. The ICD restores energy resolution by providing additional sampling in this region. As a part of D0, the ICD will be used in searches for the Higgs, new phenomena, and precision measurements of the Standard Model. Louisiana Tech University is responsible for the design and the construction and testing of ICD electronics called "electronic drawers".

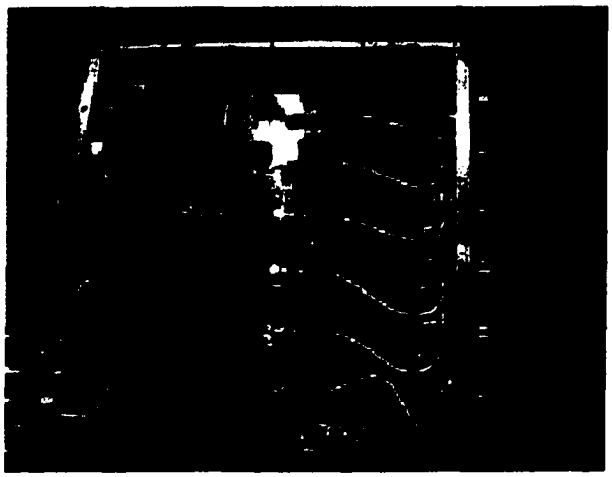


Figure 34.Electronic Drawers

In order to finish the project with high quality, a group of people were dedicated to the construction and testing. We had a testing system for the electronic drawer testing, we also implemented several other testing stands for the testing of electronic parts and components. I participated in both construction and testing phase. Most of my time was spent on testing the whole drawer and repairing the bad drawers and motherboard. Currently, I continue to do the maintenance job on these electronic drawers during the shutdown time of D0 detector.

APPENDIX C

PROGRAM LIST

The following are the C++ programs used to select good electrons, construct the ROOT trees and do the multiparameter fitting.

Qun.C and MCtmb355833.C are responsible for the selection of good electrons using real data and Monte Carlo data, respectively. Similarly, electron_tree.C and MCtmb355833_tree.C are used to construct electron ROOT tree from real data and Monte Carlo data, in that order. The program to do the fitting is presented in Chapter Five and will not be listed here.

```
/* ---------- Qun.C ----------*/
#define qun_cxx
#include "qun.h"
#include "TStyle.h"
#include "TCanvas.h"
#include "TRefArray.h"
#include "TString.h"
#include "TFile.h"
#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include "emcandidate/emcandidate/EMcandidate.hpp"
//#include <string>
using namespace std;
void  qun::Loop()
{
  if (fChain == 0) return;
  TFile *ff = new TFile("tmb_tree.root183224_plot.root","RECREATE","ROOT file for plots",1);
```

```
if(!ff->IsOpen())
 {
  cout<<"tmb_tree.root183224_plot.root is not open\n";
  cout<<"now I am opening the file"<<endl;
  ff->Open("tmb_tree.root183224_plot.root","RECREATE","ROOT file for plots",1);
 }
// write the information for good electrons to a text file
 e_output<<" good electron selection creteria\n\n"<<endl;
 e_output<<"=================================================="<<endl;
 e_output<<"\tCell energy greater than 20 GeV"<<endl;
 e_output<<"\tHMx8()<15, iso() < 0.15 and EMfrac() > 0.9 "<<endl;
 e_output<<"\tid() == 10 || TMath::Abs(id()) == 11"<<endl;
 e_output<<"\thas_track_match()"<<endl;
 e_output<<"==================================================\n\n"<<endl;
 e_output<<"The order of data :\n"<<endl;
 e_output<< "e_hmx8 | "<<"e_Cal_P | "<<"e_Cal_Pt | "<<"e_Cal_E | "<<"e_Cal_Et | "<<"e_Cal_PS
| "<<"e_Cal_EM1 | "<<"e_Cal_EM2 | "<<"e_Cal_EM3 | "<<"e_Cal_EM4 | "<<"e_Cal_FH1 |
"<<"e_Cal_Eta | "<<"e_Cal_Phi | "<<"e_Cal_Dtr_Eta | "<<"e_Cal_Dtr_Phi | "<<"e_EM_P |
"<<"e_EM_Pt | "<<"e_EM_E | "<<"e_EM_Et | "<<"e_EM_Eta | "<<"e_EM_Phi |
"<<"e_EM_EMfrac | "<<"e_EM_iso | "<<"e_EM_id\n\n"<<endl;

// histogram booking
TH1F * valid_eta= new TH1F("valid_eta","eta for valid candidates",100,-2,2);
valid_energy = new TH1F("valid_energy","energy for valid candidate",200,0,120);
total_energy = new TH1F("total_energy","energy for all",200,0,120);

CAL_E_over_P = new TH1F("CAL_E_over_P","E over P using CAL info",100,0,2);
EM_E_over_P = new TH1F("EM_E_over_P","E over P using EM info",100,0,2);
```

```
hmx8 = new TH1F("hmx8","HMx8",50,0,20);
Cal_P = new TH1F("Cal_P","Calorimeter P",50,0,150);
Cal_Pt = new TH1F("Cal_Pt","Calorimeter Pt",50,0,150);
Cal_E = new TH1F("Cal_E","Calorimeter E",50,0,150);
Cal_Et = new TH1F("Cal_Et","Calorimeter Et",50,0,150);
Cal_PS = new TH1F("Cal_PS","Calorimeter PS energy",50,0,20);
Cal_EM1 = new TH1F("Cal_EM1","Calorimeter EM1",50,0,150);
Cal_EM2 = new TH1F("Cal_EM2","Calorimeter EM2",50,0,150);
Cal_EM3 = new TH1F("Cal_EM3","Calorimeter EM3",50,0,150);
Cal_EM4 = new TH1F("Cal_EM4","Calorimeter EM4",50,0,150);
Cal_FH1 = new TH1F("Cal_FH1","Calorimeter FH1",50,0,15);
Cal_Eta = new TH1F("Cal_Eta","Calorimeter Eta",100,-2,2);
Cal_Phi = new TH1F("Cal_Phi","Calorimeter Phi",100,0,10);
Cal_Dtr_Eta = new TH1F("Cal_Dtr_Eta","Detector Eta",100,-1,2);
Cal_Dtr_Phi = new TH1F("Cal_Dtr_Phi","Detector Phi",100,0,10);

EM_P = new TH1F("EM_P","P from EM object",50,0,150);
EM_Pt = new TH1F("EM_Pt","Pt from EM object",50,0,150);
EM_E = new TH1F("EM_E","E from EM objects",50,0,150);
EM_Et = new TH1F("EM_Et","Et from EM object",50,0,150);
EM_Eta = new TH1F("EM_Eta","Eta from EM object",100,-2,2);
EM_Phi = new TH1F("EM_Phi","Phi from EM object",100,0,10);
EM_EMfrac = new TH1F("EM_EMfrac","EMfraction from EM object",50,0.8,1.2);
EM_iso = new TH1F("EM_iso","isolation from EM object",50,-0.5,0.5);
EM_id = new TH1F("EM_id","id from EM object",30,-15,15);
Int_t nentries = Int_t(fChain->GetEntries());
 //TString algo_name;
 Int_t counter=0;
```

```
Bool_t good;
Bool_t better;
Int_t my_entries = nentries;
cout<<" no. of entries "<< nentries<<endl;
  //  my_entries = nentries;
Int_t nbytes = 0, nb = 0;
for (Int_t jentry=0; jentry<my_entries;jentry++) {  //2
 Int_t ientry = LoadTree(jentry);
 //in case of a TChain, ientry is the entry number in the current file
 nb = fChain->GetEntry(jentry);   nbytes += nb;
 //Emcl
 // make a track collection
 TrackCollection tcoll;
 tcoll.Reset();
 for(Int_t i=0; i< fTrks->GetLast()+1; i++){
   EMTrack newtrack((TMBTrks*) fTrks->At(i));
   tcoll.AddTrack(newtrack);
 }
 for(Int_t i=0; i<fEmcl->GetLast()+1; i++) {    //3
   TMBEmcl* emcl = ( TMBEmcl* ) fEmcl->At(i);
    const char *algoname=emcl->algoname();
    char* c="s";
   total_energy->Fill(emcl->E());  // total energy
                          if(algoname[0]==c[0])
                           {                          //4
                           TMBVrts* vtx=emcl->GetVertex();
                           float vxyz[3]={0,0,0};
                           if(vtx)
```

```
{
vxyz[0]=vtx->vx();
vxyz[1]=vtx->vy();
vxyz[2]=vtx->vz();
}
EMcandidate emcand(emcl,fGlob,tcoll,vxyz,true,13,0);
if(emcand.is_valid_candidate())
{                                   //5
// cout<<"candidate is valid"<<endl;
  valid_eta->Fill(emcand.eta());
  counter++;

  valid_energy->Fill(emcl->E());  // energy for valid candidate
  if(counter%200==0)
  cout<<"valid candidate # "<<counter<<endl;
  // hPt->Fill(emcand.pT());
  good = emcand.E()>20 &&
  emcand.EMfrac()>0.9 &&
  (emcand.id()==10 || TMath::Abs(emcand.id())==11)&&
  emcand.iso()<0.15 &&
  emcand.HMx8()<15 &&   //15--tight,20--loose
  emcand.has_track_match();
  //require |eta|<0.8
better =( good &&TMath::Abs(emcand.eta())<=0.8);
if(good)
{                           //6
cout<<"good electron"<<endl;
Double_t cal_Eta = TMath::Abs(emcand.CalEta());
```

```
Double_t Cal_Theta = 2*TMath::ATan(TMath::Power(2.7182818,-cal_Eta));
Double_t cal_P = TMath::Abs(emcand.Calpt()/TMath::Sin(Cal_Theta));
hmx8->Fill(emcand.HMx8());
Cal_P->Fill(cal_P);
Cal_Pt->Fill(emcand.Calpt()); // (1/q/Pt) ?
Cal_E->Fill(emcand.CalE());
Cal_Et->Fill(emcand.CalE()*(TMath::Abs(TMath::Sin(Cal_Theta))));
Cal_PS->Fill(emcand.CalFloorE(PS)); //PS layer energy
Cal_EM1->Fill(emcand.CalFloorE(EM1));
Cal_EM2->Fill(emcand.CalFloorE(EM2));
Cal_EM3->Fill(emcand.CalFloorE(EM3));
Cal_EM4->Fill(emcand.CalFloorE(EM4));
Cal_FH1->Fill(emcand.CalFloorE(FH1)); //energy for FH1
Cal_Eta->Fill(emcand.CalEta());
Cal_Phi->Fill(emcand.CalPhi());
Cal_Dtr_Eta->Fill(emcand.CalDetectorEta());
Cal_Dtr_Phi->Fill(emcand.CalDetectorPhi());

// calculate theta first
Double_t em_Eta = TMath::Abs(emcand.eta());
Double_t EM_Theta = 2*TMath::ATan(TMath::Power(2.7182818,-em_Eta));
EMTrack *thisTrack=emcand.track_match();//pointer to EMTrack object
Double_t momentum_for_thisTrack = TMath::Abs(thisTrack->QOverPt_at_DCA());
momentum_for_thisTrack = 1/momentum_for_thisTrack;
Double_t em_Pt = momentum_for_thisTrack;
Double_t em_P = TMath::Abs(em_Pt/TMath::Sin(EM_Theta));
EM_P->Fill(em_P);
EM_Pt->Fill(em_Pt);
```

```
EM_E->Fill(emcand.E());
EM_Et->Fill(emcand.E()*(TMath::Abs(TMath::Sin(EM_Theta))));
EM_Eta->Fill(emcand.eta());
EM_Phi->Fill(emcand.phi());
EM_EMfrac->Fill(emcand.EMfrac());
EM_iso->Fill(emcand.iso());
EM_id->Fill(emcand.id());
// fill histogram
CAL_E_over_P->Fill(emcand.CalE()/cal_P);
EM_E_over_P->Fill(emcand.E()/em_P);
// assign electrons info to corresponding variables
                    e_hmx8 = emcand.HMx8();
                    e_Cal_P = cal_P;
                    e_Cal_Pt = emcand.Calpt();
                    e_Cal_E = emcand.CalE();
                    e_Cal_Et = emcand.CalE()*TMath::Abs(TMath::Sin(Cal_Theta));
                    e_Cal_PS = emcand.CalFloorE(PS);  // PS layer energy
                    e_Cal_EM1 = emcand.CalFloorE(EM1);
                    e_Cal_EM2 = emcand.CalFloorE(EM2);
                    e_Cal_EM3 = emcand.CalFloorE(EM3);
                    e_Cal_EM4 = emcand.CalFloorE(EM4);
                    e_Cal_FH1 = emcand.CalFloorE(FH1);  // energy for FH1 layer
                    e_Cal_Eta = emcand.CalEta();
                    e_Cal_Phi = emcand.CalPhi();
                    e_Cal_Dtr_Eta = emcand.CalDetectorEta();  // detector Eta
                    e_Cal_Dtr_Phi = emcand.CalDetectorPhi();
                    e_EM_P = em_P;
                    e_EM_Pt = em_Pt;
```

```
e_EM_E = emcand.E();
e_EM_Et = e_EM_E *TMath::Abs(TMath::Sin(EM_Theta));
e_EM_Eta = emcand.eta();
e_EM_Phi = emcand.phi();
e_EM_EMfrac = emcand.EMfrac();
e_EM_iso = emcand.iso();
e_EM_id = emcand.id();
// output to file electron_120513.dat
e_output<<setiosflags(ios::left)<<setw(12)<<e_hmx8;
e_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_P;
e_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_Pt;
e_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_E ;
e_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_Et ;
e_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_PS;
e_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_EM1;
e_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_EM2;
e_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_EM3;
e_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_EM4;
e_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_FH1;
e_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_Eta;
e_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_Phi;
e_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_Dtr_Eta;
e_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_Dtr_Phi;

e_output<<setiosflags(ios::left)<<setw(12)<<e_EM_P;
e_output<<setiosflags(ios::left)<<setw(12)<<e_EM_Pt;
e_output<<setiosflags(ios::left)<<setw(12)<<e_EM_E;
e_output<<setiosflags(ios::left)<<setw(12)<<e_EM_Et;
```

```
                    e_output<<setiosflags(ios::left)<<setw(12)<<e_EM_Eta;
                    e_output<<setiosflags(ios::left)<<setw(12)<<e_EM_Phi;
                    e_output<<setiosflags(ios::left)<<setw(12)<<e_EM_EMfrac;
                    e_output<<setiosflags(ios::left)<<setw(12)<<e_EM_iso;
                    e_output<<setiosflags(ios::left)<<setw(12)<<e_EM_id<<endl;
                    //output electrons with |eta|<0.8 requirement
        } // if good   //end 6
if(better)
 {   //7
 cout<<"better electron"<<endl;
 Double_t cal_Eta = TMath::Abs(emcand.CalEta());
 Double_t Cal_Theta = 2*TMath::ATan(TMath::Power(2.7182818,-cal_Eta));
 Double_t cal_P = TMath::Abs(emcand.Calpt()/TMath::Sin(Cal_Theta));
 // calculate theta first
 Double_t em_Eta = TMath::Abs(emcand.eta());
 Double_t EM_Theta = 2*TMath::ATan(TMath::Power(2.7182818,-em_Eta));
EMTrack *thisTrack=emcand.track_match();//pointer to EMTrack object
Double_t momentum_for_thisTrack = TMath::Abs(thisTrack->QOverPt_at_DCA());
momentum_for_thisTrack = 1/momentum_for_thisTrack;
Double_t em_Pt = momentum_for_thisTrack;
Double_t em_P = TMath::Abs(em_Pt/TMath::Sin(EM_Theta));



// assign electrons info to corresponding variables
e_hmx8 = emcand.HMx8();
e_Cal_P = cal_P;
e_Cal_Pt = emcand.Calpt();
e_Cal_E = emcand.CalE();
```

```
e_Cal_Et = emcand.CalE()*TMath::Abs(TMath::Sin(Cal_Theta));
e_Cal_PS = emcand.CalFloorE(PS);  // PS layer energy
e_Cal_EM1 = emcand.CalFloorE(EM1);
e_Cal_EM2 = emcand.CalFloorE(EM2);
e_Cal_EM3 = emcand.CalFloorE(EM3);
e_Cal_EM4 = emcand.CalFloorE(EM4);
e_Cal_FH1 = emcand.CalFloorE(FH1);  // energy for FH1 layer
e_Cal_Eta = emcand.CalEta();
e_Cal_Phi = emcand.CalPhi();
e_Cal_Dtr_Eta = emcand.CalDetectorEta();  // detector Eta
e_Cal_Dtr_Phi = emcand.CalDetectorPhi();

e_EM_P = em_P;
e_EM_Pt = em_Pt;
e_EM_E = emcand.E();
e_EM_Et = e_EM_E *TMath::Abs(TMath::Sin(EM_Theta));
e_EM_Eta = emcand.eta();
e_EM_Phi = emcand.phi();
e_EM_EMfrac = emcand.EMfrac();
e_EM_iso = emcand.iso();
e_EM_id = emcand.id();

// output to file electron_120513.dat
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_hmx8;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_P;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_Pt;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_E ;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_Et ;
```

```
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_PS;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_EM1;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_EM2;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_EM3;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_EM4;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_FH1;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_Eta;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_Phi;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_Dtr_Eta;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_Cal_Dtr_Phi;

e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_EM_P;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_EM_Pt;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_EM_E;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_EM_Et;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_EM_Eta;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_EM_Phi;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_EM_EMfrac;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_EM_iso;
e_eta08_output<<setiosflags(ios::left)<<setw(12)<<e_EM_id<<endl;
} //end better //end 7
                                } // valid candidate  end 5
                               } // if algo    end 4
    } //fEmcl loop     end 3
  } // jentry    end 2
    // draw histogram
TCanvas *ISOandID = new TCanvas("ISOandID"," ISO and ID for good electrons",1);
TCanvas *EandHMx8 = new TCanvas("EandHMx8"," Cal energy and HMx8 for good electrons",1);
```

```
TCanvas *CAL_EP = new TCanvas("CAL_EP","E over P from CAL info",1);
TCanvas *EM_EP = new TCanvas("EM_EP","E over P from EM info",1);
TCanvas *FloorEM12 = new TCanvas("FloorEM12","Floor EM1 EM2 Energy",1);
TCanvas *FloorEM34 = new TCanvas("FloorEM34","Floor EM3 EM4 Energy",1);
TCanvas *FloorFH1 = new TCanvas("FloorFH1","Floor FH1 Energy",1);
TCanvas *TOTAL_ENERGY = new TCanvas("TOTAL_ENERGY","total energy distribution",1);
TCanvas *VALID_ENERGY = new TCanvas("VALID_ENERGY","energy dist for valid candidates",1);
TCanvas *EMFRACTION = new TCanvas("EMFRACTION","emfraction",1);

 TCanvas *totalenergy;
 TCanvas *validenergy;
 TCanvas *ISO_ID;
 TCanvas *E_HMx8;
 TCanvas *em_ep;
 TCanvas *cal_ep;
 TCanvas *floorem12;
 TCanvas *floorem34;
 TCanvas *floorfh1;
 TCanvas *emfraction;
ISO_ID = (TCanvas*)gROOT->GetListOfCanvases()->FindObject("ISOandID");
ISO_ID->Divide(2,1);
ISO_ID->cd(1);
 ISO_ID->SetTicks(0,2);
 ISO_ID->SetTicks(0,1);
 EM_iso->GetYaxis()->SetTitle("# of entries");
 EM_iso->GetYaxis()->SetLabelOffset(-0.85);
 EM_iso->GetYaxis()->SetTitleOffset(0.3);
 EM_iso->Draw();
```

```
ISO_ID->cd(2);
 ISO_ID->SetTicks(0,2);
 ISO_ID->SetTicks(0,1);
 EM_id->GetXaxis()->SetTitle("id");
 EM_id->GetYaxis()->SetTitle("# of entries");
 EM_id->GetYaxis()->SetLabelOffset(-0.85);
 EM_id->GetYaxis()->SetTitleOffset(0.3);
 EM_id->Draw();

E_HMx8 = (TCanvas*)gROOT->GetListOfCanvases()->FindObject("EandHMx8");
E_HMx8->Divide(2,1);
E_HMx8->cd(1);
 E_HMx8->SetTicks(0,2);
 E_HMx8->SetTicks(0,1);
 Cal_E->GetXaxis()->SetTitle("GeV");
 Cal_E->GetYaxis()->SetTitle("# of entries");
 Cal_E->GetYaxis()->SetLabelOffset(-0.85);
 Cal_E->GetYaxis()->SetTitleOffset(0.3);
 Cal_E->Draw();

E_HMx8->cd(2);
 E_HMx8->SetTicks(0,2);
 E_HMx8->SetTicks(0,1);
 hmx8->GetYaxis()->SetTitle("# of entries");
 hmx8->GetYaxis()->SetLabelOffset(-0.85);
 hmx8->GetYaxis()->SetTitleOffset(0.3);
 hmx8->Draw();
```

```
cal_ep = (TCanvas*)gROOT->GetListOfCanvases()->FindObject("CAL_EP");
cal_ep->cd();
CAL_E_over_P->Draw();
em_ep = (TCanvas*)gROOT->GetListOfCanvases()->FindObject("EM_EP");
 em_ep->cd();
EM_E_over_P->Draw();
floorem12 =(TCanvas*)gROOT->GetListOfCanvases()->FindObject("FloorEM12");
floorem12->Divide(2,1);
floorem12->cd(1);
 floorem12->SetTicks(0,2);
 floorem12->SetTicks(0,1);
 Cal_EM1->GetXaxis()->SetTitle("GeV");
 Cal_EM1->GetYaxis()->SetTitle("# of entries");
 Cal_EM1->GetYaxis()->SetLabelOffset(-0.85);
 Cal_EM1->GetYaxis()->SetTitleOffset(0.3);
 Cal_EM1->Draw();
floorem12->cd(2);
 floorem12->SetTicks(0,2);
 floorem12->SetTicks(0,1);
 Cal_EM2->GetXaxis()->SetTitle("GeV");
 Cal_EM2->GetYaxis()->SetTitle("# of entries");
 Cal_EM2->GetYaxis()->SetLabelOffset(-0.85);
 Cal_EM2->GetYaxis()->SetTitleOffset(0.3);
 Cal_EM2->Draw();
floorem34 =(TCanvas*)gROOT->GetListOfCanvases()->FindObject("FloorEM34");
floorem34->Divide(2,1);
floorem34->cd(1);
 floorem34->SetTicks(0,2);
```

```
 floorem34->SetTicks(0,1);
 Cal_EM3->GetXaxis()->SetTitle("GeV");
 Cal_EM3->GetYaxis()->SetTitle("# of entries");
 Cal_EM3->GetYaxis()->SetLabelOffset(-0.85);
 Cal_EM3->GetYaxis()->SetTitleOffset(0.3);
 Cal_EM3->Draw();
floorem34->cd(2);
 floorem34->SetTicks(0,2);
 floorem34->SetTicks(0,1);
 Cal_EM4->GetXaxis()->SetTitle("GeV");
 Cal_EM4->GetYaxis()->SetTitle("# of entries");
 Cal_EM4->GetYaxis()->SetLabelOffset(-0.85);
 Cal_EM4->GetYaxis()->SetTitleOffset(0.3);
 Cal_EM4->Draw();
floorfh1= (TCanvas*)gROOT->GetListOfCanvases()->FindObject("FloorFH1");
 floorfh1->cd();
 floorfh1->SetTicks(0,2);
 floorfh1->SetTicks(0,1);
 Cal_FH1->GetXaxis()->SetTitle("GeV");
 Cal_FH1->GetYaxis()->SetTitle("# of entries");
 Cal_FH1->GetYaxis()->SetLabelOffset(-0.85);
 Cal_FH1->GetYaxis()->SetTitleOffset(0.3);
 Cal_FH1->Draw();
totalenergy =(TCanvas*)gROOT->GetListOfCanvases()->FindObject("TOTAL_ENERGY");
 totalenergy->cd();
 totalenergy->SetTicks(0,2);
 totalenergy->SetTicks(0,1);
 total_energy->GetXaxis()->SetTitle("GeV");
```

```
total_energy->GetYaxis()->SetTitle("# of entries");
total_energy->GetYaxis()->SetLabelOffset(-0.85);
total_energy->GetYaxis()->SetTitleOffset(0.3);
total_energy->Draw();
validenergy = (TCanvas*)gROOT->GetListOfCanvases()->FindObject("VALID_ENERGY");
validenergy->cd();
validenergy->SetTicks(0,2);
validenergy->SetTicks(0,1);
valid_energy->GetXaxis()->SetTitle("GeV");
valid_energy->GetYaxis()->SetTitle("# of entries");
valid_energy->GetYaxis()->SetLabelOffset(-0.85);
valid_energy->GetYaxis()->SetTitleOffset(0.3);
valid_energy->Draw();
emfraction =(TCanvas*)gROOT->GetListOfCanvases()->FindObject("EMFRACTION");
emfraction->cd();
emfraction->SetTicks(0,2);
EM_EMfrac->GetYaxis()->SetTitle("# of entries");
EM_EMfrac->GetYaxis()->SetLabelOffset(-0.85);
EM_EMfrac->GetYaxis()->SetTitleOffset(0.3);
emfraction->SetTicks(0,1);
EM_EMfrac->Draw();
} // Loop  end 1
```

```
/* ---------- MCtmb355833.C ----------*/

#define MCtmb355833_cxx

#include "MCtmb355833.h"

#include "TStyle.h"

#include "TCanvas.h"

#include "TRefArray.h"

#include "TString.h"

#include "TFile.h"

#include <iostream.h>

#include <fstream.h>

#include <iomanip.h>

#include "emcandidate/emcandidate/EMcandidate.hpp"

//#include <string>

using namespace std;


void   MCtmb355833::Loop()

{

  if (fChain == 0) return;

  E_over_P = new TH1F("E_over_P","E over P", 100, 0, 2);

  Int_t nentries = Int_t(fChain->GetEntries());

  //TString algo_name;

  // Int_t counter=0;

  int total_candidate = 0;

  Int_t my_entries = nentries;

  cout<<" no. of entries "<< nentries<<endl;

    // my_entries = nentries;

  Int_t nbytes = 0, nb = 0;
```

```
for (Int_t jentry=0; jentry<my_entries;jentry++) {    //2
  Int_t ientry = LoadTree(jentry);
       //in case of a TChain, ientry is the entry number in the current file
  nb = fChain->GetEntry(jentry);   nbytes += nb;
  //Emcl:
  //nel=0;
  //int ngam=0;

  int valid_candidate = 0;
  // make a track collection
  TrackCollection tcoll;
  tcoll.Reset();
  for(Int_t i=0; i< fTrks->GetLast()+1; i++){
    EMTrack newtrack((TMBTrks*) fTrks->At(i));
    tcoll.AddTrack(newtrack);
  }
  for(Int_t i=0; i<fEmcl->GetLast()+1; i++) {        // 3 Emcl
    TMBEmcl* emcl = ( TMBEmcl* ) fEmcl->At(i);
                          // int ncells= emcl->ncells();  //get number of cells
                          const char *algoname=emcl->algoname();
                          char* c="s";
                          if(algoname[0]==c[0])
                           {          // 4 algo
                             TMBVrts* vtx=emcl->GetVertex();
                             float vxyz[3]={0,0,0};
                             if(vtx)
                            {
                             vxyz[0]=vtx->vx();
```

```
  vxyz[1]=vtx->vy();
  vxyz[2]=vtx->vz();
 }
EMcandidate emcand(emcl,fGlob,tcoll,vxyz,false,13,0);
if(emcand.is_valid_candidate())
 {                              // 5 emcandidate
  Float_t J_E,J_eta,J_phi,J_ntrk,det_eta,det_phi,jdr;
  Float_t J_Et,J_ICD_E,J_ccmg_E,J_ecmg_E,J_icdf,J_ccmg,J_ecmg;
  J_E = 0;
  J_Et = 0;
  J_ICD_E = 0;
  J_ccmg_E = 0;
  J_ecmg_E = 0;
  J_eta = 0;
  J_phi = 0;
  int matched_jet = 0;
  valid_candidate++;
  total_candidate++;
  Bool_t good = emcand.E()>50 &&
  emcand.EMfrac()>0.9 &&
      (emcand.id()==10 || TMath::Abs(emcand.id())==11)&&
      emcand.iso()<0.15 &&
      emcand.HMx8()<15 &&   //15--tight,20--loose
  TMath::Abs(emcand.eta())<=1.0 &&
          emcand.has_track_match();
  if(good)
{             // 6
```

```
// calculation for emcl part
// calculate Pt, but theta first
Double_t em_Eta = TMath::Abs(emcand.eta());
Double_t EM_Theta =2*TMath::ATan(TMath::Power(2.7182818,-em_Eta));
EMTrack *thisTrack=emcand.track_match();//pointer to EMTrack object
Double_t momentum_for_thisTrack=TMath::Abs(thisTrack->at_DCA());
momentum_for_thisTrack = 1/momentum_for_thisTrack;
Double_t em_Pt = momentum_for_thisTrack;
Double_t em_P = TMath::Abs(em_Pt/TMath::Sin(EM_Theta));

// calculate other quantities
e_Cal_EM1 = emcand.CalFloorE(EM1);
e_Cal_EM2 = emcand.CalFloorE(EM2);
e_Cal_EM3 = emcand.CalFloorE(EM3);
e_Cal_EM4 = emcand.CalFloorE(EM4);
e_Cal_FH1 = emcand.CalFloorE(FH1); // energy for FH1 layer
e_Cal_Eta = emcand.CalEta();
e_Cal_Phi = emcand.CalPhi();
J_ICD_E = 0;
J_ccmg_E = 0;
J_ecmg_E = 0;

// output info for good e's
e_ICR_output<<"event #: "<<jentry<<"  valid candidate: "<<valid_candidate;
e_ICR_output<<"  matched jet: "<<matched_jet<<endl;
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<emcand.E();
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<em_P;
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<emcand.CalFloorE(EM1);
```

```
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<emcand.CalFloorE(EM2);
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<emcand.CalFloorE(EM3);
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<emcand.CalFloorE(EM4);
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<emcand.CalFloorE(FH1);
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<emcand.CalEta();
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<emcand.CalPhi();
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<J_ICD_E;
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<J_ccmg_E;
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<J_ecmg_E<<endl;
E_over_P->Fill(emcand.E()/em_P);
} // if good 6 <===
  if(!good)
{       // 6

// for this valid EMcandidate, loop over all jets corresponding
// to this event
  for(Int_t i=0; i<fJets->GetLast()+1; i++) {  // 7  all jets
    TMBJets* thisJet = ( TMBJets* ) fJets->At(i);
                TString J_algoname=thisJet->algoname();
                TString ch="JCCB";
                Bool_t good_jet=0;
                if(strcmp(J_algoname,ch)==0)
                {       // 8  Jet algo matched

                  e_Cal_Eta = emcand.CalEta();
                  e_Cal_Phi = emcand.CalPhi();
                  J_E = thisJet->E();
                  J_eta = thisJet->eta();
```

```
                J_phi = thisJet->phi();
                det_eta = e_Cal_Eta - J_eta;
                det_phi = e_Cal_Phi - J_phi;
                jdr = TMath::Sqrt(det_eta * det_eta + det_phi * det_phi);
                J_ntrk = thisJet->ntrk();
                good_jet = thisJet->E()>50.0 &&
                        jdr<0.5 &&
                        e_Cal_Eta > 1.0 &&
                        e_Cal_Eta < 1.5 &&
                    // TMath::Abs(det_eta)<0.2&&
                    // TMath::Abs(det_phi)<0.2&&
                        J_ntrk == 1;
                if(good_jet)
            {           // 9 matched ICD jet
                matched_jet++;

Double_t em_Eta = TMath::Abs(emcand.eta());
                Double_t EM_Theta =2*TMath::ATan(TMath::Power(2.7182818,-em_Eta));
                EMTrack *thisTrack=emcand.track_match();//pointer to EMTrack object
                Double_t momentum_for_thisTrack=TMath::Abs(thisTrack->at_DCA());
                momentum_for_thisTrack = 1/momentum_for_thisTrack;
                Double_t em_Pt = momentum_for_thisTrack;
                Double_t em_P = TMath::Abs(em_Pt/TMath::Sin(EM_Theta));

                // calculate other quantities
                e_Cal_EM1 = emcand.CalFloorE(EM1);
                e_Cal_EM2 = emcand.CalFloorE(EM2);
                e_Cal_EM3 = emcand.CalFloorE(EM3);
```

```
e_Cal_EM4 = emcand.CalFloorE(EM4);
e_Cal_FH1 = emcand.CalFloorE(FH1); // energy for FH1 layer
e_Cal_Eta = emcand.CalEta();
e_Cal_Phi = emcand.CalPhi();
// matched, get jet information
//calculate theta first in order to calculate J_Et
Double_t J_Theta = 2*TMath::ATan(TMath::Power(2.7182818,-J_eta));

J_icdf = thisJet->icdf();
J_E = thisJet->E();
J_ccmg = thisJet->ccmg();
J_ecmg = thisJet->ecmg();
J_Et = J_E *TMath::Sin(J_Theta); //output this one
J_ccmg_E = J_Et*J_ccmg; //output this one
J_ecmg_E = J_Et*J_ecmg; //output this one
J_ICD_E = J_Et*J_icdf; //output this one

e_ICR_output<<"event #: "<<jentry<<" valid candidate: "<<valid_candidate;
e_ICR_output<<" matched jet: "<<matched_jet<<endl;
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<emcand.E();
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<em_P;
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<emcand.CalFloorE(EM1);
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<emcand.CalFloorE(EM2);
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<emcand.CalFloorE(EM3);
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<emcand.CalFloorE(EM4);
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<emcand.CalFloorE(FH1);
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<emcand.CalEta();
e_ICR_output<<setiosflags(ios::left)<<setw(12)<<emcand.CalPhi();
```

```
                    e_ICR_output<<setiosflags(ios::left)<<setw(12)<<J_ICD_E;
                    e_ICR_output<<setiosflags(ios::left)<<setw(12)<<J_ccmg_E;
                    e_ICR_output<<setiosflags(ios::left)<<setw(12)<<J_ecmg_E<<endl;
                    E_over_P->Fill(emcand.E()/em_P);
                     } // 9 <==
                       }// Jet algorithm  8 <===
                      } // all jets        7 <==
                    } // !good      6 <===
        } // valid candidate  5 <===
      } // if algo           4 <===
     } //fEmcl loop          3 <===

   } // jentry              2 <===

   // draw histogram
  E_over_P->Draw();
  cout<<" total entry is: "<< my_entries<<endl;
  cout<<"total candidate is: "<<total_candidate<<endl;
} // Loop
```

```
/* ---------- electron_tree.C ----------*/

{

// macro to read data from a data file and fill a tree

// using the data

gROOT->Reset();

// the structure to hold the variables for the branch

struct GoodElectron_t {

        Float_t e_hmx8;

        Float_t e_Cal_P;

        Float_t e_Cal_Pt;

        Float_t e_Cal_E ;

        Float_t e_Cal_Et ;

        Float_t e_Cal_PS;

        Float_t e_Cal_EM1;

        Float_t e_Cal_EM2;

        Float_t e_Cal_EM3;

        Float_t e_Cal_EM4;

        Float_t e_Cal_FH1;

        Float_t e_Cal_Eta;

        Float_t e_Cal_Phi;

        Float_t e_Cal_Dtr_Eta;

        Float_t e_Cal_Dtr_Phi;

        Float_t e_EM_P;

        Float_t e_EM_Pt;

        Float_t e_EM_E;

        Float_t e_EM_Et;

        Float_t e_EM_Eta;

        Float_t e_EM_Phi;
```

```
                Float_t e_EM_EMfrac;
                Float_t e_EM_iso;
                Float_t e_EM_id;
                };
GoodElectron_t electron;
TH1F *ht = new TH1F("ht","Et_over_Pt",100,0,2);
TH1F *newht = new TH1F("newht"," new E over P using new weights",100,0,2);
// open the data file
FILE *fp = fopen("electron_eta08_id11.dat","r");
char line[300];
// create a new ROOT file
TFile *f = new TFile("electron_eta08_id11_dat.root","RECREATE");
//TFile *histogram = new TFile("HistogramForTotalgoodelectron.root","RECREATE");

// create a TTree
TTree *tree = new TTree("tree", "good electron data from electron_MCZee_eta08_id11.dat file");
// create one branch with all the information from
// the stucture
tree-
>Branch("electron",&electron.e_hmx8,"e_hmx8/F:e_Cal_P:e_Cal_Pt:e_Cal_E:e_Cal_Et:e_Cal_PS:e_Cal_
EM1:e_Cal_EM2:e_Cal_EM3:e_Cal_EM4:e_Cal_FH1:e_Cal_Eta:e_Cal_Phi:e_Cal_Dtr_Eta:e_Cal_Dtr_P
hi:e_EM_P:e_EM_Pt:e_EM_E:e_EM_Et:e_EM_Eta:e_EM_Phi:e_EM_EMfrac:e_EM_iso:e_EM_id");
Int_t counter = 0;
// fill the tree from the values in ASCII file
while (fgets(&line,299,fp))
{
sscanf(&line[0],"%f%f%f%f",electron.e_hmx8,&electron.e_Cal_P,&electron.e_Cal_Pt,&electron.e_Cal_E
);
```

```
sscanf(&line[48],"%f%f%f%f",&electron.e_Cal_Et,&electron.e_Cal_PS,&electron.e_Cal_EM1,&electron.
e_Cal_EM2);
sscanf(&line[96],"%f%f%f%f",&electron.e_Cal_EM3,&electron.e_Cal_EM4,&electron.e_Cal_FH1,&elect
ron.e_Cal_Eta);
sscanf(&line[144],"%f%f%f%f",&electron.e_Cal_Phi,
&electron.e_Cal_Dtr_Eta,&electron.e_Cal_Dtr_Phi,&electron.e_EM_P);
sscanf(&line[192],"%f%f%f%f",&electron.e_EM_Pt,&electron.e_EM_E,&electron.e_EM_Et,
&electron.e_EM_Eta);
 sscanf(&line[240],"%f%f%f%f", &electron.e_EM_Phi, &electron.e_EM_EMfrac, &electron.e_EM_iso,
&electron.e_EM_id);
  printf("%s",line);
  tree->Fill();//Qun
  Float_t newE= 0.551105 * electron.e_Cal_EM1 + 1.26691 * electron.e_Cal_EM2 + 0.677347 *
electron.e_Cal_EM3 + 2.4208 * electron.e_Cal_EM4 + 0.0220158 * electron.e_Cal_FH1;
  ht->Fill(electron.e_EM_Et/electron.e_EM_Pt);
  newht->Fill(newE/electron.e_EM_Pt);
}
// check what the tree looks like
//tree->Print();
TCanvas *c = new TCanvas("c","E over P",1);
c->Divide(2,1);
c->cd(1);
ht->GetXaxis()->SetTitle("E_over_P");
ht->GetYaxis()->SetTitle("Frequency");
ht->Draw();
c->cd(2);
newht->GetXaxis()->SetTitle("E_over_P");
```

```
newht->GetYaxis()->SetTitle("Frequency");

newht->Draw();

fclose(fp);

f->Write();

//histogram->Write();

}
```

```
/* ----------MCtmb355833_tree.C ----------*/

{
// macro to read data from a data file and fill a tree
// using the MC data
gROOT->Reset();
// the structure to hold the variables for the branch
struct MCGoodElectron_t {
                Float_t em_E;
                Float_t em_P;
                Float_t em_EM1;
                Float_t em_EM2;
                Float_t em_EM3;
                Float_t em_EM4;
                Float_t em_FH1;
                Float_t Cal_eta;
                Float_t Cal_phi;
                Float_t J_ICD_E;
                Float_t J_ccmg_E;
                Float_t J_ecmg_E;
               };
MCGoodElectron_t electron;
TH1F *ht = new TH1F("ht","E_over_P",100,0,2);
TH1F *newht = new TH1F("newht","new E over P using new weights",100,0,2);
TH1F *oldht = new TH1F("oldht","old E over P using sum of layered energy",100,0,2);
// open the data file
FILE *fp = fopen("electron_MCtmb355-833.dat","r");
char line[150];
```

```
// create a new ROOT file
TFile *f = new TFile("electron_MCtmb355-833.dat.root","RECREATE");
// create a TTree
TTree *tree = new TTree("tree", "good electron data from electron_MCtmb355-833.dat file");
// create one branch with all the information from the stucture
tree-
>Branch("electron",&electron.em_E,"em_E/F:em_P:em_EM1:em_EM2:em_EM3:em_EM4:em_FH1:Cal_
eta:Cal_phi:J_ICD_E:J_ccmg_E:J_ecmg_E");
Int_t counter = 0;
// fill the tree from the values in ASCII file
while (fgets(&line,149,fp))
{
 if(isdigit(line[0]))
  {
sscanf(&line[0],"%f%f%f%f",&electron.em_E,&electron.em_P,&electron.em_EM1,&electron.em_EM2);
sscanf(&line[48],"%f%f%f%f",&electron.em_EM3,&electron.em_EM4,&electron.em_FH1,&electron.Cal
_eta);
sscanf(&line[96],"%f%f%f%f",&electron.Cal_phi,&electron.J_ICD_E,&electron.J_ccmg_E,&electron.J_e
cmg_E);

Float_t newE =
0.827812*electron.em_EM1+1.03701*electron.em_EM2+1.20386*electron.em_EM3+0.790852*electron.e
m_EM4+5.11584*electron.em_FH1+17.4084*electron.J_ICD_E+2.41062*electron.J_ccmg_E+12.9962*el
ectron.J_ecmg_E;

Float_t oldE =
electron.em_EM1+electron.em_EM2+electron.em_EM3+electron.em_EM4+electron.em_FH1+electron.J_
ICD_E+electron.J_ccmg_E+electron.J_ecmg_E;
```

```
 oldht->Fill(oldE/electron.em_P);
 ht->Fill(electron.em_E/electron.em_P);
 newht->Fill(newE/electron.em_P);
 tree->Fill();
   } // end of if
} // end of while
TCanvas *c = new TCanvas("c","new canvas",1);
c->Divide(2,1);
c->cd(1);
oldht->GetXaxis()->SetTitle("E_over_P");
oldht->GetYaxis()->SetTitle("Frequency");
oldht->Draw();
c->cd(2);
newht->GetXaxis()->SetTitle("E_over_P");
newht->GetYaxis()->SetTitle("Frequency");
newht->Draw();

//c->cd(3);
//ht->Draw();
fclose(fp);
f->Write();
}
```

REFERENCES

[1] http://www.infoplease.com/ce6/sci/A0804257.html

[2] http://www.hep.yorku.ca/yhep/mainb.html

[3] http://www.academicpress.com/inscight/01301997/electro2.htm

[4] http://www.academicpress.com/inscight/02292000/muon1.htm

[5] http://www.academicpress.com/inscight/03171998/neutrin1.htm

[6] http://www-sldnt.slac.stanford.edu/alr/standard_model.htm

[7] http://www.slac.stanford.edu/gen/edu/about-1.html

[8] http://www-d0.fnal.gov/results/publications_talks/thesis/snyder/html/node28.html

[9] D0 note 2365

[10] http://rd11.web.cern.ch/RD11/rkb/PH14pp/node19.html

[11] http://www-d0.fnal.gov/results/publications_talks/thesis/snyder/html/node48.html

no reference added from "experiment"

[12] http://www.hicb.org/jtb/HIJTB-011203/Bos-D0-Grid-Ideas.pdf

[13] http://www-d0.fnal.gov/computing/algorithms/howto/howtoreco.html

[14] D0note 3979

[15] D0note 3978

[16] http://root.cern.ch/

[17] http://www-d0.fnal.gov/Run2Physics/wz/d0-private/wzskim/WZskim-em.html, WZ is one of many groups at Fermilab.

[18] http://www -d0.fnal.gov/phys_id/emid

/d0_private/certification/main_v2_2.html#cuts

EM-ID group is one of the groups working on D0 project at Fermilab.

[19] D0note 3888

* Refer to Chapter 3: D0 Calorimeter Readout

[20] http://root.cern.ch/root/HowtoFit.html

[21] http://www-d0.fnal.gov/Run2Physics/wz/

[22] These electrons are selected according to the criteria established

in chapter 4 – data reduction and analysis.

[23] http://www-clued0.fnal.gov/runjob/current/tutorials/

MC_Generation_Description.html