Reconstruction of $D^0 \rightarrow K^+ + \pi^-$

Guide : Dr. Kavita Lalwani
Students : Siddharth Singh (MNIT Jaipur), Patel Mihir Hemantkumar (SVNIT Surat)
Institute name : MNIT Jaipur

Reading Multiple Input Files:

- We will be reading multiple input files stored in the location specified by the user.
- For reading and scanning the file we will be making the use of classes : *TList, TSystemFile, TSystemDirectory, TIter*

```
#include <TLorentzVector.h>
#include <iostream>
#include <vector>
using namespace std;

void Dmassrecreation1()
{

  char *outputpath = "/home/siddharth/EIC/ATHENA/minQ2=1000/EICOutputFiles/OutputRootFiles/";
  char *dirname="/home/siddharth/EIC/ATHENA/minQ2=1000/EICInputFiles/10x100/";
  char *ext=".root";
  TString o = "OUTPUT";
  TSystemDirectory dir(dirname, dirname);

  TList *files = dir.GetListOfFiles();
  if (files) {
    TSystemFile *file;
    TString fname;
    TIter next(files);
    while ((file=(TSystemFile*)next())) {
      fname = file->GetName();
      if (!file->IsDirectory() && fname.EndsWith(ext)) {
         cout << "READING FILE : " << fname.Data() << endl;
         TString fullfname = dirname + fname;
         cout << "SOURCE DIRECTORY : " << fullfname.Data() << endl;

         TFile * myFile = new TFile(fullfname.Data());
```

Input Files (26 input .root files)

pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0002**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0003**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0004**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0005**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0006**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0007**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0008**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0009**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0010**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0011**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0013**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0015**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0051**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0052**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0053**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0055**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0056**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0058**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0059**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0060**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0061**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0062**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0063**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0064**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0067**.root
pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_**1.0068**.root

Accessing the Trees, Branches and Leaves from Input ROOT Files:

- **TTreeReader** is used to open the “events” *Tree* of the input root file.
- **TTreeReaderArray<Float_t>** is used to read the *Branches* and *Leaves*.

```
TTreeReader myReader("events", myFile);                    //Pass the Tree's name and the TFile name in TTreeReader


//Get Dynamical Variables
TTreeReaderArray<Float_t> px(myReader, "ReconstructedParticles.p.x");
TTreeReaderArray<Float_t> py(myReader, "ReconstructedParticles.p.y");
TTreeReaderArray<Float_t> pz(myReader, "ReconstructedParticles.p.z");
TTreeReaderArray<Float_t> energy(myReader, "ReconstructedParticles.energy");
TTreeReaderArray<Int_t> pid(myReader, "ReconstructedParticles.pid");
```

Defining 4 vectors

- We will define Lorentz Vectors d, pi and k for reading and storing values of four-vectors of π^- , K^+ and Combinations of the four vector i.e. by adding all the possible values of π^- and K^+ four-vectors, are stored in d.

- These values are then stored in a dynamical array of the data type TLorentzVector, i.e the arrays *pivalues* and *kvalues* will store the 4 vectors that we calculate via the use of pi and k as mentioned above.

```
//We are creating here TLorentzVector to store 4 vectors for d, k and pi
TLorentzVector d;
TLorentzVector pi;
TLorentzVector k;

//We will now store the 4vectors of pi and k in pivalues and kvalues respectively
std::vector<TLorentzVector> *pivalues = new std::vector<TLorentzVector>();
std::vector<TLorentzVector> *kvalues = new std::vector<TLorentzVector>();
```

Reading/Accessing the entries of an input root file:

- We can access the root file and it's entries by running a *while* loop followed by a nested *for* loop.

- While loop will access the events by myReader.Next( ) function calling and for loop will then access that particular event by reading the PIDs of the reconstructed particles, we can then use a PID check which will give us the particles that we need in our analysis.

- Here since we are dealing with the decay of $\mathbf{D^0 \rightarrow K^+ \pi^-}$, we will put a PID check of -221 (π^-) and 321 (K^+).

Reconstructing particles by PID check

- Here we have used a *while* loop to study the events and further have utilised *for* loop to study the reconstructed particles.
- We have then stored the necessary data in the respective histograms.

```
int p=0, q=0, r=0, s=0, t=0, u=0;   //counters

while (myReader.Next())   //while loop to access events
{
  p++;  //counters
  for (int i = 0; i < pid.GetSize(); i++)              //for loop from 0 to pid.GetSize()
  {
    if(pid[i] == -211)                                 //recreate mass of Pi- PID = -221
    {
      while (true)
      {
        r++;  //counter
        pi.SetPxPyPzE(px[i],py[i],pz[i],energy[i]);    //Creating the 4 vector for pi
        pihpx->Fill(pi.Px());                          //Momentum in x direction
        pihpy->Fill(pi.Py());                          //Momentum in y direction
        pihpz->Fill(pi.Pz());                          //Momentum in z direction
        pien->Fill(pi.E());                            //Energy
        pi_mass->Fill(pi.M());                         //Mass
        pipseudorapidity->Fill(pi.PseudoRapidity());   //PseudoRapidity
        pipt->Fill(pi.Pt());
        pivalues->push_back(pi);                       //pivalues stores the 4 vector of pi
        break;
      }
    }
```

PID Check for Pions- (PID = 211)

Reconstructing particles by PID check

- We store the 4 vectors (p_x, p_y, p_z and energy) in the array kvalues by using **push_back(k)**

```
    else if(pid[i] == 321)                                  //recreate mass of K+ PID = 321
    {
      while (true)
      {
        s++;  //counters
        k.SetPxPyPzE(px[i],py[i],pz[i],energy[i]);          //Creating the 4 vector for k
        khpx->Fill(k.Px());                                 //Momentum in x direction
        khpy->Fill(k.Py());                                 //Momentum in y direction
        khpz->Fill(k.Pz());                                 //Momentum in z direction
        ken->Fill(k.E());                                   //Energy
        k_mass->Fill(k.M());                                //Mass
        kpseudorapidity->Fill(pi.PseudoRapidity());         //PseudoRapidity
        kpt->Fill(k.Pt());
        kvalues->push_back(k);                              //kvalues stores the 4 vector of k
        break;
      }
    }
  }
}
```

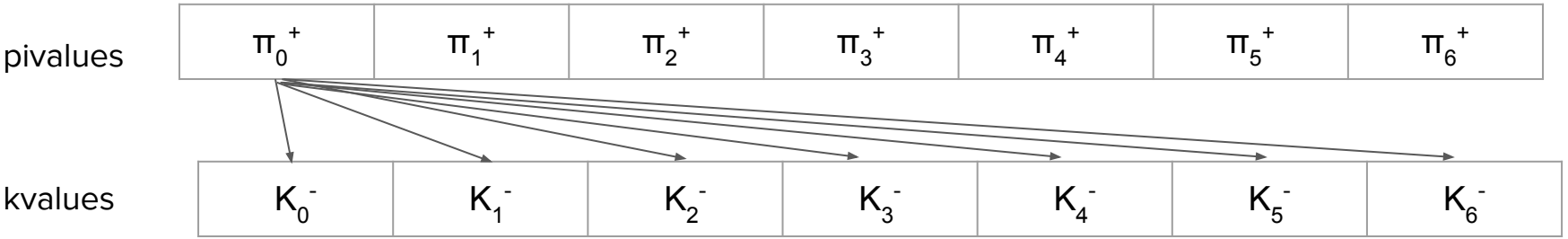
PID Check for Kaons+ (PID = 321)

Combinational Mass : Addition of K+ and Pi-

1. pivalues and kvalues store the 4 vectors of π^- and K^+ that we have accessed by reading the input files from EIC.
2. We will now take combinations of each four vectors of kvalues with each component of pivalues and plot the distribution in the histogram.
3. For the purpose of combinations we have to run 2 nested for loops each will run till the end of the array length of pivalues and kvalues respectively.
4. By use of temporary variables and cell pointers we can add the 4 vectors and plot them on a histograms simultaneously, along with the corresponding p_t checks and the four momentums.
5. **This histogram of combinational mass when put through proper conditions can give us a peak in the vicinity of the mass of D^0.**

Combinational Mass : A brute force approach

All the final state pions are stored in the dynamical array '**pivalues**'



dmass contains all the possible combinations of π_i^+ and K_j^- where,

i = all the $\boldsymbol{\pi^+}$ produced, and
j = all the $\boldsymbol{K^-}$ produced

Similarly, all the Kaons are stored in the **'kvalues'**

Combinational Mass

```
//Temporary variables xx and yy used to call the 4 vectors to add them
TLorentzVector xx;
TLorentzVector yy;

int xcount =0, ycount=0;  //Creating variabkes to act as pointers for pivalues and kvalues

//Nested for loops to create pairings of Pi and K
for (auto x = pivalues->begin(); x != pivalues->end(); ++x)
{
  xx = (*pivalues)[xcount];
  for (auto y = kvalues->begin(); y != kvalues->end(); ++y)
  {
    yy = xx + (*kvalues)[ycount];   //Adding the 4 vectors of k and pi to give a pair of D
    dmass = yy.M();                 //dmass is the mass of the pair of pi and k

    dmass0->Fill(dmass);            //Filling the mass of the 4 vector calculated, gives a distribution of Combinations

    dpseudo = yy.PseudoRapidity();
    dpseudorapidity->Fill(dpseudo); //Pseudorapidity of the combinational mass

    dptcut=yy.Pt();
    dpt->Fill(dptcut);              //Pt of the combinational mass
```

PseudoRapidity is called by *yy.psuedorapidity( )*
Pt Cut is called by *yy.Pt( )*

Adding all the output root files using ‘hadd’ command

- Go to the folder where all the output root files are stored, open the terminal and add all the individual root files using **hadd** command-*hadd output.root*
- *pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_1.0002.root pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_1.0003.root ……**
- This will create one **output.root** file on which we will then do the analysis.

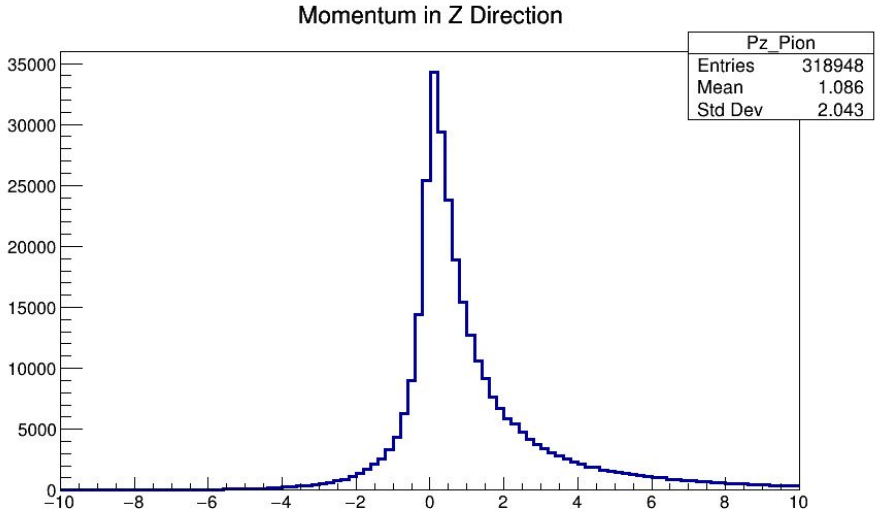
*The output file name is the same as that of the input files

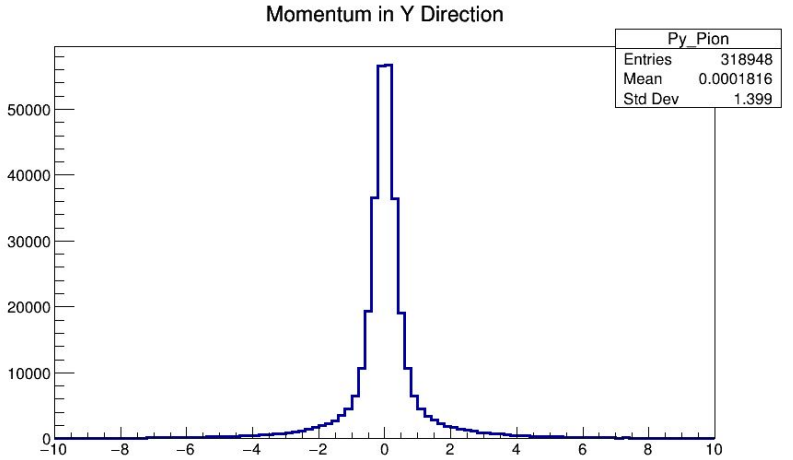
Output Histograms:

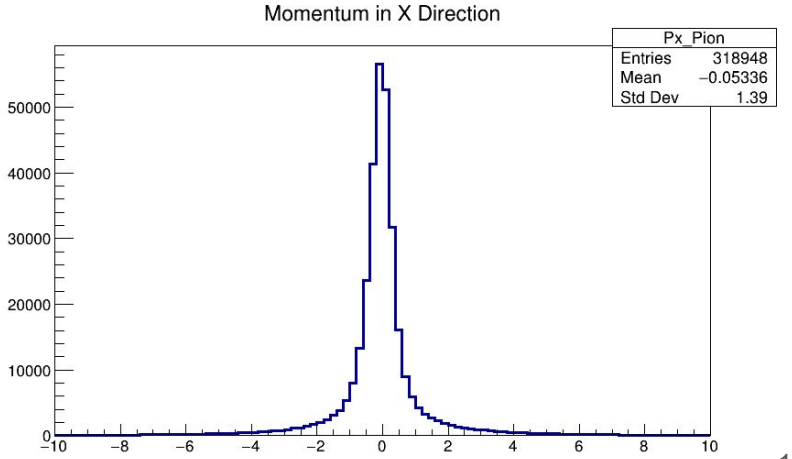
When we run our program (on the input files) we read the output in the form of histograms which we then store in the output.root. The histograms stored are:

1. p_x, p_y, p_z and E, Mass of Pion
2. p_x, p_y, p_z and E, Mass of Kaon
3. (K^+ and π^-) Pairing Mass (By adding respective 4 vectors)
4. Pseudorapidity (Eta) of K-, Pi+ and Combinational Mass ($K^+ + \pi^-$)
5. Transverse Momentum of Combinational Mass ($K^+ + \pi^-$)
6. Combinational Mass ($K^+ + \pi^-$) without any condition
7. Conditions on Combinational Mass (Different Pseudorapidity Conditions)
8. Combinational Mass with Pt Cut >0.5 GeV
9. Fitting Histograms.

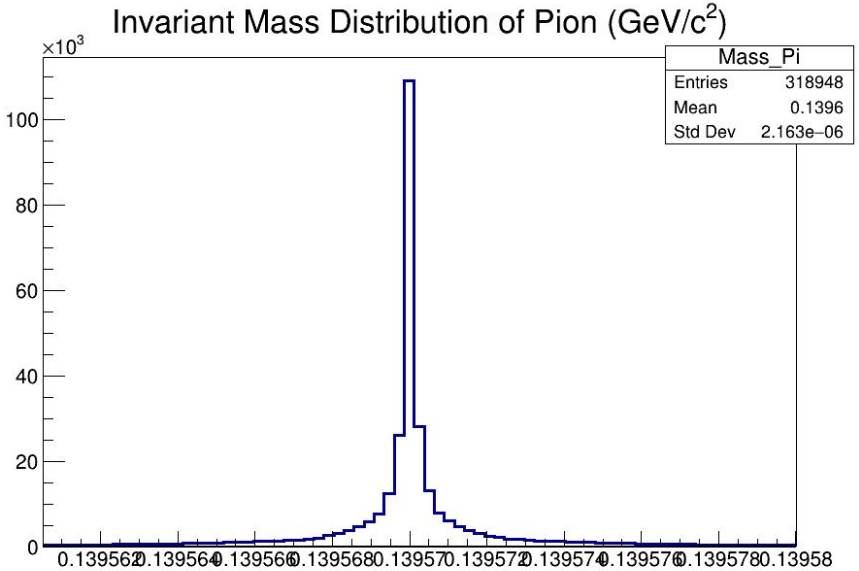
Momentum (p_x, p_y, p_z) of π^- :

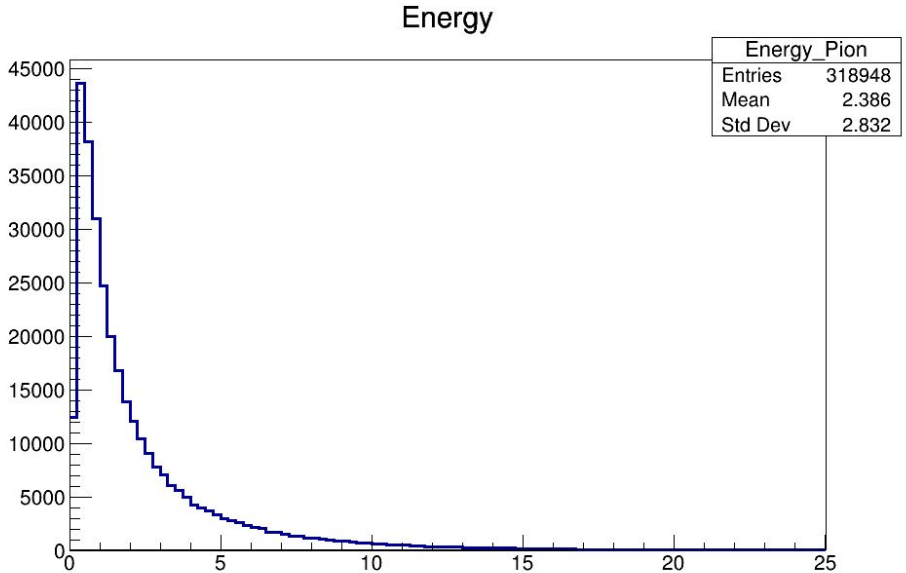




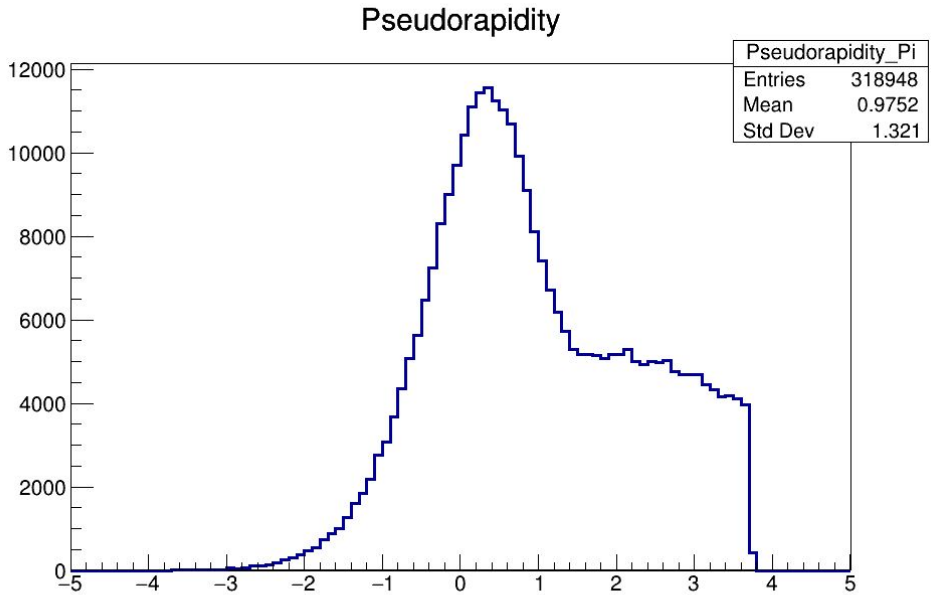


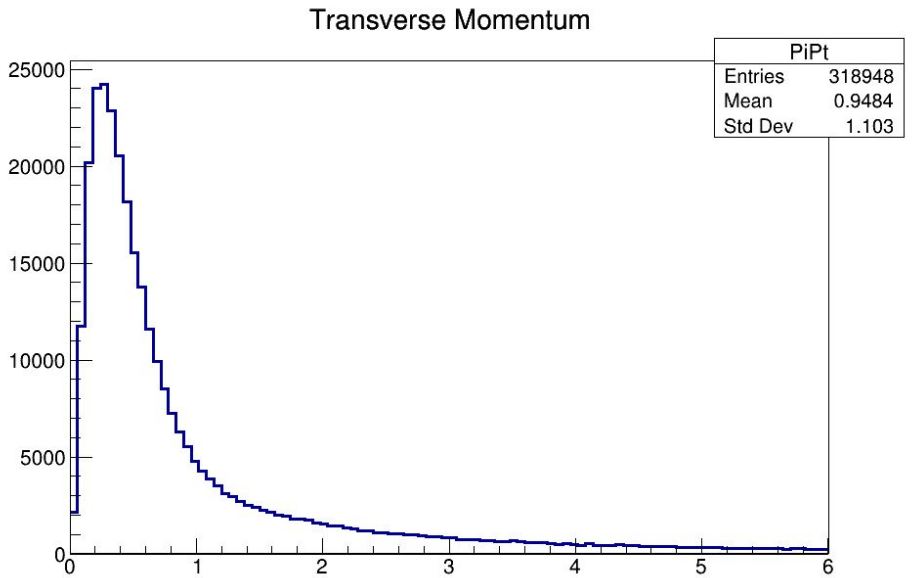
Mass and Energy of π^- :



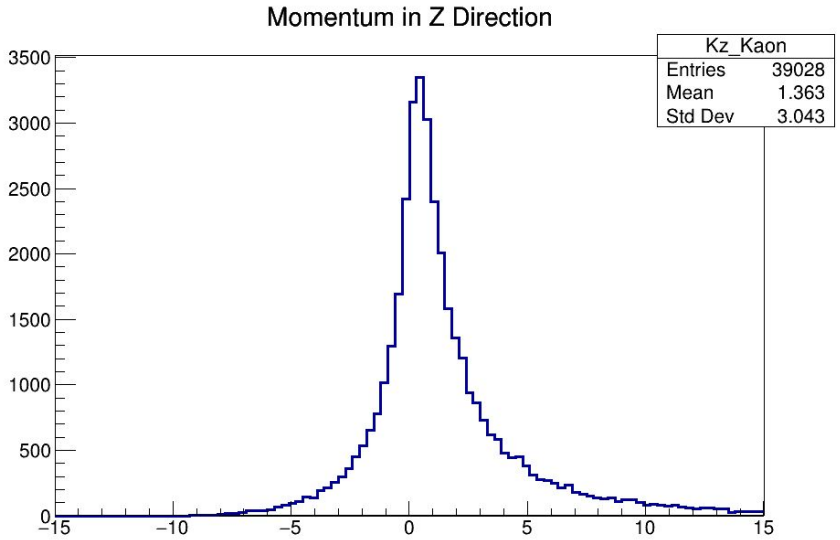


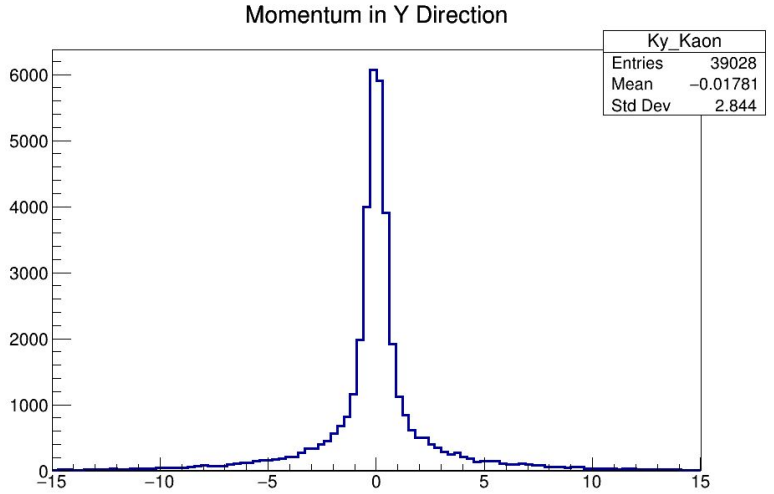
Pseudorapidity and Transverse Momentum of π^- :

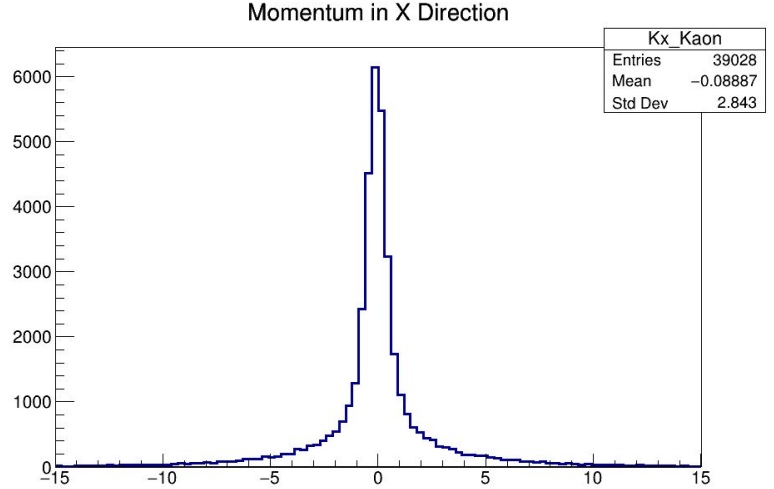




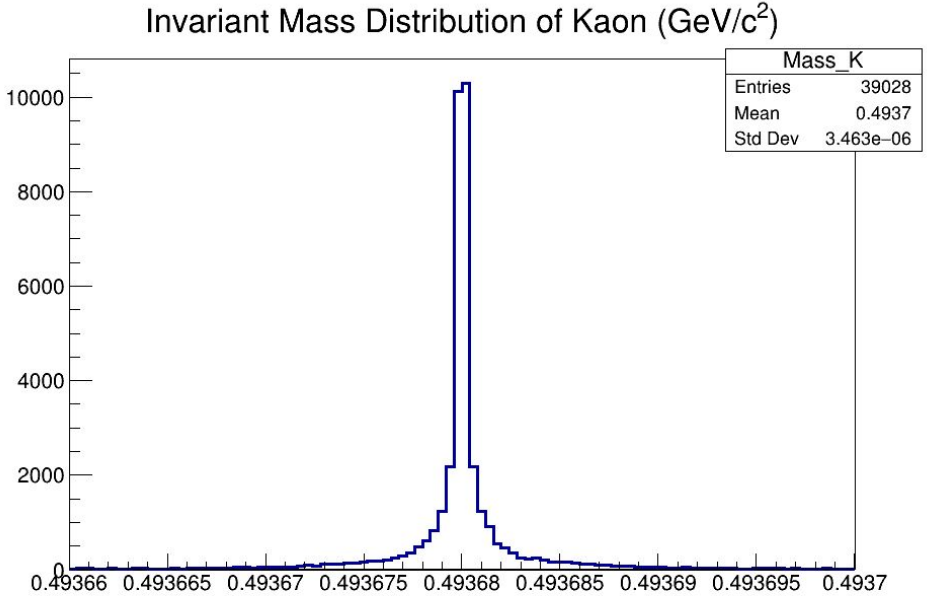
Momentum (p_x, p_y, p_z) of **K+** :

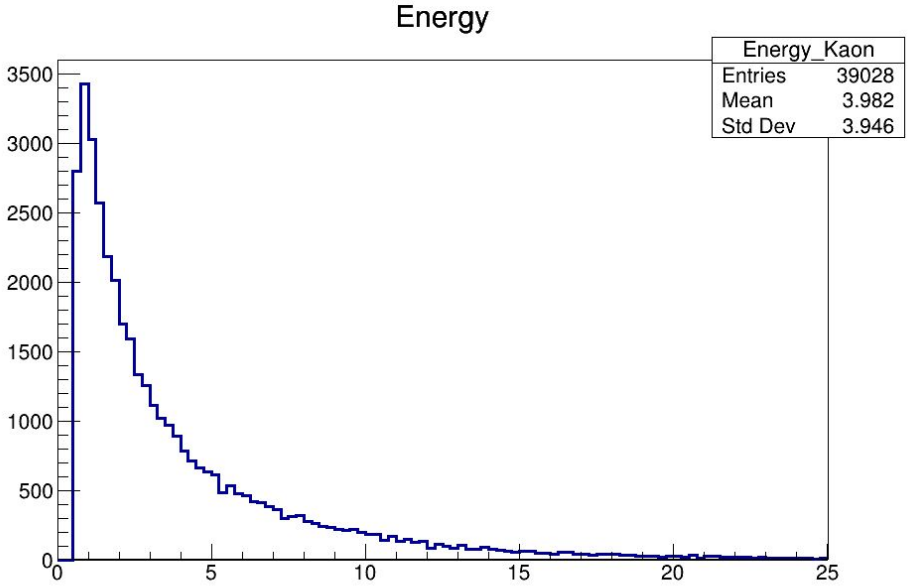




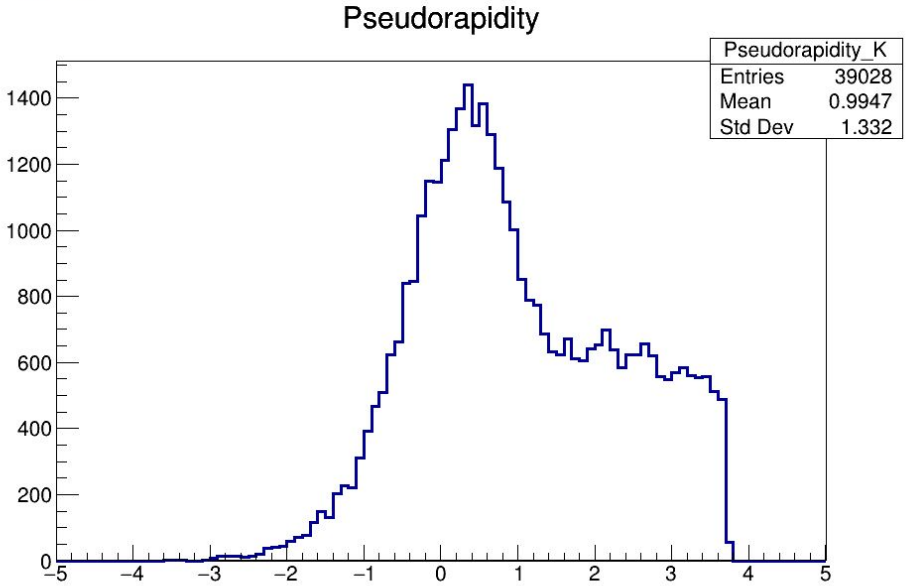


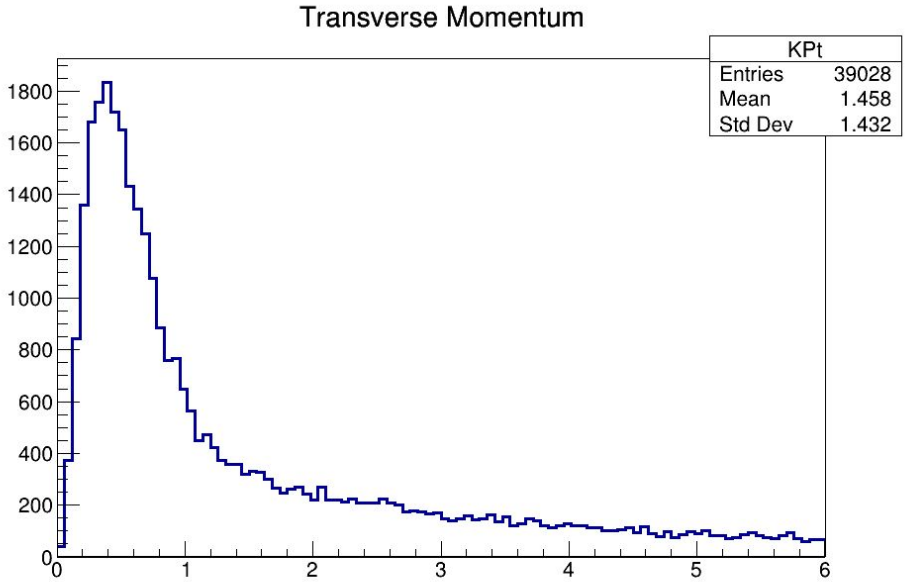
Mass and Energy of **K+** :



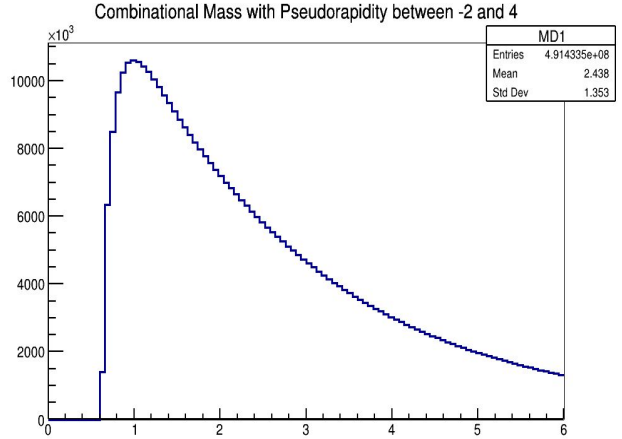


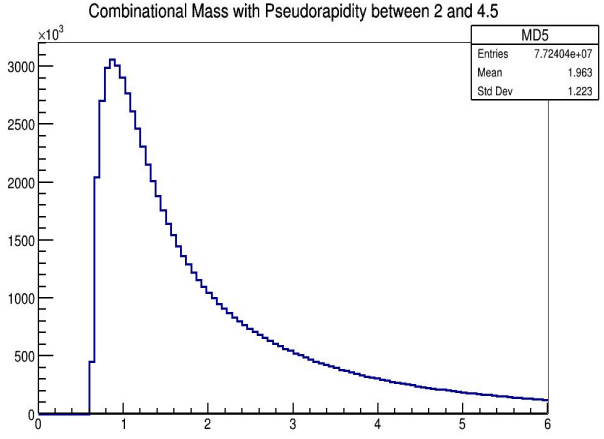
Pseudorapidity and Transverse Momentum of **K+** :

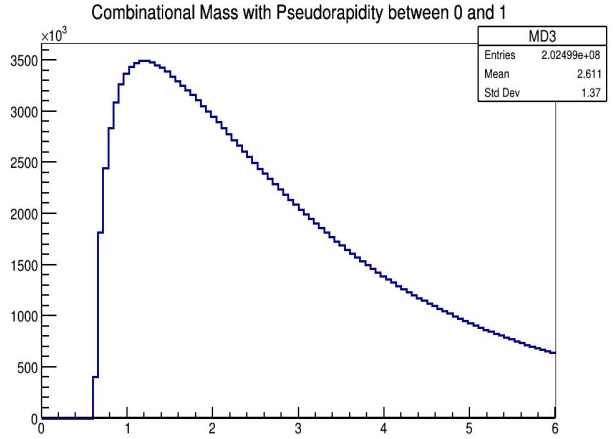


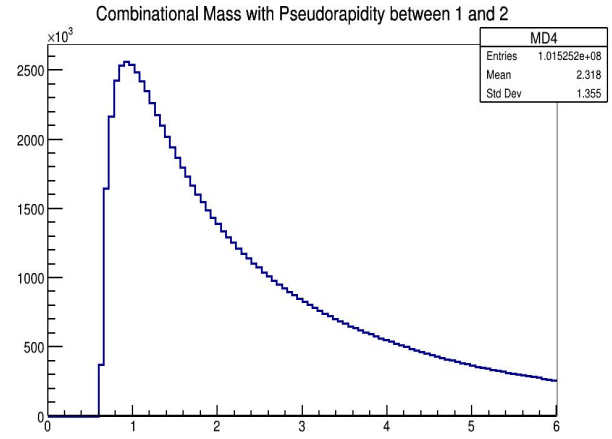


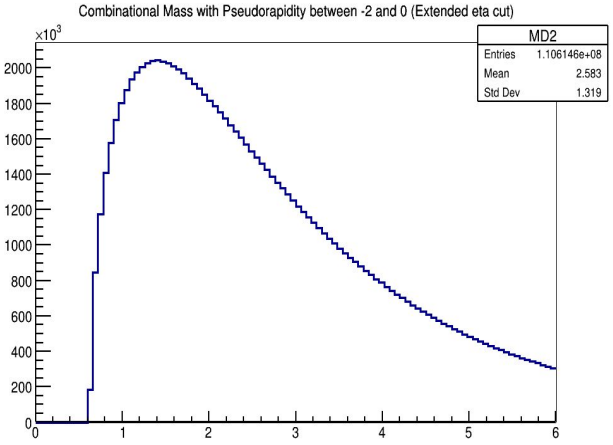
Combinational Mass with various eta cuts to observe the effect of Pseudorapidity

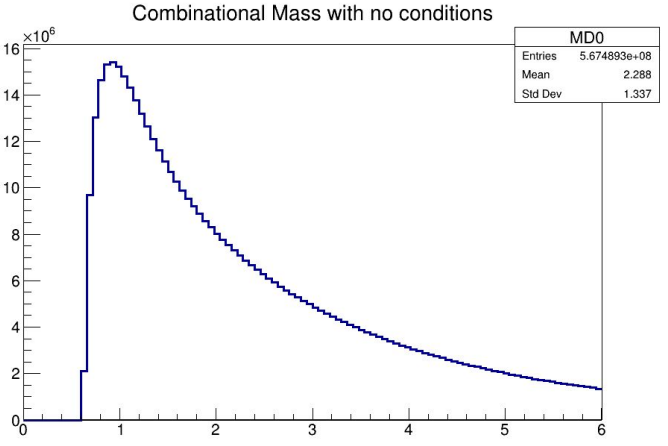


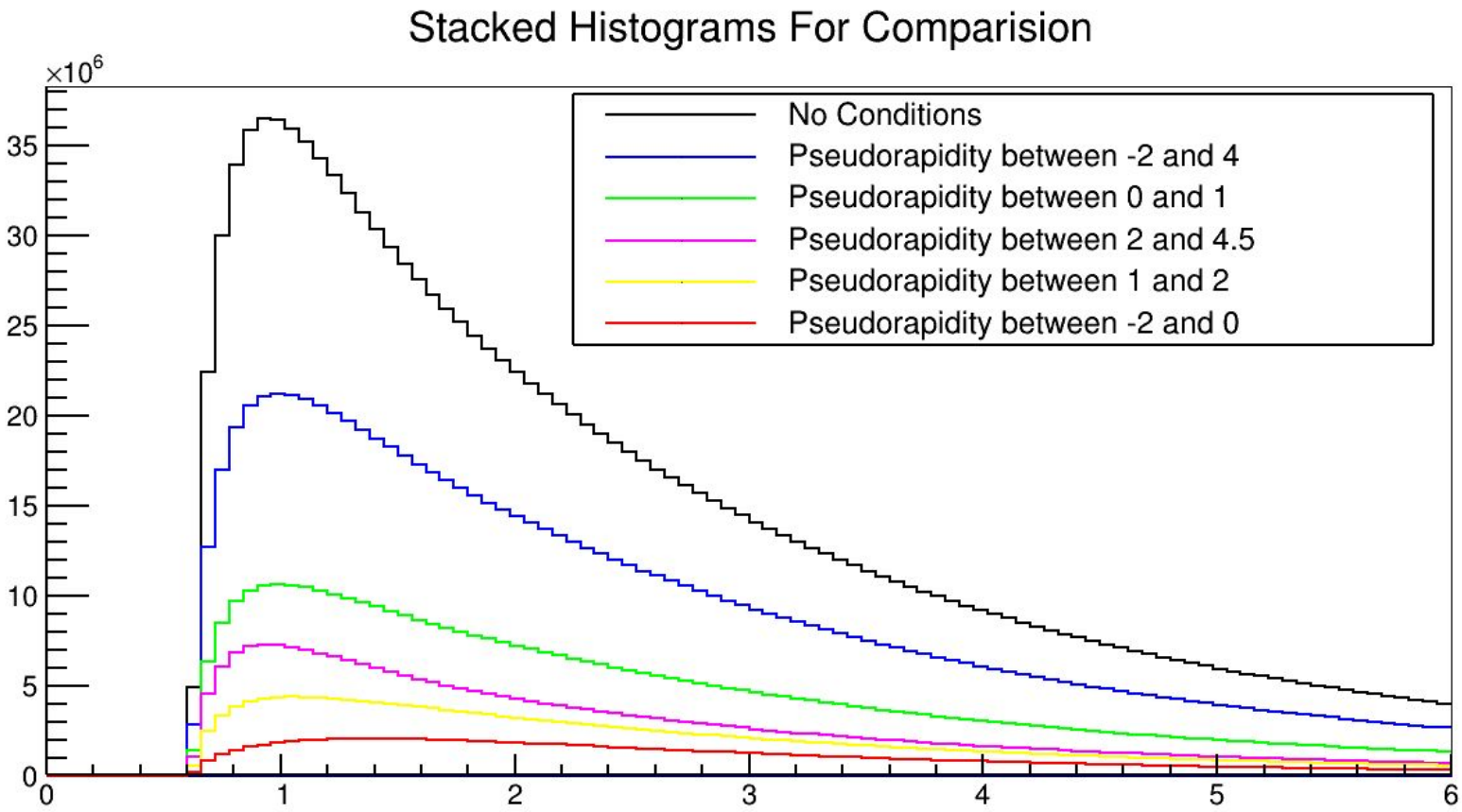










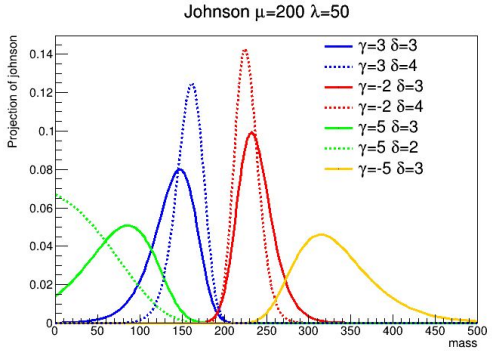

Stacked Histograms For Comparision
×10^6
35
30
25
20
15
10
5
0
0
1
2
3
4
5
6
No Conditions
Pseudorapidity between -2 and 4
Pseudorapidity between 0 and 1
Pseudorapidity between 2 and 4.5
Pseudorapidity between 1 and 2
Pseudorapidity between -2 and 0

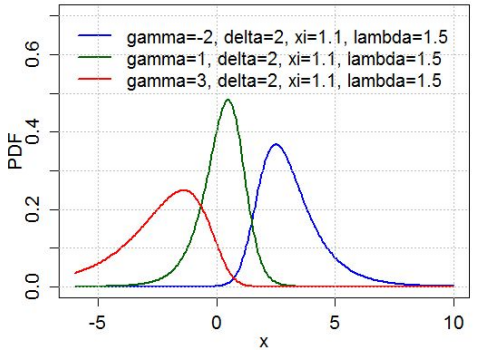
Fitting Histogram MD0 (Combinational Mass without any conditions) :

- While using RooFit we have made use of the **Johnson's SU distributions** as the Signal
- For the Background we have made use of the Chebyshev Polynomial

```
using namespace RooFit ;
using namespace std;

void fittingD0()
{
  TFile * f1 = new TFile("output.root");
  TH1 *h1 = (TH1*)f1-> Get("MD0");            //Combinational Mass with no conditions

  RooRealVar mass("mass","", 0, 6);
  RooDataHist data("data","data", RooArgList(mass),h1);

  //signal
  RooRealVar mu_J1("#mu_{J}","mean_johnson", 2.288, 0, 6);
  RooRealVar sigma_J1 ("#sigma_{J}", "sigma_johnson", 0.03, 0.0000001, 1);
  RooRealVar gamma_J1("#gamma_{J}", "gamma",  1.0, -10, 10);
  RooRealVar delta_J1("#delta_{J}", "delta", 12, 0.0000001, 20);
  RooJohnson sig("johnson1", "Johnson PDF", mass, mu_J1, sigma_J1, gamma_J1, delta_J1);

  //background
  RooRealVar a1("a1", "Slope1 of Polynomial", 0.4, -1e6, 1e6);
  RooChebychev bkg("bkg","Chebyshev Polynomial", mass,RooArgList(a1));
```

Johnson's Distribution





The **Johnson's S_U-distribution** is a four-parameter family of probability distributions, namely :

- ***Mean*** is the arithmetic mean of the distribution.
- ***Stdev*** is the standard deviation of the distribution (Positive value).
- ***Skew*** is the skewness of the distribution.
- ***Kurt*** is the kurtosis of the distribution (Positive value).

This PDF results from transforming a normally distributed variable x to this form:

$$z = \gamma + \delta \sinh^{-1}\left(\frac{x-\mu}{\lambda}\right)$$

The resulting PDF is

$$\mathrm{PDF}[\text{Johnson } S_U] = \frac{\delta}{\lambda\sqrt{2\pi}} \frac{1}{\sqrt{1+\left(\frac{x-\mu}{\lambda}\right)^2}} \exp\left[-\frac{1}{2}\left(\gamma + \delta \sinh^{-1}\left(\frac{x-\mu}{\lambda}\right)\right)^2\right].$$

It is often used to fit a mass difference for charm decays, and therefore the variable x is called "mass" in the implementation. A mass threshold allows to set the PDF to zero to the left of the threshold.

RooJohnson( )

RooJohnson (const char ***name**, const char *title, **RooAbsReal** &mass, **RooAbsReal** &mu, **RooAbsReal** &lambda, **RooAbsReal** &gamma, **RooAbsReal** &delta, **double** massThreshold=-std::numeric_limits< **double** >::max())

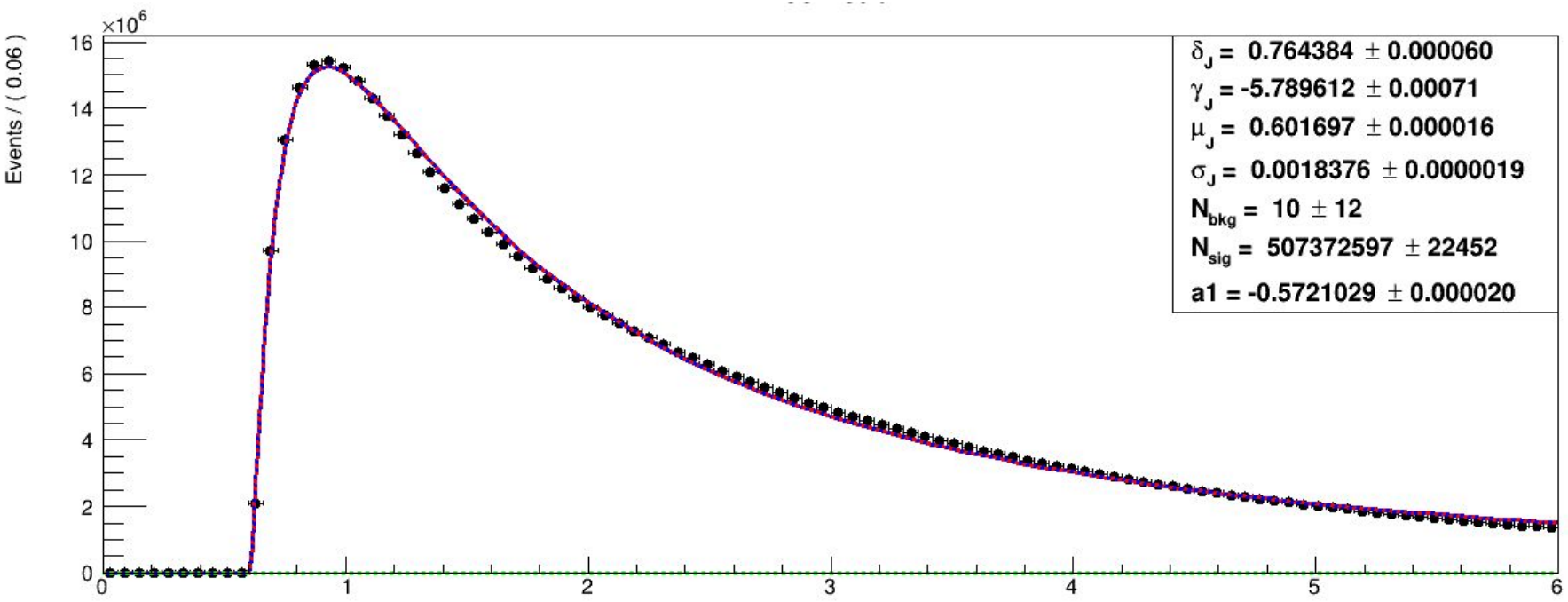
Construct a new Johnson PDF. More...

Enumerator	
kMass	
kMean	
kLambda	
kGamma	
kDelta	

Parameters	$\gamma, \xi, \delta > 0, \lambda > 0$ (real)
Support	$-\infty$ to $+\infty$
PDF	$\frac{\delta}{\lambda\sqrt{2\pi}} \frac{1}{\sqrt{1+\left(\frac{x-\xi}{\lambda}\right)^2}} e^{-\frac{1}{2}\left(\gamma+\delta \sinh^{-1}\left(\frac{x-\xi}{\lambda}\right)\right)^2}$
CDF	$\Phi\left(\gamma+\delta \sinh^{-1}\left(\frac{x-\xi}{\lambda}\right)\right)$
Mean	$\xi - \lambda \exp\frac{\delta^{-2}}{2} \sinh\left(\frac{\gamma}{\delta}\right)$
Median	$\xi + \lambda \sinh\left(-\frac{\gamma}{\delta}\right)$
Variance	$\frac{\lambda^2}{2}(\exp(\delta^{-2})-1)\left(\exp(\delta^{-2})\cosh\left(\frac{2\gamma}{\delta}\right)+1\right)$

name	Name that identifies the PDF in computations
title	Title for plotting
mass	The variable of the PDF. Often this is a mass.
mu	Location parameter of the Gaussian component.
lambda	Width parameter (>0) of the Gaussian component.
gamma	Shape parameter that distorts distribution to left/right.
delta	Shape parameter (>0) that determines strength of Gaussian-like component.
massThreshold	Set PDF to zero below this threshold.

Fitted Histogram



Warnings/Further Questions :

1. ROOT shows a Warning Message while reading the Pythia files (no dictionary)

2. There are certain Zombie files present.

```
Warning in <TClass::Init>: no dictionary for class podio::CollectionIDTable is available
Warning in <TClass::Init>: no dictionary for class dd4pod::Geant4ParticleData is available
Warning in <TClass::Init>: no dictionary for class dd4pod::VectorXYZ is available
Warning in <TClass::Init>: no dictionary for class eic::ReconstructedParticleData is available
Warning in <TClass::Init>: no dictionary for class eic::Index is available
Warning in <TClass::Init>: no dictionary for class eic::VectorXYZ is available
Warning in <TClass::Init>: no dictionary for class eic::Weight is available
Warning in <TClass::Init>: no dictionary for class eic::Direction is available
Warning in <TClass::Init>: no dictionary for class eic::InclusiveKinematicsData is available
Warning in <TClass::Init>: no dictionary for class eic::ClusterData is available
Warning in <TClass::Init>: no dictionary for class eic::CovXYZ is available
Warning in <TClass::Init>: no dictionary for class eic::VectorPolar is available
Warning in <TClass::Init>: no dictionary for class eic::MergedClusterRelationsData is available
Warning in <TClass::Init>: no dictionary for class eic::ClusterLayerData is available
Warning in <TClass::Init>: no dictionary for class eic::TrackParametersData is available
Warning in <TClass::Init>: no dictionary for class eic::FloatPair is available
Warning in <TClass::Init>: no dictionary for class eic::TrajectoryData is available
Warning in <TClass::Init>: no dictionary for class eic::VectorXYZT is available
```

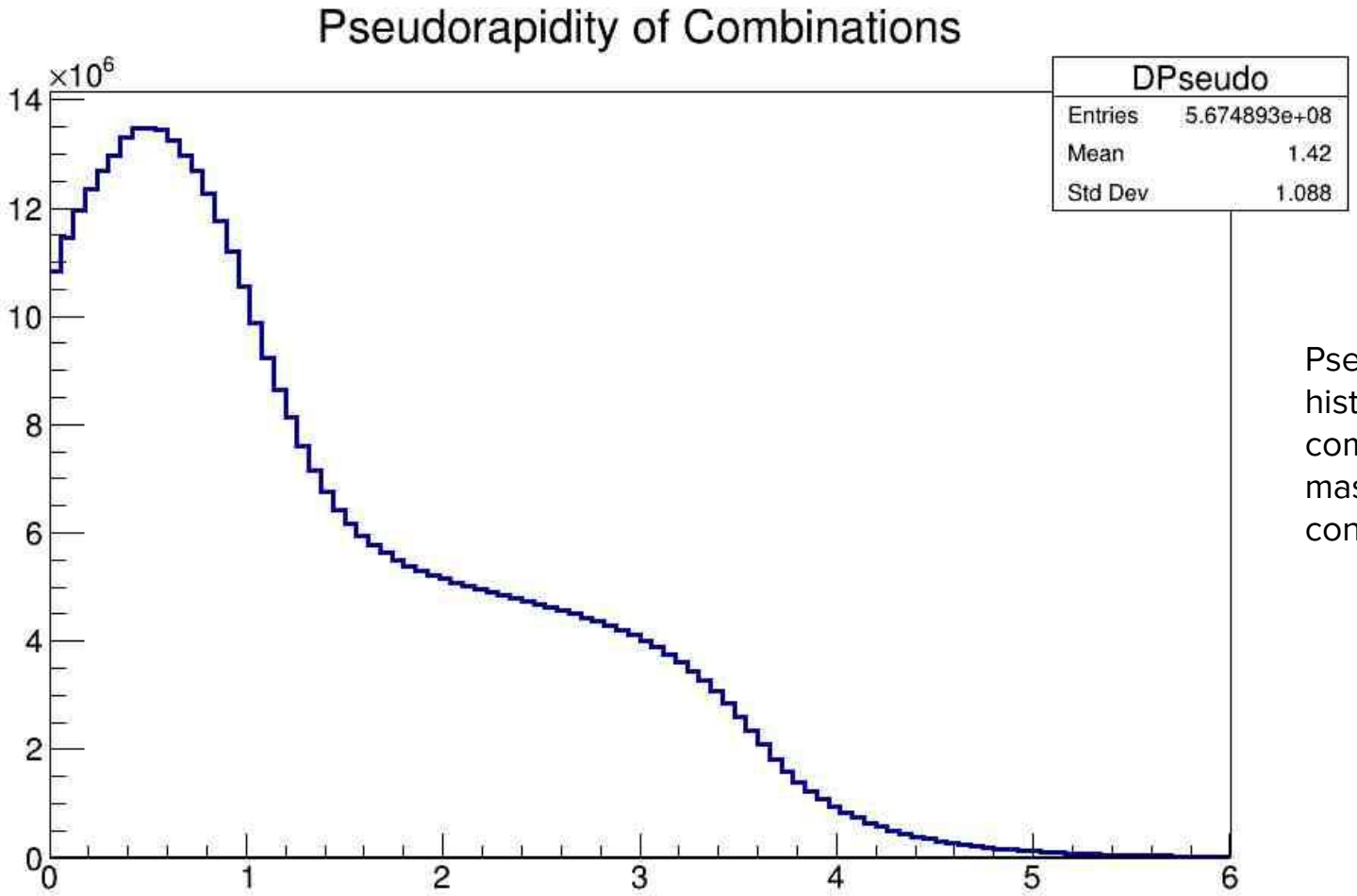
```
Warning in <TFile::Init>: no keys recovered, file has been made a Zombie
Error in <TTreeReader::TTreeReader>: No TTree called events was found in the selected TDirectory.
 p:0 r: 0 s: 0
Number of D0 Mass Recreations : 0
```

Summary

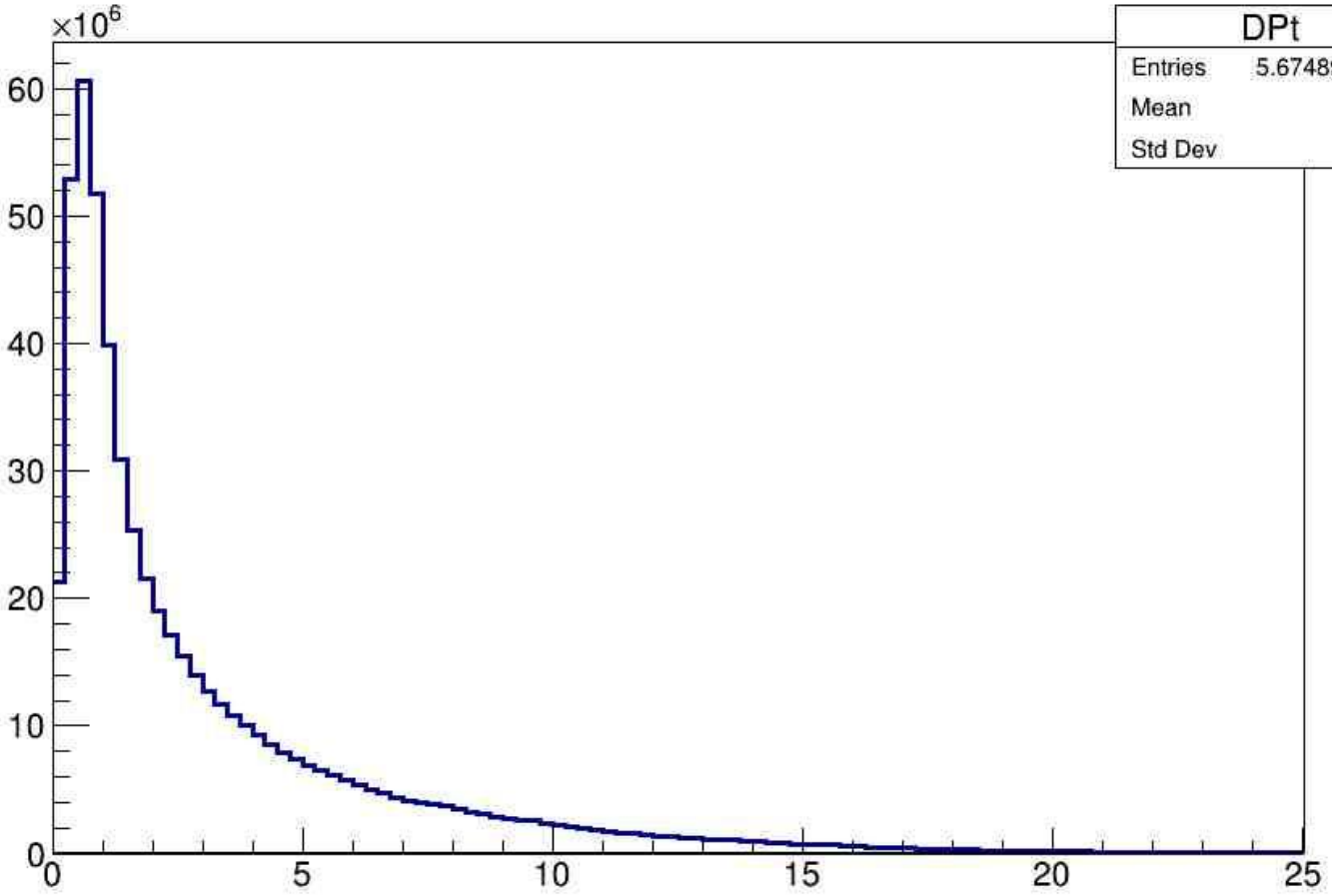
- We have successfully reconstructed mass of D0 meson from input files and analysed them with different pseudorapidity and transverse momentum cuts.

- We have compared pseudorapidity histograms using stacking method.

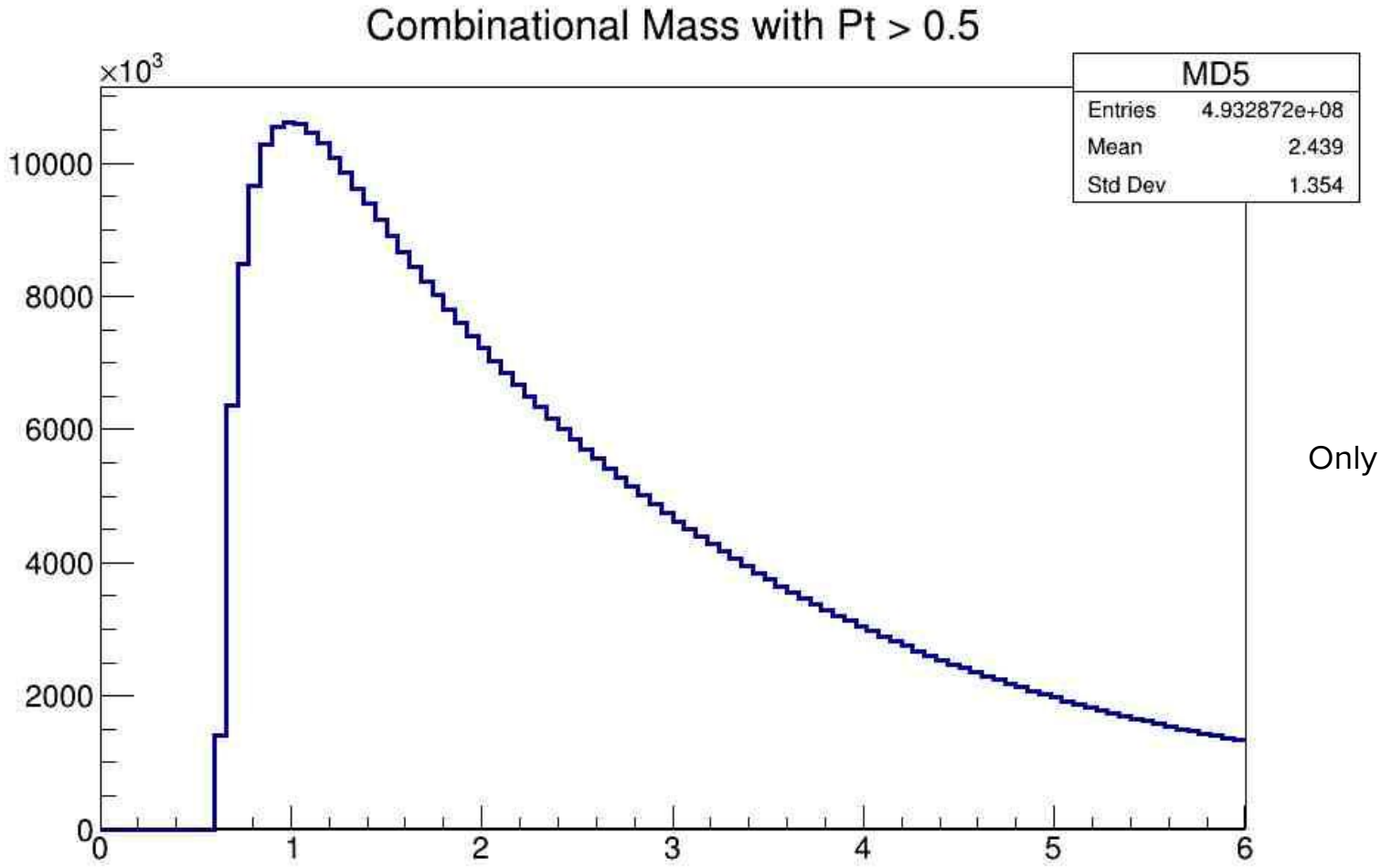
- Johnson distribution is used to fitting our final histogram.



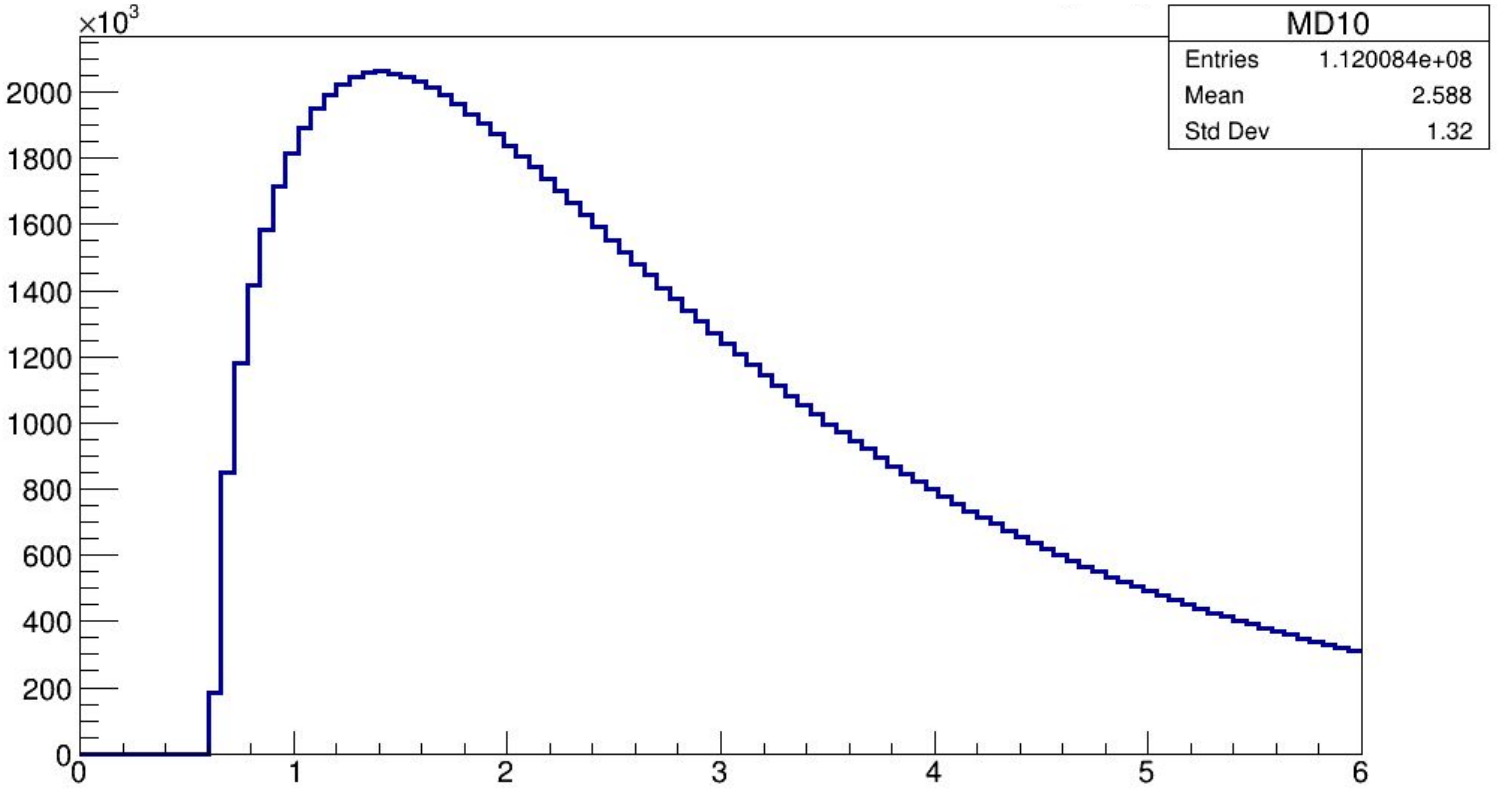
Pseudorapidity histogram for the combinational mass without any conditions.



Transverse Momentum of Combinational mass without any conditions.


Combinational Mass with Pt > 0.5
×10^3
MD5
Entries 4.932872e+08
Mean 2.439
Std Dev 1.354
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
6

Only Pt Cut



Both Pt cut and a Pseudorapidity conditions are present.

Reading the ‘Generated Particle’ Tree and access various parameters :

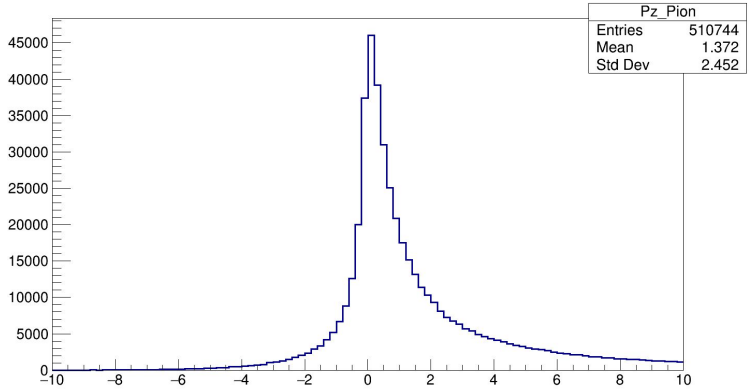
```
TTreeReaderArray<Float_t> px(myReader, "GeneratedParticles.p.x");
TTreeReaderArray<Float_t> py(myReader, "GeneratedParticles.p.y");
TTreeReaderArray<Float_t> pz(myReader, "GeneratedParticles.p.z");
TTreeReaderArray<Float_t> energy(myReader, "GeneratedParticles.energy");
TTreeReaderArray<Int_t> pid(myReader, "GeneratedParticles.pid");
TTreeReaderArray<Float_t> momentum(myReader, "GeneratedParticles.momentum");
```

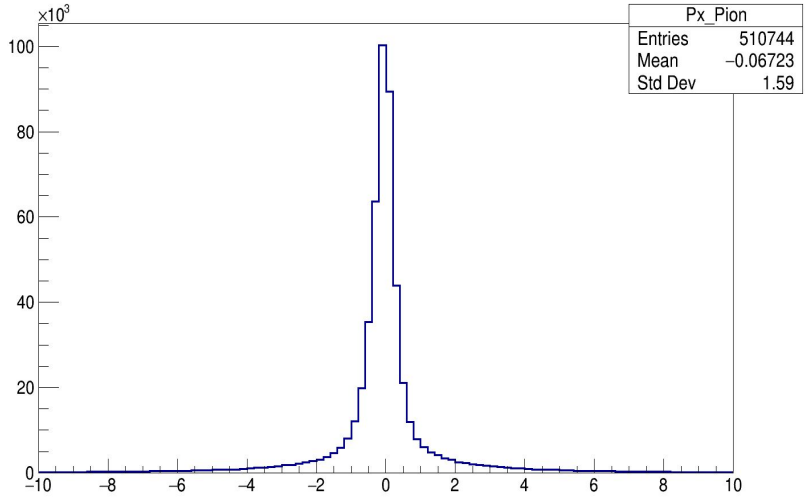
Reading Generated Particles Tree

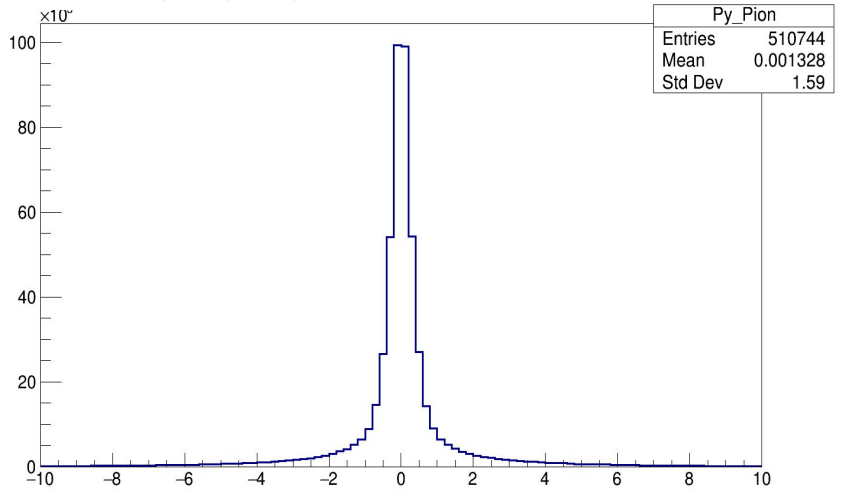
```
TTreeReaderArray<Float_t> px(myReader, "ReconstructedParticles.p.x");
TTreeReaderArray<Float_t> py(myReader, "ReconstructedParticles.p.y");
TTreeReaderArray<Float_t> pz(myReader, "ReconstructedParticles.p.z");
TTreeReaderArray<Float_t> energy(myReader, "ReconstructedParticles.energy");
TTreeReaderArray<Int_t> pid(myReader, "ReconstructedParticles.pid");
TTreeReaderArray<Float_t> momentum(myReader, "ReconstructedParticles.momentum");
```

Reading Recreated Particles Tree

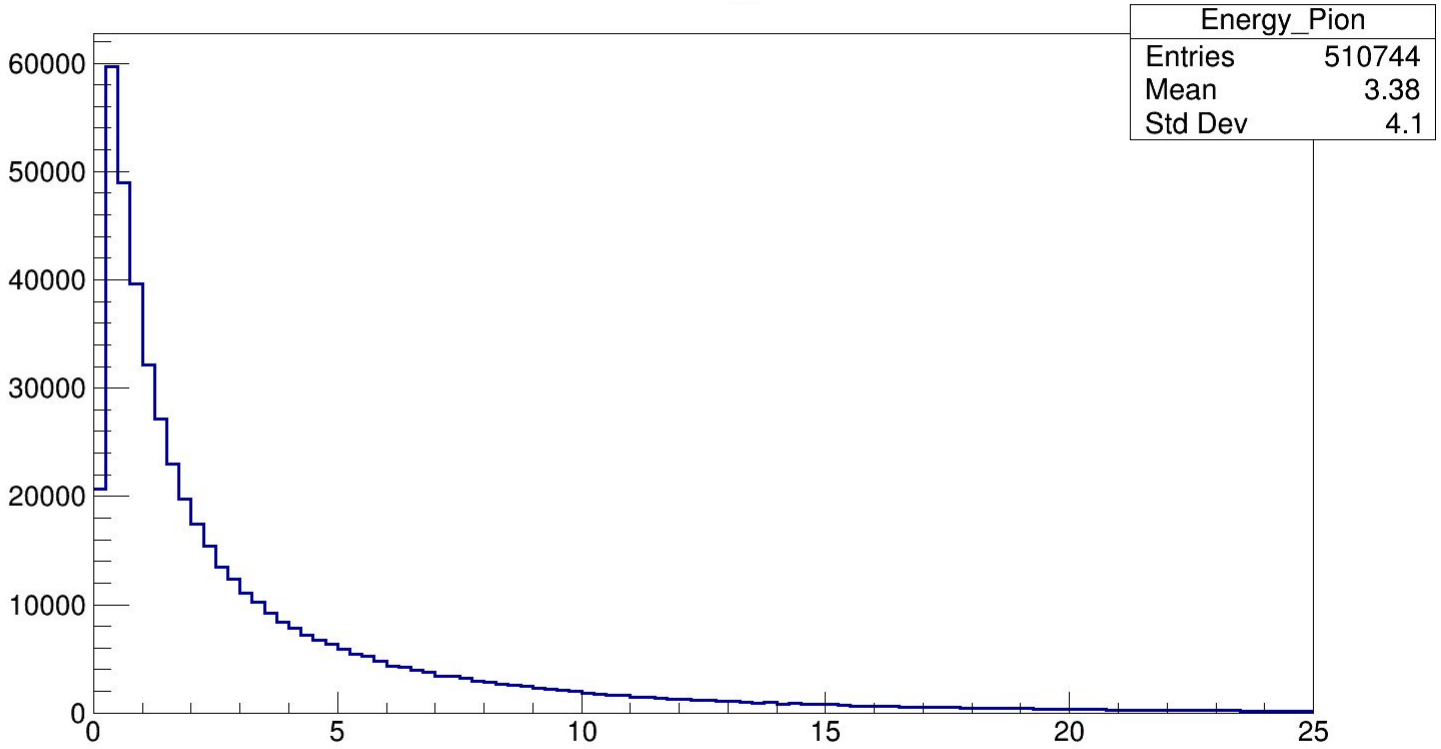
Generated Momentum (p_x, p_y, p_z) of π^+ :



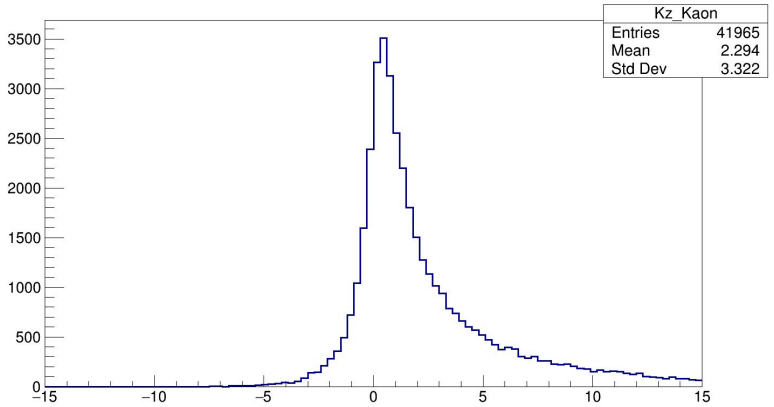


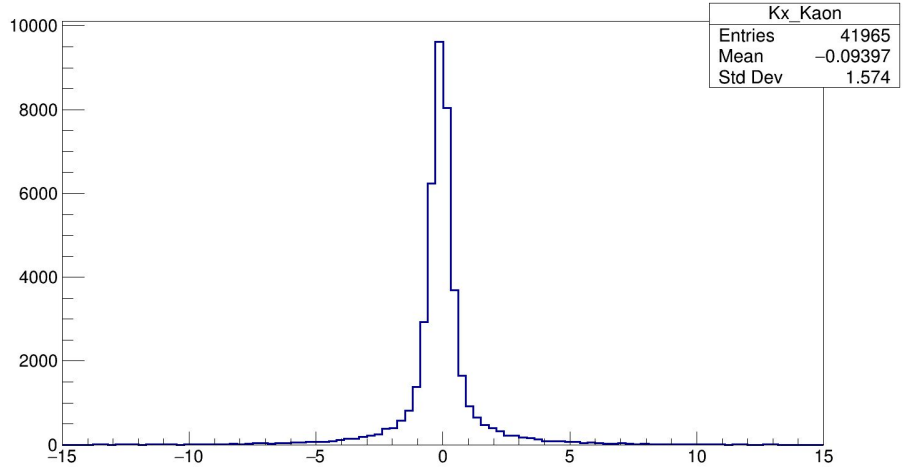


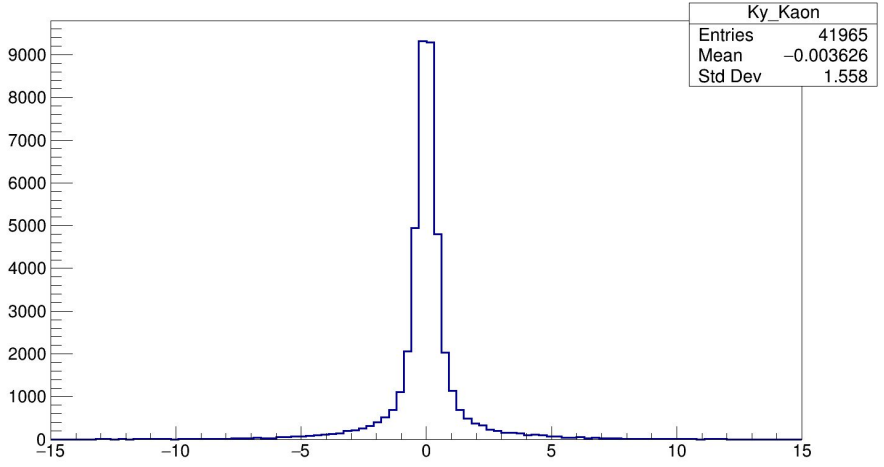
Generated Energy of π^+ :



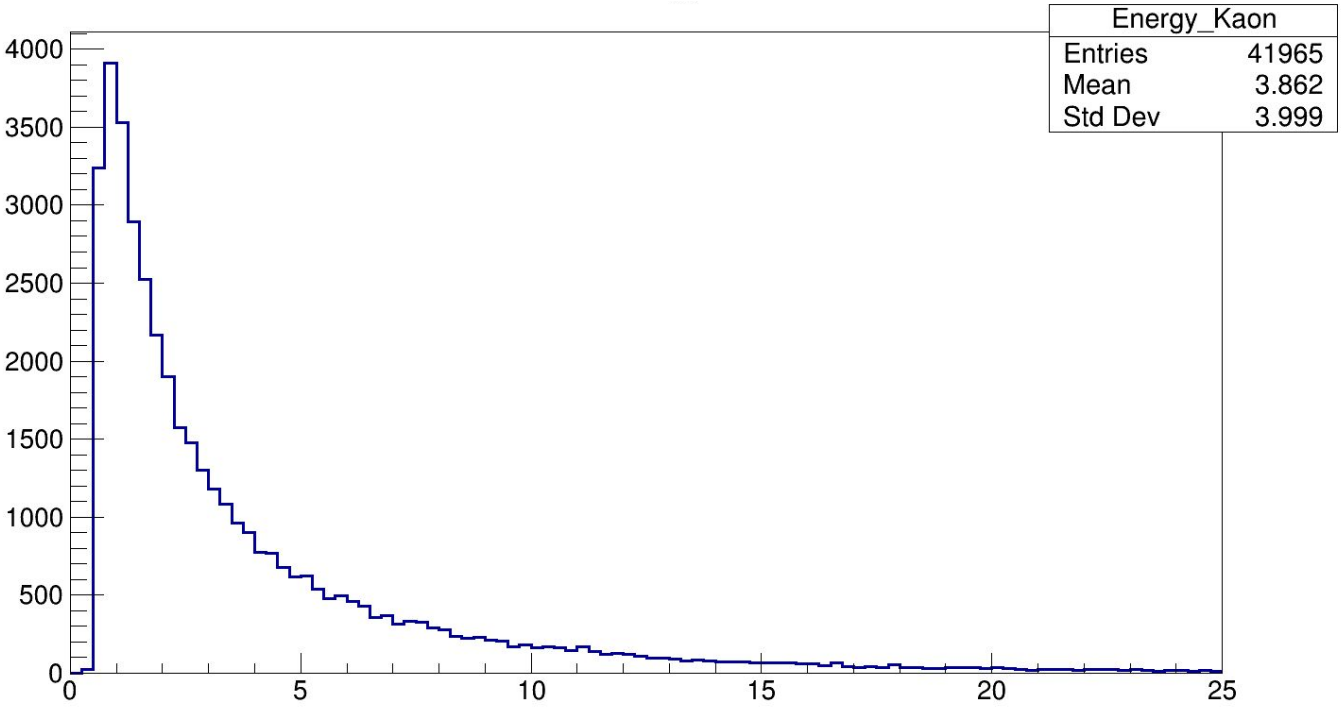
Generated Momentum (p_x, p_y, p_z) of K^- :



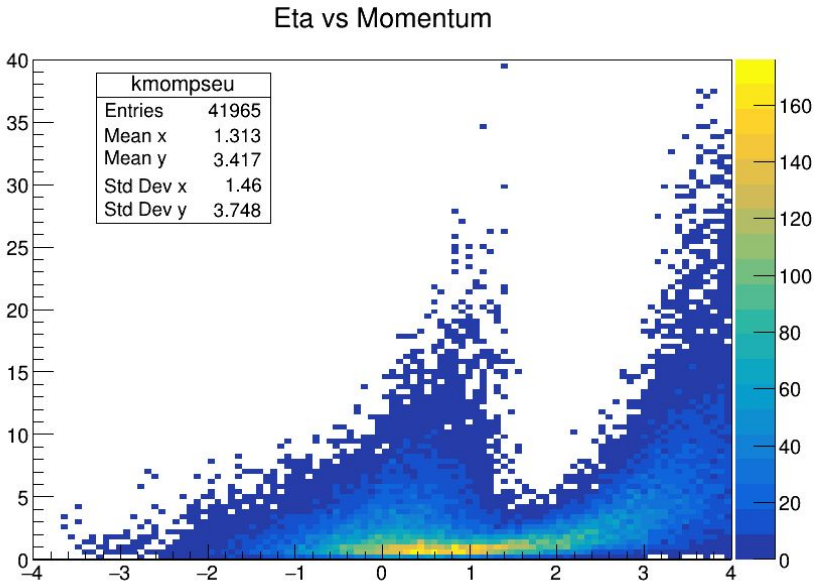




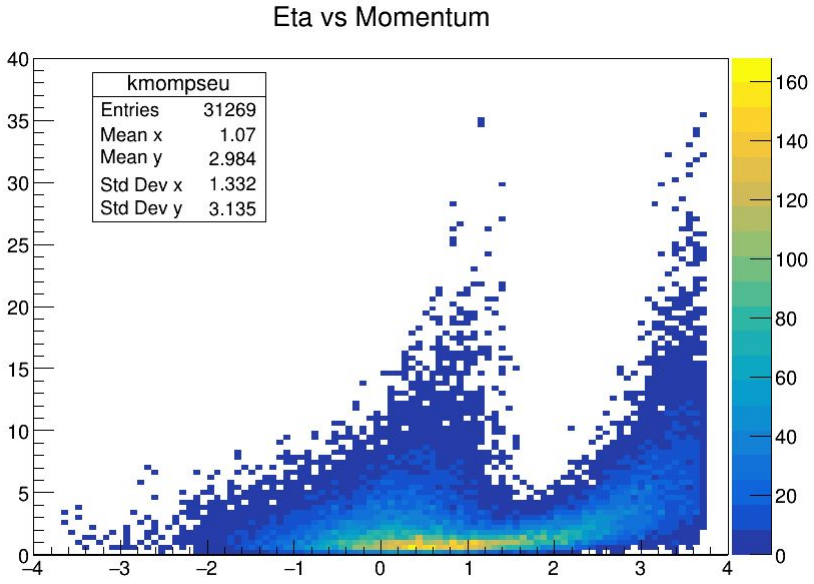
Generated Energy of K^- :



Pseudorapidity Vs Momentum for K-



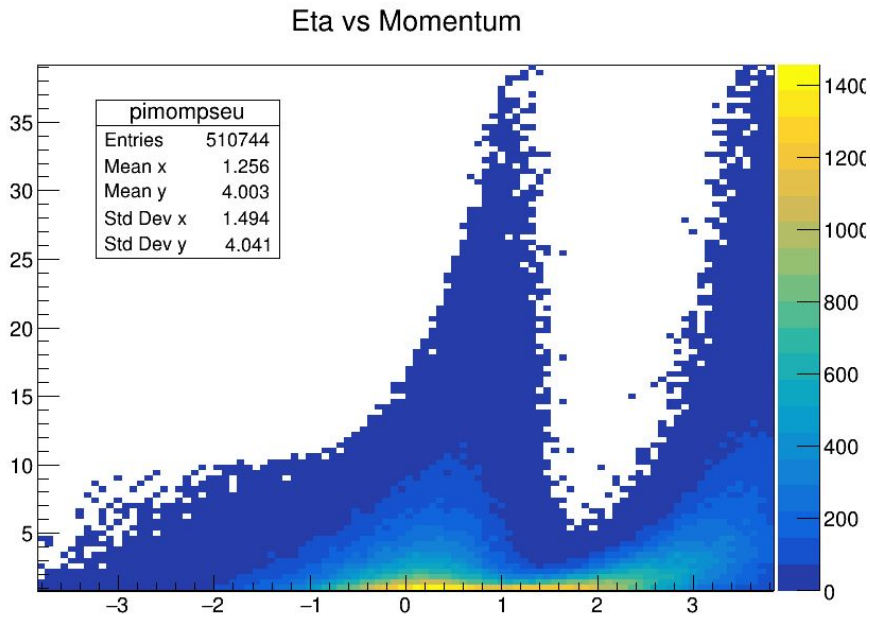
K- for Generated Particles



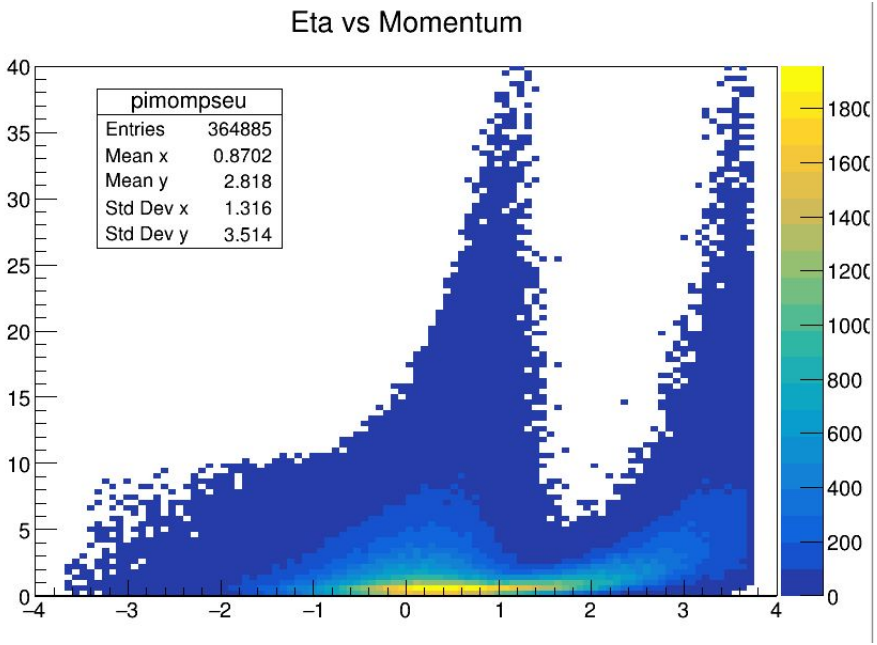
K- for Reconstructed Particles

X-axis : Eta
Y- axis : Momentum

Pseudorapidity Vs Momentum for Pi+



Pi+ for Generated Particles



Pi+ for Reconstructed Particles

X-axis : Eta
Y- axis : Momentum

Reference from yellow report

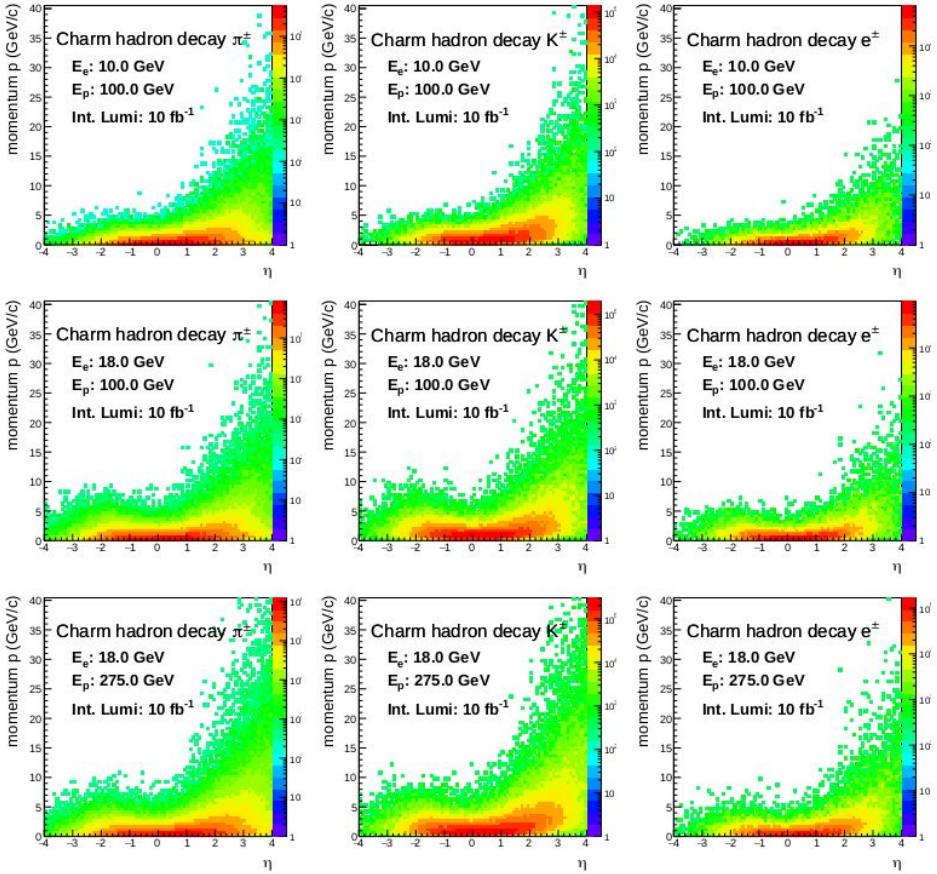
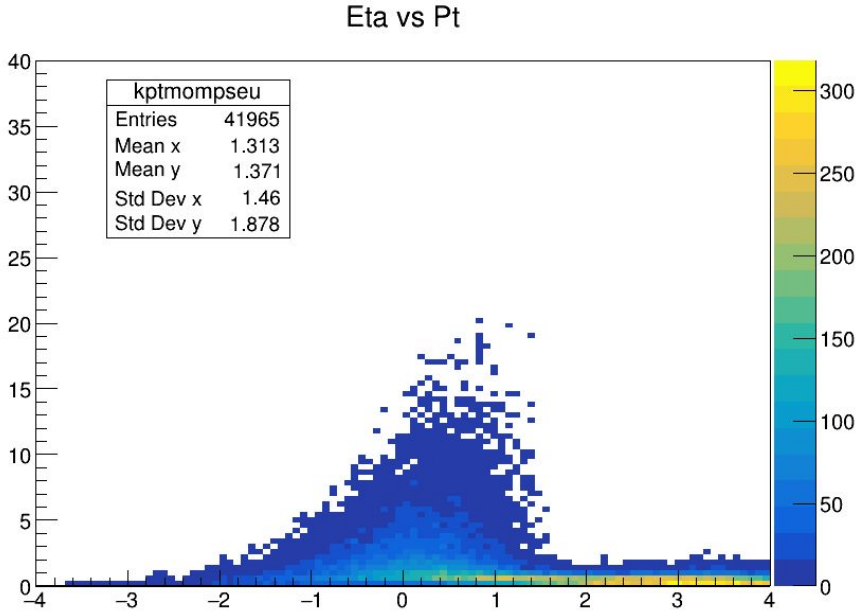
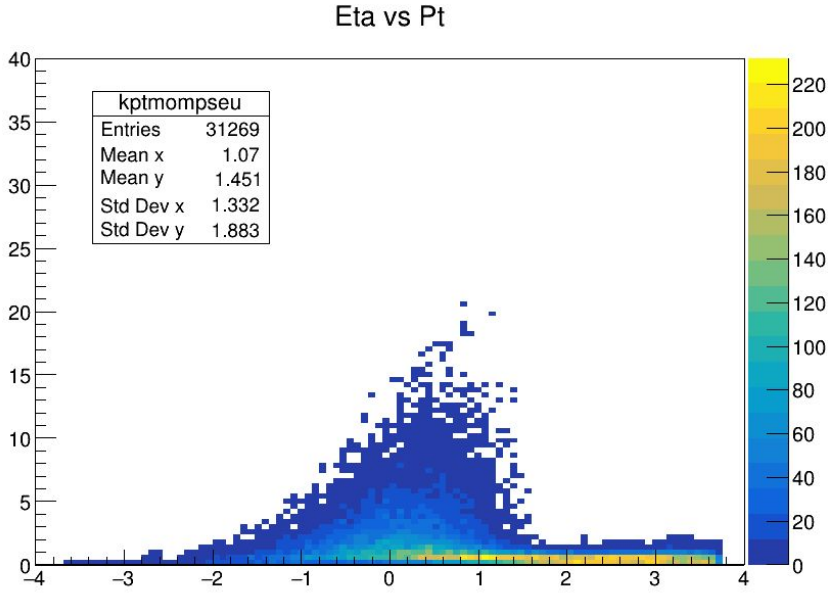


Figure 8.42: Momentum vs pseudorapidity for the decay products of D^0 mesons for beam energies of 10x100 GeV (top row), 18x100 GeV (middle row), and 18x275 GeV (bottom row). Charged pions are in the left column, charged kaons in the middle column, and electrons/positrons in the right column. Counts have been scaled to correspond to an integrated luminosity of 10 fb^{-1}.

Pseudorapidity Vs Pt for K-



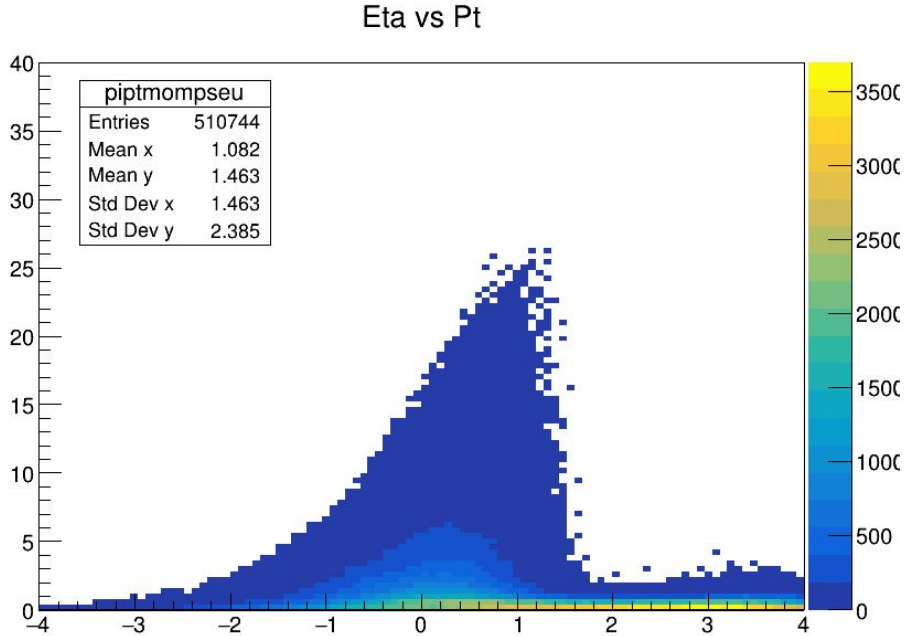
K- for Generated Particles



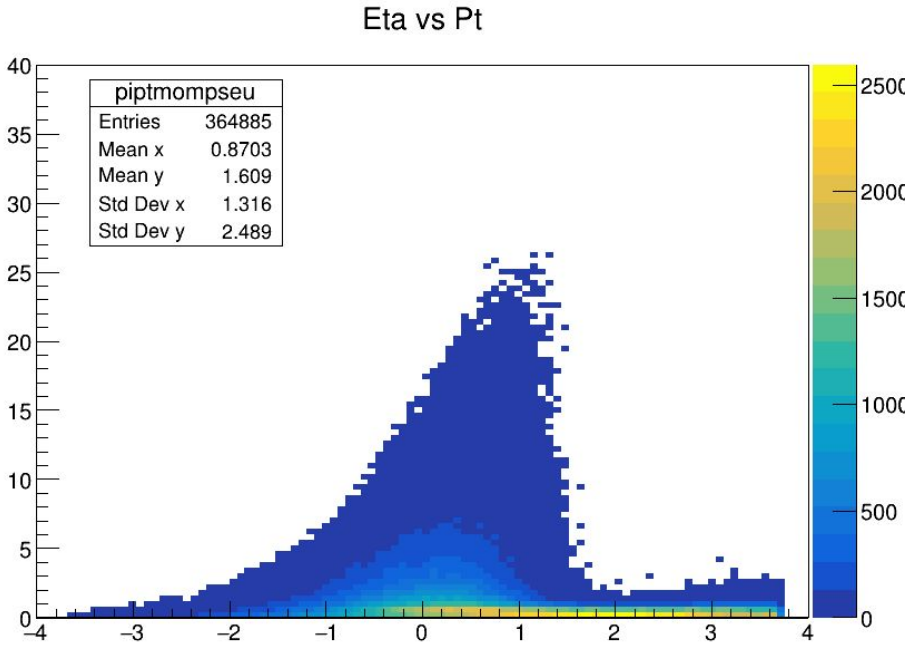
K- for Reconstructed Particles

X-axis : Eta
Y- axis : Transverse Momentum

Pseudorapidity Vs Pt for Pi+



Pi+ for Generated Particles



Pi+ for Reconstructed Particles

X-axis : Eta

Y- axis : Transverse Momentum