Reconstruction of D^0 Mesons
&
Production of Jets and substructures

Siddharth S. (MS Physics, Malaviya National Institute of Technology, Jaipur, India)
Mihir P. (MS Physics, Sardar Vallabhbhai National Institute of Technology, Surat, India)
Siddharth J. (MS Physics, Jamia Millia Islamia University, New Delhi, India)

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

Students: **Siddharth Singh & Mihir Patel**

Past supervisor: **Dr. Kavita Lalwani (MNIT, jaipur)**

Process followed :

1. Download the ROOT files (Deep Inelastic Scattering, Neutral Current) from :
   https://eic.phy.anl.gov/ip6/productions/physics.html#deep-inelastic-scattering
   Path : eictest / ATHENA / EVGEN / DIS / NC / 10x100 / minQ2=1000

2. Downloaded files:
   geometry: master : 10x100 : minQ2 = 1000

Work done till now:

I) Reconstruction of D^o mesons and the analysis codes are written by **Siddharth Singh** and **Mihir Patel**.

II) Jet studies are done by **Siddharth Jain**, under the guidance of **Dr. Manjit Kaur (Panjab University)**, **Dr. Ritu aggarwal** (Savitribai Phule Pune University).

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 …..etc!

Some of the files were zombies/unreadable, only those files are selected for which no error message is shown.

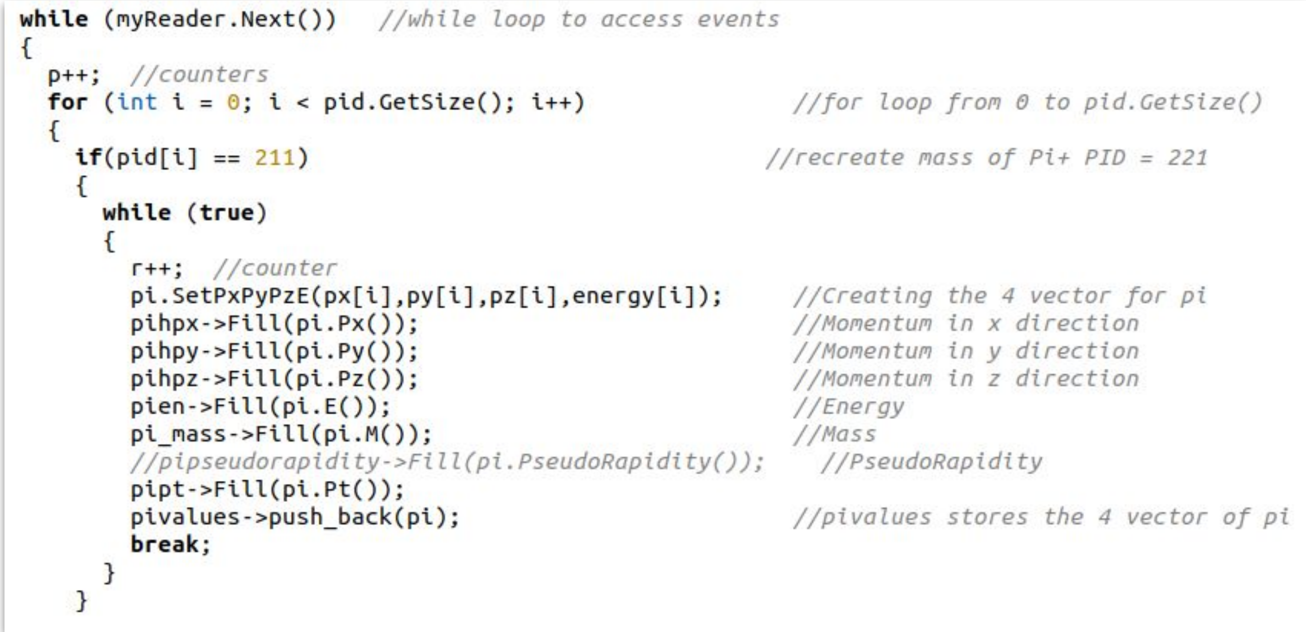
Reconstructing particles by PID check

Reconstructed data is used.

Pions four momentum, mass and pseudorapidity are stored.

$\mathbf{K}^-$ PID : -321

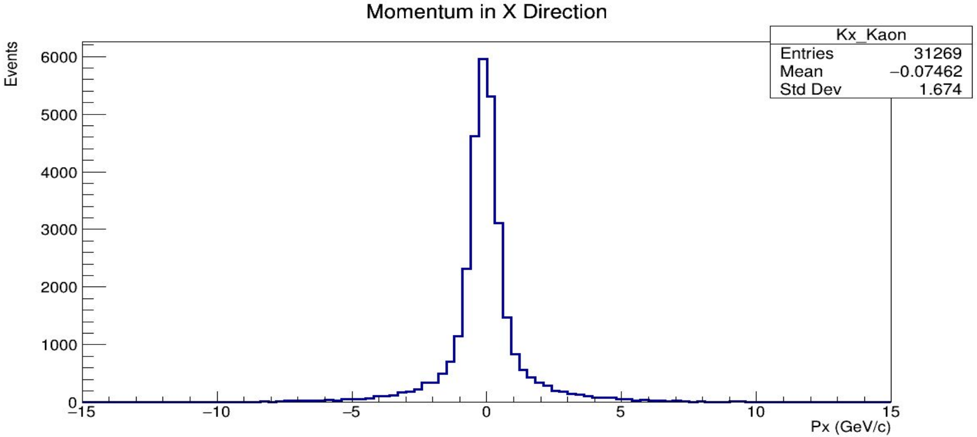
$\boldsymbol{\pi}^+$ PID: 221



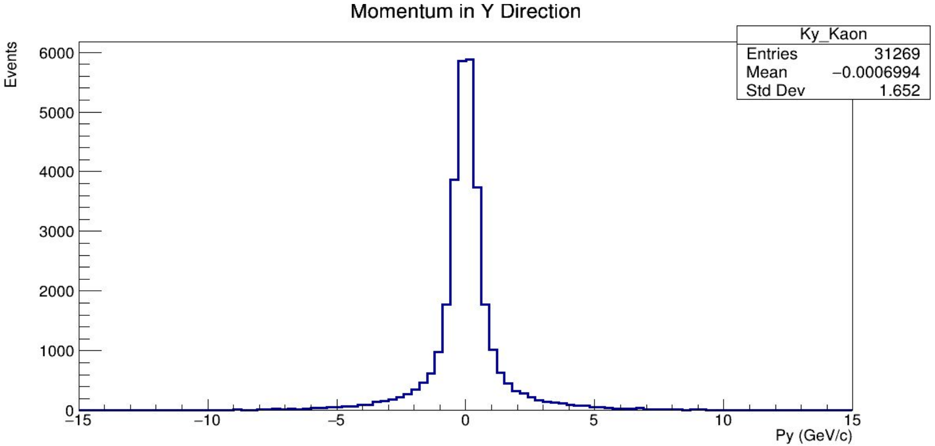
```
while (myReader.Next())   //while loop to access events
{
  p++;  //counters
  for (int i = 0; i < pid.GetSize(); i++)               //for loop from 0 to pid.GetSize()
  {
    if(pid[i] == 211)                                 //recreate mass of Pi+ PID = 221
    {
      while (true)
      {
        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;
      }
    }
```

Event Loop for PID Check for Pions : (PID = 211)

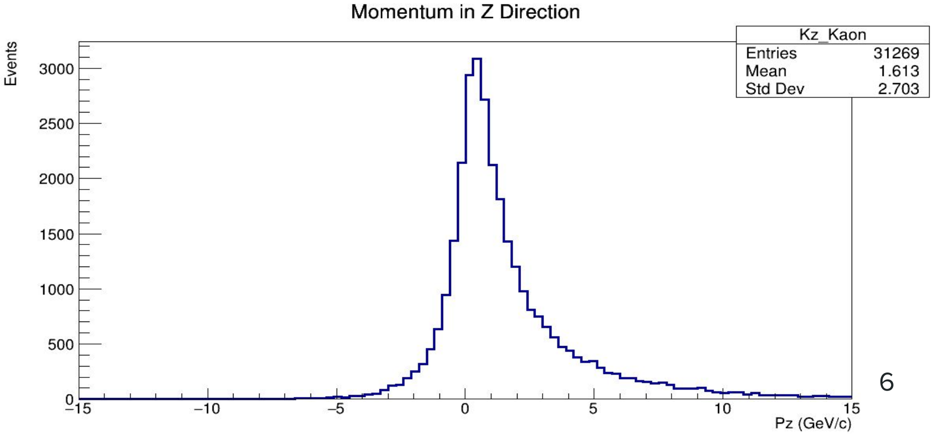
Momentum (p_x, p_y, p_z) of K^- :



Momentum in X Direction

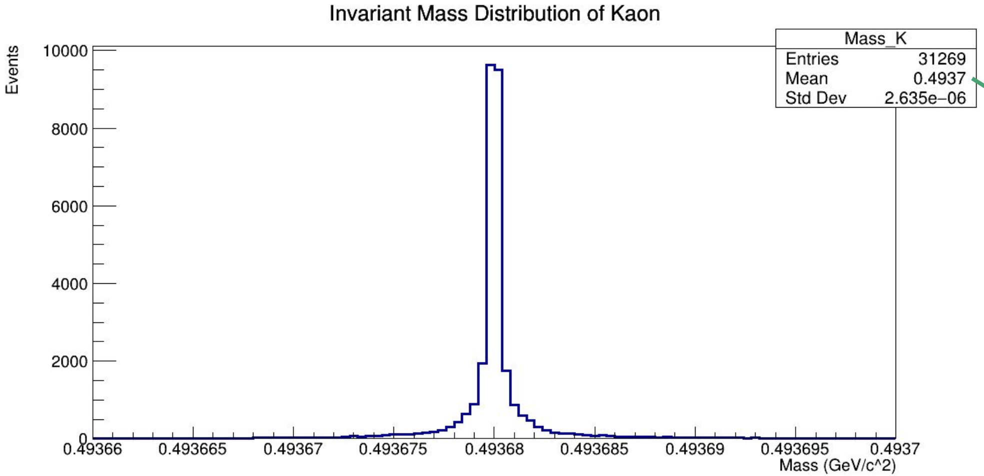


Momentum in Y Direction

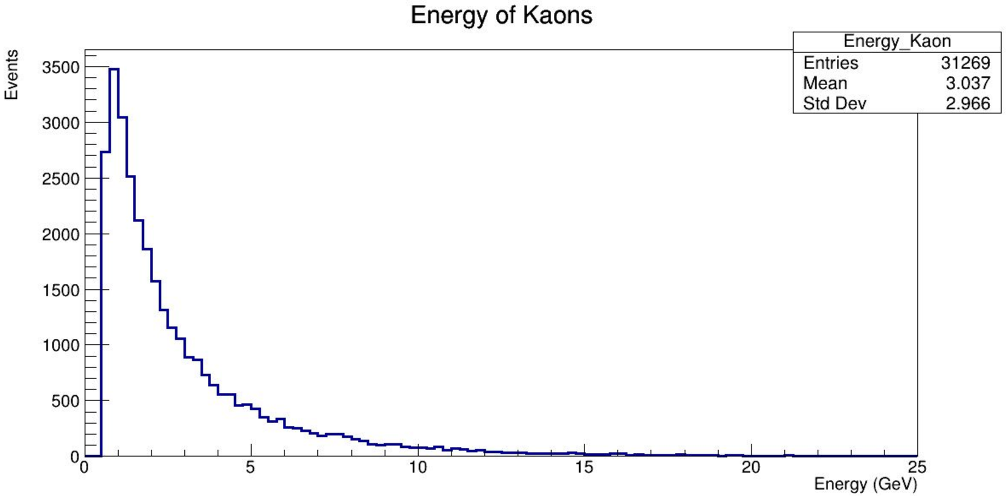


Momentum in Z Direction

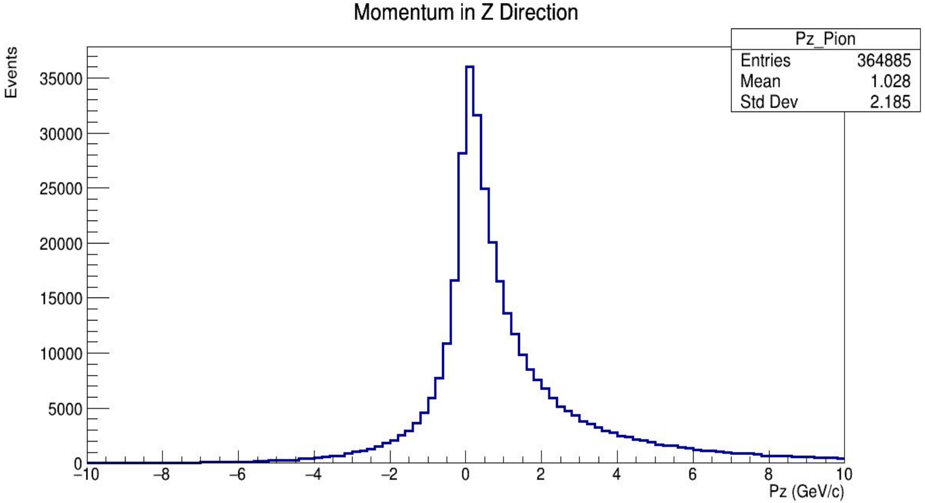
Mass and Energy of K^- :

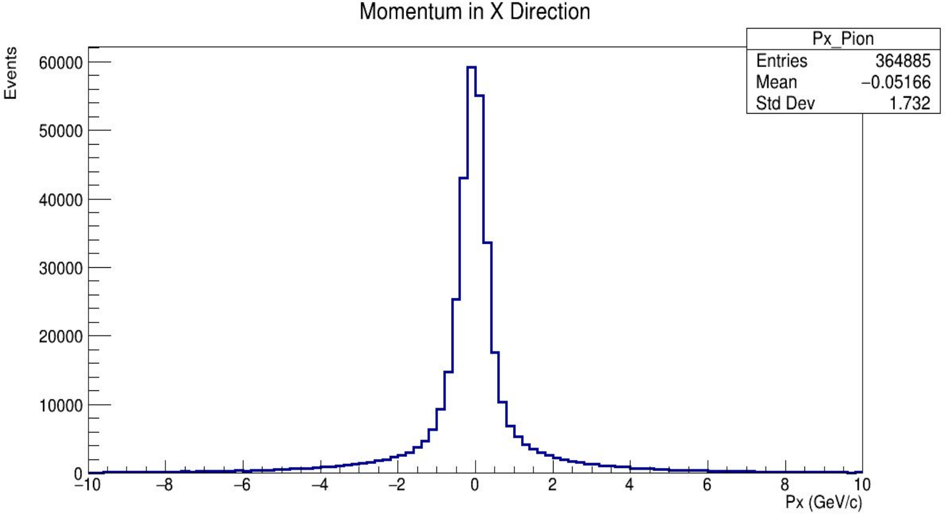


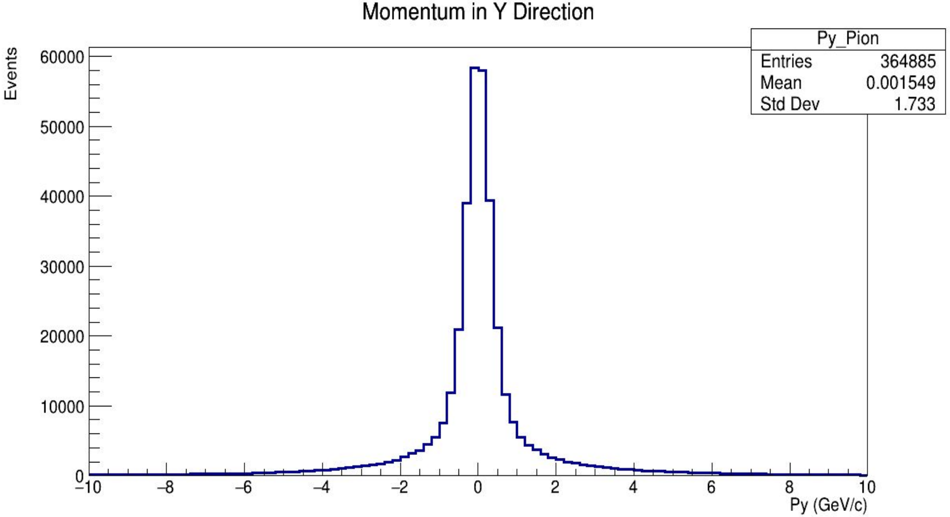
NOTE : Kaon has a mass of 493.677 ±0.016 MeV/c^2 which can be cross checked from the histogram mean value “0.4937 GeV/c^2”



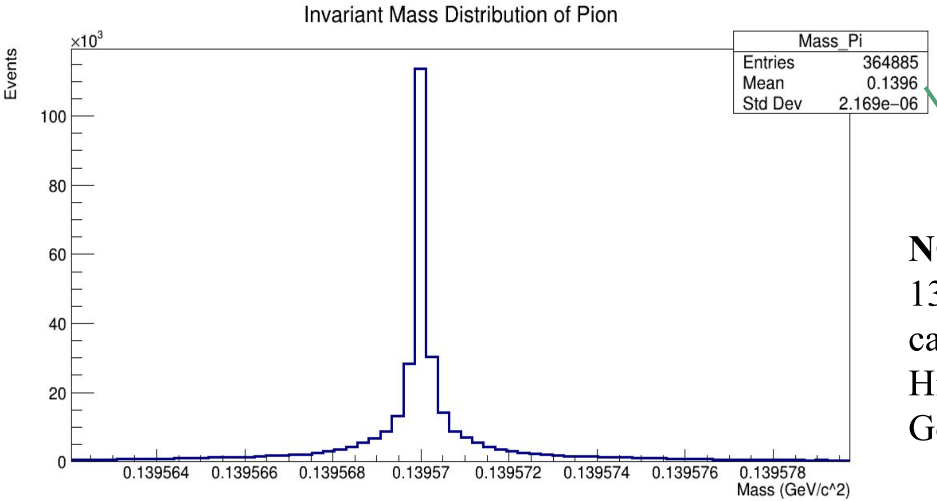
Momentum (p_x, p_y, p_z) of π^+ :



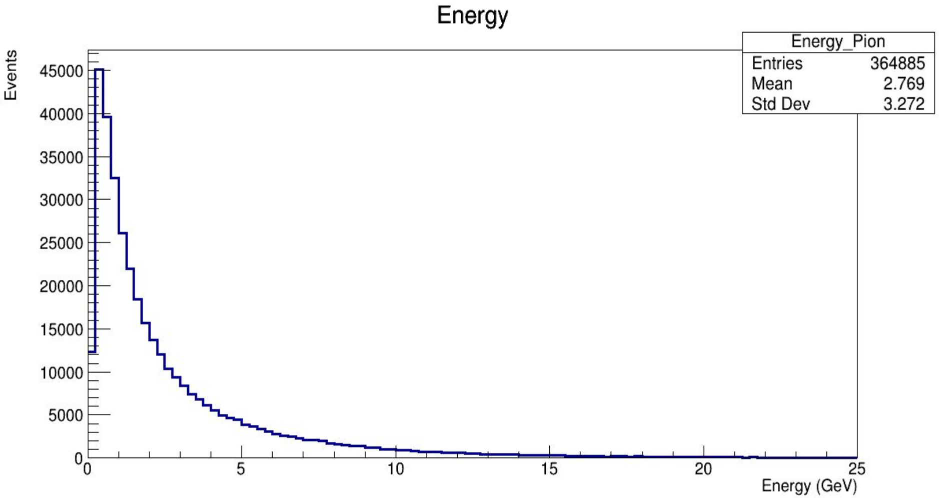




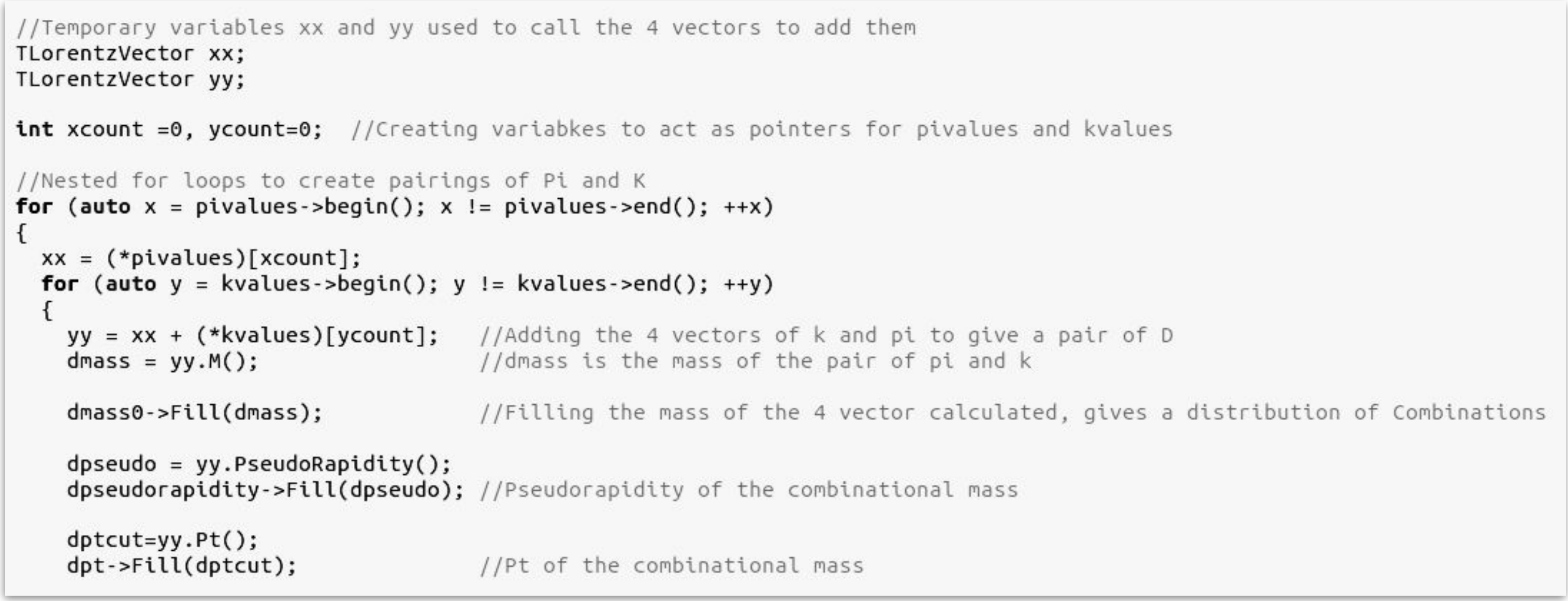
Mass and Energy of π^+



NOTE : Pion has a mass of 139.57039 ± 0.00018 MeV/c^2 Which can be cross checked from the Histograms mean value “0.1396 GeV/c^2”



Combinational Mass : A brute force approach

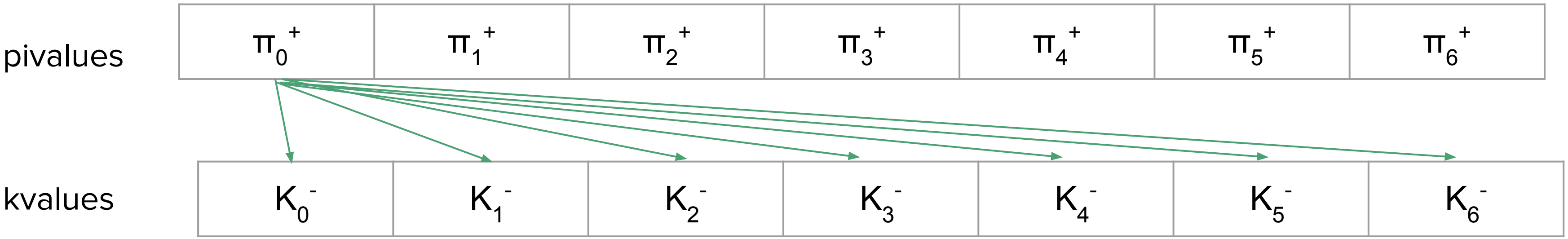


```
//Temporary variables xx and yy used to call the 4 vectors to add them
TLorentzVector xx;
TLorentzVector yy;

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( )*

Combinational Mass : A brute force approach

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

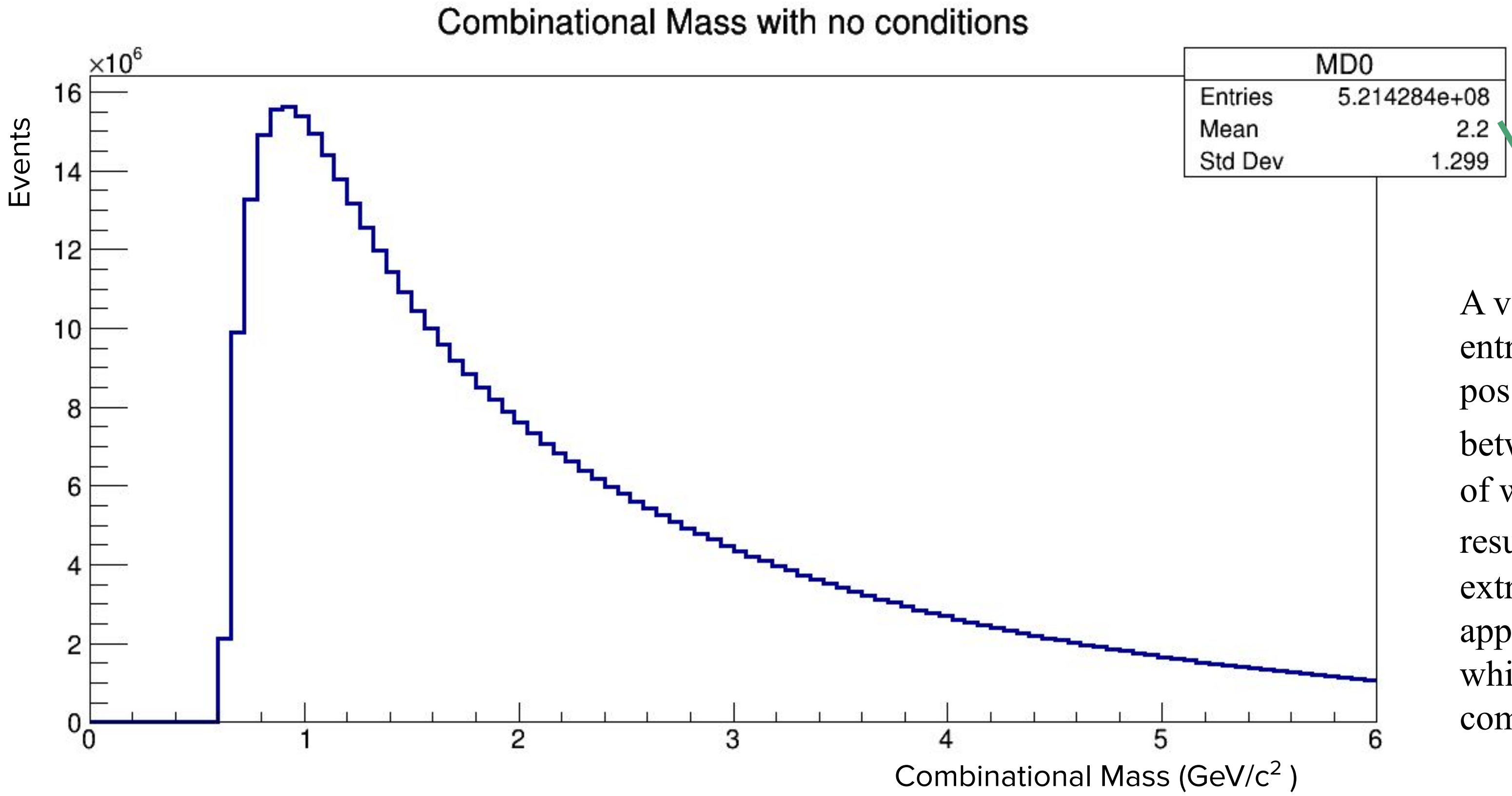


pivalues	π_0^+	π_1^+	π_2^+	π_3^+	π_4^+	π_5^+	π_6^+
kvalues	K_0^-	K_1^-	K_2^-	K_3^-	K_4^-	K_5^-	K_6^-

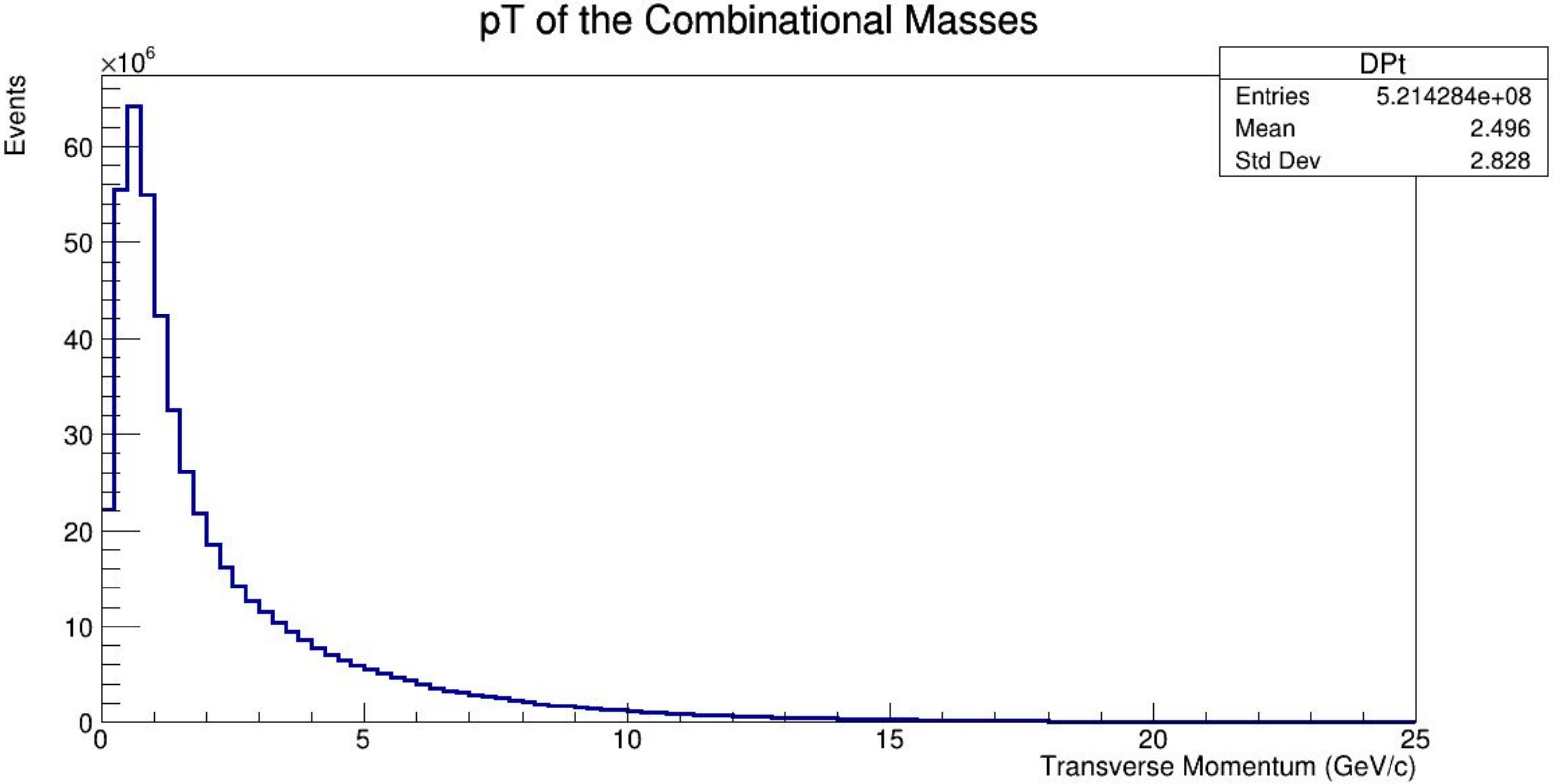
dmass contains all the possible combinations of π_i^+ and K_j^- where,

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

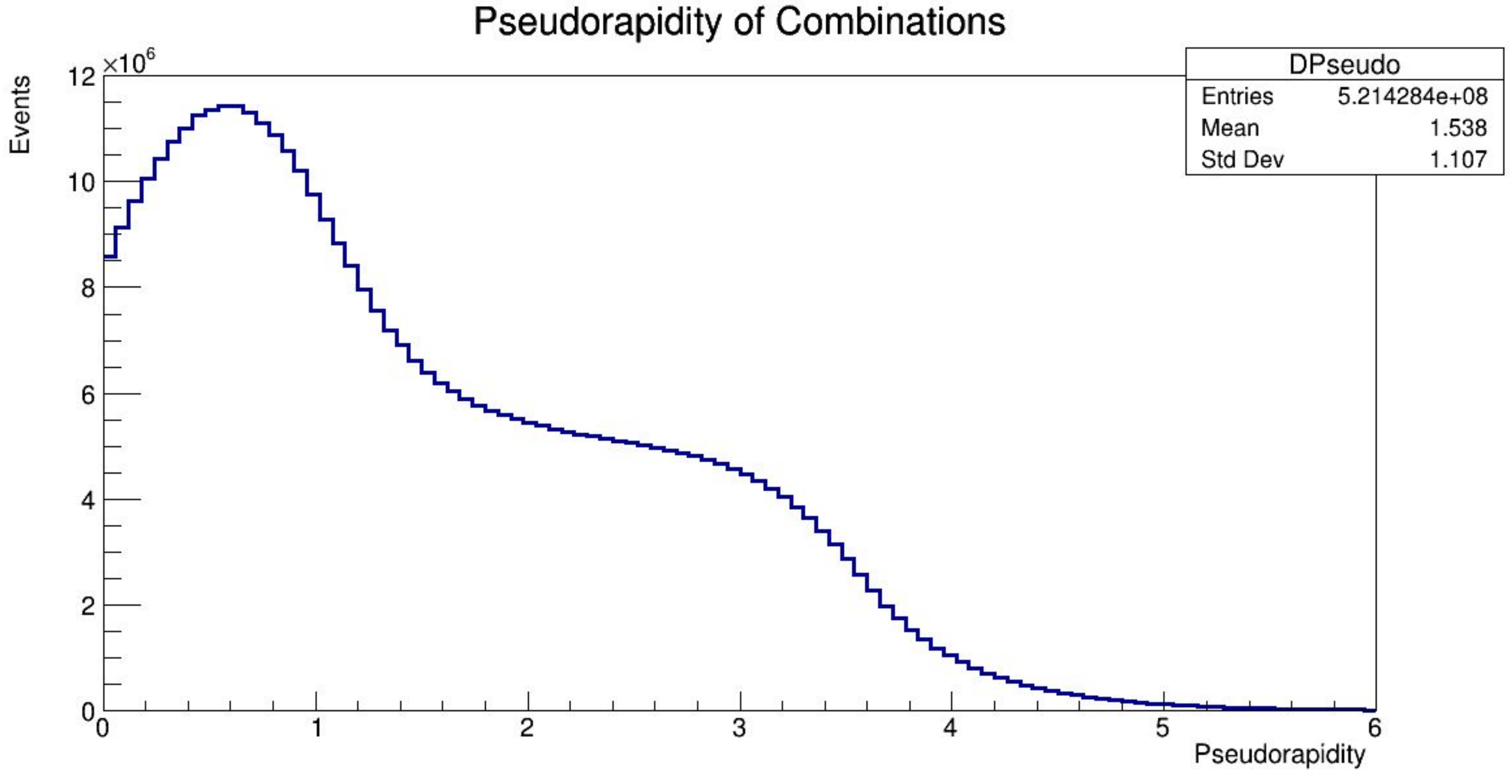
Similarly, all the Kaons are stored in the **‘kvalues’**



A very high number of entries due to all the possible combinations between **K**$^-$ and **π**$^+$, out of which only a few result in **D**$^{\mathbf{0}}$, we can extract those by putting appropriate conditions while taking the particle combinations.

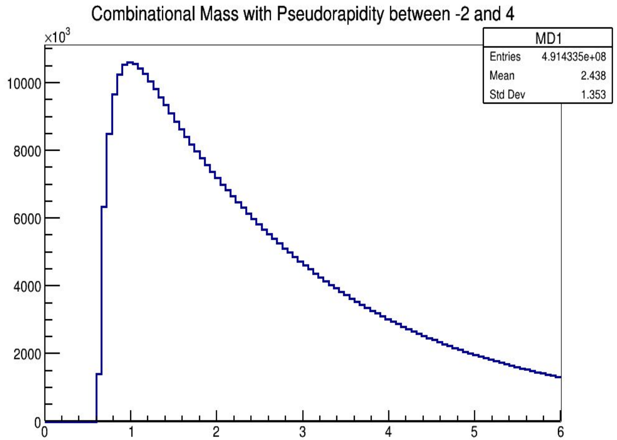


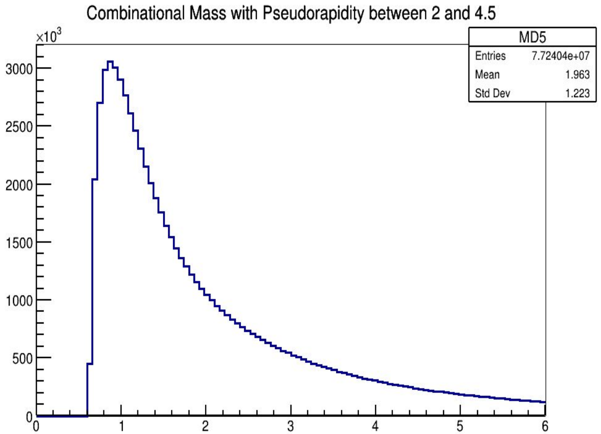
We can record the transverse momentum of these combinations.

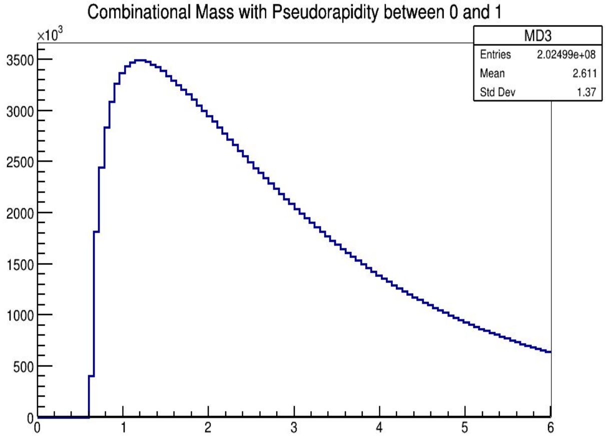


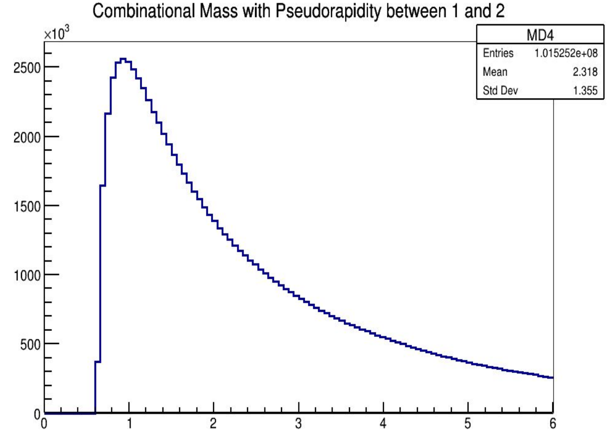
If we put a Pseudorapidity condition on the combinational mass, we can actually observe the changes in the output entries.

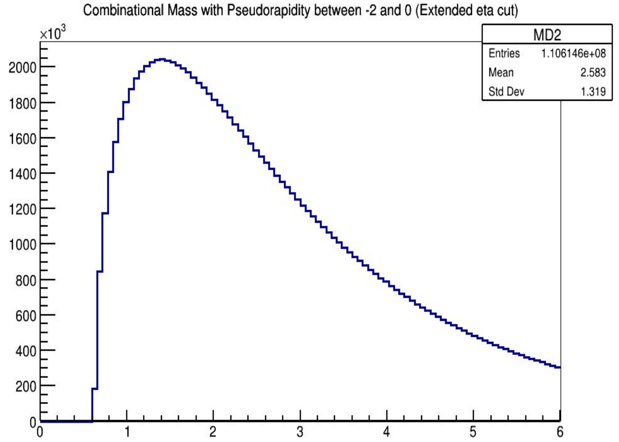
Combinational Mass with various eta cuts to observe the effect of Pseudorapidity

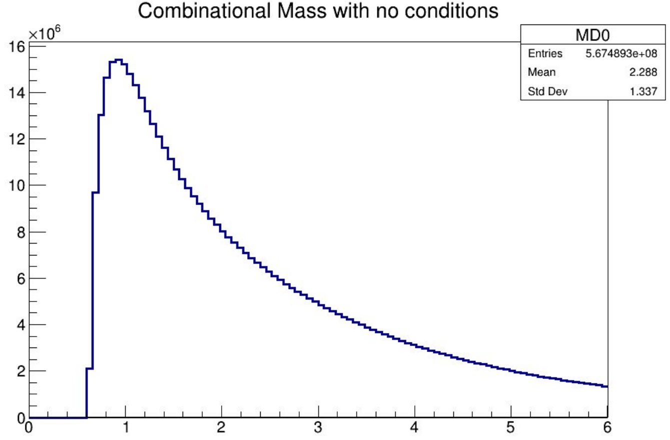


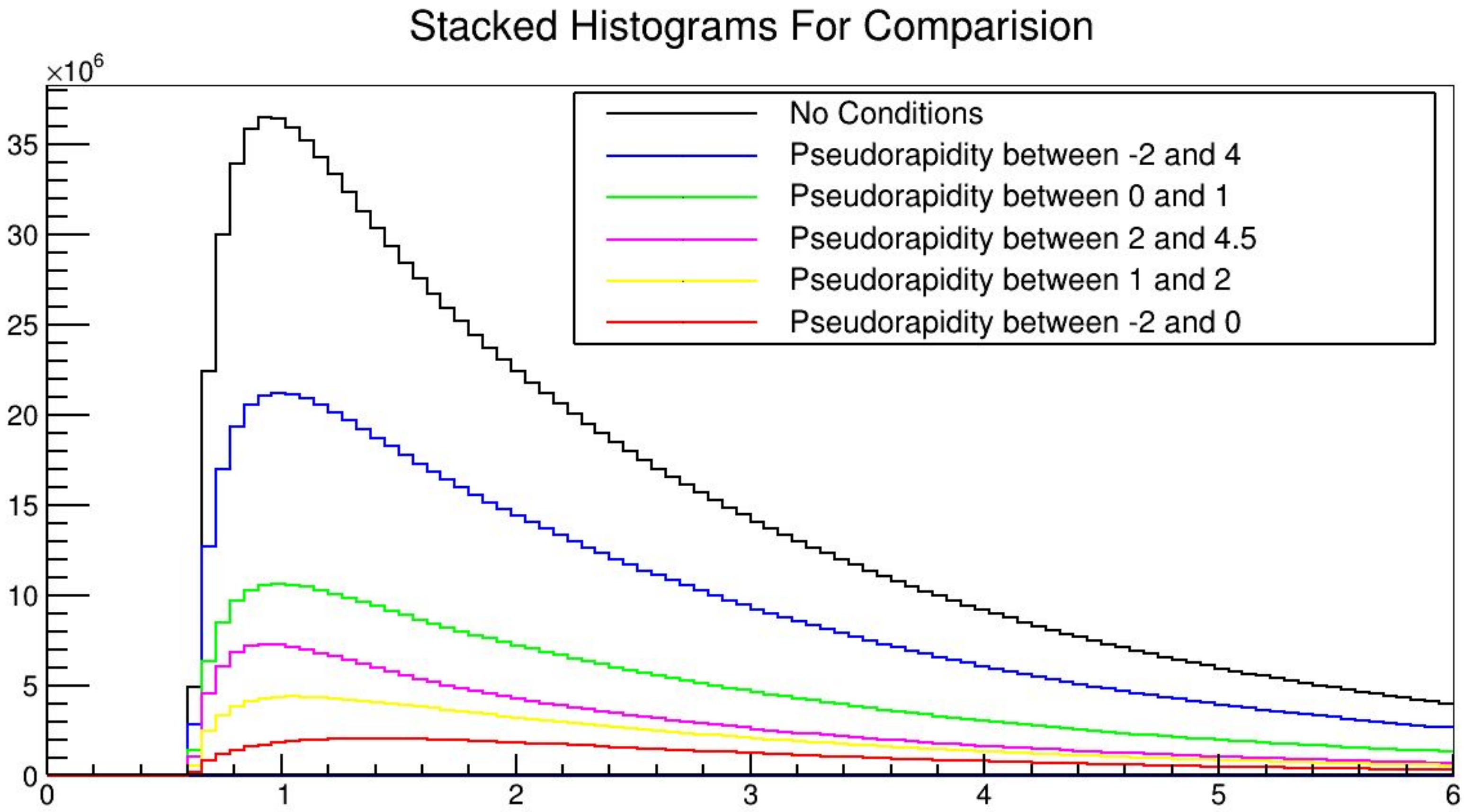








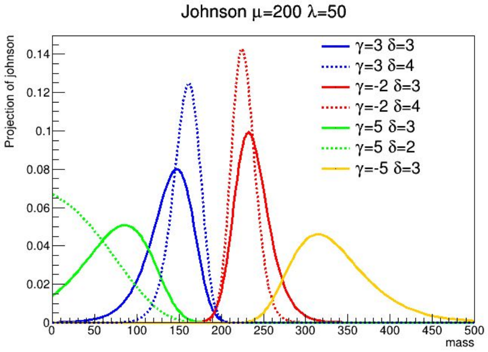


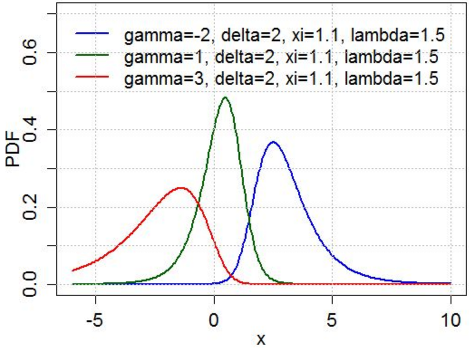

Stacked Histograms For Comparision
×10^6
35
30
25
20
15
10
5
0
0
1
2
3
4
5
6
No Conditions
Pseudorapidity between -2 and 4
Pseudorapidity between 0 and 1
Pseudorapidity between 2 and 4.5
Pseudorapidity between 1 and 2
Pseudorapidity between -2 and 0

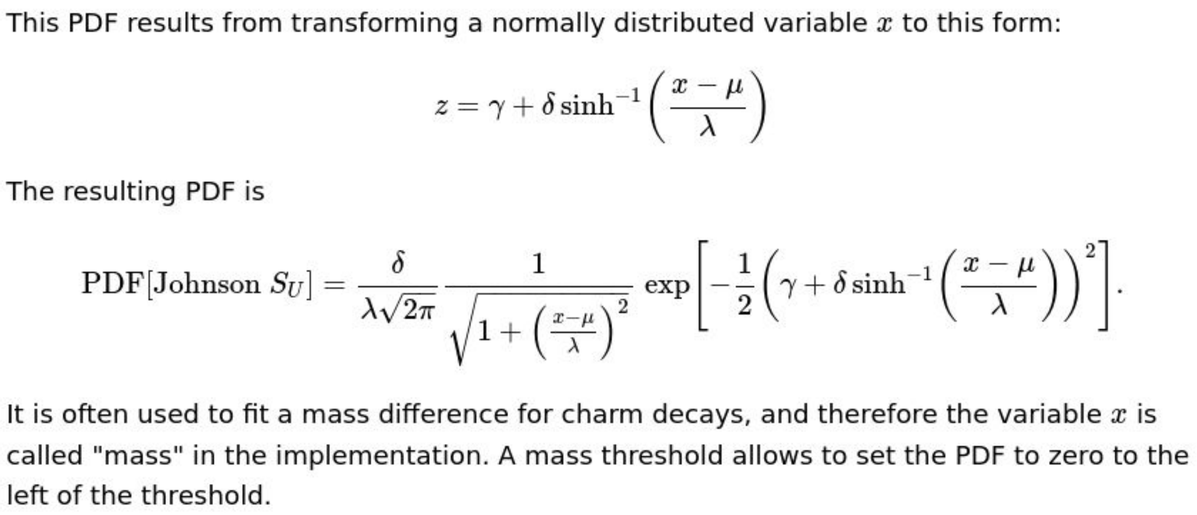
Johnson's Distribution

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

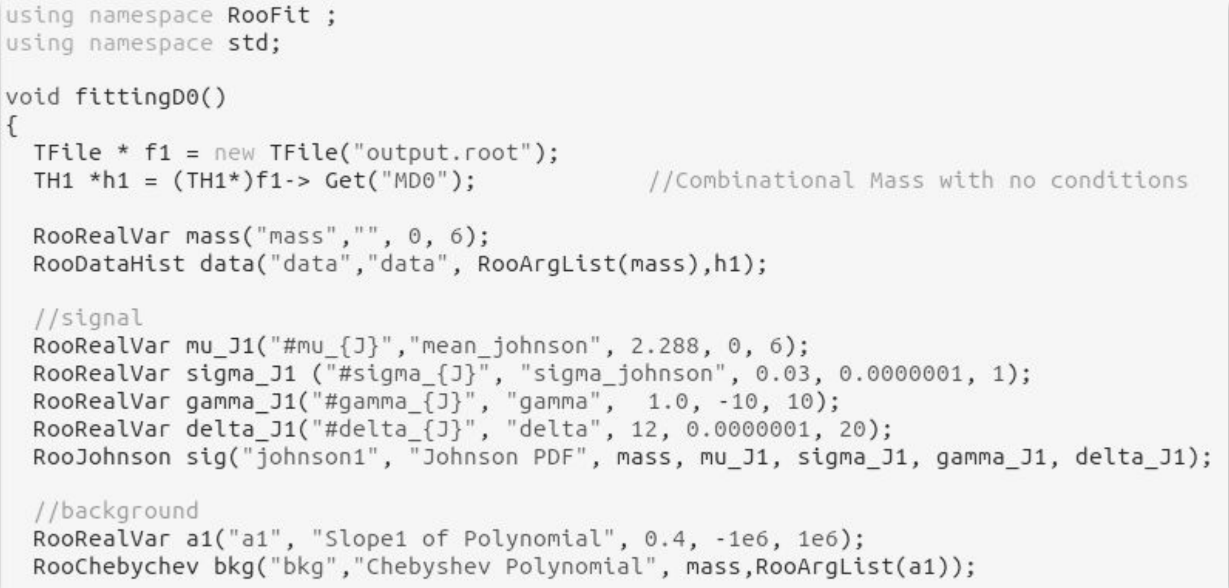
$$\mathrm{PDF}[\text{Johnson } S_U] = \frac{\delta}{\lambda\sqrt{2\pi}} \frac{1}{\sqrt{1+\left(\frac{x-\mu}{\lambda}\right)^2}} \exp\left[-\frac{1}{2}\left(\gamma + \delta \sinh^{-1}\left(\frac{x-\mu}{\lambda}\right)\right)^2\right].$$

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.

Fitting Histogram (Combinational Mass without any conditions) :

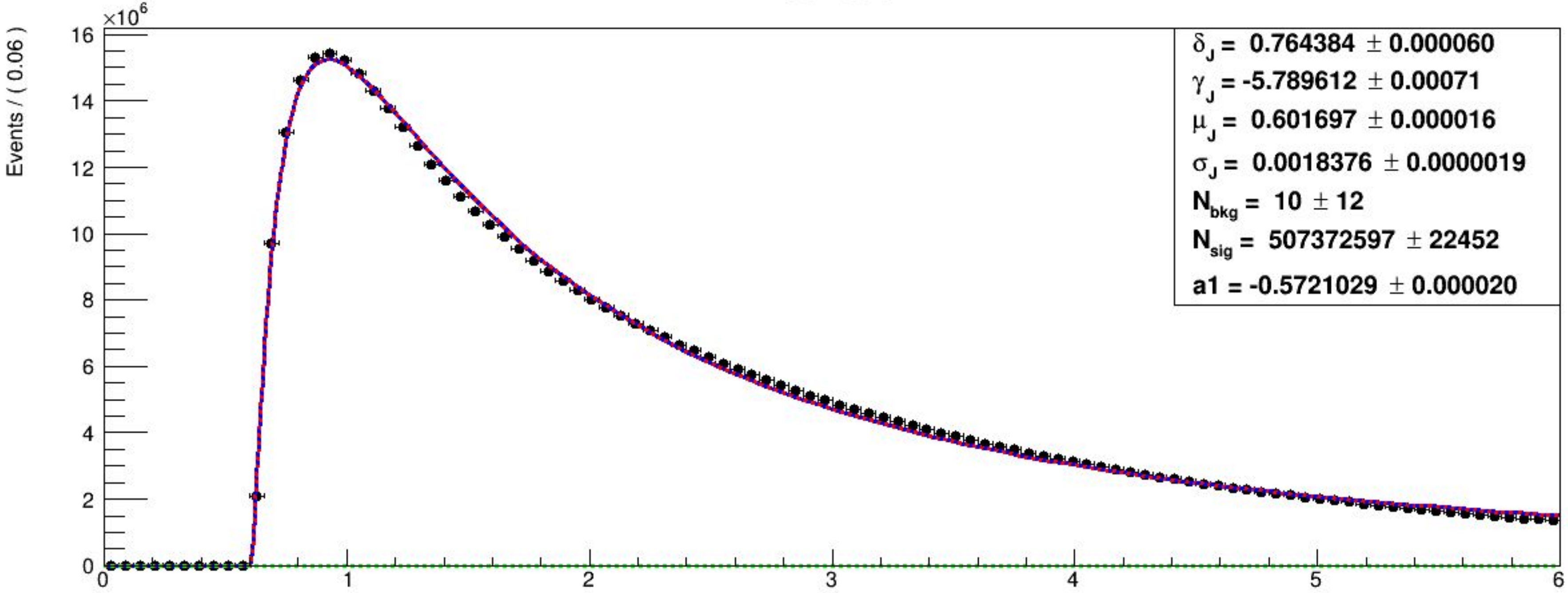
- While using RooFit we have made use of the **Johnson’s SU distributions** as the Signal

- 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));
```

Fitted Histogram



Reference from Yellow Report

Furthermore, we have recreated the Momentum v/s Pseudorapidity graphs for **K**$^-$ and **π**$^+$ particles (from the Reconstructed and the Generated particle Trees) and compared it to the ones in EIC yellow report.

Yellow Report Oct. 2021 Pg. 300

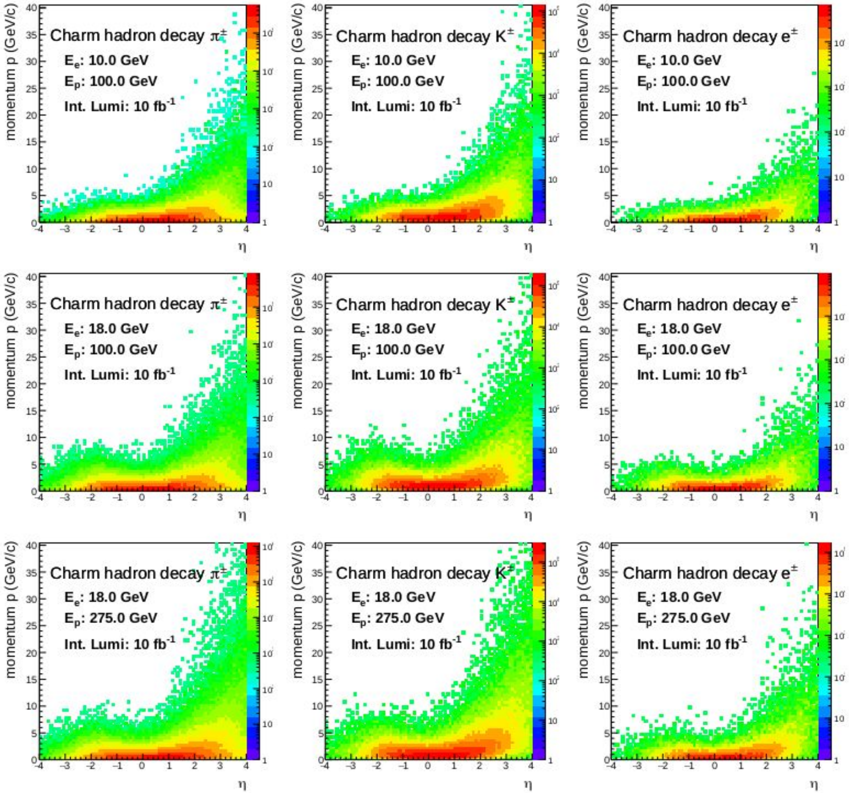
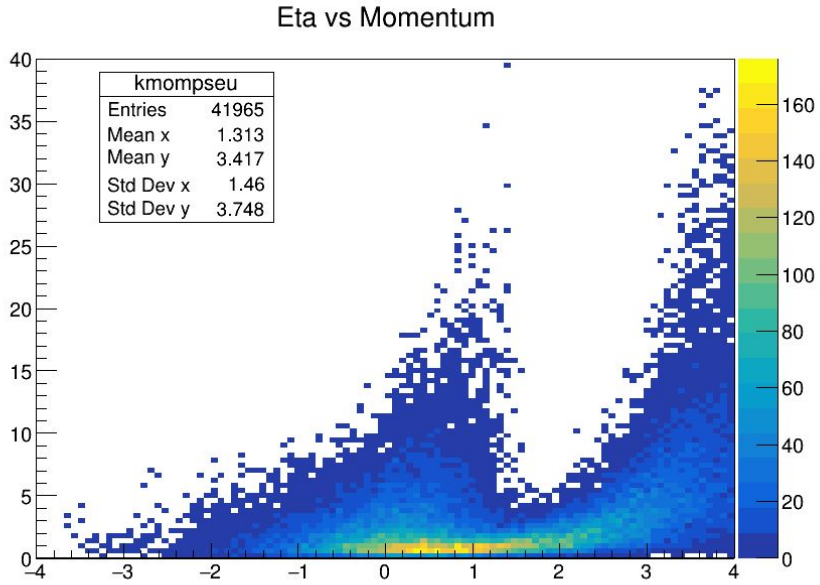
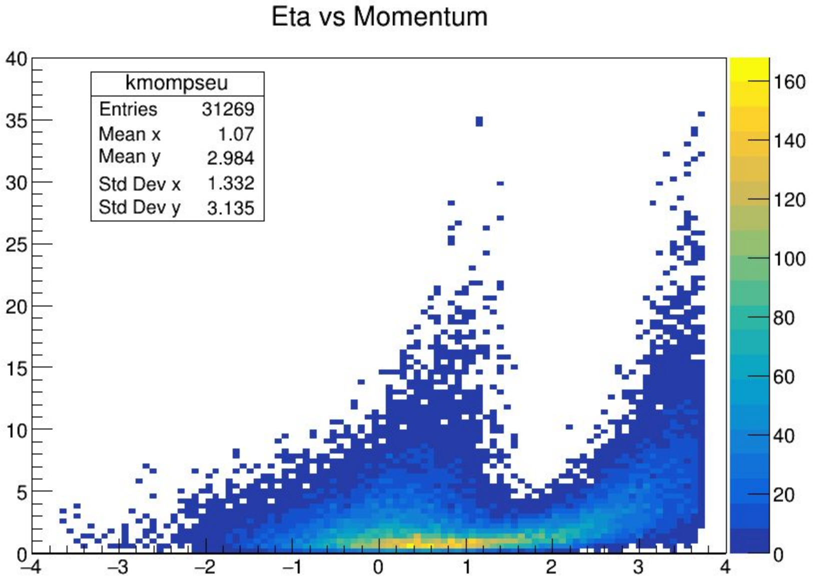


Figure 8.42: Momentum vs pseudorapidity for the decay products of D^0 mesons for beam energies of 10x100 GeV (top row), 18x100 GeV (middle row), and 18x275 GeV (bottom row). Charged pions are in the left column, charged kaons in the middle column, and electrons/positrons in the right column. Counts have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

Pseudorapidity(η) Vs Momentum(p) for K$^-$



K$^-$ for Generated Particles

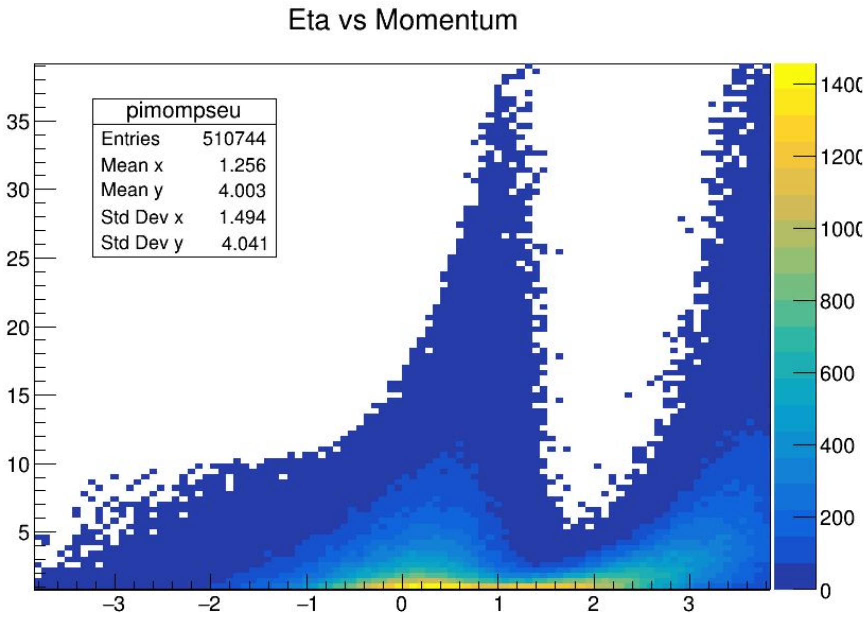


K$^-$ for Reconstructed Particles

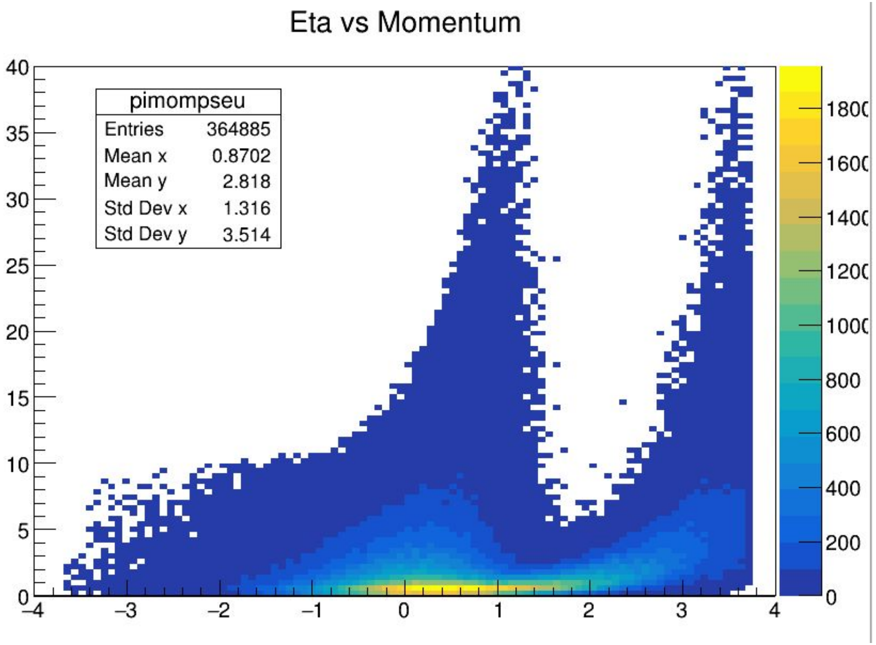
X-axis : Eta

Y- axis : Momentum

Pseudorapidity(η) Vs Momentum(p) for π^+



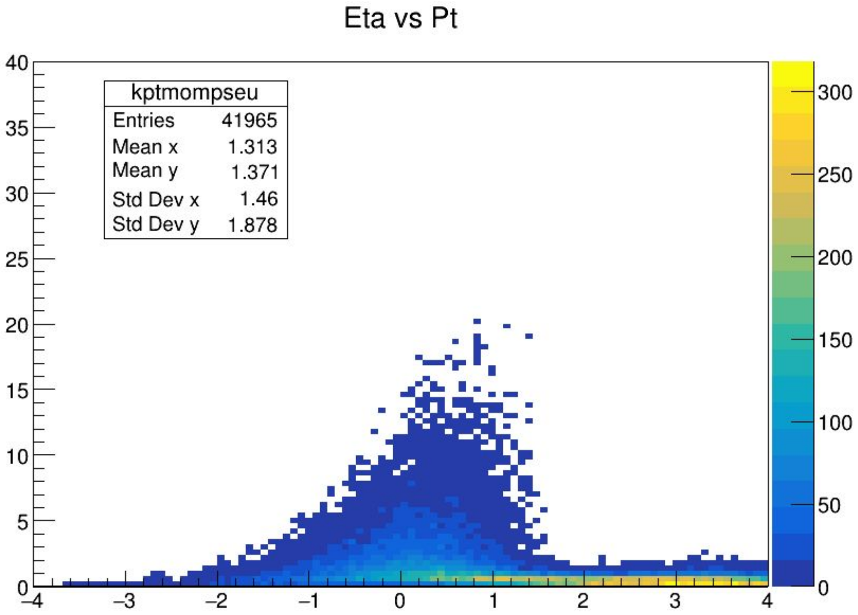
Pi+ for Generated Particles



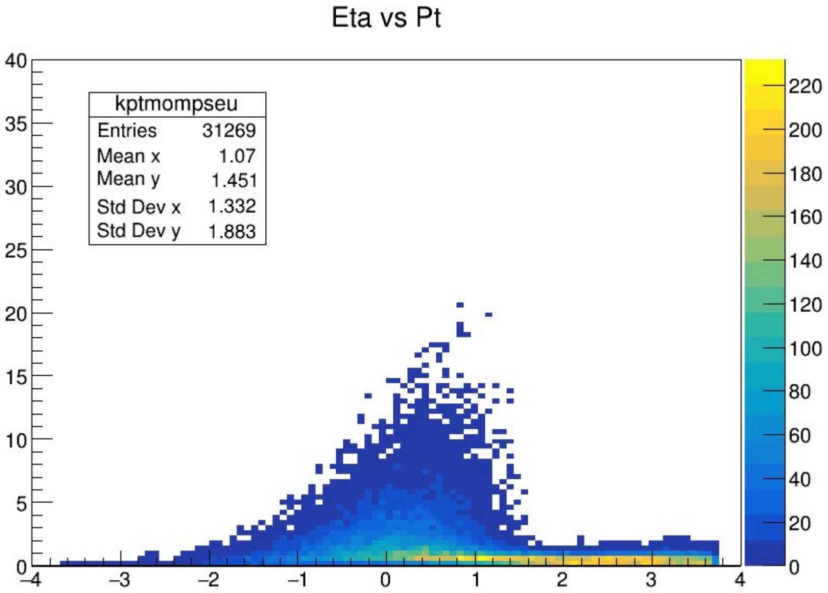
Pi+ for Reconstructed Particles

X-axis : Eta
Y- axis : Momentum

Pseudorapidity(η) Vs Transverse Momentum (p_T) for **K⁻**



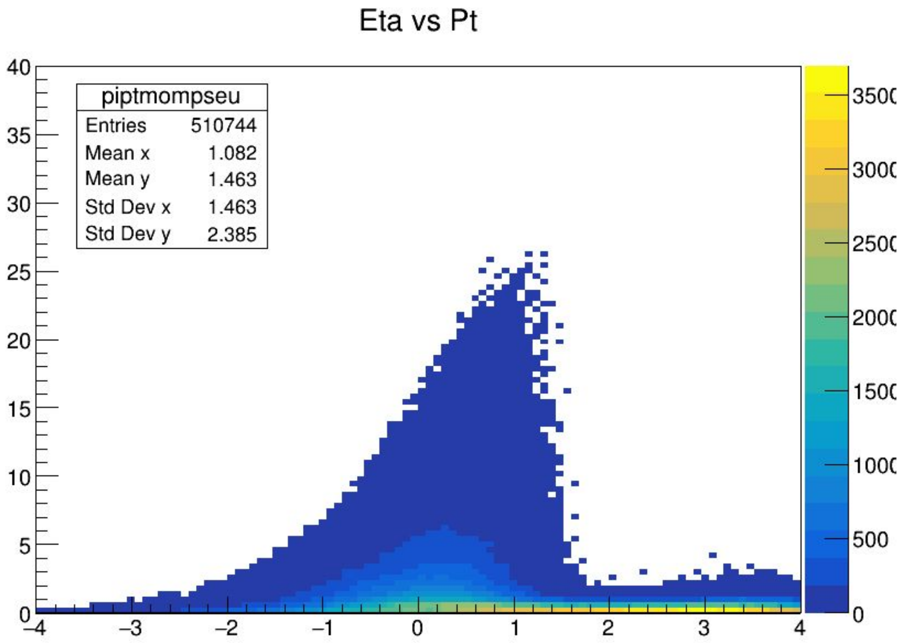
K- for Generated Particles



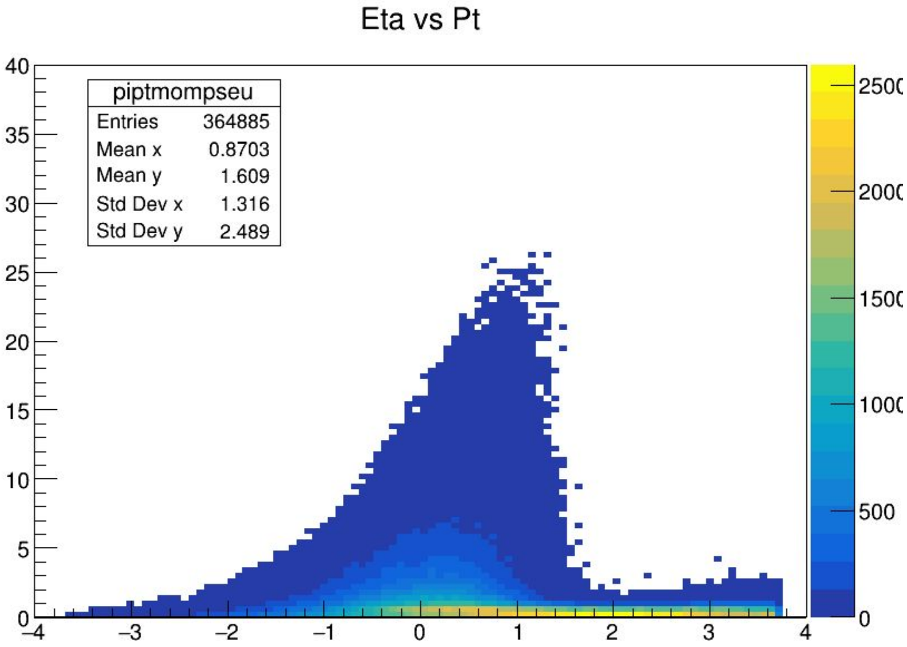
K- for Reconstructed Particles

X-axis : Eta
Y- axis : Transverse Momentum

Pseudorapidity(η) Vs Transverse Momentum (p_T) for π^+



Pi+ for Generated Particles



Pi+ for Reconstructed Particles

X-axis : Eta

Y- axis : Transverse Momentum

II. Production of Jets and Jet substructures

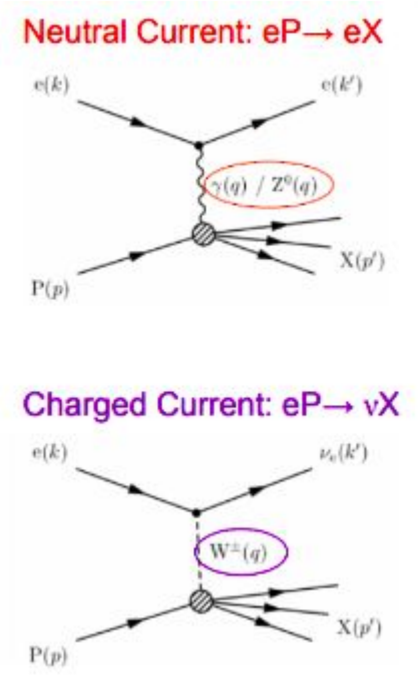
Student: **Siddharth Jain**

Supervisors: **Dr. Manjit Kaur (Panjab University, India),**
Dr. Ritu Aggarwal (Savitribai Phule Pune University)

Jet and subjet production in DIS ep

- **PYTHIA 8** and **RAPGAP** packages are used for event generation.
- **FastJet** package used for production of jets and subjets.

- **Beam energies**:
  20 GeV for electron and 250 GeV for proton.
- **Virtuality** $Q^2 > 125\ \text{GeV}^2$
- **Jet algorithm** used: Longitudinally invariant k_T cluster algorithm for both jets and subjets.
- **Jet radius**(R) = 0.8
- **Transverse energy** $E_T > 10$ GeV for jets



Feynman diagrams of Neutral Current(NC) and Charged Current(CC) DIS ep

Longitudinally invariant k_t jet algorithm

1. For each pair of particles i, j work out the distance $\mathbf{d_{ij}}$**:**

$$d_{ij} = \min(1/p_{ti}^2, 1/p_{tj}^2)\,\Delta R_{ij}^2/R^2\,,$$
$$d_{iB} = 1/p_{ti}^2\,.$$

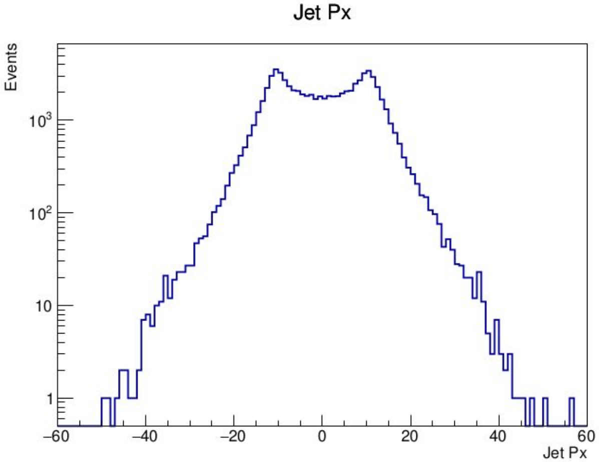
with $\Delta R_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2$, where p_{ti}, y_i and φ_i are the transverse momentum (with respect to the beam direction), rapidity and azimuth of particle i. R is a jet-radius parameter.

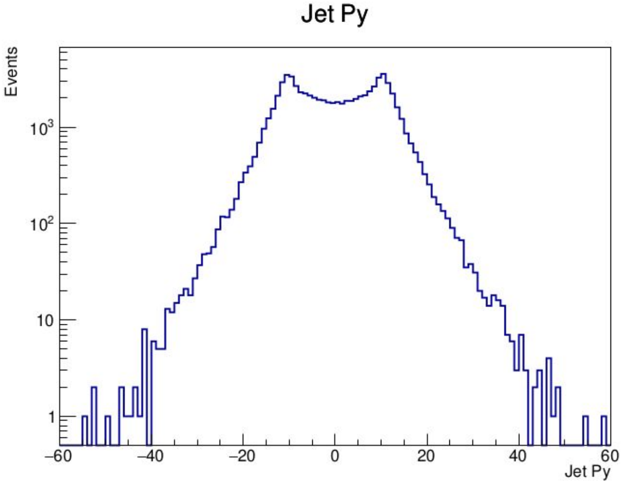
2. For each parton, work out the beam distance $\mathbf{d_{iB} = p^2_{ti}}$**.**

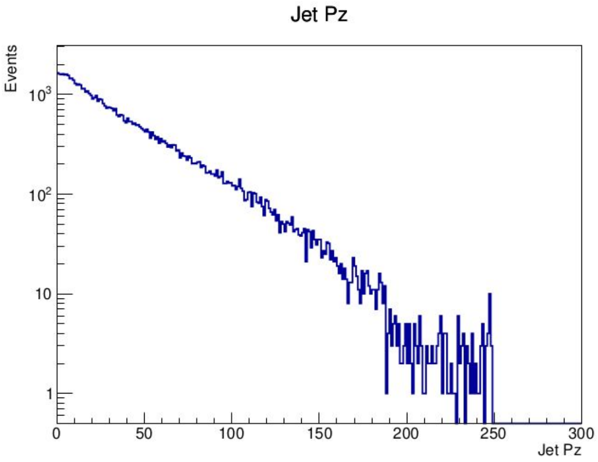
3. Find the minimum d_{min} of all the d_{ij} , d_{iB}. If d_{min} is a d_{ij} merge particles i and j into a single particle, summing their four-momenta; if it is a d_{iB} then declare particle i to be a final jet and remove it from the list.

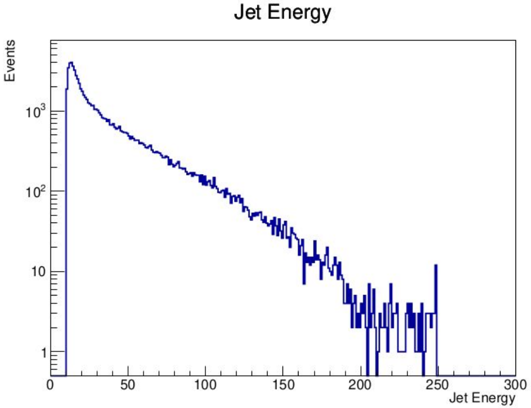
4. Repeat from step 1 until no particles are left.

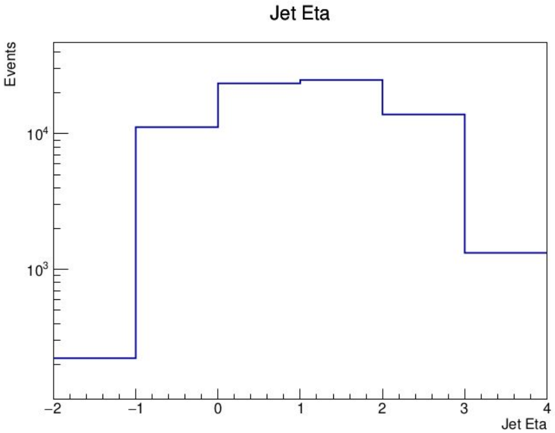
Jet properties

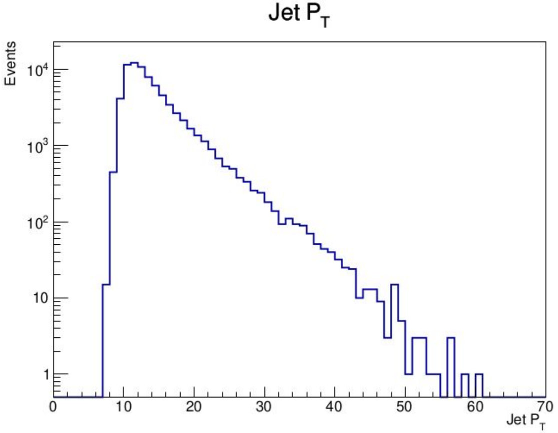




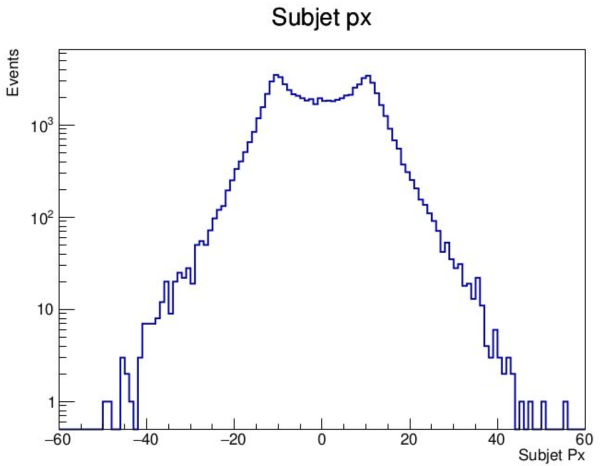


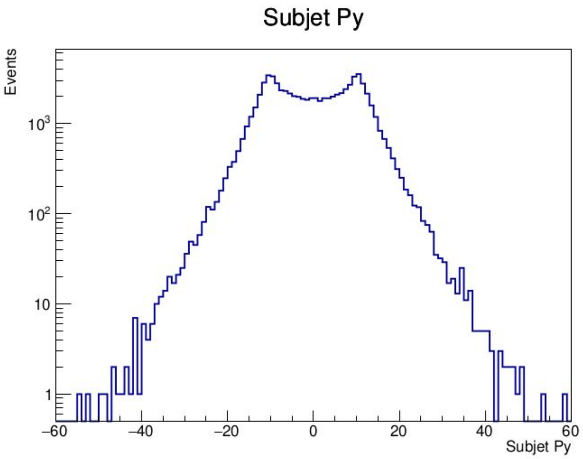


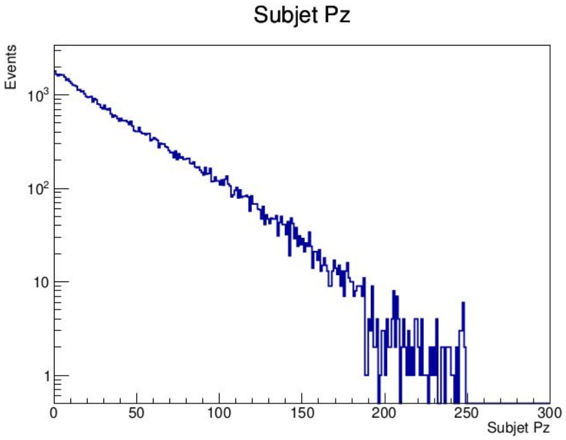


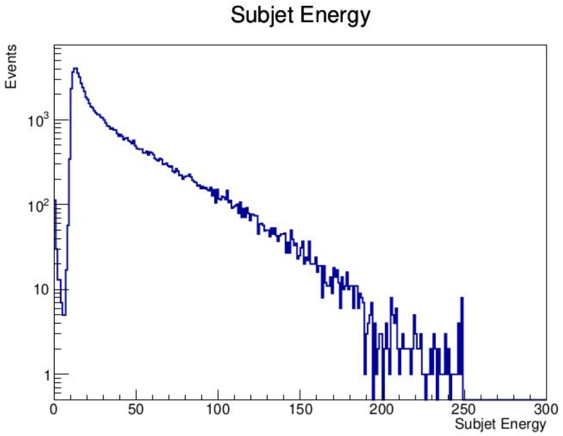


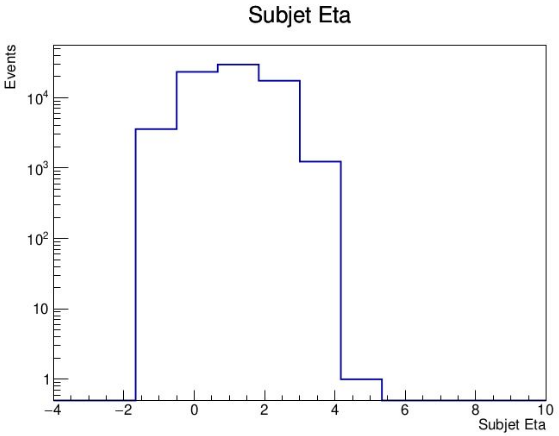
Subjet properties

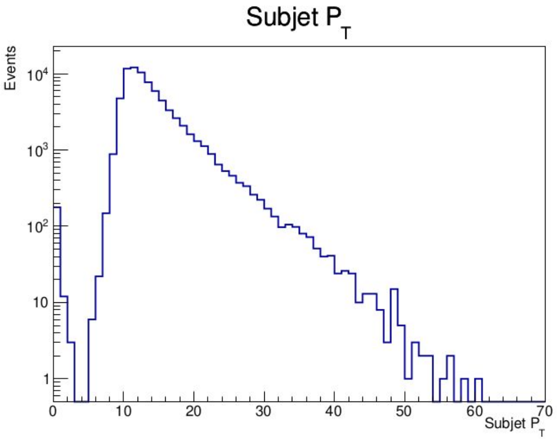












Thank you!

Request for Research Opportunity

We are very interested in EIC research,
and actively looking for research opportunities related to HF and Jet physics.

Siddharth Singh :
Email : 2020pph5107@mnit.ac.in (or, siddx25@gmail.com)
Github : https://github.com/swayze25
Skills: PYTHIA, ROOT, RAPGAP, FASTJet, HEPMC, GEANT4, Delphes simulator.

Mihir Patel :
Email : i18ph037@phy.svnit.ac.in or tobephysicistmihirpatel@gmail.com
Skills: PYTHIA MC event Generator, ROOT framework, GEANT4, MadGraph MC NLO, Rivet toolkit, Delphes simulator.

Siddharth Jain :
Email : siddharth2006163@st.jmi.ac.in and siddharthjain513@gmail.com
Skills: PYTHIA AND RAPGAP event generators, FastJet package, ROOT framework, RIVET toolkit, HEPMC, Delphes simulator.