JOINT INSTITUTE FOR NUCLEAR RESEARCH

Veksler and Baldin laboratory of High Energy Physics

Final Report of INTEREST program

MPD detector performance study at the NICA collider

Supervisor

Dr. Ivonne Maldonado

Dr. Vadim Kolesnikov

Student

San Juan López Alejandro

Universidad Autónoma Metropolitana

Wave 10. February 26th - April 14th, Dubna 2023

Contents

Abstract

In this project, measures and histograms of the distribution of energy loss by ionization of 6 particles are presented using the MpdRoot software, specifically: helium-3, deuterium, kaon, proton, electron and pion. These distributions were obtained for both primary and secondary particles. In addition, adjustments were made to these distributions using proposed functions, which depended on parameters obtained from the analysis of the distributions of each particle. This was done in order to identify the region in which one or another species of particle is found.

1 Introduction

1.1 Multi Purpose Detector (MPD)

The Multi-Purpose Detector (MPD) is a spectrometer located in the Nuclotron-based Ion Collider fAcility (NICA) at the Joint Institute for Nuclear Research (JINR). It works in the energy range of heavy ion collisions of 4 GeV $\leq \sqrt{s_{NN}} \leq$ 11 GeV, exploring the high density region of baryons. This experimental program will fill a gap in the energy scale not yet fully explored and provide a deeper understanding of hadron dynamics and the production of multiple particles in the high-density baryon domain. In addition, the MPD will provide accurate 3D tracking and a high-performance particle identification system based on a high-volume gaseous temporal projection chamber (TPC), flight time measurements (TOF) and calorimetry. Its main objectives include: to study the equation of state for high barionic densities, to search for Critical End Points (CEP), understand exotic hadron formation and explore the influence of light nuclear structure on heavy ion physics. Its particle analysis process involves a detailed event-by-event study, providing information on trajectories, particle identification and collision centrality.

The installation of the MPD is anticipated to occur in two phases. The initial phase of the detector setup is scheduled for completion in 2025, with plans for commissioning. The figure 1 illustrates the overall arrangement of the MPD and the spatial organization of its detector subsystems during this initial stage.

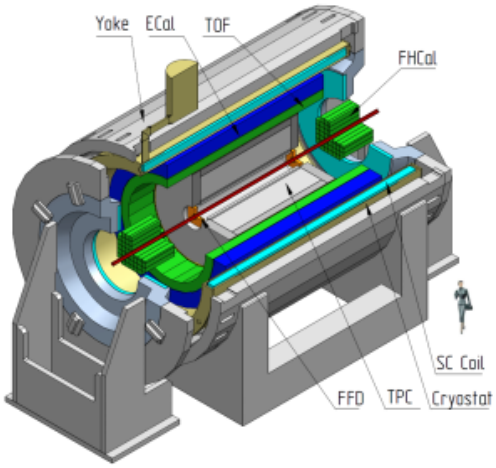

Figure 1: *The overall schematic of the MPD subsystems in the first stage of operation (Stage 1) – three-dimensional view.*

The "central barrel" components have an approximate cylindrical symmetry within $|\eta| <$ 1.5. The beam line is surrounded by the large volume Time Projection Chamber (TPC) which is enclosed by the TOF barrel. The TPC is the main tracker, and in conjunction with the TOF they will provide precise momentum measurements and particle identification. The Electromagnetic Calorimeter (ECal) is placed in between the TOF and the MPD magnet. It will be used for detection of electromagnetic showers, and will play the central role in photon and electron measurements. The MPD superconducting solenoid magnet is designed to provide a highly homogeneous magnetic field of up to 0.57 T (with a default operational setting of 0.5 T), uniform along the beam direction, to ensure appropriate transverse momentum

resolution for reconstructed particles within the range of momenta of 0.1-3 GeV/c. As the average transverse momentum of the particles produced in a collision at NICA energies is below 500 MeV/c, the detector was designed to have a very low material budget. In the forward direction, the Fast Forward Detector (FFD) is located still within the TPC barrel. It will play the role of a wake-up trigger. The Forward Hadronic Calorimeter (FHCal) is located near the magnet end-caps. It will serve for determination of the collision centrality and the orientation of the reaction plane for collective flow studies.

For the second stage, some of the proposed systems and detectors [2], currently under development to be installed after the commissioning of the Stage 1 subdetectors, are as follows:

- The inner tracking system
- The miniBeBe detector
- The cosmic ray detector

1.2 The MpdRoot

MpdRoot is a software framework developed for the MPD (Multi Purpose Detector) experiment that aims to facilitate the simulation, reconstruction and analysis of physical data obtained during the experiment. Assists the MPD experiment by providing a comprehensive platform for software development, including detailed simulations of heavy ion collisions in the NICA energy range, as well as the reconstruction of particle trajectories (see figure 2) and the analysis of experimental data.

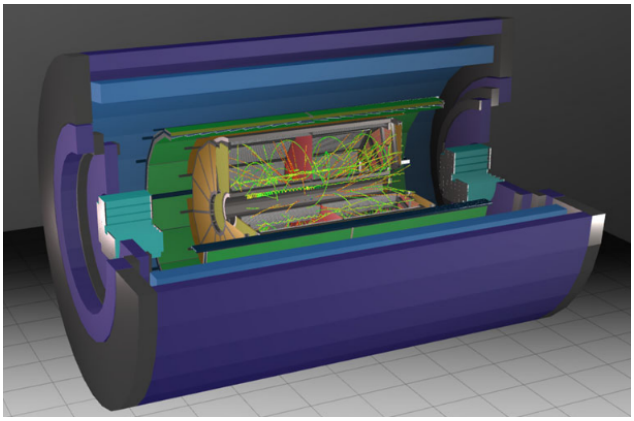

Figure 2: *Simulation of particle trajectory reconstruction.*

MpdRoot's operation is based on providing an interface for various Monte Carlo event generators, such as GEANT4, that model collisions and particle transport through detectors. In addition, another of its uses is to carry out research and simulation of the technical design of all subdetectors in order to optimize their designs.

Information about the software, including its installation and requirements, can be found on the site of the software [12], while this document will explain the use of MpdRoot to carry out the requested activities.

2 Project goals

The main objective of the practice is to study the performance of the MPD.

1. Study of MPD phase space coverage for hadrons and light cores in fixed-target mode, with different beam-objective combinations.
2. Study of the ability to identify light hadrons using ionization loss measurements (dE/dx).
3. Estimation of the secondary contribution of particles in the yields of (anti)protons and deuterons.
4. Assessment of MPD's tracking capabilities in A+A collisions based on particle multiplicity.

While the objectives of the project are those mentioned above, they will depend on subtasks. It is for this reason that each member of the project was assigned a specific objective. In my case, I was assigned task number 2, which will be developed later.

3 Study of the ability to identify light hadrons using ionization loss measurements (dE/dx)

Loss of energy by ionization is the process by which a charged particle loses energy as it passes through a medium, such as a detector. This loss of energy is crucial for particle identification in experimental particle physics. By measuring the loss of ionizing energy (dE/dx), along with the moment of the particle, you can identify the type of particle and determine its kinetic energy.

The theoretical expression that predicts loss of ionizing energy (the mean rate of energy loss) is the *Bethe equation* (1), which describes how the loss of energy increases proportionally with β^{-2} as the particle velocity decreases. This equation also takes into account the density of the medium through which the particle is moving.

$$\langle -\frac{dE}{dx} \rangle = Kz^2 \frac{Z}{A} \frac{1}{\beta^2} \left[\frac{1}{2} ln \frac{2m_e c^2 \beta^2 \gamma^2 W_{max}}{I^2} - \beta^2 - \frac{\delta(\beta\gamma)}{2} \right]. \tag{1}$$

The energy loss is not the same for each particle, as it depends on the charge, mass and velocity of the particle, as well as the material through which it is moving. Slower particles are more ionizing than faster particles, and energy loss reaches a minimum for a specific velocity range, known as the minimum ionization particle regime (see figure 3).

For MPD, the loss of ionization energy will change according to the detector material through which the particles travel. For example, the TPC detector will measure the ionization of charged particles in a gas, while the TOF detector will measure the flight time of particles through a solid material. Both detectors will provide valuable information on particle energy loss, and it is for this reason that they are used in combination to identify and characterize charged particles in MPD experiments.

Symb.	Definition	Value or (usual) units
E	energy	MeV
x	mass per unit area	g cm^2
K	coefficient for dE/dx	0.307075 MeV mol^{-1} cm^2
z	charge number of incident particle	
Z	atomic number of absorber	
A	atomic mass of absorber	g mol^{-1}
$m_e c^2$	electron mass x c^2	0.5109989461(31) MeV
I	mean excitation energy	eV
$\delta(\beta\gamma)$	density effect correction to ionization energy loss	
W_{max}	Maximum possible energy transfer to an electron in a single collision	MeV

Table 1: *Variables of Bethe equation. The kinematic variables β and γ have their usual relativistic meanings.*

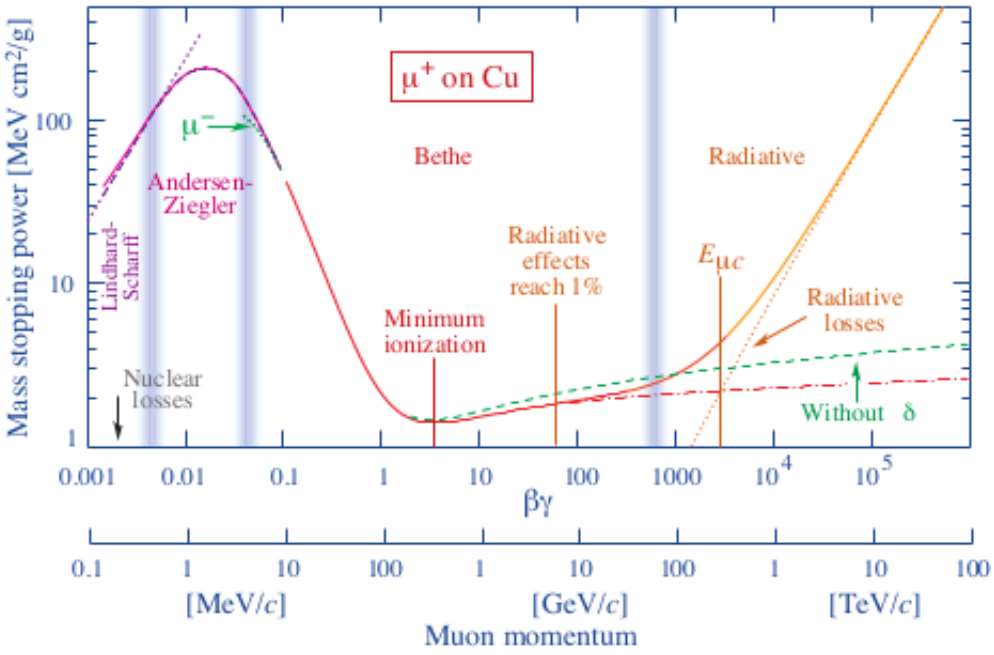

Figure 3: *Energy loss of muons in copper. The y-axis is given in units of MeV cm^2 /g so that one can multiply by the density of any material and compute the energy loss in MeV/cm. The energy loss depends on the relativistic quantity $\beta\gamma$ which can be converted to momentum.*

As you can see in the figure 3, we will have different regions marked by vertical lines. This implies that the energy loss will have a different behavior at low and high energies. This behavior in the dE/dx function will be reflected as corrections to the Bethe equation; that is, terms will be added or removed to adjust the function. Some of these corrections are: shell correction, Barkas correction and Bloch correction. In addition, there will be other proposed functions that can describe the behavior of energy loss in these regions. One of them is the Landau-Vavilov-Bichsel function.

3.1 Simulate A+A collisions (model and/or box generators)

To begin collision analysis using MpdRoot software, we will need to generate the information to be analyzed. For this, we will use the event generator "BOX". This generator will allow us to simulate basic nuclear collisions. Because of its simplicity, the generator does not offer a way to specify the types of atomic nuclei that will interact. It is usually used to produce a statistical sample of generic nuclear collisions, which are then processed through the entire chain of simulations and data analysis to study the response of detectors and evaluate event reconstruction algorithms. However, it will allow us to select some features of the collision, as shown in the table 2:

Symb.	Definition	Value or (usual) units
$\|P\|$	moment range	(0.0,5.0) GeV/c
ϕ	azimuth angle range	(0.0,360) °
θ	Polar angle range	(0.0,180) °
XYZ	position	(0,0,0) cm

Table 2: *Parameter set (BOX) for simulation of events used*

In addition to these parameters set for the generator BOX, other parameters were defined, but outside the generator and which were:

Symb.	Definition	Value or (usual) units
vmc	virtual monte carlo generator	GEANT3
nEvents	number of events	1000
partPdgC	particles created	π,e,P,K,D,He^3
multipl	multiplicity	100

Table 3: *Parameters set for simulation of events used*

The above values were introduced into the macros *runMC.C* and *runReco.C* (see figure 4), both files obtained from the NICA repository [10].

```
for (Int_t i = 0; i < 6; i++)
   {
   switch (generator)
   {
   case EGenerators::BOX: // Box generator
   {
   //for (Int_t i = 0; i < 6; i++)
   //{
      gRandom->SetSeed(0);
      FairBoxGenerator *boxGen = new FairBoxGenerator(partPdgC[i], 100);
      //FairBoxGenerator *boxGen = new FairBoxGenerator(partPdgC[4], 100);
      //FairBoxGenerator *boxGen = new FairBoxGenerator(13, 100); // 13 = muon;
      boxGen->SetPRange(0.0, 5.0); // GeV/c, setPRange vs setPtRange
      boxGen->SetPhiRange(0, 360); // Azimuth angle range [degree]
      boxGen->SetThetaRange(0, 180);// Polar angle in lab system range [degree]
      boxGen->SetXYZ(0., 0., 0.); // mm o cm ??
      primGen->AddGenerator(boxGen);
      break;
     //}
   }
```

Figure 4: *BOX generator parameters in the macro runMC.C*

```
// Available generators
enum EGenerators
{
   BOX = 1,
   FLUID,
   HSD,
   ION,
   LAQGSM,
   MCDST,
   PART,
   SMASH,
   UNIGEN,
   URQMD,
   VHLLE
};

// Available VMC types
enum EVMCType
{
   GEANT3 = 1,
   GEANT4
};

Int_t partPdgC[] = {211, 11, 2212,321,1000010020,1000020030};//pi, e, p, K, d, He3
```

Figure 5: *Parameters in the macro runMC.C*

After modifying the macros, we proceeded to run them using the MpdRoot software with the following codes *root -b -q runMC.C* and *root -b -q runReco.C*. This generated an mpddst.root file, which contains the information of the events to be analyzed.

In order to use this information, a file called "*listabox.txt*" had to be created, in which the path of the file mpddst.root. This was done in order to be used in the macro RunAnalyses. C (see figure 6), which will be the one containing the wagon system, according to the analysis framework.

```
bool CheckFileExist(TString fileName){
    gSystem->ExpandPathName(fileName);
    if (gSystem->AccessPathName(fileName.Data()) == true)
    {
        cout<<endl<<"no specified file: "<<fileName<<endl;
        return false;
    }

    return true;
}
//_____________________________________________________________________
void RunAnalyses(int nEvents = -1, TString inFileList = "listabox.txt"){

  gROOT->LoadMacro("mpdloadlibs.C"); // looks unnecessary, but it is to load at
  gROOT->ProcessLine("mpdloadlibs()"); // you can add here the functions directl

  TStopwatch timer;
  timer.Start();

  ProcInfo_t proc;
  MemInfo_t memory;

   MpdAnalysisManager man("ManagerAnal", nEvents) ;
if (!CheckFileExist(inFileList)) return;
        man.InputFileList(inFileList) ;
        man.ReadBranches("*") ;
        man.SetOutput("histos.root") ;

   MpdPtMCAnalysisTask MpdPtMCAnalysisTask("V0","V0") ;
   man.AddTask(&MpdPtMCAnalysisTask) ;

   man.Process() ;
```

Figure 6: *File listabox.txt within the RunAnalyses.C macro.*

3.2 Optimize track selection criteria

Once the events are generated, we will be ready to analyze them. To do this, we will use the classes *MpdPtMCAnalysisTask.cxx* and *MpdPtMCAnalysisTask.h*, both macros will be available in the repository [11].

As already mentioned the macro RunAnalyses. C will be that containing wagons, in this case, the wagon *MpdPtMCAnalysisTask*. This vague will have the tasks of analysis, but divided into three functions: *UserInit*(), *ProcessEvent*() and *Finish*(). For our case only tasks were established in the first two functions.

3.2.1 UserInit()

Within this function, two-dimensional histograms were defined for the respective energy loss distributions (dE/dx) of each of the particles mentioned above.

```
//Histograms of ionization energy loss distribution_reconstructed tracks
mhdEdx = new TH2F("hdEdx", "Reconstructed dE/dx distribution; |P|(GeV/c); dE/dx(KeV/cm)", 200, 0, 5., 1000, -100, 10000);
fOutputList->Add(mhdEdx);

mhdEdxHe3 = new TH2F("hdEdxHe3", "Reconstructed dE/dx distribution for He3; |P|(GeV/c); dE/dx(KeV/cm)", 200, 0, 5., 1000, -100, 10000);
fOutputList->Add(mhdEdxHe3);

mhdEdxd = new TH2F("hdEdxd", "Reconstructed dE/dx distribution for d; |P|(GeV/c); dE/dx(KeV/cm)", 200, 0, 5., 1000, -100, 10000);
fOutputList->Add(mhdEdxd);

mhdEdxK = new TH2F("hdEdxK", "Reconstructed dE/dx distribution for K; |P|(GeV/c); dE/dx(KeV/cm)", 200, 0, 5., 1000, -100, 10000);
fOutputList->Add(mhdEdxK);

mhdEdxp = new TH2F("hdEdxP", "Reconstructed dE/dx distribution for p; |P|(GeV/c); dE/dx(KeV/cm)", 200, 0, 5., 1000, -100, 10000);
fOutputList->Add(mhdEdxp);

mhdEdxe = new TH2F("hdEdxe", "Reconstructed dE/dx distribution for e; |P|(GeV/c); dE/dx(KeV/cm)", 200, 0, 5., 1000, -100, 10000);
fOutputList->Add(mhdEdxe);

mhdEdxpi = new TH2F("hdEdxpi", "Reconstructed dE/dx distribution for Pi; |P|(GeV/c); dE/dx(KeV/cm)", 200, 0, 5., 1000, -100, 10000);
fOutputList->Add(mhdEdxpi);

mhdEdxpie = new TH2F("hdEdxpie", "Reconstructed dE/dx distribution for pi-e; |P|(GeV/c); dE/dx(KeV/cm)", 200, 0, 5., 1000, -100, 800);
fOutputList->Add(mhdEdxpie);

mhdEdxpara = new TH2F("hdEdxpara", "Reconstructed dE/dx distribution for e parame; |P|(GeV/c); dE/dx(KeV/cm)", 200, 0, 5., 1000, -100, 10000);
fOutputList->Add(mhdEdxpara);

mhdEdxPpara = new TH2F("hdEdxPpara", "Reconstructed dE/dx distribution for P parame; |P|(GeV/c); dE/dx(KeV/cm)", 200, 0, 5., 1000, -100, 10000

fOutputList->Add(mhdEdxPpara);
//_____________________________
```

Figure 7: *dE/dx histograms defined in the UserInit() farm of the MpdPtMCAnalysisTask.cxx macro.*

3.2.2 ProcessEvent()

Within this function were established the cuts and the type of track with which the data was to be analyzed.

Reconstructed tracks

For the analysis of the distribution of the energy loss of each particle, it was necessary to work with the generated events associated with the *reconstructed tracks* or *Global Tracks* instead of the *Monte Carlo tracks*. This is because, as mentioned above, energy loss occurs when passing through a medium (the detector), which implies the need to know the characteristics of the detector material, dispersion effects, detector resolution, etc. This information will be taken into account by the *Global Tracks*, as they are based on real experimental data, while the *Monte Carlo tracks* provide a theoretical or ideal reference of how the trajectories

of particles would be under ideal conditions, without taking into account possible distortions or errors in the detection.

```
//Reconstructed tracks
fTDstEvent = event.fMPDEvent;
fTMpdGlobalTracks = event.fMPDEvent->GetGlobalTracks();
Int_t ntracks=fTMpdGlobalTracks->GetEntriesFast();

for (Int_t i = 0; i < ntracks; i++)
{
    MpdTrack *track = (MpdTrack*) fTMpdGlobalTracks->UncheckedAt(i);
    Int_t idtrack = track->GetID();
    MpdMCTrack *mcTr = (MpdMCTrack*)fTMCTracks->UncheckedAt(idtrack);
```

Figure 8: *Global Tracks association to events generated within the MpdPtMCAnalysisTask.cxx macro.*

Cuts

The variables to which cuts were applied were pseudorapidity, transverse momentum and number of hits.

Symb.	Definition	Value or (usual) units
$\|\eta\|$	pseudorapidity	>1.3
$\|P_T\|$	transverse momentum	< 0.1
$nHits$	number of hits	<16

Table 4: *Variables with restricted values*

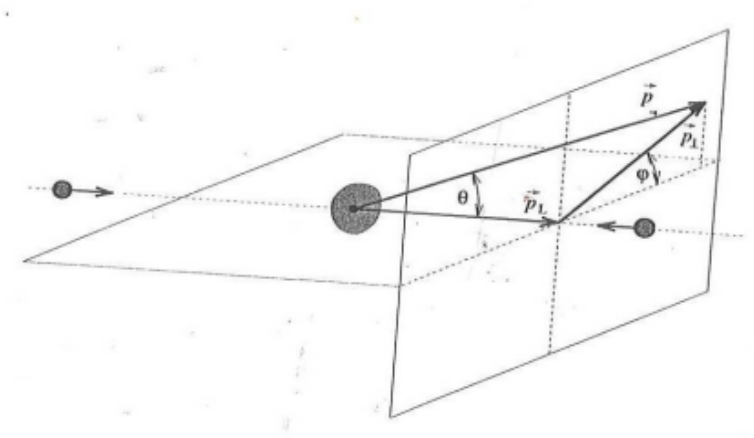

Figure 9: *Pictorial Representation of Detector System.*

The selection of these values, as can be seen in figure 9, and according to the expressions:

$$\eta = -ln\left[tan\left(\frac{\theta}{2}\right)\right], \tag{2}$$

$$|P_T| = |P|Sin\,(\theta)\,. \tag{3}$$

This is because the region near the collision axis, that is, in the direction of the particle beam, was analyzed. The assignment of these values within the macro can be seen in the figure 10.

```
//_CUTS_____________________________
    Double_t ptmc=track->GetPt();
    fhistPt->Fill(ptmc);

    Double_t etamc=track->GetEta();
    fhisteta->Fill(etamc);

    //if(etamc > 1.3) continue;
    //if(etamc < -1.3) continue;

    Int_t nhits  = track->GetNofHits();

    if (TMath::Abs(ptmc) < 0.1) continue;
    if (TMath::Abs(etamc) > 1.3) continue;
    if (nhits < 16) continue;
//__________________________________________
```

Figure 10: *Restricción de valores en las variables η, p_T y nhits.*

Particle choice and values for histograms

PDG code

The Monte Carlo particle numbering scheme described in the text [15] provides a structured approach to encoding information about particles used in particle physics simulations. This scheme facilitates the interface between event generators, detector simulators and analysis packages. It assigns a 7-digit number to each particle, encoding details about its spin, flavor content, and internal quantum numbers.

The PDG code, maintained by the Particle Data Group, is a standardized particle numbering system used in particle physics. It assigns unique numeric codes to particles, facilitating data exchange and comparison between different experiments and analyses. It includes a wide range of particles, from quarks and fundamental leptons to composite hadrons and exotic particles.

```
//Particle choice_____________________________________________________________

    float dEdx = track->GetdEdXTPC();
    //mhdEdx->Fill(TMath::Abs(ptmc)*TMath::CosH(etamc), dEdx);
    mhdEdx->Fill(P.Mag(), dEdx);

    if (pdg == 211 || pdg == 11)
    {
        //Piones,electrones
        mhdEdxpie->Fill(TMath::Abs(ptmc)*TMath::CosH(etamc), dEdx);
    }

    if (pdg == 1000020030)
    {
        //He3
        mhdEdxHe3->Fill(TMath::Abs(ptmc)*TMath::CosH(etamc), dEdx);
    }else if (pdg == 1000010020)
    {
        //Deuterions
        mhdEdxd->Fill(TMath::Abs(ptmc)*TMath::CosH(etamc), dEdx);
    }else if (pdg == 321)
    {
        //Kaons
        mhdEdxK->Fill(TMath::Abs(ptmc)*TMath::CosH(etamc), dEdx);
    }else if (pdg == 2212)
    {
        //Protons
        mhdEdxp->Fill(TMath::Abs(ptmc)*TMath::CosH(etamc),dEdx);

    }else if (pdg == 11)
    {
        //Electrons
        mhdEdxe->Fill(TMath::Abs(ptmc)*TMath::CosH(etamc), dEdx);
    }else if (pdg == 211)
```

Figure 11: *Selection of particles according to their PDG code.*

As shown in figure 11, the classification of particulates was made with the help of *if* conditional structures when comparing the pdg variable (previously initialized with the code line *Int t pdg = TMath::Abs(mcTr->GetPdgCode());*) with the code pdg corresponding to each particulate.

Then, within the *if* blocks, the previously defined histograms are called to assign the values of each axis with the *Fill(TMath::Abs(ptmc)*TMath::CosH(etamc),dEdx);* command.

MpdPtMCAnalysisTask.h

Within this macro, histograms and functions defined in the MpdPtMCAnalysisTask.cxx macro are called.

```
#ifndef MPDPTMCANALYSISTASK_H
#define MPDPTMCANALYSISTASK_H

#include "MpdAnalysisTask.h"
#include "MpdMCTrack.h"
#include "TH1.h"
#include "TH2.h"

class MpdPtMCAnalysisTask : public MpdAnalysisTask {

public:
   MpdPtMCAnalysisTask() {}
   MpdPtMCAnalysisTask(const char *name, const char *outputName = "taskName");

   virtual ~MpdPtMCAnalysisTask() {}


   virtual void UserInit();
   virtual void ProcessEvent(MpdAnalysisEvent &event);
   virtual void Finish();

   void setOutFile(std::string filename = "histos.root") { mOutFile = filename; }

private:

   std::string mOutFile = "histos.root" ;

   MpdEvent *fTDstEvent;
   TClonesArray *fTMCTracks;
   TClonesArray  *fTMpdGlobalTracks = nullptr;
   //Histograms
   TH1F *fhistPtMC;
   TH1F *fhistetaMC;
   TH1F *fhistEnposMC;
```

Figure 12: *Macro MpdPtMCAnalysisTask.h*

3.2.3 Output file V0.root from the MpdPtMCAnalysisTask wagon.

To observe the histograms created, we must follow the following steps:

1.Load variables

Within the working environment, we must access */storage/mexnica/TallerMPD/AlejandroSanJuan/soft v23.12.23/build*. Once there, we must use the command *source variables.sh*, which will load the instructions shown in figure 13.

```
[tallermpd@xook build]$ more variables.sh
source /cvmfs/nica.jinr.ru/sw/os/login.sh legacy
module add mpddev
export MPDROOT=/storage/mexnica/TallerMPD/NombreApellido/v23.12.23/mpd
source $MPDROOT/config/env.sh
[tallermpd@xook build]$ source variables.sh
Loading mpddev/v23.06.23_wagons-1
```

Figure 13: *Instructions for the variable.sh file*

In addition, we must use the commands *cmake ..* and *make -j16 install* to check for errors in the files . cxx and .h.

2.Load RunAnalyses.C

Within the path /storage/mexnica/TallerMPD /AlejandroSanJuan/software/mpdroot-v23.12.23/physics /simplePt/macros ,we must use the *root RunAnalyses.C* command, which will load the generated events (see figure 14).

```
N of MC tracks = 13183
N of MC tracks = 13878
N of MC tracks = 14059
N of MC tracks = 13094
N of MC tracks = 12905
N of MC tracks = 12934
N of MC tracks = 13338
N of MC tracks = 13863
N of MC tracks = 13559
N of MC tracks = 13408
N of MC tracks = 13656
N of MC tracks = 13468
N of MC tracks = 13730
N of MC tracks = 13154
N of MC tracks = 13478
N of MC tracks = 13761
N of MC tracks = 12776
N of MC tracks = 14330
N of MC tracks = 13101
N of MC tracks = 13952
N of MC tracks = 12480
N of MC tracks = 14449
N of MC tracks = 13765
Writing output to file V0.root
 User CPU time: 27.5367seconds, Memory: resident 394704KB, virtual 663600K
 Total RAM: 128830MB, Used RAM: 124001 MB, Free RAM: 4829
RealTime=30.190764 seconds, CpuTime=26.820000 seconds
Macro finished successfully.
Error in <TList::Delete>: A list is accessing an object (0x69c8900) alread
Error in <TList::Delete>: A list is accessing an object (0x69c8900) alread
root [1]
```

Figure 14: *Events loaded with the RunAnalyses macro*

3.Load V0.root

The previous step will generate an output file called "V0", which will contain all created histograms. To open the file, we must use the command *root V0.root* and then the command *TBrowser m*, which will allow us to open a graphical interface to visualize the histograms.

```
[tallermpd@xook macros]$ root V0.root
   ------------------------------------------------------------------
  | Welcome to ROOT 6.26/10                        https://root.cern |
  | (c) 1995-2021, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Aug 17 2023, 08:42:00                 |
  | From tags/v6-26-10@v6-26-10                                      |
  | With c++ (GCC) 10.2.0                                            |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'       |
   ------------------------------------------------------------------

root [0]
Attaching file V0.root as _file0...
(TFile *) 0x29ece60
root [1] TBrowser m
```

Figure 15: *Output file V0.root*

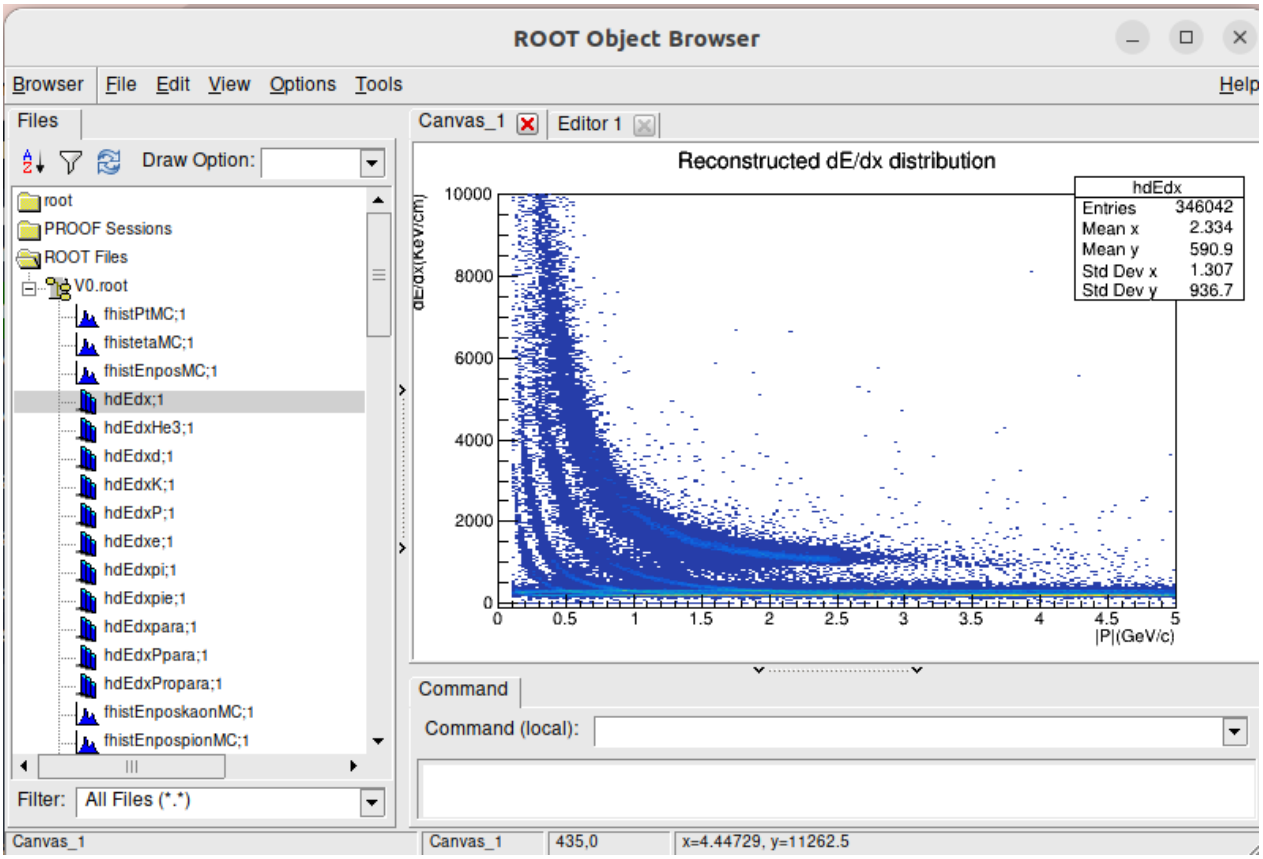

Figure 16: *Graphical interface to visualize the histograms*

3.3 Parametrize dE/dx bands for particle specie (i.e. positions and width at every p/q)

3.3.1 Fit function

As mentioned above, for the minimum ionization region, which is the region being worked on, the function that best fits this region is given by the Bethe equation. However, it was also mentioned that there might be other proposed functions describing the study region. According to this and two classes (MpdPairK and MpdPid) hosted in the NICA repository, two functions were obtained that fulfill the function of adjusting the simulated data, which are the following:

$$\langle \frac{dE}{dx} \rangle = \frac{P_0}{p} \left[P_1 log\left(p^2\right) - P_2 p^2 - P_3 p - P_4 - P_5 p^3 \right] + P_6, \tag{4}$$

$$\langle \frac{dE}{dx} \rangle = \frac{P_0}{\beta^{P_3}} \left[P_2 - \beta^{P_3} - ln\{P_2 + \left(\frac{1}{\beta\gamma}\right)^{P_4}\} \right], \qquad \beta = \frac{P}{\sqrt{p^2 + M^2}}. \tag{5}$$

Where P_N are the parameter of the function and p the moment.

```
// dE/dx parameterizations for el/Pi/K/p
float MpdPairKK::dEdx_sigma_El(float dEdx, float mom) const
{
   if (mom < 0.04) return -999;
   if (dEdx <= 0) return -999;

   dEdx = log(dEdx);

   if (mom < 0.06) mom = 0.06;
   if (mom > 2.0) mom = 2.0;

   float mean[7]  = {6.869878e-003, -1.573146e-001, 1.847371e+000, -9.253678e-001,
                     1.073907e+000, -3.394239e+000, 6.451762e-001};
   float width[7] = {-4.359368e+006, -8.508504e-012, -3.958364e-009, 1.526816e-009,
                      1.353776e-011,  3.426352e-009,  6.591542e-002};

   float mean_exp, width_exp;

   mean_exp =
      mean[0] / mom / mom *
         (mean[1] * log(mom * mom) - mean[2] * mom * mom - mean[3] * mom - mean[4] - mean[5] * mom * mom * mom) +
      mean[6];
   width_exp =
      width[0] / mom / mom *
         (width[1] * log(mom * mom) - width[2] * mom * mom - width[3] * mom - width[4] - width[5] * mom * mom * mom) +
      width[6];

   return (dEdx - mean_exp) / width_exp;
}
```

Figure 17: *MpdPairKk.cxx class [10].*

```
Double_t MpdPid::parPrBB(Double_t *x, Double_t *par)
{
   Double_t x1, x2, x3, p[5], xint = fTrackingState != MpdPidUtils::kCFHM ? 0.384089 : 0.34173458, ans;
   for (Int_t k = 0; k < 5; k++) p[k] = x[0] < xint ? par[k] : par[k + 5];
   x1  = p[0] / TMath::Power(x[0] / TMath::Sqrt(x[0] * x[0] + 0.88), p[3]);
   x2  = p[1] - TMath::Power(x[0] / TMath::Sqrt(x[0] * x[0] + 0.88), p[3]);
   x3  = TMath::Log(p[2] + TMath::Power(1.0 / (x[0] / 0.9383), p[4]));
   ans = x1 * (x2 - x3);
```

Figure 18: *MpdPid.cxx class [11]*

To use equation (5), we must implement the MpdPid class, as this class automatically calculates the values of the necessary parameters. However, due to lack of time, it was not possible to fully understand the functions on which the MpdPid class depends. Therefore, equation (4) of the MpdPairK class was chosen, which involved manually obtaining the parameter values.

3.3.2 Cuts in the dE/dx distributions

Because we have 6 parameters this implies that our degrees of freedom should be greater than 6, so to properly estimate the parameters we will take 21 degrees of freedom or cuts in the dE/dx distribution. These cuts will be made in the following way:

1. On the histogram to work on, select the *setShowProjectionY* option and specify a bin value.

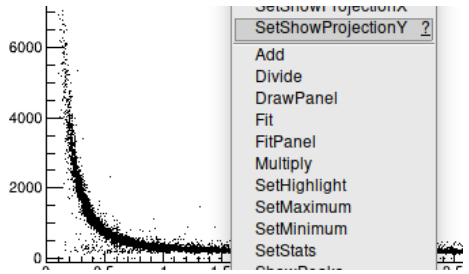

Figure 19: *setShowProjectionY option.*

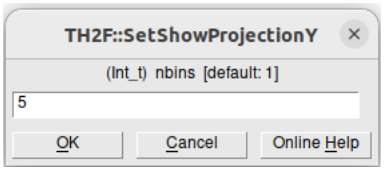

Figure 20: *Value of bin equal to 5*

2. A second window will be displayed showing the distribution of the number of entries that belong to the selected cut.

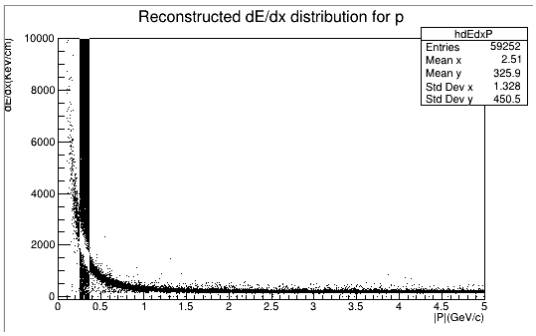

Figure 21: *First cut in the dE/dx distribution.*

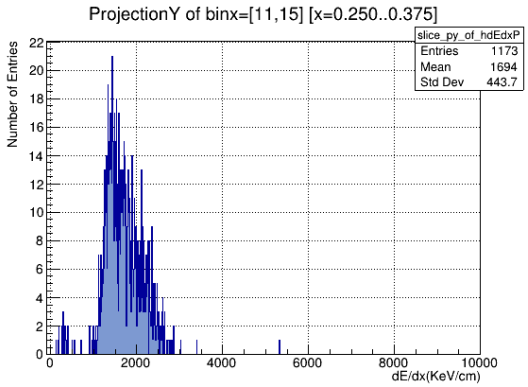

Figure 22: *Projection of the distribution on the y axis.*

3.3.3 Gaussian fit for coutting

Now a Gaussian function must be fitted to the input distribution, it will be done as follows:

1. In the distribution select the *FitPanel* option and in the pop-up window select the *Predef-1D* function type, as well as the fit region.

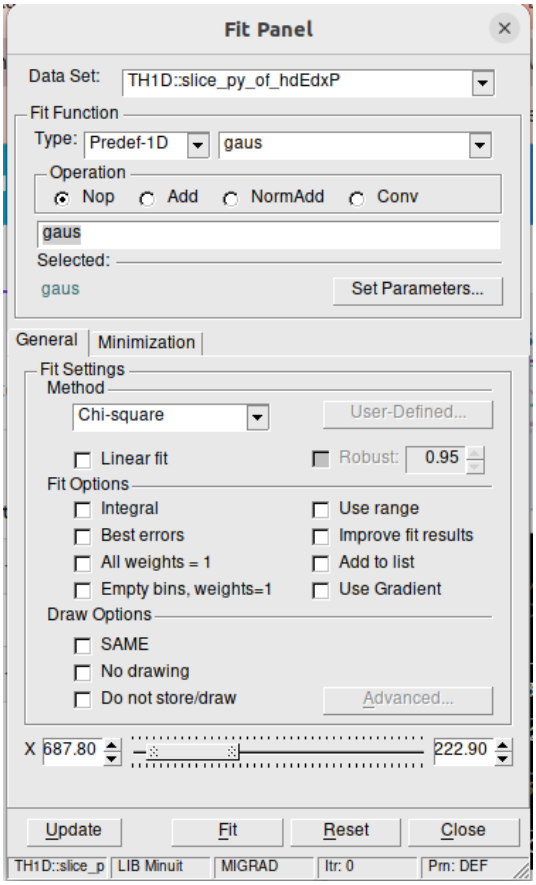

Figure 23: *Setting options within the Fitpanel.*

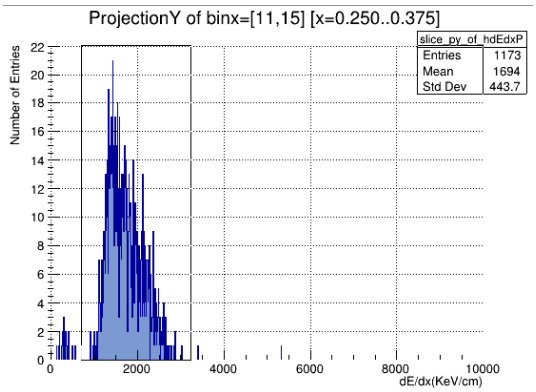

Figure 24: *Projection region to adjust according to Fitpanel settings.*

2. Once the options are specified, the "Fit" option is selected and we obtain.

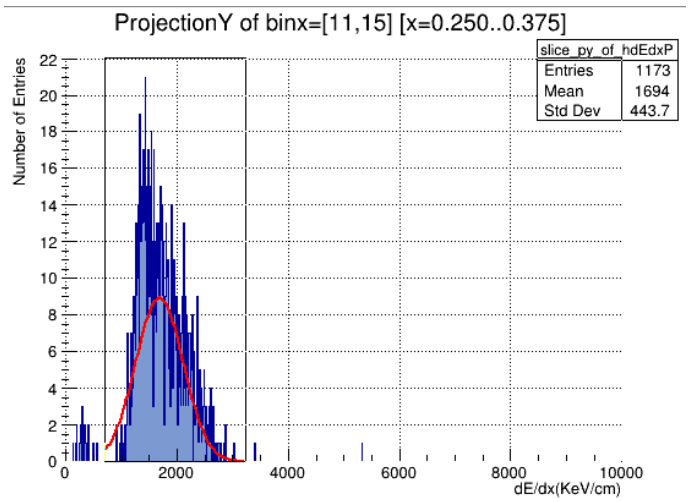

Figure 25: *Gaussian adjustment to the selected projection region.*

EXT NO.	PARAMETER NAME	VALUE	ERROR	STEP SIZE	FIRST DERIVATIVE
1	Constant	8.93855e+00	4.12379e-01	2.42165e-03	-6.75276e-04
2	Mean	1.67643e+03	1.71176e+01	1.34244e-01	-1.38172e-05
3	Sigma	4.27806e+02	1.75406e+01	8.48956e-05	-4.49508e-02

Figure 26: *Values of degrees of freedom (mean) and errors of the adjustment, described in the terminal.*

3.3.4 Degrees of freedom and parameters

1. Once the 21 cuts have been made and, therefore, the 21 degrees of freedom with their respective errors have been obtained, a histogram will be defined that graphs said values.

```
void plotProton(){

   TH1F *fa = new TH1F("fa", "fa title", 21, 0.050, 2.675);
   fa->SetBinContent(1,5.26459e+03);
   fa->SetBinContent(2,2.33194e+03);
   fa->SetBinContent(3,1.31347e+03);
   fa->SetBinContent(4,8.07440e+02);
   fa->SetBinContent(5,5.69834e+02);
   fa->SetBinContent(6,4.40979e+02);
   fa->SetBinContent(7,3.67455e+02);
   fa->SetBinContent(8,3.18473e+02);
   fa->SetBinContent(9,2.87055e+02);
   fa->SetBinContent(10,2.64130e+02);
   fa->SetBinContent(11,2.49205e+02);
   fa->SetBinContent(12,2.35646e+02);
   fa->SetBinContent(13,2.26963e+02);
   fa->SetBinContent(14,2.20440e+02);
   fa->SetBinContent(15,2.15692e+02);
   fa->SetBinContent(16,2.10908e+02);
   fa->SetBinContent(17,2.07808e+02);
   fa->SetBinContent(18,2.06383e+02);
   fa->SetBinContent(19,2.03644e+02);
   fa->SetBinContent(20,2.01427e+02);
   fa->SetBinContent(21,2.00220e+02);
   fa->SetBinError(1,4.05030e+03);
   fa->SetBinError(2,9.19109e+01);
   fa->SetBinError(3,9.94266e+00);
   fa->SetBinError(4,3.42809e+00);
   fa->SetBinError(5,1.83025e+00);
   fa->SetBinError(6,1.33770e+00);
   fa->SetBinError(7,9.42899e-01);
   fa->SetBinError(8,7.54512e-01);
   fa->SetBinError(9,6.50320e-01);
   fa->SetBinError(10,5.78024e-01);
   fa->SetBinError(11,5.70373e-01);
   fa->SetBinError(12,5.19311e-01);
   fa->SetBinError(13,5.18305e-01);
   fa->SetBinError(14,5.23691e-01);
   fa->SetBinError(15,4.87780e-01);
   fa->SetBinError(16,4.68820e-01);
   fa->SetBinError(17,4.54431e-01);
   fa->SetBinError(18,4.57337e-01);
   fa->SetBinError(19,4.71617e-01);
   fa->SetBinError(20,4.31029e-01);
   fa->SetBinError(21,4.59270e-01);

   fa->Draw();
   }
```

Figure 27: *Definition of the histogram with their respective degrees of freedom and errors.*

2. With the root command *plotProton.C*(where plotProton is the name of the macro containing the histogram), we get the graph of the points and their respective errors.

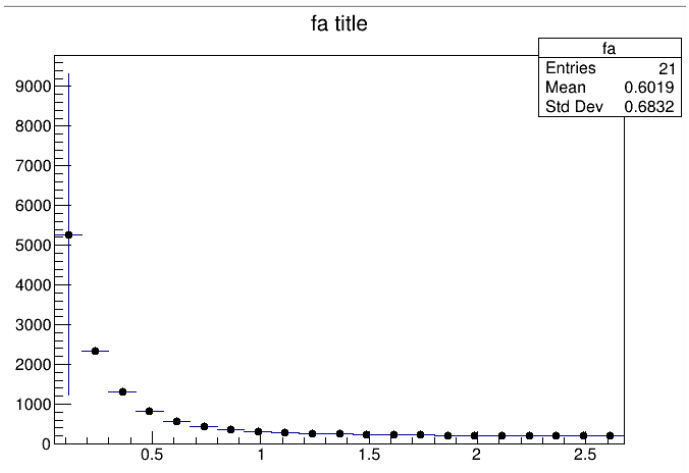

Figure 28: *Distribution of energy loss using degrees of freedom.*

3. The plotted points will be fitted with the help of the FitPanel. To do this, the "pol9" option will be selected, which refers to a ninth-degree polynomial. This fitting will provide tentative values for the 6 parameters.

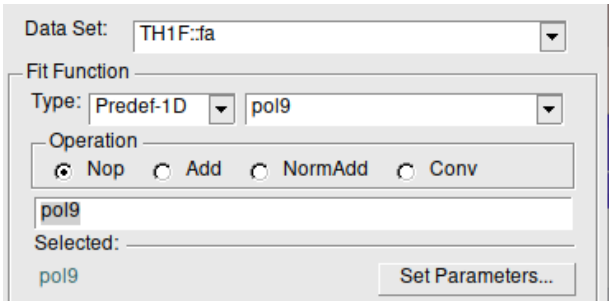

Figure 29: *Selection of the adjustment function for the graphed points.*

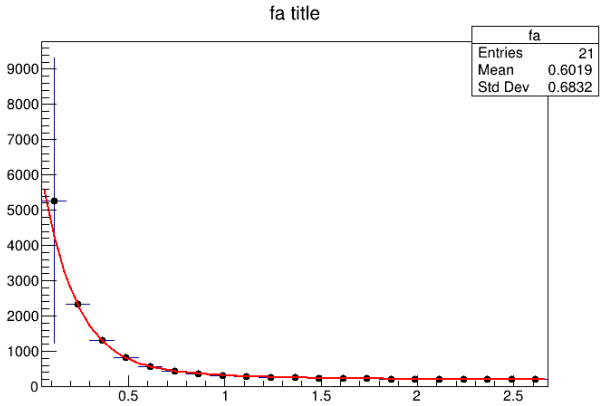

Figure 30: *First fit : ninth degree polynomial.*

```
Minimizer is Linear / Migrad
Chi2                      =      15.7892
NDf                       =           11
p0                        =      7676.36   +/-   371.248
p1                        =       -39040   +/-   2952.39
p2                        =      96366.4   +/-   9913.83
p3                        =      -141337   +/-   18506.7
p4                        =       133083   +/-   21243.6
p5                        =     -82655.6   +/-   15608
p6                        =      33709.3   +/-   7365.77
p7                        =     -8686.79   +/-   2160.04
p8                        =      1282.56   +/-   358.235
p9                        =     -82.6626   +/-   25.6682
```

Figure 31: *First parameter values: ninth degree polynomial.*

4. From the previous step 10 values for the parameters turned out, but because our selected function (equation (4)) only depends on 7, then we will only select the first 7 and enter them into the plotProton.C macro, in addition to the expression of equation (4).

```
TF1 *dEdx_mean = new TF1("dEdx_mean", " [0] / x / x * ([1] * log(x * x) - [2]
* x * x - [3] * x - [4] - [5] * x * x * x) + [6]", 0.050, 2.675);

  dEdx_mean->SetParameter(0,7676.36);
  dEdx_mean->SetParameter(1,-39040);
  dEdx_mean->SetParameter(2,96366.4);
  dEdx_mean->SetParameter(3,-141337);
  dEdx_mean->SetParameter(4,133083);
  dEdx_mean->SetParameter(5,-82655.6);
  dEdx_mean->SetParameter(6,33709.3);

  fa->Fit(dEdx_mean, "R");

  fa->Draw();
  dEdx_mean->Draw("same");
  }
```

Figure 32: *Function of energy loss with the first values of parameters.*

3.3.5 Fit function for each dE/dx

The steps 3 and 4 will be repeated to obtain a better estimation of the parameters, but the option chosen will be the $< \frac{dE}{dx} >$ function introduced in the macro, then we will get:

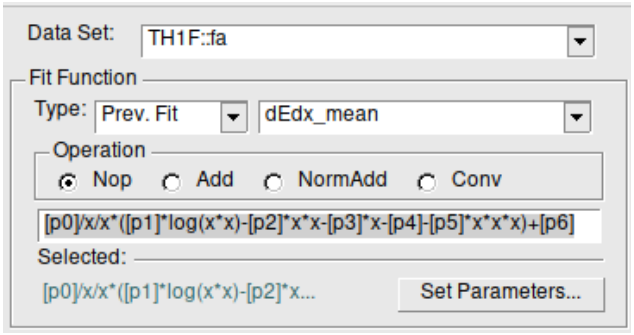

Figure 33: $< dE/dx >$ *as an second fit function.*

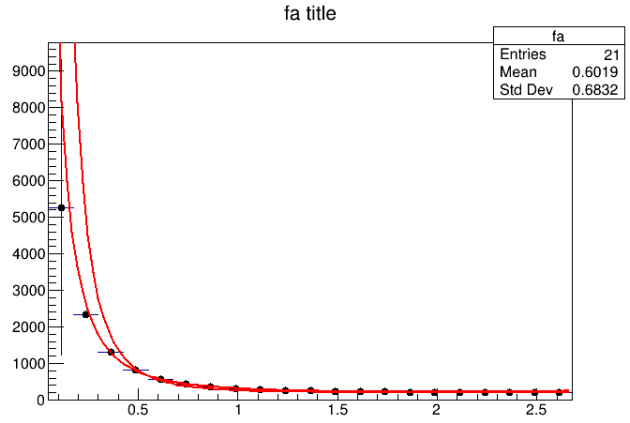

Figure 34: *Second fit:* $< dE/dx >$*, A better fit is achieved.*

```
TFitEditor::DoFit - using function dEdx_mean  0x539dd00
 FCN=30.3397 FROM HESSE     STATUS=NOT POSDEF     50 CALLS         188 TOTAL
                     EDM=6.27277e-10    STRATEGY= 1      ERR MATRIX NOT POS-DEF
  EXT PARAMETER                APPROXIMATE        STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  p0           1.89931e-04   3.98175e-06   2.51828e-10  -3.76397e+00
   2  p1           2.03523e+05   5.49913e+03   9.70479e-02  -1.43884e-08
   3  p2          -1.33570e+05   8.78392e+03   6.36911e-02   2.00450e-08
   4  p3           1.19415e+06   1.33680e+04   5.69417e-01   1.57012e-08
   5  p4          -1.57318e+06   1.52985e+04   6.83644e-01   9.33352e-09
   6  p5           3.47145e+04   2.77417e+03   1.65529e-02   2.38130e-08
   7  p6           2.24198e+02   1.66795e+00   7.28362e-05  -1.05567e-04
```

Figure 35: *Second parameter values:* $< dE/dx >$*.*

```
  TF1 *dEdx_mean = new TF1("dEdx_mean", " [0] / x / x * ([1] * log(x * x) - [2
* x * x - [3] * x - [4] - [5] * x * x * x) + [6]", 0.050, 2.675);

  dEdx_mean->SetParameter(0,1.90080e-04);
  dEdx_mean->SetParameter(1,2.03365e+05);
  dEdx_mean->SetParameter(2,-1.33086e+05);
  dEdx_mean->SetParameter(3,1.19323e+06);
  dEdx_mean->SetParameter(4,-1.57195e+06);
  dEdx_mean->SetParameter(5,3.46886e+04);
  dEdx_mean->SetParameter(6,2.24271e+02);


  fa->Fit(dEdx_mean, "R");

  fa->Draw();
  dEdx_mean->Draw("same");
  }
```

Figure 36: *Function of energy loss with the second values of parameters.*

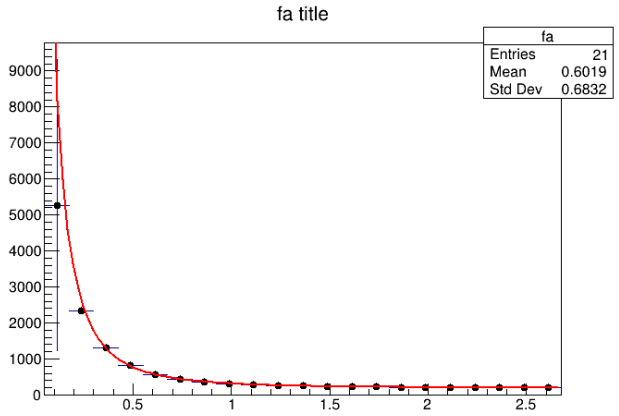

Figure 37: *Final adjustment of energy loss distribution.*

4 Results

4.1 Ionization energy loss for primary particles

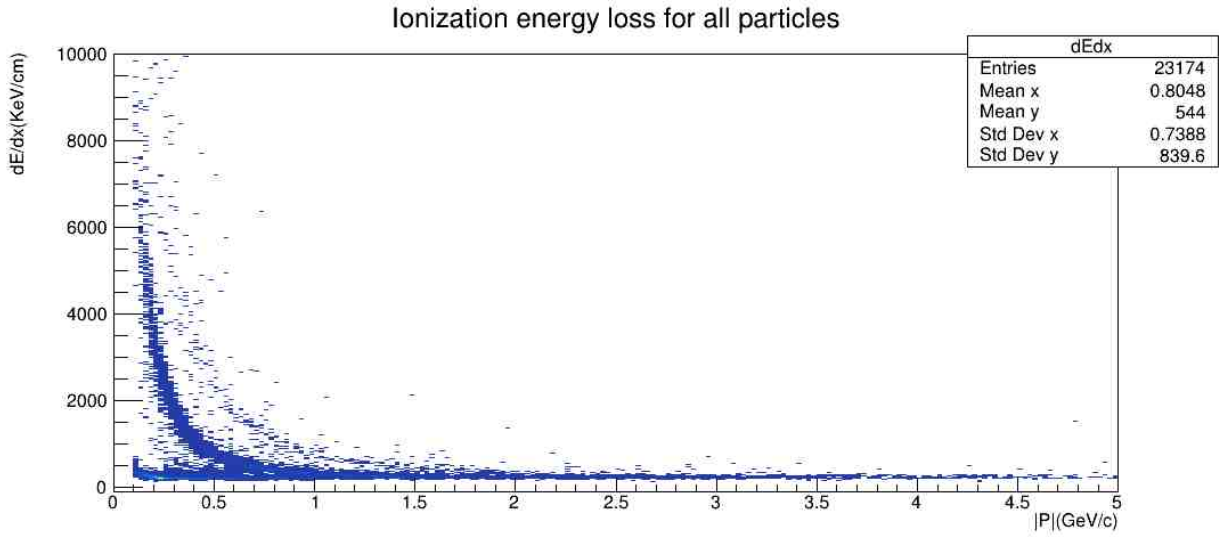

Figure 38: *Loss of ionization energy as a function of momentum for primary particles* (He^3, D, K, P, e, π).

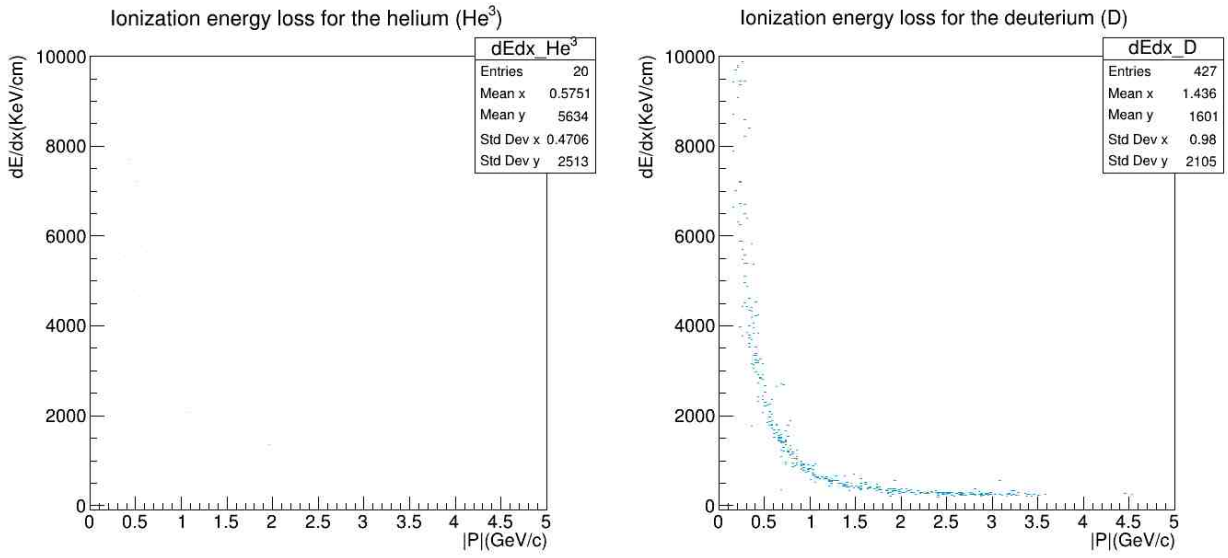

Figure 39: *Loss of ionization energy as a function of momentum for primary particles (He^3 and D).*

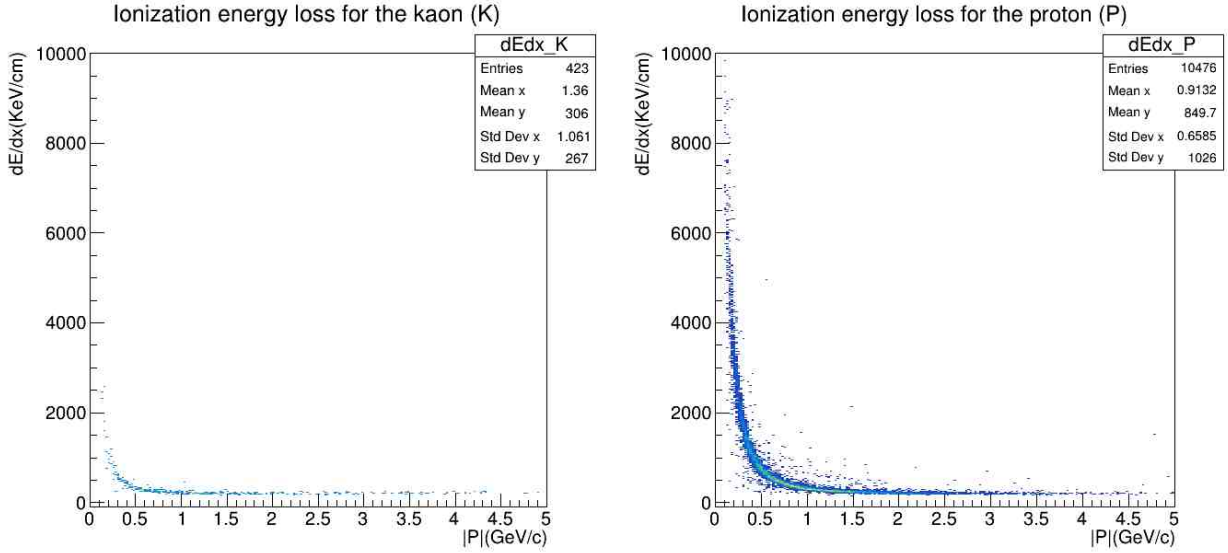

Figure 40: *Loss of ionization energy as a function of momentum for primary particles (K and P).*

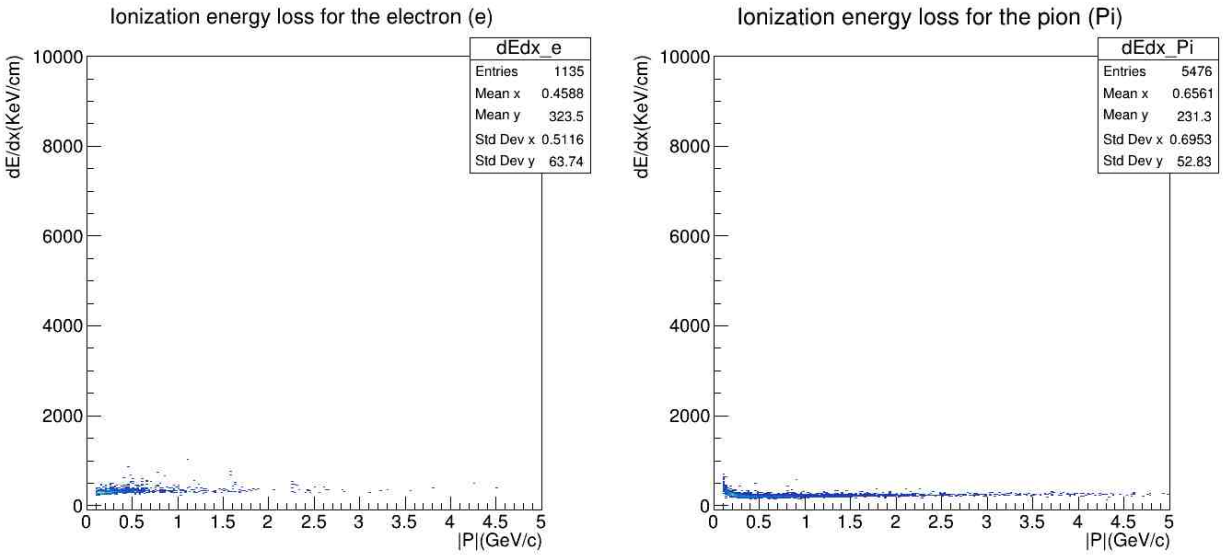

Figure 41: *Loss of ionization energy as a function of momentum for primary particles (e and π).*

4.2 Ionization energy loss for secondary particles

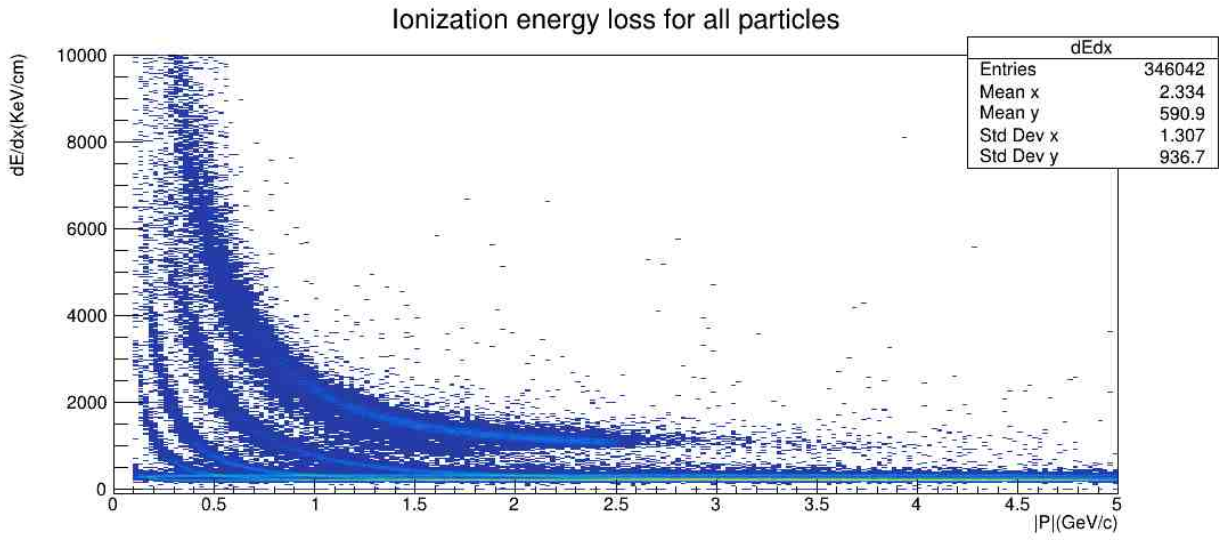

Figure 42: *Loss of ionization energy as a function of momentum for secondary particles (He^3, D, K, P, e, π).*

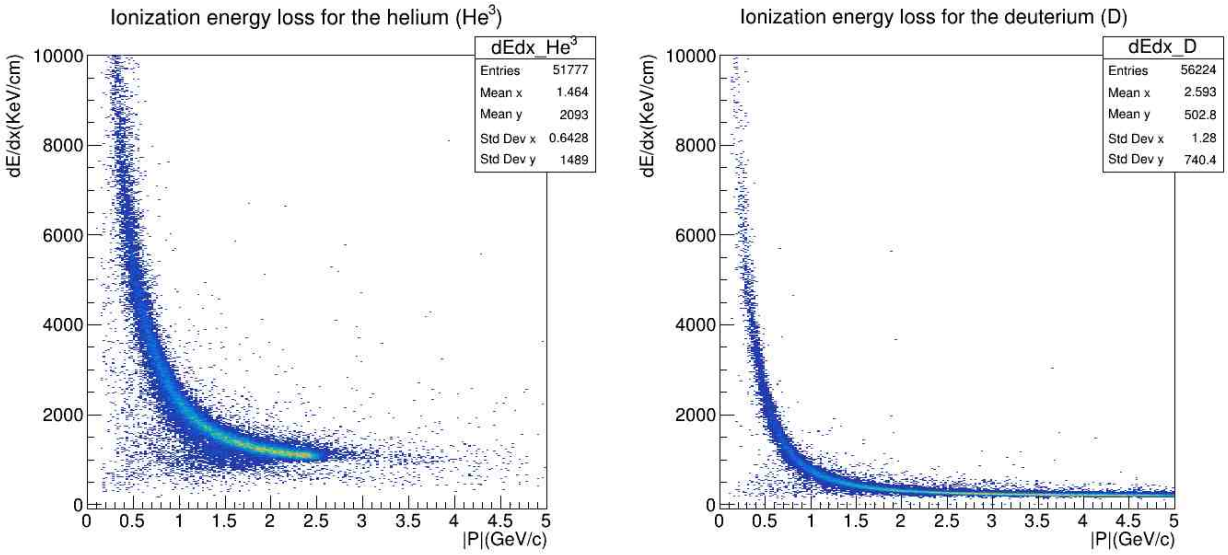

Figure 43: *Loss of ionization energy as a function of momentum for secondary particles (He^3 and D).*

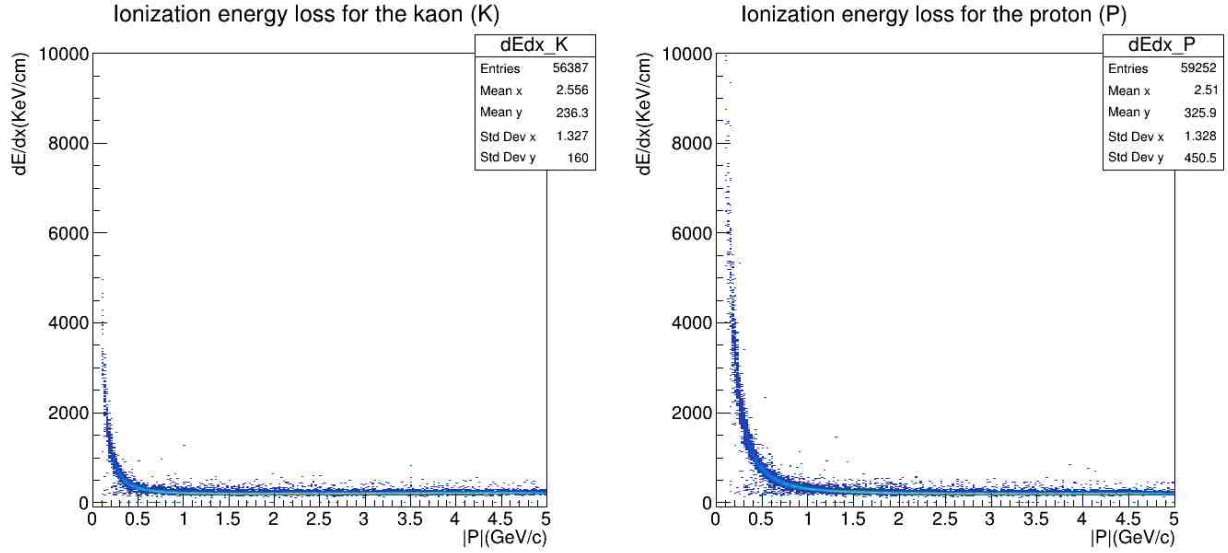

Figure 44: *Loss of ionization energy as a function of momentum for secondary particles (K^3 and P).*

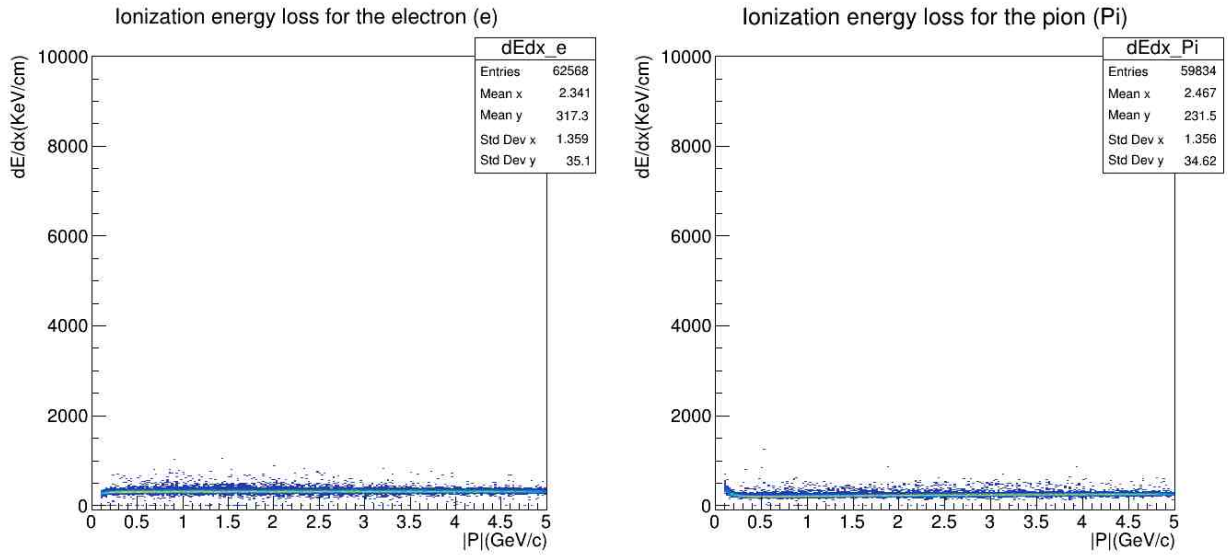

Figure 45: *Loss of ionization energy as a function of momentum for secondary particles (e and π).*

4.3 Ionization energy loss distribution for secondary particles adjusted by function $< dE/dx >$

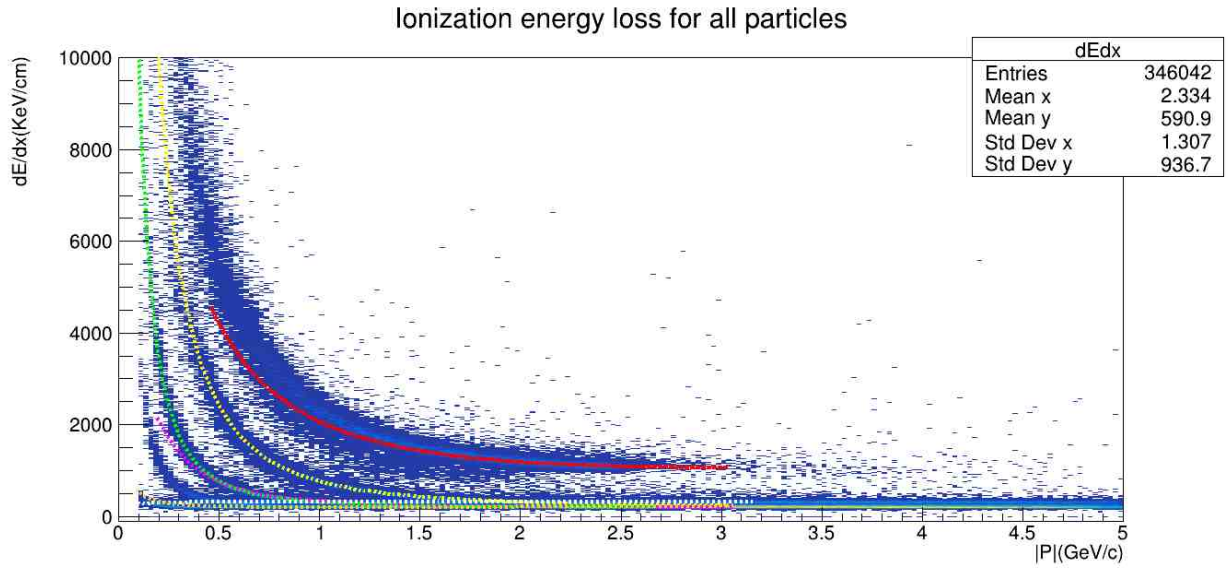

Figure 46: *Distribution of dE/dx and adjustment function <dE/dx> for secondary particles (He^3, D, K, P, e, π).*

4.3.1 Helium (He^3)

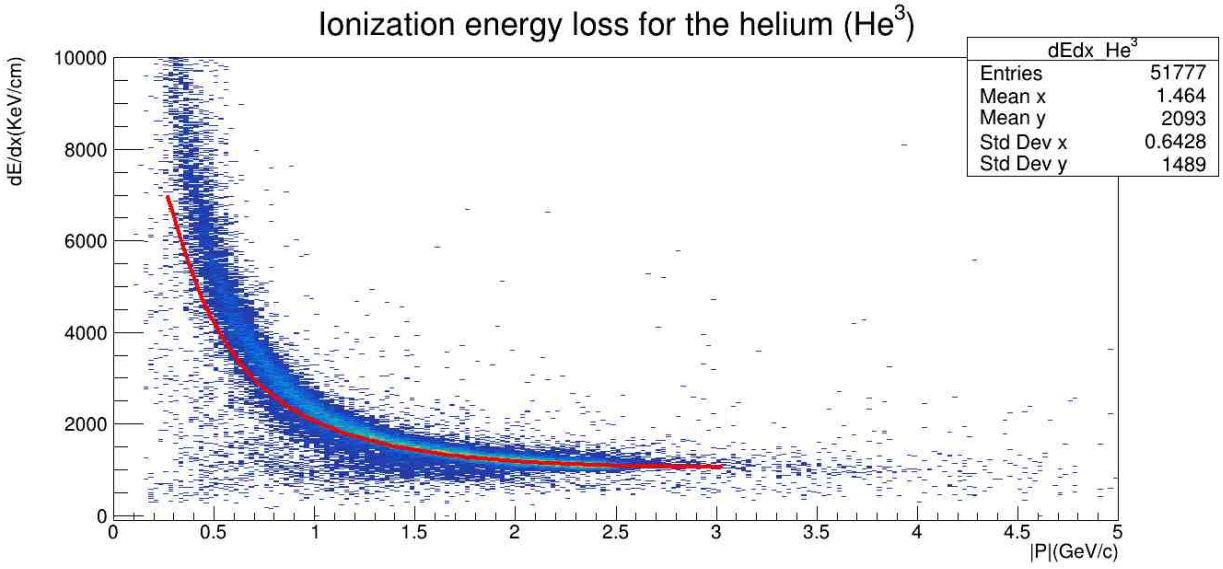

Figure 47: *Distribution of dE/dx and adjustment function <dE/dx> for secondary particles (He^3).*

Parameter	Value
P_0	3.79118e-03
P_1	5.58846e+04
P_2	1.98720e+05
P_3	-3.24806e+05
P_4	-1.92612e+05
P_5	-5.48625e+04
P_6	6.51674e+02

Table 5: *Parameter values of the fit function <dE/dx> for helium 3.*

4.3.2 Deuterium (D)

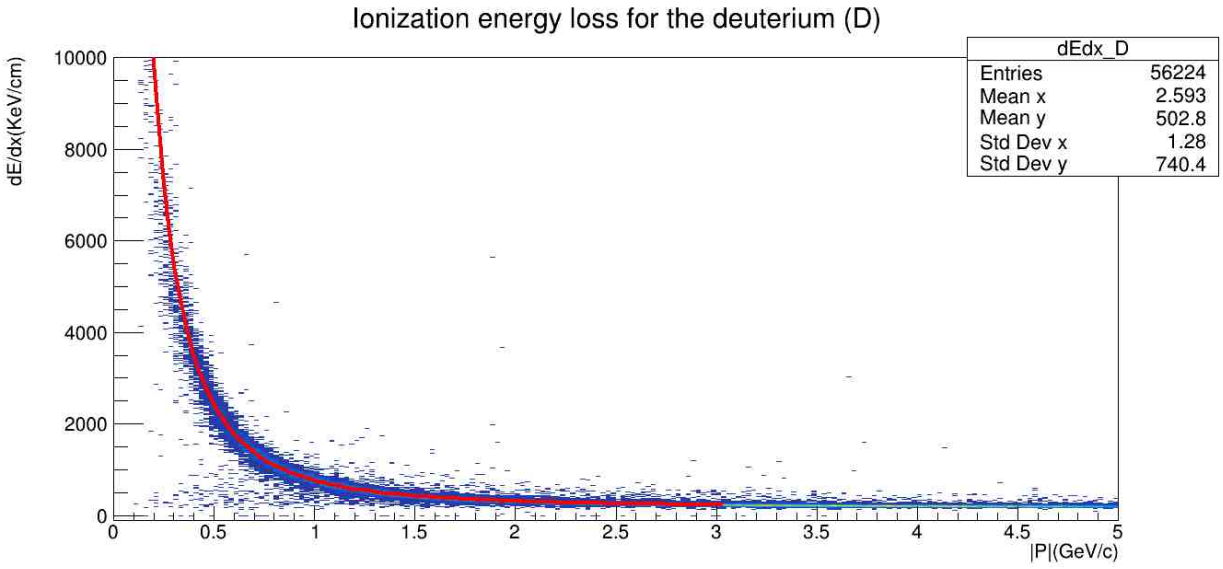

Figure 48: *Distribution of dE/dx and adjustment function <dE/dx> for secondary particles (D).*

Parameter	Value
P_0	-3.38998e-03
P_1	-5.60063e+04
P_2	7.21682e+04
P_3	-2.02277e+05
P_4	3.32763e+05
P_5	-5.24458e+03
P_6	9.93744e+01

Table 6: *Parameter values of the fit function <dE/dx> for deuterium.*

4.3.3 Kaon (K)

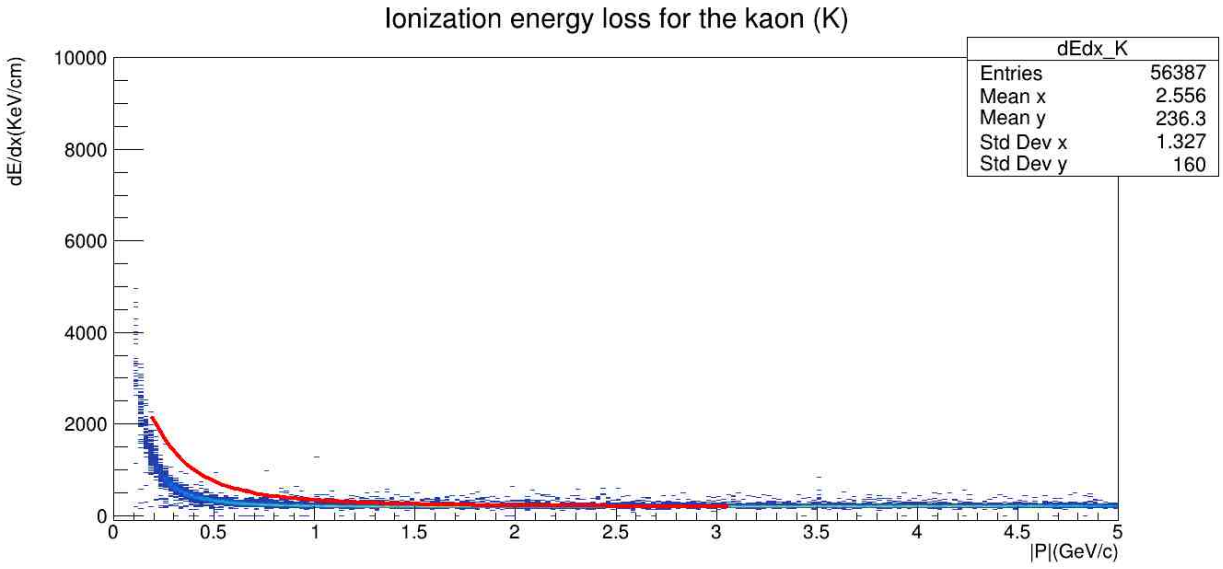

Figure 49: *Distribution of dE/dx and adjustment function <dE/dx> for secondary particles (K).*

Parameter	Value
P_0	-1.41242e-03
P_1	-4.47156e+04
P_2	7.77709e+04
P_3	-1.28144e+05
P_4	2.22833e+05
P_5	1.85692e+03
P_6	9.22746e+01

Table 7: *Parameter values of the fit function <dE/dx> for kaon.*

4.3.4 Proton (P)

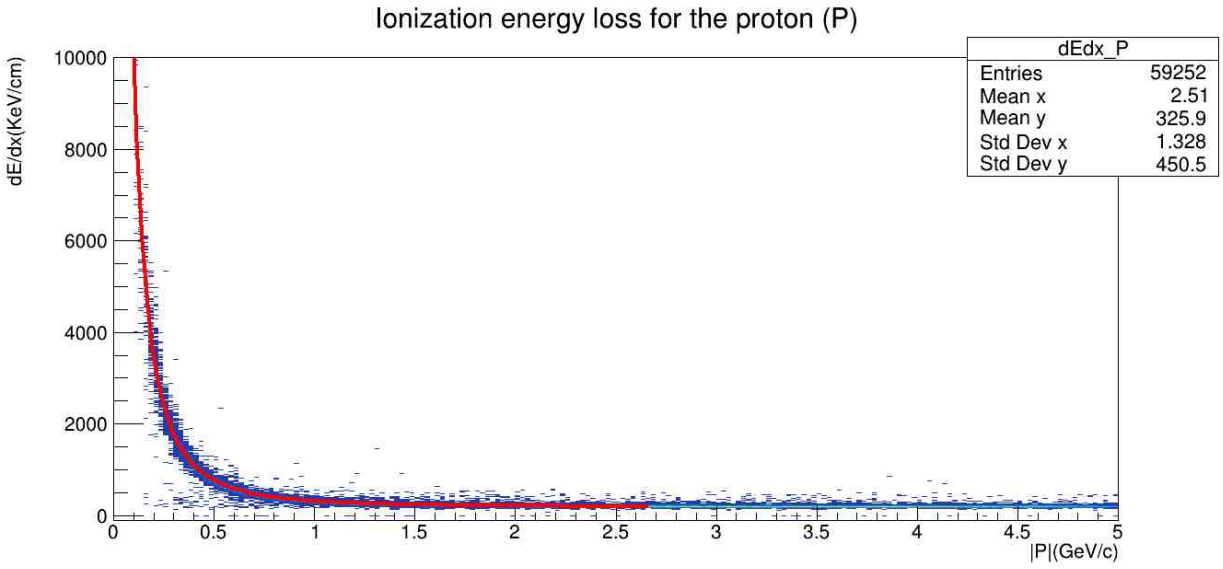

Figure 50: *Distribution of dE/dx and adjustment function <dE/dx> for secondary particles (P).*

Parameter	Value
P_0	1.90080e-04
P_1	2.03365e+05
P_2	-1.33086e+05
P_3	1.19323e+06
P_4	-1.57195e+06
P_5	3.46886e+04
P_6	2.24271e+02

Table 8: *Parameter values of the fit function <dE/dx> for proton.*

4.3.5 Electron (e)

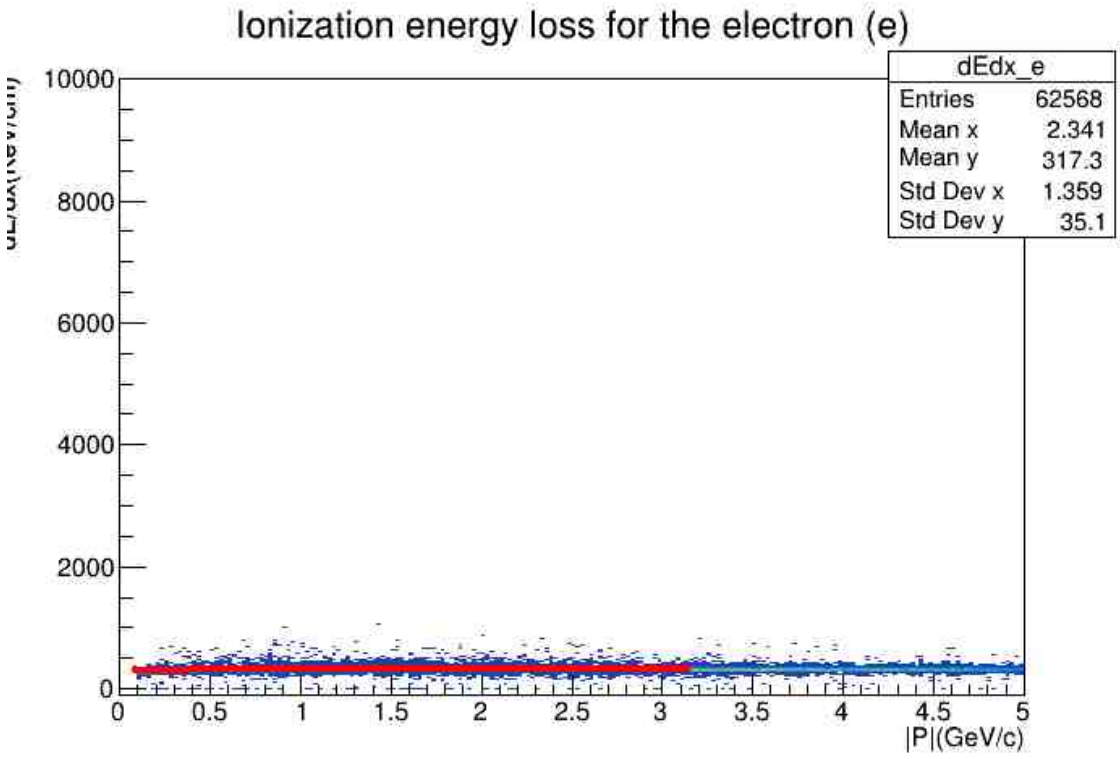

Figure 51: *Distribution of dE/dx and adjustment function <dE/dx> for secondary particles (e).*

Parameter	Value
P_0	5.90671e-05
P_1	-1.20552e+04
P_2	6.85698e+04
P_3	-1.01501e+05
P_4	7.02024e+04
P_5	-3.11711e+04
P_6	3.15599e+02

Table 9: *Parameter values of the fit function <dE/dx> for electron.*

4.3.6 Pion (π)

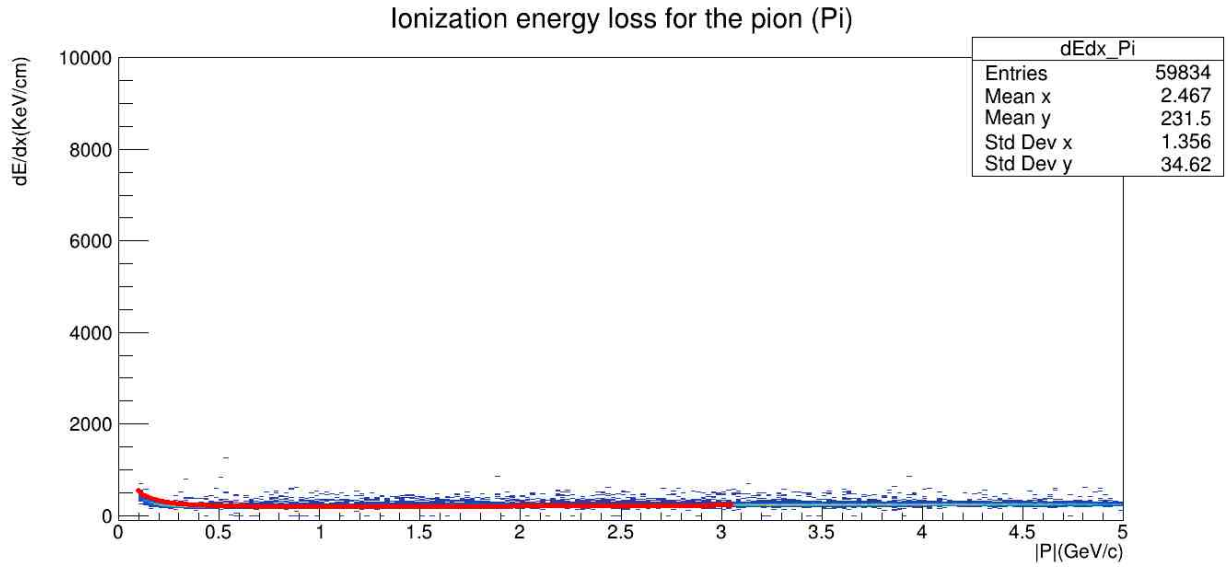

Figure 52: *Distribution of dE/dx and adjustment function <dE/dx> for secondary particles (π).*

Parameter	Value
P_0	7.14237e-04
P_1	1.04743e+03
P_2	4.33130e+04
P_3	-3.31036e+04
P_4	-6.89364e+03
P_5	-3.52237e+04
P_6	1.73607e+02

Table 10: *Parameter values of the fit function <dE/dx> for pion.*

5 Conclusions

As can be seen in the results, energy loss distributions were obtained as a function of the moment for primary and secondary particles. The most notable difference when comparing these charts is the number of entries recorded. An example of this is the distribution dE/dx for He^3, because when we consider this particle as primary, we notice that there is no record of its information. In contrast, considering it as a secondary particle, a clear distribution of its energy loss is observed, being the largest distribution of all the particles analyzed. This indicates that He^3 can be formed by the interaction of primary particles with the medium or by the decay of other particles.

For the other particles (deuterons, kaones, protons, electrons and pions) something similar happens, although not so drastic, since for both cases (primary and secondary), these particles show information about their loss of energy. At this point some of the objectives are met,

since important information was obtained about the generation of events, although it was still difficult to notice the separation of the energy loss distributions (dE/dx) for each type of particle. As can be seen in the figure 42, for moment values approximately greater than 1.5 GeV/c, the energy loss distributions begin to overlap.

To better understand these distributions, we obtained the functions <dE/dx> (figure 46) of each particle, which represent the expected average theoretical values. From figures 47 and 49, we note that the adjustment functions for He^3 and Kaons do not follow the distribution in certain regions. One possible explanation for this could be the fact that, when we made the cuts and obtained the values for the degrees of freedom, we also obtained errors of the same order of magnitude. In spite of this, the adjustment functions for the other particles are good, which gives us an idea of how should be the trend that follow the distributions of loss of ionization energy for each type of particle, and thus achieve a better identification.

6 Acknowledgments

I express my gratitude to Dr. Ivonne Maldonado for her invaluable advice, support, and patience in guiding me through the utilization of the MPD software, as well as for her availability for meetings and discussions beyond the established schedules. I also extend my appreciation to Dr. Vadim Kolesnikov for his valuable contributions and guidance. Furthermore, I would like to thank Dr. Luis Alberto Hernández for his insightful advice and discussions regarding the activities carried out within the INTEREST Wave 10 program.

Bibliography

[1] Abelev, B. I. and Adams, J. and Aggarwal, M. M. and Ahammed, Z. and Amonett, J. and Anderson, B. D. and Anderson, M. and Arkhipkin, D. and Averichev, G. S. and Bai, Y. and Balewski, J. and Barannikova, O. and Barnby, L. S. and Baudot, J. and Bekele, S. and Belaga, V. V. and Bellingeri-Laurikainen, A. and Bellwied, R. and Benedosso, F., ... Bhardwaj, S.(2007). *Measurements of Strange Particle Production in p+p Collisions at* $\sqrt{s}$*= 200 GeV*Physical Review C, 75, 064901. http://dx.doi.org/10.1103/PhysRevC.75.064901

[2] Abgaryan, V. et al. (MPD Collaboration) *Status and initial physics performance studies of the MPD experiment at NICA.* Eur. Phys. J. A (2022) 58, 140.

[3] Adam, S.C., Adamczyk, L., Adams, J.R., Adkins, J.K., Agakishiev, G., Aggarwal, M.M., Ahammed, Z., Alekseev, I.G., Anderson, D.M., Aoyama, R., Aparin, A., Aschenauer, E.C., Ashraf, M.U., Atetalla, F., Attri, A., Averichev, G.S., Bairathi, V., Barish, K.N., Bassill, A.J.,... Zyzak, M. (2019). *Bulk properties of the system formed in Au+Au collisions at sNN =14.5 GeV at the BNL STAR detector.*Physical Review C, 101, 024905. https://api.semanticscholar.org/CorpusID:199543556

[4] Adamczyk, J. K. Adkins, G. Agakishiev, M. M. Aggarwal, Z. Ahammed, N. N. Ajitanand, I. Alekseev, D. M. Anderson, R. Aoyama, A. Aparin, D. Arkhipkin, E. C. Aschenauer, M. U. Ashraf, A. Attri, G. S. Averichev, X. Bai, V. Bairathi, A. Behera, R. BellwiedA,... Bhasin.(2017). *Bulk properties of the medium produced in relativistic heavy-ion collisions from the beam energy scan program.*Physical Review C, 96, 044904. http://dx.doi.org/10.1103/PhysRevC.96.044904

[5] CERN. (2009). (s.f.). *Measurement of the Inclusive Jet Cross Section in pp Collisions at* $\sqrt{s}$ *= 7 TeV Using the ATLAS Detector. CERN Document Server.* Retrieved April 17th, 2024 https://cds.cern.ch/record/1169055/files/ATL-PHYS-PUB-2009-009.pdf?version=3

[6] Groom,D.E. and Klein, S.R. (2019) *Passage of Particles Through Matter.* Retrieved April 18, 2024, from https://pdg.lbl.gov/2019/reviews/rpp2018-rev-passage-particles-matter.pdf

[7] Iamaldonado. (s.f.). *CoreCoronaTask.* GitHub. Retrieved April 17th, 2024, from https://github.com/iamaldonado/CoreCoronaTask

[8] Iamaldonado. (s.f.). *Macros ANA*. GitHub. Retrieved April 17th, 2024, from https://github.com/iamaldonado/Macros_ANA/blob/main/README.md

[9] Lechner, A. *Particle Interations with Matter.* CERN Document Server. Retrieved April 17th, 2024, from https://cds.cern.ch/record/2674116/files/660.pdf

[10] MPDROOT Developers. (s.f.). *MpdPairKK.cxx* . MPDROOT. Retrieved April 18th, 2024, from https://git.jinr.ru/nica/mpdroot/-/blob/v23.03.23/physics/pairKK/MpdPairKK.cxx?ref_type=tags

[11] MPDROOT Developers. (s.f.). *MpdPid.cxx* . MPDROOT. Retrieved April 18th, 2024, from https://git.jinr.ru/nica/mpdroot/-/blob/v23.03.23/core/mpdPid/MpdPid.cxx?ref_type=tags

[12] MPDROOT *Simulation and Analysis Framework for the MPD experiment of the NICA project. (s. f.).* Retrieved 18 April 2024, from https://mpdroot.jinr.ru/

[13] Multi purpose detector: the Mega-Science Project «NICA». (2024). Retrieved April 17th, 2024, from https://mpd.jinr.ru/

[14] Raghunath Sahoo. *Relativistic Kinematics.* E-print: 1604.02651 [nucl-ex] https://arxiv.org/pdf/1604.02651.pdf

[15] L. Garren (Fermilab), I.G. Knowles (Edinburgh U.), S. Navas (U. Granada), P. Richardson (Durham U.), T. Sj¨ostrand (Lund U.), and T. Trippe (LBNL).(2006). *MONTE CARLO PARTICLE NUMBERING SCHEME* Retrieved April 18th, 2024, from https://pdg.lbl.gov/2007/reviews/montecarlorpp.pdf