JOINT INSTITUTE FOR NUCLEAR RESEARCH

Veksler and Baldin laboratory of High Energy Physics

Final Report of INTEREST program

Efficiency of the Mini Beam Beam trigger detector

Supervisor

Dr. Ivonne Maldonado

Student
José Francisco Reyes

Universidad Nacional Autónoma de México

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

Abstract

This paper presents a brief explanation of the MPD experiment and its relationship with the Mini Beam Beam detector, as well as a description of the main macro used, The functions that can be added were also explained and from these the efficiency for the detector was calculated, in addition to the calculation for the estimation of the primary vertex from the exclusive use of the particle flight time the fastest particle time to reach the detector for further advances, the data used were generated in PHSD for Xe-Xe to $\sqrt{s_{NN}} = 9.2$ GeV and for the special case of muons BOX was used.

0.1 Introduction

The Multi-Purpose Detector (MPD) is one of two Nuclotron-based ion collider faility (NICA) in Dubná Russia, being part of the The Joint Institute for Nuclear Research (JINR) complex, aims to study Quark Gluon Plasma (QGP) and its phase diagram in a region not so studied in an energy range of 4 to 11 GeV. [MPD24a] [Col22]

For the MPD experiment is developing the MiniBeBe detector, which will serve as an trigger for the TOF (time of flight). The MiniBeBe is a particle detector optimized for low multiplicity events involving collisions between protons (p+p), protons and atomic nuclei (p+A), and atomic nuclei (A+A). These events, which involve a small number of particles or nucleons, offer valuable insights into particle interactions. The detector role in studying heavy ion collisions is crucial for exploring properties of matter, antimatter, and post-Big Bang universe characteristics. [MPD24b] The detector geometry has changed from its original design, notably with the center position on the rail now at -1.5cm. The MiniBeBe detector consists of 8 H-shaped rails with 20 plastic scintillators (20x20x5 mm^3) each, paired with two SiPMs on electronic cards and cooling plates. The scintillators are EJ232 similar to BC404. SiPMs belong to the J-Series with a 3.07 x 3.07 mm^2 active area, fixed between PCB and cold plates. Cold plates are like those in the ITS detector, with carbon fiber sheets embedding two cooling pipes filled with water. The primary MiniBeBe component is the rail holding 20 scintillators with two SiPMs each. For Inner Barrel (IB) which is a casing belonging to the Inner Tracking System (ITS), only 8 rails are used, requiring 160 scintillators and 320 SiPMs. The radial position of each scintillator changes from 22.3 cm to 10.1 cm in this configuration. [Mal24b]

Essentially you have to simulate what we could see with the MiniBeBe detector so that when installed on the accelerator we give a correct reading of the data and its operation is the expected. For this simulation analysis, MPDroot was used, which consists of the main operation of analysis wagons. Although only the use of the MPDRoot wagons was learned for the detector case it was not used because the software has not yet reached that point, so in the future they will have to create reconstruction classes, so for the work done here a macro was used (readDST) that reads the evetest.root files. For the efficient processing of the data that the experiment would give us. Simulations were generated with muon BOX and with PHSD for Xe-Xe to $\sqrt{S_{NN}} = 9.2$ GeV. Although there is an earlier version that is the one that is preceded in the quotes for this work has a new version with the scintillators to 1.5 closer to increase efficiency.[Mal24a]

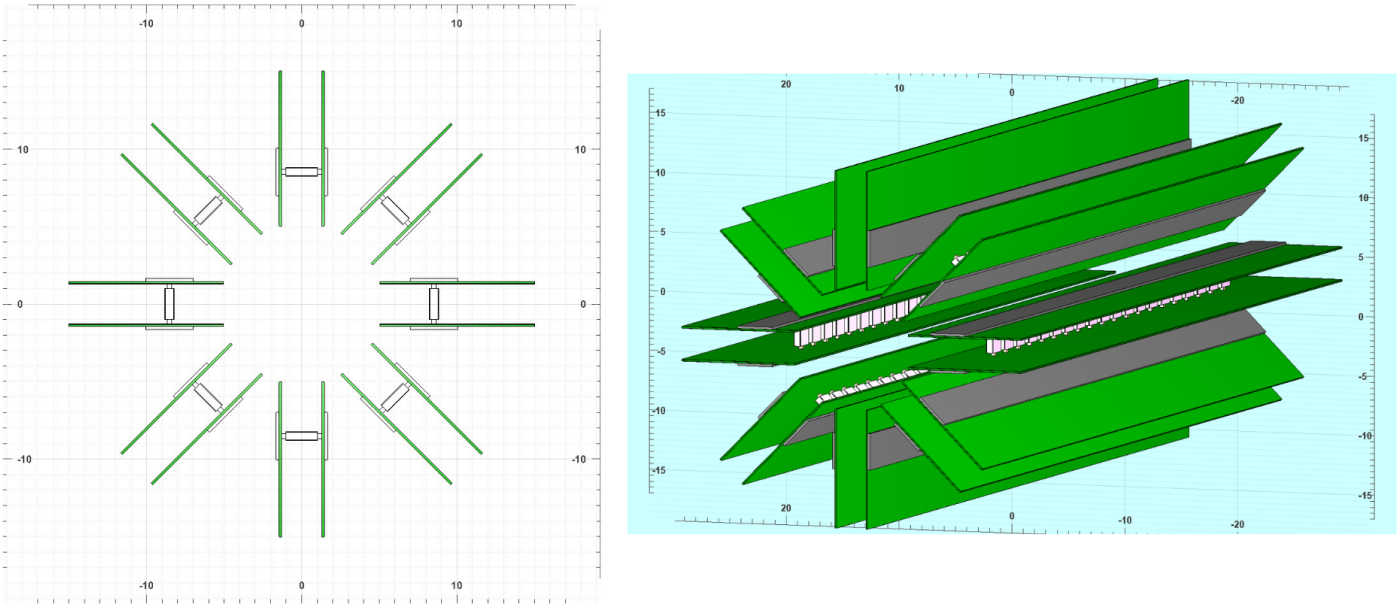


Figure 1: Detector diagram.

In figure 1 there is a representation of the detector as well as the scintillating plastics with the position corrected according to the latest update

0.2 Analysis macro

At the beginning of macro readDST with MPDRoot is in the first lines the call of the classes that are needed for use as are the use of Monte Carlo, after it is in a set of lines the beginning of a string that is used to read multiple files in a folder, saving us time and allowing us to have more information by histogram, from the above we find the difference in the saving of the processed data, such as the name that we want to save at the end of the analysis process (.root). Finally there is the difference of the histograms, each one begins with the type of histogram, that is of how many dimensions it is (TH1F for a dimension, TH2F for two dimensions and TH3F for three dimensions) after the historgram name and at the end the axes name as well as the ranges and number of bins (figure 2).

```
/* Macro reads DST file produced by macro reco.C */

#include <Rtypes.h>
#include "MbbPoint.h"
#include <TChain.h>
#include <TClonesArray.h>
#include <TStopwatch.h>



       char name[100];
       void readDST(TString in = "/storage/mexnica/ivonne/GEN/phsd/sim_XeXe9.2GeV/evetest%d.root") {
       if (in.IsNull()) {
       cout << "Please, provide an input DST-file!" << endl;
       return;
        }

       TStopwatch timer;
       timer.Start();
       TFile out("tiempo.root","recreate");



       //Histogramas

       TH1F *h1a = new TH1F("h1a","z distribution;z[cm];Entries",140,-35,35);
       TH2F *h2a = new TH2F("posicionD2","Posicion radial;x;y",100,-12,12,100,-12,12);
       TH1F *h3 = new TH1F("h3","t distribution;t(ns);Entries",400,0.02,5);
       TH2F *h4 = new TH2F("posicionD2t","Posicion x vs t;x[cm];t[ns]",100,-12,12,100,0,12);
       TH2F *h5 = new TH2F("posicionD2t2","Posicion y vs t;y[cm];t[ns]",100,-12,12,100,0,12);
       TH2F *h6 = new TH2F("posicionradial en t","Posicion Radial vs t;xy2[cm2];t[ns]",100,68,79,100,0.0,350);
       TH2F *h7 = new TH2F("pos_zt","t vs Posicion en z;t[ns];z[cm]",100,0.7,100,-35,35);
       TH2F *h8 = new TH2F("Pseudorapidezt","t vs Psudorapidez;t[ns];eta",100,0,20,100,-5,5);
       TH2F *h9 = new TH2F("posicionxz","Posicion ;x;z",100,-2,2,100,-12,12);
       TH2F *h10 = new TH2F("posicionyz","Posicion ;y;z",100,-2,2,100,-12,12);
       TH3F *h11 = new TH3F("posicionD3","Posicion x,y,z;x;y:z",100,-12,12,100,-12,12,100,-12,12);
       TH2F *h12 = new TH2F("elosz","E loss vs Posiclon en z;Eloss[MeV];z[cm]",100,-30,30,100,0,5);
       TH1F *h1 = new TH1F("h1","Energy loss distribution;E_{loss}(MeV);Entries",140,0,10);
       TH2F *h2 = new TH2F("h2","#Ez distribition;Eloss[MeV];Z[cm]",140,0,5,140,-30,30);
       TH2F *hLossEnergyvsRho_mu = new TH2F("hLossEvsRho_mu","hLossEvsRho",1000,0.,5.,2000,7.5,9.5);
       TH1F *h13 = new TH1F("h13","Primary Vertex;z(cm);Entries",140,-100,100);
       TH2F *h14 = new TH2F("h14","Primary Vertex vs Minimum time;Zvtx(cm);t(ns)",240,-35,30,100,0.0,15);
       TH1F *h15 = new TH1F("h15","Minimum time;t(ns);Entries",400,0.02,5);
       TH2F *h16 = new TH2F("h16","Primary Vertex vs Time;Zvtx(cm);t(ns)",240,-35,30,100,0.0,6);
       //TH1F *hb = new TH1F("hb","impact parameter;b(fm);Entries",160,0,16);
```

Figure 2: Start the readDST macro, defining classes and histograms and cycle for

In figure 3, on the left side is the difference of our string to read multiple files defined as a for cycle, as well as where the path of the files and how many we want to read, on the right side we have the call of data such as the Monte Carlo and the recontruded, essential for the use of the functions with which we will read the simulated data.

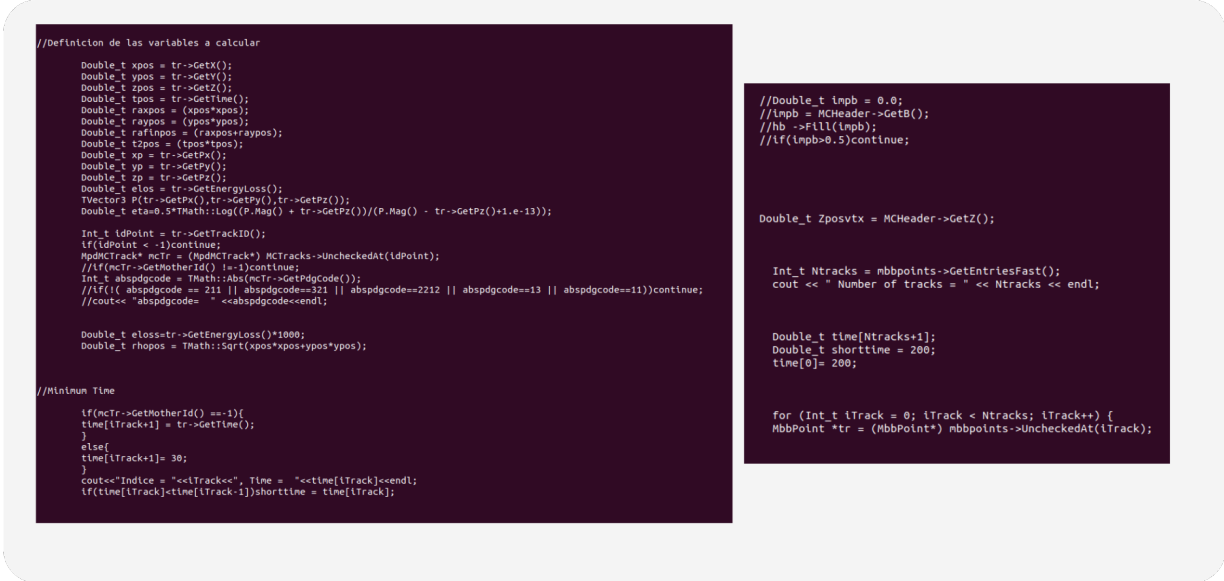
```
//Ciclo para leer multiples archivos


       for(Int_t nf=0; nf<400; nf++){
       sprintf(name," /storage/mexnica/ivonne/GEN/phsd/sim_XeXe9.2GeV/evetest%d.root",nf);
       TString inputFile = name;
       TFile fileInput(inputFile.Data());
       if (fileInput.IsZombie()) continue;
       TChain *dstTree = new TChain("mpdsim");
       dstTree->Add(inputFile.Data());
```

```
TClonesArray *MCTracks;
TClonesArray *mbbpoints;


//MpdEvent *event = nullptr;
//dstTree->SetBranchAddress("MPDEvent.", &event);
//TClonesArray *fMCTracks = nullptr;
//dstTree->SetBranchAddress("MCTrack", &fMCTracks);



MCHeader = nullptr;
MCTracks = nullptr;
mbbpoints = nullptr;

dstTree->SetBranchAddress("MCEventHeader.",&MCHeader);
dstTree->SetBranchAddress("MCTrack",&MCTracks);
dstTree->SetBranchAddress("MbbPoint",&mbbpoints);


Int_t events = dstTree->GetEntries();
cout << " Number of events in DST file = " << events << endl;

for (Int_t i = 0; i < events; i++) {
dstTree->GetEntry(i);
```

Figure 3: Difinition of the cycle for and Monte Carlo.

Now you have on the left side of fugue 4, the call of the different functions for the detector as they are the position in x, in y and in z, as well as the moment, the loss of energy and the

time of flight, At the same time, it is restricted to the exclusive use of primary and charged particles. At the end is the definition for the use of minimum time where for each file only the minimum value of time is used. On the right side we have the use of tracks and the definition of variables for minimum time.



```
//Definicion de las variables a calcular

        Double_t xpos = tr->GetX();
        Double_t ypos = tr->GetY();
        Double_t zpos = tr->GetZ();
        Double_t tpos = tr->GetTime();
        Double_t raxpos = (xpos*xpos);
        Double_t raypos = (ypos*ypos);
        Double_t rafinpos = (raxpos+raypos);
        Double_t t2pos = (tpos*tpos);
        Double_t xp = tr->GetPx();
        Double_t yp = tr->GetPy();
        Double_t zp = tr->GetPz();
        Double_t elos = tr->GetEnergyLoss();
        TVector3 P(tr->GetPx(),tr->GetPy(),tr->GetPz());
        Double_t eta=0.5*TMath::Log((P.Mag() + tr->GetPz())/(P.Mag() - tr->GetPz()+1.e-13));

        Int_t idPoint = tr->GetTrackID();
        if(idPoint < -1)continue;
        MpdMCTrack* mcTr = (MpdMCTrack*) MCTracks->UncheckedAt(idPoint);
        //if(mcTr->GetMotherId() !=-1)continue;
        Int_t abspdgcode = TMath::Abs(mcTr->GetPdgCode());
        //if(!( abspdgcode == 211 || abspdgcode==321 || abspdgcode==2212 || abspdgcode==13 || abspdgcode==11))continue;
        //cout<< "abspdgcode=  " <<abspdgcode<<endl;


        Double_t eloss=tr->GetEnergyLoss()*1000;
        Double_t rhopos = TMath::Sqrt(xpos*xpos+ypos*ypos);


//Minimun Time

        if(mcTr->GetMotherId() ==-1){
        time[iTrack+1] = tr->GetTime();
        }
        else{
        time[iTrack+1]= 30;
        }
        cout<<"Indice = "<<iTrack<<", Time =  "<<time[iTrack]<<endl;
        if(time[iTrack]<time[iTrack-1])shorttime = time[iTrack];
```

```
//Double_t impb = 0.0;
//impb = MCHeader->GetB();
//hb ->Fill(impb);
//if(impb>0.5)continue;



Double_t Zposvtx = MCHeader->GetZ();


  Int_t Ntracks = mbbpoints->GetEntriesFast();
  cout << " Number of tracks = " << Ntracks << endl;


  Double_t time[Ntracks+1];
  Double_t shorttime = 200;
  time[0]= 200;


  for (Int_t iTrack = 0; iTrack < Ntracks; iTrack++) {
  MbbPoint *tr = (MbbPoint*) mbbpoints->UncheckedAt(iTrack);
```

Figure 4: Definition of functions to call simulations data.

Now the filling of the histograms is defined, starting with the name of the histogram defined in figure 2, filled with the name assigned to each function. On the right side is the key that closes the loops and the difference of the minimum time, as well as two last histograms, at the end is the closing of the macro (figure 5).

```
hLossEnergyvsRho_mu->Fill(eloss,rhopos);
if (rhopos > 8.33) continue;
h1->Fill(eloss);
h2->Fill(eloss,zpos);
h1a->Fill(zpos);
h2a->Fill(xpos,ypos);
h3->Fill(tpos);
h4->Fill(xpos,tpos);
h5->Fill(ypos,tpos);
h6->Fill(rafinpos,t2pos);
h7->Fill(tpos,zpos);
h8->Fill(tpos,eta);
h9->Fill(xpos,zpos);
h10->Fill(ypos,zpos);
h11->Fill(xpos,ypos,zpos);
h12->Fill(zpos,elos);
h13->Fill(Zposvtx);
h16->Fill(Zposvtx,tpos);
```

```
} // track loop


cout << " numero de traks = "  << Ntracks   << " El valor minimo es: " << shorttime << endl;
double* minElement = std::min_element(time, time+Ntracks);
std::cout << "El valor minimo es: " << *minElement << std::endl;



h14->Fill(Zposvtx,*minElement);
h15->Fill(*minElement);




} // event loop
}//files loop
out.Write();
out.Close();
timer.Print();

cout << "Macro finished successfully." << endl;  // marker of successful execution for Gitlab test framework
cout << " Test passed" << endl;
cout << " All ok " << endl;
}
```

Figure 5: Filling histograms and closing loops.

The different functions were taken from the website of the following figure, such as positions, moments, loss of energy and flight time (figure 6).[Fai24]

Figure 6: Function definition page for MiniBeBe

0.3 Results

To begin we obtained histograms of the different positions as well as x, y and z, having a histogram for the projection of x vs z, one for y vs z, one for radial position (x vs y) and a last of our 3 positions in 3D. This first analysis is of Xe-Xe collisions at $\sqrt{S_{NN}} = 9.2$ GeV This graph shows the position of the different scintillating plastics arranged in the form of rings such as those of the radial position in Figure 7 and how these rings are along the z axis as is the 3D histogram in Figure 7.

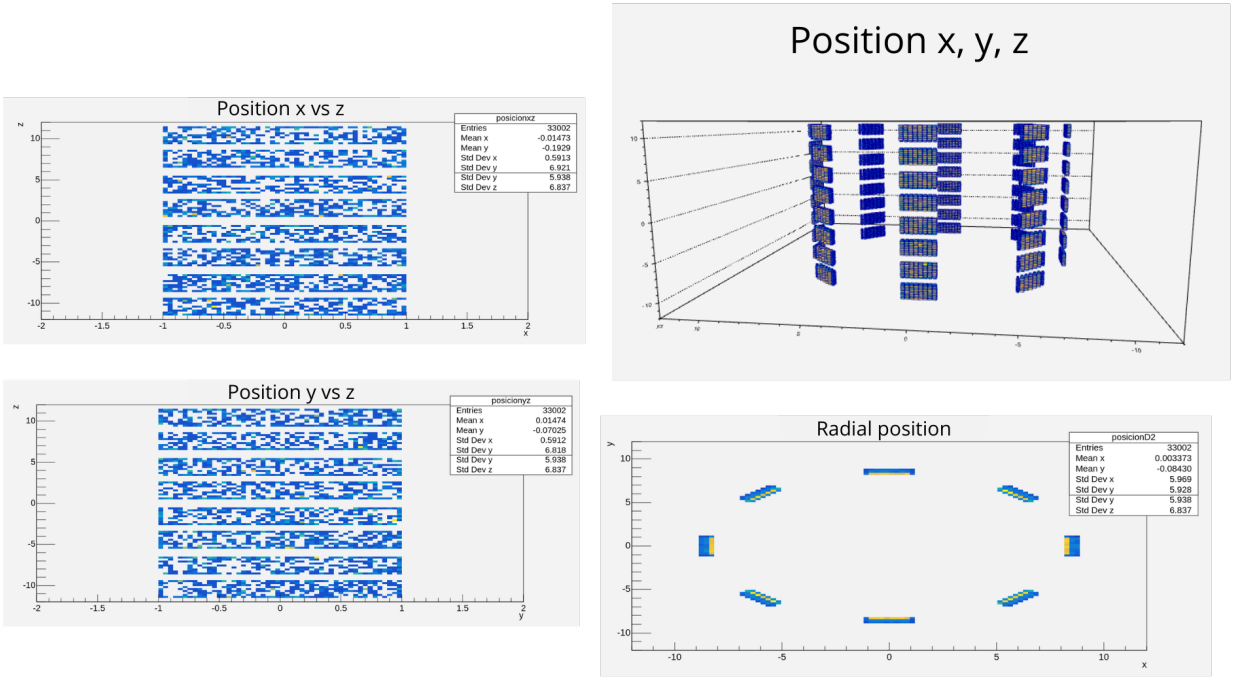


Figure 7: Histograms of the multiple axes.

The functions for Mini Beam Beam also allow us to create histograms in 1D and apart from the physical positions allows us to introduce time, over time also allows us to perform an analysis regarding the position in x and y. These new histograms as well as the position in z are shown in Figure 8.

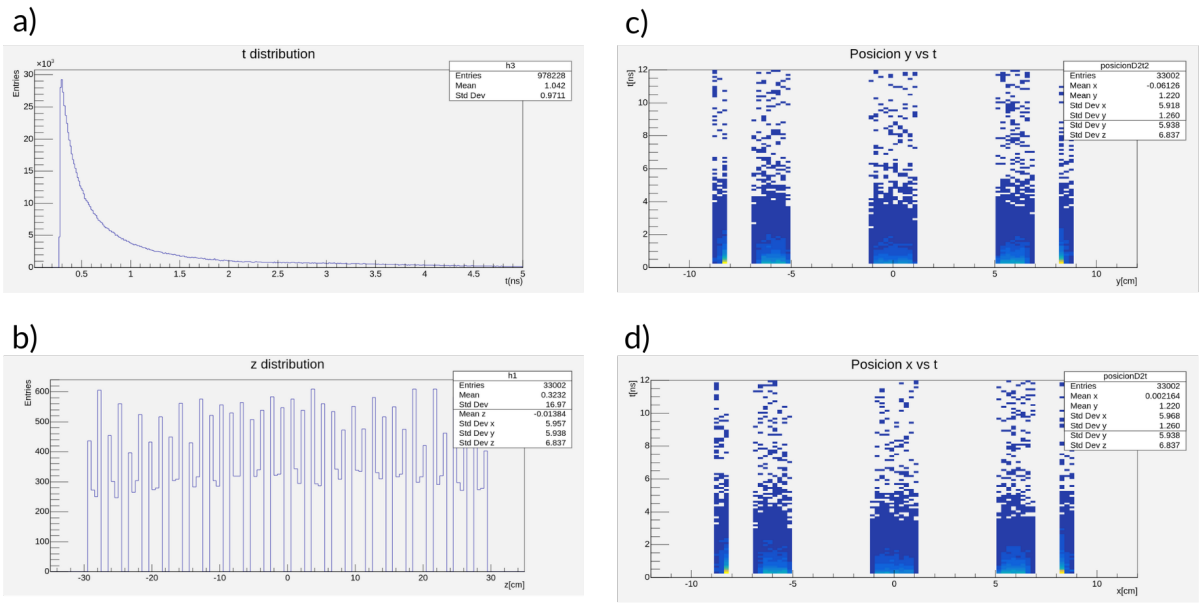


Figure 8: Time histograms with x and y and z 1D position.

The use of the time variable allows us to "play" a little with the histograms, including the moment you can get the psudorapidity as exercise (figure 9 b)). Also as flight time vs z position and radial position.

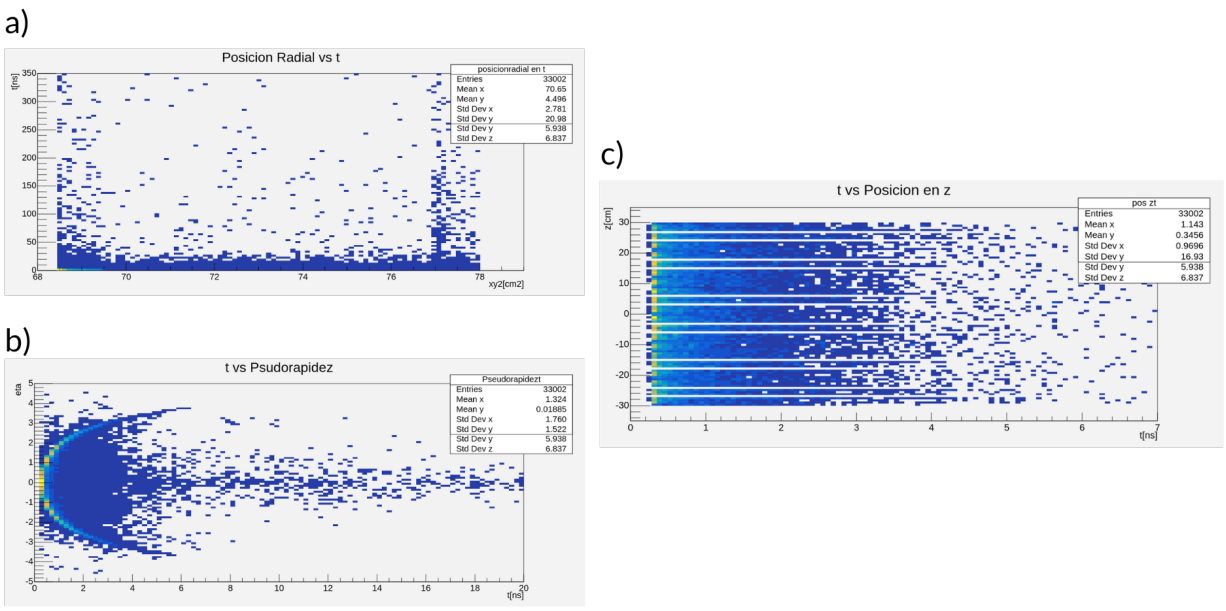


Figure 9: Flight time as pseudorapidity function, z position and radial position.

Once you become familiar with the functions of started working on the calculation for the efficiency of the detector starting with obtaining the loss energy, For this point it is essential

the use of the Monte Carlo to ask the restriction of particulates and to have a strict analysis of the are of our interest. In Figure 10 we have the distribution of energy loss for muons generated by BOX.

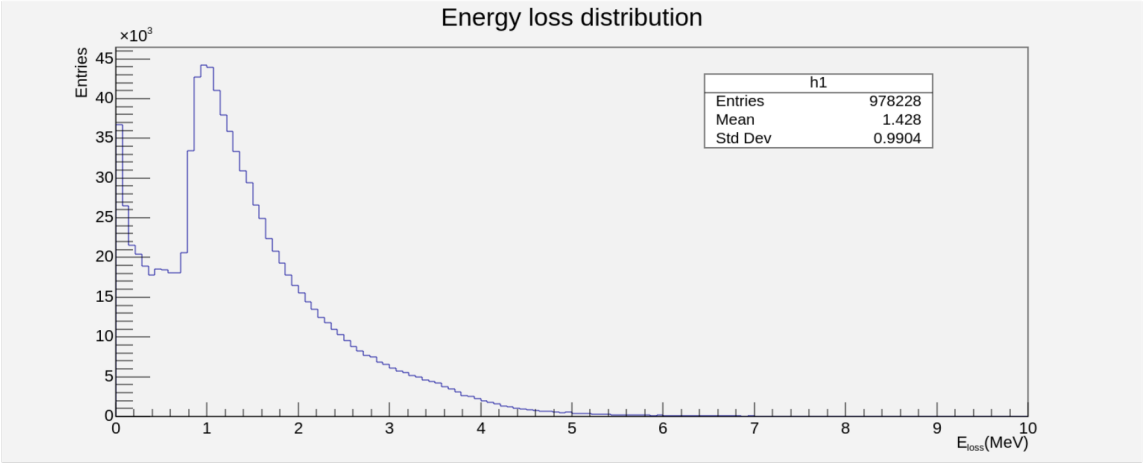


Figure 10: Loss energy for muons.

With the use of energy loss can be implemented to a histogram as position function in z. The same procedure was performed for pions but these generated with PHSD to an energy of $\sqrt{S_{NN}} = 9.2$ GeV (Figurate 11).

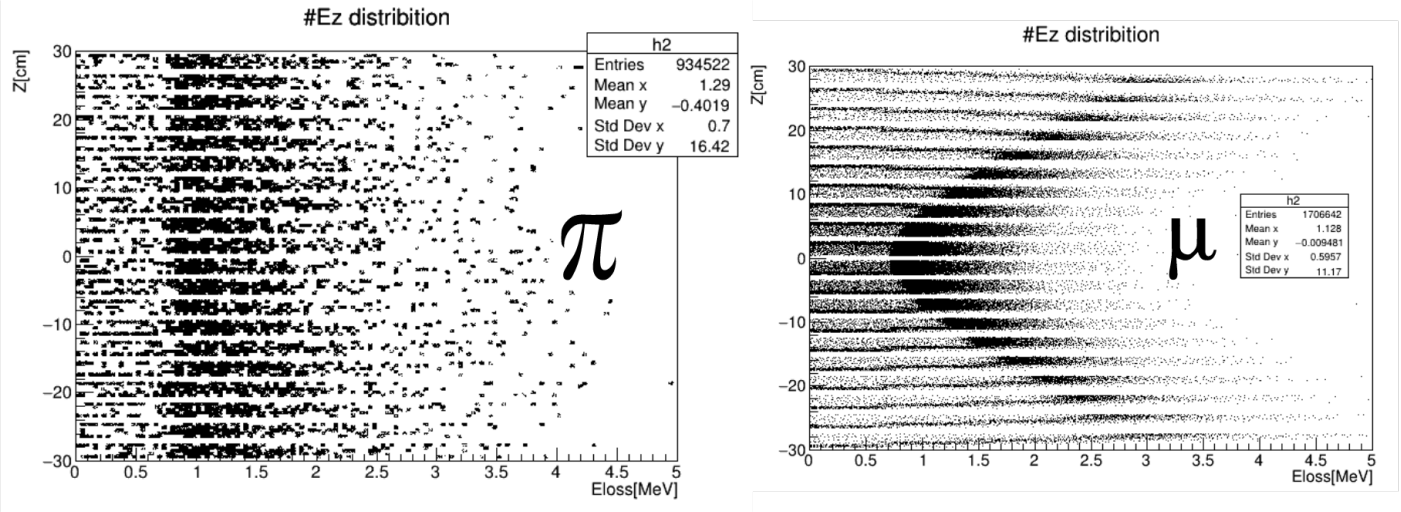


Figure 11: Loss energy as z position function.

The intention of having the histograms in figure 11 is to obtain the bins of each colored strip and make some cuts to determine the loss of energy for what would be each ring of the detector, as at the beginning you have a loss of energy with a lot of noise was restricted the range where it is analyzed given a rho that represents our plastic and analyzing only the part that interests us where the particles will stick. For the case of the analysis of the pions the impact parameter was restricted to others having a centrality of 00-05.

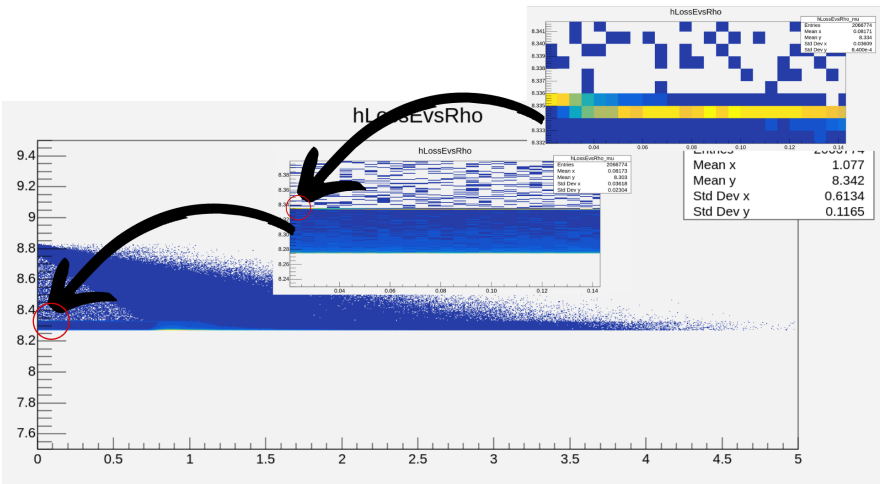


Figure 12: Shows the restriction of our Rho.

Now cleaner our signal for vines allows us to make the corresponding cuts, in figure 13 cuts are presented for muons since they are particles that do not interact directly with our

signal of the plastics, the selection of these particles will allow us to obtain a sum of all these loss energy cuts to give us the value of the probability of having particles with a certain amount of deposited energy.

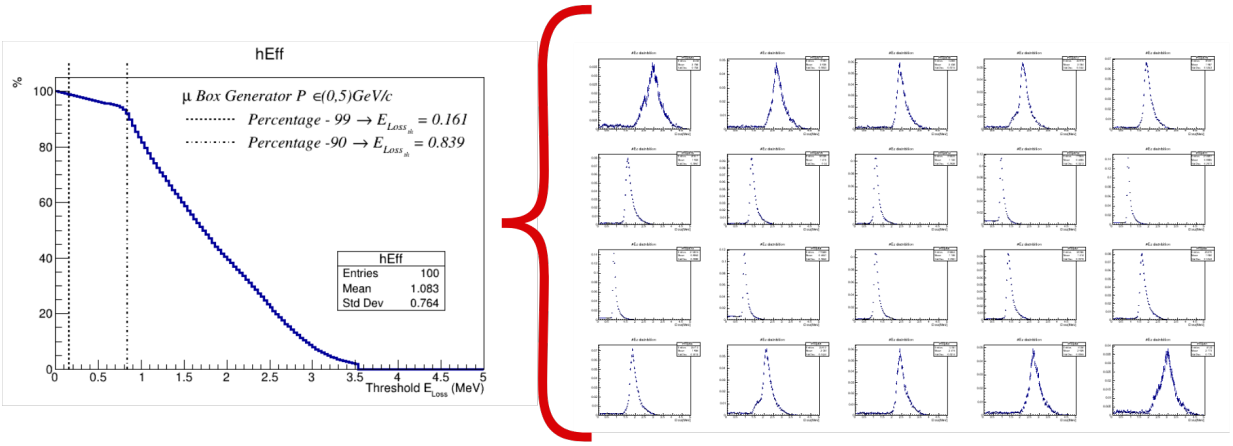


Figure 13: Energy loss per ring and average cuts

With the probability value of having particles with certain deposit energy we can continue with the next step which is to have for each ring the probability of having a hit. With our hits we can normalize them to the value of the impact parameter (Figure 14 d)), each series of normalized hits give us a series of points that we can notice in the (figure 14 e)). This normalization can be performed for charged particles such as muons, pions, etc. (figure a), b,) c)) including also the set of all charged particles.

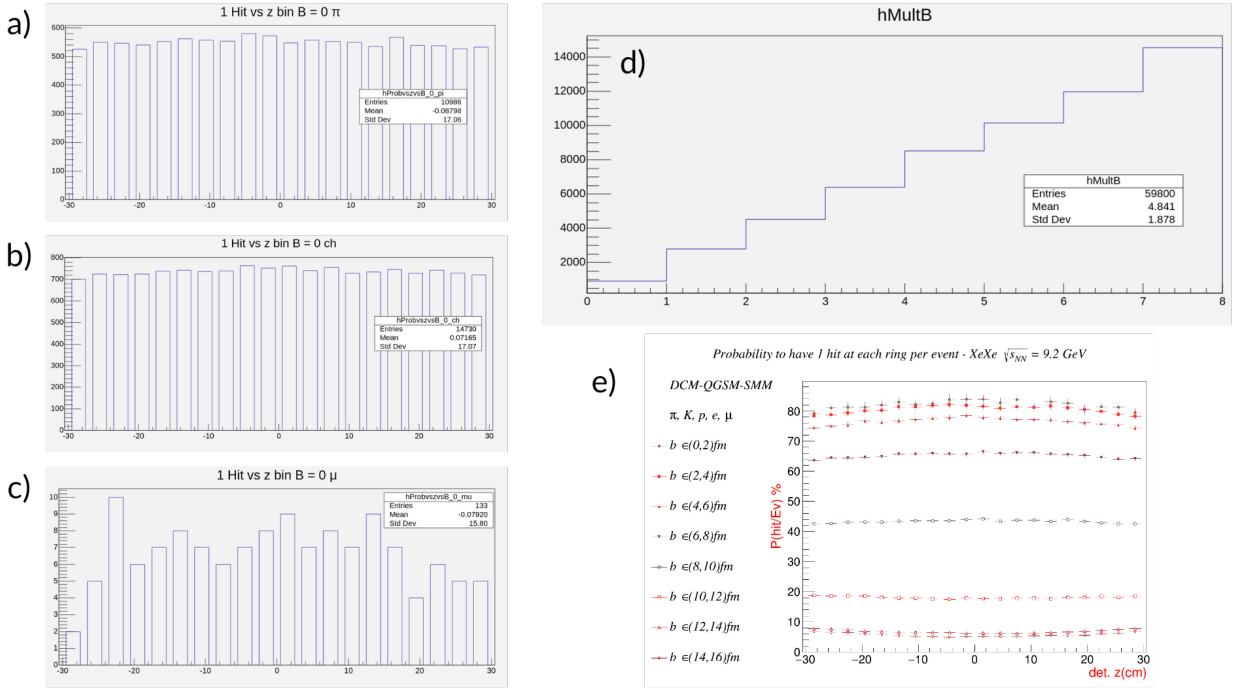


Figure 14: Hit per event of muons, pions and charged particles, impact parameter and normalised probability to impact parameter

With our normalized data for each ring we can apply equation for each value of the impact parameter having a last point which is the trigger efficiency.

$$Eff = 1 - (1 - P_1)(1 - P_2)(1 - P_3)...(1 - P_{20})$$

The efficiency for protons, pions and charged particles was calculated, resulting in figure 15 where the expected behavior is observed.[Mal24b]

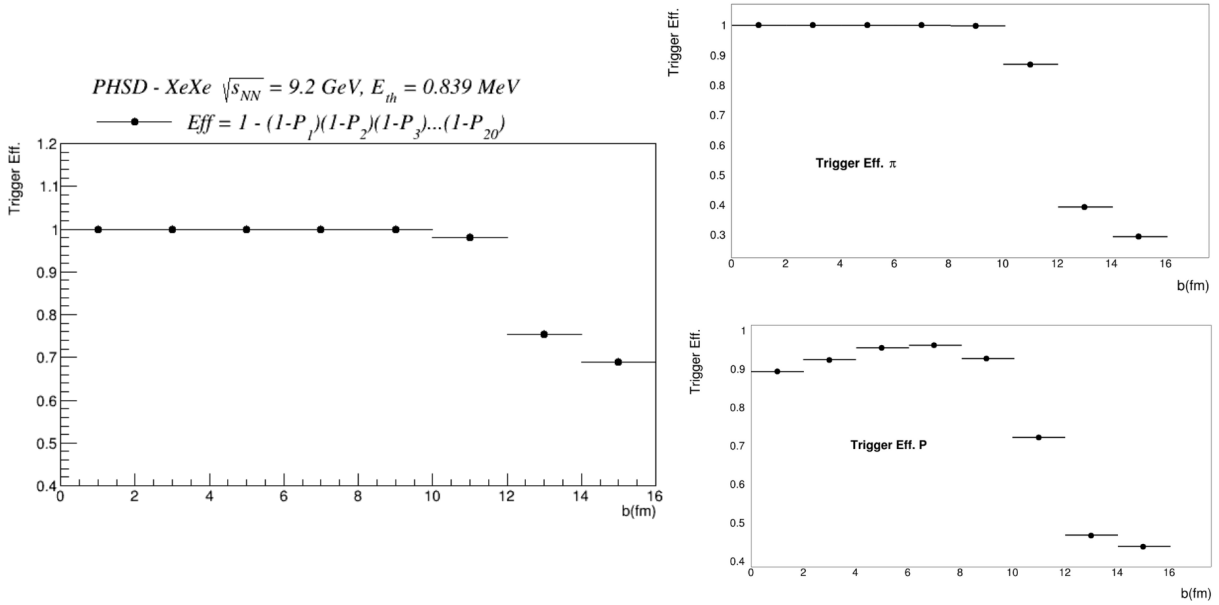


Figure 15: Trigger efficiency for different particles

Since the detector can only possibly measure the flight time of the particles, it was proposed to calculate the minimum time of the particles to see if it is possible to know the primary vertex apart from the single use of time. In the first instance, the time, the minimum time and the primary vertex were calculated and each variable mentioned in figure 16 was plotted in a histogram of 1 dimension

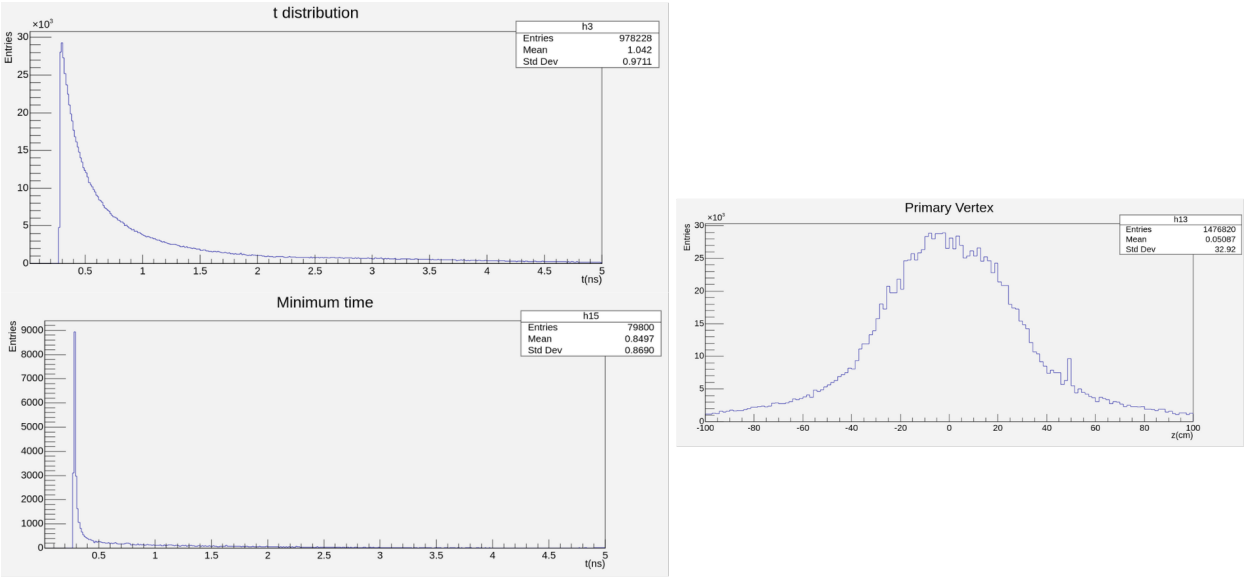


Figure 16: Time, primary vertex and minimum time

Having our different variables we can have histograms for both time and minimal time, these are in Figure 17 where the time and minimal time is plotted as function of the primary vertex and the impact parameter.

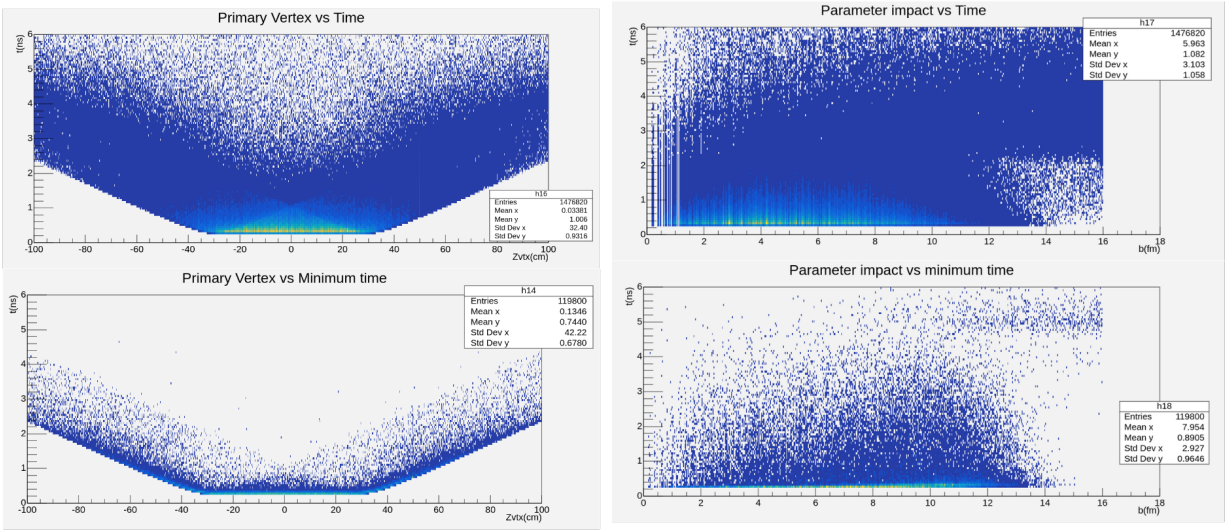


Figure 17: Minimal time and time as primary vertex function and impact parameter.

The development of these last histograms allows us to make comparisons and gives us a better understanding of our system to determine if it is possible to know the primary vertex from the time that is the last task that is posed.

0.4 Discussion and conclusions

It was possible to obtain basic histograms of the different functions for the MiniBeBe detector, supported by the mpdroot software and mainly the readDST macro, with the basic functions of positions in the different axes was able to see the ring of the scintillating plastics, in addition to the plastic ring cylinder. Subsequently results were obtained with loss of energy for cuts in z, was used for the analysis of the loss energy in z a new macro created from scratch, which used to generate power cuts for each band in z for muons, with this an average was obtained, which calculates the probability of having a particle with an amount of depositing energy, restricted to a radial position with respect the beam axis "Rho" to eliminate noise, the same procedure was tried to replicate for pions, but no reliable results were obtained, because there was a lot of noise in the cuts, this prevented the necessary value to be obtained, to eliminate part of the noise the impact parameter was restricted to the primary vertex, in this way it would have to reduce the noise, leaving a new problem that being very restricted, the amount of data analyzed was insufficient, so that it was not possible to have a good distribution for its analysis, even with this limitation one mbbpoint was obtained for each ring normalized to the values of the impact parameter for each point in z, which allowed us to have the result of the trigger efficiency, finding what was expected from the start, the data used were generated for Xe-Xe a $\sqrt{S_{NN}} = 9.2$ with PHSD for charged particles, For the case of muon analysis BOX was used as naturally there is not much production of these in collitions.

As last task it was identified if it was possible to determine the primary vertex apart from the exclusive use of time since the detector is limited in the measurement of variables so it was estimated the minimal time and histograms of the general time were performed and minimum time as function of the primary vertex and the impact parameter for an analysis of the relationship that exists between these variables observing how in the minimum time there is a definition more pressure of the data allowing us to see that in principle to exist this relationship is possible to determine the primary vertex and with the use of minimal time we can have a more accurate estimate by having a cleaner resonance.

Although further work was done on determining the primary vertex with the minimum time difference in the z position.

Bibliography

[Col22] The MPD Collaboration. Status and initial physics performance studies of the mpd experiment at nica). *Eur. Phys*, 50:58–140, 2022.

[Fai24] FairMCPoint Class Reference. https://fairrootgroup.github.io/fairroot/classfairmcpoint.html, 2024.

[Mal24a] Ivonne Maldonado. Mpdroot software introduction, what is it? what you can do with it?). 21, 2024.

[Mal24b] Ivonne Maldonado. Report: Minibebe trigger efficiency. 15, 2024.

[MPD24a] MPDroot Team. https://mpd.jinr.ru/, 2024.

[MPD24b] MPDroot Team. `https://mpdroot.jinr.ru/running-mpdroot-locally-using-alibuild/`, 2024.