

PID with different detectors

3rd Computing/Analysis Workshop of the MexNICA
Collaboration

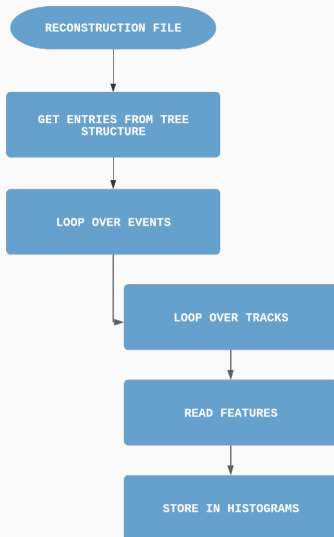
Julio César Maldonado González

03 de febrero de 2021



- La **reconstrucción de eventos** consiste en encontrar las trazas de las partículas usando técnicas de reconocimiento de patrones como el filtrado de Kalman.
- El **análisis de la física** consiste en encontrar el PID a partir de señales observables de los detectores a través de los datos de la reconstrucción.

Un breve esquema de un macro para leer los archivos de reconstrucción:



- Detector: MPD (TPC, TOF)
- Input file: rectestf140.root file (DST)
- Generador de eventos: UrQMD
- Bi-Bi a 11 GeV (MB)
- de eventos: 10k
- Macro base: CompareSpectra.C, anaDST.C

Leer archivos de reconstrucción

Leemos los archivos de la reconstrucción y las ramas del árbol *.root* con la información de los eventos y trazas:

```
for (Int_t nf = 0; nf <1; nf++){          // files name rectestf14#.root, from nf to nf++

    // Input files
    sprintf(name, "/home/julio/Research/Taller3/12_PID-with-different-detectors/rectestf14%d.root", nf);
    cout << " Opening file " << name << endl;
    TString fileName = name;

    // Define Chain to tree mpdsim
    TChain *mpdSim = new TChain("mpdsim");
    mpdSim->Add(fileName);    //put the filename

    // Set branches
    MpEvent *mpdEvents = NULL;
    mpdSim->SetBranchAddress("MPDEvent.", &mpdEvents);
    TBranch *DSTBranch = mpdSim->GetBranch("MPDEvent.");

    TClonesArray *mcTracks = NULL;
    mpdSim->SetBranchAddress("MCTrack", &mcTracks);
    TBranch *MCBranch = mpdSim->GetBranch("MCTrack");
```

```

////////////////////////////////////
// MC tracks loop

Int_t mcNtracks = mcTracks->GetEntries();

//cout << " *** Event # " << i+1 << " No. of MC tracks = " << mcNtracks << endl;

for (Int_t j = 0; j < mcNtracks; j++){

    Ntr++;

    MpdMCTrack *mctrack = (MpdMCTrack*)mcTracks->At(j);

    MotherID = mctrack->GetMotherId();
    PDGID = mctrack->GetPdgCode();

    PtMC = mctrack->GetPt();
    PtMC = TMath::Abs(PtMC);

    Pxyz.SetXYZ(mctrack->GetPx(),mctrack->GetPy(),mctrack->GetPz());
    EtaMC = Pxyz.PseudoRapidity();

    NMC++;

    hEtaMC->Fill(EtaMC);
    hPtMC->Fill(PtMC);

} //end MC track loop

```

```

////////////////////////////////////
// MPD tracks loop

TClonesArray *mpdTracks = mpdEvents->GetGlobalTracks();
Int_t mpdNtracks = mpdTracks->GetEntriesFast();

for (Int_t k = 0; k < mpdNtracks; k++){

    Ntrmpd++;

    MpdtTrack *mpdttrack = (MpdtTrack*)mpdTracks->UncheckedAt(k);

    ID = mpdttrack->GetID();

    MpdtMCTrack *mctrack1 = (MpdtMCTrack*)mcTracks->UncheckedAt(ID);

    MotherID = mctrack1->GetMotherId();

    Pt = mpdttrack->GetPt();
    Pt = TMath::Abs(Pt);

    Eta = mpdttrack->GetEta();

    N++;

    hEta->Fill(Eta);
    hPt->Fill(Pt);
}

```

```
Px = mpdtrack->GetPx();  
Py = mpdtrack->GetPy();  
Pz = mpdtrack->GetPz();  
  
P = TMath::Sqrt(Pz*Pz + Px*Px + Py*Py);  
  
dEdx = mpdtrack->GetdEdXTPC();  
  
hEta->Fill(Eta);  
hPt->Fill(Pt);  
  
hdEdx->Fill(dEdx);  
  
hdEdxP->Fill(P,dEdx);  
  
} // MPD track loop  
  
hN->Fill(N);
```


La probabilidad de una partícula i , si se observa una señal s ,

$$P(i|s) = \frac{r(s|i)C_i}{\sum_k r(s|k)C_k}$$

$r(s|i)$ función densidad de probabilidad condicional de la señal observada s de un detector, si una partícula $i(e, \mu, \pi, K, p, \dots)$ es detectada. Esta refleja las propiedades del detector.

C_i es la probabilidad de encontrar un tipo de partícula en el detector (frecuencia).

Utilizamos la información de las trazas del Monte Carlo (*PDGID*) para

```
if (TMath::Abs(PDGID) == 211 ){ //pion

    Npi++;

    hpiEta->Fill(Eta);
    hpiPt->Fill(Pt);

    Px = mpdtrack->GetPx();
    Py = mpdtrack->GetPy();
    Pz = mpdtrack->GetPz();
    P = TMath::Sqrt(Pz*Pz + Px*Px + Py*Py);

    dEdx = mpdtrack->GetdEdXTPC();
    hpidEdx->Fill(dEdx);
    hpidEdxP->Fill(P,dEdx);

    hpiEtaMC->Fill(EtaMC);
    hpiPtMC->Fill(PtMC);

    M2 = mpdtrack->GetTofMass2();
    hpiM2->Fill(M2);

}
```

Histograma de dE/dx y dE/dx vs P para piones

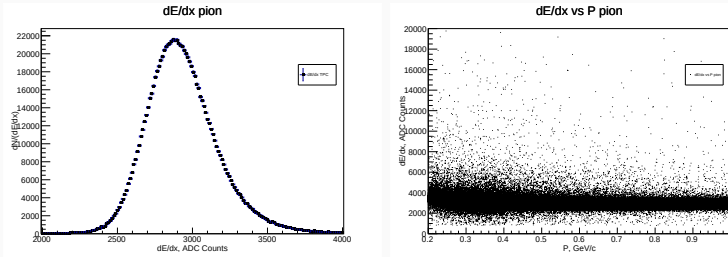


Figure 1: dE/dx en TPC para 10 mil eventos tomando partículas primarias y secundarias.

Ajuste al histograma dE/dx

```
//pion  
  
Double_t parpi[6];  
  
TF1* fitgausspi = new TF1("fitgausspi","gaus(0)",1000.,2900.00);  
TF1* fitlandaupi = new TF1("fitlandaupi","landau(0)",2900.00,5000.);  
TF1* fittotalpi = new TF1("fittotalpi","gaus(0)+landau(3)",1000.,5000.);  
  
fittotalpi->SetLineColor(2);  
  
hpidEdx->Fit(fitgausspi,"R");  
hpidEdx->Fit(fitlandaupi,"R+");  
  
fitgausspi->GetParameters(&parpi[0]);  
fitlandaupi->GetParameters(&parpi[3]);  
  
fittotalpi->SetParameters(parpi);  
hpidEdx->Fit(fittotalpi,"R");
```

Histograma de dE/dx y ajuste

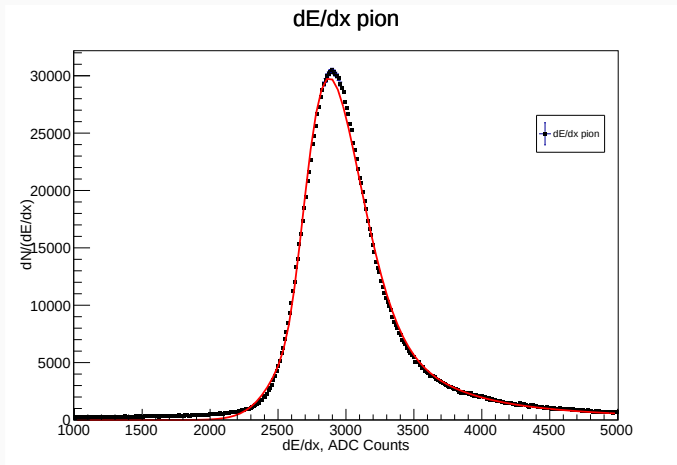


Figure 2: dE/dx en TPC para 10 mil eventos tomando partículas primarias y secundarias.

Probabilidades a priori con medidas del TOF

