Roteiro de Atividades do Doutorado

1º Semestre de 2018

André Vieira

19 de Maio de 2018

Resumo

Na busca por uma compreensão dos eventos de raios cósmicos em altas energias, estamos interessados em usar o ferramental computacional fornecido pelos geradores PYTHIA e o EPOS-LHC, com o objetivo de simular processos físicos elementares que podem ocorrer em colisões em altas energias na atmosfera.

Conteúdo

1 PYTHIA 8226 [1]

Esses programas foram rodados no linux ubuntu:

a. Download o PYTHIA 8226 no link, ou usando linhas de comando no ubuntu no terminal:

```
wget http://home.thep.lu.se/~torbjorn/pythia8/pythia8226.tgz
tar -zxf pythia8226.tgz
cd pythia8226
./configure --enable-shared --with-root=$ROOTSYS
sudo make -j4 && sudo make install
```

NOTE: ./configure tem algumas flag --with-root"chama as bibliotecas do ROOT

b. Na pasta examples:

```
~/pythia8226/examples$ ls
main92.cc
$ make main92
$ ./main92
```

1.1 PYTHIA 8226: main92.cc

```
// main92.cc is a part of the PYTHIA event generator.
// Copyright (C) 2017 Torbjorn Sjostrand.
// PYTHIA is licenced under the GNU GPL version 2,
// see COPYING for details.
// Please respect the MCnet Guidelines, see GUIDELINES for details.

// This is a simple test program.
// Modified by Rene Brun and Axel Naumann to put the Pythia::event
// into a TTree.

// Header file to access Pythia 8 program elements.
#include "Pythia8/Pythia.h"

// ROOT, for saving Pythia events as trees in a file.
#include "TTree.h"
#include "TFile.h"

#include "TMath.h"
#include "sstream"

using namespace Pythia8;
```

```
int main() {
    // Create Pythia instance and set it up
    // to generate soft/hard QCD processes
    Pythia pythia;

    //Precision:
    cout<<fixed;
    cout<<setprecision(5);

    //define variables:
    Double_t px, py, pz, pt, e, Eta, Phi;
    Double_t gama, beta, bx,by,bz, Energy,somaE=0, somaM=0, P=0;
    Double_t somapx=0, somapy=0, somapz=0, somae=0, E1[90];
    int cont=0, id, Nev, Nparticles, aux =0;

    //Number of events:
    Nev = 10;

    pythia.readString("Beams:idA=2212");// Proton Beam
    pythia.readString("Beams:idB=2212");// Proton Beam

    //Processes QCD:

    //Hard
    //pythia.readString("hardQCD:all=on");

    //pythia.readString("hardQCD:gg2gg = on");
    //pythia.readString("hardQCD:gg2qqbar = on");
    //pythia.readString("hardQCD:qg2qg = on");
    //pythia.readString("hardQCD:qq2qq = on");
    //pythia.readString("hardQCD:qqbar2gg = on");
    //pythia.readString("hardQCD:qqbar2qqbarNew = on");
    //pythia.readString("hardQCD:gg2ccbar = on");
    //pythia.readString("hardQCD:qqbar2ccbar = on");
    //pythia.readString("hardQCD:gg2bbbar = on");
    //pythia.readString("hardQCD:qqbar2bbbar = on");

    // Set up the ROOT TFile and TTree.
    TFile *file = TFile::Open("softQCD.root","recreate");

    //TTRee " T " with the variable of the event
    TTree *T = new TTree("T","Tree");

    //TTRee " T1 " number of events
    TTree *T1 = new TTree("T1","Tree1");

    //Soft
```

```
    pythia.readString("softQCD:all=on");
    //pythia.readString("SoftQCD:nonDiffractive = on");
    //pythia.readString("softQCD:elastic = on");
    //pythia.readString("softQCD:singleDiffractive = on");
    //pythia.readString("softQCD:doubleDiffractive = on");

    //pythia.readString("24:mayDecay = off");
    //pythia.readString("PhaseSpace:pTHatMin = 20.");
    //pythia.readString("PartonLevel:MPI = off");
    //pythia.readString("PhaseSpace:pTHatMax = 20.");
    //pythia.readString("Beams:eCM = 433.19");

    pythia.readString("Beams:frameType=2"); // frame LAB
    pythia.readString("Beams:eA=130000");   // GeV
    pythia.readString("Beams:eB=0.");       // p at rest frame
    pythia.init();


    //The Branchs of the TTree "T".
    T->Branch("px",&px);
    T->Branch("py",&py);
    T->Branch("pz",&pz);
    T->Branch("pt",&pt);
    T->Branch("e",&e);
    T->Branch("id",&id);
    T->Branch("Eta",&Eta);
    T->Branch("Phi",&Phi);

    //The Branchs of the TTree "T1".
    T1->Branch("gama",&gama);
    T1->Branch("bx",&bx);
    T1->Branch("by",&by);
    T1->Branch("bz",&bz);
    T1->Branch("Nparticles",&Nparticles);

    // Begin event loop. Generate event; skip if generation aborted.
    for (int iEvent = 0; iEvent < Nev; ++iEvent) {

        if (!pythia.next()) continue;
        for (int i = 0; i < pythia.event.size(); ++i) {


        // Fill the pythia event into the TTree.
        // Warning: the files will rapidly become large if all events
        // are saved. In some cases it may be convenient to do some
        // processing of events and only save those that appear
        // interesting for future analyses.
```

```
// Particle in the Final State (POSITIVE).
// Are not included PARTONS!
if(pythia.event[i].isFinal() > 0 && pythia.event[i].e() > 500){

   //Variables of the particles
   px = pythia.event[i].px();
   py = pythia.event[i].py();
   pz = pythia.event[i].pz();
   pt = pythia.event[i].pT();
   e  = pythia.event[i].e();
   id = pythia.event[i].id();
   Eta = pythia.event[i].eta();
   Phi = pythia.event[i].phi();

   //total momentum of the particle i
   somapx = somapx + px;
   somapy = somapy + py;
   somapz = somapz + pz;
   somae = somae + e;

   //Fill the TTRee "T"
   T->Fill();

   //This count the number of particle in the event.
   cont++;

   aux =1;
   }

   // End event loop.
   }

   if (aux == 1){
       //betax, betay, betaz.
       bx = somapx/somae;
       by = somapy/somae;
       bz = somapz/somae;

       //total moment calculation
       P = sqrt(pow(somapx,2)+pow(somapy,2)+pow(somapz,2));

       //speed parameter:
       beta = P/somae;

       //Lorentz Factor
       gama = pow(1-(beta*beta),-0.5);
```

```
            Nparticles = cont;
            //Fill the TTRee "T1"
            T1->Fill();

            aux=0;
        }
        //zerar the variables
        cont = somaE = somae = somapx = 0;
        somapy = somapz = somaM = P = 0;
    }

    // Statistics on event generation.
    pythia.stat();

    //  Write tree.
    // Print the screen (terminal) the information about the TTRee T
    //T->Print();
    T->Write();
    // Print the screen (terminal) the information about the TTRee T1
    //T1->Print();
    T1->Write();
    delete file;

    // Well Done!
    return 0;

}
```

1.2 PYTHIA: Output

```
:~/pythia8226/examples$ sudo make main92
g++ main92.cc main92.so -o main92 -w -I/opt/root6/include
-I../include -O2 -std=c++98 -pedantic -W -Wall -Wshadow
-fPIC -L../lib -Wl,-rpath,../lib -lpythia8 -ldl \
 `root-config --cflags` -Wl,-rpath,./\
 -Wl,-rpath,/opt/root6/lib `/opt/root6/bin/root-config --glibs`
:~/pythia8226/examples$ sudo ./main92
```

1.3 main92.cc: Output softQCD.root

```
~/pythia8226/examples$ rootls -t softQCD.root
TTree  May 19 09:13 2018 T   "Tree"
  px   "px/D"   1330
  py   "py/D"   1330
  pz   "pz/D"   1330
```

```
  pt    "pt/D"    1330
  e     "e/D"     1329
  id    "id/I"    698
  Eta   "Eta/D"   1331
  Phi   "Phi/D"   1331
TTree   May 19 09:13 2018 T1   "Tree1"
  gama          "gama/D"          149
  bx            "bx/D"            147
  by            "by/D"            147
  bz            "bz/D"            147
  Nparticles    "Nparticles/I"    115
```

1.4 Macro para ler o arquivo: softQCD.root

Essa macro chamada **EventosCBJ1.C** contém um script para comparar os dados experimentais da CBJ (disponíveis no link no github (LAB e CM)) e a simulação gerada pelo PYTHIA (pp).

```
//  ---------- macro: Eventos_CBJ1.C ------------------

#include "TCanvas.h"
#include "TStyle.h"
#include "TH1.h"
#include "TGaxis.h"
#include "TLatex.h"
#include "TROOT.h"
#include "TGraphErrors.h"
#include "TF1.h"
#include "TLegend.h"
#include "TArrow.h"
#include <math.h>
#include "readevento2.C"
#include "WriteTxt.C"
#include "TFile.h"
#include "TTree.h"


//gStyle->SetOptStat(000000); //NOT put subtitles on the graphics.


//  Transformacao de Lorentz (geral em todas as direcoes)...

Double_t* BOOST(Double_t betax,Double_t betay,Double_t betaz,
                Double_t px,Double_t py,Double_t pz,Double_t E1){

    //Array Elements of the matrix of Lorentz:
    Double_t A00, A11, A22, A33, A01, A02, A03, A12, A13, A23, beta;
```

```
 //Speed Parameter
 beta = sqrt(pow(betax,2.0)+pow(betay,2.0)+pow(betaz,2.0));

 // Lorentz Factor.
 Double_t gama = pow(1-pow(beta,2),-0.5);

//gama:
A00 = gama;
if(beta !=0){
   //Symmetrical elements of the matrix:
   A01 = -1*(betax*gama);
   A02 = -1*(betay*gama);
   A03 = -1*(betaz*gama);

   //Elements outside the diagonal:
   A12 = (gama-1)*((betax*betay)/(beta*beta));
   A13 = (gama-1)*((betax*betaz)/(beta*beta));
   A23 = (gama-1)*((betay*betaz)/(beta*beta));


   //Elements of the diagonal:
   A11 = (1+((gama-1)*((betax*betax)/(beta*beta))));
   A22 = (1+((gama-1)*((betay*betay)/(beta*beta))));
   A33 = (1+((gama-1)*((betaz*betaz)/(beta*beta))));
 }else{
   //Elements of the identity matrix:
   A11 = A22 = A33 = 1;
   A12 = A13 = A23 = 0;
   A01 = A02 = A03 = 0;

 }

 //Lorentz Transformation:

 Double_t P0 = (A00*E1) + (A01*px) + (A02*py) + (A03*pz);
 Double_t P1 = (A01*E1) + (A11*px) + (A12*py) + (A13*pz);
 Double_t P2 = (A02*E1) + (A12*px) + (A22*py) + (A23*pz);
 Double_t P3 = (A03*E1) + (A13*px) + (A23*py) + (A33*pz);

 //cout<<" Matriz de Transforma  o "<<endl;
 //cout<<" "<<A00<<"  "<<A01<<"  "<<A02<<"  "<<A03<<"  "<<endl;
 //cout<<" "<<A01<<"  "<<A11<<"  "<<A12<<"  "<<A13<<"  "<<endl;
 //cout<<" "<<A02<<"  "<<A12<<"  "<<A22<<"  "<<A23<<"  "<<endl;
 //cout<<" "<<A03<<"  "<<A13<<"  "<<A23<<"  "<<A33<<"  "<<endl;

 Double_t *vector = (Double_t*)malloc(4*sizeof(Double_t));
```

```
//return the 4-vector (E,px,py,pz) = (P0,P1,P2,P3).
vector[0] = P0;
vector[1] = P1;
vector[2] = P2;
vector[3] = P3;

//return pointer
return vector;
}

// ********************************************************************

// Let's begin the function here!
void Eventos_CBJ1(const char* fileNameIn="softQCD.root"){

// histogram statistics box can be selected
gStyle->SetOptStat(111);

// vector size
Int_t N = 100000;

//Precision:
cout<<fixed;
cout<<setprecision(15);

// Open the file that have the object Tree
TFile *fin = new TFile(fileNameIn,"READ");

// Take the objects TTree that be inside of the file .root
TTree *T = (TTree*)fin->Get("T");
TTree *T1 = (TTree*)fin->Get("T1");

//definition  of variables
Double_t x[N], y[N], eta[N], phi[N], E1[N], PT[N], PX[N], PY[N];
Double_t PZ[N], theta[N],GAMA[0], Ecut, InvariantMass,P;
Double_t *vector, en, Px, Py, Pz, somapz =0, somaE=0,somaE1=0;
Double_t somae=0, Theta, pseudorapidity, norm = 1, scale;
int id, Nparticles, cont=0, l, Np=0;


Double_t px, py, pz, pt, e, Eta, Phi, gama;
Double_t Gama, beta, bx, by, bz, Bx, By, Bz, ETA, rapidity;

// Access the Branch of the TTree "T"
T->SetBranchAddress("px",&px);
T->SetBranchAddress("py",&py);
```

```
T->SetBranchAddress("pz",&pz);
T->SetBranchAddress("pt",&pt);
T->SetBranchAddress("e",&e);
T->SetBranchAddress("id",&id);
T->SetBranchAddress("Eta",&Eta);
T->SetBranchAddress("Phi",&Phi);

// Access the Branch of the TTree "T1"
T1->SetBranchAddress("gama",&gama);
T1->SetBranchAddress("bx",&bx);
T1->SetBranchAddress("by",&by);
T1->SetBranchAddress("bz",&bz);
T1->SetBranchAddress("Nparticles",&Nparticles);

// From PYTHIA the number of events and
// the number of the particles.
Int_t nentries = (Int_t)T->GetEntries();
Int_t nevents = (Int_t)T1->GetEntries();

cout << "nentries=" << nentries << endl;

// 82 Events "C-jets" with 1334 showers in the frame LAB.
int lNpts = readevento2("Dados_de_todos_Eventos_LAB.txt",
x,y,eta,phi,E1,PT); // Energy GeV.

THStack *hs = new THStack("hs","Histograms");

//LAB Frame
// create one histogram
TH1F *hist1 = new TH1F("hist1","Multiplicity␣LAB",100,0,20);//CBJ
TH1F *hist2 = new TH1F("hist2","",100,0,20); // pythia

//Energy Distribution
TH1F *hist3 = new TH1F("hist3","Energy␣Distribution␣LAB",
100,0,20); //CBJ
TH1F *hist4 = new TH1F("hist4","",100,0,20); //pythia

// CBJ data Fill the histograms in the LAB Frame:
for (Int_t i=0; i<lNpts; i++) {

    theta[i] = 2*TMath::ATan(TMath::Exp(-eta[i]));
    PX[i] = PT[i]*cos(phi[i]);
    PY[i] = PT[i]*sin(phi[i]);
    PZ[i] = E1[i]*cos(theta[i]);

    //Sum E and pz:
    somaE = somaE + E1[i];
```

```
        somapz = somapz + PZ[i];

        hist1->Fill(eta[i]); // CBJ Data Multiplicity
        hist3->Fill(eta[i],E1[i]); // CBJ Data Energy Distribution

    }
    somapz =0;

    // PYTHIA in the LAB Frame:
    for(Int_t i=0; i<nentries; i++) {

        T->GetEntry(i);

        if(id == 22 || id == 211 || id == -211 ){

                hist2->Fill(Eta); // From PYTHIA with Energy cut
                hist4->Fill(Eta,e); // From PYTHIA with Energy cut
                cont++;

                somapz = somapz + pz;

        }
    }

    somapz =0;

    // ***************************************************************

    // 82 Events "C-jets" with 1334 showers in the frame CM.
    lNpts = readevento2("Dados_de_todos_Eventos_CM.txt",x,y,
    eta,phi,E1,PT); // Energy GeV.

    //CM Frame
    // create one histogram
    TH1F *hist5 = new TH1F("hist5","Multiplicity CM",100,-10,10);
    TH1F *hist6 = new TH1F("hist6","",100,-10,10);

    //Energy Distribution
    TH1F *hist7 = new TH1F("hist7","Energy Distribution CM",
    100,-10,10);
    TH1F *hist8 = new TH1F("hist8","",100,-10,10);

    // CBJ data:
    for (Int_t i=0; i<lNpts; i++) {

        theta[i] = 2*TMath::ATan(TMath::Exp(-eta[i]));
        PX[i] = PT[i]*cos(phi[i]);
```

```
        PY[i] = PT[i]*sin(phi[i]);
        PZ[i] = E1[i]*cos(theta[i]);

        //Sum E and pz:
        somaE1 = somaE1 + E1[i];
        somapz = somapz + PZ[i];

        hist5->Fill(eta[i]); // CBJ Data Multiplicity
        hist7->Fill(eta[i],E1[i]); // CBJ Data Energy Distribution

    }

    cout << "CM(CBJ):" << somapz << endl;

    somapz =0;

    // ************************************************************

    l = 0;
    Double_t somaen=0;

      // PYTHIA pp:

      for(Int_t i=0; i<nevents; i++) {

          T1->GetEntry(i);
          Bx = bx;
          By = by;
          Bz = bz;
          Np = Np + Nparticles;

          gama = pow(1-(Bz*Bz),-0.5);
          //cout << i << " bz = "<< Bz <<" gama = "<< gama << endl;

          for(Int_t j=l; j<Np; j++) {

            T->GetEntry(j);
               if(id == 22 || id == 211 || id == -211 ){

               //Lorentz Transformation
               vector = BOOST(Bx, By, Bz, px, py, pz, e);

               //The return of the BOOST:
               en = vector[0];
               Px = vector[1];
               Py = vector[2];
               Pz = vector[3];
```

```
            // Sum momentum in the CM frame:
            somapz = somapz + Pz;

            //Angle in the CM frame:
            Theta = atan2(pt,Pz);

            //pseudo-rapidity.
            ETA = -1*log(tan(Theta/2));

            // Fill the Histograms multiplicity
            //and Energy distribution
            hist6->Fill(ETA);
            hist8->Fill(ETA, en);

            somae = somae + e;
            somaen = somaen + en;

            }
            l = Np;
        }
    }

    cout << "CM(pythia):" << somapz << endl;

    somapz = 0;

// ***************Compare histograms*****************************

TCanvas *c1 = new TCanvas("c1","Histograms",10,10,3000,3000);

// one TCanvas divide in four parts:
c1->Divide(2,2);
// The first histogram:
c1->cd(1);
c1->cd(1)->SetGridx();     // Horizontal grid
c1->cd(1)->SetGridy();     // Vertical grid
c1->cd(1)->SetLogy(1);

//normalize histogram
scale = norm/(lNpts);
hist1->Scale(scale);

// Chi2 Test ...
//Double_t res[100];
//hist1->Chi2Test(hist2,"UW P",res);
```

```
    hist1->SetLineColor(kBlack);
    hist1->GetYaxis()->SetTitle("dN/d#eta");
    hist1->GetXaxis()->SetTitle("#eta");
    hist1->GetXaxis()->CenterTitle();
    hist1->GetYaxis()->CenterTitle();
    hist1->SetMaximum(1);

    hist1->GetYaxis()->SetTitleSize(30);
    hist1->GetYaxis()->SetTitleFont(43);
    hist1->GetYaxis()->SetTitleOffset(1.4);
    hist1->GetYaxis()->SetLabelFont(43);// font size in pixel
    hist1->GetYaxis()->SetLabelSize(20);

    hist1->GetXaxis()->SetTitleSize(30);
    hist1->GetXaxis()->SetTitleFont(43);
    hist1->GetXaxis()->SetTitleOffset(1.5);
    hist1->GetXaxis()->SetLabelFont(43); // Absolute font size
    hist1->GetXaxis()->SetLabelSize(20);

    hs->Add(hist1);
    hist1->SetFillStyle(1);
    hist1->SetMarkerStyle(8);
    hist1->SetMarkerSize(1);
    hist1->SetMarkerColor(1);
    hist1->Draw("phistE1");

    //normalize histogram
    scale = norm/(cont);
    hist2->Scale(scale);

    hist2->SetLineColor(kRed);
    hist2->SetFillStyle(3001);
    hist2->SetLineWidth(2);
    hist2->SetLineStyle(1);
    hs->Add(hist2);
    hist2->Draw("histsame");

    // draw the legend
    TLegend *legend1a=new TLegend(0.7,0.7,0.9,0.8);
    legend1a->SetTextFont(72);
    legend1a->SetTextSize(0.04);
    legend1a->AddEntry(hist1,"CBJ␣Data","lpe");
    legend1a->Draw();

    // draw the legend
    TLegend *legend2a=new TLegend(0.1,0.8,0.5,0.9);
    legend2a->SetTextFont(72);
```

```
legend2a->SetTextSize(0.04);
legend2a->AddEntry(hist2,"PYTHIA 8 (pp) softQCD: E_{cut}","lpe");
legend2a->Draw();

c1->cd(2);
c1->cd(2)->SetGridx();     // Horizontal grid
c1->cd(2)->SetGridy();     // Vertical grid
c1->cd(2)->SetLogy(1);

//normalize histogram
scale = norm/(somaE);
hist3->Scale(scale);

hist3->SetLineColor(kBlack);
hist3->GetYaxis()->SetTitle("dE/d#eta");
hist3->GetXaxis()->SetTitle("#eta");
hist3->GetXaxis()->CenterTitle();
hist3->GetYaxis()->CenterTitle();
hist3->SetMaximum(1);

hist3->GetYaxis()->SetTitleSize(30);
hist3->GetYaxis()->SetTitleFont(43);
hist3->GetYaxis()->SetTitleOffset(1.4);
hist3->GetYaxis()->SetLabelFont(43); // Absolute font size
hist3->GetYaxis()->SetLabelSize(20);

hist3->GetXaxis()->SetTitleSize(30);
hist3->GetXaxis()->SetTitleFont(43);
hist3->GetXaxis()->SetTitleOffset(1.5);
hist3->GetXaxis()->SetLabelFont(43); // Absolute font size
hist3->GetXaxis()->SetLabelSize(20);

hs->Add(hist3);
hist3->SetFillStyle(1);
hist3->SetMarkerStyle(8);
hist3->SetMarkerSize(1);
hist3->SetMarkerColor(1);
hist3->Draw("phistE1");

//normalize histogram
scale = norm/(somae);
hist4->Scale(scale);

hist4->SetLineColor(kRed);
hist4->SetFillStyle(3001);
hist4->SetLineWidth(2);
hist4->SetLineStyle(1);
```

```
    hs->Add(hist4);
    hist4->Draw("histsame");

    // draw the legend
    TLegend *legend4a=new TLegend(0.7,0.7,0.9,0.8);
    legend4a->SetTextFont(72);
    legend4a->SetTextSize(0.04);
    legend4a->AddEntry(hist3,"CBJ Data","lpe");
    legend4a->Draw();

    // draw the legend
    TLegend *legend5a=new TLegend(0.1,0.8,0.5,0.9);
    legend5a->SetTextFont(72);
    legend5a->SetTextSize(0.04);
    legend5a->AddEntry(hist4,"PYTHIA 8 (pp) softQCD: E_{cut}","lpe");
    legend5a->Draw();

    c1->cd(3);
    c1->cd(3)->SetGridx();     // Horizontal grid
    c1->cd(3)->SetGridy();     // Vertical grid
    c1->cd(3)->SetLogy(1);

    //normalize histogram
    scale = norm/(lNpts);
    hist5->Scale(scale);

    hist5->SetLineColor(kBlack);
    //hist5->SetTitle("Multiplicity CM");
    hist5->GetYaxis()->SetTitle("dN/d#eta");
    hist5->GetXaxis()->SetTitle("#eta");
    hist5->GetXaxis()->CenterTitle();
    hist5->GetYaxis()->CenterTitle();
    hist5->SetMaximum(1);

    hist5->GetYaxis()->SetTitleSize(30);
    hist5->GetYaxis()->SetTitleFont(43);
    hist5->GetYaxis()->SetTitleOffset(1.4);
    hist5->GetYaxis()->SetLabelFont(43); // Absolute font size
    hist5->GetYaxis()->SetLabelSize(20);

    hist5->GetXaxis()->SetTitleSize(30);
    hist5->GetXaxis()->SetTitleFont(43);
    hist5->GetXaxis()->SetTitleOffset(1.5);
    hist5->GetXaxis()->SetLabelFont(43); // Absolute font size
    hist5->GetXaxis()->SetLabelSize(20);

    hs->Add(hist5);
```

```
    hist5->SetFillStyle(1);
    hist5->SetMarkerStyle(8);
    hist5->SetMarkerSize(1);
    hist5->SetMarkerColor(1);
    hist5->Draw("phistE1");

    //c1->Update();

    //normalize histogram
    //cout << hist2->Integral()  << endl;
    scale = norm/(cont);
    hist6->Scale(scale);

    hist6->SetLineColor(kRed);
    hist6->SetFillStyle(3001);
    hist6->SetLineWidth(2);
    hist6->SetLineStyle(1);
    hs->Add(hist6);
    hist6->Draw("histsame");

    // draw the legend
    TLegend *legend7a=new TLegend(0.7,0.7,0.9,0.8);
    legend7a->SetTextFont(72);
    legend7a->SetTextSize(0.04);
    legend7a->AddEntry(hist5,"CBJ␣Data","lpe");
    legend7a->Draw();

    // draw the legend
    TLegend *legend8a=new TLegend(0.1,0.8,0.5,0.9);
    legend8a->SetTextFont(72);
    legend8a->SetTextSize(0.04);
    legend8a->AddEntry(hist6,"PYTHIA␣8␣(pp)␣softQCD:␣E_{cut}","lpe");
    legend8a->Draw();

    c1->cd(4);
    c1->cd(4)->SetGridx();     // Horizontal grid
    c1->cd(4)->SetGridy();     // Vertical grid
    c1->cd(4)->SetLogy(1);

    //normalize histogram
    scale = norm/(somaE1);
    hist7->Scale(scale);

    hist7->SetLineColor(kBlack);
    hist7->GetYaxis()->SetTitle("dE/d#eta");
    hist7->GetXaxis()->SetTitle("#eta");
    hist7->GetXaxis()->CenterTitle();
```

```
hist7->GetYaxis()->CenterTitle();
hist7->SetMaximum(1);

hist7->GetYaxis()->SetTitleSize(30);
hist7->GetYaxis()->SetTitleFont(43);
hist7->GetYaxis()->SetTitleOffset(1.4);
hist7->GetYaxis()->SetLabelFont(43); // Absolute font size
hist7->GetYaxis()->SetLabelSize(20);

hist7->GetXaxis()->SetTitleSize(30);
hist7->GetXaxis()->SetTitleFont(43);
hist7->GetXaxis()->SetTitleOffset(1.5);
hist7->GetXaxis()->SetLabelFont(43); // Absolute font size
hist7->GetXaxis()->SetLabelSize(20);

hs->Add(hist7);
hist7->SetFillStyle(1);
hist7->SetMarkerStyle(8);
hist7->SetMarkerSize(1);
hist7->SetMarkerColor(1);
hist7->Draw("phistE1");

// normalize histogram
scale = norm/(somaen);

//cout << " somaen = " << somaen << endl;

hist8->Scale(scale);
hist8->SetLineColor(kRed);
hist8->SetFillStyle(3001);
hist8->SetLineWidth(2);
hist8->SetLineStyle(1);
hs->Add(hist8);
hist8->Draw("histsame");

// draw the legend
TLegend *legend10a=new TLegend(0.7,0.7,0.9,0.8);
legend10a->SetTextFont(72);
legend10a->SetTextSize(0.04);
legend10a->AddEntry(hist7,"CBJ␣Data","lpe");
legend10a->Draw();

// draw the legend
TLegend *legend11a=new TLegend(0.1,0.8,0.5,0.9);
legend11a->SetTextFont(72);
legend11a->SetTextSize(0.04);
legend11a->AddEntry(hist8,"PYTHIA8(pp)softQCD:E_{cut}","lpe");
```

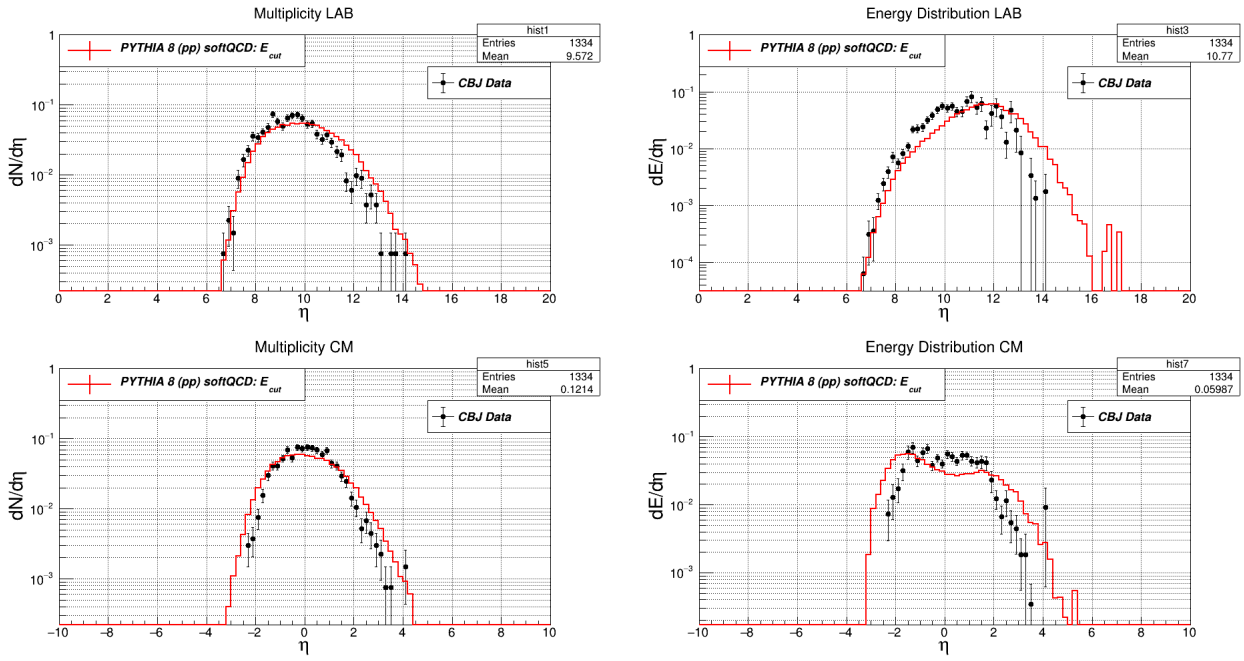

Figura 1: pythia pp collisions

```
    legend11a->Draw();

    // Save the Canvas in .png:
    c1->SaveAs("Graph_EQD/PYTHIA/pythia_pp.png");
    }
```

2 PYTHIA 8230(Heavy Ions) [2, 3, 4]

2.1 PYTHIA 8230: main112.cc

```
// main112.cc is a part of the PYTHIA event generator.
// Copyright (C) 2017 Torbjorn Sjostrand.
// PYTHIA is licenced under the GNU GPL version 2,
// see COPYING for details.
// Please respect the MCnet Guidelines, see GUIDELINES for details.

// Sample program for outputting proton Carbon events to a root file.

#include "Pythia8/Pythia.h"

// ROOT, for saving Pythia events as trees in a file.
#include "TTree.h"
#include "TFile.h"
#include "TMath.h"
```

```
#include "sstream"

// You need to include this to get access to the
// HIInfo object for HeavyIons.
#include "Pythia8/HeavyIons.h"

using namespace Pythia8;

int main() {

  Pythia pythia;

  //define variables:
  Double_t px, py, pz, pt, e;
  Double_t Eta, eta, Phi, gama, beta, bx,by,bz, E1[82], Energy;
  Double_t P=0, somapx=0, somapy=0, somapz=0, somae=0;
  int cont=0, id, Nparticles;

  // Set up the ROOT TFile and TTree.
  TFile *file = TFile::Open("pytHIa.root","recreate");

  //TTRee " T " with the variable of the event
  TTree *T = new TTree("T","Tree");

  //TTRee " T1 " number of events
  TTree *T1 = new TTree("T1","Tree1");

  // Setup the beams.
  pythia.readString("Beams:idA = 2212");  // proton.
  //pythia.readString("Beams:idB = 1000822080"); // Lead.
  pythia.readString("Beams:idB = 1000060120"); // Carbon.
  pythia.readString("Beams:eA = 130000");   // Energy GeV
  pythia.readString("Beams:eB = 1");
  pythia.readString("Beams:frameType = 2");

  //Initialize the Angantyr model to fit the total and semi-includive
  // cross sections in Pythia within some tolerance.
  pythia.readString("HeavyIon:SigFitErr = "
                    "0.02,0.02,0.1,0.05,0.05,0.0,0.1,0.0");
  // These parameters are typicall suitable for sqrt(S_NN)=5TeV
  pythia.readString("HeavyIon:SigFitDefPar = "
                    "17.24,2.15,0.33,0.0,0.0,0.0,0.0,0.0");
  //A simple genetic algorithm is run for 20 generations to fit the
  // parameters.
  pythia.readString("HeavyIon:SigFitNGen = 20");

  // Initialise Pythia.
```

```
pythia.init();

  //The Branchs of the TTree "T".
  T->Branch("px",&px);
  T->Branch("py",&py);
  T->Branch("pz",&pz);
  T->Branch("pt",&pt);
  T->Branch("e",&e);
  T->Branch("id",&id);
  T->Branch("Eta",&Eta);
  T->Branch("Phi",&Phi);

  //The Branchs of the TTree "T1".
  T1->Branch("gama",&gama);
  T1->Branch("bx",&bx);
  T1->Branch("by",&by);
  T1->Branch("bz",&bz);
  T1->Branch("Nparticles",&Nparticles);

  // ********************************************************

  // Loop over events.
  int nEvents = 100;
  for ( int iEvent = 0; iEvent < nEvents; ++iEvent ) {
      if ( !pythia.next() ) continue;
         for (int i = 0; i < pythia.event.size(); ++i) {

                  Particle & p = pythia.event[i];
                  if ( p.isFinal() && p.e() > 500 ) {
                       //Variables of the particles
                       px = p.px();
                       py = p.py();
                       pz = p.pz();
                       pt = p.pT();
                       e  = p.e();
                       id = p.id();
                       Eta = p.eta();
                       Phi = p.phi();

                       //total momentum of the particle i
                       somapx = somapx + px;
                       somapy = somapy + py;
                       somapz = somapz + pz;
                       somae = somae + e;

                       //Fill the TTRee "T"
                       T->Fill();
```

```
                    //Count the number of particle in the event.
                    cont++;
                }
        }

        //betax, betay, betaz.
        bx = somapx/somae;
        by = somapy/somae;
        bz = somapz/somae;

        //total moment calculation
        P = sqrt(pow(somapx,2)+pow(somapy,2)+pow(somapz,2));

        //speed parameter:
        beta = P/somae;

        cout.precision(7);
        //cout << " beta = " << beta << endl;

        //Lorentz Factor
        gama = pow(1-(beta*beta),-0.5);

        Nparticles = cont;
        //Fill the TTRee "T1"
        T1->Fill();

        //null the variables
        cont = somae = somapx = somapy = somapz = P = 0;


 //*************************************************************

 }

// Write tree.
// Print the screen (terminal) the information about the TTRee T
//T->Print();
T->Write();
// Print the screen (terminal) the information about the TTRee T1
//T1->Print();
T1->Write();

// The run is over, so we write out some statistics.
// I moved the delete statement here.
delete file;
```

```
    // And we're done!
    return 0;


}
```

2.2 PYTHIA 8: output

```
*------ HeavyIon fitting of SubCollisionModel to cross sections ----*
 |                    Total:    62.996 mb       (+-  2%)             |
 |          non-diffractive:    34.672 mb       (+-  2%)             |
 |           XX diffractive:     5.582 mb       (+- 10%)             |
 |       wounded target (B):    45.513 mb       (+-  5%)             |
 |   wounded projectile (A):    45.513 mb       (+-  5%)             |
 |          AXB diffractive:     0.000 mb       not used             |
 |                  elastic:    12.225 mb       (+- 10%)             |
 |          elastic b-slope:    16.585 GeV^-2   not used             |
 |                                                                    |
 |   Using a genetic algorithm                                        |
 |   Generation       best Chi2/Ndf                                   |
 |            0               2.28                                    |
 |            1               1.01                                    |
 |            2               0.61                                    |
 |            3               0.56                                    |
 |            4               0.55                                    |
 |            5               0.58                                    |
 |            6               0.57                                    |
 |            7               0.49                                    |
 |            8               0.39                                    |
 |            9               0.55                                    |
 |           10               0.46                                    |
 |           11               0.45                                    |
 |           12               0.43                                    |
 |           13               0.46                                    |
 |           14               0.43                                    |
 |           15               0.45                                    |
 |           16               0.44                                    |
 |           17               0.47                                    |
 |           18               0.37                                    |
 |           19               0.41                                    |
 |                                                                    |
 |                                                                    |
 |     Resulting cross sections (target value)                        |
 |                    Total:     63.07 *( 63.00) mb                   |
 |          non-diffractive:     35.06 *( 34.67) mb                   |
 |           XX diffractive:      6.13 *(  5.58) mb                   |
```

```
 |       wounded target (B):     45.71 *( 45.51) mb                    |
 |   wounded projectile (A):     45.67 *( 45.51) mb                    |
 |          AXB diffractive:      0.00  (  0.00) mb                    |
 |                  elastic:     12.87 *( 12.23) mb                    |
 |          elastic b-slope:     20.67  ( 16.59) GeV^-2                |
 |                 Chi2/Ndf:     0.52                                   |
 |                                                                      |
 |     Resulting parameters:                                            |
 |                   0:    16.75                                        |
 |                   1:     1.50                                        |
 |                   2:     0.34                                        |
 |                                                                      |
 *--- End HeavyIon fitting of parameters in nucleon collision model-*

HeavyIon Info: To avoid refitting, use the following settings
for next run:
  HeavyIon:SigFitNGen = 0
  HeavyIon:SigFitDefPar = 16.75,1.50,0.34,0.0,0.0,0.0,0.0,0.0
 HeavyIon Info: Initializing impact parameter generator
 with width 4.36 fm.
 Angantyr Info: No signal process specified. Assuming minimum bias.
 Angantyr Info: Initializing minimum bias processes.
```

2.3 main112.cc: Output pytHIa.root

```
TTree  May 19 12:35 2018 T   "Tree"
  px   "px/D"    156706
  py   "py/D"    156706
  pz   "pz/D"    156706
  pt   "pt/D"    156706
  e    "e/D"     156701
  id   "id/I"    78386
  Eta  "Eta/D"   156711
  Phi  "Phi/D"   156711
TTree  May 19 12:35 2018 T1  "Tree1"
  gama         "gama/D"         8069
  bx           "bx/D"           8067
  by           "by/D"           8067
  bz           "bz/D"           8067
  Nparticles   "Nparticles/I"   4075
```

2.4 Macro para ler o arquivo: pytHIa.root

Essa macro chamada **pytHIaCBJ.C** contém um script para comparar os dados experimentais da CBJ (disponíveis no link no github (LAB e CM)) e a simulação gerada pelo PYTHIA (pC).

```
//  ---------- macro: pytHIa_CBJ1.C ------------------
```

```
#include "TCanvas.h"
#include "TStyle.h"
#include "TH1.h"
#include "TGaxis.h"
#include "TLatex.h"
#include "TROOT.h"
#include "TGraphErrors.h"
#include "TF1.h"
#include "TLegend.h"
#include "TArrow.h"
#include <math.h>
#include "readevento2.C"
#include "WriteTxt.C"
#include "TFile.h"
#include "TTree.h"

#include "sstream"

//gStyle->SetOptStat(000000); //NOT put subtitles on the graphics.


//  Transformacao de Lorentz (geral em todas as direcoes)...

    Double_t* BOOST(Double_t betax,Double_t betay,Double_t betaz,
                    Double_t px,Double_t py,Double_t pz,Double_t E1){

    //Array Elements of the matrix of Lorentz:
    Double_t A00, A11, A22, A33, A01, A02, A03, A12, A13, A23, beta;

    //Speed Parameter
    beta = sqrt(pow(betax,2)+pow(betay,2)+pow(betaz,2));

    // Lorentz Factor.
    Double_t gama = pow(1-(beta*beta),-0.5);

    //gama:
    A00 = gama;
    if(beta !=0){
       //Symmetrical elements of the matrix:
       A01 = -1*(betax*gama);
       A02 = -1*(betay*gama);
       A03 = -1*(betaz*gama);

       //Elements outside the diagonal:
       A12 = (gama-1)*((betax*betay)/(beta*beta));
       A13 = (gama-1)*((betax*betaz)/(beta*beta));
       A23 = (gama-1)*((betay*betaz)/(beta*beta));
```

```
    //Elements of the diagonal:
    A11 = (1+((gama-1)*((betax*betax)/(beta*beta))));
    A22 = (1+((gama-1)*((betay*betay)/(beta*beta))));
    A33 = (1+((gama-1)*((betaz*betaz)/(beta*beta))));
    }else{
    //Elements of the identity matrix:
    A11 = A22 = A33 = 1;
    A12 = A13 = A23 = 0;
    A01 = A02 = A03 = 0;

    }

    //Lorentz Transformation:
    Double_t P0 = (A00*E1) + (A01*px) + (A02*py) + (A03*pz);
    Double_t P1 = (A01*E1) + (A11*px) + (A12*py) + (A13*pz);
    Double_t P2 = (A02*E1) + (A12*px) + (A22*py) + (A23*pz);
    Double_t P3 = (A03*E1) + (A13*px) + (A23*py) + (A33*pz);

    //cout<<" Matriz de Transforma  o "<<endl;
    //cout<<" "<<A00<<"  "<<A01<<"  "<<A02<<"  "<<A03<<"  "<<endl;
    //cout<<" "<<A01<<"  "<<A11<<"  "<<A12<<"  "<<A13<<"  "<<endl;
    //cout<<" "<<A02<<"  "<<A12<<"  "<<A22<<"  "<<A23<<"  "<<endl;
    //cout<<" "<<A03<<"  "<<A13<<"  "<<A23<<"  "<<A33<<"  "<<endl;

    Double_t *vector = (Double_t*)malloc(4*sizeof(Double_t));

    //return the 4-vector (E,px,py,pz) = (P0,P1,P2,P3).
    vector[0] = P0;
    vector[1] = P1;
    vector[2] = P2;
    vector[3] = P3;

    //return pointer, well done!
    return vector;
}

// ***************************************************************

// Let's begin the function here!
void pytHIa_CBJ1(const char* fileNameIn="pytHIa.root"){

// histogram statistics box can be selected
gStyle->SetOptStat(111);

// vector size
```

```
Int_t N = 100000;

//Precision:
cout<<fixed;
cout<<setprecision(10);

// Open the file that have the object Tree
TFile *fin = new TFile(fileNameIn,"READ");

// Take the objects TTree that be inside of the file .root
TTree *T = (TTree*)fin->Get("T");
TTree *T1 = (TTree*)fin->Get("T1");

//definition  of variables
Double_t x[N], y[N], eta[N], phi[N], E1[N];
Double_t PT[N], PX[N], PY[N], PZ[N], theta[N], Ecut;
Double_t *vector, en, Px, Py, Pz, somapz =0, somaE=0,somaE1=0;
Double_t somae=0, somaen=0, Theta, pseudorapidity, scale;
int id, Nparticles, cont=0, l, Np=0;

Double_t px, py, pz, pt, e, Eta, Phi;
Double_t gama, Gama, beta, bx, by, bz, Bx, By, Bz, ETA, rapidity;

// Access the Branch of the TTree "T"
T->SetBranchAddress("px",&px);
T->SetBranchAddress("py",&py);
T->SetBranchAddress("pz",&pz);
T->SetBranchAddress("pt",&pt);
T->SetBranchAddress("e",&e);
T->SetBranchAddress("id",&id);
T->SetBranchAddress("Eta",&Eta);
T->SetBranchAddress("Phi",&Phi);

// Access the Branch of the TTree "T1"
T1->SetBranchAddress("gama",&gama);
T1->SetBranchAddress("bx",&bx);
T1->SetBranchAddress("by",&by);
T1->SetBranchAddress("bz",&bz);
T1->SetBranchAddress("Nparticles",&Nparticles);

// From PYTHIA the number of events
// and the number of the particles.
Int_t nentries = (Int_t)T->GetEntries();
Int_t nevents = (Int_t)T1->GetEntries();

// 82 Events "C-jets" with 1334 showers in the frame LAB.
int lNpts = readevento2("Dados_de_todos_Eventos_LAB.txt",x,y,
```

```
                            eta,phi,E1,PT); // Energy GeV.

    THStack *hs = new THStack("hs","Histograms");

    //LAB Frame
    // create one histogram
    TH1F *hist1 = new TH1F("hist1","Multiplicity LAB",100,
                                              0,20);//CBJ
    TH1F *hist2 = new TH1F("hist2","",100,0,20); //pythia

    //Energy Distribution CBJ
    TH1F *hist3 = new TH1F("hist3","Energy Distribution LAB",100,
                           0,20);
    TH1F *hist4 = new TH1F("hist4","",100,0,20); //pythia

    // CBJ data Fill the histograms in the LAB Frame:
    for (Int_t i=0; i<lNpts; i++) {

        theta[i] = 2*TMath::ATan(TMath::Exp(-eta[i]));
        PX[i] = PT[i]*cos(phi[i]);
        PY[i] = PT[i]*sin(phi[i]);
        PZ[i] = E1[i]*cos(theta[i]);

        //Sum E and pz:
        somaE = somaE + E1[i];
        somapz = somapz + PZ[i];

        hist1->Fill(eta[i]); // CBJ Data Multiplicity
        hist3->Fill(eta[i],E1[i]); // CBJ Data Energy Distribution

    }
    // PYTHIA in the LAB Frame:
    for(Int_t i=0; i<nentries; i++) {

        T->GetEntry(i);

        if(id == 22 || id == 211 || id == -211 ){

            hist2->Fill(Eta); // From PYTHIA with Energy cut
            hist4->Fill(Eta,e); // From PYTHIA with Energy cut
            cont++;

       }
    }
    // ************************************************************

    // 82 Events "C-jets" with 1334 showers in the frame CM.
```

```
lNpts = readevento2("Dados_de_todos_Eventos_CM.txt",x,y,
                    eta,phi,E1,PT); // Energy GeV.

//CM Frame
// create one histogram
TH1F *hist5 = new TH1F("hist5","Multiplicity␣CM",100,-10,10);
TH1F *hist6 = new TH1F("hist6","",100,-10,10);

//Energy Distribution
TH1F *hist7 = new TH1F("hist7","Energy␣Distribution␣CM",100,
                       -10,10);
TH1F *hist8 = new TH1F("hist8","",100,-10,10);

// CBJ data:
for (Int_t i=0; i<lNpts; i++) {

    theta[i] = 2*TMath::ATan(TMath::Exp(-eta[i]));
    PX[i] = PT[i]*cos(phi[i]);
    PY[i] = PT[i]*sin(phi[i]);
    PZ[i] = E1[i]*cos(theta[i]);

    //Sum E and pz:
    somaE1 = somaE1 + E1[i];
    somapz = somapz + PZ[i];

    hist5->Fill(eta[i]); // CBJ Data Multiplicity
    hist7->Fill(eta[i],E1[i]); // CBJ Data Energy Distribution

}
// ************************************************************

l = 0;

// PYTHIA Heavy Ion:
for(Int_t i=0; i<nevents; i++) {

    T1->GetEntry(i);
    Bx = bx;
    By = by;
    Bz = bz;
    Np = Np + Nparticles;

    for(Int_t j=l; j<Np; j++) {

        T->GetEntry(j);
        if(id == 22 || id == 211 || id == -211 ){
```

```
                //Lorentz Transformation
                vector = BOOST(Bx, By, Bz, px, py, pz, e);

                //The return of the BOOST:
                en = vector[0];
                Px = vector[1];
                Py = vector[2];
                Pz = vector[3];

                //CM
                Theta = atan2(pt,Pz);

                ETA = -1*log(tan(Theta/2)) - 0.35; //pseudo-rapidity.

                hist6->Fill(ETA);
                hist8->Fill(ETA, en);

                somae = somae + e;
                somaen = somaen + en;


            }
            l = Np;

        }

    }

    cout << "sum_pz*=" << somapz << endl;


//****************Compare histograms*****************************

TCanvas *c1 = new TCanvas("c1","Histograms",10,10,3000,3000);

// one TCanvas divide in four parts:
c1->Divide(2,2);
// The first histogram:
c1->cd(1);
c1->cd(1)->SetGridx();     // Horizontal grid
c1->cd(1)->SetGridy();     // Vertical grid
c1->cd(1)->SetLogy(1);

//normalize histogram
scale = norm/(lNpts);
hist1->Scale(scale);
```

```
// Chi2 Test ...
Double_t res[100];
hist1->Chi2Test(hist2,"UW P",res);

hist1->SetLineColor(kBlack);
hist1->GetYaxis()->SetTitle("dN/d#eta");
hist1->GetXaxis()->SetTitle("#eta");
hist1->GetXaxis()->CenterTitle();
hist1->GetYaxis()->CenterTitle();
hist1->SetMaximum(1);

hist1->GetYaxis()->SetTitleSize(30);
hist1->GetYaxis()->SetTitleFont(43);
hist1->GetYaxis()->SetTitleOffset(1.4);
hist1->GetYaxis()->SetLabelFont(43); // Absolute font size
hist1->GetYaxis()->SetLabelSize(20);

hist1->GetXaxis()->SetTitleSize(30);
hist1->GetXaxis()->SetTitleFont(43);
hist1->GetXaxis()->SetTitleOffset(1.5);
hist1->GetXaxis()->SetLabelFont(43); // Absolute font size
hist1->GetXaxis()->SetLabelSize(20);

hs->Add(hist1);
hist1->SetFillStyle(1);
hist1->SetMarkerStyle(8);
hist1->SetMarkerSize(1);
hist1->SetMarkerColor(1);
hist1->Draw("phistE1");

//normalize histogram
scale = 1/(cont);
hist2->Scale(scale);

hist2->SetLineColor(kRed);
hist2->SetFillStyle(3001);
hist2->SetLineWidth(2);
hist2->SetLineStyle(1);
hs->Add(hist2);
hist2->Draw("histsame");

// draw the legend
TLegend *legend1a=new TLegend(0.7,0.7,0.9,0.8);
legend1a->SetTextFont(72);
legend1a->SetTextSize(0.04);
legend1a->AddEntry(hist1,"CBJ Data","lpe");
legend1a->Draw();
```

```
// draw the legend
TLegend *legend2a=new TLegend(0.1,0.8,0.55,0.9);
legend2a->SetTextFont(72);
legend2a->SetTextSize(0.04);
legend2a->AddEntry(hist2,"PYTHIA 8 (pC) softQCD: E_{cut}","lpe");
legend2a->Draw();

c1->cd(2);
c1->cd(2)->SetGridx();     // Horizontal grid
c1->cd(2)->SetGridy();     // Vertical grid
c1->cd(2)->SetLogy(1);

//normalize histogram
scale = 1/(somaE);
hist3->Scale(scale);

hist3->SetLineColor(kBlack);
hist3->GetYaxis()->SetTitle("dE/d#eta");
hist3->GetXaxis()->SetTitle("#eta");
hist3->GetXaxis()->CenterTitle();
hist3->GetYaxis()->CenterTitle();
hist3->SetMaximum(1);

hist3->GetYaxis()->SetTitleSize(30);
hist3->GetYaxis()->SetTitleFont(43);
hist3->GetYaxis()->SetTitleOffset(1.4);
hist3->GetYaxis()->SetLabelFont(43); // Absolute font size
hist3->GetYaxis()->SetLabelSize(20);

hist3->GetXaxis()->SetTitleSize(30);
hist3->GetXaxis()->SetTitleFont(43);
hist3->GetXaxis()->SetTitleOffset(1.5);
hist3->GetXaxis()->SetLabelFont(43); // Absolute font size
hist3->GetXaxis()->SetLabelSize(20);

hs->Add(hist3);
hist3->SetFillStyle(1);
hist3->SetMarkerStyle(8);
hist3->SetMarkerSize(1);
hist3->SetMarkerColor(1);
hist3->Draw("phistE1");

//normalize histogram
scale = 1/(somae);
hist4->Scale(scale);
```

```
hist4->SetLineColor(kRed);
hist4->SetFillStyle(3001);
hist4->SetLineWidth(2);
hist4->SetLineStyle(1);
hs->Add(hist4);
hist4->Draw("histsame");

// draw the legend
TLegend *legend4a=new TLegend(0.7,0.7,0.9,0.8);
legend4a->SetTextFont(72);
legend4a->SetTextSize(0.04);
legend4a->AddEntry(hist3,"CBJ Data","lpe");
legend4a->Draw();

// draw the legend
TLegend *legend5a=new TLegend(0.1,0.8,0.55,0.9);
legend5a->SetTextFont(72);
legend5a->SetTextSize(0.04);
legend5a->AddEntry(hist4,"PYTHIA 8 (pC) softQCD: E_{cut}","lpe");
legend5a->Draw();

c1->cd(3);
c1->cd(3)->SetGridx();     // Horizontal grid
c1->cd(3)->SetGridy();     // Vertical grid
c1->cd(3)->SetLogy(1);

//normalize histogram
scale = norm/(lNpts);
hist5->Scale(scale);

hist5->SetLineColor(kBlack);
//hist5->SetTitle("Multiplicity CM");
hist5->GetYaxis()->SetTitle("dN/d#eta");
hist5->GetXaxis()->SetTitle("#eta");
hist5->GetXaxis()->CenterTitle();
hist5->GetYaxis()->CenterTitle();
hist5->SetMaximum(1);

hist5->GetYaxis()->SetTitleSize(30);
hist5->GetYaxis()->SetTitleFont(43);
hist5->GetYaxis()->SetTitleOffset(1.4);
hist5->GetYaxis()->SetLabelFont(43); // Absolute font size
hist5->GetYaxis()->SetLabelSize(20);

hist5->GetXaxis()->SetTitleSize(30);
hist5->GetXaxis()->SetTitleFont(43);
hist5->GetXaxis()->SetTitleOffset(1.5);
```

```
hist5->GetXaxis()->SetLabelFont(43); // Absolute font size
hist5->GetXaxis()->SetLabelSize(20);


hs->Add(hist5);
hist5->SetFillStyle(1);
hist5->SetMarkerStyle(8);
hist5->SetMarkerSize(1);
hist5->SetMarkerColor(1);
hist5->Draw("phistE1");

//c1->Update();

//normalize histogram
//cout << hist2->Integral()  << endl;
scale = 1/(cont);
hist6->Scale(scale);


hist6->SetLineColor(kRed);
hist6->SetFillStyle(3001);
hist6->SetLineWidth(2);
hist6->SetLineStyle(1);
hs->Add(hist6);
hist6->Draw("histsame");

// draw the legend
TLegend *legend7a=new TLegend(0.7,0.7,0.9,0.8);
legend7a->SetTextFont(72);
legend7a->SetTextSize(0.04);
legend7a->AddEntry(hist5,"CBJ Data","lpe");
legend7a->Draw();

// draw the legend
TLegend *legend8a=new TLegend(0.1,0.8,0.55,0.9);
legend8a->SetTextFont(72);
legend8a->SetTextSize(0.04);
legend8a->AddEntry(hist6,"PYTHIA 8 (pC) softQCD: E_{cut}","lpe");
legend8a->Draw();

c1->cd(4);
c1->cd(4)->SetGridx();     // Horizontal grid
c1->cd(4)->SetGridy();     // Vertical grid
c1->cd(4)->SetLogy(1);

//normalize histogram
scale = norm/(somaE1);
```

```
    hist7->Scale(scale);

    hist7->SetLineColor(kBlack);
    hist7->GetYaxis()->SetTitle("dE/d#eta");
    hist7->GetXaxis()->SetTitle("#eta");
    hist7->GetXaxis()->CenterTitle();
    hist7->GetYaxis()->CenterTitle();
    hist7->SetMaximum(1);

    hist7->GetYaxis()->SetTitleSize(30);
    hist7->GetYaxis()->SetTitleFont(43);
    hist7->GetYaxis()->SetTitleOffset(1.4);
    hist7->GetYaxis()->SetLabelFont(43); // Absolute font size
    hist7->GetYaxis()->SetLabelSize(20);

    hist7->GetXaxis()->SetTitleSize(30);
    hist7->GetXaxis()->SetTitleFont(43);
    hist7->GetXaxis()->SetTitleOffset(1.5);
    hist7->GetXaxis()->SetLabelFont(43); // Absolute font size
    hist7->GetXaxis()->SetLabelSize(20);

    hs->Add(hist7);
    hist7->SetFillStyle(1);
    hist7->SetMarkerStyle(8);
    hist7->SetMarkerSize(1);
    hist7->SetMarkerColor(1);
    hist7->Draw("phistE1");


    // normalize histogram
    scale = 1/(somaen);
    hist8->Scale(scale);

    hist8->SetLineColor(kRed);
    hist8->SetFillStyle(3001);
    hist8->SetLineWidth(2);
    hist8->SetLineStyle(1);
    hs->Add(hist8);
    hist8->Draw("histsame");


    // draw the legend
    TLegend *legend10a=new TLegend(0.7,0.7,0.9,0.8);
    legend10a->SetTextFont(72);
    legend10a->SetTextSize(0.04);
    legend10a->AddEntry(hist7,"CBJ␣Data","lpe");
    legend10a->Draw();
```

```
    // draw the legend
    TLegend *legend11a=new TLegend(0.1,0.8,0.55,0.9);
    legend11a->SetTextFont(72);
    legend11a->SetTextSize(0.04);
    legend11a->AddEntry(hist8,"PYTHIA 8 (pC) softQCD: E_{cut}","lpe");
    legend11a->Draw();

    // Save the Canvas in .png:
    c1->SaveAs("Graph_EQD/PYTHIA/pythia_pC.png");


    }
```

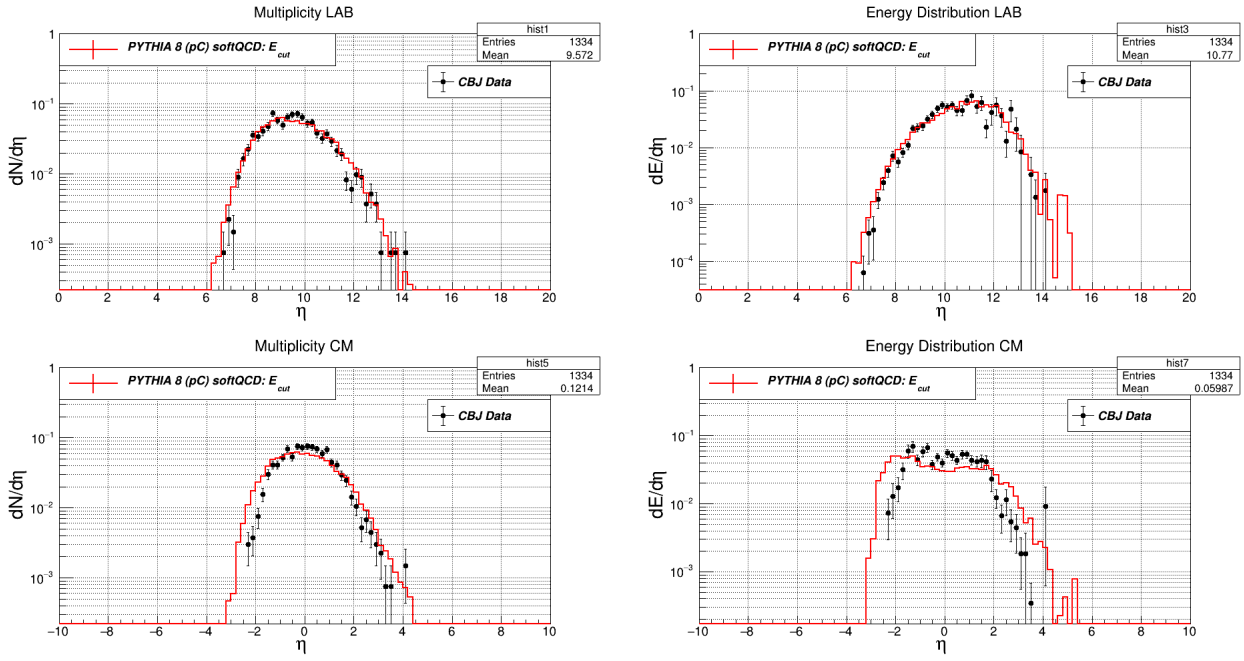

Figura 2: pythia pC collisions

3 CRMC : EPOS LHC [5]

- Download: https://devel-ik.fzk.de/wsvn/mc/crmc/tags/crmc.v1.6.0/?op=dl
  Important: if you are asked login credentials for the download: user: download password: download

```
CRMC EPOS LHC

mkdir -p installed
cd installed
cmake /home/andre/crmc.v1.6.0.r5324
```

```
make
make install
make test ARGS=-V
./bin/crmc -h  // help

// pp collisions in CM Frame in 7 TeV (save in hepmc):
bin/crmc -o hepmc -p3500 -P-3500 -n100 -m0

// pPb collisions
bin/crmc -o root -p3500 -P-1380 -n100 -m0 -i2212 -I822080

// pC collisions
bin/crmc -o root -p130000 -P-0 -n82 -m0 -i2212 -I12
```

3.1 EPOS LHC: Output

```
// pC collisions
$ bin/crmc -o root -p130000 -P-0 -n1000 -m0 -i2212 -I12

          >> crmc <<

  seed:                    130842692 (automatic)
  projectile id:           2212 (p)
  projectile momentum:     130000
  target id:               12 (C)
  target momentum:         -0

  number of collisions:    1000
  parameter file name:     crmc.param
  output file name:        /home/andre/installed/crmc_eposlhc_130842692_p_C_
  HE model:                    0 (EPOS-LHC)

Opening: libEpos.so
initializations ...
######################################################################
#        EPOS LHC      K. WERNER, T. PIEROG                          #
#                      Contact: tanguy.pierog@kit.edu                #
######################################################################
#        WARNING: This is a special retuned version !!!              #
#     Do not publish results without contacting the authors.         #
######################################################################
read from /home/andre/installed/tabs/epos.iniev ...
read from /home/andre/installed/tabs/epos.initl ...
read from /home/andre/installed/tabs/epos.inirj.lhc ...
read from /home/andre/installed/tabs/epos.inics.lhc ...
seedj: 130842692     0.2000000000000000D+01
```

```
 EPOS used with FUSION option
 ==[crmc]==> Collision number 10
 ==[crmc]==> Collision number 20
 ==[crmc]==> Collision number 30
 ...
 ...
 ...
 ==[crmc]==> Collision number 980
 ==[crmc]==> Collision number 990
 ==[crmc]==> Collision number 1000

 succesfully processed 1000 collisions


# Output:
$ rootls -t crmc_eposlhc_130842692_p_C_130000.root
TTree  May 19 17:09 2018 Particle  "particles produced"
  nPart             "nPart/I"             4076
  ImpactParameter   "ImpactParameter/D"   8086
  pdgid             "pdgid[nPart]/I"      570612
  status            "status[nPart]/I"     570631
  px                "px[nPart]/D"         1136948
  py                "py[nPart]/D"         1136948
  pz                "pz[nPart]/D"         1136948
  E                 "E[nPart]/D"          1136912
  m                 "m[nPart]/D"          1136912
```

3.2 Macro para ler o arquivo: crmc....root

Essa macro chamada **EPOS LHC CBJ1.C** contém um script para comparar os dados experimentais da CBJ (disponíveis no link no github (LAB e CM)) e a simulação gerada pelo PYTHIA (pC).

```
#include <ROOT/TDataFrame.hxx>
#include <TCanvas.h>
#include <TApplication.h>
#include <math.h>

using namespace std;
using namespace ROOT::Experimental;
using namespace ROOT::Experimental::VecOps;

void EPOS_LHC_CBJ1(){

  // Multiplicity LAB: CBJ Data/EPOS LHC

  // histogram statistics box can be selected
  gStyle->SetOptStat(111);
```

```
  // File from EPOS LHC proton-proton collisions
  // in the LAB frame Energy: 130 TeV.
  //TDataFrame d("Particle","crmc_eposlhc_63906640_p_p_130000.root");

  // File from EPOS LHC proton-Carbon collisions
  // in the LAB frame Energy: 130 TeV.
  TDataFrame d("Particle","crmc_eposlhc_130842692_p_C_130000.root");

// File from EPOS LHC proton-Lead collisions
// in the LAB frame Energy: 130 TeV.
//TDataFrame d("Particle","crmc_eposlhc_702489137_p_Pb_130000.root");

// File from EPOS LHC pion-Carbon collisions
// in the LAB frame Energy: 130 TeV.
//TDataFrame d("Particle","crmc_eposlhc_169365978_pi_C_130000.root");

// File from EPOS LHC pion-Lead collisions
// in the LAB frame Energy: 130 TeV.
//TDataFrame d("Particle","
//               crmc_eposlhc_392248601_pi_Pb_130000.root");


  //CBJ Data in the LAB frame
    TDataFrame d1("T","Dados_de_todos_Eventos_LAB.root");

  //CBJ Data in the CM frame
    TDataFrame d2("T1","Dados_de_todos_Eventos_CM.root");

 // ************************************************************

  auto c1 = new TCanvas("c1", "c1", 10, 10, 3000, 3000);
  c1->Divide(2,2);

  // One TCanvas.
  auto hs = new THStack("hs","Histograms");

  // Histograms: Multiplicity in the LAB Frame:

  // From CBJ Data:
     auto hist1 = d1.Histo1D({"hist1", "Multiplicity␣LAB", 100,
                              0, 20},"Eta");
     auto hist1_ptr = (TH1D*)&hist1.GetValue();
     hs->Add(hist1_ptr);

  // precision:
     cout<<fixed;
```

```
    cout<<setprecision(10);

    // TVec
    using doubles = TVec<double>;
    using ints = TVec<int>;

    // Calculate rapidity and pass by a filter (Energy Cut)
    auto RapidityCalc = [](doubles Es, doubles pzs, ints pdgids) {
    auto all_RapidityCalc = 0.5*log((Es+pzs)/(Es-pzs));
    auto good_RapidityCalc = all_RapidityCalc[Es > 500.
                             && pdgids == 22];
       return good_RapidityCalc;
    };

    auto dd = d.Define("y", RapidityCalc, {"E", "pz", "pdgid"});

    // From EPOS LHC:
    auto hist2 = dd.Histo1D({"hist2", "Multiplicity LAB", 100,
                            0, 20}, "y");
    auto hist2_ptr = (TH1D*)&hist2.GetValue();
    hs->Add(hist2_ptr);

    double etaMedia = hist2->GetMean();
    cout << " Multiplicity average/hist EPOS LHC = "
              << etaMedia << endl;

double beta = (pow(exp(1),2*etaMedia)-1)/(pow(exp(1),2*etaMedia)+1);

    double gama = pow(1-pow(beta,2),-0.5);

    cout << " beta " << beta << "  gama = " << gama << endl;

    // Chi2 Test ...
    Double_t res[100];
    hist1->Chi2Test(hist2_ptr,"UW P",res);


    // ***********************************************************

    // The histograms:
    c1->cd(1);
    c1->cd(1)->SetGridx();     // Horizontal grid
    c1->cd(1)->SetGridy();     // Vertical grid
    hs->Draw();
    c1->cd(1)->SetLogy(1);

    //normalize histogram
```

```
    hist1->Scale(1/(hist1->Integral()));

    //normalize histogram
    hist2->Scale(1/(hist2->Integral()));

    hist1->GetYaxis()->SetTitle("dN/d#eta");
    hist1->GetXaxis()->SetTitle("#eta");
    hist1->GetXaxis()->CenterTitle();
    hist1->GetYaxis()->CenterTitle();

    hist1->GetYaxis()->SetTitleSize(30);
    hist1->GetYaxis()->SetTitleFont(43);
    hist1->GetYaxis()->SetTitleOffset(1.4);
    hist1->GetYaxis()->SetLabelFont(43); // Absolute font size
    hist1->GetYaxis()->SetLabelSize(20);

    hist1->GetXaxis()->SetTitleSize(30);
    hist1->GetXaxis()->SetTitleFont(43);
    hist1->GetXaxis()->SetTitleOffset(1.5);
    hist1->GetXaxis()->SetLabelFont(43); // Absolute font size
    hist1->GetXaxis()->SetLabelSize(20);

    hist1->SetMaximum(1);
    hist1->SetFillStyle(1);
    hist1->SetMarkerStyle(8);
    hist1->SetMarkerSize(1);
    hist1->SetMarkerColor(1);
    hist1->DrawClone("phistE1");

    hist2->SetLineColor(kRed);
    hist2->SetFillStyle(3001);
    hist2->SetLineWidth(2);
    hist2->SetLineStyle(1);
    hist2->DrawClone("histsame");

    // Draw the Legend
    auto legend = new TLegend(0.7,0.7,0.9,0.8);
    legend->AddEntry("hist1","CBJ␣Data","lep");
    legend->Draw();

    // Draw the Legend
    auto legend1 = new TLegend(0.1,0.8,0.5,0.9);
    legend1->AddEntry("hist2","EPOS␣LHC␣(pC):␣E_{cut}","lep");
    legend1->Draw();

    TLegendEntry *header1 = (TLegendEntry*)legend1
                            ->GetListOfPrimitives()->First();
```

```
header1->SetTextSize(.06);

// ********************************************************
//Energy Distribuition LAB: CBJ Data/EPOS LHC

// TVec
using doubles = TVec<double>;
using ints = TVec<int>;

// From CBJ Data:
auto hist3 = d1.Histo1D({"hist3", "Energy Distribution LAB",
                         100, 0, 20}, "Eta", "e");
auto hist3_ptr = (TH1D*)&hist3.GetValue();
hs->Add(hist3_ptr);

cout << " Energy: average/hist CBJ = " <<
                  hist3->GetMean() << endl;

// Calculate rapidity and pass by a filter (Energy Cut)
auto RapCalc = [](doubles Es, doubles pzs, ints pdgids) {
     auto all_RapCalc  = 0.5*log((Es+pzs)/(Es-pzs));
     auto good_RapCalc = all_RapCalc[Es > 500. && pdgids == 22];
     return good_RapCalc;
};

auto dd1 = d.Define("rap", RapCalc, {"E", "pz", "pdgid"}).
                    Define("good_E", "E[E>500. && pdgid == 22]");
auto hist4 = dd1.Histo1D({"hist4", "Energy Distribution LAB",
                          100, 0, 20}, "rap", "good_E");
auto hist4_ptr = (TH1D*)&hist4.GetValue();
hs->Add(hist4_ptr);

cout << " Energy: average/hist EPOS LHC = "
           << hist4->GetMean() << endl;

c1->cd(2);
c1->cd(2)->SetGridx();     // Horizontal grid
c1->cd(2)->SetGridy();     // Vertical grid
hs->Draw();
c1->cd(2)->SetLogy(1);


//normalize histogram
hist3->Scale(1/(hist3->Integral()));

//normalize histogram
hist4->Scale(1/(hist4->Integral()));
```

```
hist3->GetYaxis()->SetTitle("dE/d#eta");
hist3->GetXaxis()->SetTitle("#eta");
hist3->GetXaxis()->CenterTitle();
hist3->GetYaxis()->CenterTitle();

hist3->GetYaxis()->SetTitleSize(30);
hist3->GetYaxis()->SetTitleFont(43);
hist3->GetYaxis()->SetTitleOffset(1.4);
hist3->GetYaxis()->SetLabelFont(43); // Absolute font size
hist3->GetYaxis()->SetLabelSize(20);

hist3->GetXaxis()->SetTitleSize(30);
hist3->GetXaxis()->SetTitleFont(43);
hist3->GetXaxis()->SetTitleOffset(1.5);
hist3->GetXaxis()->SetLabelFont(43); // Absolute font size
hist3->GetXaxis()->SetLabelSize(20);

hist3->SetMaximum(1);
hist3->SetFillStyle(1);
hist3->SetMarkerStyle(8);
hist3->SetMarkerSize(1);
hist3->SetMarkerColor(1);
hist3->DrawClone("phistE1");

hist4->SetLineColor(kRed);
hist4->SetFillStyle(3001);
hist4->SetLineWidth(2);
hist4->SetLineStyle(1);
hist4->DrawClone("histsame");

// Draw the Legend
auto legend2 = new TLegend(0.7,0.7,0.9,0.8);
legend2->AddEntry("hist3","CBJ␣Data","lep");
legend2->Draw();

// Draw the Legend
auto legend3 = new TLegend(0.1,0.8,0.5,0.9);
legend3->AddEntry("hist4","EPOS␣LHC␣(pC):␣E_{cut}","lep");
legend3->Draw();

TLegendEntry *header3 = (TLegendEntry*)legend3
                         ->GetListOfPrimitives()->First();
header3->SetTextSize(.06);

// ************************************************************
```

```
// TVec
using doubles = TVec<double>;
using ints = TVec<int>;

// ************************************************************
// Multiplicity CM: CBJ Data/EPOS LHC

// Histograms: Multiplicity in the LAB Frame:

// From CBJ Data:
auto hist5 = d2.Histo1D({"hist5", "Multiplicity␣CM", 100,
                         -10, 10},"Eta");
auto hist5_ptr = (TH1D*)&hist5.GetValue();
hs->Add(hist5_ptr);

// Calculate rapidity and pass by a filter (Energy Cut)
auto RapidityCal = [&beta](doubles Es, doubles pzs,
                           ints pdgids) {
     auto all_RapidityCal = 0.5*log((Es +pzs)/(Es - pzs))
                            - 0.5*log((1+beta)/(1-beta));
     auto good_RapidityCal = all_RapidityCal[Es > 500.
                             && pdgids == 22];
     return good_RapidityCal;
};

auto dd3 = d.Define("rap1", RapidityCal, {"E", "pz", "pdgid"});

// From EPOS LHC:
auto hist6 = dd3.Histo1D({"hist6", "Multiplicity␣CM",
                          100, -10, 10}, "rap1");
auto hist6_ptr = (TH1D*)&hist6.GetValue();
hs->Add(hist6_ptr);
// ************************************************************

c1->cd(3);
c1->cd(3)->SetGridx();     // Horizontal grid
c1->cd(3)->SetGridy();     // Vertical grid
hs->Draw();
c1->cd(3)->SetLogy(1);

//normalize histogram
hist5->Scale(1/(hist5->Integral()));

//normalize histogram
hist6->Scale(1/(hist6->Integral()));

hist5->GetYaxis()->SetTitle("dN/d#eta");
```

```
    hist5->GetXaxis()->SetTitle("#eta");
    hist5->GetXaxis()->CenterTitle();
    hist5->GetYaxis()->CenterTitle();

    hist5->GetYaxis()->SetTitleSize(30);
    hist5->GetYaxis()->SetTitleFont(43);
    hist5->GetYaxis()->SetTitleOffset(1.4);
    hist5->GetYaxis()->SetLabelFont(43); // Absolute font size
    hist5->GetYaxis()->SetLabelSize(20);

    hist5->GetXaxis()->SetTitleSize(30);
    hist5->GetXaxis()->SetTitleFont(43);
    hist5->GetXaxis()->SetTitleOffset(1.5);
    hist5->GetXaxis()->SetLabelFont(43); // Absolute font size
    hist5->GetXaxis()->SetLabelSize(20);

    hist5->SetMaximum(1);
    hist5->SetFillStyle(1);
    hist5->SetMarkerStyle(8);
    hist5->SetMarkerSize(1);
    hist5->SetMarkerColor(1);
    hist5->DrawClone("phistE1");

    hist6->SetLineColor(kRed);
    hist6->SetFillStyle(3001);
    hist6->SetLineWidth(2);
    hist6->SetLineStyle(1);
    hist6->DrawClone("histsame");

    // Draw the Legend
    auto legend4 = new TLegend(0.7,0.7,0.9,0.8);
    legend4->AddEntry("hist5","CBJ␣Data","lep");
    legend4->Draw();

    // Draw the Legend
    auto legend5 = new TLegend(0.1,0.8,0.5,0.9);
    legend5->AddEntry("hist6","EPOS␣LHC␣(pC):␣E_{cut}","lep");
    legend5->Draw();

    TLegendEntry *header5 =
            (TLegendEntry*)legend5->GetListOfPrimitives()->First();
    header5->SetTextSize(.06);


    // ************************************************************
    // Energy Distribuition CM: CBJ Data/EPOS LHC
```

```
// From CBJ Data:
auto hist7 = d2.Histo1D({"hist7", "Energy␣Distribution␣CM", 100,
                         -10, 10}, "Eta", "e");
auto hist7_ptr = (TH1D*)&hist7.GetValue();
hs->Add(hist7_ptr);

// Calculate rapidity and pass by a filter (Energy Cut)
auto RapCal = [&beta, &gama](doubles Es, doubles pzs) {
     auto all_RapCal  = 0.5*log((Es+pzs)/(Es-pzs))
                              - 0.5*log((1+beta)/(1-beta));
     auto good_RapCal = all_RapCal[Es > 500.];
     return good_RapCal;
};

auto EnergyCM = [&beta, &gama](doubles Es, doubles pzs) {
     auto all_Energy  = gama*Es - beta*gama*pzs;
     auto good_Energy = all_Energy[Es > 500.];
     return good_Energy;
};

auto ddd1 = d.Define("rap2", RapCal, {"E", "pz"})
                    .Define("good_E2", EnergyCM, {"E", "pz"});
auto hist8 = ddd1.Histo1D({"hist8", "Energy␣Distribution␣CM",
                           100, -10, 10}, "rap2", "good_E2");
auto hist8_ptr = (TH1D*)&hist8.GetValue();
hs->Add(hist8_ptr);

c1->cd(4);
c1->cd(4)->SetGridx();     // Horizontal grid
c1->cd(4)->SetGridy();     // Vertical grid
hs->Draw();
c1->cd(4)->SetLogy(1);

//normalize histogram
hist7->Scale(1/(hist7->Integral()));

//normalize histogram
hist8->Scale(1/(hist8->Integral()));

hist7->GetYaxis()->SetTitle("dE/d#eta");
hist7->GetXaxis()->SetTitle("#eta");
hist7->GetXaxis()->CenterTitle();
hist7->GetYaxis()->CenterTitle();

hist7->GetYaxis()->SetTitleSize(30);
hist7->GetYaxis()->SetTitleFont(43);
hist7->GetYaxis()->SetTitleOffset(1.4);
```

```
    hist7->GetYaxis()->SetLabelFont(43); // Absolute font size
    hist7->GetYaxis()->SetLabelSize(20);

    hist7->GetXaxis()->SetTitleSize(30);
    hist7->GetXaxis()->SetTitleFont(43);
    hist7->GetXaxis()->SetTitleOffset(1.5);
    hist7->GetXaxis()->SetLabelFont(43); // Absolute font size
    hist7->GetXaxis()->SetLabelSize(20);

    hist7->SetMaximum(1);
    hist7->SetFillStyle(1);
    hist7->SetMarkerStyle(8);
    hist7->SetMarkerSize(1);
    hist7->SetMarkerColor(1);
    hist7->DrawClone("phistE1");

    hist8->SetLineColor(kRed);
    hist8->SetFillStyle(3001);
    hist8->SetLineWidth(2);
    hist8->SetLineStyle(1);
    hist8->DrawClone("histsame");

    // Draw the Legend
    auto legend6 = new TLegend(0.7,0.7,0.9,0.8);
    legend6->AddEntry("hist7","CBJ␣Data","lep");
    legend6->Draw();

    // Draw the Legend
    auto legend7 = new TLegend(0.1,0.8,0.5,0.9);
    legend7->AddEntry("hist8","EPOS␣LHC␣(pC):␣E_{cut}","lep");
    legend7->Draw();

    TLegendEntry *header7 =
             (TLegendEntry*)legend7->GetListOfPrimitives()->First();
    header7->SetTextSize(.06);

    // Save in .png
    c1->SaveAs("Graph_EQD/EPOS_LHC/EPOS_pp.png");

    //Well Done! Great Job.

}
int main(){
    TApplication app("app", nullptr, nullptr);
    EPOS_LHC_CBJ1();
    app.Run();
return 0;
```

```
}
```

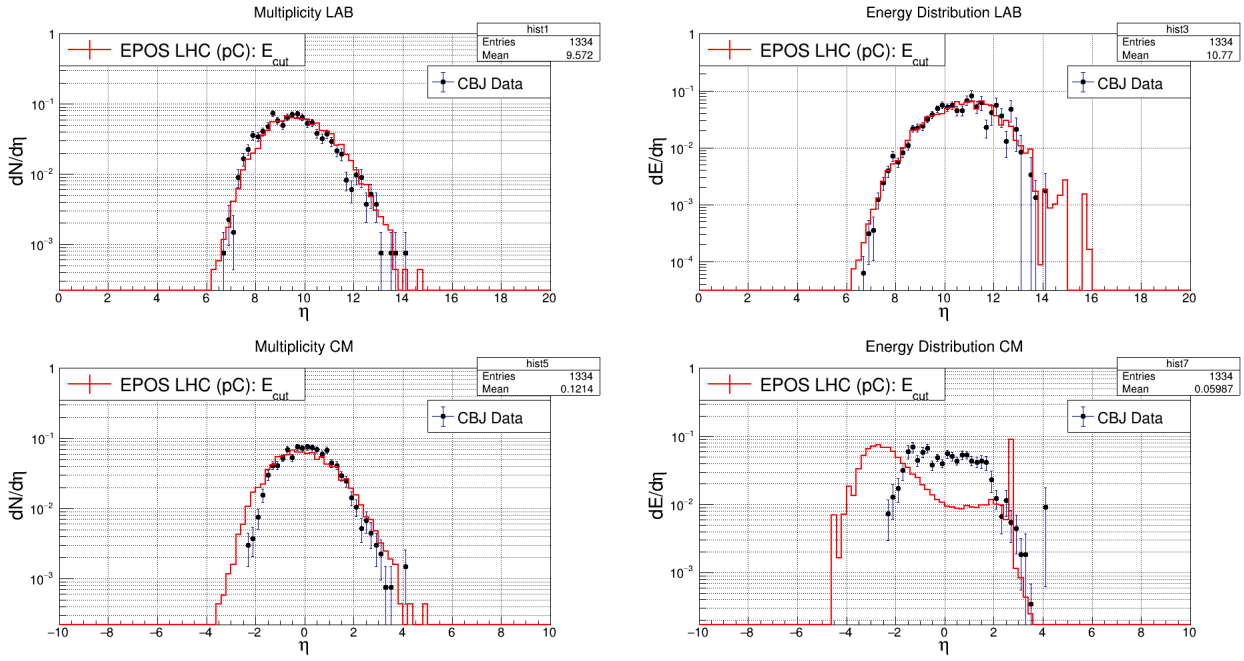

Figura 3: EPOS LHC pC collisions

Referências

[1] Torbjörn Sjöstrand et al. "An Introduction to PYTHIA 8.2". Em: *Comput. Phys. Commun.* 191 (2015), pp. 159–177. DOI: 10.1016/j.cpc.2015.01.024. arXiv: 1410.3012 [hep-ph].

[2] C. Loizides. "Glauber modeling of high-energy nuclear collisions at the subnucleon level". Em: *Phys. Rev. C* 94 (2 ago. de 2016), p. 024914. DOI: 10.1103/PhysRevC.94.024914. URL: https://link.aps.org/doi/10.1103/PhysRevC.94.024914.

[3] Georges Aad et al. "Measurement of the centrality dependence of the charged-particle pseudorapidity distribution in proton–lead collisions at $\sqrt{s_{\mathrm{NN}}} = 5.02$ TeV with the ATLAS detector". Em: *Eur. Phys. J.* C76.4 (2016), p. 199. DOI: 10.1140/epjc/s10052-016-4002-3. arXiv: 1508.00848 [hep-ex].

[4] Christian Bierlich, Gösta Gustafson e Leif Lönnblad. "Diffractive and non-diffractive wounded nucleons and final states in pA collisions". Em: *JHEP* 10 (2016), p. 139. DOI: 10.1007/JHEP10(2016)139. arXiv: 1607.04434 [hep-ph].

[5] T. Pierog et al. "EPOS LHC: Test of collective hadronization with data measured at the CERN Large Hadron Collider". Em: *Phys. Rev.* C92.3 (2015), p. 034906. DOI: 10.1103/PhysRevC.92.034906. arXiv: 1306.0121 [hep-ph].

A ROOT

- Building ROOT — ROOT a Data analysis Framework - ROOT @ cern
- Install and Build ROOT

```
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04 LTS
Release:        18.04
Codename:       bionic

PRE-BUILDING:
```

```
$ sudo apt install ttf-xfree86-nonfree x11proto-xf86bigfont-dev
libfontenc1 libfreetype6 rpm ubuntu-restricted-extras git dpkg-dev
gcc binutils libxext-dev gfortran libssl-dev libpcre3-dev
libftgl-dev libfftw3-dev libglu1-mesa-dev libglew-dev libxml2-dev
graphviz-dev libavahi-compat-libdnssd-dev flashplugin-installer
t1-xfree86-nonfree xfonts-75dpi libfontconfig1 libxrender1
libgtk2.0-0 libgdk-pixbuf2.0-0 libglib2.0-0 libxft2 libc6 zlib1g
libgcc1 gcc-multilib g++-multilib libx11-6 libkrb5-dev libgsl-dev
libqt4-dev make g++ libx11-dev xfstt libiodbc2 libiodbc2-dev
mpi-default-dev libhdf5-mpi-dev x11-common build-essential
libjpeg-turbo8-dev libjpeg8-dev libjpeg-dev x11-utils gsl-bin
libgif-dev libgmp3-dev freeglut3-dev subversion libxmu-dev
libimlib2 linux-firmware synaptic vlc gimp gimp-data
gimp-plugin-registry libglu1-mesa-dev libglew-dev cernlib
libxpm-dev libxft-dev libglu1-mesa-dev libglew-dev
libmysqlclient-dev graphviz-dev libldap2-dev libxml2-dev
libgsl-dev ccache libfastjet-dev libfastjettools-dev
libfastjet0v5 fastjet-doc fastjet-examples libfastjet-fortran-dev
libfastjetplugins-dev libfastjet-fortran0 libfastjetplugins0
libfastjettools0 libopencv-dev libhepmcfio-dev libhepmc4
libhepmc-dev libjs-jquery libboost-all-dev aptitude python-dev
autotools-dev libicu-dev libbz2-dev liblhapdf-dev libgsl-dev
libssl-dev libglu1-mesa-dev libglew-dev


// and download the new version from CERN...
$ sudo mkdir /opt/root6
$ sudo chown seu nome /opt/root6

$ git clone http://root.cern.ch/git/root.git root6_src  // download

cmake -DPYTHIA8_DIR=/home/seu nome/pythia8226 -Dpythia8=ON -Dpython=ON
-Dall=On -DPYTHON_EXECUTABLE=/usr/bin/python3 -Droofit=ON
```

```
-Dhttp=ON -Dccache=ON -Dcling=ON /opt/root6/root6_src/

$ sudo make -j4 && sudo make install

~$ root
------------------------------------------------------------------
| Welcome to ROOT 6.13/03                    http://root.cern.ch |
|                                  (c) 1995-2018, The ROOT Team  |
| Built for linuxx8664gcc                                        |
| From heads/master@v6-13-02-916-gd8ac52b72c, mai 07 2018, 19:24:17 |
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'     |
------------------------------------------------------------------

root [0]
```

B ROOT Data Analysis Framework

- **About ROOT Forum support and discussion**:
  O registro das minhas questões ao longo da minha aprendizagem sobre o ROOT no link ROOT Talk