Estudio de las fluctuaciones del momento transverso promedio a energías del experimento NICA

4th Computing/Analysis Workshop of the MexNICA Collaboration

Kitzia M. Hernandez C.

Facultad de Ciencias Físico-Matemáticas
Universidad Autonoma de Sinaloa

June 15, 2022

Contents

Objetivo

- Calcular las fluctuaciones del momento transverso promedio evento por evento a energía del experimento NICA [2] mediante simulaciones Monte Carlo en preparación para comparar con los primeros resultados del detector MPD (Multi-Purpose Detector).

Fluctuaciones Evento por Evento

- Las fluctuaciones de diferentes observables como el momento transverso promedio $\langle p_t \rangle$, la multiplicidad (N), la energía transversa o las cargas conservadas (Q,B,S), juegan un papel importante en el estudio de la transición de fase, pues se espera que alrededor del punto critico, estas muestren un cambio abrupto en las propiedades físicas del sistema [3].
- El enfoque utilizado para medir estas fluctuaciones consiste en caracterizar la distribución del parámetro observado mediante la medición de variables estadísticas como la varianza, la covarianza y los momentos superiores.

Definición del observable

El momento transverso promedio es el observable a estudiar en este análisis, se construye a partir de:

- El p_T de la partícula.
- El p_T promedio por evento $\rightarrow \langle p_T \rangle$
- El promedio de la distribución del momento transverso promedio$\rightarrow \langle\langle p_T \rangle\rangle$

Definición del observable

En este trabajo se propone;

- Extraer las fluctuaciones dinámicas de $\langle p_T \rangle \rightarrow \sigma^2_{\langle p_T \rangle dyn}$ para diferente rangos de centralidad.
- Estudiar el comportamiento de el tercer momento de la distribución de $\langle p_t \rangle$

Para ello se utilizan las siguientes expresiones:

$$\langle p_t \rangle \equiv \langle\langle p_t \rangle\rangle = \left\langle \frac{\sum_{i=1}^{N_{ch}} p_{t,i}}{N_{ch}} \right\rangle_{ev} \tag{1}$$

Propuestas en [1, 4] y de las cuáles se define la correlación de el momento transverso de dos partículas, asociado con las fluctuaciones dinámicas.

$$\langle \Delta p_{t,i} \Delta p_{t,j} \rangle = \left\langle \frac{\sum_{i,j} (p_i - \langle\langle p_t \rangle\rangle)(p_j - \langle\langle p_t \rangle\rangle)}{N - ch(N_{ch} - 1)} \right\rangle_{ev} \tag{2}$$

El cual es equivalente a la diferencia entre las fluctuaciones totales y las fluctuaciones estadísticas.

$$\sigma^2_{dyn} = \sigma^2_{\langle p_t \rangle} - \left\langle \frac{\sigma^2_{\langle p_t \rangle}}{N_{ch}} \right\rangle_{ev} \tag{3}$$

Similarmente, puede definirse el tercer momento de la distribución de $\langle p_t \rangle$ como:

$$\mu_3 = \frac{\sum_{i=1}^{N_{ev}} (\langle p_t \rangle_i - \langle\langle p_t \rangle\rangle)^3}{N_{ev}} \tag{4}$$

Generación de eventos con UrQMD

Se generaron eventos de colisiones de Bi+Bi con el programa UrQMD con energía en el centro de masa de 9 GeV y rango de parámetro de impacto fijo de acuerdo a la siguiente tabla:

Centralidad(%)	b[fm]
0-10	0-4
10-20	4-6
20-30	6-7
30-40	7-8
40-50	8-9
50-60	9-10
60-70	10-11
70-80	11-12
80-90	12-13.5
90-100	>13.5

Table: Valores del parámetro de impacto

- Se generaron 50,000 eventos por rango de centralidad.
- Se encuentran en */storage/mexnica/kitziahc/gen/BiBi/*

```
[kitziamhc@xook ~]$ cd /storage/mexnica/kitziahc/
[kitziamhc@xook kitziahc]$ cd gen/BiBi
[kitziamhc@xook BiBi]$ ls
0-4fm  10-11fm  11-12fm  11GeV  12-13.5fm  13-14fm  4-6fm  6-7fm  7.7GeV  7-8fm  8-9fm  9-10fm  9MB
[kitziamhc@xook BiBi]$ cd 11GeV/
[kitziamhc@xook 11GeV]$ ls
0-4fm  10-11fm  11-12fm  12-13fm  13-14fm  4-6fm  6-7fm  7-8fm  8-9fm  9-10fm
[kitziamhc@xook 11GeV]$
```

Figure: Carpeta con los datos generados en UrQMD

- En cada carpeta se encuentran 500 archivos *test.f13.Bi2b.0* con 100 eventos cada uno.

Procesamiento Evento por Evento mediante Unigen

Para realizar el análisis evento por evento se utilizó el programa Unigen el cual convierte los archivos de salida de UrQMD en un archivo root con la información obtenida agrupada en eventos. Los macros para ellos se encuentran en:

- */storage/mexnica/kitziahc/Analysis/testUnigen*
- Los datos de salida se encuentran en */storage/mexnica/kitziahc/Analysis/Trees/BiBi9GeV/0-4fm/TreeUrqmd_BiBi9GeV_0.root*
- Después con ayuda del macro */storage/mexnica/kitziahc/Unigen/merge.C* se unen los 500 archivos en uno solo el cuál se encuentra en */storage/mexnica/kitziahc/Unigen/outData*

```
#PBS -N UniGenBiBi

#PBS -S /bin/bash

# Specific the queue type
#PBS -q short

# Archivo de salida
#PBS -o /storage/mexnica/kitziahc/Analysis/Trees/BiBi9GeVHydro/output$JOBID.log

# Archivo de salida de error
#PBS -e /storage/mexnica/kitziahc/Analysis/Trees/BiBi9GeVHydro/error$JOBID.log

##PBS -l walltime=2400:00:00
##PBS -l cput=2400:00:00

echo "------------------Nodo ----------------------------------------------------------------"
/bin/hostname

echo "-----------------Directorio iniciar de trabajo-----------------------------------"
pwd

echo "-----------------Trabajo---------------------------------------------------------"
cd /storage/mexnica/kitziahc/Analysis/testUnigen
source config.sh
cd /storage/mexnica/kitziahc/Analysis/Trees/BiBi9GeVHydro

urqmd2u /storage/mexnica/isabel/BiBi/9GeVMBhydro/test.f13.Bi9MBhydro.$JOBID TreeUrqmdHydro_BiBi_$JOBID.root 10
module unload ROOT/5-26-00
~
~
~
~
```

Figure: Run2Ana.pbs

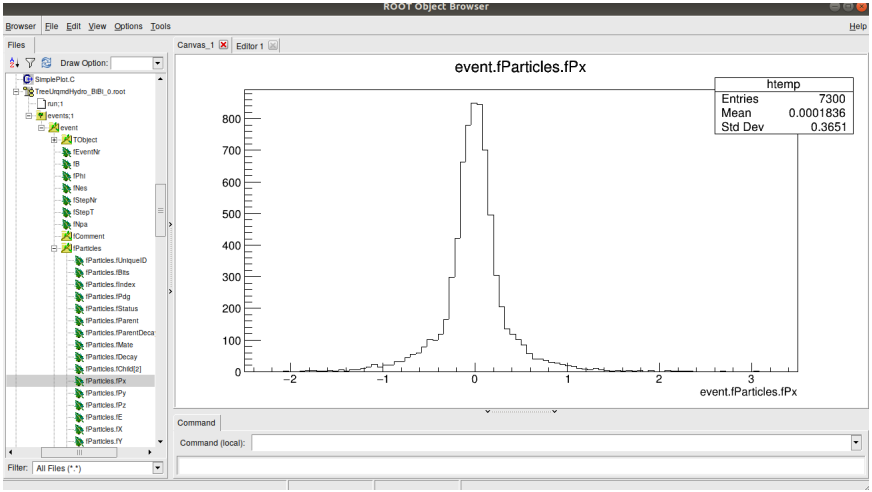


Figure: Ejemplo archivo de salida

- Para poder ejecutar los macros es necesario cargar las variables de ambiente tanto de ROOT como de UniGen. Para ello en la carpeta */storage/mexnica/kitziahc/Unigen*

```
[kitziamhc@xook Unigen]$ source module.v3.sh
[kitziamhc@xook Unigen]$ source config.sh
[kitziamhc@xook Unigen]$ source config/unigenlogin.sh
```

Cálculo de las distribuciones de pt promedio evento por evento

- El macro está basado en el método TTree::MakeClass de ROOT.

```
root [0] TFile* f = new TFile("BiBi9GeVMB_Hydro.root")
Warning in <TUnixSystem::SetDisplay>: DISPLAY not set, setting it to 2806:1016:f:473a:794b:f98:3026:3a9f
(TFile *) 0x20bae60
root [1] f->ls()
TFile**         BiBi9GeVMB_Hydro.root   chain files
 TFile*         BiBi9GeVMB_Hydro.root   chain files
  KEY: TTree    events;1        UrQMD tree
root [2] events->MakeClass("ptdist_test")
Info in <TTreePlayer::MakeClass>: Files: ptdist_test.h and ptdist_test.C generated from TTree: events
(int) 0
root [3]
```

Figure: Ejemplo del uso de TTree::MakeClass

en donde ptdist.h contiene la definición de la clase, el contenido de cada rama y el macro ptdist.C contiene los metodos de la clases los cuales incluyen Loop(), GetEntry(), LoadTree, Cut(), Show().

```
// from TTree events/UrQMD tree
// found on file: BiBi_946Gev_1M.root
//////////////////////////////////////////////////////////

#ifndef ptdist_h
#define ptdist_h

#include <TROOT.h>
#include <TChain.h>
#include <TFile.h>

// Header file for the classes stored in the TTree if any.
//#include "UEvent.h"
#include "TObject.h"
//#include "UParticle.h"

class ptdist {
public :
   TTree          *fChain;   //!pointer to the analyzed TTree or TChain
   Int_t           fCurrent; //!current Tree number in a TChain

// Fixed size dimensions of array or collections stored in the TTree if any.
   static constexpr Int_t kMaxfParticles = 2101;

   // Declaration of leaf types
 //UEvent          *event;
   UInt_t          fUniqueID;
   UInt_t          fBits;
   Int_t           fEventNr;
   Double_t        fB;
   Double_t        fPhi;
   Int_t           fNes;
   Int_t           fStepNr;
   Double_t        fStepT;
   Int_t           fNpa;
   TString         fComment;
```

Figure: ptdist.h

Mientras que la clase ptdist::Loop() consiste de un for-loop que llama a la función *GetEntry* para cada entrada. Las clases Loop() en ptdist.C se modifica de acuerdo a lo que queramos hacer.

```
Float_t nParticles=0;
Float_t nPt=0;
Float_t nAv = 0;
Float_t nEta = 0;
Float_t bmi = 13;
Float_t bma = 14.5;
Float_t nCh=0;
//Event loop
for (Long64_t jentry=0; jentry<nentries;jentry++) {
   Long64_t ientry = LoadTree(jentry);
   if (ientry < 0) break;
   nb = fChain->GetEntry(jentry);   nbytes += nb;
   // if (Cut(ientry) < 0) continue;
   //cout << "evento: " << jentry << endl;
    nPlons = 0;
    nKaons = 0;
    nProtons = 0;
    ptpion = 0;
    ptkaon = 0;
    ptProtons = 0;
    nParticles = 0;
    nPt=0;
    nEta=0;
    nAv = 0;
    nCh = 0;
    Double_t b = fB; //Impact parameter
    Int_t nGenParticles  = fParticles_;
    //Particle loop
    for (Int_t ipart=0; ipart<nGenParticles; ipart++)
        {
            Int_t   particle_pdg = fParticles_fPdg[ipart];
            Double32_t  Px = fParticles_fPx[ipart];   //[nPart]
            Double32_t  Py = fParticles_fPy[ipart];   //[nPart]
            Double32_t  Pz = fParticles_fPz[ipart];   //[nPart]
            Double32_t  E = fParticles_fE[ipart];     //[nPart]
```

Figure: ptdist.C

```
      //  if (b<bmi) continue;
      // if (b > bma) continue;
          //cout << "b " << b << endl;
      //    if (TMath::Abs(eta)> 1) continue;
        // if (pT_part < 0.10) continue;
          //if (pT_part > 0.60) continue;

            nEta = eta;
            nParticles++;
            nPt+=pT_part;
            nEta++;
            nAv = nPt/nParticles;
            nCh = nParticles*(nParticles -1);
            nCh++;
      //cout << "nch: " << nCh << endl;
      } //end particle loop
   //Fill histograms
   //ptH->Sumw2(true);
   ptH->Fill(nPt/nParticles);
   NchPt->Fill(nParticles,nPt/nParticles);
   Nev->Fill(nParticles);
   ptEta->Fill(nEta, nPt/nParticles);
//   Npt->Fill(nCh);
  //Npt->Scale(nParticles);
 }//End event loop

  //Saving
  TFile Output("ptdistBiBi11GeV_04fm_nocuts.root","recreate");
  Output.cd();
  ptH->Write();
  NchPt->Write();
  Nev->Write();
  ptEta->Write();
//   Npt->Write();
  Output.Close();
}
```

Figure: ptdist.C

Para ejecutar el macro es mediante los siguientes comandos:

```
[kitziamhc@xook ~]$ cd /storage/mexnica/kitziahc/Unigen/
[kitziamhc@xook Unigen]$ source module.v3.sh
[kitziamhc@xook Unigen]$ source config.sh
[kitziamhc@xook Unigen]$ source config/unigenlogin.sh
UNIGEN = /storage/mexnica/kitziahc/Unigen
[kitziamhc@xook Unigen]$ root
   ------------------------------------------------------------------
  | Welcome to ROOT 6.24/02                        https://root.cern |
  | (c) 1995-2021, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Aug 18 2021, 19:03:00                 |
  | From tag , 28 June 2021                                          |
  | With g++ (GCC) 7.2.1 20170829 (Red Hat 7.2.1-1)                  |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'       |
   ------------------------------------------------------------------

root [0] .L ptdist.C
root [1] ptdist t
Error in <TFile::TFile>: file /storage/mexnica/kitziahc/Unigen/outData/BiBi11Gev_04fm.root does not exist
(ptdist &) @0x2b3b7e221000
root [2] t.Loop();
Warning in <TUnixSystem::SetDisplay>: DISPLAY not set, setting it to 2806:1016:f:473a:794b:f98:3026:3a9f
root [3]
```

Figure: Ejecutar Loop();

Los archivos de salida es un archivo .root con los histogramas de las cantidades calculadas.

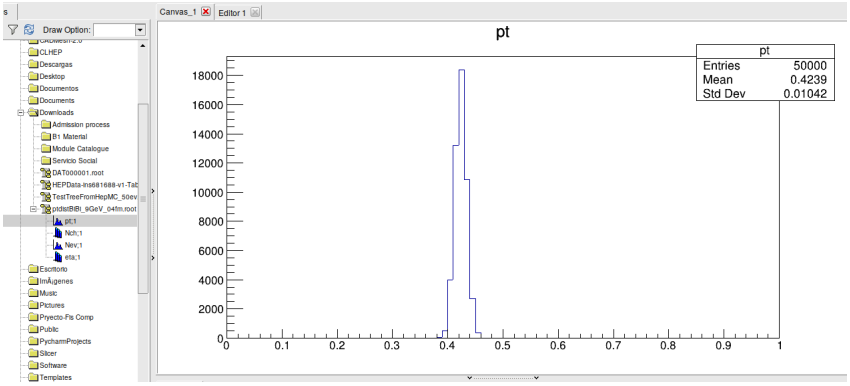


Figure: Distribuciones de pt promedio

Análisis de los resultados

El macro utilizado para el análisis de los dato, *ptAnalysis.C* se encuentra en ▸ ptdist GitHub junto con los datos necesarios para probarlo y el resto de los códigos utilizados en este trabajo.
El macro *ptAnalysis.C* realiza lo siguiente:

- Realizar el fit de la distribución de $\langle p_t \rangle$ a una función gamma de acuerdo al formalismo descrito en [5], ya que se ha observado que puede ser descrita de la forma,

$$f(y) = f_\Gamma(y, \alpha, \beta) = \frac{\beta}{\Gamma(\alpha)}(\beta y)^{\alpha-1} e^{-\beta y} \tag{5}$$

  Los parámetros, α, β, de la distribución gamma están relacionados con la media y la varianza de la distribución de $\langle p_t \rangle$

```
void ptAnalysis(){


//macro para el analisis de los datos producidos
gROOT->SetStyle("Plain");
//load .root file
TFile* f = new TFile("/home/kitzia/Software/ptdist/hydro/ptdistBiBi9GeVHydro_1314fm_nocuts.root");

//Get histograms

TH1D *ptdist = new TH1D(*((TH1D*)f->Get("pt;1"))); // <p_t> vs events

//Fitting
//ptdist->ResetStats(); //binned analysis
//Define Gamma function
auto myfunc =[](double *x, double *p){ return p[0]*ROOT::Math::gamma_pdf(x[0], p[1],p[2]);};
auto f1 =new TF1("f1", myfunc, 0,0,3);
//Fit Gaussian first
ptdist->Fit("gaus");
TF1* g = (TF1*)ptdist->GetListOfFunctions()->FindObject("gaus");
double p0 = g->GetParameter(0);
double p1 = g->GetParameter(1);
double p2 = g->GetParameter(2);
//Fit Gamma Second
f1->SetParameters(5,p1,p2);
f1->SetParName(0,"k");
f1->SetParName(1,"alpha");
f1->SetParName(2, "beta");
f1->SetLineColor(kRed);
f1->SetLineWidth(2);
//Fit
TFitResultPtr fit = ptdist->Fit(f1, "ELSR");
ptdist->Fit("f1", "ELSR");
TF1 *fit2 = ptdist->GetFunction("f1");
//Chisquare test
cout << "error " << ptdist->Chisquare(f1,"R") << endl;
```

Figure: ptAnalysis.C

- En la segunda parte del análisis se obtiene la media, varianza y asimetría de las distribuciones de momento transverso promedio y sus errores estadísticos. La información es entonces guardada en un archivo .txt.

```
double p1 = g->GetParameter(1);
double p2 = g->GetParameter(2);
//Fit Gamma Second
f1->SetParameters(5,p1,p2);
f1->SetParName(0,"k");
f1->SetParName(1,"alpha");
f1->SetParName(2, "beta");
f1->SetLineColor(kRed);
f1->SetLineWidth(2);
//Fit
TFitResultPtr fit = ptdist->Fit(f1, "ELSR");
ptdist->Fit("f1", "ELSR");
TF1 *flt2 = ptdist->GetFunction("f1");
//Chisquare test
cout << "error " << ptdist->Chisquare(f1,"R") << endl;
//Parameters fit
Double_t a = fit->Parameter(1);
Double_t b = fit->Parameter(2);
Double_t k = fit->Parameter(0);
Double_t mean =  a*b;
Double_t sigma = a*b*b;
Double_t skew = 2/TMath::Sqrt(a);
//associated errors (fit)
Double_t a_e = fit2->GetParError(1);
Double_t b_e = fit2->GetParError(2);
Double_t k_e = fit2->GetParError(0);
Double_t mean_error = (TMath::Sqrt(TMath::Power(a_e/a,2)+TMath::Power(b_e/b,2)))*mean;
Double_t sigma_error = (TMath::Sqrt(TMath::Power(a_e/a,2)+ TMath::Power(2*b_e/b,2)))*sigma;
//Double_t skew_error = (a/a_e);
//Parameter distribution
Double_t pt_mean = ptdist->GetMean();
Double_t pt_sigma = ptdist->GetStdDev()*ptdist->GetStdDev();
Double_t pt_skew = ptdist->GetSkewness();
//asociated errots (distribution)
Double_t ptMean_error = ptdist->GetMeanError(); //error del promedio en x
```

Figure: ptAnalysis.C

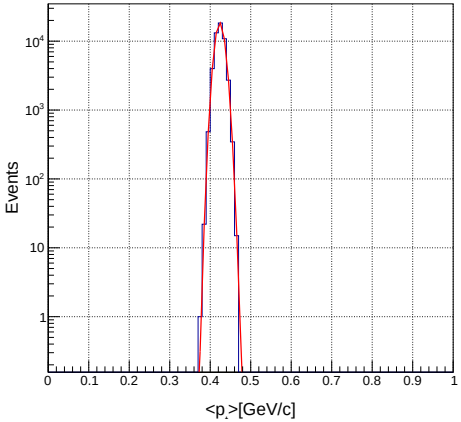
```
//associated errors (distribution)
Double_t ptMean_error = ptdist->GetMeanError(); //error del promedio en x
Double_t ptSigma_error = 2*ptdist->GetStdDev(11); //error de la devStd en x
Double_t ptSkew_error = ptdist->GetSkewness(11); //error del Skew en x

//Save info
TFile* myFile = new TFile("/home/kitzia/Software/ptdist/hydro/fit_info_1314fm.root", "recreate");
fit->Write();
myFile->Close();

 fstream fs;
        fs.open("/home/kitzia/Software/ptdist/hydro/fit_info_1314fm.txt", ios::out);
        string lin;
// Make a backup of stream buffer
        streambuf* sb_cout = cout.rdbuf();
        streambuf* sb_cin = cin.rdbuf();
// Get the file stream buffer
        streambuf* sb_file = fs.rdbuf();
// Now cout will point to file
        cout.rdbuf(sb_file);
// everything printed after this line will go to the fs
fit->Print("V");

cout << "mean(mu) " << mean << " error " << mean_error << " pt_mean " << pt_mean << " error " << ptMean_error << endl;

cout << "Var(sigma) " << sigma << " error " << sigma_error << " p_t sigma " << pt_sigma << " error " << ptSigma_error << " Resta " <<
TMath::Abs(sigma - pt_sigma) << " error " << ptSigma_error + sigma_error <<endl;

cout << "StdDev " << TMath::Sqrt(sigma) << " error " << sigma_error/2  << " p_t StdDev " << ptdist->GetStdDev() << " error " << ptdist-
>GetStdDev(11) << endl;

cout << "skewness(s) " << skew  << " p_t skewness " << pt_skew << " error " << ptSkew_error << endl;

cout << "alpha(fit) " << a << " error " << a_e << " alpha(dist) " << (pt_mean*pt_mean)/pt_sigma  <<" beta(fit) " << b << " error " << b_e << "
beta(dist) " << pt_sigma/pt_mean  << endl;
// get the previous buffer from backup
```

Figure: ptAnalysis.C

```
// get the previous buffer from backup
        cout.rdbuf(sb_cout);
// everything printed after this line will go to whereever it went before
        fs.close();
        delete myFile;

//Print information(on screen)

cout << "mean(mu) " << mean << " error " << mean_error << " pt_mean " << pt_mean << " error " << ptMean_error << endl;

cout << "Var(sigma) " << sigma << " error " << sigma_error << " p_t sigma " << pt_sigma << " error " << ptSigma_error << " Resta " <<
TMath::Abs(pt_sigma - sigma) << " error " << ptSigma_error + sigma_error <<endl;

cout << "StdDev " << TMath::Sqrt(sigma) << " error " << sigma_error/2  << " p_t StdDev " << ptdist->GetStdDev() << " error " << ptdist-
>GetStdDev(11) << endl;

cout << "skewness(s) " << skew  << " p_t skewness " << pt_skew << " error " << ptSkew_error << endl;

cout << "alpha(fit) " << a << " error " << a_e << " alpha(dist) " << (pt_mean*pt_mean)/pt_sigma  <<" beta(fit) " << b << " error " << b_e << "
beta(dist) " << pt_sigma/pt_mean  << endl;

//Plot pt distributions
TCanvas* c1 = new TCanvas("c1", "c1", 800, 800);
c1->SetGrid();
gStyle->SetOptFit(false);
gStyle->SetOptStat(false);
c1->SetRightMargin(0.05);
c1->SetTopMargin(0.1);
c1->SetFillColor(0);
gPad->SetLogy();
ptdist->Draw();
//ptdist->Draw("e1p same");
ptdist->SetTitle("UrQMD Bi+Bi #sqrt{s_{NN}} = 9GeV, 90-100%");//");//, , 0-10%");
ptdist->GetXaxis()->SetTitle("<p_{t}>[GeV/c]");
ptdist->GetYaxis()->SetTitle("Events");
ptdist->GetXaxis()->CenterTitle(true);
```

Figure: ptAnalysis.C

UrQMD Bi+Bi $\sqrt{s_{NN}}$ = 9GeV, 0-10%




```
******************************************
Minimizer is Minuit / Migrad
MinFCN                    =      4.49824
Chi2                      =      7.87708
NDf                       =           97
Edm                       =  1.88209e-07
NCalls                    =         1182
k                         =      500.001   +/-   2.23551     -2.23337     +2.23876     (Minos)
alpha                     =      1538.54   +/-   8.50739     -9.70637     +9.75273     (Minos)
beta                      =  0.000275539   +/-   1.52392e-06   -1.73581e-06 +1.74975e-06  (Minos)

Covariance Matrix:

                          k       alpha         beta
k                      4.9975  2.4085e-05 -4.7453e-12
alpha            2.4085e-05      72.376 -1.2962e-05
beta            -4.7453e-12 -1.2962e-05  2.3223e-12

Correlation Matrix:

                          k       alpha         beta
k                         1  1.2664e-06 -1.3929e-06
alpha            1.2664e-06           1    -0.99979
beta            -1.3929e-06    -0.99979           1
mean(mu) 0.423928 error 0.00332104 pt_mean 0.42392 error 4.65954e-05
Var(sigma) 0.000116809 error 1.44696e-06 p_t sigma 0.000108557 error 6.58959e-05 Resta 8.2521e-06 er
StdDev 0.0108078 error 7.23478e-07 p_t StdDev 0.0104191 error 3.29479e-05
skewness(s) 0.0509889 p_t skewness 0.03699024 error 0.0189545
alpha(fit) 1538.54 error 8.52177 alpha(dist) 1655.43 beta(fit) 0.000275539 error 1.5265e-06 beta(dist
```

Gracias por su atención :)

References I

J. Adams et al.
Incident energy dependence of pt correlations at RHIC.
Phys. Rev. C, 72:044902, 2005.

N. Geraksiev.
The nuclotron-based ion collider facility project. the physiscs programme for the multi-purpose detector.
Journal of Physics: Conference Series, 1023:12030, 2018.

Tapan K. Nayak.
Probing the QCD phase structure using event-by-event fluctuations.
J. Phys. Conf. Ser., 1602(1):012003, 2020.

References II


John F. Novak.
Energy Dependence of Fluctuation and Correlation Observables of Transverse Momentum in Heavy-Ion Collisions.
PhD thesis, Michigan State University, 7 2013.


M.J. Tannenbaum.
The distribution function of the event-by-event average pt for statistically independent emission.
Physics Letters B, 498(1-2):29–34, 2001.