Fit B spectrum(by bayes/th2) with blast wave function

李东升

The Blast-Wave (BW) model is extensively used to study the particle kinetic freeze-out properties [27, 41]. Assuming a hard-sphere uniform particle source with a kinetic freeze-out temperature $T_{\rm kin}$ and a transverse radial flow velocity β, the particle transverse momentum spectral shape is given by [42]:

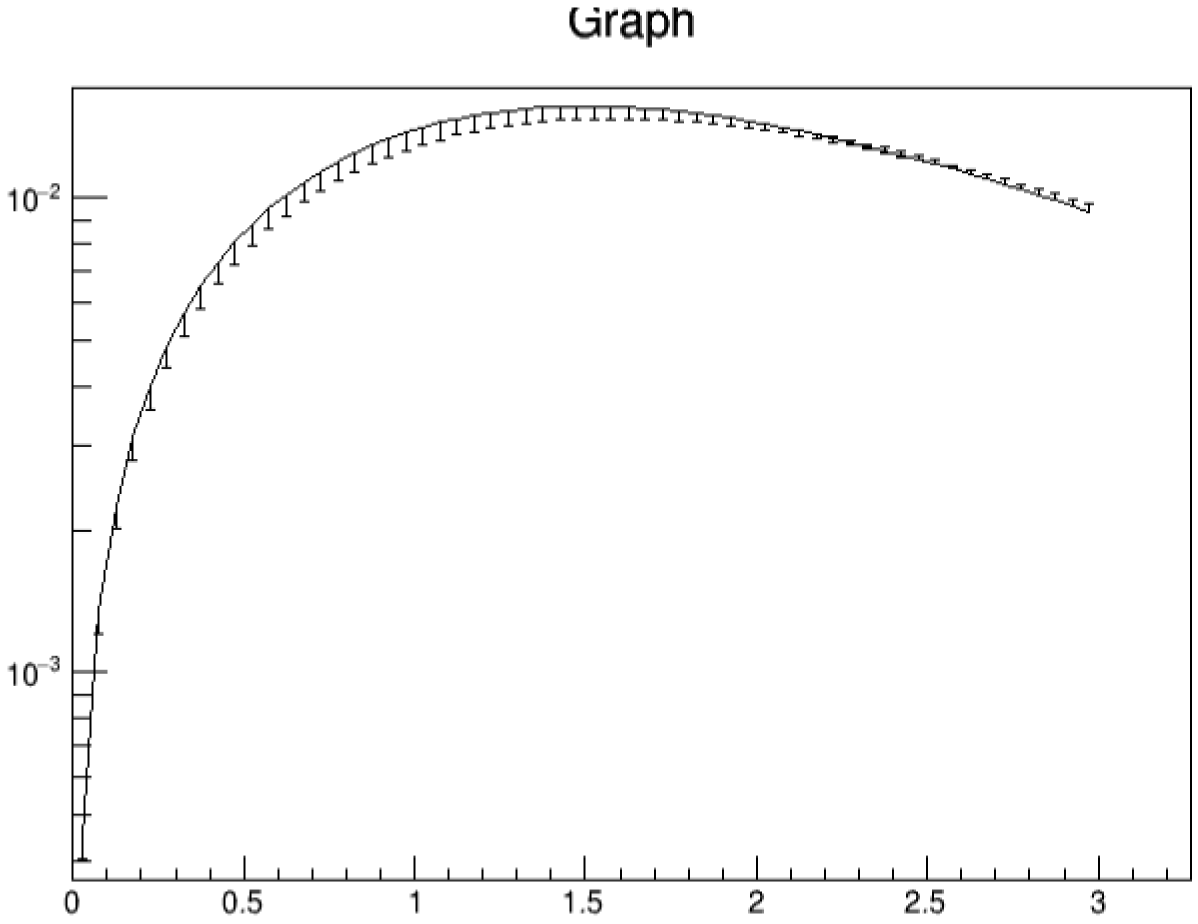
$$\frac{dN}{p_T dp_T} = \frac{dN}{m_T dm_T} \propto \int_0^R r dr m_T I_0\left(\frac{p_T \sinh\rho}{T_{\rm kin}}\right) K_1\left(\frac{m_T \cosh\rho}{T_{\rm kin}}\right), \tag{9}$$

where $\rho = \tanh^{-1}\beta$, and I_0 and K_1 are the modified Bessel functions. The flow velocity profile is taken as:

$$\beta = \beta_s \left(\frac{r}{R}\right)^n, \tag{10}$$

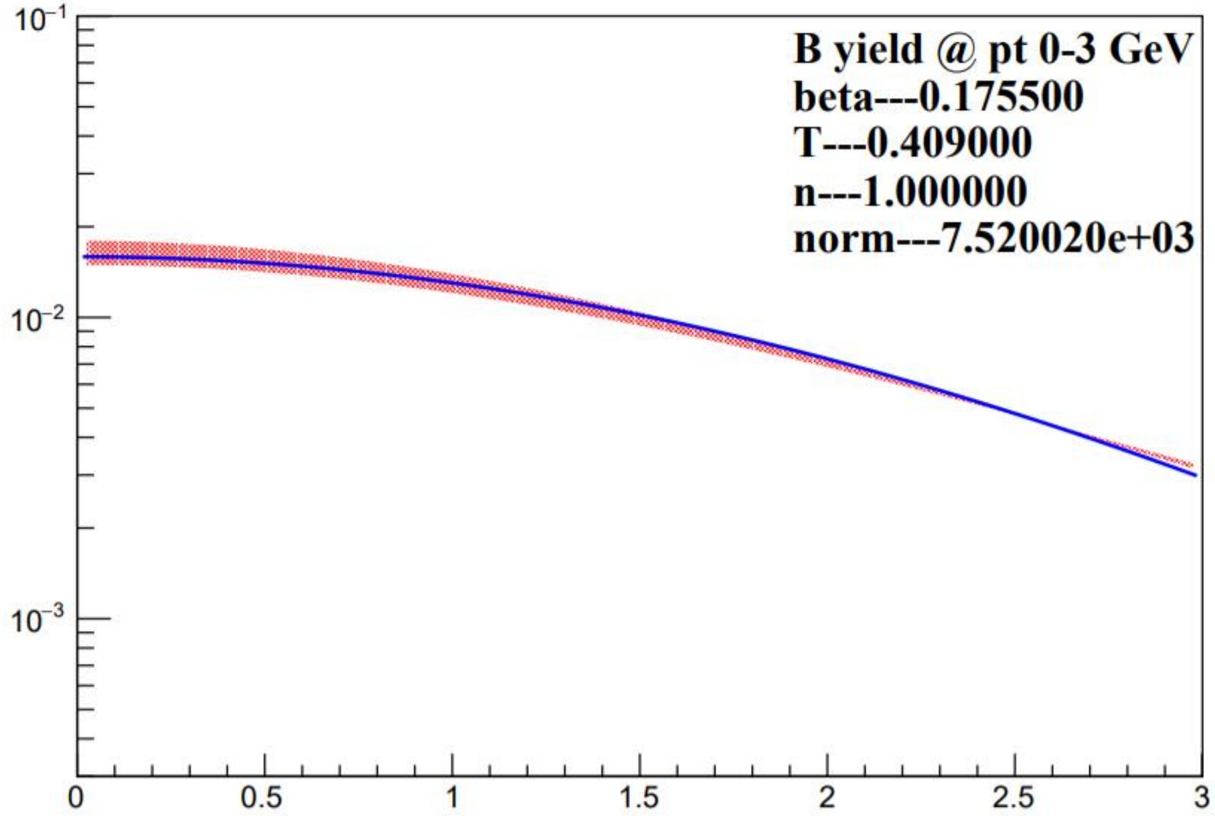
where β_s is the maximum velocity at the surface and r/R is the relative radial position in the thermal source. The choice of R only affects the overall spectrum magnitude while the spectrum shape constrains the three free parameters $T_{\rm kin}$, $\langle\beta\rangle = 2/(2+n)\beta_s$, and n.

```
//////////////////////////////////////////////////////////////////////////////
// Integrand for boltzman-gibbs blast wave free n, beta_r = beta_s * (r/R)^n
//////////////////////////////////////////////////////////////////////////////
Double_t IntegrandBGN(const double *x, const double *p)
{

  double x0   = x[0];
  double mass = p[0];
  double pT   = p[1];
  double beta = p[2];
  double T    = p[3];
  double n    = p[4];
  double mT   = TMath::Sqrt(mass*mass + pT*pT);

  double rho0 = TMath::ATanH(beta*pow(x0,n));
  double a0   = pT*TMath::SinH(rho0)/T;
  double a1   = mT*TMath::CosH(rho0)/T;

  return x0*mT*TMath::BesselI0(a0)*TMath::BesselK1(a1);
}


//////////////////////////////////////////////////////////////////////////////
// Implementation of B-G blast wave for (1/pt dNdpt)
//////////////////////////////////////////////////////////////////////////////
Double_t StaticBGdNdPtN(const double *x, const double *p)
{
  double pT   = x[0];
  double mass = p[0];
  double beta = p[1];
  double T    = p[2];
  double n    = p[3];

  TF1 *fIntBG = 0;
  if(!fIntBG) fIntBG = new TF1("fIntBG", IntegrandBGN, 0, 1, 5);

  fIntBG->SetParameters(mass, pT, beta, T, n);
  //  fIntBG->SetNpx(100000);
  return fIntBG->Integral(0,1) * p[4]; //p[4] norm
}
```



$$\frac{dN}{dp_T}, 0 - 10\%\ \mathrm{B\ spectrum\ @\ 5.02\ TeV\ Pb + Pb}$$

by bayesian method



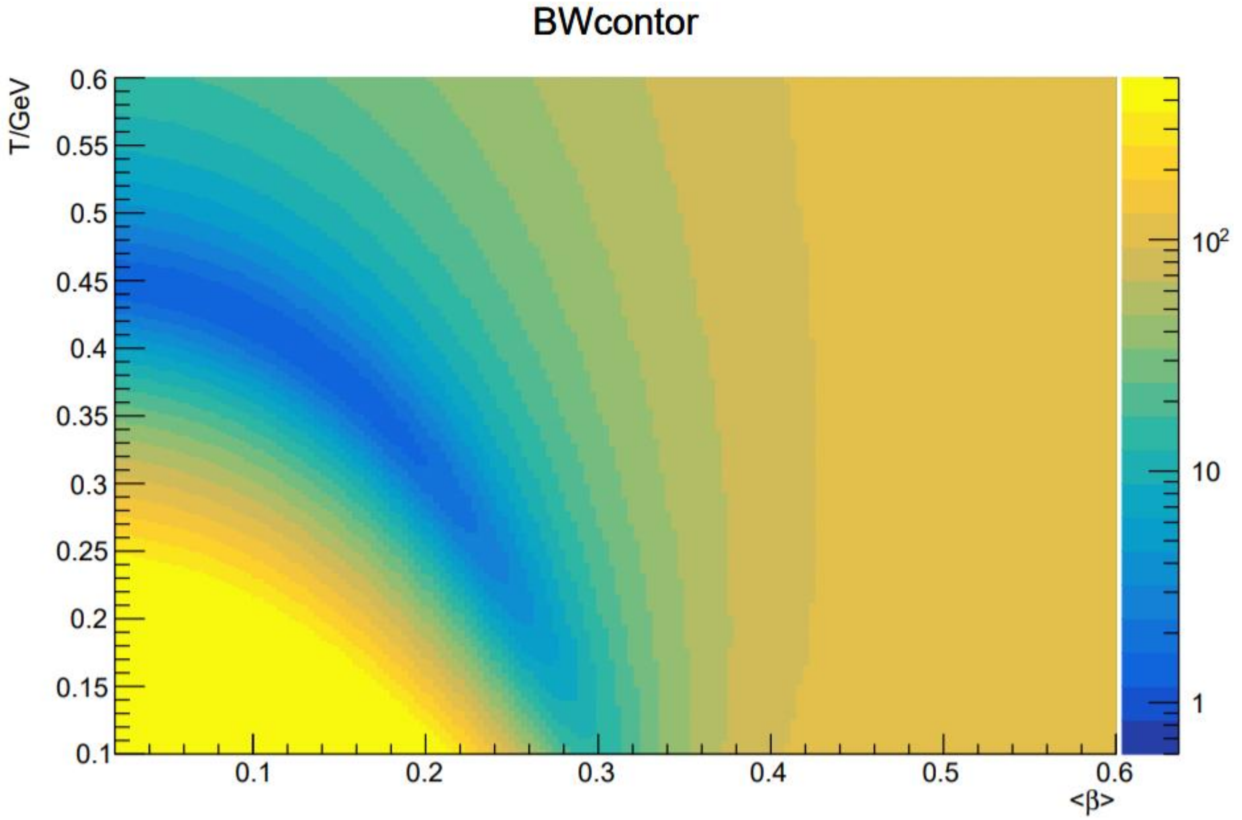
$$\frac{dN}{p_T dp_T}\ (p_T = 0 - 3\ \mathrm{GeV})$$

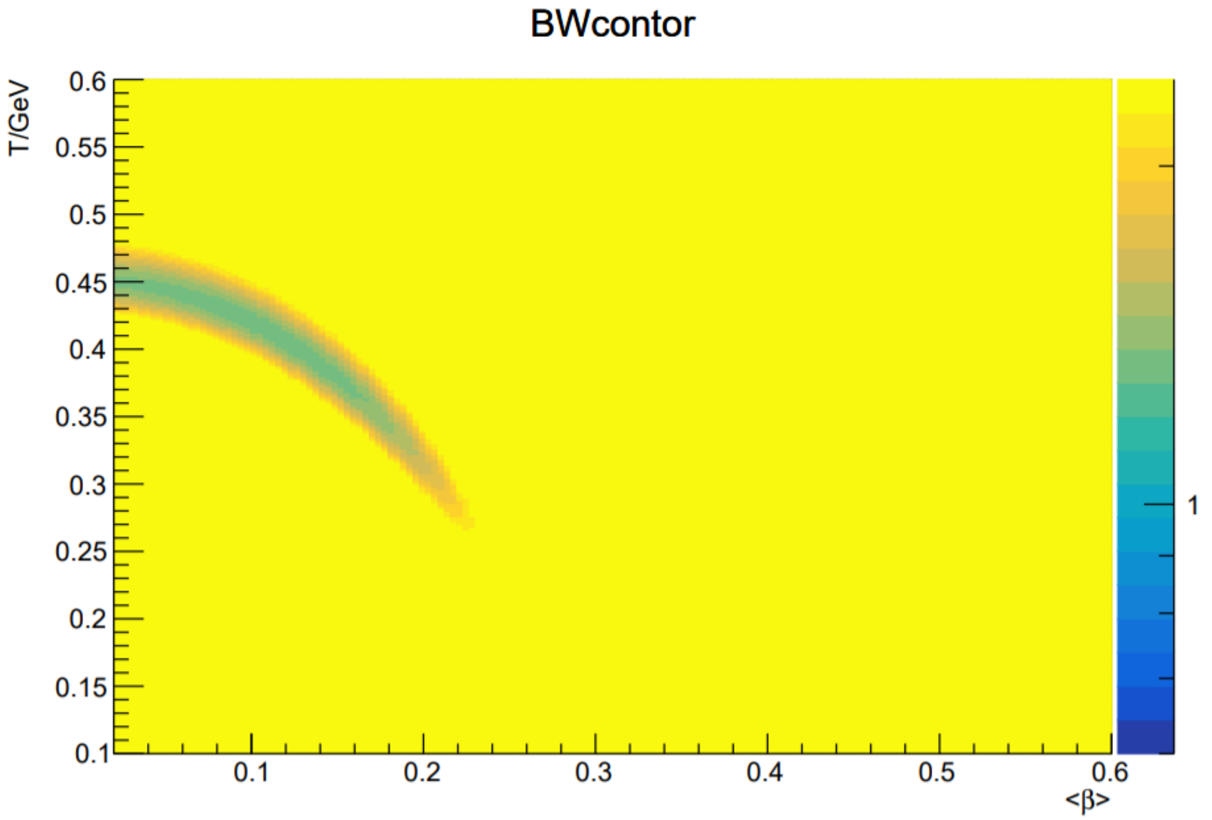
fitted by blast wave function

0-10%, bayes

1.Fix n(=1.)
2.Fix&Scan beta(0.03-0.9)
T(0.1-0.6)
3.Fit get norm and chi2

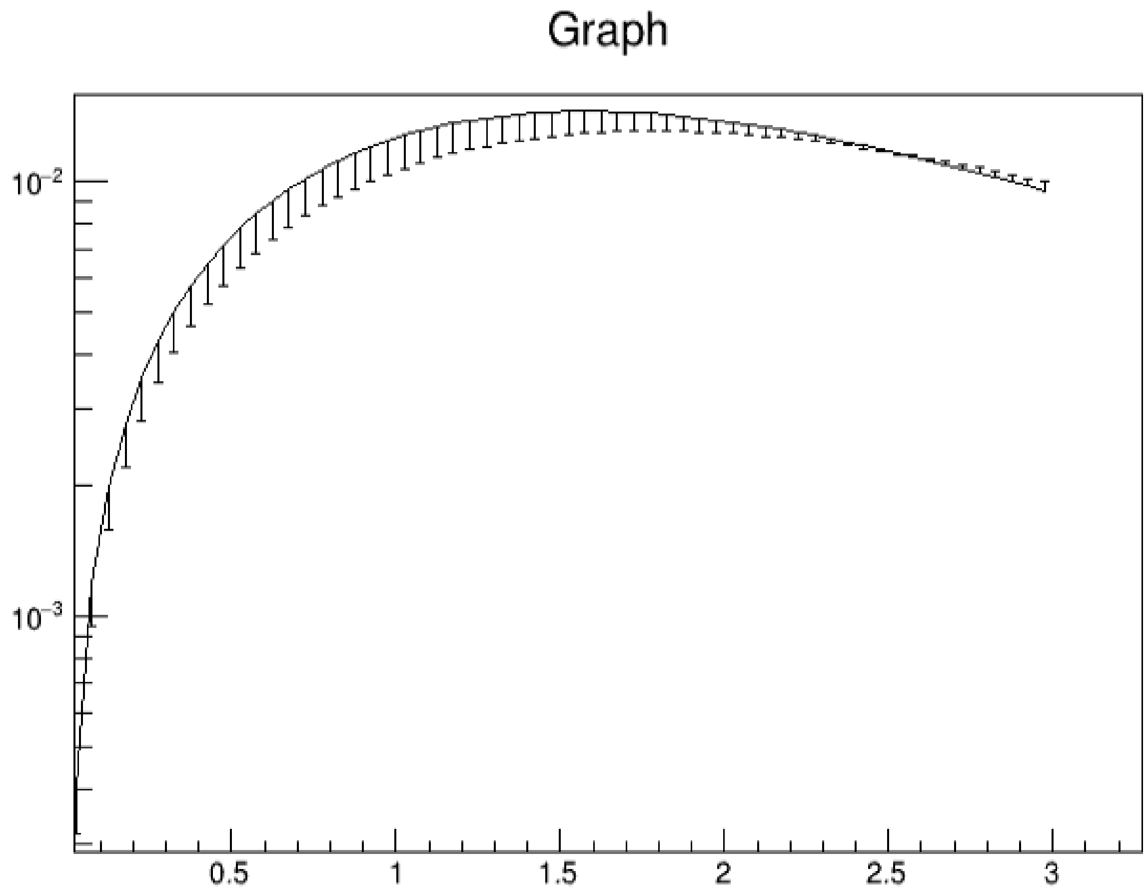
4.Fill x=beta*2./(2.+n)
y=T
z=chi2/ndf (ndf=60)





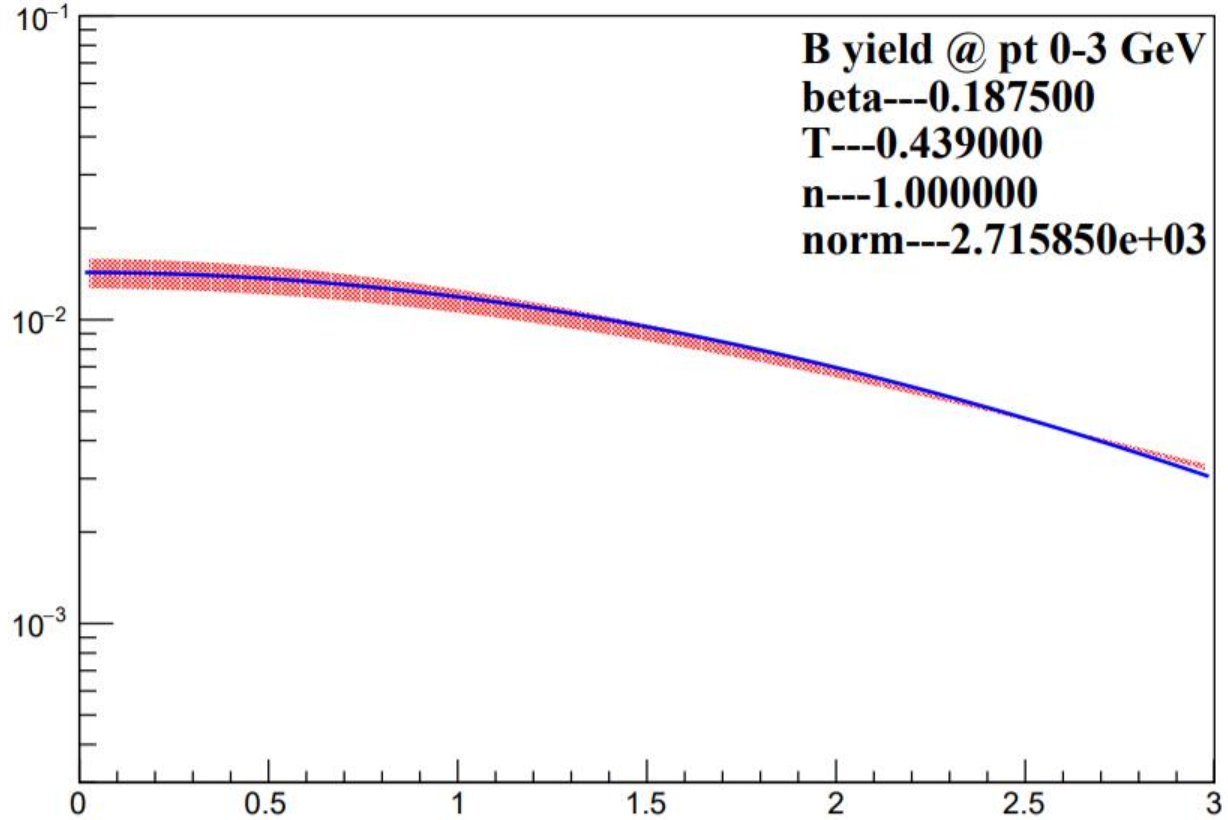
Left: h->GetZaxis()->SetRangeUser(6.e-1,5.e2);
Right: h->GetZaxis()->SetRangeUser(6.e-1,64.6554/60+1.31068);//64.6554(chi2min/ndf=1.31068)

```
root [0] 1-TMath::Prob(64.6554,60)
(double) 0.68260006
```



$\frac{dN}{dp_T}$, $30-50\%$ B spectrum @ 5.02 TeV Pb + Pb

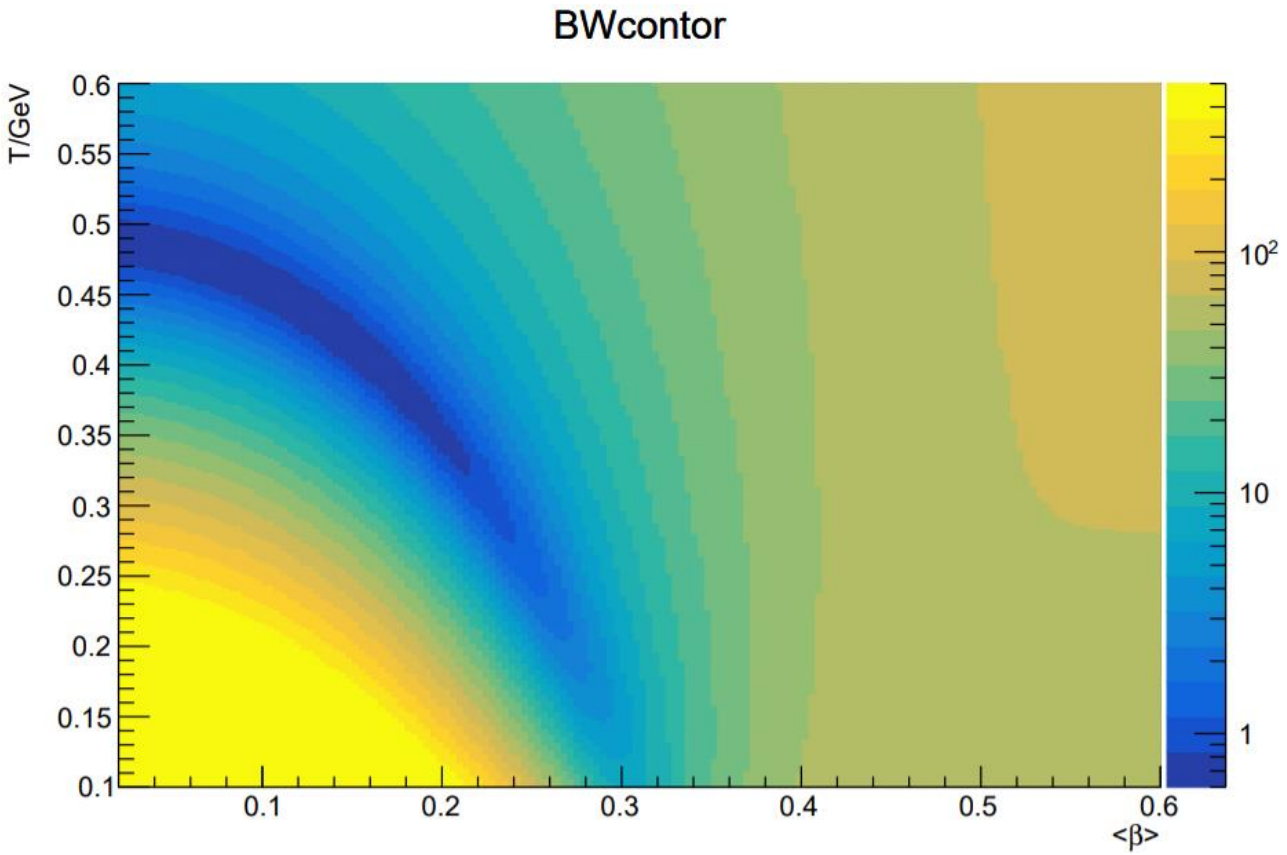
by bayesian method

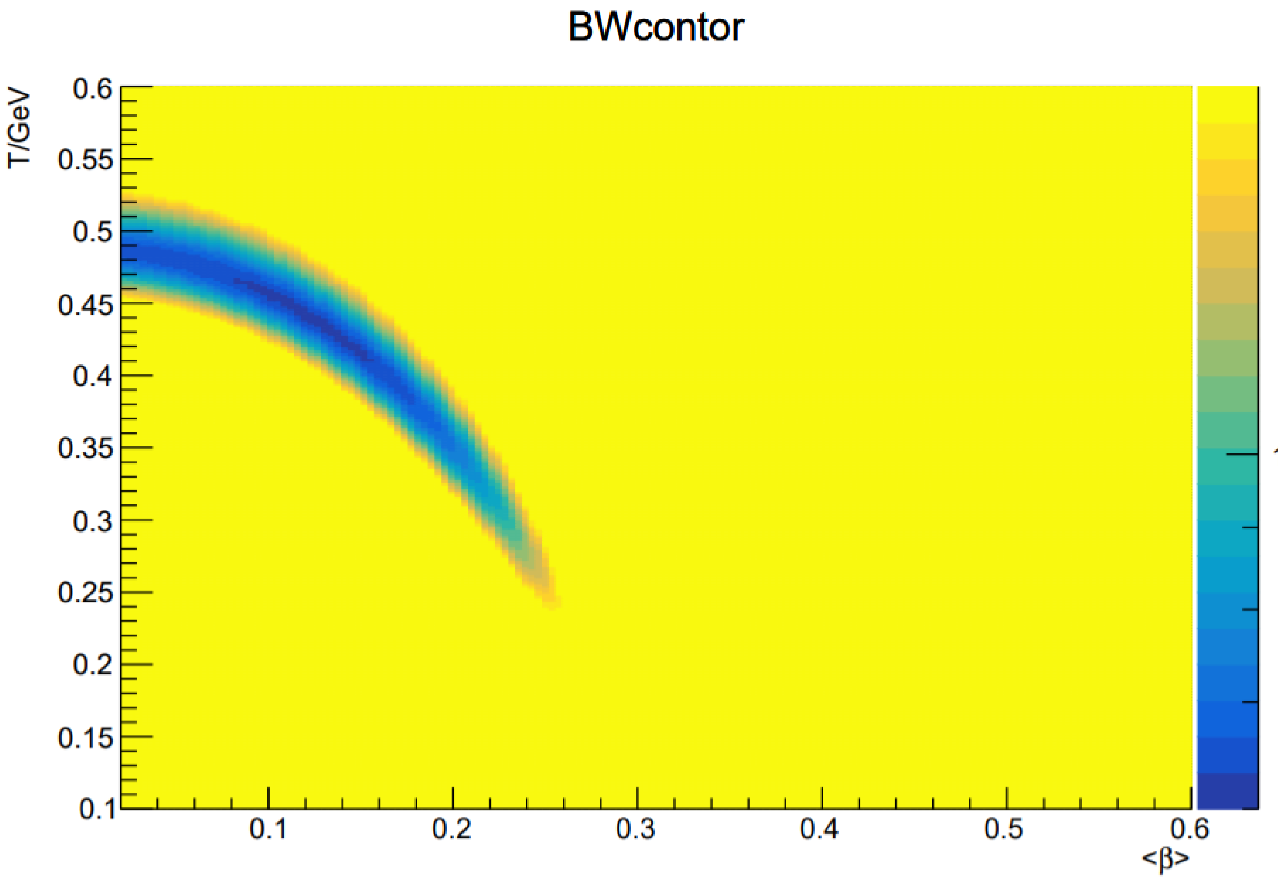


30-50%, bayes

1.Fix n(=1.)
2.Fix&Scan beta(0.03-0.9)
T(0.1-0.6)
3.Fit get norm and chi2

4.Fill x=beta*2./(2.+n)
y=T
z=chi2/ndf (ndf=60)

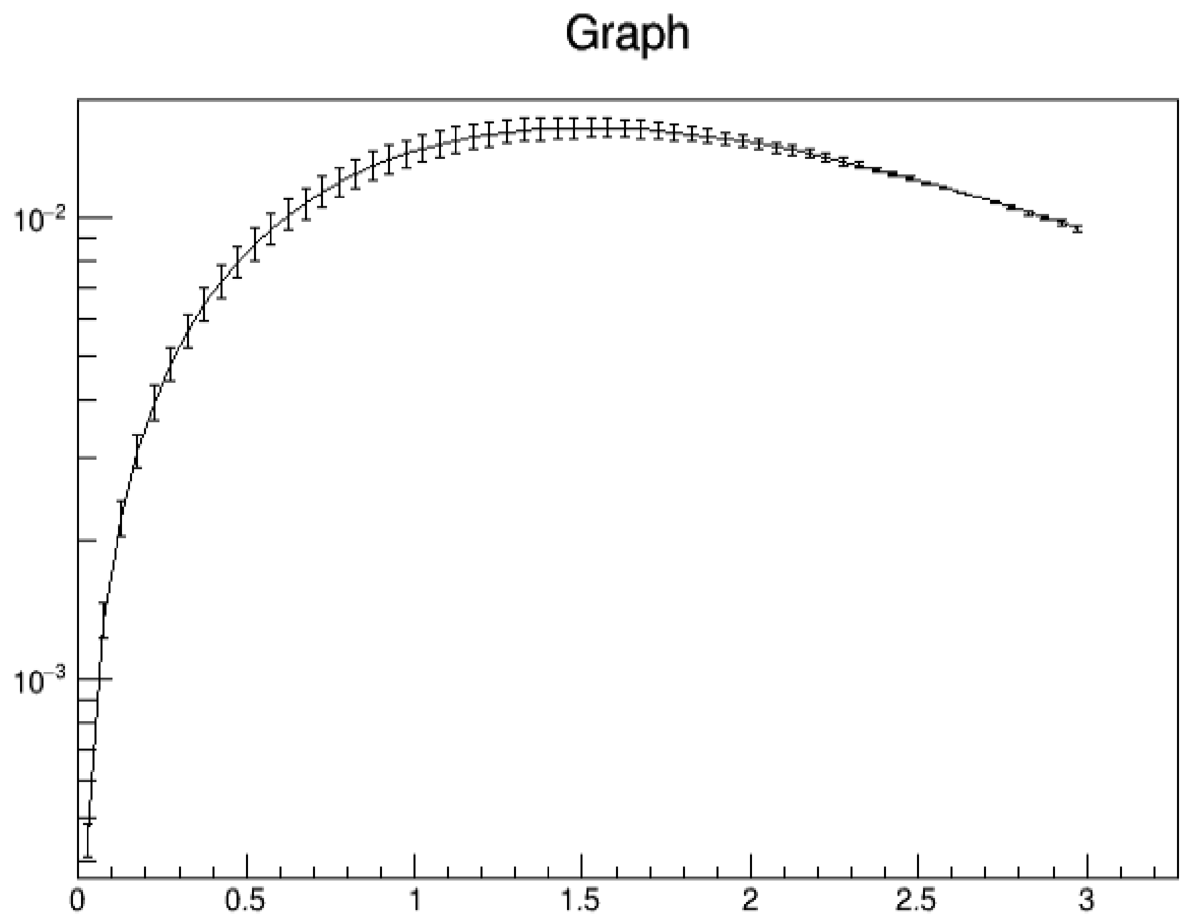




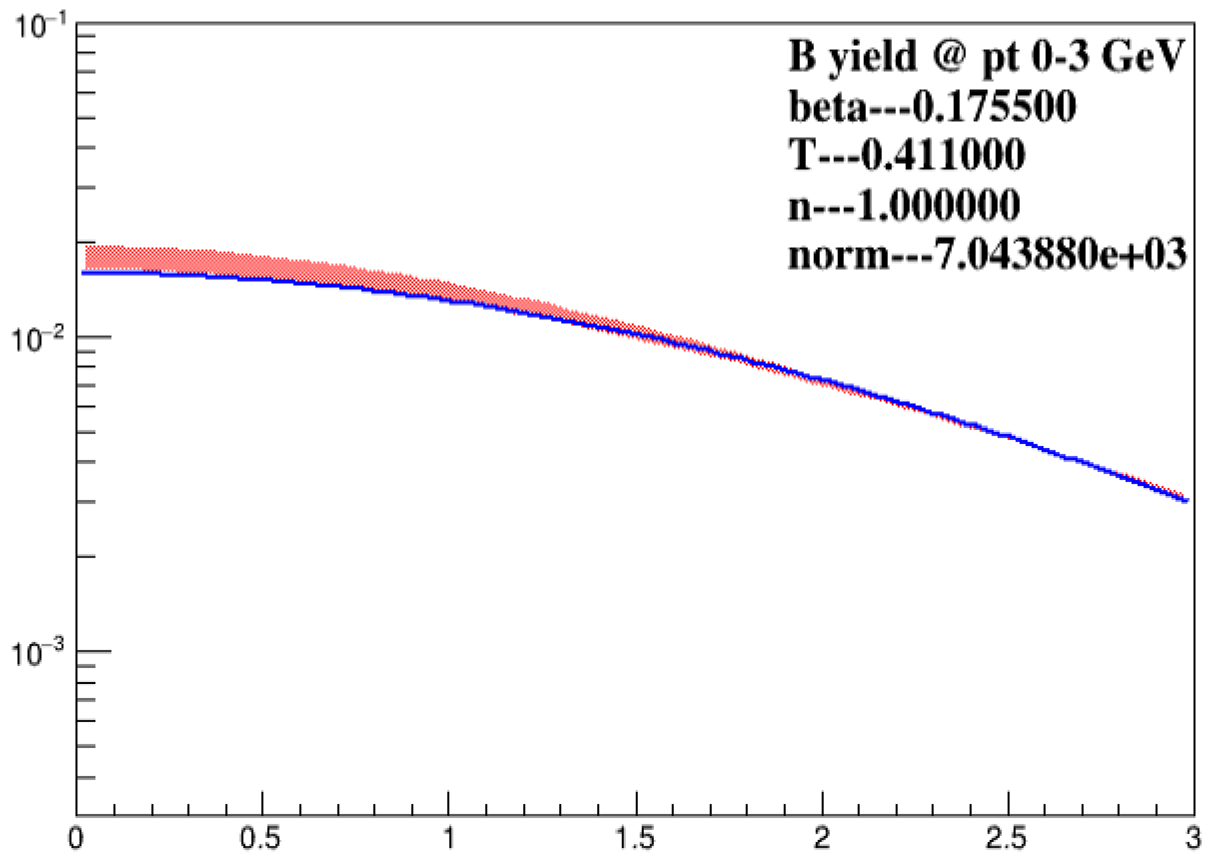
Left: h->GetZaxis()->SetRangeUser(6.e-1,5.e2);/

Right: h->GetZaxis()->SetRangeUser(6.e-1,64.6554/60+0.621829);//64.6554(chi2min/ndf=0.621829)

```
root [0] 1-TMath::Prob(64.6554,60)
(double) 0.68260006
```



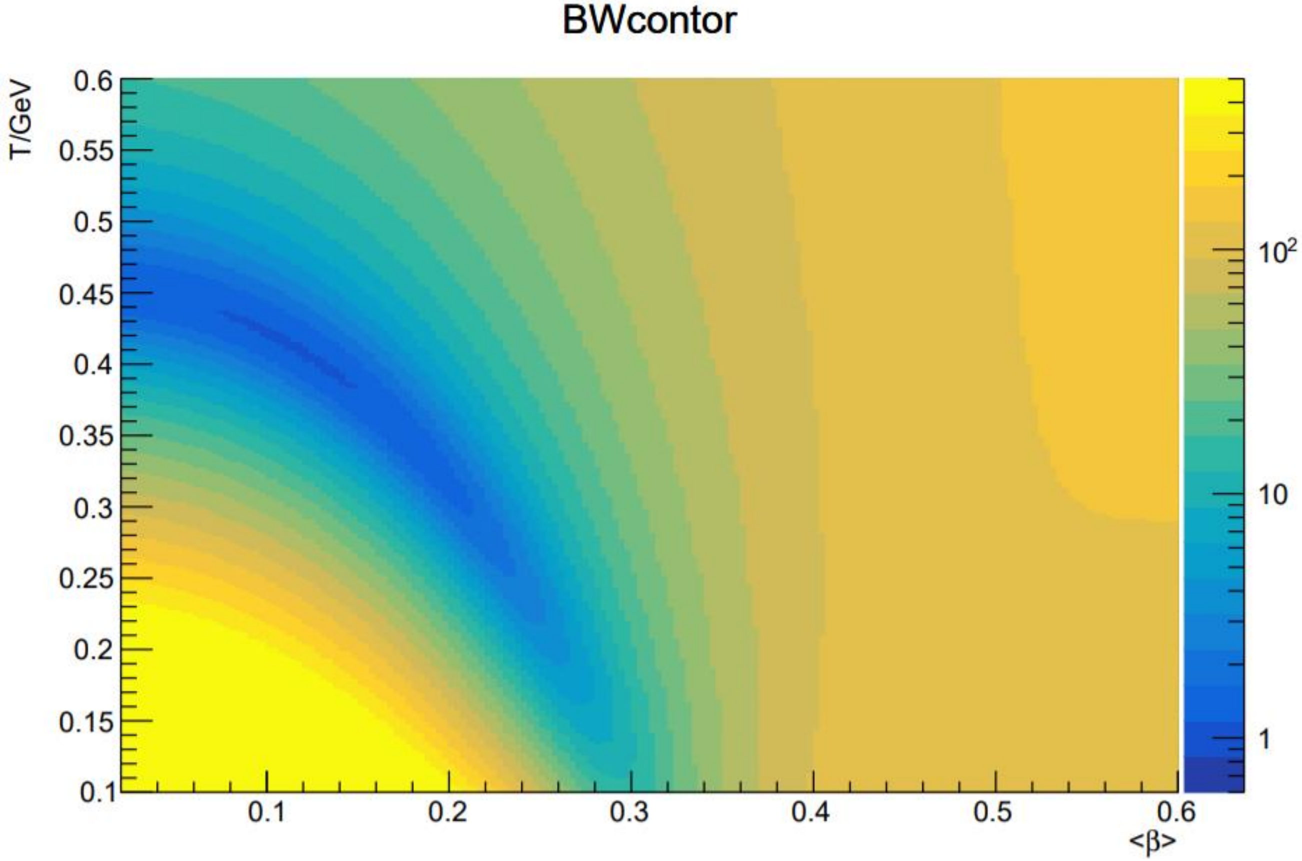
$\frac{dN}{dp_T}$, 0 − 10% B spectrum @ 5.02 TeV Pb + Pb
by th2 method

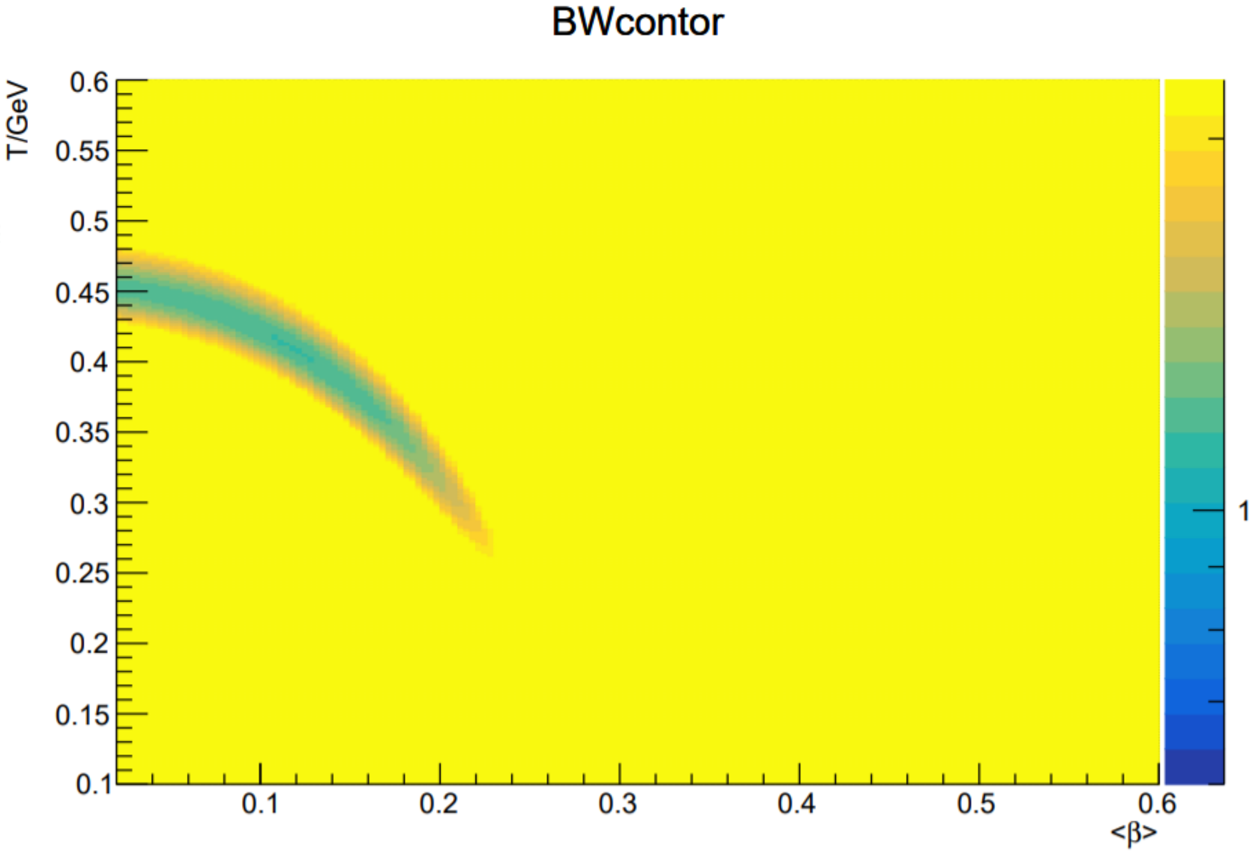


0-10%, th2

1.Fix n(=1.)
2.Fix&Scan beta(0.03-0.9)
T(0.1-0.6)
3.Fit get norm and chi2

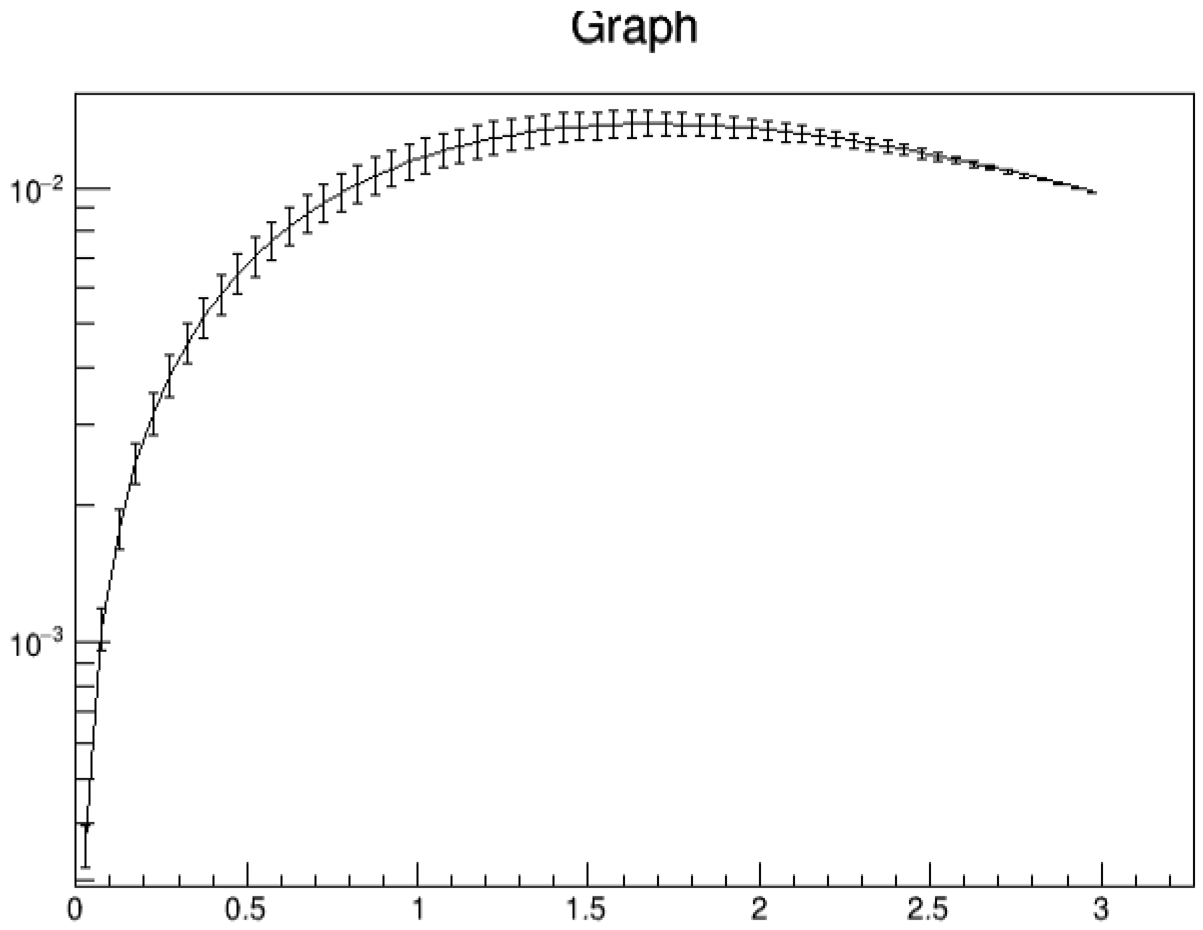
4.Fill x=beta*2./(2.+n)
y=T
z=chi2/ndf (ndf=60)





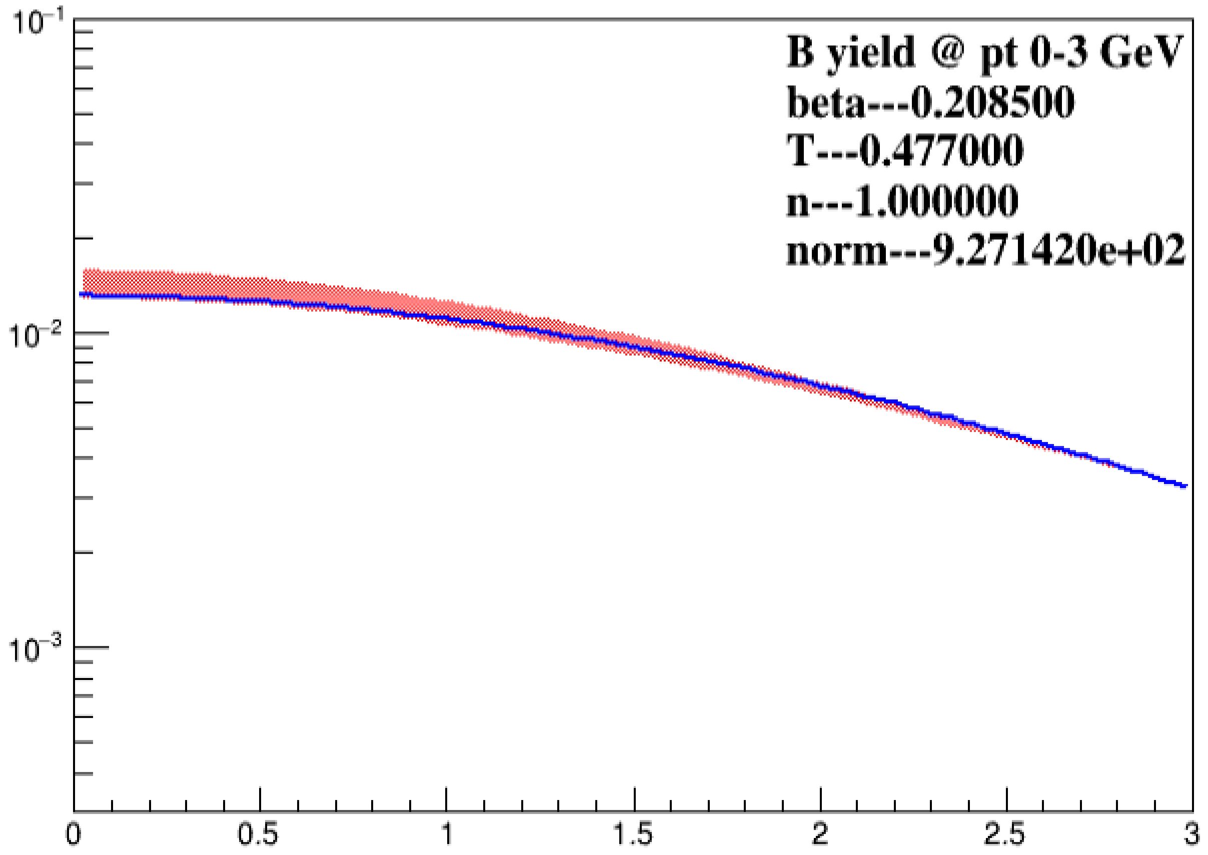
Left: h->GetZaxis()->SetRangeUser(6.e-1,5e2);
Right: h->GetZaxis()->SetRangeUser(6.e-1,64.6554/60+1.15366);//64.6554(chi2min/ndf=1.15366)

```
root [0] 1-TMath::Prob(64.6554,60)
(double) 0.68260006
```



$\frac{dN}{dp_T}$, $30 - 50\%$ B spectrum @ 5.02 TeV Pb + Pb

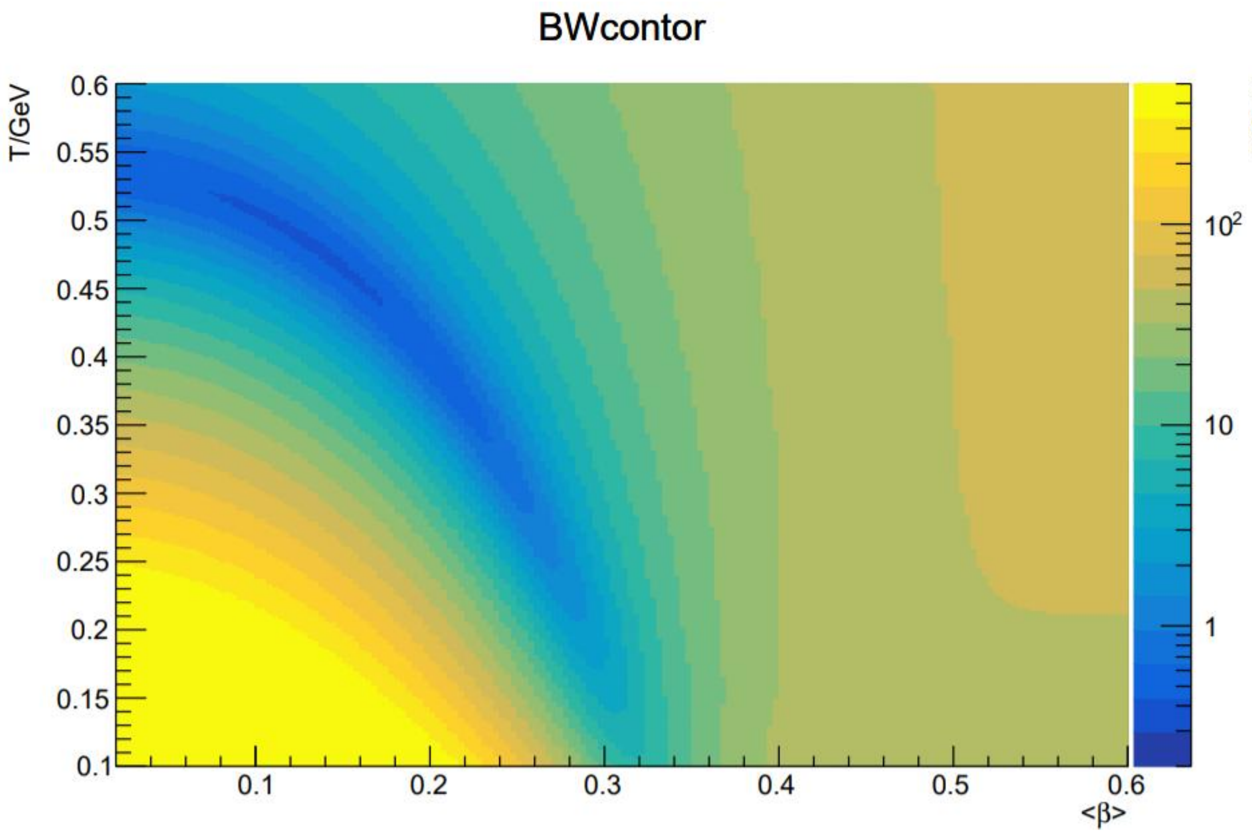
by th2 method

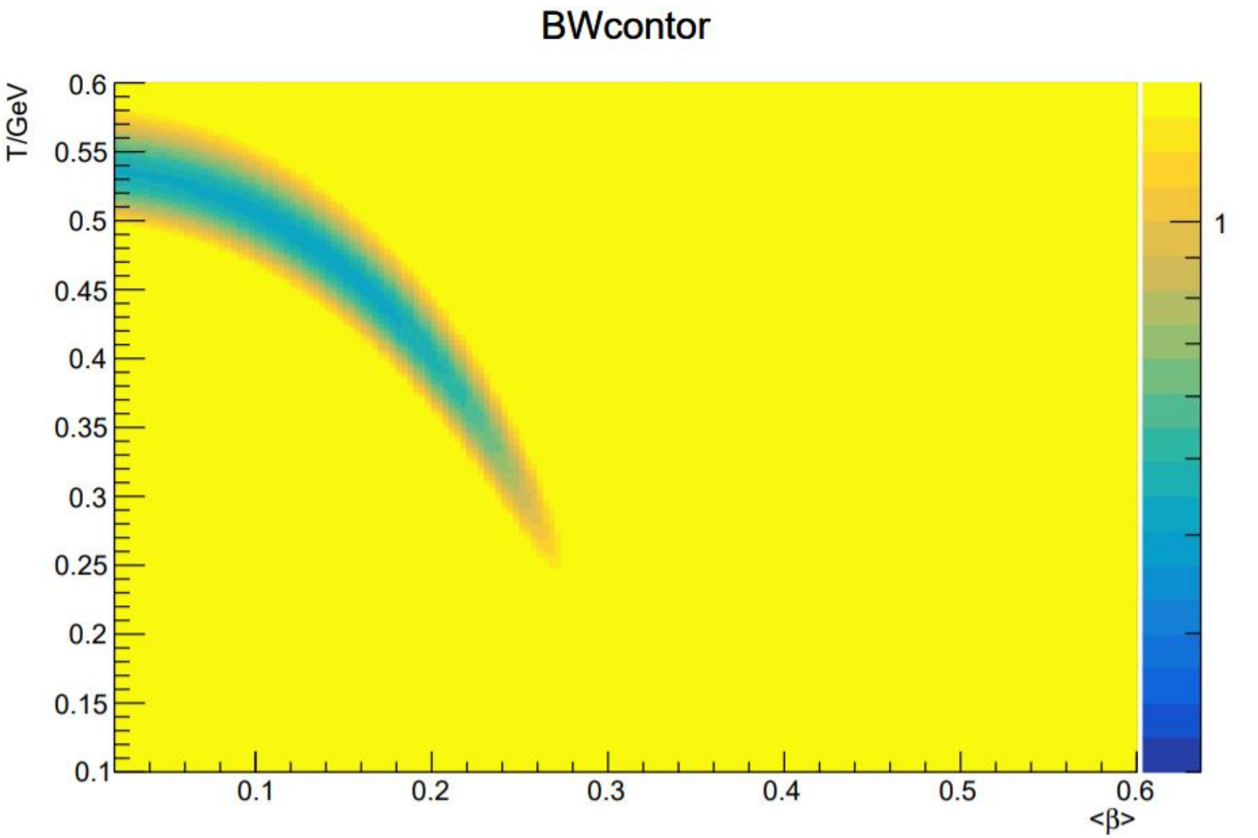


30-50%, th2

1.Fix n(=1.)
2.Fix&Scan beta(0.03-0.9)
T(0.1-0.6)
3.Fit get norm and chi2

4.Fill x=beta*2./(2.+n)
y=T
z=chi2/ndf (ndf=60)





Left: h->GetZaxis()->SetRangeUser(2.e-1,5e2);
Right: h->GetZaxis()->SetRangeUser(2.e-1,64.6554/60+0.423552);//64.6554(chi2min/ndf=0.423552)

```
root [0] 1-TMath::Prob(64.6554,60)
(double) 0.68260006
```

	Bayesian method		Th2 method	
cent	0-10	30-50	0-10	30-50
<beta>	<0.232	<0.167	<0.232	<0.278
T/GeV	0.259-0.481	0.363-0.529	0.257-0.483	0.239-0.583