Utrecht University

Institute for Subatomic Physics

The Chiral Magnetic Effect in Heavy-Ion Collisions

Author:
S. de Vries

Supervisors:
J. Margutti MSc
Prof Dr. R.J.M. Snellings

13th January 2016

Abstract

We study the Chiral Magnetic Effect (CME) in heavy-ion collisions at a centre of mass energy of $\sqrt{2.76}\ TeV$. We use a Blast-Wave Model incorporating Local Charge Conservation to simulate these collisions. We find good agreement with charge-independent global observables, such as charged particle yield spectra and the elliptic flow. By making use of Q-vectors we measure two charge-dependent correlations and find them to be a factor of approximately 10 and 7 times smaller than values measured by ALICE at the LHC, while being of similar shape. This indicates that at least part of the measured correlators can be explained by other sources, one of which could be the CME. Possible improvements of the model and, in addition, the research of the CME in general are discussed.

Contents

1 Introduction

Research in the field of high-energy particle physics is currently largely focussed on studying interactions of matter in relativistic collisions, analysing the products of these interactions and comparing results to theory predictions. At present, the highest collision energies of such particle collisions (sometimes referred to as events in this thesis) are conducted at the Large Hadron Collider (LHC) found at CERN.

At the LHC, seven experiments use detectors for analysis of particle collisions, from these 'A Large Ion Collider Experiment' (ALICE) is of interest to us. Its purpose is to study strongly interacting matter at enormous energy densities, allowing a for a Quark Gluon Plasma (QGP) to form.

The QGP is a phase of matter in which quarks and gluons, fundamental particles of which baryonic matter consists, become deconfined. This state can be achieved by putting a gigantic amount of energy into a system to raise the temperature to values theorized to have been present in the early stages of the universe, a fraction of a second after the Big Bang. These energy densities can currently only be reached in large particle accelerators. At ALICE, the evolution and temperature dependence of the QGP is studied, by measuring the properties of the matter that forms when the QGP cools down. There is much yet to be discovered about the QGP and it is a subject of great importance in particle physics.

Before the detector was temporarily shut down to receive a hardware upgrade, Lead-Lead collisions were measured at ALICE, at a centre of mass energy of $\sqrt{2.76}\,TeV$ [1]. Here two Lead-ions are accelerated to velocities of only a fraction less than the speed of light and then made to collide at the centre of the ALICE detector.

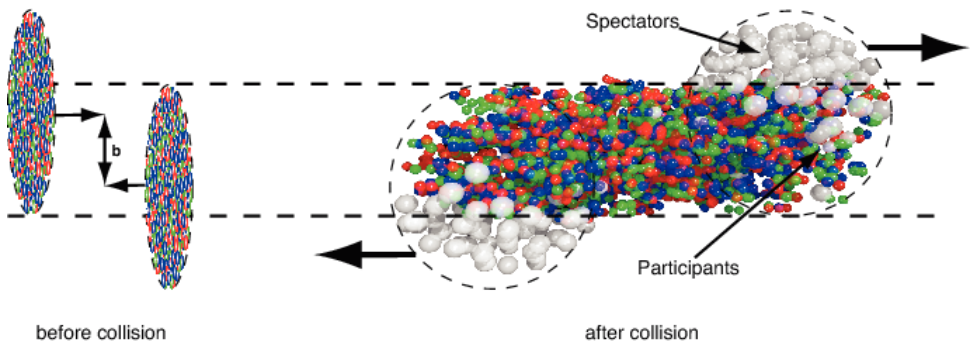

Figure 1.1: Illustration of a heavy-ion collision, demonstrating the concept of centrality, taken from Ref. [9]

When heavy-ions collide head-on, a key difference in each collision is the area of impact between the particles involved. This is commonly referred to as the centrality of the collision, which can be related to the impact parameter b [9]. This has been illustrated in Figure 1.1, with b the centre-to-centre distance of the heavy-ions. Because

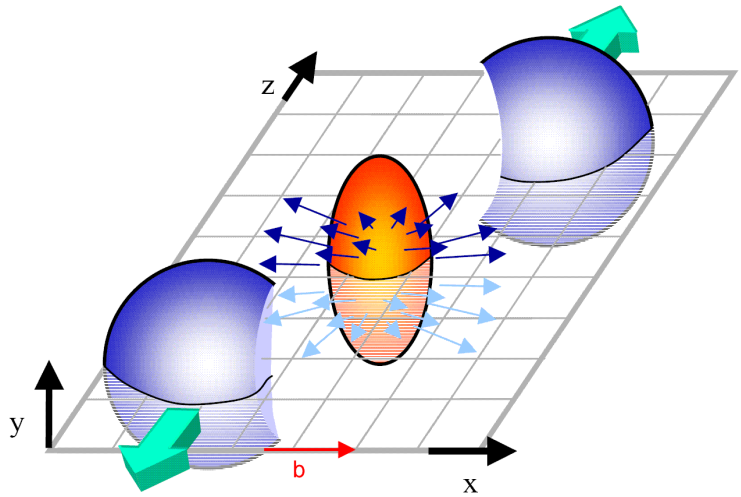

Figure 1.2: The Reaction Plane in a heavy-ion collosion, spanned by the beam direction z and the impact parameter b [6]

collisions at different centralities result in different particle multiplicities, virtually all observables depend on them and measurements are usually grouped by centrality bin.

Another important characteristic of a particular event is the orientation of the reaction plane. This we define as the plane spanned by the direction of the heavy-ion beams as the z-axis and the impact parameter vector b as the x-axis, as shown in Figure 1.2. The direction of the reaction plane differs in each collision and to accurately measure observables with angular dependence this must be taken into account.

When a collision takes place, the participants, shown in Figure 1.1 as the coloured particles and in Figure 1.2 as the orange ellipse, will form a (QGP) due to the high energies present.

In this thesis we focus on a particular effect, the Chiral Magnetic Effect (CME), that is predicted to occur at collisions of such energies at which a QGP is formed. The CME is theorized to separate charged particles in the out-of-plane (y-axis in Figure 1.2) direction.

Two conditions required for the effect to occur are the presence of a strong magnetic field and a net amount of topological charge change [7].

Both of these conditions are met in the heavy-ion collisions studied at the ALICE detector. The spectators, shown as white particles in Figure 1.1, consist partly of positively charged protons, and because they travel at relativistic speeds these produce very strong currents near the point of collision, although only for a very brief amount of time. A very large magnetic field is caused by these currents according to

the Biot-Savart law. An estimation of the size of the magnetic field can be found in Ref. [4] and in Appendix A of Ref. [7]. In the latter thesis the presence of topological charge is explained in more detail as well.

At present, experimental research to detect the CME has been conducted at the STAR collaboration at the Brookhaven's Relativistic Heavy Ion Collider [2] and at the LHC [1]. No conclusive evidence for the occurence of the CME has been presented yet, as there are currently too many other background effects that could explain findings at both of these detectors. An in-depth look at the results from the STAR and ALICE Collaborations and some of the possible background sources can be found in Ref. [3].

Our goal is to measure to what extent certain charge-dependent correlations associated with the CME are affected by background-sources, specifically Local Charge Conservation (LCC). We achieve this by simulating collisions as they occur at the ALICE detector, using a so called Blast-Wave Model (BWM) incorporating LCC, making sure charge-independent observables match LHC values and then comparing charge-dependent correlations of the BWM with real data.

In the following, we first we provide a better explanation of the CME, describe effects to take into account when attempting to detect the CME, explain to which extend we manage this and which exact correlations we measure in Section 2. Then, in Section 3, we describe how our BWM works in great detail, how we measure previously mentioned observables as well as how the model varies with respect to the parameters used. Next we present our findings, together with observables calculated from real data in Section 4. Finally we will discuss our findings and possible improvements to the model and future research in Section 5.

2 Theory

In this chapter we will first cover the Chiral Magnetic Effect (CME), its properties and why it is expected to occur in heavy-ion collisions. Next we will see which possible background effects exist that might complicate the observation of the CME. Finally observables that are predicted to have discriminating power with respect to these background effects will be discussed.

2.1 Chiral Magnetic Effect

The CME is defined as the property of gauge field configurations with nonzero winding number Q_w to be able to separate charged particles when sufficiently strong magnetic fields are present, such as those expected to occur at the LHC [4, 7].

The large magnetic field will cause the particles to be near the lower Landau levels, causing their spin to tend to align with the magnetic field. The particles are restricted to move in the direction of their spin, or opposite of it, depending on their helicity. Thus, we expect the magnetic field to cause particles to move predominantly along the field direction.

To illustrate the CME, we will assume the magnetic field to be very strong and homogeneous. This causes positively charged right-handed fermions and negatively charged left handed fermions to align their momentum with the field direction and negatively charged right-handed fermions and positively charged left handed fermions to do the opposite, as shown in part one of Figure 2.1. In this figure the blue arrows represent the spin and the red arrows the momentum of the particles.

When the fermions then interact with a gauge field configuration with non-zero Q_w, this results in a change in chirality of these fermions, and thus in a non-zero net chirality in the system. In the chiral limit, which assumes quark masses to be zero, a change in chirality corresponds to a change in helicity [7], meaning the system now has a non-zero net helicity. Since the spins of the fermions must remain aligned with the field direction, this results in a momentum flip proportional to Q_w, as illustrated in part 3 of Figure 2.1.

When at first there are as many positively charged right-handed fermions as negatively charged left-handed fermions, which are travelling in the magnetic field direction, the total net charge in this direction in momentum space is zero. The same goes for the fermions with opposite helicity travelling opposite to the field direction. But when the helicity of a fermion changes, e.g. a positively charged right-handed particle moving upwards becomes a positively charged left-handed particle moving downwards, a net charge difference arises, resulting in an upward current and a charged dipole moment in the momentum distribution.

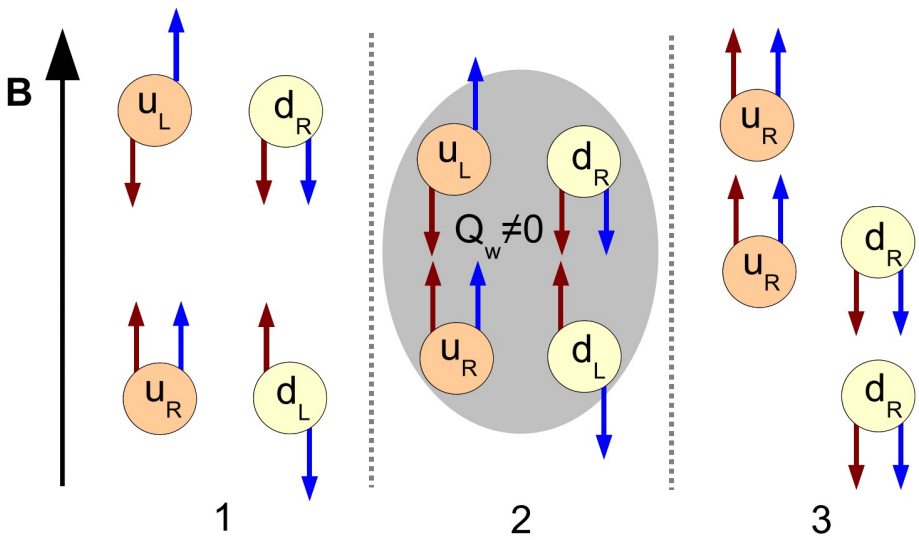

Figure 2.1: The Chiral Magnetic effect, illustrated assuming the magnetic effect to be very large and homogeneous. The blue arrows represent spin, the red arrows the momentum of the particles. Figure from Ref. [7]

2.2 Effects to take into account for possible observables

The non-zero Q_w can be both positive or negative in a collision, with equal probability, depending on the initial state of the system [7]. This in turn causes the net helicity of the system to be either along or opposite to the magnetic field direction. As this varies per event, charge separation, and the resulting electric dipole vary randomly in sign, meaning the direction will be along the magnetic field or opposite to it, with equal probability. If we then take sufficiently many events, the average vanishes.

To combat this we need to use observables which are sensitive to the variance of the dipole, and thus do not vanish. The downside is that effects other than the CME will have an effect on these observables.

Another characteristic of heavy-ion collisions we have to adjust our observables for, is that the reaction plane in each event has a random orientation Ψ_{RP}. This causes any azimuthal angle ϕ we measure in the lab frame to be randomly oriented and thus any observable depending on ϕ as measured in the lab frame to be uniformly distributed when averaged over many events.

This can be dealt with by determining the orientation of the reaction plane for every event and having every observable be dependent on the angle with respect to the reaction plane, so on $(\phi - \Psi_{RP})$. Determining the reaction plane exactly would require measurement of an infinite number of reaction products, and as there is only

a finite number in each event, it is experimentally impossible and there will always be a limited resolution. A way around this is to look at correlations depending on the difference between two or more azimuthal angles ($\phi_1 - \phi_2$).

As mentioned earlier, there are additional, two- and multi-particle effects to take into account because we are looking at correlators depending on the variance. Some potential background sources are clusters, momentum conservation, local charge conservation and the chiral magnetic spiral effect [8].

2.3 Gamma and Delta

Two correlators that incorporate the previously mentioned solutions for dealing with the random orientation of the event plane and the need for dependence on the variance, are the Delta and Gamma functions, originally proposed by Voloshin [10]. These are defined as

$$\gamma_{\alpha,\beta} = \langle \cos(\phi_\alpha + \phi_\beta - 2\Psi_{RP}) \rangle \tag{2.3.1}$$

and

$$\delta_{\alpha,\beta} = \langle \cos(\phi_\alpha - \phi_\beta) \rangle \,, \tag{2.3.2}$$

where α and β are used to denote the charge of the particles measured. The brackets $\langle ... \rangle$ denote a double average, both over all particles and all events.

These formulas can be rewritten in terms of Q-vectors, where we define the Q_n-vector of a subset of particles ($i = 1, ...M$) as

$$Q_n = \sum_{j=1}^{M} e^{in\phi_j} \tag{2.3.3}$$

with the real and imaginary parts

$$\mathbb{R}Q_n = \sum_{j=1}^{M} \cos(n\phi_j) \tag{2.3.4}$$

and

$$\mathbb{I}Q_n = \sum_{j=1}^{M} \sin(n\phi_j). \tag{2.3.5}$$

Rewriting γ and δ in terms of the Q_n-vectors results in:

$$\gamma_{\alpha,\beta} = \frac{\mathbb{R}(Q_{1\alpha} \cdot Q_{1\beta} \cdot Q_2^*)}{M_\alpha M_\beta - \delta_{\alpha,\beta} M_\alpha} = \tag{2.3.6}$$

$$\frac{(\mathbb{R}Q_{1\alpha}\mathbb{R}Q_{1\beta} - \mathbb{I}Q_{1\alpha}\mathbb{I}Q_{1\beta} - \delta_{\alpha,\beta}\mathbb{R}Q_{2\alpha})\mathbb{R}Q_2 + (\mathbb{R}Q_{1\alpha}\mathbb{I}Q_{1\beta} - \mathbb{I}Q_{1\alpha}\mathbb{R}Q_{1\beta} - \delta_{\alpha,\beta}\mathbb{I}Q_{2\alpha})\mathbb{I}Q_2}{M_\alpha M_\beta - \delta_{\alpha,\beta} M_\alpha}$$

and

$$\delta_{\alpha,\beta} = \frac{\mathbb{R}(Q_{1\alpha} \cdot Q_{1\beta}^*)}{M_\alpha M_\beta - \delta_{\alpha,\beta} M_\alpha} = \frac{\mathbb{R}Q_{1\alpha}\mathbb{R}Q_{1\beta} + \mathbb{I}Q_{1\alpha}\mathbb{I}Q_{1\beta}}{M_\alpha M_\beta - \delta_{\alpha,\beta} M_\alpha}. \tag{2.3.7}$$

Here the $\delta_{\alpha,\beta}$ in the denominator is the Kronecker delta and not our correlator.

For our measurements we can use an alternate formula to calculate $\gamma_{\alpha,\beta}$, because our simulation operates in the intrinsic reference frame of the system. This means Ψ_{RP} is automatically aligned with the x-axis in every event, thus we have $\Psi_{RP} = 0$. We can then calculate $\gamma_{\alpha,\beta}$ with:

$$\gamma_{\alpha,\beta} = \langle \cos(\phi_\alpha + \phi_\beta) \rangle = \frac{\mathbb{R}Q_{1\alpha}\mathbb{R}Q_{1\beta} - \mathbb{I}Q_{1\alpha}\mathbb{I}Q_{1\beta} - \delta_{\alpha,\beta}\mathbb{R}Q_{2\alpha}}{M_\alpha M_\beta - \delta_{\alpha,\beta} M_\alpha}. \tag{2.3.8}$$

We then combine our results, using

$$\delta C = \frac{1}{2}[C_{+,-} + C_{-,+} - C_{+,+} - C_{-,-}] \tag{2.3.9}$$

with for C either the results from equation 2.3.7 or 2.3.8. We use this specific correlator because it is sensitive only to charge-dependent correlations. This means that effects from charge-independent correlations should not any have effect on the results. The CME, being charge-dependent, should then contribute to δC, while for example transverse momentum conservation should not.

The value returned by Equation 2.3.9 is either our Gamma or Delta correlator. These we compare to the values obtained from ALICE data [1]. Should we get results for these correlators that match ALICE observations, without any charge-dependent effects except for LCC incorporated into our model, this would make a strong case against the presence of the CME in LHC experiments.

3 Methods

To model the LHC and more specifically the Pb-Pb collisions with centre-of-mass energy $\sqrt{2.76}\ TeV$, we implement a Blast-Wave Model (BWM) to simulate particle production from the QGP. In our model we are limited to the simulation of one type of particle, so we use pions as they are the most abundant reaction product of relativistic heavy-ion collisions. In the following we will go into the specifics of our model and how we use it to calculate values for the observables of our interest. How the parameters we use influence results will be discussed as well. The full ROOT code can be found in Appendix F.

3.1 Blast-Wave Model

Due to the centrality dependence of the observables we are interested in, the parameters we used in our model vary with centrality class as well. We used the Temperature (T), the dependence between the radius and the boost ($PradB$) the flow parameters (ρ_0, ρ_2, ρ_4) and the Eccentricity (ϵ) as our parameters. The initial values were taken from Ref. [5], we tuned them to make our model fit the LHC charged particle spectra for V_2 and dN/dP_t better and the final values can be found in Table 3.1.

Centrality (%)	$T(K)$	$PradB$	ρ_0	ρ_2	ρ_4	ϵ
$5-10\%$	0.112	0.440	0.930	0.00831	0.0015	0.0961
$10-20\%$	0.100	0.363	0.935	0.01290	0.0020	0.1300
$20-30\%$	0.099	0.384	0.938	0.01540	0.0025	0.1720
$30-40\%$	0.098	0.419	0.940	0.01600	0.0028	0.2040
$40-50\%$	0.095	0.450	0.943	0.01510	0.0033	0.2240

Table 3.1: The parameters as used in our BWM

In essence the model generates the desired data by iterating over the number of events (referred to as the outer loop), given as a parameter ($n = 60.000$ was used for this thesis) and within each such event iterating over the number of particles generated per event (referred to as the inner loop).

The model begins with randomly selecting a centrality bin in the outer loop. The $5-10\%$ bin has half the probability of being selected compared to the other bins, since its binwidth is also half that of the other centrality classes. When this is done, we have the information needed to enter the loop over the number of particles.

This number was set per centrality range so as to match the ALICE findings for the dN/dP_t spectra. Specifically, this was done by integrating these spectra to get the total number of particles within the P_t range $[0.15, 2]\ GeV/c$. We then matched

this number by performing the same integration over the results from our simulation and tuning the multiplicity of the model until a satisfying agreement was reached. Since we create particles in pairs, the real number of particles created in each event is double the value used in our model.

The reason for the pairwise creation of pions is that LCC is easily incorporated into the model. In each iteration of the inner loop, we take the first pion as positively charged and the second as negatively charged, so the total charge of these particles is zero. When they are created at the same position in space, charge is conserved locally.

Next, both of the pions are given an energy randomly selected according to a Boltzmann Distribution:

$$f_{\text{Boltzmann}}(x) = \sqrt{x^2 - m^2} \cdot e^{\frac{-x}{T}}. \tag{3.1.1}$$

The distribution depends on the mass of a pion and of course the temperature. This means the energy is centrality dependent, given that the temperature varies with centrality as in Table 3.1.

The shape of the colliding particle distribution is elliptical at the moment of the collision, which can be seen in Figure 1.2. We therefore randomly select a point on the ellipse with eccentricity that matches that of the current centrality range, as the source of the pions.

The position of the two pions on the ellipse in the $x, y-$plane corresponds to the angle ϕ as seen from the centre of the collision. From ϕ we calculate the angle ϕ_b normal to the surface of the ellipse at this point.

This angle is used to calculate a momentum boost vector, which is the same for both pions as they are created at the same location to implement LCC. To obtain this vector, first the transverse collective flow β is calculated

$$\beta = r^{PradB} \cdot (\rho_0 + \rho_2 \cos(2\phi_b) + \rho_4 \cos(4\phi_b)), \tag{3.1.2}$$

as is done in Ref. [5], with r the radius of ellipse at ϕ.

Next, we calculate the arctangent of β, and call it ρ_β.

From ρ_β and ϕ_b we now construct the boostvector, as

$$\begin{pmatrix} \sinh(\rho_\beta) \cdot \cos(\phi_b) \\ \sinh(\rho_\beta) \cdot \sin(\phi_b) \\ \sinh(\eta_s) \cdot \cosh(\rho_\beta) \\ \cosh(\eta_s) \cdot \cosh(\rho_\beta) \end{pmatrix}, \tag{3.1.3}$$

with η_s a randomly generated pseudorapidity.

The pion momenta are then calculated with the formula

$$P_i = \sqrt{E_i^2 - m_i^2}, \tag{3.1.4}$$

using the energy they independently obtained earlier from the Boltzmann Distribution. Then the direction of the momentum is independently determined for the particles, in a random manner, to simulate particle creation from the QGP.

The boostvector is then applied to both of the particle momenta and from the resulting Lorentz-vectors the final pion momenta can be obtained .

3.2 Measuring Observables

To construct the charged particle yield (dN/dP_t) spectra, we simply retrieve the final values for P_x and P_y from the Lorentz-vector and calculate

$$P_t = \sqrt{P_x^2 + P_y^2}. \tag{3.2.1}$$

The V_2 spectra are obtained by calculating the P_t values as in Equation 3.2.1 to determine which bin we need to fill, and we get our value for V_2 from the definition of the formula for the general flow coefficients V_n [9]:

$$V_n = \langle \cos(n(\phi - \Psi_{RP})) \rangle \,. \tag{3.2.2}$$

In our case $n = 2$ and $\Psi_{RP} = 0$, because we our simulation operates from the intrinsic frame.

To obtain Gamma and Delta, for each event we calculate Equations 2.3.4 and 2.3.5, for $n = \{1, 2\}$ and for both the positively and negatively charged particles separately.

These values are subsequently inserted into Equations 2.3.7 and 2.3.8 for the four possible combinations of positively and negatively charged particles, and inserted into Equation 2.3.9 to obtain Gamma and Delta.

All of the results we generated with our model are compared with values measured at the LHC, both unscaled and scaled to make a good comparison possible.

3.3 Parameter Sensitivity

To evaluate the sensitivity of the predictions to the details of our model, in this section we check how results from the model transform when we vary our physical parameters (meaning not PradB). This will also increase our understanding of the physical meaning of our parameters. To do so we increased the values we normally use, as seen in Table 3.1, by 50%, except for ρ_0, which we decreased by a factor of 0.67 instead, because otherwise some values within our model exceed their maximum range. Then we compare the resulting figures to those obtained with the standard parameter values. For this qualitative comparison we used $n = 2500$ events. All of the relevant figures can be found in Appendix E.

We start with the temperature (T). Noticeable changes can be seen in the dN/dP_t and V_2 spectra. In the lower P_t range our model now produces lower values for dN/dP_t than before, while in the higher P_t range values are much higher. This can be understood from a physical perspective, as higher T means higher energy which generally corresponds to higher momenta. More specifically, higher T means the energy the particles randomly obtain from the Boltzmann Distribution is higher, since a greater value of T in the exponent corresponds to higher values of $f_{\text{Boltzmann}}$, Equation 3.1.1.

We also notice a substantial decrease in V_2 for all values of P_t. This comes as no surprise, as the Elliptical flow should shift to higher P_t values, as momentum increases with T. Normally V_2 peaks just past our range of $P_t = [0.15, 2]GeV/c$ and now the peak will be at a somewhat higher value.

Next variation in ρ_0 is investigated. As mentioned earlier, in contrast to the other parameters we tested a decrease in ρ_0. We observe an increase in dN/dP_t in the lower P_t range and the opposite in the higher P_t range. This is as expected, for ρ_0 parametrizes the strength of our directed flow, meaning it influences how much particles are boosted isotropically. A decrease in ρ_0 has to correspond to a decrease in the total momentum boost particles receive, meaning average momenta are lower and the observed shift in the dN/dP_t spectra is to be expected.

A decrease in V_2 is also expected, as a decrease in total momenta will also result in a decrease in anisotropic flow. This is exactly what we observe from our model.

The Delta we find is also significantly lower than before. We did expect this to happen, as larger ρ_0 means particles are more correlated due to the shared boost they receive. A decrease in the total momentum boost then means particles are less correlated.

We then focus on ρ_2. This is the parameter we use to simulate elliptic flow. We therefore expect to see a change in V_2. This proves to be correct, only in V_2 a change is visible, and as we would expect it is an increase.

Now we turn our attention to ρ_4, which parametrizes Fourier coefficients of the flow of higher order than the directed and elliptic flow. variation of this parameter results in no noticeable effect on our model, which is as it should be, considering the observables we measure depend mostly on the directed and elliptic flow (V_2).

Finally we inspect the eccentricity (ϵ). An increase in eccentricity means the shape of the participants in the collision becomes more elliptic, so less circular. We see this in the elliptic flow V_2, which is increased as we should expect, since a more elliptical distribution of particles at the beginning of our model causes the flow to be more anisotropic, thus resulting in higher elliptical flow.

Gamma seems to be sensitive to an increase of the eccentricity as well, having higher values than before. The reason for this is not entirely clear and could be an interesting topic for further study.

We see that our model is sensitive to changes in our parameters in a way we would expect. All of the significant changes can be explained physically, save the eccentricity dependence of Gamma, which we currently cannot explain and might be worth conducting additional research on.

4 Results

First, to enable us to discuss the accuracy of our simulation, we compare the V_2 and dN/dP_t spectra obtained from our BWM to their counterparts as measured at the LHC. Then we present the Gamma and Delta functions, our correlations of interest. In this section any comparison will be to real measurements from the LHC.

4.1 Simulation Accuracy

To verify that our model accurately describes collisions as they occur at the ALICE detector, we have simulated V_2 and dN/dP_t for $n = 60.000$ events, with parameter values as shown in Table 3.1. Resulting values for the $30 - 40\%$ centrality class are shown in Figure 4.1 and 4.2 in blue. In black real data, measured at the LHC [1] is displayed. The relative difference between BWM and LHC values can be found below the spectra.

From Figure 4.1 we see that within the P_t range of our interest, the BWM observations fluctuate between plus and minus 8% of LHC results. In the medium P_t range they are below experimental results. At higher ranges the opposite is true. This results in our model having a slightly smaller curvature.

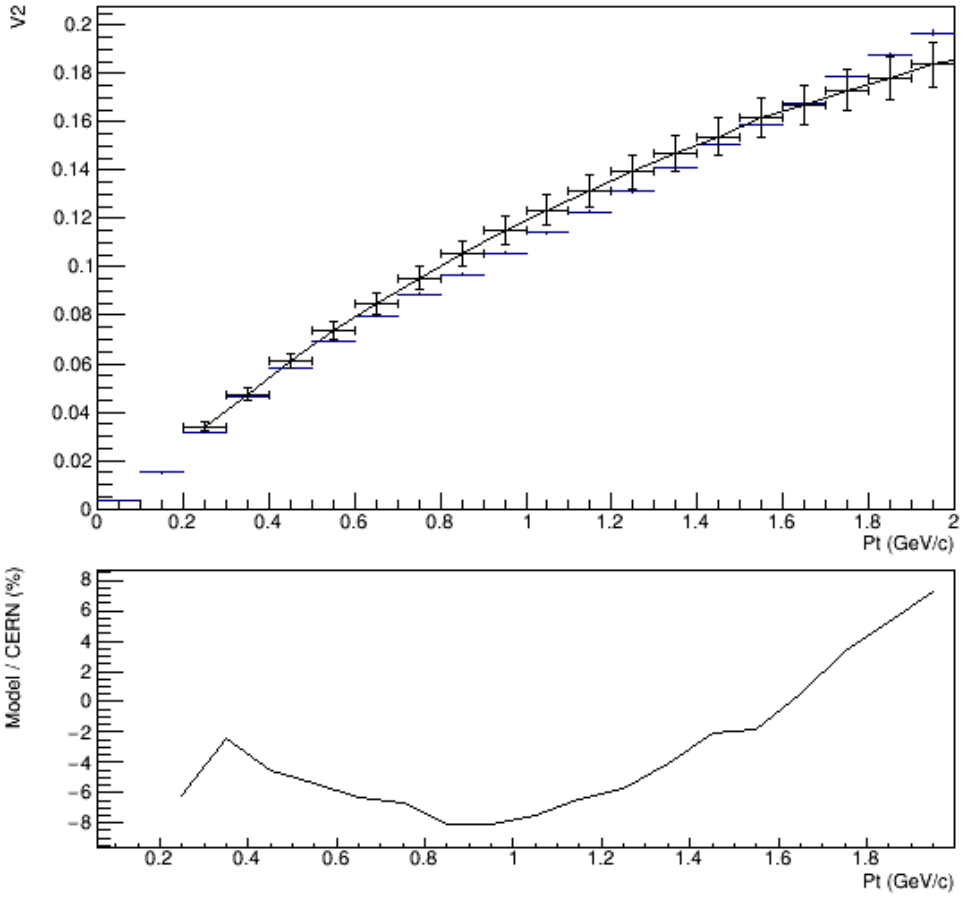

Figure 4.1: Elliptic flow (V_2) over transverse momentum (P_t) for centrality class $30 - 40\%$. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

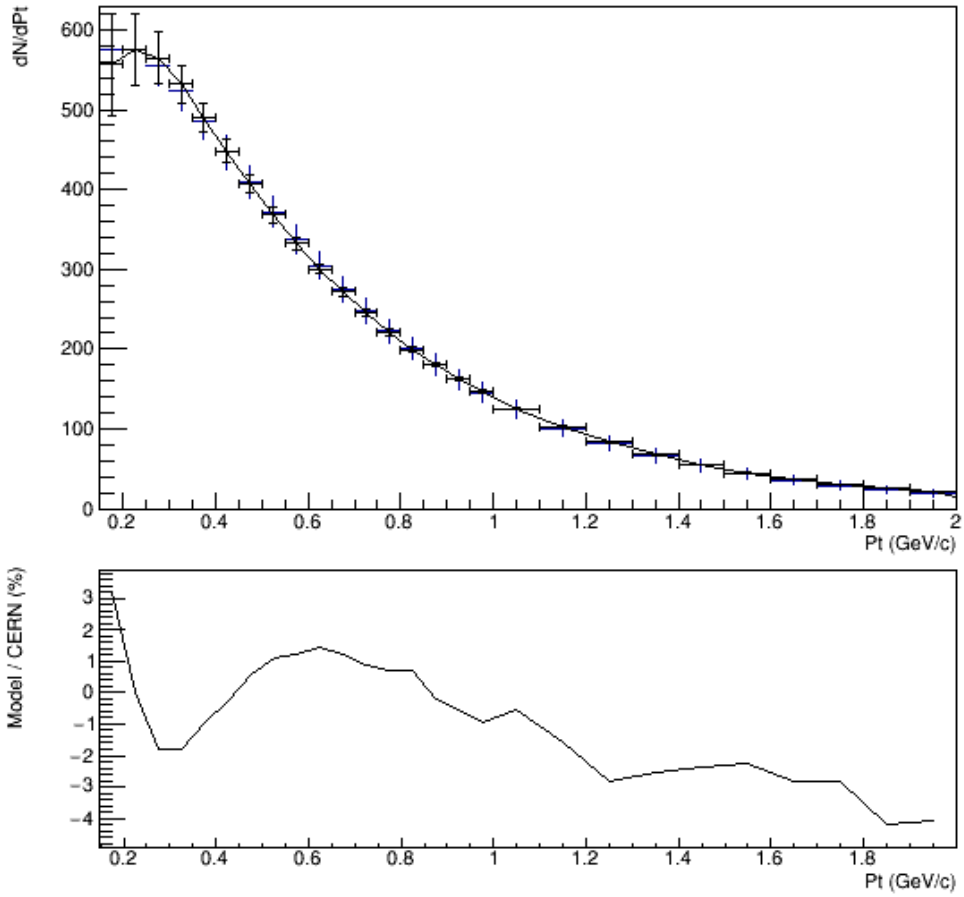

Figure 4.2: Charged particle yield (dN/dP_t) over the transverse momentum (P_t) for centrality class $30 - 40\%$. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

Looking at Figure 4.2, we see that the model presents us with values of dN/dP_t within a 4% difference. Every generated data point lies within one standard deviation from its measured counterpart.

Resulting figures for other centrality ranges for both the V_2 and dN/dP_t spectra with their relative differences can be found in Appendices A and B.

4.2 Gamma and Delta

The correlations of our interest, Gamma and Delta, calculated from our simulated data, are shown in blue in figure 4.3 and 4.4 respectively. Again we used $n = 60.000$ events and parameters as shown in Table 3.1. In black we show these correlations calculated using real data measured at the ALICE detector [1].

Looking at Figure 4.3 we see our BWM produces values for Gamma which are a little under 10 times smaller than calculations using experimental data do. To make a better comparison possible, in Appendix C we also show a graph where we scale Gamma by a factor of 0.1. An examination of the shapes of the two functions, reveals that the model has a slightly steeper curve.

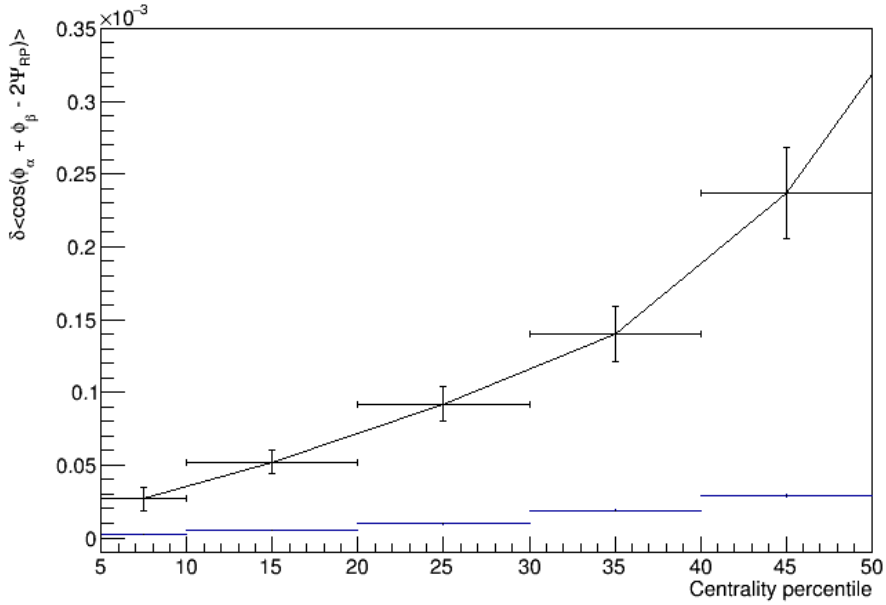

Figure 4.3: Gamma ,$\delta \langle \cos(\phi_\alpha + \phi_\beta - 2\Psi_{RP}) \rangle$, from Equations 2.3.8 and 2.3.9 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

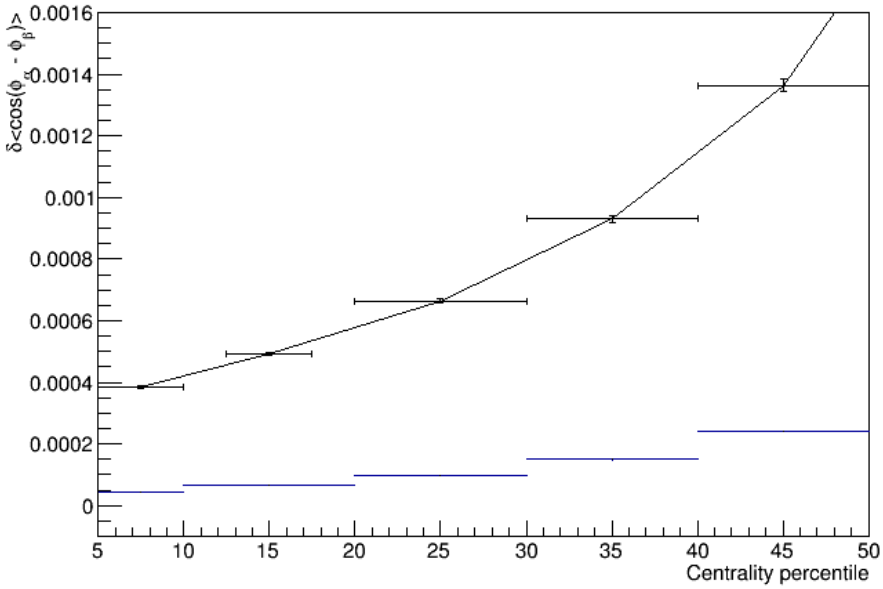

Figure 4.4: Delta ,$\delta \langle \cos(\phi_\alpha - \phi_\beta) \rangle$, from Equations 2.3.7 and 2.3.9 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

If we now look at the Delta correlator in Figure 4.4 we get similar results. Delta as calculated from BWM results is about 7 times smaller than when we use LHC data. In Appendix D we show a comparison with scaled ALICE results, using a factor of 0.15. When we analyse the shapes of the functions a slightly steeper incline for our Delta is apparent.

5 Discussion

First we will discuss the V_2 and dN/dP_t spectra to see how our model performs in simulating these characteristic observables. When taking into account all of the centrality classes, which can be found in Section 4 and the Appendices, fluctuations of about 10% from the LHC reference spectra can be observed. The highest positive difference is always seen at the higher P_t values and in the middle range our BWM generally underestimates the reference curve. This means our model lacks curvature, but as fluctuations are of the order of 10% this seems acceptable.

The dN/dP_t spectra match even better, with fluctuations for all centrality ranges staying predominantly within 5% of the LHC reference curves. The difference graphs reveal the model seems to slightly underestimate LHC data at high P_t, while overestimating as P_t decreases. At very low P_t however, a steep fall in relative difference can be observed, with at the very low ranges a steep incline. This is not very noteworthy though, since fluctuations are still small and error margins on LHC data are quite large in this region. This leads us to conclude the charged particle yield spectra generated by our model describe real data well.

As the model is deemed reasonably accurate we can now discuss Gamma and Delta with confidence. In Section 4.2 we have seen that our results underestimate LHC data by a factor of 10 and 7 for $\gamma_{\alpha,\beta}$ and $\delta_{\alpha,\beta}$ respectively, with fairly matching shapes.

To get more insight into Gamma and Delta we'll now compare the same and opposite sign values for $\gamma_{\alpha,\beta}$ and $\delta_{\alpha,\beta}$ reported by the ALICE collaboration with our simulated values. LHC results for $\gamma_{\alpha,\beta}$ can be found in a table in Figure 5.1 and for $\delta_{\alpha,\beta}$ in Figure 5.2. The data obtained from our model is also shown graphically in comparison to LHC data in Appendices C and D. Both unscaled and scaled graphs are presented there (with factors described in their caption).

First, looking at the figures for $\gamma_{+,-}$ and $\gamma_{-,+}$, we see immediately that the figures are identical. This has to be the case, as in the calculation of both correlators we average over the same particles, but with ϕ_α and ϕ_β switched when measuring $\gamma_{+,-}$ instead of $\gamma_{-,+}$. Since the cosine is a symmetric function, $\cos(\phi_1 - \phi_2)$ equals $\cos(\phi_2 - \phi_1)$, meaning results must be the same.

We see that LHC data is negative at first, staying at relatively low values and then changes sign and grows rapidly at higher centralities. Our model produces values that are positive for all centrality ranges. To match LHC data, a graph with a scaling factor of -0.75 was also made, showing similar slope. Because in the higher centrality ranges LHC data flips sign and gets to much more significant values, it would be useful to adjust the model to include these centrality ranges to allow for better comparison.

If we assume the CME is present, charge is separated along the magnetic field dir-

ection (out-of-plane), so $\phi = \pi/2$ or $\phi = 3\pi/2$. For opposite-charge pairs we would expect $\phi_\alpha = \pi/2$ while $\phi_\beta = 3\pi/2$ or the other way around, both resulting in $\cos(\phi_\alpha + \phi_\beta) = 1$, so positive. This means that LHC results should have an additional positive factor added to them due to the CME compared to the model results and this is exactly what we observe at higher centrality. That we do not observe this from data of the LHC at low centralities might be due to other background effects, which are of greater magnitude than the CME. Thus, at the low centrality ranges we cannot interpret the difference in results as a display of the CME, but as said before it would be interesting to compare at the higher ranges where the correlators take more significant values (see Figure 5.1).

RE SQRT(S)/NUCLEON	PB PB --> .GE.3CHARGED X 2760.0 GeV OPPOSITE SIGN	SAME SIGN
CENTRALITY IN PCT	MEAN(COS(PHI(A)+COS(PHI(B)-2*PHI(RP))	
0.0 – 5.0	0.000000 ± 0.000004	-0.000016 ± 0.000003
5.0 – 10.0	0.000004 ± 0.000005	-0.000023 ± 0.000003
10.0 – 20.0	-0.000003 ± 0.000005	-0.000055 ± 0.000003
20.0 – 30.0	-0.000014 ± 0.000007	-0.000106 ± 0.000005
30.0 – 40.0	-0.000026 ± 0.000011	-0.000166 ± 0.000008
40.0 – 50.0	-0.000039 ± 0.000018	-0.000276 ± 0.000013
50.0 – 60.0	0.000022 ± 0.000036	-0.000379 ± 0.000026
60.0 – 70.0	0.000249 ± 0.000087	-0.000425 ± 0.000063
70.0 – 80.0	0.001059 ± 0.000279	-0.000293 ± 0.000196
	Plot SelectPlot	Plot SelectPlot

Figure 5.1: $\gamma_{\alpha,\beta}$ for same and oppositely charged particles, as measured by ALICE [1].

Now we inspect $\gamma_{+,+}$ and $\gamma_{-,-}$. From the scaled plots $\gamma_{+,+}$ seems to possibly increase slightly, but error margins are relatively big and we cannot be sure. $\gamma_{-,-}$ seems to just fluctuate around the origin. LHC data however shows a clear downward sloping pattern with significantly larger values.

We look at what we would expect to see in case of the CME as before. The charge separation should give us $\phi_\alpha = \pi/2$ while $\phi_\beta = \pi/2$ or $\phi_\alpha = 3\pi/2$ while $\phi_\beta = 3\pi/2$, both resulting in $\cos(\phi_\alpha + \phi_\beta) = -1$. LHC data is indeed more negative than our predictions from the model, which is an indication of the possible presence of the CME.

Now we'll focus our attention on $\delta_{\alpha,\beta}$. Figure 5.2 shows data points from ALICE. We see that for both opposite and same sign measurements, $\delta_{\alpha,\beta}$ is positive and strictly increasing.

When comparing the model with LHC data, we see that for $\delta_{+,-}$ and $\delta_{-,+}$ the results

RE	PB PB --> .GE.2CHARGED X	
SQRT(S)/NUCLEON	2760.0 GeV	
	OPPOSITE SIGN	SAME SIGN
CENTRALITY IN PCT	MEAN(COS(PHI(A)-COS(PHI(B))	
0.0 – 5.0	0.000457 ± 0.000002 (stat) ± 0.000000 (sys)	0.000139 ± 0.000002 (stat) ± 0.000000 (sys)
5.0 – 10.0	0.000561 ± 0.000003 (stat) ± 0.000000 (sys)	0.000178 ± 0.000002 (stat) ± 0.000000 (sys)
10.0 – 20.0	0.000732 ± 0.000003 (stat) ± 0.000000 (sys)	0.000241 ± 0.000002 (stat) ± 0.000001 (sys)
20.0 – 30.0	0.001021 ± 0.000004 (stat) ± 0.000000 (sys)	0.000357 ± 0.000003 (stat) ± 0.000001 (sys)
30.0 – 40.0	0.001432 ± 0.000006 (stat) ± 0.000002 (sys)	0.000502 ± 0.000004 (stat) ± 0.000003 (sys)
40.0 – 50.0	0.002121 ± 0.000010 (stat) ± 0.000001 (sys)	0.000757 ± 0.000007 (stat) ± 0.000007 (sys)
50.0 – 60.0	0.003306 ± 0.000018 (stat) ± 0.000003 (sys)	0.001168 ± 0.000012 (stat) ± 0.000008 (sys)
60.0 – 70.0	0.005447 ± 0.000034 (stat) ± 0.000003 (sys)	0.002095 ± 0.000024 (stat) ± 0.000006 (sys)
	Plot SelectPlot	Plot SelectPlot

Figure 5.2: fig:$\delta_{\alpha,\beta}$ for same and oppositely charged particles, as measured by ALICE [1].

match quite well, taking into account a scaling factor of 0.1, with the slope being a bit steeper. As with $\gamma_{+,-}$ and $\gamma_{-,+}$ the figures are identical as they should be.

Charge separation due to the CME should be reflected in LHC results being lower than those from our model, as for opposite-charge pairs we would expect $\phi_\alpha = \pi/2$ while $\phi_\beta = 3\pi/2$ or the other way around, both resulting in $\cos(\phi_\alpha - \phi_\beta) = -1$. The deviation we see here is thus the opposite of what we would expect in case of the CME being present and indicates other background effects may be present.

Same sign $\delta_{\alpha,\beta}$ seems to be positive and slightly increasing, but even when scaled by a factor of 0.05, if due to the size of the error bars we are not able to say much as to how well our model matches LHC data shape wise.

That LHC data is much larger than our BWM data is as would be expected from the CME, as charge separation would give us $\phi_\alpha = \pi/2$ while $\phi_\beta = \pi/2$ or $\phi_\alpha = 3\pi/2$ while $\phi_\beta = 3\pi/2$, both resulting in $\cos(\phi_\alpha + \phi_\beta) = 1$.

Our analysis from looking at Equation 2.3.8 and Equation 2.3.7 and comparing what expected behaviour is from looking at out-of-plane angles cannot be conclusive. When the CME is strong this method will be a good estimator, but when the effect is small, as is expected, we could have a lot of particles with momenta in the in-plane direction as well, which would for example give $\phi_\alpha = 0$ while $\phi_\beta = \pi$, resulting in $\cos(\phi_\alpha + \phi_\beta) = -1$. When two particles are moving in the out-of-plane direction, so $\phi = \pi/2$, we get the exact same result. A clear distinction is thus not possible.

The differences between our model and the LHC data give an indication of charge dependent background effects, such as the CME, being present in the heavy-ion collisions at the ALICE detector. Reasoning behind this is that if our model, in which

no effect depending on charge except for LCC is implemented, accurately gives values for charge-independent correlations like V_2 and dN/dP_t, but gives different results when it comes to charge-dependent correlators, there are likely one or more charge-sensitive effects present.

However, there are many possible explanations for our model not matching ALICE data, a few of which we will discuss now.

One such explanation is that we ignore the pseudorapidity η_s, which is defined as the angle relative to the beam axis, completely in our model. However, this is likely not of major importance. As explained in Section 1, the expected charge separation due to the CME is in the out-of-plane direction, and should thus be visible in the plane spanned by the impact parameter and the azimuthal axis ϕ.

Another important realisation is that in this model we incorporated LCC and ultimately what we want is to detect the CME, but many other physical phenomena could be at play which we have not taken into account in our simulation. Some of them we have mentioned in Section 2.2. These could very well be showing in our observables and have to be disentangled from the signal of the CME in order to detect the possible presence of the CME conclusively.

Some other, probably less significant factors could also contribute to a better model and thus more accurate detection of charge-dependent effects.

Currently our model is tuned for single pion energies of up to 2 GeV/c, as energies of single pions are predominantly lower than this. We also limit the range in which we look at our observables to this energy, because our observables are mainly interesting within this range. We could however expand both of these values and while changes in the lower P_t region will likely not be significant, we would then be able to study the higher ranges properly as well.

Another limitation is the centrality range. Our maximum lies at collisions of 50% centrality. We took this as an acceptable compromise as including more centrality bins drastically increased runtime of the model in the early stages of development, however it should be possible to include these with some additional work. Especially when studying Gamma this might prove beneficial, as for larger centralities its components $\gamma_{+,-}$ exhibit interesting behaviour (a sign switch) and we cannot compare all that well to our model in its current state.

Running more iterations would also be helpful. Although for Delta, V_2 and dN/dP_t the error margin is quite small already, Gamma would benefit from some improvement in this respect. Currently the model is not able to generate proper dN/dP_t spectra for $n > 65k$ iterations, due to limitations of the mechanism we use for data storage. With some additional work, this could be improved upon.

For this model we made sure both V_2 and dN/dP_t matched experimental data to

a good extend, to establish the model described collisions at least reasonably well. Even better would be to take more charge-independent observables, for example some higher flow coefficients (V_4, V_6, etc.), and make sure the model describes them all as accurately as possible.

The parameters we used in the model were obtained from Hori's thesis [5], as mentioned in Section 3, and then varied manually to see how each parameter affected the model. Subsequently they were adjusted so V_2 and dN/dP_t matched well. If this process could be optimized by use of automatic fitting software, better fits could be obtained and thus better predictions made.

Finally, another important improvement would be to measure more charge-dependent correlators, especially if they exclude the impact of some of the aforementioned background effects on the resulting values. While two observables can be enough to give an indication, far more insight into the CME can be gathered by measuring more correlators.

6 Conclusion

From comparison with the V_2 and dN/dP_t spectra as shown in Section 4.1 we found our model to be quite accurate in simulating experimental results when it comes to charge-independent observables. However, two observables is a limited test and to be able to confidently say our model successfully describes the collisions taking place at the ALICE detector, we would need to test more charge-independent observables.

Our simulation provided us with values of our correlators of interest, Gamma and Delta, that, compared to results obtained at the LHC, were a factor of about 10 and 7 times smaller respectively. We were hoping to observe a difference in observed values, since our model did not include charge-dependent effects, save LCC, and if the CME is present in heavy-ion collisions at the LHC, charge-dependent correlations should give us deviating values.

This does not however automatically indicate the presence of the CME, as there are many other possible explanations for the observed difference. From a critical look at the different components $\gamma_{\alpha,\beta}$ and $\delta_{\alpha,\beta}$ of the final Gamma and Delta functions we saw there are some anomalies with ALICE data, which need further examining. These could very well be caused by multi-particle effects, the Chiral Magnetic Spiral effect or possibly other effects.

In conclusion, results from our model indicate the presence of physical phenomena other than LCC in heavy-ion collisions. The differences in results are at least partly charge-dependent and could be due to the CME, thus further research into the CME and how to distinguish it from background effects is definitely worthwhile. A number of suggestions for how the model could be adapted to this end for and future research into the CME have been mentioned in Section 5.

7 Acknowledgements

The Bachelors research project has been an enjoyable and informative experience. After the first years of theoretical study, getting to apply some of the knowledge I acquired to a large project like this has been a welcome change.

Without two people in particular my reasearch would not have been the same. I would like to express my gratitude to Raimond Snellings for guiding me in finding a topic for my thesis as well as supervising my work.

Aditionally, my thanks go out to Jacopo Margutti for the day-to-day supervision of my research. Not only during office hours, but in the holiday season as well you were prepared to answer questions and provide suggestions when I needed them. Your help with the model, providing me with a basis to expand upon and assisting me when I was stuck, has been invaluable.

References

[1] B. Abelev et all. Charge separation relative to the reaction plane in pb-pb collisions at snn=2.76tev. *Phys. Rev. Lett.*, 110, 2013.

[2] B.l Abelev et all. Azimuthal charged-particle correlations and possible local strong parity violation. *Phys. Rev. Lett.*, 103, 2009.

[3] Adam Bzdak, Volker Koch, and Jinfeng Liao. Charge-dependent correlations in relativistic heavy ion collisions and the chiral magnetic effect. *Strongly Interacting Matter in Magnetic Fields, Lecture Notes in Physics*, 871:503, 2013.

[4] Umut Grsoya, Dmitri E. Kharzeev, and Krishna Rajagopald. Magnetohydrodynamics and charged currents in heavy ion collisions. *Nuclear Physics A*, 931:986–991, 2014.

[5] Y. Hori, T. Gunji, H. Hamagaki, and S. Schlichting. Collective flow effects on charge balance correlations and local parity-violation observables in $\sqrt{s_{NN}} =$ 2.76 tev pb+pb collisions at the lhc. *eprint arXiv:1208.0603*, 2012.

[6] W. A. Horowitz. Probing the frontiers of qcd. *eprint arXiv:1011.4316*, 2010.

[7] Dmitri E. Kharzeev, Larry D. McLerran, and Harmen J. Warringa. The effects of topological charge change in heavy ion collisions: event by event p and cp violation. *Nuclear Physics*, 803:227–253, 2008.

[8] Jinfeng Liao, Volker Koch, and Adam Bzdak. Charge separation effect in relativistic heavy ion collisions. *Physical Review C*, 82, 2010.

[9] Raimond Snellings. Elliptic flow: a brief review. *New Journal of Physics*, 13, 2011.

[10] Sergei A. Voloshin. Parity violation in hot qcd: How to detect it. *Physical Review C*, 70, 2004.

Appendix A V_2

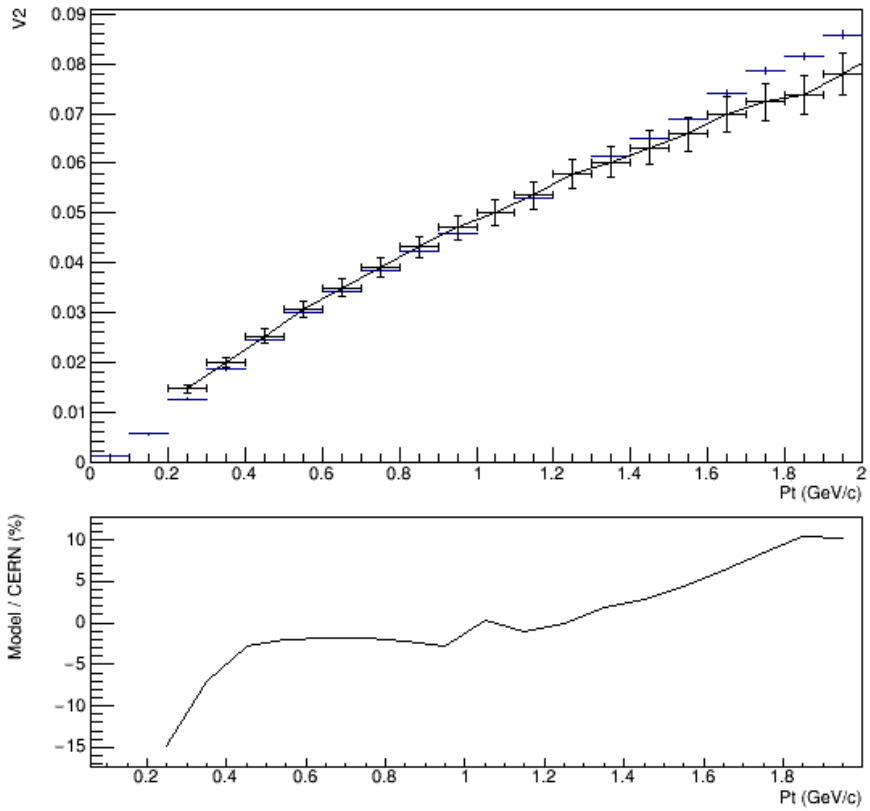

Figure A.1: Elliptic flow (V_2) over transverse momentum (P_t) for centrality class $5 - 10\%$. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

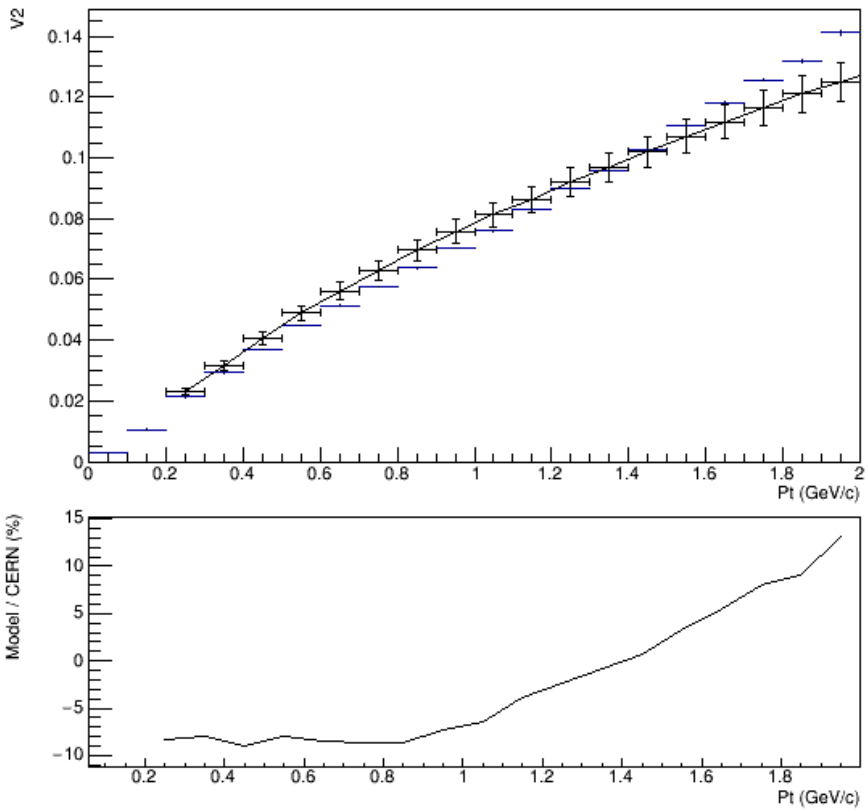

Figure A.2: Elliptic flow (V_2) over transverse momentum (P_t) for centrality class $10 - 20\%$. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

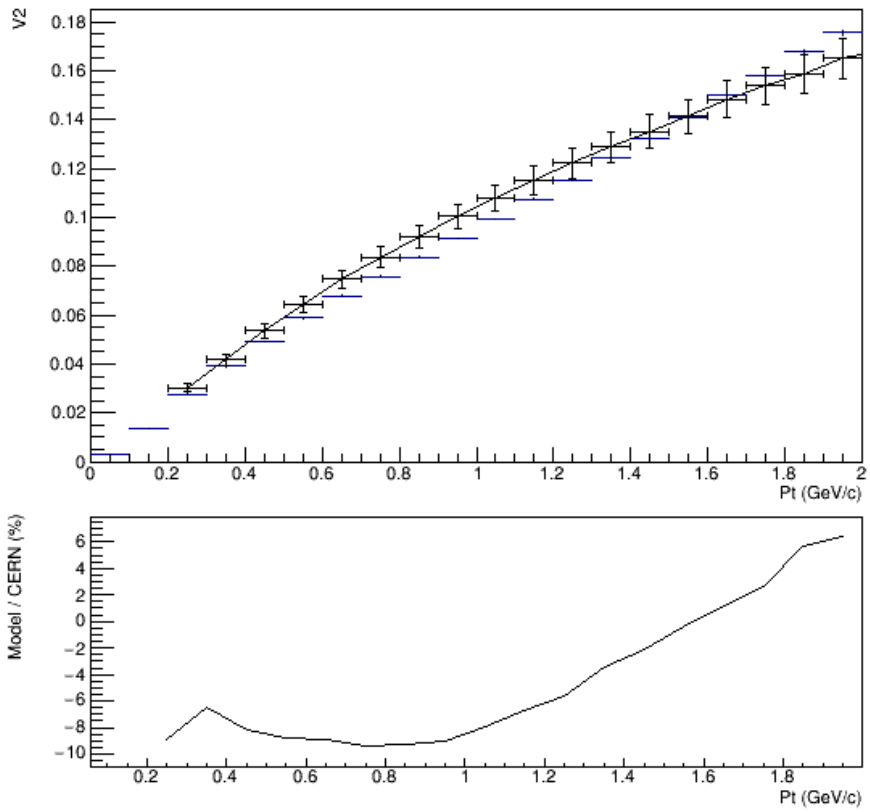

Figure A.3: Elliptic flow (V_2) over transverse momentum (P_t) for centrality class $20 - 30\%$. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

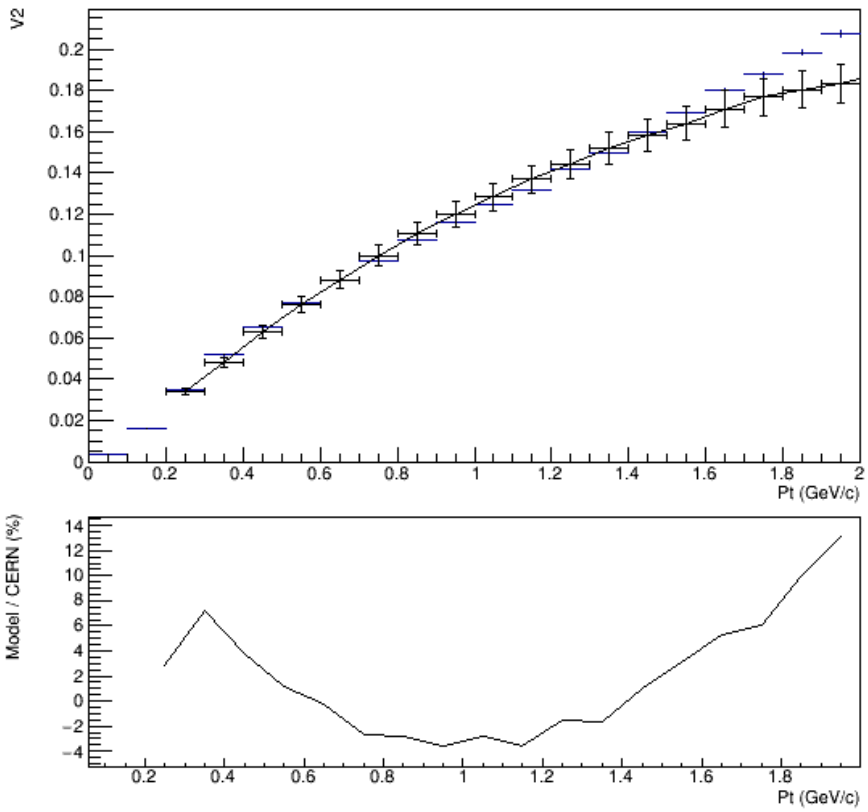

Figure A.4: Elliptic flow (V_2) over transverse momentum (P_t) for centrality class $40 - 50\%$. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

Appendix B dN/dP_t

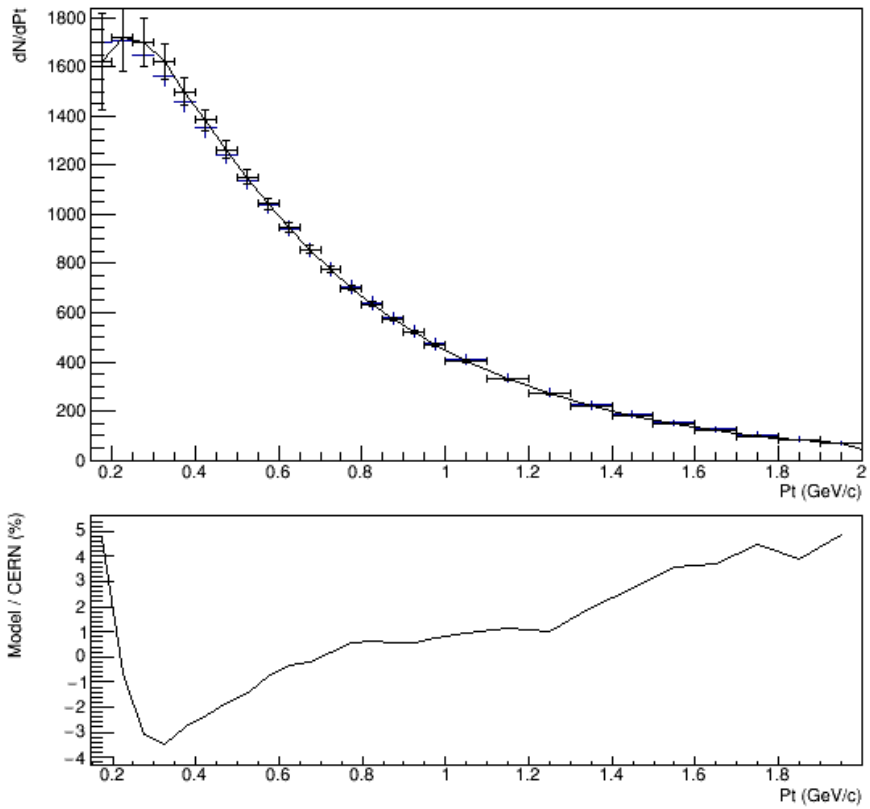

Figure B.1: Charged particle yield (dN/dP_t) over the transverse momentum (P_t) for centrality class $5-10\%$. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

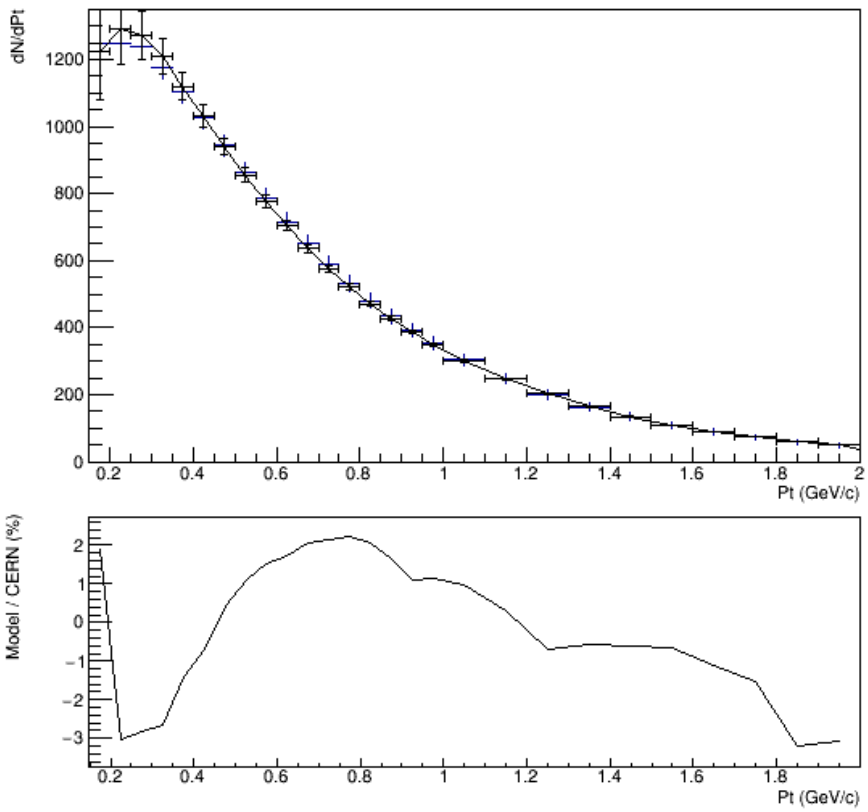

Figure B.2: Charged particle yield (dN/dP_t) over the transverse momentum (P_t) for centrality class $10 - 20\%$. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

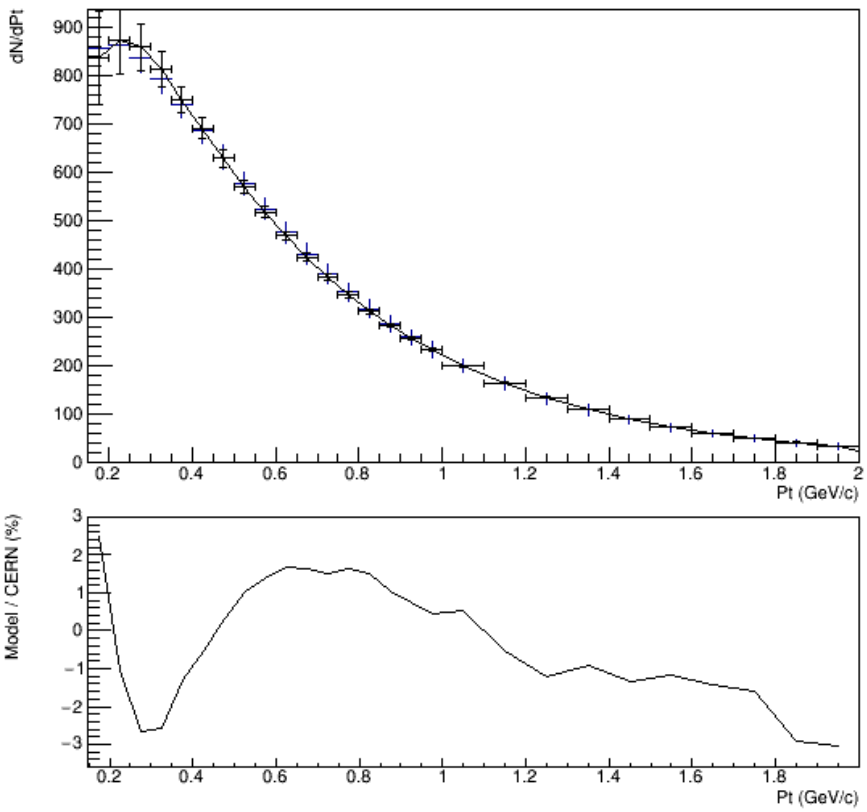

Figure B.3: Charged particle yield (dN/dP_t) over the transverse momentum (P_t) for centrality class $20 - 30\%$. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

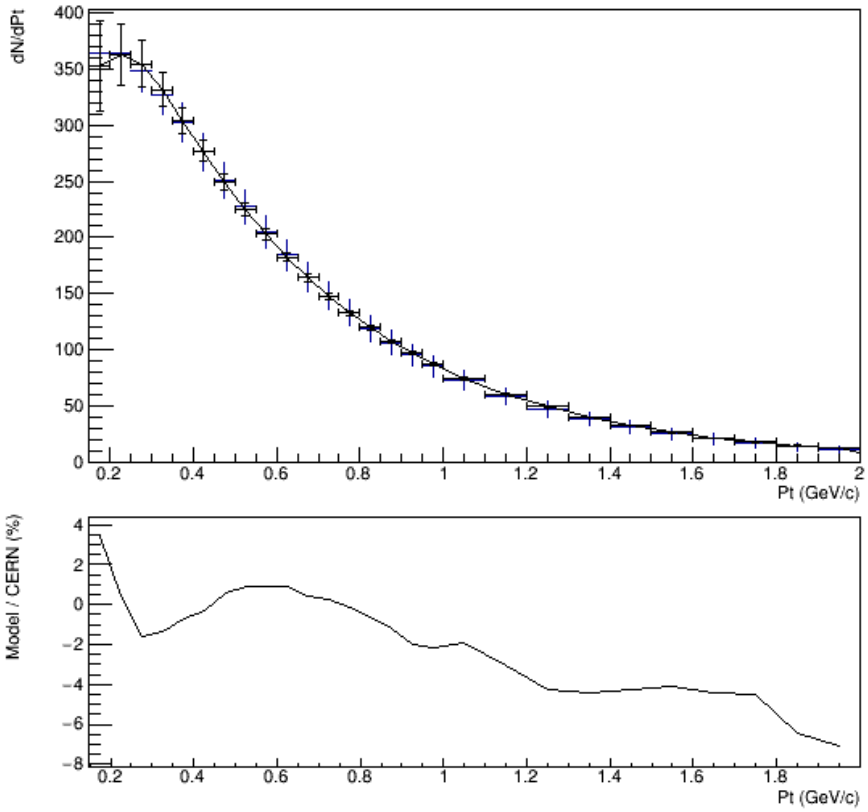

Figure B.4: Charged particle yield (dN/dP_t) over the transverse momentum (P_t) for centrality class $40 - 50\%$. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

Appendix C Gamma

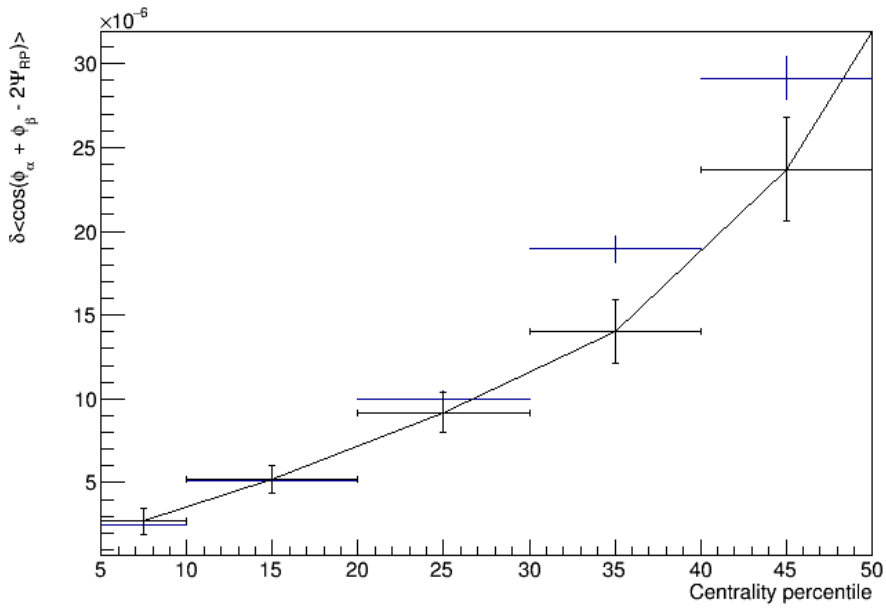

Figure C.1: Gamma, $\delta\left\langle\cos(\phi_\alpha+\phi_\beta-2\Psi_{RP})\right\rangle$, from Equations 2.3.8 and 2.3.9 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1], scaled by a factor of 0.1.

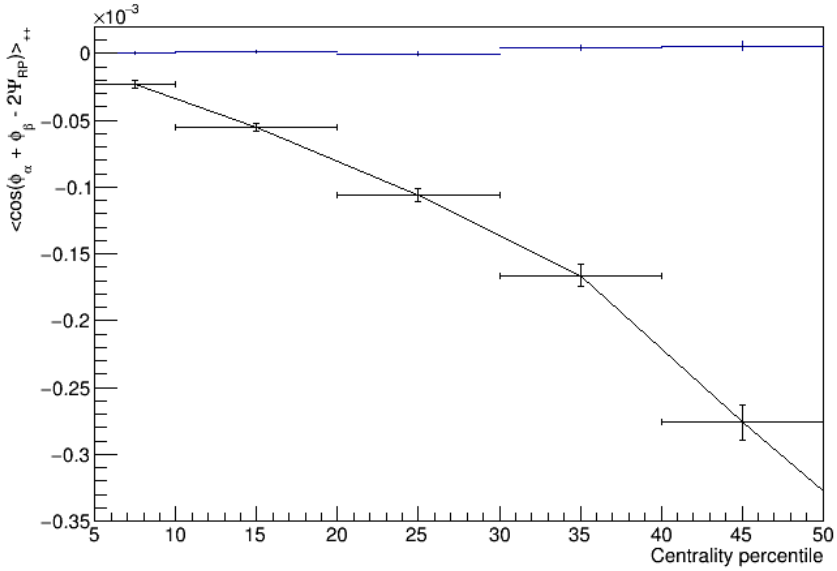

Figure C.2: $\gamma_{+,+}$, $\langle\cos(\phi_+ + \phi_+ - 2\Psi_{RP})\rangle$, from Equation 2.3.8 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

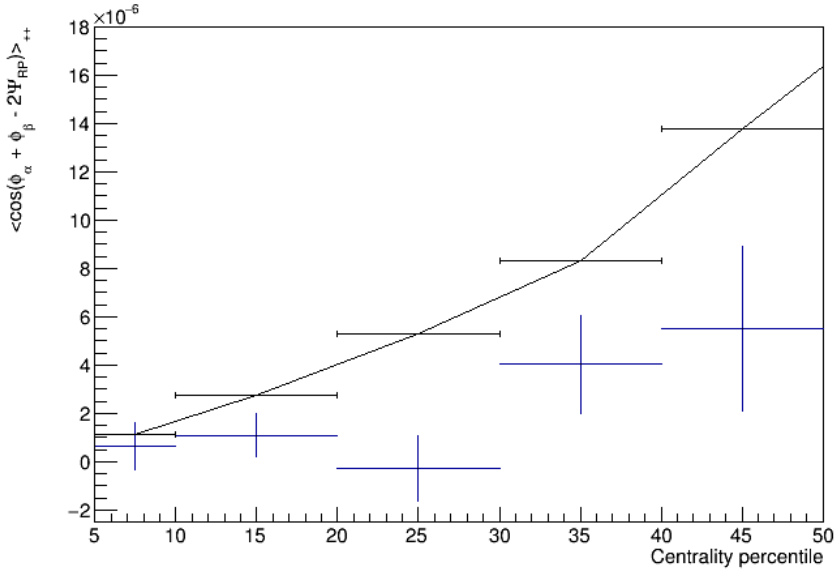

Figure C.3: $\gamma_{+,+}$, $\langle\cos(\phi_+ + \phi_+ - 2\Psi_{RP})\rangle$, from Equation 2.3.8 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1], scaled by a factor of -0.05.

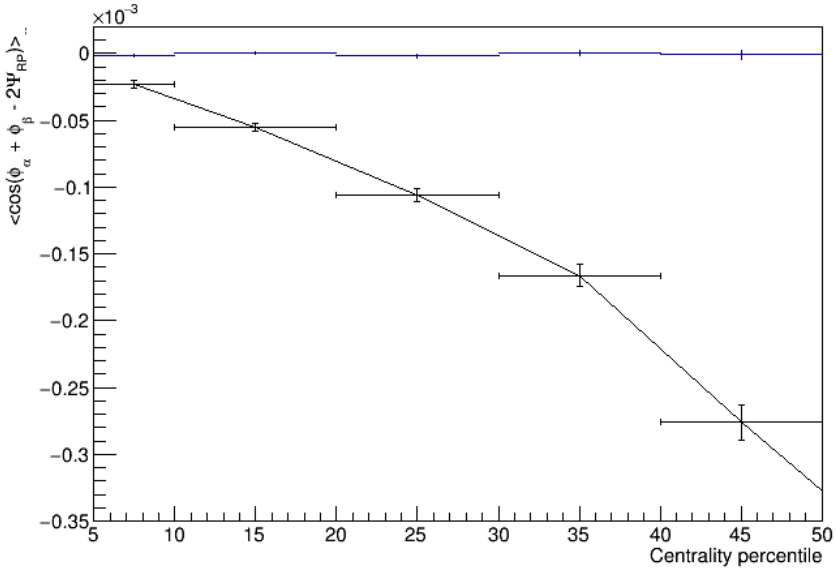

Figure C.4: $\gamma_{-,-}$, $\langle\cos(\phi_- + \phi_- - 2\Psi_{RP})\rangle$, from Equation 2.3.8 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

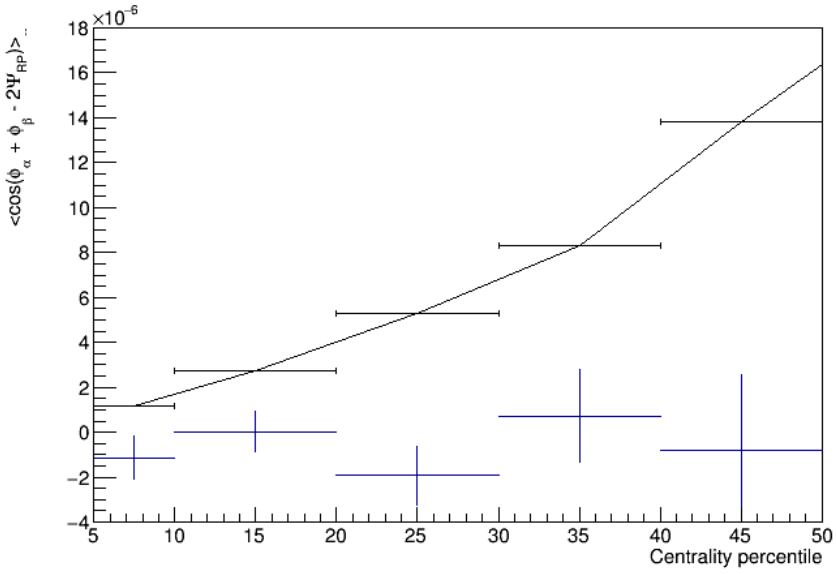

Figure C.5: $\gamma_{-,-}$, $\langle\cos(\phi_- + \phi_- - 2\Psi_{RP})\rangle$, from Equation 2.3.8 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1], scaled by a factor of -0.05.

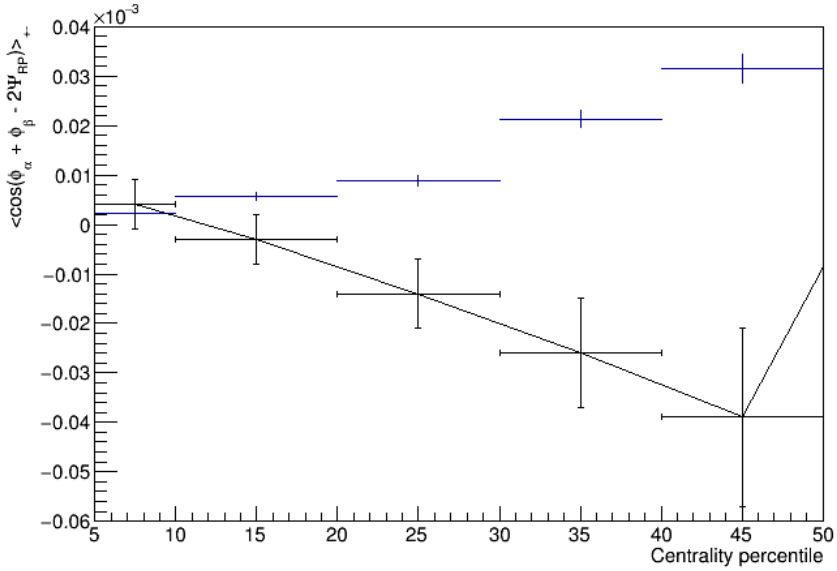

Figure C.6: $\gamma_{+,-}$, $\langle\cos(\phi_+ + \phi_- - 2\Psi_{RP})\rangle$, from Equation 2.3.8 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1]

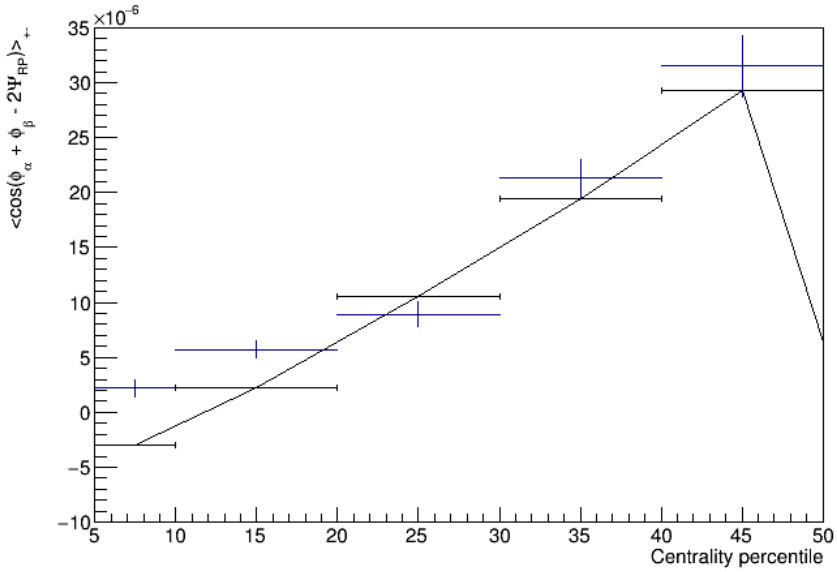

Figure C.7: $\gamma_{+,-}$, $\langle\cos(\phi_+ + \phi_- - 2\Psi_{RP})\rangle$, from Equation 2.3.8 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1], scaled by a factor of -0.75.

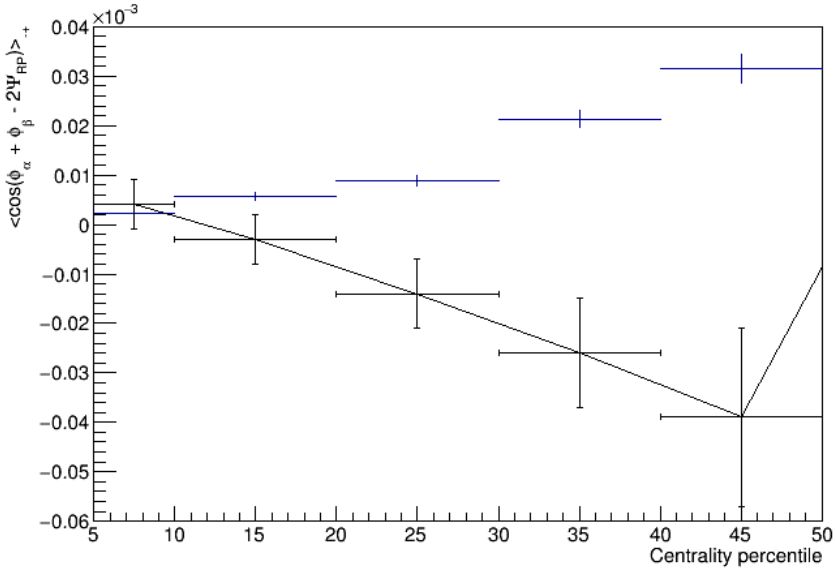

Figure C.8: $\gamma_{-,+}$, $\langle\cos(\phi_- + \phi_+ - 2\Psi_{RP})\rangle$, from Equation 2.3.8 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

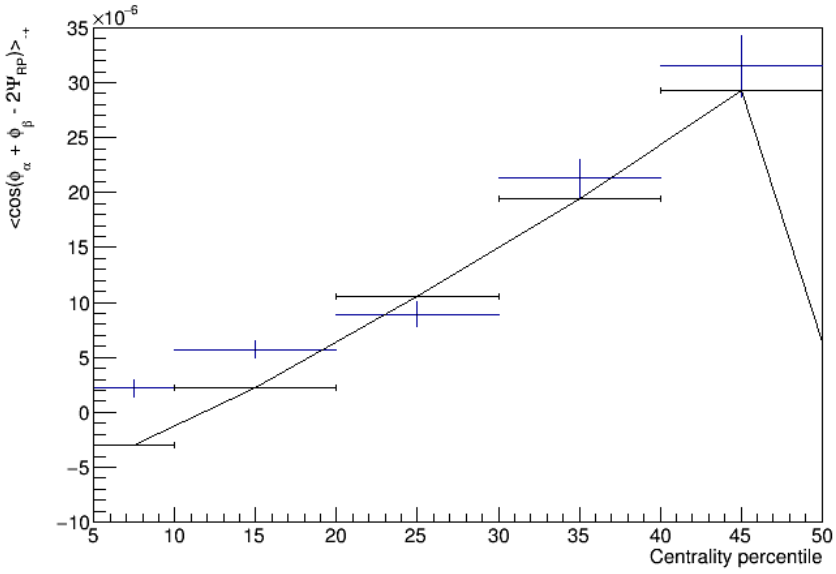

Figure C.9: $\gamma_{-,+}$, $\langle\cos(\phi_- + \phi_+ - 2\Psi_{RP})\rangle$, from Equation 2.3.8 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1], scaled by a factor of -0.75.

Appendix D Delta

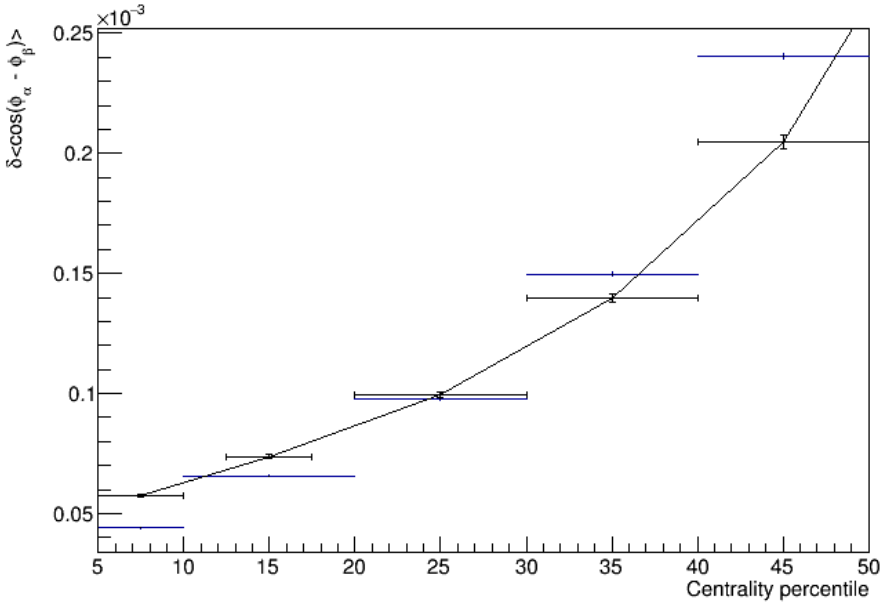

Figure D.1: Delta, $\delta \langle \cos(\phi_\alpha - \phi_\beta) \rangle$, from Equations 2.3.7 and 2.3.9 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1], scaled by a factor of 0.15.

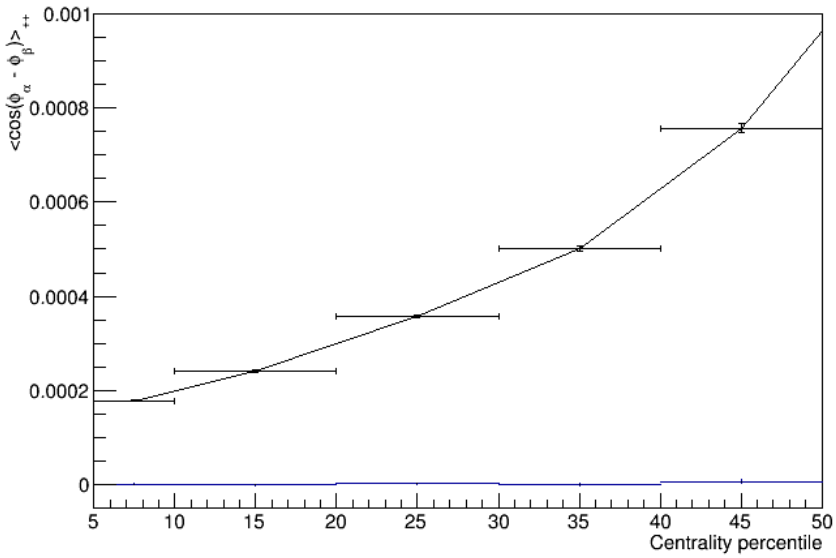

Figure D.2: $\delta_{+,+}$, $\delta_{+,+}$, $\langle\cos(\phi_+ - \phi_+)\rangle$, from Equation 2.3.7 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

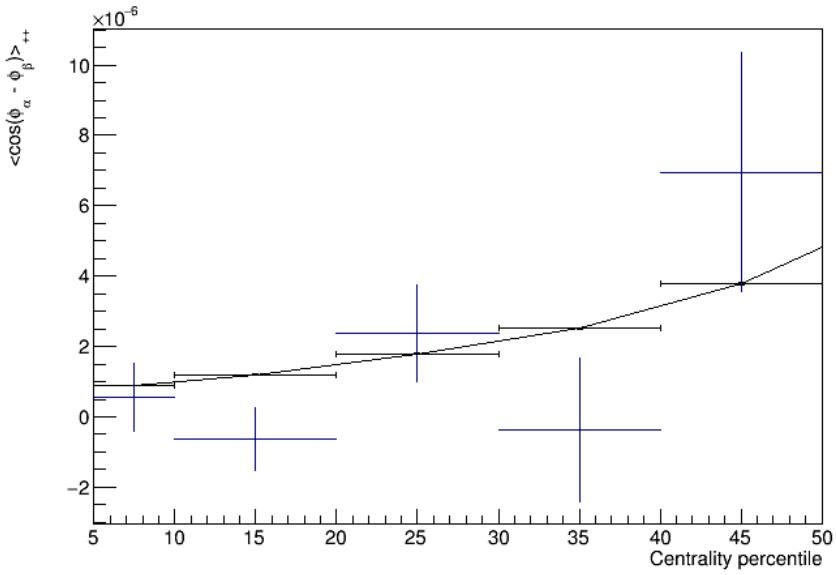

Figure D.3: $\delta_{+,+}$, $\langle\cos(\phi_+ - \phi_+)\rangle$, from Equation 2.3.7 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1], scaled by a factor of 0.05.

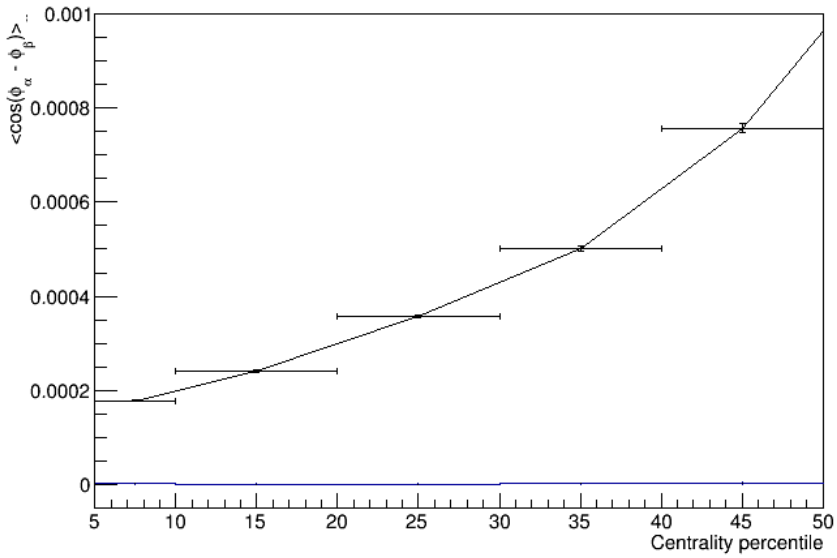

Figure D.4: $\delta_{-,-}$, $\langle\cos(\phi_- - \phi_-)\rangle$, from Equation 2.3.7 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

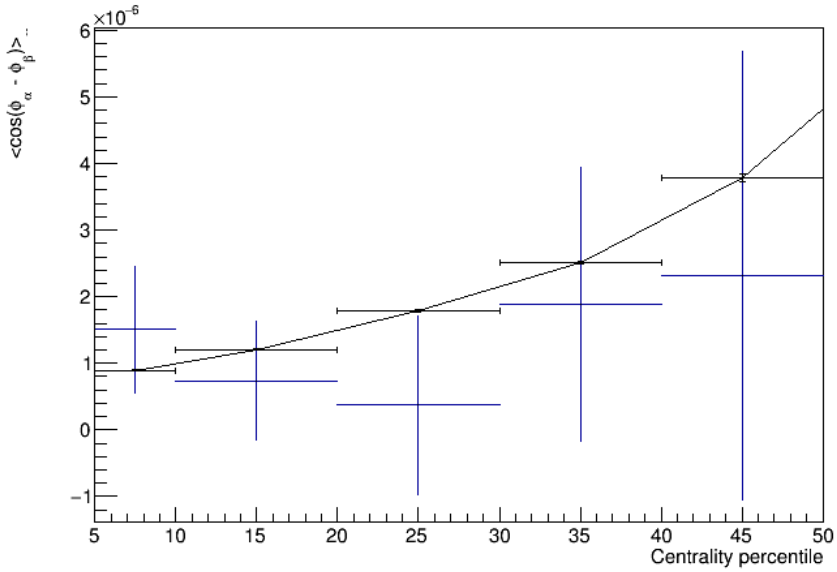

Figure D.5: $\delta_{-,-}$, $\langle\cos(\phi_- - \phi_-)\rangle$, from Equation 2.3.7 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1], scaled by a factor of 0.05.

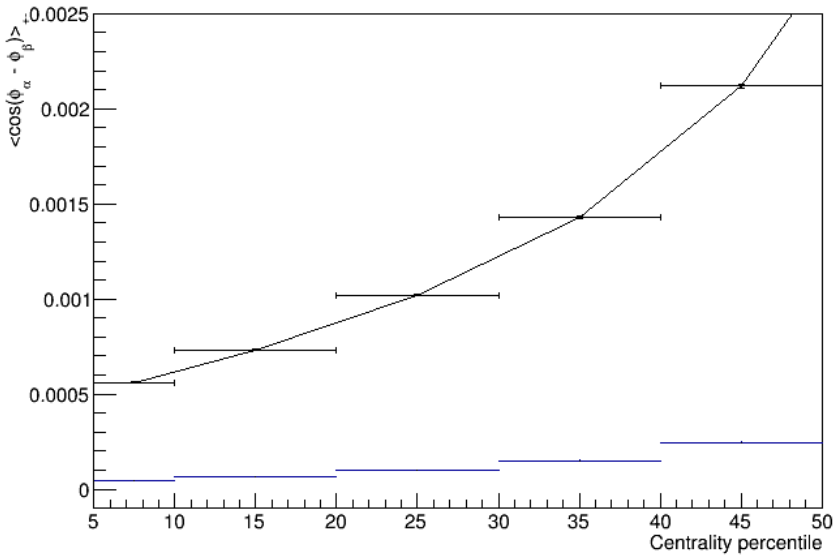

Figure D.6: $\delta_{+,-}$, $\langle\cos(\phi_+ - \phi_-)\rangle$, from Equation 2.3.7 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

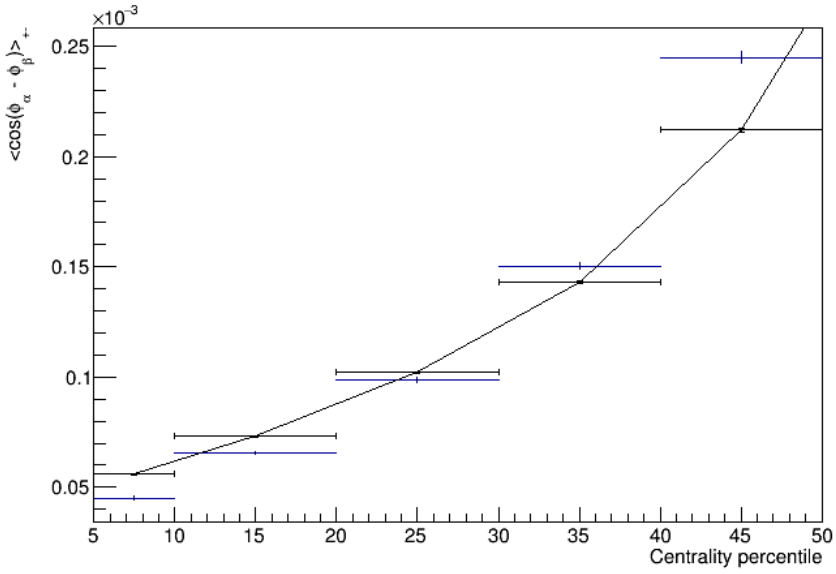

Figure D.7: $\delta_{+,-}$, $\langle\cos(\phi_+ - \phi_-)\rangle$, from Equation 2.3.7 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1], scaled by a factor of 0.1.

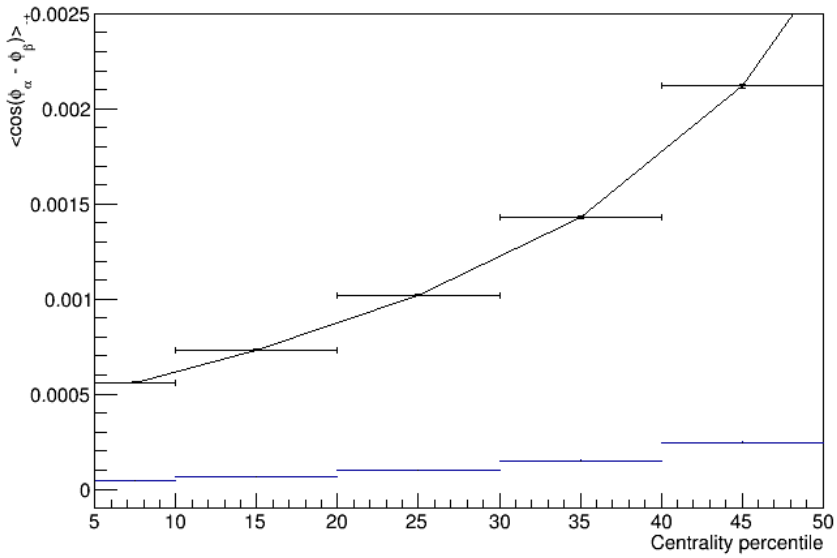

Figure D.8: $\delta_{-,+}$, $\langle\cos(\phi_- - \phi_+)\rangle$, from Equation 2.3.7 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

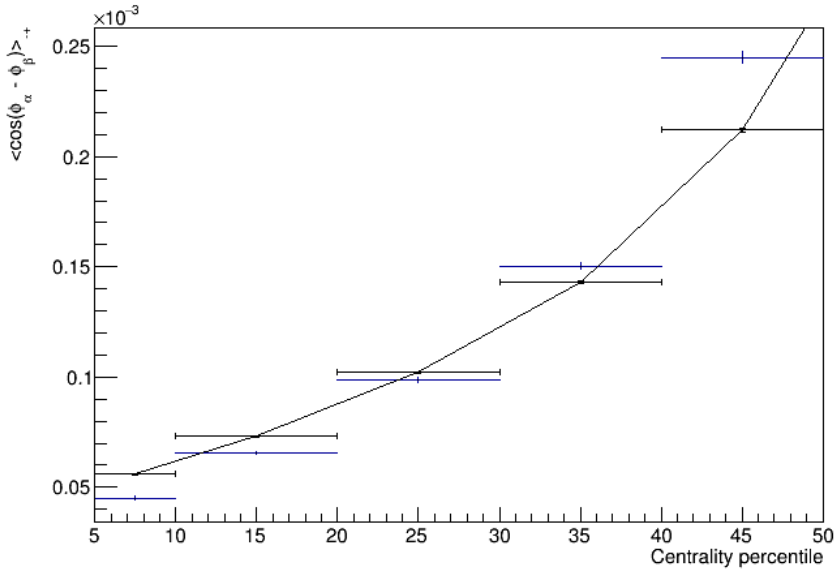

Figure D.9: $\delta_{-,+}$, $\langle\cos(\phi_- - \phi_+)\rangle$, from Equation 2.3.7 over centrality. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1], scaled by a factor of 0.1.

Appendix E Parameter Variation

E.1 Temperature

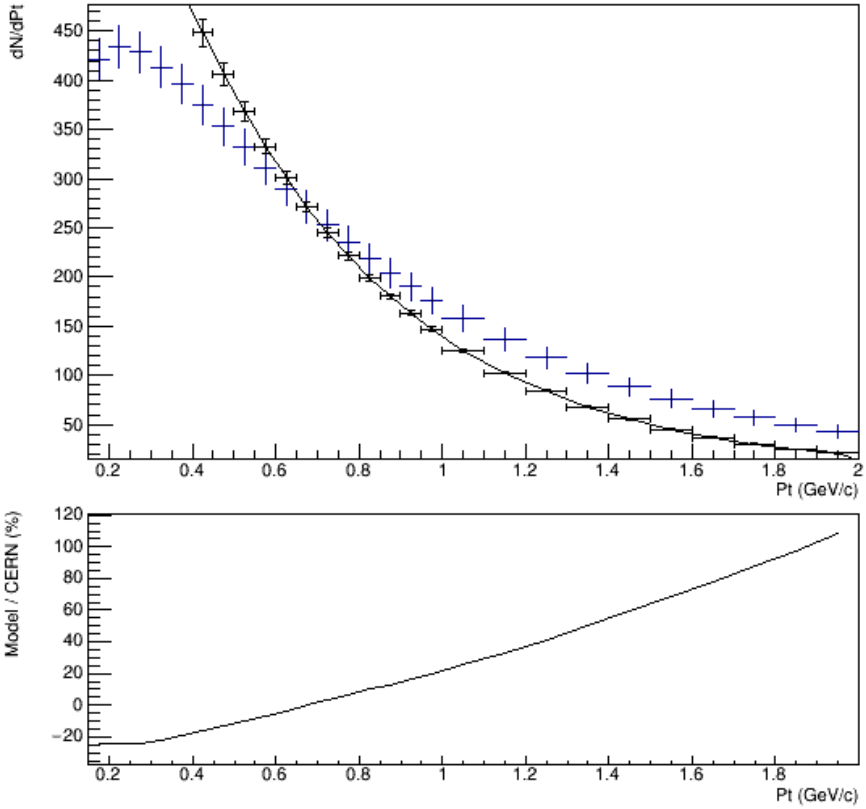

Figure E.1: Charged particle yield (dN/dP_t) over the transverse momentum (P_t) for centrality class $40 - 50\%$, with the Temperature increased by 50%. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

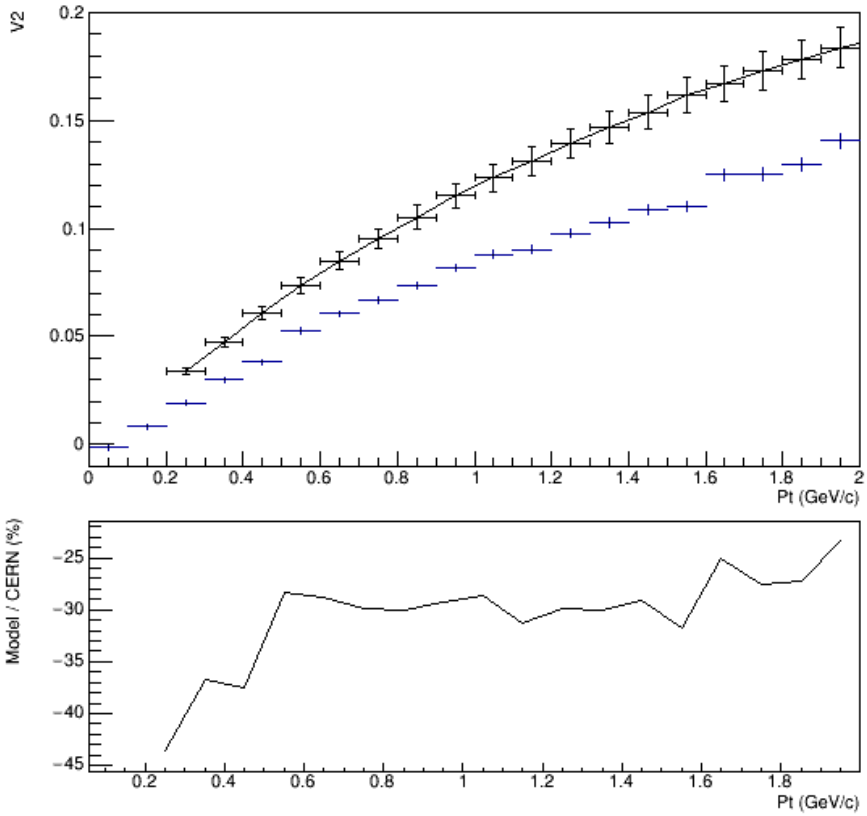

Figure E.2: Elliptic flow (V_2) over transverse momentum (P_t) for centrality class $30 - 40\%$, with the Temperature the increased by 50%. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

E.2 ρ_0

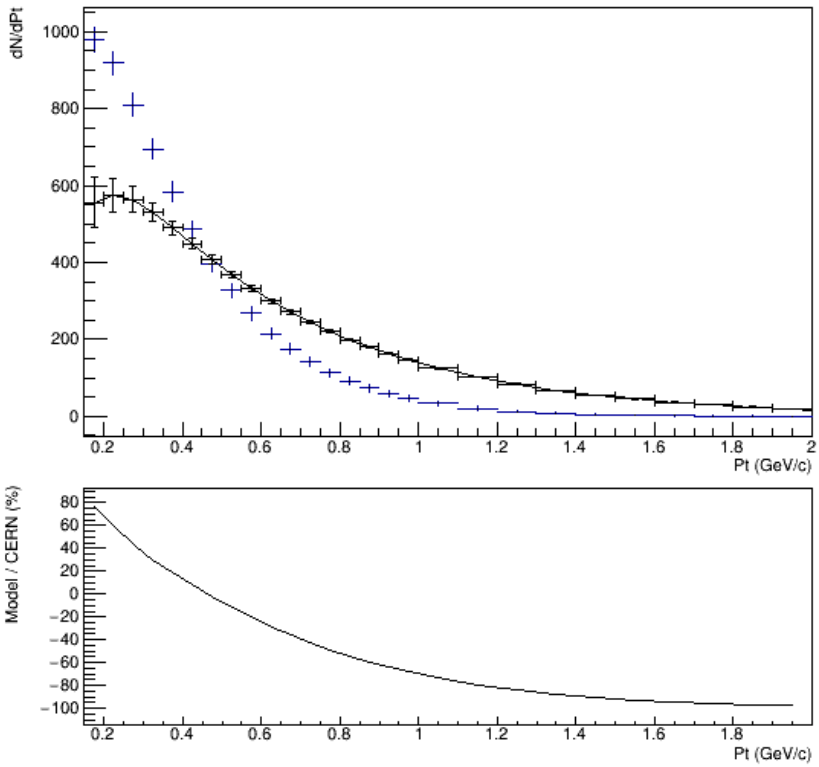

Figure E.3: Charged particle yield (dN/dP_t) over the transverse momentum (P_t) for centrality class $40 - 50\%$, with ρ_0 scaled down by a factor of 0.67. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

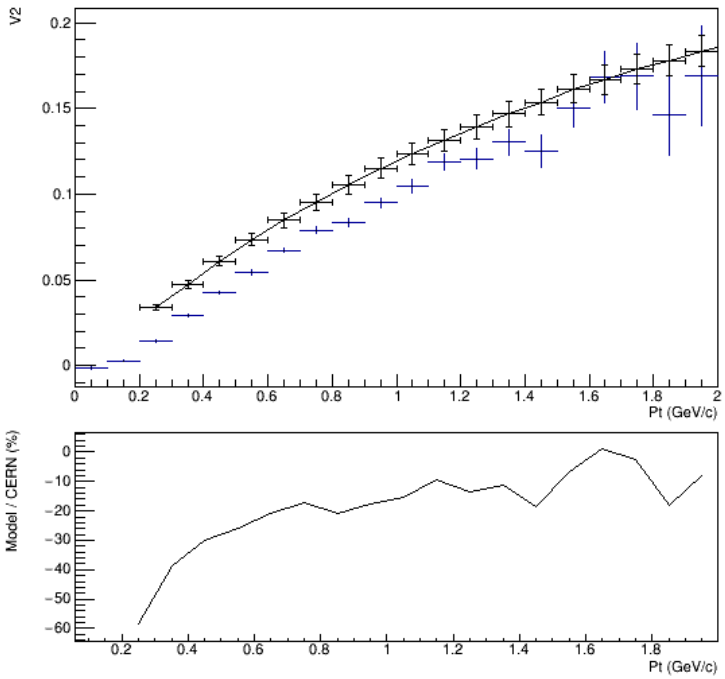

Figure E.4: Elliptic flow (V_2) over transverse momentum (P_t) for centrality class $30 - 40\%$, with ρ_0 scaled down by a factor of 0.67. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

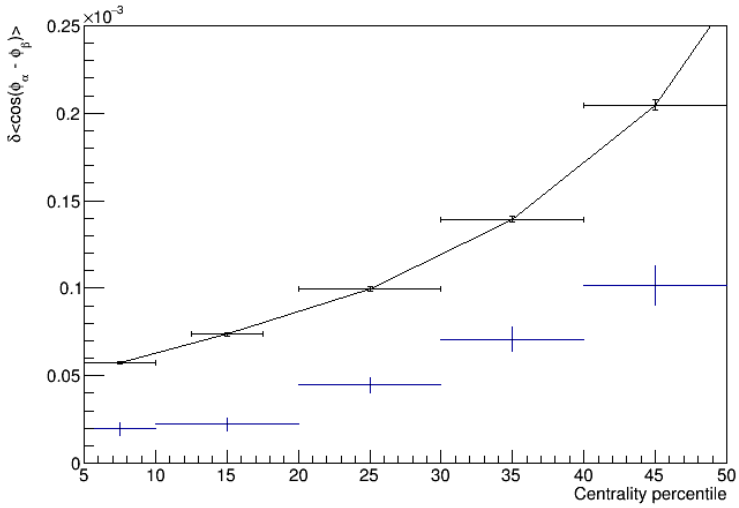

Figure E.5: Delta, $\delta\langle\cos(\phi_\alpha - \phi_\beta)\rangle$, from Equations 2.3.7 and 2.3.9 over centrality, with ρ_0 scaled down by a factor of 0.67. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1], scaled by a factor of 0.15.

E.3 ρ_2

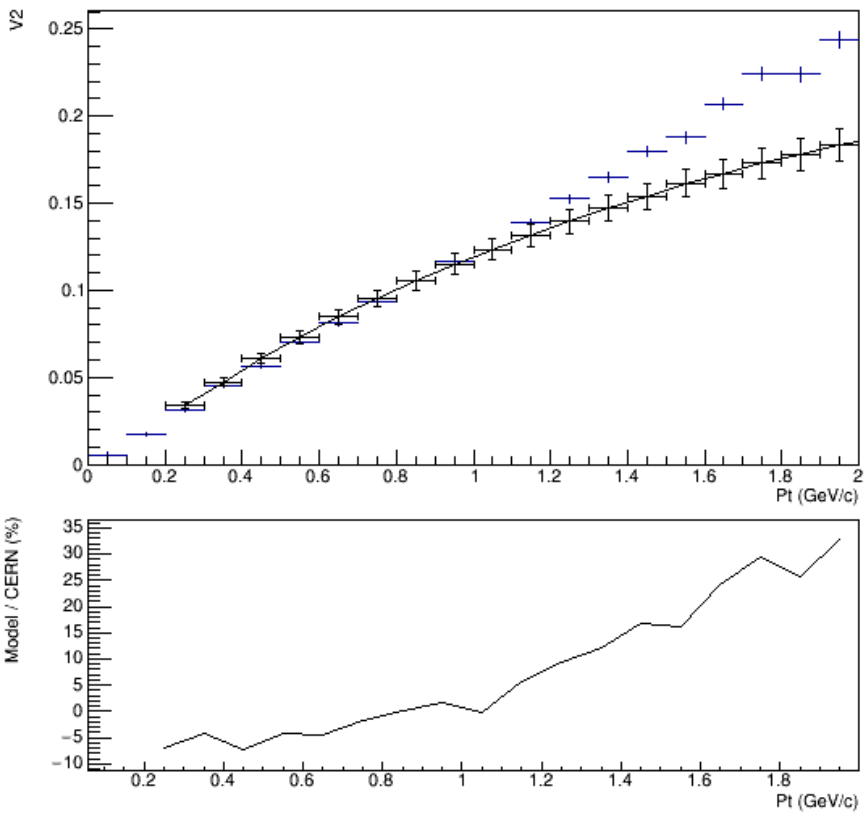

Figure E.6: Elliptic flow (V_2) over transverse momentum (P_t) for centrality class $30 - 40\%$, with ρ_2 increased by 50%. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

E.4 Eccentricity

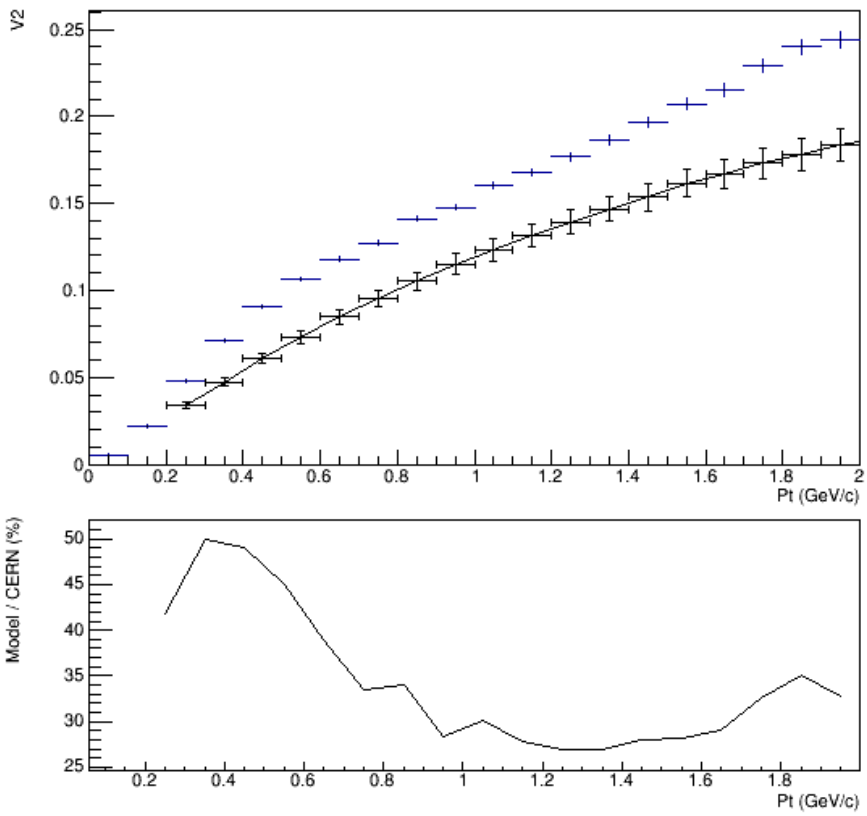

Figure E.7: Elliptic flow (V_2) over transverse momentum (P_t) for centrality class $30 - 40\%$, with the Eccentricity increased by 50%. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1].

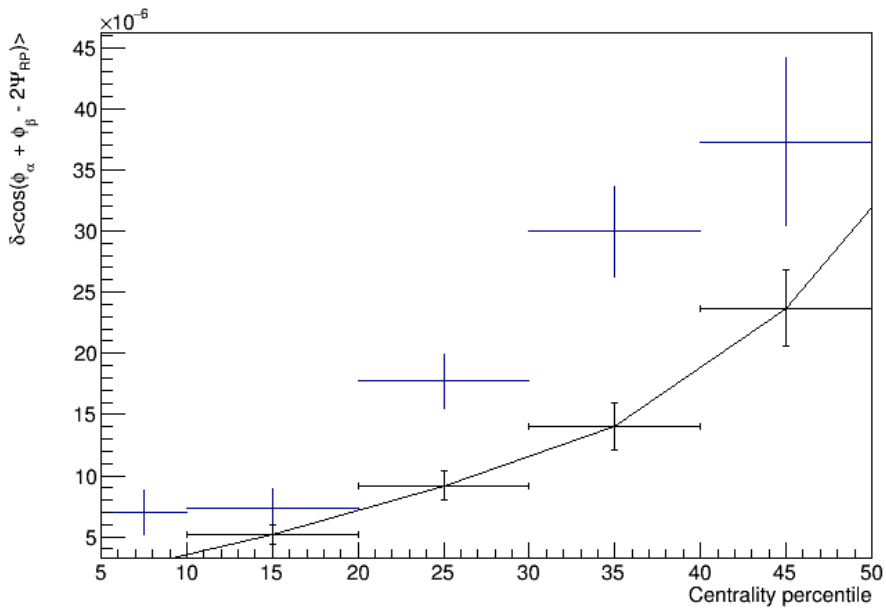

Figure E.8: Gamma, $\delta \langle \cos(\phi_\alpha + \phi_\beta - 2\Psi_{RP}) \rangle$, from Equations 2.3.8 and 2.3.9 over centrality, with the Eccentricity increased by 50%. In blue results from our blast-wave model are shown. The black line represents measurements conducted at the LHC [1], scaled by a factor of 0.1.

Appendix F Root Code

```
//
    *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

//*-*  J. Margutti
//*-*  S. de Vries
//*-*  LCC Model
//
    *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

///
//
    /=========================================================================


#include <TH1.h>
#include <TH2.h>
#include <TStyle.h>
#include <TCanvas.h>
#include <TGraphAsymmErrors.h>
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <TFile.h>
#include <TDatime.h>
#include <TRandom.h>
#include <TTree.h>
#include <TProfile.h>
#include <TString.h>
#include <TSystem.h>
#include <TF1.h>
#include <TText.h>
#include <TNtuple.h>
#include <TMath.h>
#include <fstream>
#include <string>
#include "TF2.h"
#include "Math/WrappedMultiTF1.h"
#include "Math/AdaptiveIntegratorMultiDim.h"

using namespace std;
using namespace TMath;

double PI=TMath::Pi();

void LCCModeladjusted(int nev=60000)
{
 TFile *hfile = new TFile("LCCModelOutput.root","RECREATE");

 // our parameters
 const int nCen = 5; // number of centrality bins
```

```
Double_t T[nCen]; // Temperature
Double_t avmul[nCen]; //Average Multiplicity
Double_t PRadB[nCen]; // Power of the dependence between radius and
    boost
Double_t Rho[nCen][3]; // Fourier coefficients of boost: v0, v2, v4
Double_t Ecc[nCen]; // Eccentricity of source
Double_t fPtMin = 0.15;//0.15;
Double_t fPtMax = 2;
Double_t fMass = 0.14;
long int Bin[nCen];
double Gammacortemp;
double Deltacortemp;

TVector3 bvec3; // boost vector
TLorentzVector* bvec; // boost vector
TLorentzVector* lp1; // particle 1 vector
TLorentzVector* lp2; // particle 2 vector
TF1 *fBoltz1[nCen]; // Boltz. distrib. particle 1
TF1 *fBoltz2[nCen]; // Boltz. distrib. particle 2

// Determine seed for gRandom:
gRandom = new TRandom3(0);

// Set parameters for different centrality classes (from arXiv
    :1208.0603v1):
T[0]=0.112; avmul[0]=13072; PRadB[0]=0.440; Rho[0][0]=0.930; Rho
    [0][1]=0.00831; Rho[0][2]=0.0015; Ecc[0]=0.0961;
T[1]=0.100; avmul[1]=9752; PRadB[1]=0.363; Rho[1][0]=0.935; Rho
    [1][1]=0.01290; Rho[1][2]=0.0020; Ecc[1]=0.1300;
T[2]=0.099; avmul[2]=6521; PRadB[2]=0.384; Rho[2][0]=0.938; Rho
    [2][1]=0.01540; Rho[2][2]=0.0025; Ecc[2]=0.1720;
T[3]=0.098; avmul[3]=4223; PRadB[3]=0.419; Rho[3][0]=0.940; Rho
    [3][1]=0.01600; Rho[3][2]=0.0028; Ecc[3]=0.2040;
T[4]=0.095; avmul[4]=2583; PRadB[4]=0.450;Rho[4][0]=0.943; Rho
    [4][1]=0.01510; Rho[4][2]=0.0033; Ecc[4]=0.2240;

bvec = new TLorentzVector(); // boost vector
lp1 = new TLorentzVector(); // particle 1 vector
lp2 = new TLorentzVector(); // particle 2 vector

// set Boltz. distrib. for different centrality classes
for(int i=0; i<nCen; i++) {
 fBoltz1[i] = new TF1("Boltz1","x*TMath::Sqrt(x*x-[0]*[0])*TMath::Exp
     (-x/[1])",fPtMin,fPtMax);
 fBoltz1[i]->SetParameter(0,fMass);
 fBoltz1[i]->SetParameter(1,T[i]);
 fBoltz2[i] = new TF1("Boltz2","x*TMath::Sqrt(x*x-[0]*[0])*TMath::Exp
     (-x/[1])",fPtMin,fPtMax);
 fBoltz2[i]->SetParameter(0,fMass);
 fBoltz2[i]->SetParameter(1,T[i]);
}

// distributions
```

```
double p8210_d1x1y1_xval[] = {0.175-0.025, 0.225-0.025, 0.275-0.025,
   0.325-0.025, 0.375-0.025, 0.425-0.025, 0.475-0.025, 0.525-0.025,
   0.575-0.025, 0.625-0.025, 0.675-0.025, 0.725-0.025, 0.775-0.025,
   0.825-0.025, 0.875-0.025, 0.925-0.025, 0.975-0.025, 1.05-0.05,
   1.15-0.05,1.25-0.05, 1.35-0.05, 1.45-0.05, 1.55-0.05, 1.65-0.05,
   1.75-0.05, 1.85-0.05, 1.95-0.05, 2};

 TH1F *PtHisto1 = new TH1F("HPt1","",27,p8210_d1x1y1_xval);
 TProfile *hvnpt1= new TProfile("hvnpt1","",20,0.,fPtMax);
 TH1F *PtHisto2 = new TH1F("HPt2","",27,p8210_d1x1y1_xval);
 TProfile *hvnpt2= new TProfile("hvnpt2","",20,0.,fPtMax);
 TH1F *PtHisto3 = new TH1F("HPt3","",27,p8210_d1x1y1_xval);
 TProfile *hvnpt3= new TProfile("hvnpt3","",20,0.,fPtMax);
 TH1F *PtHisto4 = new TH1F("HPt4","",27,p8210_d1x1y1_xval);
 TProfile *hvnpt4= new TProfile("hvnpt4","",20,0.,fPtMax);
 TH1F *PtHisto5 =  new TH1F("HPt5","",27,p8210_d1x1y1_xval);
 TProfile *hvnpt5= new TProfile("hvnpt5","",20,0.,fPtMax);

 double binxlist[] = {5,10,20,30,40,50};
 TProfile *gammatprof= new TProfile("gammatprof","",5,binxlist);
 TProfile *gammatprofpp= new TProfile("gammatprofpp","",5,binxlist);
 TProfile *gammatprofpn= new TProfile("gammtprofapn","",5,binxlist);
 TProfile *gammatprofnp= new TProfile("gammatprofnp","",5,binxlist);
 TProfile *gammatprofnn= new TProfile("gammatprofnn","",5,binxlist);
 TProfile *deltatprof= new TProfile("deltatprof","",5,binxlist);
 TProfile *deltatprofpp= new TProfile("deltatprofpp","",5,binxlist);
 TProfile *deltatprofpn= new TProfile("deltatprofpn","",5,binxlist);
 TProfile *deltatprofnp= new TProfile("deltatprofnp","",5,binxlist);
 TProfile *deltatprofnn= new TProfile("deltatprofnn","",5,binxlist);

double phioneab;
double phitwoab;
double ReQ1;
double ReQ2;
double ImQ1;
double ImQ2;
double ReQtwo1;
double ImQtwo1;
double ReQtwo2;
double ImQtwo2;
double Nhalf;
double px1;
double py1;
double px2;
double py2;
double Gammapptemp;
double Gammapntemp;
double Gammanptemp;
double Gammanntemp;
double Deltapptemp;
double Deltapntemp;
double Deltanptemp;
double Deltanntemp;
```

```
int bin;

for (int h=0; h<nCen; h++) {
  Bin[h] = 0;
}

  double centrality;
  double pradb;
  double Rx;
  double rho0;
  double rho2;
  double rho4;
  double E1;
  double E2;
  double rads;
  double csthetas;
  double thetas;
  double etas;
  double phib;
  double beta;
  double rhob;
  double p1;
  double cstheta1;
  double pz1;
  double theta1;
  double phi1;
  double pt1;
  double p2;
  double cstheta2;
  double pz2;
  double theta2;
  double phi2;
  double pt2;
  double phis;

  for (int i=0; i<nev; i++) {

    // set random centrality 0-100 %
  centrality = 45.*gRandom->Rndm()+5;
  Int_t BinCenEv=-1;
  for (Int_t c=0; c<50; c+=10) {
   if (centrality>c) BinCenEv++;
  }

  Bin[BinCenEv]++;

  ReQ1 = 0;
  ReQ2 = 0;
  ImQ1 = 0;
  ImQ2 = 0;
  ReQtwo1 = 0;
  ImQtwo1 = 0;
  ReQtwo2 = 0;
```

```
ImQtwo2 = 0;
Nhalf = 0;

for (int j=0; j<avmul[BinCenEv]; j++) {

pradb = PRadB[BinCenEv];
Rx = TMath::Sqrt((1-Ecc[BinCenEv])/(1+Ecc[BinCenEv]));
rho0 = Rho[BinCenEv][0];
rho2 = Rho[BinCenEv][1];
rho4 = Rho[BinCenEv][2];

E1 = fBoltz1[BinCenEv]->GetRandom();
E2 = fBoltz2[BinCenEv]->GetRandom();

// random source within the ellipse
rads=2.;
phis=0.;
while(RadSys(rads,phis,Rx)>1.) {
 rads = sqrt(gRandom->Rndm());
 phis = 2.*PI*gRandom->Rndm();
}
// random pseudorapidity ?
csthetas=2.*(gRandom->Rndm()-0.5);
thetas=acos(csthetas);
etas=-log(tan(thetas/2.));

// get boost vector (normal to the surface of the ellipse), see
    Yasuto's thesis
phib = BosPhi(phis,Rx);
beta = pow(RadSys(rads,phis,Rx),pradb)*(rho0+rho2*cos(2*phib)+rho4*
    cos(4*phib));
rhob = TMath::ATanH(beta);
bvec->SetPxPyPzE(sinh(rhob)*cos(phib),sinh(rhob)*sin(phib),sinh(etas
    )*cosh(rhob),cosh(etas)*cosh(rhob));

// daughters in the rest frame of the source
p1=sqrt(E1*E1-fMass*fMass);
cstheta1=2.*(gRandom->Rndm()-0.5);
pz1=p1*cstheta1;
theta1=acos(cstheta1);
phi1=2.*PI*gRandom->Rndm();
pt1=p1*sin(theta1);

p2=sqrt(E2*E2-fMass*fMass);
cstheta2=2.*(gRandom->Rndm()-0.5);
pz2=p2*cstheta2;
theta2=acos(cstheta2);
phi2=2.*PI*gRandom->Rndm();
pt2=p2*sin(theta2);

// boost daughters
lp1->SetPxPyPzE(pt1*cos(phi1),pt1*sin(phi1),pz1,E1);
lp2->SetPxPyPzE(pt2*cos(phi2),pt2*sin(phi2),pz2,E2);
```

```
  bvec3 = bvec->BoostVector();
  lp1->Boost(bvec3);
  lp2->Boost(bvec3);

// Q-vector calculations, 1 = positive, 2 = negative;
phioneab = lp1->Phi();
phitwoab = lp2->Phi();
ReQ1 += cos(phioneab);
ImQ1 += sin(phioneab);
ReQ2 += cos(phitwoab);
ImQ2 += sin(phitwoab);
ReQtwo1 += cos(2*phioneab);
ImQtwo1 += sin(2*phioneab);
ReQtwo2 += cos(2*phitwoab);
ImQtwo2 += sin(2*phitwoab);
Nhalf++;

switch(BinCenEv) {
case 0:
  PtHisto1->Fill(sqrt(pow(lp1->Px(),2.)+pow(lp1->Py(),2.)));
  PtHisto1->Fill(sqrt(pow(lp2->Px(),2.)+pow(lp2->Py(),2.)));
  hvnpt1->Fill(sqrt(pow(lp1->Px(),2.)+pow(lp1->Py(),2.)),cos(2*atan2(
      lp1->Py(),lp1->Px())));
  hvnpt1->Fill(sqrt(pow(lp2->Px(),2.)+pow(lp2->Py(),2.)),cos(2*atan2(
      lp2->Py(),lp2->Px())));
 break;
case 1:
  PtHisto2->Fill(sqrt(pow(lp1->Px(),2.)+pow(lp1->Py(),2.)));
  PtHisto2->Fill(sqrt(pow(lp2->Px(),2.)+pow(lp2->Py(),2.)));
  hvnpt2->Fill(sqrt(pow(lp1->Px(),2.)+pow(lp1->Py(),2.)),cos(2*atan2(
      lp1->Py(),lp1->Px())));
  hvnpt2->Fill(sqrt(pow(lp2->Px(),2.)+pow(lp2->Py(),2.)),cos(2*atan2(
      lp2->Py(),lp2->Px())));
 break;
case 2:
  PtHisto3->Fill(sqrt(pow(lp1->Px(),2.)+pow(lp1->Py(),2.)));
  PtHisto3->Fill(sqrt(pow(lp2->Px(),2.)+pow(lp2->Py(),2.)));
  hvnpt3->Fill(sqrt(pow(lp1->Px(),2.)+pow(lp1->Py(),2.)),cos(2*atan2(
      lp1->Py(),lp1->Px())));
  hvnpt3->Fill(sqrt(pow(lp2->Px(),2.)+pow(lp2->Py(),2.)),cos(2*atan2(
      lp2->Py(),lp2->Px())));
 break;
case 3:
  PtHisto4->Fill(sqrt(pow(lp1->Px(),2.)+pow(lp1->Py(),2.)));
  PtHisto4->Fill(sqrt(pow(lp2->Px(),2.)+pow(lp2->Py(),2.)));
  hvnpt4->Fill(sqrt(pow(lp1->Px(),2.)+pow(lp1->Py(),2.)),cos(2*atan2(
      lp1->Py(),lp1->Px())));
  hvnpt4->Fill(sqrt(pow(lp2->Px(),2.)+pow(lp2->Py(),2.)),cos(2*atan2(
      lp2->Py(),lp2->Px())));
 break;
case 4:
  PtHisto5->Fill(sqrt(pow(lp1->Px(),2.)+pow(lp1->Py(),2.)));
  PtHisto5->Fill(sqrt(pow(lp2->Px(),2.)+pow(lp2->Py(),2.)));
```

```
  hvnpt5->Fill(sqrt(pow(lp1->Px(),2.)+pow(lp1->Py(),2.)),cos(2*atan2(
      lp1->Py(),lp1->Px())));
  hvnpt5->Fill(sqrt(pow(lp2->Px(),2.)+pow(lp2->Py(),2.)),cos(2*atan2(
      lp2->Py(),lp2->Px())));
 break;
default:
 break;
 } // end of switch

 } // end of for (int j==0; j<avmul; j++)

// gamma/delta calculations
Gammapptemp= (ReQ1*ReQ1-ImQ1*ImQ1-ReQtwo1)/(Nhalf*(Nhalf-1));
Gammapntemp= (ReQ1*ReQ2-ImQ1*ImQ2)/(Nhalf*Nhalf);
Gammanptemp= (ReQ2*ReQ1-ImQ2*ImQ1)/(Nhalf*Nhalf);
Gammanntemp= (ReQ2*ReQ2-ImQ2*ImQ2-ReQtwo2)/(Nhalf*(Nhalf-1));
Deltapptemp = (ReQ1*ReQ1+ImQ1*ImQ1-Nhalf)/(Nhalf*(Nhalf-1));
Deltapntemp = (ReQ1*ReQ2+ImQ1*ImQ2)/(Nhalf*Nhalf);
Deltanptemp = (ReQ2*ReQ1+ImQ2*ImQ1)/(Nhalf*Nhalf);
Deltanntemp = (ReQ2*ReQ2+ImQ2*ImQ2-Nhalf)/(Nhalf*(Nhalf-1));
Gammacortemp = 0.5*(Gammapntemp+Gammanptemp-Gammapptemp-Gammanntemp);
Deltacortemp = 0.5*(Deltapntemp+Deltanptemp-Deltapptemp-Deltanntemp);

gammatprof->Fill(centrality,Gammacortemp);
gammatprofpp->Fill(centrality,Gammapptemp);
gammatprofpn->Fill(centrality,Gammapntemp);
gammatprofnp->Fill(centrality,Gammanptemp);
gammatprofnn->Fill(centrality,Gammanntemp);
deltatprof->Fill(centrality,Deltacortemp);
deltatprofpp->Fill(centrality,Deltapptemp);
deltatprofpn->Fill(centrality,Deltapntemp);
deltatprofnp->Fill(centrality,Deltanptemp);
deltatprofnn->Fill(centrality,Deltanntemp);

int thousand=1000;
if ((i%thousand)==0)
{
        printf("ready␣%e␣\n",i);
}

} // end of for (int i=0; i<nev; i++)

   TCanvas *c1 = new TCanvas("c1","Deltapp",200,10,700,500);
   deltatprofpp->Draw();
   deltatprofpp->GetXaxis()->SetTitle("Centrality␣percentile");
   deltatprofpp->GetYaxis()->SetTitle("<cos(#phi_{#alpha}␣-␣#phi_{#
       beta})>_{++}");
   deltatprofpp->GetYaxis()->SetTitleOffset(1.45);
   deltatprofpp->TH1::SetStats(kFALSE);
   graph = GetDelta(1);
   graph->Draw("same");

   TCanvas *c2 = new TCanvas("c2","Deltapn",200,10,700,500);
```

```
    deltatprofpn->Draw();
    deltatprofpn->GetXaxis()->SetTitle("Centrality␣percentile");
    deltatprofpn->GetYaxis()->SetTitle("<cos(#phi_{#alpha}␣-␣#phi_{#
        beta})>_{+-}");
    deltatprofpn->GetYaxis()->SetTitleOffset(1.45);
    deltatprofpn->TH1::SetStats(kFALSE);
    graph = GetDelta(0);
    graph->Draw("same");

    TCanvas *c3 = new TCanvas("c3","Gamma",200,10,700,500);
    gammatprof->Draw();
    gammatprof->GetXaxis()->SetTitle("Centrality␣percentile");
    gammatprof->GetYaxis()->SetTitle("#delta<cos(#phi_{#alpha}␣+␣#phi_
        {#beta}␣-␣2#Psi_{RP})>");
    gammatprof->GetYaxis()->SetTitleOffset(1.45);
    gammatprof->TH1::SetStats(kFALSE);
    graph = GetGamma(2);
    graph->Draw("same");

    TCanvas *c4 = new TCanvas("c4","Delta",200,10,700,500);
    deltatprof->Draw();
    deltatprof->GetXaxis()->SetTitle("Centrality␣percentile");
    deltatprof->GetYaxis()->SetTitle("#delta<cos(#phi_{#alpha}␣-␣#phi_
        {#beta})>");
    deltatprof->GetYaxis()->SetTitleOffset(1.45);
    deltatprof->TH1::SetStats(kFALSE);
    graph = GetDelta(2);
    graph->Draw("same");

    TCanvas *c9 = new TCanvas("c9","Deltanp",200,10,700,500);
    deltatprofnp->Draw();
    deltatprofnp->GetXaxis()->SetTitle("Centrality␣percentile");
    deltatprofnp->GetYaxis()->SetTitle("<cos(#phi_{#alpha}␣-␣#phi_{#
        beta})>_{-+}");
    deltatprofnp->GetYaxis()->SetTitleOffset(1.45);
    deltatprofnp->TH1::SetStats(kFALSE);
    graph = GetDelta(0);
    graph->Draw("same");

    TCanvas *c10= new TCanvas("c10","Deltann",200,10,700,500);
    deltatprofnn->Draw();
    deltatprofnn->GetXaxis()->SetTitle("Centrality␣percentile");
    deltatprofnn->GetYaxis()->SetTitle("<cos(#phi_{#alpha}␣-␣#phi_{#
        beta})>_{--}");
    deltatprofnn->GetYaxis()->SetTitleOffset(1.45);
    deltatprofnn->TH1::SetStats(kFALSE);
    graph = GetDelta(1);
    graph->Draw("same");

TCanvas *c11 = new TCanvas("c11","Gammapp",200,10,700,500);
    gammatprofpp->Draw();
    gammatprofpp->GetXaxis()->SetTitle("Centrality␣percentile");
    gammatprofpp->GetYaxis()->SetTitle("<cos(#phi_{#alpha}␣+␣#phi_{#
```

```
        beta}␣-␣2#Psi_{RP})>_{++}");
    gammatprofpp->GetYaxis()->SetTitleOffset(1.45);
    gammatprofpp->TH1::SetStats(kFALSE);
    graph = GetGamma(1);
    graph->Draw("same");

TCanvas *c12 = new TCanvas("c12","Gammapn",200,10,700,500);
    gammatprofpn->Draw();
    gammatprofpn->GetXaxis()->SetTitle("Centrality␣percentile");
    gammatprofpn->GetYaxis()->SetTitle("<cos(#phi_{#alpha}␣+␣#phi_{#
        beta}␣-␣2#Psi_{RP})>_{+-}");
    gammatprofpn->GetYaxis()->SetTitleOffset(1.45);
    gammatprofpn->TH1::SetStats(kFALSE);
    graph = GetGamma(0);
    graph->Draw("same");

TCanvas *c13 = new TCanvas("c13","Gammanp",200,10,700,500);
    gammatprofnp->Draw();
    gammatprofnp->GetXaxis()->SetTitle("Centrality␣percentile");
    gammatprofnp->GetYaxis()->SetTitle("<cos(#phi_{#alpha}␣+␣#phi_{#
        beta}␣-␣2#Psi_{RP})>_{-+}");
    gammatprofnp->GetYaxis()->SetTitleOffset(1.45);
    gammatprofnp->TH1::SetStats(kFALSE);
    graph = GetGamma(0);
    graph->Draw("same");

TCanvas *c14 = new TCanvas("c14","Gammann",200,10,700,500);
    gammatprofnn->Draw();
    gammatprofnn->GetXaxis()->SetTitle("Centrality␣percentile");
    gammatprofnn->GetYaxis()->SetTitle("<cos(#phi_{#alpha}␣+␣#phi_{#
        beta}␣-␣2#Psi_{RP})>_{--}");
    gammatprofnn->GetYaxis()->SetTitleOffset(1.45);
    gammatprofnn->TH1::SetStats(kFALSE);
    graph = GetGamma(1);
    graph->Draw("same");

//=======================================

// Scaled

TCanvas *sc1 = new TCanvas("sc1","sDeltapp",200,10,700,500);
    deltatprofpp->Draw();
    deltatprofpp->GetXaxis()->SetTitle("Centrality␣percentile");
    deltatprofpp->GetYaxis()->SetTitle("<cos(#phi_{#alpha}␣-␣#phi_{#
        beta})>_{++}");
    deltatprofpp->GetYaxis()->SetTitleOffset(1.45);
    deltatprofpp->TH1::SetStats(kFALSE);
    graph = GetDeltascaled(1);
    graph->Draw("same");

    TCanvas *sc2 = new TCanvas("sc2","sDeltapn",200,10,700,500);
    deltatprofpn->Draw();
    deltatprofpn->GetXaxis()->SetTitle("Centrality␣percentile");
```

```
    deltatprofpn->GetYaxis()->SetTitle("<cos(#phi_{#alpha}␣-␣#phi_{#
        beta})>_{+-}");
    deltatprofpn->GetYaxis()->SetTitleOffset(1.45);
    deltatprofpn->TH1::SetStats(kFALSE);
    graph = GetDeltascaled(0);
    graph->Draw("same");

    TCanvas *sc3 = new TCanvas("sc3","sGamma",200,10,700,500);
    gammatprof->Draw();
    gammatprof->GetXaxis()->SetTitle("Centrality␣percentile");
    gammatprof->GetYaxis()->SetTitle("#delta<cos(#phi_{#alpha}␣+␣#phi_
        {#beta}␣-␣2#Psi_{RP})>");
    gammatprof->GetYaxis()->SetTitleOffset(1.45);
    gammatprof->TH1::SetStats(kFALSE);
    graph = GetGammascaled(2);
    graph->Draw("same");

    TCanvas *sc4 = new TCanvas("sc4","sDelta",200,10,700,500);
    deltatprof->Draw();
    deltatprof->GetXaxis()->SetTitle("Centrality␣percentile");
    deltatprof->GetYaxis()->SetTitle("#delta<cos(#phi_{#alpha}␣-␣#phi_
        {#beta})>");
    deltatprof->GetYaxis()->SetTitleOffset(1.45);
    deltatprof->TH1::SetStats(kFALSE);
    graph = GetDeltascaled(2);
    graph->Draw("same");

    TCanvas *sc9 = new TCanvas("sc9","sDeltanp",200,10,700,500);
    deltatprofnp->Draw();
    deltatprofnp->GetXaxis()->SetTitle("Centrality␣percentile");
    deltatprofnp->GetYaxis()->SetTitle("<cos(#phi_{#alpha}␣-␣#phi_{#
        beta})>_{-+}");
    deltatprofnp->GetYaxis()->SetTitleOffset(1.45);
    deltatprofnp->TH1::SetStats(kFALSE);
    graph = GetDeltascaled(0);
    graph->Draw("same");

    TCanvas *sc10= new TCanvas("sc10","sDeltann",200,10,700,500);
    deltatprofnn->Draw();
    deltatprofnn->GetXaxis()->SetTitle("Centrality␣percentile");
    deltatprofnn->GetYaxis()->SetTitle("<cos(#phi_{#alpha}␣-␣#phi_{#
        beta})>_{--}");
    deltatprofnn->GetYaxis()->SetTitleOffset(1.45);
    deltatprofnn->TH1::SetStats(kFALSE);
    graph = GetDeltascaled(1);
    graph->Draw("same");

TCanvas *sc11 = new TCanvas("sc11","sGammapp",200,10,700,500);
    gammatprofpp->Draw();
    gammatprofpp->GetXaxis()->SetTitle("Centrality␣percentile");
    gammatprofpp->GetYaxis()->SetTitle("<cos(#phi_{#alpha}␣+␣#phi_{#
        beta}␣-␣2#Psi_{RP})>_{++}");
    gammatprofpp->GetYaxis()->SetTitleOffset(1.45);
```

```
   gammatprofpp->TH1::SetStats(kFALSE);
   graph = GetGammascaled(1);
   graph->Draw("same");

TCanvas *sc12 = new TCanvas("sc12","sGammapn",200,10,700,500);
   gammatprofpn->Draw();
   gammatprofpn->GetXaxis()->SetTitle("Centrality␣percentile");
   gammatprofpn->GetYaxis()->SetTitle("<cos(#phi_{#alpha}␣+␣#phi_{#
       beta}␣-␣2#Psi_{RP})>_{+-}");
   gammatprofpn->GetYaxis()->SetTitleOffset(1.45);
   gammatprofpn->TH1::SetStats(kFALSE);
   graph = GetGammascaled(0);
   graph->Draw("same");

TCanvas *sc13 = new TCanvas("sc13","sGammanp",200,10,700,500);
   gammatprofnp->Draw();
   gammatprofnp->GetXaxis()->SetTitle("Centrality␣percentile");
   gammatprofnp->GetYaxis()->SetTitle("<cos(#phi_{#alpha}␣+␣#phi_{#
       beta}␣-␣2#Psi_{RP})>_{-+}");
   gammatprofnp->GetYaxis()->SetTitleOffset(1.45);
   gammatprofnp->TH1::SetStats(kFALSE);
   graph = GetGammascaled(0);
   graph->Draw("same");

TCanvas *sc14 = new TCanvas("sc14","sGammann",200,10,700,500);
   gammatprofnn->Draw();
   gammatprofnn->GetXaxis()->SetTitle("Centrality␣percentile");
   gammatprofnn->GetYaxis()->SetTitle("<cos(#phi_{#alpha}␣+␣#phi_{#
       beta}␣-␣2#Psi_{RP})>_{--}");
   gammatprofnn->GetYaxis()->SetTitleOffset(1.45);
   gammatprofnn->TH1::SetStats(kFALSE);
   graph = GetGammascaled(1);
   graph->Draw("same");

long double sum;
long double sum2;
long double binContent;
double ptxlist[27] = {0.175, 0.225, 0.275, 0.325, 0.375, 0.425, 0.475,
     0.525, 0.575,
    0.625, 0.675, 0.725, 0.775, 0.825, 0.875, 0.925, 0.975, 1.05,
        1.15,
    1.25, 1.35, 1.45, 1.55, 1.65, 1.75, 1.85, 1.95};
double ptylist[27];

TCanvas* canvas1 = new TCanvas("1","Pt␣distirbution␣at␣5-10%␣
    centrality",600,600);
TPad *pad1 = new TPad("pad1", "pad1", 0, 0.4, 1, 1.0);
pad1->SetBottomMargin(0.085); // Upper and lower plot are joined
pad1->Draw();              // Draw the upper pad: pad1
pad1->cd();
graph = GetSpectra(1);
sum = 0;
sum2 = 0;
```

```
for(Int_t i=0; i < 27; i++)
{
 double x,y;
 graph->GetPoint(i,x,y);
 sum += 2*PI*x*y;
 graph->SetPoint(i,x,2*PI*y*x);
}
for (Int_t c=1; c<28; c++)
{
binContent = PtHisto1->GetBinContent(c);
if (c > 17)
{
binContent = binContent / 2;
}
PtHisto1->SetBinContent(c,binContent/Bin[0]);
}
PtHisto1->Draw("EP0");
graph->Draw("same");
PtHisto1->SetXTitle("Pt␣(GeV/c)");
PtHisto1->SetYTitle("dN/dPt");
PtHisto1->SetTitleOffset(1.35,"Y");
PtHisto1->TH1::SetStats(kFALSE);

canvas1->cd();
TPad *pad2 = new TPad("pad2", "pad2", 0, 0.05, 1, 0.4);
pad2->SetTopMargin(0.025);
pad2->SetBottomMargin(0.2);
pad2->Draw();
pad2->cd();        // pad2 becomes the current pad
for(Int_t i=0; i < 27; i++)
{
double x2,y2;
graph->GetPoint(i,x2,y2);
ptylist[i]=(((PtHisto1->GetBinContent(i+1))/y2)-1)*100;
}
gr = new TGraph(27,ptxlist,ptylist);
gr->Draw("");
gr->SetTitle("");
gr->GetXaxis()->SetTitle("Pt␣(GeV/c)");
gr->GetXaxis()->SetTitleSize(0.06f);
gr->GetXaxis()->SetLabelSize(0.06);
gr->GetXaxis()->SetRangeUser(0.15,1.995);
gr->GetYaxis()->SetTitle("Model␣/␣CERN␣(%)");
gr->GetYaxis()->SetTitleOffset(0.8);
gr->GetYaxis()->SetTitleSize(0.06f);
gr->GetYaxis()->SetLabelSize(0.06);

TCanvas* canvas2 = new TCanvas("2","Pt␣distirbution␣at␣10-20%␣
    centrality",600,600);
TPad *pad1 = new TPad("pad1", "pad1", 0, 0.4, 1, 1.0);
pad1->SetBottomMargin(0.085); // Upper and lower plot are joined
pad1->Draw();              // Draw the upper pad: pad1
pad1->cd();
```

```
graph = GetSpectra(2);
sum = 0;
sum2 = 0;
for(Int_t i=0; i < 27; i++)
{
 double x,y;
 graph->GetPoint(i,x,y);
 sum += 2*PI*x*y;
 graph->SetPoint(i,x,2*PI*y*x);
}
for (Int_t c=1; c<28; c++)
{
binContent = PtHisto2->GetBinContent(c);
if (c > 17)
{
binContent = binContent / 2;
}
PtHisto2->SetBinContent(c,binContent/Bin[1]);
}
PtHisto2->Draw("EP0");
graph->Draw("same");
PtHisto2->SetXTitle("Pt␣(GeV/c)");
PtHisto2->SetYTitle("dN/dPt");
PtHisto2->SetTitleOffset(1.35,"Y");
PtHisto2->TH1::SetStats(kFALSE);

canvas2->cd();
TPad *pad2 = new TPad("pad2", "pad2", 0, 0.05, 1, 0.4);
pad2->SetTopMargin(0.025);
pad2->SetBottomMargin(0.2);
pad2->Draw();
pad2->cd();       // pad2 becomes the current pad
for(Int_t i=0; i < 27; i++)
{
double x2,y2;
graph->GetPoint(i,x2,y2);
ptylist[i]=(((PtHisto2->GetBinContent(i+1))/y2)-1)*100;
}
gr = new TGraph(27,ptxlist,ptylist);
gr->Draw("");
gr->SetTitle("");
gr->GetXaxis()->SetTitle("Pt␣(GeV/c)");
gr->GetXaxis()->SetTitleSize(0.06f);
gr->GetXaxis()->SetLabelSize(0.06);
gr->GetXaxis()->SetRangeUser(0.15,1.995);
gr->GetYaxis()->SetTitle("Model␣/␣CERN␣(%)");
gr->GetYaxis()->SetTitleOffset(0.8);
gr->GetYaxis()->SetTitleSize(0.06f);
gr->GetYaxis()->SetLabelSize(0.06);

TCanvas* canvas3 = new TCanvas("3","Pt␣distirbution␣at␣20-30%␣
    centrality",600,600);
TPad *pad1 = new TPad("pad1", "pad1", 0, 0.4, 1, 1.0);
```

```
pad1->SetBottomMargin(0.085); // Upper and lower plot are joined
pad1->Draw();              // Draw the upper pad: pad1
pad1->cd();
graph = GetSpectra(3);
sum = 0;
sum2 = 0;
for(Int_t i=0; i < 27; i++)
{
 double x,y;
 graph->GetPoint(i,x,y);
 sum += 2*PI*x*y;
 graph->SetPoint(i,x,2*PI*y*x);
}
for (Int_t c=1; c<28; c++)
{
binContent = PtHisto3->GetBinContent(c);
if (c > 17)
{
binContent = binContent / 2;
}
PtHisto3->SetBinContent(c,binContent/Bin[2]);
}
PtHisto3->Draw("EP0");
graph->Draw("same");
PtHisto3->SetXTitle("Pt␣(GeV/c)");
PtHisto3->SetYTitle("dN/dPt");
PtHisto3->SetTitleOffset(1.35,"Y");
PtHisto3->TH1::SetStats(kFALSE);

canvas3->cd();
TPad *pad2 = new TPad("pad2", "pad2", 0, 0.05, 1, 0.4);
pad2->SetTopMargin(0.025);
pad2->SetBottomMargin(0.2);
pad2->Draw();
pad2->cd();       // pad2 becomes the current pad
for(Int_t i=0; i < 27; i++)
{
double x2,y2;
graph->GetPoint(i,x2,y2);
ptylist[i]=(((PtHisto3->GetBinContent(i+1))/y2)-1)*100;
}
gr = new TGraph(27,ptxlist,ptylist);
gr->Draw("");
gr->SetTitle("");
gr->GetXaxis()->SetTitle("Pt␣(GeV/c)");
gr->GetXaxis()->SetTitleSize(0.06f);
gr->GetXaxis()->SetLabelSize(0.06);
gr->GetXaxis()->SetRangeUser(0.15,1.995);
gr->GetYaxis()->SetTitle("Model␣/␣CERN␣(%)");
gr->GetYaxis()->SetTitleOffset(0.8);
gr->GetYaxis()->SetTitleSize(0.06f);
gr->GetYaxis()->SetLabelSize(0.06);
```

```
TCanvas* canvas4 = new TCanvas("4","Pt␣distirbution␣at␣30-40%␣
    centrality",600,600);
TPad *pad1 = new TPad("pad1", "pad1", 0, 0.4, 1, 1.0);
pad1->SetBottomMargin(0.085); // Upper and lower plot are joined
pad1->Draw();              // Draw the upper pad: pad1
pad1->cd();
graph = GetSpectra(4);
sum = 0;
sum2 = 0;
for(Int_t i=0; i < 27; i++)
{
 double x,y;
 graph->GetPoint(i,x,y);
 sum += 2*PI*x*y;
 graph->SetPoint(i,x,2*PI*y*x);
}
for (Int_t c=1; c<28; c++)
{
binContent = PtHisto4->GetBinContent(c);
if (c > 17)
{
binContent = binContent / 2;
}
PtHisto4->SetBinContent(c,binContent/Bin[3]);
}
PtHisto4->Draw("EP0");
graph->Draw("same");
PtHisto4->SetXTitle("Pt␣(GeV/c)");
PtHisto4->SetYTitle("dN/dPt");
PtHisto4->SetTitleOffset(1.35,"Y");
PtHisto4->TH1::SetStats(kFALSE);

canvas4->cd();
TPad *pad2 = new TPad("pad2", "pad2", 0, 0.05, 1, 0.4);
pad2->SetTopMargin(0.025);
pad2->SetBottomMargin(0.2);
pad2->Draw();
pad2->cd();       // pad2 becomes the current pad
for(Int_t i=0; i < 27; i++)
{
double x2,y2;
graph->GetPoint(i,x2,y2);
ptylist[i]=(((PtHisto4->GetBinContent(i+1))/y2)-1)*100;
}
gr = new TGraph(27,ptxlist,ptylist);
gr->Draw("");
gr->SetTitle("");
gr->GetXaxis()->SetTitle("Pt␣(GeV/c)");
gr->GetXaxis()->SetTitleSize(0.06f);
gr->GetXaxis()->SetLabelSize(0.06);
gr->GetXaxis()->SetRangeUser(0.15,1.995);
gr->GetYaxis()->SetTitle("Model␣/␣CERN␣(%)");
gr->GetYaxis()->SetTitleOffset(0.8);
```

```
gr->GetYaxis()->SetTitleSize(0.06f);
gr->GetYaxis()->SetLabelSize(0.06);

TCanvas* canvas5 = new TCanvas("5","Pt␣distirbution␣at␣40-50%␣
    centrality",600,600);
TPad *pad1 = new TPad("pad1", "pad1", 0, 0.4, 1, 1.0);
pad1->SetBottomMargin(0.085); // Upper and lower plot are joined
pad1->Draw();              // Draw the upper pad: pad1
pad1->cd();
graph = GetSpectra(5);
sum = 0;
sum2 = 0;
for(Int_t i=0; i < 27; i++)
{
 double x,y;
 graph->GetPoint(i,x,y);
 sum += 2*PI*x*y;
 graph->SetPoint(i,x,2*PI*y*x);
}
for (Int_t c=1; c<28; c++)
{
binContent = PtHisto5->GetBinContent(c);
if (c > 17)
{
binContent = binContent / 2;
}
PtHisto5->SetBinContent(c,binContent/Bin[4]);
}
PtHisto5->Draw("EP0");
graph->Draw("same");
PtHisto5->SetXTitle("Pt␣(GeV/c)");
PtHisto5->SetYTitle("dN/dPt");
PtHisto5->SetTitleOffset(1.35,"Y");
PtHisto5->TH1::SetStats(kFALSE);

canvas5->cd();
TPad *pad2 = new TPad("pad2", "pad2", 0, 0.05, 1, 0.4);
pad2->SetTopMargin(0.025);
pad2->SetBottomMargin(0.2);
pad2->Draw();
pad2->cd();       // pad2 becomes the current pad
for(Int_t i=0; i < 27; i++)
{
double x2,y2;
graph->GetPoint(i,x2,y2);
ptylist[i]=(((PtHisto5->GetBinContent(i+1))/y2)-1)*100;
}
gr = new TGraph(27,ptxlist,ptylist);
gr->Draw("");
gr->SetTitle("");
gr->GetXaxis()->SetTitle("Pt␣(GeV/c)");
gr->GetXaxis()->SetTitleSize(0.06f);
gr->GetXaxis()->SetLabelSize(0.06);
```

```
gr->GetXaxis()->SetRangeUser(0.15,1.995);
gr->GetYaxis()->SetTitle("Model␣/␣CERN␣(%)");
gr->GetYaxis()->SetTitleOffset(0.8);
gr->GetYaxis()->SetTitleSize(0.06f);
gr->GetYaxis()->SetLabelSize(0.06);

double ptxv2list[18] = {0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85,
    0.95, 1.05, 1.15,
     1.25, 1.35, 1.45, 1.55, 1.65, 1.75, 1.85, 1.95};
double ptyv2list[18];

TCanvas* canvas11 = new TCanvas("11","V2␣vs␣Pt␣at␣5-10%␣centrality"
    ,600,600);
TPad *pad1 = new TPad("pad1", "pad1", 0, 0.4, 1, 1.0);
pad1->SetBottomMargin(0.085); // Upper and lower plot are joined
pad1->Draw();              // Draw the upper pad: pad1
pad1->cd();
hvnpt1->Draw("");
hvnpt1->SetXTitle("Pt␣(GeV/c)");
hvnpt1->SetYTitle("V2");
hvnpt1->SetTitleOffset(1.35,"Y");
hvnpt1->TH1::SetStats(kFALSE);
graph = GetV2_QC4(1);
graph->Draw("same");

canvas11->cd();
TPad *pad2 = new TPad("pad2", "pad2", 0, 0.05, 1, 0.4);
pad2->SetTopMargin(0.025);
pad2->SetBottomMargin(0.2);
pad2->Draw();
pad2->cd();       // pad2 becomes the current pad
for(Int_t i=0; i < 18; i++)
{
double x2,y2;
graph->GetPoint(i,x2,y2);
ptyv2list[i]=(((hvnpt1->GetBinContent(i+3))/y2)-1)*100;
}
gr = new TGraph(18,ptxv2list,ptyv2list);
gr->Draw("");
gr->SetTitle("");
gr->GetXaxis()->SetTitle("Pt␣(GeV/c)");
gr->GetXaxis()->SetTitleSize(0.06f);
gr->GetXaxis()->SetLabelSize(0.06);
gr->GetXaxis()->SetRangeUser(0.,1.995);
gr->GetYaxis()->SetTitle("Model␣/␣CERN␣(%)");
gr->GetYaxis()->SetTitleOffset(0.8);
gr->GetYaxis()->SetTitleSize(0.06f);
gr->GetYaxis()->SetLabelSize(0.06);

TCanvas* canvas21 = new TCanvas("21","V2␣vs␣Pt␣at␣10-20%␣centrality"
    ,600,600);
TPad *pad1 = new TPad("pad1", "pad1", 0, 0.4, 1, 1.0);
pad1->SetBottomMargin(0.085); // Upper and lower plot are joined
```

```
pad1->Draw();              // Draw the upper pad: pad1
pad1->cd();
hvnpt2->Draw("");
hvnpt2->SetXTitle("Pt␣(GeV/c)");
hvnpt2->SetYTitle("V2");
hvnpt2->SetTitleOffset(1.35,"Y");
hvnpt2->TH1::SetStats(kFALSE);
graph = GetV2_QC4(2);
graph->Draw("same");

canvas21->cd();
TPad *pad2 = new TPad("pad2", "pad2", 0, 0.05, 1, 0.4);
pad2->SetTopMargin(0.025);
pad2->SetBottomMargin(0.2);
pad2->Draw();
pad2->cd();        // pad2 becomes the current pad
for(Int_t i=0; i < 18; i++)
{
double x2,y2;
graph->GetPoint(i,x2,y2);
ptyv2list[i]=(((hvnpt2->GetBinContent(i+3))/y2)-1)*100;
}
gr = new TGraph(18,ptxv2list,ptyv2list);
gr->Draw("");
gr->SetTitle("");
gr->GetXaxis()->SetTitle("Pt␣(GeV/c)");
gr->GetXaxis()->SetTitleSize(0.06f);
gr->GetXaxis()->SetLabelSize(0.06);
gr->GetXaxis()->SetRangeUser(0.,1.995);
gr->GetYaxis()->SetTitle("Model␣/␣CERN␣(%)");
gr->GetYaxis()->SetTitleOffset(0.8);
gr->GetYaxis()->SetTitleSize(0.06f);
gr->GetYaxis()->SetLabelSize(0.06);

TCanvas* canvas31 = new TCanvas("31","V2␣vs␣Pt␣at␣20-30%␣centrality"
    ,600,600);
TPad *pad1 = new TPad("pad1", "pad1", 0, 0.4, 1, 1.0);
pad1->SetBottomMargin(0.085); // Upper and lower plot are joined
pad1->Draw();              // Draw the upper pad: pad1
pad1->cd();
hvnpt3->Draw("");
hvnpt3->SetXTitle("Pt␣(GeV/c)");
hvnpt3->SetYTitle("V2");
hvnpt3->SetTitleOffset(1.35,"Y");
hvnpt3->TH1::SetStats(kFALSE);
graph = GetV2_QC4(3);
graph->Draw("same");

canvas31->cd();
TPad *pad2 = new TPad("pad2", "pad2", 0, 0.05, 1, 0.4);
pad2->SetTopMargin(0.025);
pad2->SetBottomMargin(0.2);
pad2->Draw();
```

```
pad2->cd();        // pad2 becomes the current pad
for(Int_t i=0; i < 18; i++)
{
double x2,y2;
graph->GetPoint(i,x2,y2);
ptyv2list[i]=(((hvnpt3->GetBinContent(i+3))/y2)-1)*100;
}
gr = new TGraph(18,ptxv2list,ptyv2list);
gr->Draw("");
gr->SetTitle("");
gr->GetXaxis()->SetTitle("Pt␣(GeV/c)");
gr->GetXaxis()->SetTitleSize(0.06f);
gr->GetXaxis()->SetLabelSize(0.06);
gr->GetXaxis()->SetRangeUser(0.,1.995);
gr->GetYaxis()->SetTitle("Model␣/␣CERN␣(%)");
gr->GetYaxis()->SetTitleOffset(0.8);
gr->GetYaxis()->SetTitleSize(0.06f);
gr->GetYaxis()->SetLabelSize(0.06);

TCanvas* canvas41 = new TCanvas("41","V2␣vs␣Pt␣at␣30-40%␣centrality"
    ,600,600);
TPad *pad1 = new TPad("pad1", "pad1", 0, 0.4, 1, 1.0);
pad1->SetBottomMargin(0.085); // Upper and lower plot are joined
pad1->Draw();              // Draw the upper pad: pad1
pad1->cd();
hvnpt4->Draw("");
hvnpt4->SetXTitle("Pt␣(GeV/c)");
hvnpt4->SetYTitle("V2");
hvnpt4->SetTitleOffset(1.35,"Y");
hvnpt4->TH1::SetStats(kFALSE);
graph = GetV2_QC4(4);
graph->Draw("same");

canvas41->cd();
TPad *pad2 = new TPad("pad2", "pad2", 0, 0.05, 1, 0.4);
pad2->SetTopMargin(0.025);
pad2->SetBottomMargin(0.2);
pad2->Draw();
pad2->cd();        // pad2 becomes the current pad
for(Int_t i=0; i < 18; i++)
{
double x2,y2;
graph->GetPoint(i,x2,y2);
ptyv2list[i]=(((hvnpt4->GetBinContent(i+3))/y2)-1)*100;
}
gr = new TGraph(18,ptxv2list,ptyv2list);
gr->Draw("");
gr->SetTitle("");
gr->GetXaxis()->SetTitle("Pt␣(GeV/c)");
gr->GetXaxis()->SetTitleSize(0.06f);
gr->GetXaxis()->SetLabelSize(0.06);
gr->GetXaxis()->SetRangeUser(0.,1.995);
gr->GetYaxis()->SetTitle("Model␣/␣CERN␣(%)");
```

```
gr->GetYaxis()->SetTitleOffset(0.8);
gr->GetYaxis()->SetTitleSize(0.06f);
gr->GetYaxis()->SetLabelSize(0.06);

TCanvas* canvas51 = new TCanvas("51","V2␣vs␣Pt␣at␣40-50%␣centrality"
    ,600,600);
TPad *pad1 = new TPad("pad1", "pad1", 0, 0.4, 1, 1.0);
pad1->SetBottomMargin(0.085); // Upper and lower plot are joined
pad1->Draw();              // Draw the upper pad: pad1
pad1->cd();
hvnpt5->Draw("");
hvnpt5->SetXTitle("Pt␣(GeV/c)");
hvnpt5->SetYTitle("V2");
hvnpt5->SetTitleOffset(1.35,"Y");
hvnpt5->TH1::SetStats(kFALSE);
graph = GetV2_QC4(5);
graph->Draw("same");

canvas51->cd();
TPad *pad2 = new TPad("pad2", "pad2", 0, 0.05, 1, 0.4);
pad2->SetTopMargin(0.025);
pad2->SetBottomMargin(0.2);
pad2->Draw();
pad2->cd();       // pad2 becomes the current pad
for(Int_t i=0; i < 18; i++)
{
double x2,y2;
graph->GetPoint(i,x2,y2);
ptyv2list[i]=(((hvnpt5->GetBinContent(i+3))/y2)-1)*100;
}
gr = new TGraph(18,ptxv2list,ptyv2list);
gr->Draw("");
gr->SetTitle("");
gr->GetXaxis()->SetTitle("Pt␣(GeV/c)");
gr->GetXaxis()->SetTitleSize(0.06f);
gr->GetXaxis()->SetLabelSize(0.06);
gr->GetXaxis()->SetRangeUser(0.,1.995);
gr->GetYaxis()->SetTitle("Model␣/␣CERN␣(%)");
gr->GetYaxis()->SetTitleOffset(0.8);
gr->GetYaxis()->SetTitleSize(0.06f);
gr->GetYaxis()->SetLabelSize(0.06);

 hfile->Write();

}

TGraphAsymmErrors* GetSpectra(Int_t cen=0)
{
 TGraphAsymmErrors* spec;
 switch (cen) {
  case 0:
   // Centrality 0-5 %
   double p8210_d1x1y1_xval[] = { 0.175, 0.225, 0.275, 0.325, 0.375,
```

```
    0.425, 0.475, 0.525, 0.575,
 0.625, 0.675, 0.725, 0.775, 0.825, 0.875, 0.925, 0.975, 1.05,
     1.15,
 1.25, 1.35, 1.45, 1.55, 1.65, 1.75, 1.85, 1.95, 2.1, 2.3,
 2.5, 2.7, 2.9, 3.1, 3.3, 3.5, 3.7, 3.9, 4.25, 4.75,
 5.25, 5.75, 6.25, 6.75, 7.5, 8.5, 9.5, 10.5, 11.5, 12.5,
 13.5, 14.5, 15.5, 17.0, 19.0, 21.0, 23.0, 25.0, 27.0, 29.0,
 31.0, 33.0, 35.0, 38.0, 42.5, 47.5 };
double p8210_d1x1y1_xerrminus[] = { 0.024999999999999994,
    0.024999999999999994, 0.025000000000000022,
    0.025000000000000022, 0.025000000000000022,
    0.024999999999999967, 0.024999999999999967,
    0.025000000000000022, 0.02499999999999991,
 0.025000000000000022, 0.025000000000000022, 0.025000000000000022,
     0.025000000000000022, 0.02499999999999991,
     0.025000000000000022, 0.025000000000000022,
     0.025000000000000022, 0.050000000000000044,
     0.04999999999999982,
 0.050000000000000044, 0.050000000000000044, 0.050000000000000044,
     0.050000000000000044, 0.04999999999999982,
     0.050000000000000044, 0.050000000000000044,
     0.050000000000000044, 0.10000000000000009, 0.09999999999999964,
 0.10000000000000009, 0.10000000000000009, 0.10000000000000009,
     0.10000000000000009, 0.09999999999999964, 0.10000000000000009,
     0.10000000000000009, 0.10000000000000009, 0.25, 0.25,
 0.25, 0.25, 0.25, 0.25, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
 0.5, 0.5, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
 1.0, 1.0, 1.0, 2.0, 2.5, 2.5 };
double p8210_d1x1y1_xerrplus[] = { 0.025000000000000022,
    0.024999999999999994, 0.024999999999999967,
    0.024999999999999967, 0.025000000000000022,
    0.025000000000000022, 0.025000000000000022,
    0.025000000000000022, 0.025000000000000022,
 0.025000000000000022, 0.02499999999999991, 0.025000000000000022,
     0.025000000000000022, 0.025000000000000022,
     0.025000000000000022, 0.02499999999999991,
     0.025000000000000022, 0.050000000000000044,
     0.050000000000000044,
 0.050000000000000044, 0.04999999999999982, 0.050000000000000044,
     0.050000000000000044, 0.050000000000000044,
     0.050000000000000044, 0.04999999999999982,
     0.050000000000000044, 0.10000000000000009, 0.10000000000000009,
 0.10000000000000009, 0.09999999999999964, 0.10000000000000009,
     0.10000000000000009, 0.10000000000000009, 0.10000000000000009,
     0.09999999999999964, 0.10000000000000009, 0.25, 0.25,
 0.25, 0.25, 0.25, 0.25, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
 0.5, 0.5, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
 1.0, 1.0, 1.0, 2.0, 2.5, 2.5 };
double p8210_d1x1y1_yval[] = { 1810.0, 1509.0, 1226.0, 985.5,
    791.4, 644.1, 526.6, 433.1, 358.5,
 298.5, 250.2, 210.9, 178.4, 151.4, 129.3, 110.9, 94.85, 75.5,
     56.54,
 42.52, 31.96, 24.34, 18.52, 14.29, 11.03, 8.641, 6.72, 4.683,
```

```
    2.865,
 1.774, 1.107, 0.6974, 0.4408, 0.2836, 0.1851, 0.1222, 0.08176,
    0.0425, 0.01805,
 0.008502, 0.004461, 0.002534, 0.001543, 8.184E-4, 3.937E-4, 2.071E
    -4, 1.145E-4, 6.889E-5, 4.269E-5,
 2.767E-5, 1.833E-5, 1.298E-5, 7.735E-6, 4.401E-6, 2.236E-6, 1.386E
    -6, 9.194E-7, 5.54E-7, 3.821E-7,
 2.27E-7, 1.879E-7, 1.143E-7, 7.511E-8, 3.39E-8, 1.731E-8 };
double p8210_d1x1y1_yerrminus[] = { 243.01851781294363,
   171.00292395160966, 121.00413216084813, 89.30358335475682,
   67.60362416320592, 54.00333323045903, 43.80285378830927,
   35.802234567132814, 29.602702579325424,
 24.501836665850174, 20.50219500443794, 17.202616080119906,
    14.50137924474772, 12.301625908797584, 10.501904589168577,
    9.00222194794152, 7.7018763947495295, 6.1208169389387885,
    4.590882268148465,
 3.470518693221519, 2.6103064954139006, 2.000399960007998,
    1.570286598045083, 1.2103718436910205, 0.9302150289046076,
    0.7292742968184193, 0.5672257046361703, 0.3930623360231809,
    0.24003333101883997,
 0.1480304022827743, 0.09202173656261872, 0.05781245540538821,
    0.03642320688791694, 0.023413671220037235,
    0.015211837495845136, 0.010007996802557442,
    0.006665470726062789, 0.0034537081521170835,
    0.0014616771189288008,
 6.900470998417428E-4, 3.6272992708074144E-4, 2.0554804791094464E
    -4, 1.2548306658669128E-4, 6.651150276455945E-5,
    3.206009981269553E-5, 1.6930741271426954E-5, 9.51892851112981E
    -6, 5.752295194094266E-6, 3.6119939091864486E-6,
 2.3724459951703854E-6, 1.6065490966665162E-6, 1.161077086157504E
    -6, 6.859774048757E-7, 4.027828198918122E-7, 2.160208323287363E
    -7, 1.4406248644251563E-7, 1.0400807660946337E-7,
    6.796212180325155E-8, 5.02864792961289E-8,
 3.3853360246805636E-8, 2.9173275441746337E-8, 2.04707107839469E-8,
     1.1987635296421058E-8, 6.315473062249573E-9, 3.998812323678119
    E-9 };
double p8210_d1x1y1_yerrplus[] = { 243.01851781294363,
   171.00292395160966, 121.00413216084813, 89.30358335475682,
   67.60362416320592, 54.00333323045903, 43.80285378830927,
   35.802234567132814, 29.602702579325424,
 24.501836665850174, 20.50219500443794, 17.202616080119906,
    14.50137924474772, 12.301625908797584, 10.501904589168577,
    9.00222194794152, 7.7018763947495295, 6.1208169389387885,
    4.590882268148465,
 3.470518693221519, 2.6103064954139006, 2.000399960007998,
    1.570286598045083, 1.2103718436910205, 0.9302150289046076,
    0.7292742968184193, 0.5672257046361703, 0.3930623360231809,
    0.24003333101883997,
 0.1480304022827743, 0.09202173656261872, 0.05781245540538821,
    0.03642320688791694, 0.023413671220037235,
    0.015211837495845136, 0.010007996802557442,
    0.006665470726062789, 0.0034537081521170835,
    0.0014616771189288008,
```

```
  6.900470998417428E-4, 3.6272992708074144E-4, 2.0554804791094464E
      -4, 1.2548306658669128E-4, 6.651150276455945E-5,
      3.206009981269553E-5, 1.6930741271426954E-5, 9.51892851112981E
      -6, 5.752295194094266E-6, 3.6119939091864486E-6,
  2.3724459951703854E-6, 1.6065490966665162E-6, 1.161077086157504E
      -6, 6.859774048757E-7, 4.027828198918122E-7, 2.160208323287363E
      -7, 1.4406248644251563E-7, 1.0400807660946337E-7,
      6.796212180325155E-8, 5.02864792961289E-8,
  3.3853360246805636E-8, 2.9173275441746337E-8, 2.04707107839469E-8,
       1.1987635296421058E-8, 6.315473062249573E-9, 3.998812323678119
      E-9 };
 double p8210_d1x1y1_ystatminus[] = { 3.0, 1.0, 1.0, 0.8, 0.7, 0.6,
     0.5, 0.4, 0.4,
  0.3, 0.3, 0.3, 0.2, 0.2, 0.2, 0.2, 0.17, 0.1, 0.09,
  0.06, 0.04, 0.04, 0.03, 0.03, 0.02, 0.02, 0.016, 0.007, 0.004,
  0.003, 0.002, 0.0012, 0.0013, 8.0E-4, 6.0E-4, 4.0E-4, 2.7E-4, 1.6E
      -4, 7.0E-5,
  3.8E-5, 2.3E-5, 1.5E-5, 1.1E-5, 5.3E-6, 3.2E-6, 2.1E-6, 1.5E-6,
      1.08E-6, 8.1E-7,
  6.2E-7, 4.9E-7, 4.0E-7, 2.07E-7, 1.47E-7, 9.9E-8, 7.5E-8, 5.82E-8,
       4.34E-8, 3.48E-8,
  2.59E-8, 2.28E-8, 1.72E-8, 9.47E-9, 5.36E-9, 3.61E-9 };
 double p8210_d1x1y1_ystatplus[] = { 3.0, 1.0, 1.0, 0.8, 0.7, 0.6,
     0.5, 0.4, 0.4,
  0.3, 0.3, 0.3, 0.2, 0.2, 0.2, 0.2, 0.17, 0.1, 0.09,
  0.06, 0.04, 0.04, 0.03, 0.03, 0.02, 0.02, 0.016, 0.007, 0.004,
  0.003, 0.002, 0.0012, 0.0013, 8.0E-4, 6.0E-4, 4.0E-4, 2.7E-4, 1.6E
      -4, 7.0E-5,
  3.8E-5, 2.3E-5, 1.5E-5, 1.1E-5, 5.3E-6, 3.2E-6, 2.1E-6, 1.5E-6,
      1.08E-6, 8.1E-7,
  6.2E-7, 4.9E-7, 4.0E-7, 2.07E-7, 1.47E-7, 9.9E-8, 7.5E-8, 5.82E-8,
       4.34E-8, 3.48E-8,
  2.59E-8, 2.28E-8, 1.72E-8, 9.47E-9, 5.36E-9, 3.61E-9 };
 int p8210_d1x1y1_numpoints = 65;
 spec = new TGraphAsymmErrors(p8210_d1x1y1_numpoints,
     p8210_d1x1y1_xval, p8210_d1x1y1_yval, p8210_d1x1y1_xerrminus,
     p8210_d1x1y1_xerrplus, p8210_d1x1y1_yerrminus,
     p8210_d1x1y1_yerrplus);
 spec->SetName("/HepData/8210/d1x1y1");
 spec->SetTitle("/HepData/8210/d1x1y1");
 break;

case 1:
 // Centrality 5-10 %
 double p8210_d2x1y1_xval[] = { 0.175, 0.225, 0.275, 0.325, 0.375,
     0.425, 0.475, 0.525, 0.575,
  0.625, 0.675, 0.725, 0.775, 0.825, 0.875, 0.925, 0.975, 1.05,
      1.15,
  1.25, 1.35, 1.45, 1.55, 1.65, 1.75, 1.85, 1.95, 2.1, 2.3,
  2.5, 2.7, 2.9, 3.1, 3.3, 3.5, 3.7, 3.9, 4.25, 4.75,
  5.25, 5.75, 6.25, 6.75, 7.5, 8.5, 9.5, 10.5, 11.5, 12.5,
  13.5, 14.5, 15.5, 17.0, 19.0, 21.0, 23.0, 25.0, 27.0, 29.0,
  31.0, 33.0, 35.0, 38.0, 42.5, 47.5 };
```

```
double p8210_d2x1y1_xerrminus[] = { 0.024999999999999994,
    0.024999999999999994, 0.025000000000000022,
    0.025000000000000022, 0.025000000000000022,
    0.024999999999999967, 0.024999999999999967,
    0.025000000000000022, 0.02499999999999991,
 0.025000000000000022, 0.025000000000000022, 0.025000000000000022,
     0.025000000000000022, 0.02499999999999991,
     0.025000000000000022, 0.025000000000000022,
     0.025000000000000022, 0.050000000000000044,
     0.04999999999999982,
 0.050000000000000044, 0.050000000000000044, 0.050000000000000044,
     0.050000000000000044, 0.04999999999999982,
     0.050000000000000044, 0.050000000000000044,
     0.050000000000000044, 0.10000000000000009, 0.09999999999999964,
 0.10000000000000009, 0.10000000000000009, 0.10000000000000009,
     0.10000000000000009, 0.09999999999999964, 0.10000000000000009,
     0.10000000000000009, 0.10000000000000009, 0.25, 0.25,
 0.25, 0.25, 0.25, 0.25, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
 0.5, 0.5, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
 1.0, 1.0, 1.0, 2.0, 2.5, 2.5 };
double p8210_d2x1y1_xerrplus[] = { 0.025000000000000022,
    0.024999999999999994, 0.024999999999999967,
    0.024999999999999967, 0.025000000000000022,
    0.025000000000000022, 0.025000000000000022,
    0.025000000000000022, 0.025000000000000022,
 0.025000000000000022, 0.02499999999999991, 0.025000000000000022,
     0.025000000000000022, 0.025000000000000022,
     0.025000000000000022, 0.02499999999999991,
     0.025000000000000022, 0.050000000000000044,
     0.050000000000000044,
 0.050000000000000044, 0.04999999999999982, 0.050000000000000044,
     0.050000000000000044, 0.050000000000000044,
     0.050000000000000044, 0.04999999999999982,
     0.050000000000000044, 0.10000000000000009, 0.10000000000000009,
 0.10000000000000009, 0.09999999999999964, 0.10000000000000009,
     0.10000000000000009, 0.10000000000000009, 0.10000000000000009,
     0.09999999999999964, 0.10000000000000009, 0.25, 0.25,
 0.25, 0.25, 0.25, 0.25, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
 0.5, 0.5, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
 1.0, 1.0, 1.0, 2.0, 2.5, 2.5 };
double p8210_d2x1y1_yval[] = { 1476.0, 1217.0, 985.6, 792.9, 636.9,
     518.0, 423.8, 349.3, 289.1,
 240.8, 202.2, 170.3, 143.9, 122.4, 104.6, 89.56, 76.84, 61.31,
     45.79,
 34.61, 26.05, 19.8, 15.09, 11.66, 9.001, 7.07, 5.497, 3.84, 2.36,
 1.466, 0.9206, 0.5838, 0.3726, 0.2417, 0.1589, 0.1055, 0.07158,
     0.03753, 0.01618,
 0.007753, 0.004119, 0.002319, 0.001427, 7.513E-4, 3.564E-4, 1.891E
     -4, 1.048E-4, 6.167E-5, 3.843E-5,
 2.402E-5, 1.664E-5, 1.046E-5, 7.069E-6, 3.517E-6, 1.998E-6, 1.194E
     -6, 6.929E-7, 4.235E-7, 3.399E-7,
 2.203E-7, 1.073E-7, 1.01E-7, 5.817E-8, 3.036E-8, 2.02E-8 };
double p8210_d2x1y1_yerrminus[] = { 198.01010075246163,
```

```
    138.00362314084367, 97.60327863345574, 72.00249995659873,
    54.602289329294614, 43.60183482377777, 35.302266216207705,
    29.001551682625536, 23.901882771028728,
 19.901004999748128, 16.601204775557708, 13.901438774457844,
     11.801694793545543, 10.00199980003999, 8.50235261559999,
     7.291344183345071, 6.251351853799305, 4.980642528830995,
     3.720658543860213,
 2.830282671395209, 2.1402102700435766, 1.6302760502442524,
     1.280156240464421, 0.9902019995940222, 0.7621896089556719,
     0.599213651379873, 0.46518168493611184, 0.3240555507933786,
     0.19804039991880446,
 0.12301625908797585, 0.07681464704078252, 0.04861028697714096,
     0.030919573088902766, 0.020012246250733574,
     0.013109538512091111, 0.008705170877128144,
     0.005854921007152872, 0.003053211424058282,
     0.001321854757528224,
 6.31969935993794E-4, 3.356575635971876E-4, 1.8951780918953238E-4,
     1.1643023662262307E-4, 6.130424128883743E-5, 2.9154759474226502
     E-5, 1.552932709424333E-5, 8.713208364316787E-6,
     5.171585830284556E-6, 3.2718954751030786E-6,
 2.082402458700047E-6, 1.4641379716406511E-6, 9.470480452437459E-7,
      6.324634376784163E-7, 3.28271229321121E-7, 1.9601020381602587E
     -7, 1.273184982632139E-7, 8.239089755549457E-8,
     5.5103629644516163E-8, 4.6033140236138576E-8,
 3.308473968463406E-8, 2.009975124224178E-8, 1.8985520798756087E-8,
      1.0082668297628361E-8, 5.882482469162148E-9, 4.378607084450487
     E-9 };
double p8210_d2x1y1_yerrplus[] = { 198.01010075246163,
    138.00362314084367, 97.60327863345574, 72.00249995659873,
    54.602289329294614, 43.60183482377777, 35.302266216207705,
    29.001551682625536, 23.901882771028728,
 19.901004999748128, 16.601204775557708, 13.901438774457844,
     11.801694793545543, 10.00199980003999, 8.50235261559999,
     7.291344183345071, 6.251351853799305, 4.980642528830995,
     3.720658543860213,
 2.830282671395209, 2.1402102700435766, 1.6302760502442524,
     1.280156240464421, 0.9902019995940222, 0.7621896089556719,
     0.599213651379873, 0.46518168493611184, 0.3240555507933786,
     0.19804039991880446,
 0.12301625908797585, 0.07681464704078252, 0.04861028697714096,
     0.030919573088902766, 0.020012246250733574,
     0.013109538512091111, 0.008705170877128144,
     0.005854921007152872, 0.003053211424058282,
     0.001321854757528224,
 6.31969935993794E-4, 3.356575635971876E-4, 1.8951780918953238E-4,
     1.1643023662262307E-4, 6.130424128883743E-5, 2.9154759474226502
     E-5, 1.552932709424333E-5, 8.713208364316787E-6,
     5.171585830284556E-6, 3.2718954751030786E-6,
 2.082402458700047E-6, 1.4641379716406511E-6, 9.470480452437459E-7,
      6.324634376784163E-7, 3.28271229321121E-7, 1.9601020381602587E
     -7, 1.273184982632139E-7, 8.239089755549457E-8,
     5.5103629644516163E-8, 4.6033140236138576E-8,
 3.308473968463406E-8, 2.009975124224178E-8, 1.8985520798756087E-8,
```

```
      1.0082668297628361E-8, 5.882482469162148E-9, 4.378607084450487
     E-9 };
 double p8210_d2x1y1_ystatminus[] = { 2.0, 1.0, 0.8, 0.6, 0.5, 0.4,
    0.4, 0.3, 0.3,
  0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.14, 0.13, 0.08, 0.07,
  0.04, 0.03, 0.03, 0.02, 0.02, 0.017, 0.016, 0.013, 0.006, 0.004,
  0.002, 0.0015, 0.001, 0.0011, 7.0E-4, 5.0E-4, 3.0E-4, 2.4E-4, 1.4E
     -4, 7.0E-5,
  3.5E-5, 2.1E-5, 1.4E-5, 1.0E-5, 5.0E-6, 3.0E-6, 2.0E-6, 1.4E-6,
     1.02E-6, 7.7E-7,
  5.8E-7, 4.6E-7, 3.5E-7, 1.97E-7, 1.31E-7, 9.4E-8, 6.9E-8, 5.05E-8,
      3.79E-8, 3.27E-8,
  2.54E-8, 1.72E-8, 1.62E-8, 8.31E-9, 5.06E-9, 3.89E-9 };
 double p8210_d2x1y1_ystatplus[] = { 2.0, 1.0, 0.8, 0.6, 0.5, 0.4,
    0.4, 0.3, 0.3,
  0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.14, 0.13, 0.08, 0.07,
  0.04, 0.03, 0.03, 0.02, 0.02, 0.017, 0.016, 0.013, 0.006, 0.004,
  0.002, 0.0015, 0.001, 0.0011, 7.0E-4, 5.0E-4, 3.0E-4, 2.4E-4, 1.4E
     -4, 7.0E-5,
  3.5E-5, 2.1E-5, 1.4E-5, 1.0E-5, 5.0E-6, 3.0E-6, 2.0E-6, 1.4E-6,
     1.02E-6, 7.7E-7,
  5.8E-7, 4.6E-7, 3.5E-7, 1.97E-7, 1.31E-7, 9.4E-8, 6.9E-8, 5.05E-8,
      3.79E-8, 3.27E-8,
  2.54E-8, 1.72E-8, 1.62E-8, 8.31E-9, 5.06E-9, 3.89E-9 };
 int p8210_d2x1y1_numpoints = 65;
 spec = new TGraphAsymmErrors(p8210_d2x1y1_numpoints,
    p8210_d2x1y1_xval, p8210_d2x1y1_yval, p8210_d2x1y1_xerrminus,
    p8210_d2x1y1_xerrplus, p8210_d2x1y1_yerrminus,
    p8210_d2x1y1_yerrplus);
 spec->SetName("/HepData/8210/d2x1y1");
 spec->SetTitle("/HepData/8210/d2x1y1");
 break;

case 2:
 // Centrality 10-20 %
 double p8210_d3x1y1_xval[] = { 0.175, 0.225, 0.275, 0.325, 0.375,
    0.425, 0.475, 0.525, 0.575,
  0.625, 0.675, 0.725, 0.775, 0.825, 0.875, 0.925, 0.975, 1.05,
     1.15,
  1.25, 1.35, 1.45, 1.55, 1.65, 1.75, 1.85, 1.95, 2.1, 2.3,
  2.5, 2.7, 2.9, 3.1, 3.3, 3.5, 3.7, 3.9, 4.25, 4.75,
  5.25, 5.75, 6.25, 6.75, 7.5, 8.5, 9.5, 10.5, 11.5, 12.5,
  13.5, 14.5, 15.5, 17.0, 19.0, 21.0, 23.0, 25.0, 27.0, 29.0,
  31.0, 33.0, 35.0, 38.0, 42.5, 47.5 };
 double p8210_d3x1y1_xerrminus[] = { 0.024999999999999994,
    0.024999999999999994, 0.025000000000000022,
    0.025000000000000022, 0.025000000000000022,
    0.024999999999999967, 0.024999999999999967,
    0.025000000000000022, 0.02499999999999991,
  0.025000000000000022, 0.025000000000000022, 0.025000000000000022,
     0.025000000000000022, 0.02499999999999991,
     0.025000000000000022, 0.025000000000000022,
     0.025000000000000022, 0.050000000000000044,
```

```
    0.04999999999999982,
 0.050000000000000044, 0.050000000000000044, 0.050000000000000044,
    0.050000000000000044, 0.04999999999999982,
    0.050000000000000044, 0.050000000000000044,
    0.050000000000000044, 0.10000000000000009, 0.09999999999999964,
 0.10000000000000009, 0.10000000000000009, 0.10000000000000009,
    0.10000000000000009, 0.09999999999999964, 0.10000000000000009,
    0.10000000000000009, 0.10000000000000009, 0.25, 0.25,
 0.25, 0.25, 0.25, 0.25, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
 0.5, 0.5, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
 1.0, 1.0, 1.0, 2.0, 2.5, 2.5 };
double p8210_d3x1y1_xerrplus[] = { 0.025000000000000022,
   0.024999999999999994, 0.024999999999999967,
   0.024999999999999967, 0.025000000000000022,
   0.025000000000000022, 0.025000000000000022,
   0.025000000000000022, 0.025000000000000022,
 0.025000000000000022, 0.02499999999999991, 0.025000000000000022,
    0.025000000000000022, 0.025000000000000022,
    0.025000000000000022, 0.02499999999999991,
    0.025000000000000022, 0.050000000000000044,
    0.050000000000000044,
 0.050000000000000044, 0.04999999999999982, 0.050000000000000044,
    0.050000000000000044, 0.050000000000000044,
    0.050000000000000044, 0.04999999999999982,
    0.050000000000000044, 0.10000000000000009, 0.10000000000000009,
 0.10000000000000009, 0.09999999999999964, 0.10000000000000009,
    0.10000000000000009, 0.10000000000000009, 0.10000000000000009,
    0.09999999999999964, 0.10000000000000009, 0.25, 0.25,
 0.25, 0.25, 0.25, 0.25, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
 0.5, 0.5, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
 1.0, 1.0, 1.0, 2.0, 2.5, 2.5 };
double p8210_d3x1y1_yval[] = { 1115.0, 911.5, 736.6, 592.0, 475.0,
   386.4, 315.0, 259.3, 215.0,
 179.4, 150.2, 126.5, 107.0, 90.89, 77.73, 66.73, 57.11, 45.6,
    34.1,
 25.8, 19.43, 14.75, 11.25, 8.69, 6.718, 5.288, 4.118, 2.873,
    1.773,
 1.105, 0.698, 0.4454, 0.2865, 0.1874, 0.124, 0.08371, 0.05689,
    0.03057, 0.01346,
 0.00657, 0.003489, 0.002004, 0.001222, 6.46E-4, 3.042E-4, 1.561E
    -4, 8.796E-5, 5.246E-5, 3.253E-5,
 2.025E-5, 1.355E-5, 9.35E-6, 5.502E-6, 2.896E-6, 1.726E-6, 9.364E
    -7, 5.683E-7, 3.861E-7, 2.517E-7,
 1.542E-7, 1.018E-7, 8.667E-8, 4.864E-8, 2.57E-8, 1.308E-8 };
double p8210_d3x1y1_yerrminus[] = { 147.00340132119393,
   101.8017681575325, 72.70171937444121, 53.30150091695354,
   40.60110835925541, 32.5013845858911, 26.200763347658402,
   21.500930212434998, 17.801123560045305,
 14.800337833982034, 12.300406497347964, 10.300485425454474,
    8.700574693662482, 7.3906765590167725, 6.310641805711998,
    5.420590373750815, 4.6306910931307, 3.700337822415678,
    2.7702887936097924,
 2.110213259365034, 1.590125781188394, 1.2101652779682617,
```

```
    0.9500526301210896, 0.7341151135891427, 0.5670881765651616,
    0.4460907979324389, 0.3470705980056507, 0.2410186714758838,
    0.1480135128966271,
 0.09200543462209175, 0.05800698233833579, 0.036904877726392754,
    0.0236103790736193, 0.015405193929321371, 0.01020441081101697,
    0.006833227641459049, 0.004622434423547834,
    0.002471639132235934, 0.0010907336980216574,
 5.324969483480633E-4, 2.8234730386529286E-4, 1.6224980739587954E
    -4, 9.924716620639604E-5, 5.240400748034448E-5,
    2.4681166909204275E-5, 1.2766362050325849E-5, 7.23743739178447E
    -6, 4.340472324528749E-6, 2.7262428358456996E-6,
 1.722440129583609E-6, 1.169144986731757E-6, 8.190390711071115E-7,
    4.809927234376836E-7, 2.5994614826921364E-7, 1.5953996364547662
    E-7, 9.326119235780764E-8, 6.147015536014205E-8,
    4.393005804685444E-8, 3.0717584540455E-8,
 2.0862646045025065E-8, 1.5211837495845135E-8, 1.3424254169226683E
    -8, 7.1034780213639015E-9, 4.114000486144842E-9,
    2.5489213404889528E-9 };
double p8210_d3x1y1_yerrplus[] = { 147.00340132119393,
   101.8017681575325, 72.70171937444121, 53.30150091695354,
   40.60110835925541, 32.5013845858911, 26.200763347658402,
   21.500930212434998, 17.801123560045305,
 14.800337833982034, 12.300406497347964, 10.300485425454474,
    8.700574693662482, 7.3906765590167725, 6.310641805711998,
    5.420590373750815, 4.6306910931307, 3.700337822415678,
    2.7702887936097924,
 2.110213259365034, 1.590125781188394, 1.2101652779682617,
    0.9500526301210896, 0.7341151135891427, 0.5670881765651616,
    0.4460907979324389, 0.3470705980056507, 0.2410186714758838,
    0.1480135128966271,
 0.09200543462209175, 0.05800698233833579, 0.036904877726392754,
    0.0236103790736193, 0.015405193929321371, 0.01020441081101697,
    0.006833227641459049, 0.004622434423547834,
    0.002471639132235934, 0.0010907336980216574,
 5.324969483480633E-4, 2.8234730386529286E-4, 1.6224980739587954E
    -4, 9.924716620639604E-5, 5.240400748034448E-5,
    2.4681166909204275E-5, 1.2766362050325849E-5, 7.23743739178447E
    -6, 4.340472324528749E-6, 2.7262428358456996E-6,
 1.722440129583609E-6, 1.169144986731757E-6, 8.190390711071115E-7,
    4.809927234376836E-7, 2.5994614826921364E-7, 1.5953996364547662
    E-7, 9.326119235780764E-8, 6.147015536014205E-8,
    4.393005804685444E-8, 3.0717584540455E-8,
 2.0862646045025065E-8, 1.5211837495845135E-8, 1.3424254169226683E
    -8, 7.1034780213639015E-9, 4.114000486144842E-9,
    2.5489213404889528E-9 };
double p8210_d3x1y1_ystatminus[] = { 1.0, 0.6, 0.5, 0.4, 0.3, 0.3,
   0.2, 0.2, 0.2,
 0.1, 0.1, 0.1, 0.1, 0.1, 0.09, 0.08, 0.08, 0.05, 0.04,
 0.03, 0.02, 0.02, 0.01, 0.013, 0.01, 0.009, 0.007, 0.003, 0.002,
 0.001, 9.0E-4, 6.0E-4, 7.0E-4, 4.0E-4, 3.0E-4, 2.1E-4, 1.5E-4, 9.0
    E-5, 4.0E-5,
 2.3E-5, 1.4E-5, 9.0E-6, 7.0E-6, 3.3E-6, 2.0E-6, 1.3E-6, 9.1E-7,
    6.6E-7, 5.0E-7,
```

```
  3.8E-7, 3.0E-7, 2.37E-7, 1.23E-7, 8.4E-8, 6.2E-8, 4.33E-8, 3.23E
      -8, 2.56E-8, 1.99E-8,
  1.5E-8, 1.18E-8, 1.059E-8, 5.37E-9, 3.29E-9, 2.21E-9 };
 double p8210_d3x1y1_ystatplus[] = { 1.0, 0.6, 0.5, 0.4, 0.3, 0.3,
     0.2, 0.2, 0.2,
  0.1, 0.1, 0.1, 0.1, 0.1, 0.09, 0.08, 0.08, 0.05, 0.04,
  0.03, 0.02, 0.02, 0.01, 0.013, 0.01, 0.009, 0.007, 0.003, 0.002,
  0.001, 9.0E-4, 6.0E-4, 7.0E-4, 4.0E-4, 3.0E-4, 2.1E-4, 1.5E-4, 9.0
      E-5, 4.0E-5,
  2.3E-5, 1.4E-5, 9.0E-6, 7.0E-6, 3.3E-6, 2.0E-6, 1.3E-6, 9.1E-7,
      6.6E-7, 5.0E-7,
  3.8E-7, 3.0E-7, 2.37E-7, 1.23E-7, 8.4E-8, 6.2E-8, 4.33E-8, 3.23E
      -8, 2.56E-8, 1.99E-8,
  1.5E-8, 1.18E-8, 1.059E-8, 5.37E-9, 3.29E-9, 2.21E-9 };
 int p8210_d3x1y1_numpoints = 65;
 spec = new TGraphAsymmErrors(p8210_d3x1y1_numpoints,
     p8210_d3x1y1_xval, p8210_d3x1y1_yval, p8210_d3x1y1_xerrminus,
     p8210_d3x1y1_xerrplus, p8210_d3x1y1_yerrminus,
     p8210_d3x1y1_yerrplus);
 spec->SetName("/HepData/8210/d3x1y1");
 spec->SetTitle("/HepData/8210/d3x1y1");
 break;

case 3:
 // Centrality 20-30 %
 double p8210_d4x1y1_xval[] = { 0.175, 0.225, 0.275, 0.325, 0.375,
     0.425, 0.475, 0.525, 0.575,
  0.625, 0.675, 0.725, 0.775, 0.825, 0.875, 0.925, 0.975, 1.05,
      1.15,
  1.25, 1.35, 1.45, 1.55, 1.65, 1.75, 1.85, 1.95, 2.1, 2.3,
  2.5, 2.7, 2.9, 3.1, 3.3, 3.5, 3.7, 3.9, 4.25, 4.75,
  5.25, 5.75, 6.25, 6.75, 7.5, 8.5, 9.5, 10.5, 11.5, 12.5,
  13.5, 14.5, 15.5, 17.0, 19.0, 21.0, 23.0, 25.0, 27.0, 29.0,
  31.0, 33.0, 35.0, 38.0, 42.5, 47.5 };
 double p8210_d4x1y1_xerrminus[] = { 0.024999999999999994,
     0.024999999999999994, 0.025000000000000022,
     0.025000000000000022, 0.025000000000000022,
     0.024999999999999967, 0.024999999999999967,
     0.025000000000000022, 0.02499999999999991,
  0.025000000000000022, 0.025000000000000022, 0.025000000000000022,
      0.025000000000000022, 0.02499999999999991,
      0.025000000000000022, 0.025000000000000022,
      0.025000000000000022, 0.050000000000000044,
      0.04999999999999982,
  0.050000000000000044, 0.050000000000000044, 0.050000000000000044,
      0.050000000000000044, 0.04999999999999982,
      0.050000000000000044, 0.050000000000000044,
      0.050000000000000044, 0.10000000000000009, 0.09999999999999964,
  0.10000000000000009, 0.10000000000000009, 0.10000000000000009,
      0.10000000000000009, 0.09999999999999964, 0.10000000000000009,
      0.10000000000000009, 0.10000000000000009, 0.25, 0.25,
  0.25, 0.25, 0.25, 0.25, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
  0.5, 0.5, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
```

```
 1.0, 1.0, 1.0, 2.0, 2.5, 2.5 };
double p8210_d4x1y1_xerrplus[] = { 0.025000000000000022,
    0.024999999999999994, 0.024999999999999967,
    0.024999999999999967, 0.025000000000000022,
    0.025000000000000022, 0.025000000000000022,
    0.025000000000000022, 0.025000000000000022,
 0.025000000000000022, 0.02499999999999991, 0.025000000000000022,
     0.025000000000000022, 0.025000000000000022,
     0.025000000000000022, 0.02499999999999991,
     0.025000000000000022, 0.050000000000000044,
     0.050000000000000044,
 0.050000000000000044, 0.04999999999999982, 0.050000000000000044,
     0.050000000000000044, 0.050000000000000044,
     0.050000000000000044, 0.04999999999999982,
     0.050000000000000044, 0.10000000000000009, 0.10000000000000009,
 0.10000000000000009, 0.09999999999999964, 0.10000000000000009,
     0.10000000000000009, 0.10000000000000009, 0.10000000000000009,
     0.09999999999999964, 0.10000000000000009, 0.25, 0.25,
 0.25, 0.25, 0.25, 0.25, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
 0.5, 0.5, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
 1.0, 1.0, 1.0, 2.0, 2.5, 2.5 };
double p8210_d4x1y1_yval[] = { 761.4, 617.1, 497.3, 398.5, 318.9,
    258.5, 210.6, 172.9, 143.2,
 119.2, 99.9, 84.13, 71.06, 60.29, 51.54, 44.06, 37.78, 30.07,
     22.53,
 16.99, 12.78, 9.733, 7.418, 5.722, 4.42, 3.472, 2.709, 1.895,
     1.172,
 0.7354, 0.4675, 0.3009, 0.1941, 0.1289, 0.08636, 0.05874, 0.04051,
      0.02226, 0.01009,
 0.005024, 0.002696, 0.001561, 9.552E-4, 5.087E-4, 2.358E-4, 1.228E
     -4, 6.807E-5, 4.01E-5, 2.458E-5,
 1.545E-5, 1.106E-5, 7.367E-6, 4.053E-6, 2.187E-6, 1.216E-6, 7.816E
     -7, 4.364E-7, 2.429E-7, 1.833E-7,
 9.796E-8, 1.206E-7, 5.925E-8, 3.602E-8, 1.929E-8, 1.078E-8 };
double p8210_d4x1y1_yerrminus[] = { 98.10412835349997,
    67.90184091760695, 48.901635964454194, 35.701260481949376,
    27.20073528417936, 21.800917411888886, 17.60113632695344,
    14.4003472180361, 11.90042016064979,
 9.800510190801294, 8.200493887565553, 6.8805886376094305,
     5.800551697899088, 4.910498956317983, 4.190584684742691,
     3.5805027579936315, 3.0605881787656437, 2.440327846827143,
     1.8302458851203574,
 1.3901438774457842, 1.050047617967871, 0.7991226439039255,
     0.6280963301914763, 0.4831035085776132, 0.3730857810209336,
     0.29308360581922693, 0.22807893370497856, 0.15902829936838286,
     0.09802040603874278,
 0.06120988482263301 6, 0.03880631391925803, 0.02490501957437496,
     0.016007810593582122, 0.010604244433244642,
     0.007054083923515511 5, 0.004783022057235363,
     0.003282194387905749 6, 0.0018013605968822566, 8.209750300709516
     E-4,
 4.0649231235043056E-4, 2.1833002542023396E-4, 1.2625371281669304E
     -4, 7.732541626140787E-5, 4.120218440811118E-5,
```

```
    1.9184629264074924E-5, 9.972462083156796E-6, 5.597463711360709E
    -6, 3.3308857680803163E-6, 2.0652602741543256E-6,
 1.3218547575282242E-6, 9.588013350011565E-7, 6.517583908167197E-7,
     3.5679966367697156E-7, 1.9888187448835048E-7,
    1.1538197432874859E-7, 7.889955639925994E-8, 4.851030818290067E
    -8, 3.000833217624731E-8, 2.383044271514904E-8,
 1.4970003340012987E-8, 1.7018225524419402E-8, 1.0331877854485117E
    -8, 5.704533285028671E-9, 3.373129111077725E-9,
    2.2496444163467257E-9 };
double p8210_d4x1y1_yerrplus[] = { 98.10412835349997,
   67.90184091760695, 48.901635964454194, 35.701260481949376,
   27.20073528417936, 21.800917411888886, 17.60113632695344,
   14.4003472180361, 11.90042016064979,
 9.800510190801294, 8.200493887565553, 6.8805886376094305,
    5.800551697899088, 4.910498956317983, 4.190584684742691,
    3.5805027579936315, 3.0605881787656437, 2.440327846827143,
    1.8302458851203574,
 1.3901438774457842, 1.050047617967871, 0.7991226439039255,
    0.6280963301914763, 0.4831035085776132, 0.3730857810209336,
    0.29308360581922693, 0.22807893370497856, 0.15902829936838286,
    0.09802040603874278,
 0.061209884822633016, 0.03880631391925803, 0.02490501957437496,
    0.016007810593582122, 0.010604244433244642,
    0.007054083923515511 5, 0.004783022057235363,
    0.0032821943879057496, 0.001801360596882256 6, 8.209750300709516
    E-4,
 4.0649231235043056E-4, 2.1833002542023396E-4, 1.2625371281669304E
    -4, 7.732541626140787E-5, 4.120218440811118E-5,
    1.9184629264074924E-5, 9.972462083156796E-6, 5.597463711360709E
    -6, 3.3308857680803163E-6, 2.0652602741543256E-6,
 1.3218547575282242E-6, 9.588013350011565E-7, 6.517583908167197E-7,
     3.5679966367697156E-7, 1.9888187448835048E-7,
    1.1538197432874859E-7, 7.889955639925994E-8, 4.851030818290067E
    -8, 3.000833217624731E-8, 2.383044271514904E-8,
 1.4970003340012987E-8, 1.7018225524419402E-8, 1.0331877854485117E
    -8, 5.704533285028671E-9, 3.373129111077725E-9,
    2.2496444163467257E-9 };
double p8210_d4x1y1_ystatminus[] = { 0.9, 0.5, 0.4, 0.3, 0.2, 0.2,
   0.2, 0.1, 0.1,
 0.1, 0.09, 0.09, 0.08, 0.07, 0.07, 0.06, 0.06, 0.04, 0.03,
 0.02, 0.01, 0.014, 0.011, 0.01, 0.008, 0.007, 0.006, 0.003, 0.002,
 0.0011, 7.0E-4, 5.0E-4, 5.0E-4, 3.0E-4, 2.4E-4, 1.7E-4, 1.2E-4,
    7.0E-5, 4.0E-5,
 2.0E-5, 1.2E-5, 8.0E-6, 5.9E-6, 2.9E-6, 1.8E-6, 1.2E-6, 8.0E-7,
    5.8E-7, 4.3E-7,
 3.3E-7, 2.7E-7, 2.1E-7, 1.05E-7, 7.3E-8, 5.2E-8, 3.95E-8, 2.83E-8,
     2.03E-8, 1.7E-8,
 1.197E-8, 1.29E-8, 8.74E-9, 4.61E-9, 2.84E-9, 2.0E-9 };
double p8210_d4x1y1_ystatplus[] = { 0.9, 0.5, 0.4, 0.3, 0.2, 0.2,
   0.2, 0.1, 0.1,
 0.1, 0.09, 0.09, 0.08, 0.07, 0.07, 0.06, 0.06, 0.04, 0.03,
 0.02, 0.01, 0.014, 0.011, 0.01, 0.008, 0.007, 0.006, 0.003, 0.002,
 0.0011, 7.0E-4, 5.0E-4, 5.0E-4, 3.0E-4, 2.4E-4, 1.7E-4, 1.2E-4,
```

```
      7.0E-5, 4.0E-5,
   2.0E-5, 1.2E-5, 8.0E-6, 5.9E-6, 2.9E-6, 1.8E-6, 1.2E-6, 8.0E-7,
       5.8E-7, 4.3E-7,
   3.3E-7, 2.7E-7, 2.1E-7, 1.05E-7, 7.3E-8, 5.2E-8, 3.95E-8, 2.83E-8,
        2.03E-8, 1.7E-8,
   1.197E-8, 1.29E-8, 8.74E-9, 4.61E-9, 2.84E-9, 2.0E-9 };
  int p8210_d4x1y1_numpoints = 65;
  spec = new TGraphAsymmErrors(p8210_d4x1y1_numpoints,
      p8210_d4x1y1_xval, p8210_d4x1y1_yval, p8210_d4x1y1_xerrminus,
      p8210_d4x1y1_xerrplus, p8210_d4x1y1_yerrminus,
      p8210_d4x1y1_yerrplus);
  spec->SetName("/HepData/8210/d4x1y1");
  spec->SetTitle("/HepData/8210/d4x1y1");
  break;

 case 4:
  // Centrality 30-40 %
  double p8210_d5x1y1_xval[] = { 0.175, 0.225, 0.275, 0.325, 0.375,
      0.425, 0.475, 0.525, 0.575,
   0.625, 0.675, 0.725, 0.775, 0.825, 0.875, 0.925, 0.975, 1.05,
       1.15,
   1.25, 1.35, 1.45, 1.55, 1.65, 1.75, 1.85, 1.95, 2.1, 2.3,
   2.5, 2.7, 2.9, 3.1, 3.3, 3.5, 3.7, 3.9, 4.25, 4.75,
   5.25, 5.75, 6.25, 6.75, 7.5, 8.5, 9.5, 10.5, 11.5, 12.5,
   13.5, 14.5, 15.5, 17.0, 19.0, 21.0, 23.0, 25.0, 27.0, 29.0,
   31.0, 33.0, 35.0, 38.0, 42.5, 47.5 };
  double p8210_d5x1y1_xerrminus[] = { 0.024999999999999994,
      0.024999999999999994, 0.025000000000000022,
      0.025000000000000022, 0.025000000000000022,
      0.024999999999999967, 0.024999999999999967,
      0.025000000000000022, 0.02499999999999991,
   0.025000000000000022, 0.025000000000000022, 0.025000000000000022,
       0.025000000000000022, 0.02499999999999991,
       0.025000000000000022, 0.025000000000000022,
       0.025000000000000022, 0.050000000000000044,
       0.04999999999999982,
   0.050000000000000044, 0.050000000000000044, 0.050000000000000044,
       0.050000000000000044, 0.04999999999999982,
       0.050000000000000044, 0.050000000000000044,
       0.050000000000000044, 0.10000000000000009, 0.09999999999999964,
   0.10000000000000009, 0.10000000000000009, 0.10000000000000009,
       0.10000000000000009, 0.09999999999999964, 0.10000000000000009,
       0.10000000000000009, 0.10000000000000009, 0.25, 0.25,
   0.25, 0.25, 0.25, 0.25, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
   0.5, 0.5, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
   1.0, 1.0, 1.0, 2.0, 2.5, 2.5 };
  double p8210_d5x1y1_xerrplus[] = { 0.025000000000000022,
      0.024999999999999994, 0.024999999999999967,
      0.024999999999999967, 0.025000000000000022,
      0.025000000000000022, 0.025000000000000022,
      0.025000000000000022, 0.025000000000000022,
   0.025000000000000022, 0.02499999999999991, 0.025000000000000022,
       0.025000000000000022, 0.025000000000000022,
```

```
    0.025000000000000022, 0.02499999999999991,
    0.025000000000000022, 0.050000000000000044,
    0.050000000000000044,
 0.050000000000000044, 0.04999999999999982, 0.050000000000000044,
    0.050000000000000044, 0.050000000000000044,
    0.050000000000000044, 0.04999999999999982,
    0.050000000000000044, 0.10000000000000009, 0.10000000000000009,
 0.10000000000000009, 0.09999999999999964, 0.10000000000000009,
    0.10000000000000009, 0.10000000000000009, 0.10000000000000009,
    0.09999999999999964, 0.10000000000000009, 0.25, 0.25,
 0.25, 0.25, 0.25, 0.25, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
 0.5, 0.5, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
 1.0, 1.0, 1.0, 2.0, 2.5, 2.5 };
double p8210_d5x1y1_yval[] = { 506.0, 406.7, 326.6, 260.7, 208.2,
   167.9, 136.2, 111.6, 92.13,
 76.48, 64.04, 53.84, 45.45, 38.41, 32.85, 28.05, 23.99, 19.0,
    14.19,
 10.7, 8.031, 6.08, 4.624, 3.566, 2.754, 2.165, 1.685, 1.177,
    0.7309,
 0.4604, 0.2943, 0.1907, 0.125, 0.08338, 0.05653, 0.0389, 0.02727,
    0.01515, 0.007038,
 0.003573, 0.001949, 0.001135, 6.974E-4, 3.686E-4, 1.734E-4, 8.897E
    -5, 4.967E-5, 2.843E-5, 1.77E-5,
 1.071E-5, 7.14E-6, 4.601E-6, 2.867E-6, 1.417E-6, 7.756E-7, 4.396E
    -7, 2.63E-7, 1.863E-7, 1.035E-7,
 6.195E-8, 4.992E-8, 4.565E-8, 1.975E-8, 9.478E-9, 7.298E-9 };
double p8210_d5x1y1_yerrminus[] = { 64.50379833777232,
   44.60179368590461, 32.401388859121454, 23.600847442411894,
   18.101104938649467, 14.4003472180361, 11.600431026474835,
   9.400531899844818, 7.770521218039366,
 6.42049842301982, 5.360457070064082, 4.49040087297337,
    3.780476160485607, 3.190391825465957, 2.7204595200076036,
    2.320538730553748, 1.9804039991880444, 1.570286598045083,
    1.1801694793545543,
 0.8900561780022652, 0.6700902924233421, 0.5090982223500687,
    0.3980803938904804, 0.30608005488760615, 0.23707593720156417,
    0.18606719216455114, 0.14405554484295283, 0.10001999800039992,
    0.06201161181585268,
 0.03890822535146007, 0.024805039810490124, 0.016004999218994044,
    0.010507616285342742, 0.006955242626968523,
    0.004693847036280582, 0.003222623155133097 3,
    0.0022522211259110416, 0.0012414507642270797, 5.788307179132772
    E-4,
 2.93492759706266E-4, 1.603776792449623E-4, 9.32630687893123E-5,
    5.7517736394959075E-5, 3.0402960382173315E-5,
    1.4378456106272329E-5, 7.385309742996566E-6, 4.165873257793617E
    -6, 2.410331927349426E-6, 1.5255490814785343E-6,
 9.396275858019495E-7, 6.404568681808323E-7, 4.2438897252402777E-7,
     2.6032479712851017E-7, 1.3508515832614626E-7, 7.83788236706829
    E-8, 4.860092591710574E-8, 3.220139748520241E-8,
    2.4400204917172313E-8, 1.5720050890502867E-8,
 1.1020499081257618E-8, 9.410850121003946E-9, 8.701752697014551E-9,
     3.857162169263823E-9, 2.168093402047061E-9, 1.7734317015323708
```

```
    E-9 };
double p8210_d5x1y1_yerrplus[] = { 64.50379833777232,
    44.60179368590461, 32.401388859121454, 23.600847442411894,
    18.101104938649467, 14.4003472180361, 11.600431026474835,
    9.400531899844818, 7.770521218039366,
 6.42049842301982, 5.360457070064082, 4.49040087297337,
    3.780476160485607, 3.190391825465957, 2.7204595200076036,
    2.320538730553748, 1.9804039991880444, 1.570286598045083,
    1.1801694793545543,
 0.8900561780022652, 0.6700902924233421, 0.5090982223500687,
    0.3980803938904804, 0.30608005488760615, 0.23707593720156417,
    0.18606719216455114, 0.14405554484295283, 0.10001999800039992,
    0.06201161181585268,
 0.03890822535146007, 0.024805039810490124, 0.016004999218994044,
    0.010507616285342742, 0.006955242626968523,
    0.004693847036280582, 0.0032226231551330973,
    0.0022522211259110416, 0.0012414507642270797, 5.788307179132772
    E-4,
 2.93492759706266E-4, 1.603776792449623E-4, 9.32630687893123E-5,
    5.7517736394959075E-5, 3.0402960382173315E-5,
    1.4378456106272329E-5, 7.385309742996566E-6, 4.165873257793617E
    -6, 2.410331927349426E-6, 1.5255490814785343E-6,
 9.396275858019495E-7, 6.404568681808323E-7, 4.2438897252402777E-7,
     2.6032479712851017E-7, 1.3508515832614626E-7, 7.83788236706829
    E-8, 4.860092591710574E-8, 3.220139748520241E-8,
    2.4400204917172313E-8, 1.5720050890502867E-8,
 1.1020499081257618E-8, 9.410850121003946E-9, 8.701752697014551E-9,
     3.857162169263823E-9, 2.168093402047061E-9, 1.7734317015323708
    E-9 };
double p8210_d5x1y1_ystatminus[] = { 0.7, 0.4, 0.3, 0.2, 0.2, 0.1,
    0.1, 0.1, 0.09,
 0.08, 0.07, 0.06, 0.06, 0.05, 0.05, 0.05, 0.04, 0.03, 0.02,
 0.01, 0.011, 0.01, 0.008, 0.007, 0.006, 0.005, 0.004, 0.002,
    0.0012,
 8.0E-4, 5.0E-4, 4.0E-4, 4.0E-4, 2.7E-4, 1.9E-4, 1.3E-4, 1.0E-4,
    6.0E-5, 3.1E-5,
 1.7E-5, 1.1E-5, 7.0E-6, 5.0E-6, 2.5E-6, 1.5E-6, 9.8E-7, 6.8E-7,
    4.9E-7, 3.7E-7,
 2.7E-7, 2.13E-7, 1.65E-7, 8.8E-8, 5.8E-8, 4.1E-8, 2.94E-8, 2.18E
    -8, 1.76E-8, 1.26E-8,
 9.45E-9, 8.21E-9, 7.61E-9, 3.39E-9, 1.977E-9, 1.632E-9 };
double p8210_d5x1y1_ystatplus[] = { 0.7, 0.4, 0.3, 0.2, 0.2, 0.1,
    0.1, 0.1, 0.09,
 0.08, 0.07, 0.06, 0.06, 0.05, 0.05, 0.05, 0.04, 0.03, 0.02,
 0.01, 0.011, 0.01, 0.008, 0.007, 0.006, 0.005, 0.004, 0.002,
    0.0012,
 8.0E-4, 5.0E-4, 4.0E-4, 4.0E-4, 2.7E-4, 1.9E-4, 1.3E-4, 1.0E-4,
    6.0E-5, 3.1E-5,
 1.7E-5, 1.1E-5, 7.0E-6, 5.0E-6, 2.5E-6, 1.5E-6, 9.8E-7, 6.8E-7,
    4.9E-7, 3.7E-7,
 2.7E-7, 2.13E-7, 1.65E-7, 8.8E-8, 5.8E-8, 4.1E-8, 2.94E-8, 2.18E
    -8, 1.76E-8, 1.26E-8,
 9.45E-9, 8.21E-9, 7.61E-9, 3.39E-9, 1.977E-9, 1.632E-9 };
```

```
 int p8210_d5x1y1_numpoints = 65;
 spec = new TGraphAsymmErrors(p8210_d5x1y1_numpoints,
     p8210_d5x1y1_xval, p8210_d5x1y1_yval, p8210_d5x1y1_xerrminus,
     p8210_d5x1y1_xerrplus, p8210_d5x1y1_yerrminus,
     p8210_d5x1y1_yerrplus);
 spec->SetName("/HepData/8210/d5x1y1");
 spec->SetTitle("/HepData/8210/d5x1y1");
 break;

case 5:
 // Centrality 40-50 %
 double p8210_d6x1y1_xval[] = { 0.175, 0.225, 0.275, 0.325, 0.375,
     0.425, 0.475, 0.525, 0.575,
  0.625, 0.675, 0.725, 0.775, 0.825, 0.875, 0.925, 0.975, 1.05,
      1.15,
  1.25, 1.35, 1.45, 1.55, 1.65, 1.75, 1.85, 1.95, 2.1, 2.3,
  2.5, 2.7, 2.9, 3.1, 3.3, 3.5, 3.7, 3.9, 4.25, 4.75,
  5.25, 5.75, 6.25, 6.75, 7.5, 8.5, 9.5, 10.5, 11.5, 12.5,
  13.5, 14.5, 15.5, 17.0, 19.0, 21.0, 23.0, 25.0, 27.0, 29.0,
  31.0, 33.0, 35.0, 38.0, 42.5, 47.5 };
 double p8210_d6x1y1_xerrminus[] = { 0.024999999999999994,
     0.024999999999999994, 0.025000000000000022,
     0.025000000000000022, 0.025000000000000022,
     0.024999999999999967, 0.024999999999999967,
     0.025000000000000022, 0.02499999999999991,
  0.025000000000000022, 0.025000000000000022, 0.025000000000000022,
      0.025000000000000022, 0.02499999999999991,
      0.025000000000000022, 0.025000000000000022,
      0.025000000000000022, 0.050000000000000044,
      0.04999999999999982,
  0.050000000000000044, 0.050000000000000044, 0.050000000000000044,
      0.050000000000000044, 0.04999999999999982,
      0.050000000000000044, 0.050000000000000044,
      0.050000000000000044, 0.10000000000000009, 0.09999999999999964,
  0.10000000000000009, 0.10000000000000009, 0.10000000000000009,
      0.10000000000000009, 0.09999999999999964, 0.10000000000000009,
      0.10000000000000009, 0.10000000000000009, 0.25, 0.25,
  0.25, 0.25, 0.25, 0.25, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
  0.5, 0.5, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
  1.0, 1.0, 1.0, 2.0, 2.5, 2.5 };
 double p8210_d6x1y1_xerrplus[] = { 0.025000000000000022,
     0.024999999999999994, 0.024999999999999967,
     0.024999999999999967, 0.025000000000000022,
     0.025000000000000022, 0.025000000000000022,
     0.025000000000000022, 0.025000000000000022,
  0.025000000000000022, 0.02499999999999991, 0.025000000000000022,
      0.025000000000000022, 0.025000000000000022,
      0.025000000000000022, 0.02499999999999991,
      0.025000000000000022, 0.050000000000000044,
      0.050000000000000044,
  0.050000000000000044, 0.04999999999999982, 0.050000000000000044,
      0.050000000000000044, 0.050000000000000044,
      0.050000000000000044, 0.04999999999999982,
```

```
    0.050000000000000044, 0.10000000000000009, 0.10000000000000009,
 0.10000000000000009, 0.09999999999999964, 0.10000000000000009,
    0.10000000000000009, 0.10000000000000009, 0.10000000000000009,
    0.09999999999999964, 0.10000000000000009, 0.25, 0.25,
 0.25, 0.25, 0.25, 0.25, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
 0.5, 0.5, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
 1.0, 1.0, 1.0, 2.0, 2.5, 2.5 };
double p8210_d6x1y1_yval[] = { 321.1, 256.7, 205.3, 162.5, 129.2,
    103.8, 83.67, 68.27, 56.17,
 46.49, 38.75, 32.42, 27.27, 23.04, 19.59, 16.72, 14.27, 11.23,
    8.343,
 6.277, 4.698, 3.538, 2.682, 2.06, 1.588, 1.25, 0.9743, 0.6817,
    0.4231,
 0.2681, 0.1725, 0.1128, 0.07501, 0.0508, 0.03483, 0.02432,
    0.01714, 0.009657, 0.004603,
 0.002371, 0.001316, 7.757E-4, 4.777E-4, 2.529E-4, 1.173E-4, 5.86E
    -5, 3.166E-5, 1.866E-5, 1.104E-5,
 7.137E-6, 4.794E-6, 3.088E-6, 1.962E-6, 9.817E-7, 5.071E-7, 2.902E
    -7, 1.504E-7, 1.054E-7, 7.147E-8,
 3.189E-8, 2.715E-8, 1.659E-8, 2.046E-8, 6.635E-9, 4.407E-9 };
double p8210_d6x1y1_yerrminus[] = { 40.303101617617465,
    27.90161285660741, 20.500975586542218, 14.701360481261588,
    11.30044246921332, 9.000555538409838, 7.190563260273843,
    5.8405479195020735, 4.790511454949252,
 3.9404568263083406, 3.2703822406562812, 2.73045783706689,
    2.290349318335524, 1.9304144632694813, 1.64048773235279,
    1.4003213916812096, 1.1903780911962383, 0.9402127418834526,
    0.6982069893663339,
 0.5281145709029433, 0.3960807998376089, 0.29908192857476357,
    0.2330772404161333, 0.17906981878585793, 0.13805795884337854,
    0.10807404868885037, 0.08425704718301015, 0.05851674973885683,
    0.03621118611699981,
 0.02290785891348207, 0.014605478424207815, 0.009504735661763561,
    0.006327598280548474, 0.004275160815688692,
    0.0029138634147811387, 0.0020329781110479278,
    0.0014222517358048821, 8.015609770940698E-4, 3.818193290026056E
    -4,
 1.964993638666548E-4, 1.0937092849564733E-4, 6.447976426755917E-5,
     3.972266355621183E-5, 2.100523744212381E-5, 9.786725703727474E
    -6, 4.925403536767317E-6, 2.696683147868878E-6,
    1.6104657711357918E-6, 9.741663102366044E-7,
 6.466668384879496E-7, 4.448662720413855E-7, 2.9786070569982874E-7,
     1.840923681199196E-7, 9.783905150807627E-8, 5.5120232220120405
    E-8, 3.501785258978626E-8, 2.1319005605327843E-8,
    1.634441800738099E-8, 1.2356945415433379E-8,
 7.388646696114248E-9, 6.55708014286847E-9, 4.8446258885490836E-9,
    3.937765864040167E-9, 1.7700197739008455E-9, 1.3379869207133527
    E-9 };
double p8210_d6x1y1_yerrplus[] = { 40.303101617617465,
    27.90161285660741, 20.500975586542218, 14.701360481261588,
    11.30044246921332, 9.000555538409838, 7.190563260273843,
    5.8405479195020735, 4.790511454949252,
 3.9404568263083406, 3.2703822406562812, 2.73045783706689,
```

```
    2.290349318335524, 1.9304144632694813, 1.64048773235279,
    1.4003213916812096, 1.1903780911962383, 0.9402127418834526,
    0.6982069893663339,
 0.5281145709029433, 0.3960807998376089, 0.29908192857476357,
    0.2330772404161333, 0.17906981878585793, 0.13805795884337854,
    0.10807404868885037, 0.08425704718301015, 0.05851674973885683,
    0.03621118611699981,
 0.02290785891348207, 0.014605478424207815, 0.009504735661763561,
    0.006327598280548474, 0.004275160815688692,
    0.0029138634147811387, 0.0020329781110479278,
    0.0014222517358048821, 8.015609770940698E-4, 3.818193290026056E
    -4,
 1.964993638666548E-4, 1.093709284956473 3E-4, 6.447976426755917E-5,
     3.972266355621183E-5, 2.100523744212381E-5, 9.786725703727474E
    -6, 4.925403536767317E-6, 2.696683147868878E-6,
    1.6104657711357918E-6, 9.741663102366044E-7,
 6.466668384879496E-7, 4.448662720413855E-7, 2.9786070569982874E-7,
     1.840923681199196E-7, 9.783905150807627E-8, 5.5120232220120405
    E-8, 3.501785258978626E-8, 2.1319005605327843E-8,
    1.634441800738099E-8, 1.2356945415433379E-8,
 7.388646696114248E-9, 6.55708014286847E-9, 4.8446258885490836E-9,
    3.937765864040167E-9, 1.7700197739008455E-9, 1.3379869207133527
    E-9 };
double p8210_d6x1y1_ystatminus[] = { 0.5, 0.3, 0.2, 0.2, 0.1, 0.1,
   0.09, 0.08, 0.07,
 0.06, 0.05, 0.05, 0.04, 0.04, 0.04, 0.03, 0.03, 0.02, 0.017,
 0.011, 0.008, 0.007, 0.006, 0.005, 0.004, 0.004, 0.0031, 0.0014,
    9.0E-4,
 6.0E-4, 4.0E-4, 3.0E-4, 3.1E-4, 2.1E-4, 1.5E-4, 1.1E-4, 8.0E-5,
    5.0E-5, 2.5E-5,
 1.4E-5, 9.0E-6, 6.0E-6, 4.2E-6, 2.1E-6, 1.3E-6, 8.0E-7, 5.5E-7,
    4.0E-7, 2.9E-7,
 2.23E-7, 1.75E-7, 1.36E-7, 7.3E-8, 4.88E-8, 3.32E-8, 2.4E-8, 1.65E
    -8, 1.33E-8, 1.054E-8,
 6.8E-9, 6.07E-9, 4.6E-9, 3.46E-9, 1.659E-9, 1.272E-9 };
double p8210_d6x1y1_ystatplus[] = { 0.5, 0.3, 0.2, 0.2, 0.1, 0.1,
   0.09, 0.08, 0.07,
 0.06, 0.05, 0.05, 0.04, 0.04, 0.04, 0.03, 0.03, 0.02, 0.017,
 0.011, 0.008, 0.007, 0.006, 0.005, 0.004, 0.004, 0.0031, 0.0014,
    9.0E-4,
 6.0E-4, 4.0E-4, 3.0E-4, 3.1E-4, 2.1E-4, 1.5E-4, 1.1E-4, 8.0E-5,
    5.0E-5, 2.5E-5,
 1.4E-5, 9.0E-6, 6.0E-6, 4.2E-6, 2.1E-6, 1.3E-6, 8.0E-7, 5.5E-7,
    4.0E-7, 2.9E-7,
 2.23E-7, 1.75E-7, 1.36E-7, 7.3E-8, 4.88E-8, 3.32E-8, 2.4E-8, 1.65E
    -8, 1.33E-8, 1.054E-8,
 6.8E-9, 6.07E-9, 4.6E-9, 3.46E-9, 1.659E-9, 1.272E-9 };
int p8210_d6x1y1_numpoints = 65;
spec = new TGraphAsymmErrors(p8210_d6x1y1_numpoints,
   p8210_d6x1y1_xval, p8210_d6x1y1_yval, p8210_d6x1y1_xerrminus,
   p8210_d6x1y1_xerrplus, p8210_d6x1y1_yerrminus,
   p8210_d6x1y1_yerrplus);
spec->SetName("/HepData/8210/d6x1y1");
```

```
   spec->SetTitle("/HepData/8210/d6x1y1");
   break;
}


 return spec;
}

//=========================================================

TGraphAsymmErrors* GetV2_QC4(Int_t cen)
{
 TGraphAsymmErrors* spec;
 switch (cen) {
  case 0: // centrality 0-5 %
   // Plot: p8508_d2x1y6
   double p8508_d2x1y1_xval[] = { 0.30000000000000004, 0.5, 0.7,
    0.9500000000000001, 1.25, 1.5499999999999998, 2.15, 2.7, 3.5, 5.0
        };
   double p8508_d2x1y1_xerrminus[] = { 0.10000000000000003,
       0.09999999999999998, 0.09999999999999998,
    0.15000000000000002, 0.1499999999999999, 0.1499999999999999,
        0.44999999999999996, 0.10000000000000009, 0.7000000000000002,
        0.7999999999999998 };
   double p8508_d2x1y1_xerrplus[] = { 0.09999999999999998,
       0.09999999999999998, 0.10000000000000009,
    0.15000000000000002, 0.1499999999999999, 0.15000000000000013,
        0.4500000000000002, 0.09999999999999964, 0.7000000000000002,
        0.7999999999999998 };
   double p8508_d2x1y1_yval[] = { 0.01159, 0.0162, 0.01908,
    0.02334, 0.02403, 0.02536, 0.0259, 0.02603, 0.01124, -0.0385 };
   double p8508_d2x1y1_yerrminus[] = { 0.00148771637081804,
       0.0018831091311976584, 0.002250355527466716,
    0.002369071548096427, 0.002872646863086376, 0.003638818489564985,
        0.004013228127081738, 0.006566559220779175,
        0.01008678838877866, 0.018125253101681088 };
   double p8508_d2x1y1_yerrplus[] = { 0.00148771637081804,
       0.0018831091311976584, 0.002250355527466716,
    0.002369071548096427, 0.002872646863086376, 0.003638818489564985,
        0.004013228127081738, 0.006566559220779175,
        0.01008678838877866, 0.018125253101681088 };
   double p8508_d2x1y1_ystatminus[] = { 0.00137, 0.0017, 0.00204,
    0.00206, 0.00261, 0.00341, 0.00392, 0.00651, 0.01008, 0.01808 };
   double p8508_d2x1y1_ystatplus[] = { 0.00137, 0.0017, 0.00204,
    0.00206, 0.00261, 0.00341, 0.00392, 0.00651, 0.01008, 0.01808 };
   int p8508_d2x1y1_numpoints = 10;
   spec = new TGraphAsymmErrors(p8508_d2x1y1_numpoints,
       p8508_d2x1y1_xval, p8508_d2x1y1_yval, p8508_d2x1y1_xerrminus,
       p8508_d2x1y1_xerrplus, p8508_d2x1y1_yerrminus,
       p8508_d2x1y1_yerrplus);
break;

  case 1: // centrality 5-10 %
```

```
double p8508_d2x1y2_xval[] = { 0.25, 0.35, 0.45, 0.55,
    0.6499999999999999, 0.75, 0.8500000000000001, 0.95, 1.05, 1.15,
    1.25, 1.35, 1.45,
 1.55, 1.65, 1.75, 1.85, 1.95, 2.05, 2.1500000000000004, 2.25,
     2.3499999999999996, 2.45, 2.55, 2.6500000000000004, 2.8,
 3.0, 3.2, 3.4, 3.6, 3.8, 4.05, 4.5, 5.1, 5.800000000000001, 6.6,
     7.6, 9.1, 11.25, 13.75, 17.5 };
double p8508_d2x1y2_xerrminus[] = { 0.04999999999999999,
    0.04999999999999999, 0.04999999999999999, 0.050000000000000044,
    0.04999999999999993, 0.050000000000000044, 0.050000000000000044,
     0.04999999999999993, 0.050000000000000044, 0.04999999999999982,
     0.050000000000000044, 0.050000000000000044,
    0.050000000000000044,
 0.050000000000000044, 0.04999999999999982, 0.050000000000000044,
     0.050000000000000044, 0.050000000000000044,
     0.04999999999999982, 0.050000000000000266, 0.04999999999999982,
      0.04999999999999982, 0.050000000000000266,
     0.04999999999999982, 0.050000000000000266, 0.09999999999999964,
 0.10000000000000009, 0.10000000000000009, 0.10000000000000009,
     0.10000000000000009, 0.09999999999999964, 0.1499999999999999,
     0.2999999999999998, 0.2999999999999998, 0.40000000000000036,
     0.39999999999999947, 0.5999999999999996, 0.9000000000000004,
     1.25, 1.25, 2.5 };
double p8508_d2x1y2_xerrplus[] = { 0.04999999999999999,
    0.050000000000000044, 0.04999999999999999, 0.04999999999999993,
    0.050000000000000044, 0.050000000000000044, 0.04999999999999993,
     0.050000000000000044, 0.050000000000000044,
    0.050000000000000044, 0.050000000000000044, 0.04999999999999982,
     0.050000000000000044,
 0.050000000000000044, 0.050000000000000044, 0.050000000000000044,
     0.04999999999999982, 0.050000000000000044,
     0.050000000000000266, 0.04999999999999982, 0.04999999999999982,
      0.050000000000000266, 0.04999999999999982,
     0.050000000000000266, 0.04999999999999982, 0.10000000000000009,
 0.10000000000000009, 0.09999999999999964, 0.10000000000000009,
     0.10000000000000009, 0.10000000000000009, 0.15000000000000036,
     0.2999999999999998, 0.3000000000000007, 0.39999999999999947,
     0.40000000000000036, 0.5999999999999996, 0.9000000000000004,
     1.25, 1.25, 2.5 };
double p8508_d2x1y2_yval[] = { 0.01476, 0.01998, 0.02531, 0.03061,
    0.03495, 0.03916, 0.04319, 0.04706, 0.05005, 0.05359, 0.05782,
    0.0602, 0.06321,
 0.06589, 0.06983, 0.07234, 0.07387, 0.0779, 0.08238, 0.08441,
     0.08498, 0.08915, 0.08993, 0.09138, 0.09301, 0.09554,
 0.09772, 0.09576, 0.09421, 0.096, 0.0948, 0.09504, 0.09093,
     0.07513, 0.06828, 0.05378, 0.0436, 0.06303, 0.03952, 0.05903,
     0.01454 };
double p8508_d2x1y2_yerrminus[] = { 7.912016177940992E-4,
    0.001044030650891055, 0.001314724305700628 6,
    0.001574102919125684, 0.0017996944185055418,
    0.002013256069157622, 0.0022171152428324516,
    0.002413503677229434, 0.0025709920264364882,
    0.0027576983156248256, 0.0029759368272864934,
```

```
    0.0031069277429641006, 0.0032671853329739346,
 0.003423171044514136, 0.0036332079489068613, 0.003786357088284199,
      0.0038864508230517982, 0.0041303268635787165,
     0.0031246279778559235, 0.0032653330611133683,
     0.0034003529228596257, 0.003652560745559203,
     0.003832557892582968, 0.00407176865747552,
     0.004320705960835567, 0.004037189616552584,
 0.004429221150495875, 0.004751136706094659, 0.005136224294167847,
     0.005641630969852601, 0.006194771989347146,
     0.005977365640480763, 0.005620320275571491,
     0.007194622992207444, 0.008563597374935372,
     0.012071950960801655, 0.013866023222250856,
     0.01715776792009963, 0.024065680958576675,
     0.037681009806001746, 0.039482917825307697 };
double p8508_d2x1y2_yerrplus[] = { 7.912016177940992E-4,
    0.001044030650891055, 0.0013147243057006286,
    0.001574102919125684, 0.0017996944185055418,
    0.002013256069157622, 0.0022171152428324516,
    0.002413503677229434, 0.0025709920264364882,
    0.0027576983156248256, 0.0029759368272864934,
    0.0031069277429641006, 0.0032671853329739346,
 0.003423171044514136, 0.0036332079489068613, 0.003786357088284199,
      0.0038864508230517982, 0.0041303268635787165,
     0.0031246279778559235, 0.0032653330611133683,
     0.0034003529228596257, 0.003652560745559203,
     0.003832557892582968, 0.00407176865747552,
     0.004320705960835567, 0.004037189616552584,
 0.004429221150495875, 0.004751136706094659, 0.005136224294167847,
     0.005641630969852601, 0.006194771989347146,
     0.005977365640480763, 0.005620320275571491,
     0.007194622992207444, 0.008563597374935372,
     0.012071950960801655, 0.013866023222250856,
     0.01715776792009963, 0.024065680958576675,
     0.037681009806001746, 0.039482917825307697 };
double p8508_d2x1y2_ystatminus[] = { 2.8E-4, 3.0E-4, 3.4E-4, 3.7E
    -4, 4.2E-4, 4.6E-4, 5.0E-4, 5.5E-4, 6.0E-4, 6.5E-4, 7.1E-4, 7.7E
    -4, 8.3E-4,
 9.1E-4, 0.00101, 0.00111, 0.00122, 0.00136, 0.00152, 0.00168,
     0.0019, 0.00214, 0.00241, 0.00272, 0.00302, 0.0025,
 0.00302, 0.00353, 0.00408, 0.00466, 0.00534, 0.00508, 0.00474,
     0.00675, 0.00826, 0.01194, 0.01379, 0.01703, 0.02403, 0.03763,
     0.03948 };
double p8508_d2x1y2_ystatplus[] = { 2.8E-4, 3.0E-4, 3.4E-4, 3.7E-4,
     4.2E-4, 4.6E-4, 5.0E-4, 5.5E-4, 6.0E-4, 6.5E-4, 7.1E-4, 7.7E-4,
     8.3E-4,
 9.1E-4, 0.00101, 0.00111, 0.00122, 0.00136, 0.00152, 0.00168,
     0.0019, 0.00214, 0.00241, 0.00272, 0.00302, 0.0025,
 0.00302, 0.00353, 0.00408, 0.00466, 0.00534, 0.00508, 0.00474,
     0.00675, 0.00826, 0.01194, 0.01379, 0.01703, 0.02403, 0.03763,
     0.03948 };
int p8508_d2x1y2_numpoints = 41;
spec = new TGraphAsymmErrors(p8508_d2x1y2_numpoints,
    p8508_d2x1y2_xval, p8508_d2x1y2_yval, p8508_d2x1y2_xerrminus,
```

```
      p8508_d2x1y2_xerrplus, p8508_d2x1y2_yerrminus,
      p8508_d2x1y2_yerrplus);
break;

  case 2:  // centrality 10-20 %
   double p8508_d2x1y3_xval[] = { 0.25, 0.35, 0.45, 0.55,
      0.6499999999999999, 0.75, 0.8500000000000001, 0.95, 1.05, 1.15,
      1.25, 1.35, 1.45,
    1.55, 1.65, 1.75, 1.85, 1.95, 2.1,
    2.3, 2.5, 2.7, 2.9, 3.1, 3.3, 3.5, 3.7, 3.9, 4.15,
    4.449999999999999, 4.75, 5.1,
    5.5, 6.0, 6.6, 7.300000000000001, 8.2, 9.2, 10.2, 11.2, 12.35,
       14.0, 17.5 };
   double p8508_d2x1y3_xerrminus[] = { 0.04999999999999999,
      0.04999999999999999, 0.04999999999999999, 0.050000000000000044,
      0.04999999999999993, 0.050000000000000044, 0.050000000000000044,
       0.04999999999999993, 0.050000000000000044, 0.04999999999999982,
       0.050000000000000044, 0.050000000000000044,
      0.050000000000000044,
    0.050000000000000044, 0.04999999999999982, 0.050000000000000044,
       0.050000000000000044, 0.050000000000000044,
       0.10000000000000009,
    0.09999999999999964, 0.10000000000000009, 0.10000000000000009,
       0.10000000000000009, 0.10000000000000009, 0.09999999999999964,
       0.10000000000000009, 0.10000000000000009, 0.10000000000000009,
       0.15000000000000036,
    0.14999999999999947, 0.15000000000000036, 0.1999999999999993,
    0.20000000000000018, 0.2999999999999998, 0.2999999999999998,
       0.40000000000000036, 0.4999999999999991, 0.5, 0.5, 0.5,
       0.6500000000000004, 1.0, 2.5 };
   double p8508_d2x1y3_xerrplus[] = { 0.04999999999999999,
      0.050000000000000044, 0.04999999999999999, 0.04999999999999993,
      0.050000000000000044, 0.050000000000000044, 0.04999999999999993,
       0.050000000000000044, 0.050000000000000044,
      0.050000000000000044, 0.050000000000000044, 0.04999999999999982,
       0.050000000000000044,
    0.050000000000000044, 0.050000000000000044, 0.050000000000000044,
       0.04999999999999982, 0.050000000000000044, 0.10000000000000009,
    0.10000000000000009, 0.10000000000000009, 0.09999999999999964,
       0.10000000000000009, 0.10000000000000009, 0.10000000000000009,
       0.10000000000000009, 0.09999999999999964, 0.10000000000000009,
       0.14999999999999947,
    0.15000000000000036, 0.15000000000000036, 0.20000000000000018,
    0.20000000000000018, 0.2999999999999998, 0.3000000000000007,
       0.39999999999999947, 0.5, 0.5, 0.5, 0.5, 0.6500000000000004,
       1.0, 2.5 };
   double p8508_d2x1y3_yval[] = { 0.02322, 0.03179, 0.0406, 0.04885,
      0.05618, 0.06307, 0.06968, 0.07578, 0.08126, 0.08623, 0.09196,
      0.09674, 0.102,
    0.1071, 0.1118, 0.1164, 0.1211, 0.1248, 0.1308,
    0.1382, 0.1434, 0.1483, 0.1512, 0.1527, 0.1524, 0.1545, 0.1502,
       0.146, 0.1434,
    0.1401, 0.1256, 0.1172,
```

```
 0.1067, 0.09874, 0.09194, 0.07765, 0.06907, 0.06649, 0.05918,
     0.06494, 0.06097, 0.04162, 0.009953 };
double p8508_d2x1y3_yerrminus[] = { 0.0011709824934643558,
    0.0016013119621110686, 0.0020418863827353375,
    0.0024517748673155127, 0.002823915721122003,
    0.0031652172121356854, 0.003496569747624091,
    0.003808017857100988, 0.004079656848314574,
    0.004331397003277349, 0.0046229427857156094,
    0.004865757906020397, 0.005124451190127583,
 0.00543323108288245, 0.00563205113613149, 0.0058420886675914115,
     0.0061400325732035, 0.006251399843235113,
     0.0043566041821583934,
 0.004669047011971501, 0.004903060268852505, 0.005044799302251776,
     0.005192301994298868, 0.005345091205957107,
     0.005442425929675112, 0.005515432893255071,
     0.005546169849544818, 0.005507267925205745,
     0.005458937625582472,
 0.005547071299343465, 0.005597320787662612, 0.00551543289325507,
 0.006021627686929839, 0.006046883494826075, 0.0071800069637849235,
      0.007846304862800068, 0.009102757823868544,
     0.011924697061141637, 0.014730974170094793,
     0.018088230980391643, 0.02054952310882177,
     0.021344657411164977, 0.02101459121658092 };
double p8508_d2x1y3_yerrplus[] = { 0.0011709824934643558,
    0.0016013119621110686, 0.0020418863827353375,
    0.0024517748673155127, 0.002823915721122003,
    0.0031652172121356854, 0.003496569747624091,
    0.003808017857100988, 0.004079656848314574,
    0.004331397003277349, 0.0046229427857156094,
    0.004865757906020397, 0.005124451190127583,
 0.00543323108288245, 0.00563205113613149, 0.0058420886675914115,
     0.0061400325732035, 0.006251399843235113,
     0.0043566041821583934,
 0.004669047011971501, 0.004903060268852505, 0.005044799302251776,
     0.005192301994298868, 0.005345091205957107,
     0.005442425929675112, 0.005515432893255071,
     0.005546169849544818, 0.005507267925205745,
     0.005458937625582472,
 0.005547071299343465, 0.005597320787662612, 0.00551543289325507,
 0.006021627686929839, 0.006046883494826075, 0.0071800069637849235,
      0.007846304862800068, 0.009102757823868544,
     0.011924697061141637, 0.014730974170094793,
     0.018088230980391643, 0.02054952310882177,
     0.021344657411164977, 0.02101459121658092 };
double p8508_d2x1y3_ystatminus[] = { 1.6E-4, 1.9E-4, 2.2E-4, 2.4E
    -4, 2.8E-4, 3.1E-4, 3.4E-4, 3.7E-4, 4.0E-4, 4.3E-4, 4.6E-4, 5.0E
    -4, 5.0E-4,
 6.0E-4, 6.0E-4, 7.0E-4, 7.0E-4, 8.0E-4, 7.0E-4,
 8.0E-4, 0.001, 0.0012, 0.0014, 0.0016, 0.0019, 0.0021, 0.0024,
     0.0027, 0.0026,
 0.0031, 0.0037, 0.0039,
 0.0049, 0.00508, 0.0065, 0.00741, 0.00881, 0.01172, 0.0146,
     0.01796, 0.02045, 0.0213, 0.021012 };
```

```
   double p8508_d2x1y3_ystatplus[] = { 1.6E-4, 1.9E-4, 2.2E-4, 2.4E-4,
        2.8E-4, 3.1E-4, 3.4E-4, 3.7E-4, 4.0E-4, 4.3E-4, 4.6E-4, 5.0E-4,
        5.0E-4,
    6.0E-4, 6.0E-4, 7.0E-4, 7.0E-4, 8.0E-4, 7.0E-4,
    8.0E-4, 0.001, 0.0012, 0.0014, 0.0016, 0.0019, 0.0021, 0.0024,
        0.0027, 0.0026,
    0.0031, 0.0037, 0.0039,
    0.0049, 0.00508, 0.0065, 0.00741, 0.00881, 0.01172, 0.0146,
        0.01796, 0.02045, 0.0213, 0.021012 };
   int p8508_d2x1y3_numpoints = 43;
   spec = new TGraphAsymmErrors(p8508_d2x1y3_numpoints,
       p8508_d2x1y3_xval, p8508_d2x1y3_yval, p8508_d2x1y3_xerrminus,
       p8508_d2x1y3_xerrplus, p8508_d2x1y3_yerrminus,
       p8508_d2x1y3_yerrplus);
break;

  case 3:  // centrality 20-30 %
   double p8508_d2x1y4_xval[] = { 0.25, 0.35, 0.45, 0.55,
       0.6499999999999999, 0.75, 0.8500000000000001, 0.95, 1.05, 1.15,
       1.25, 1.35, 1.45,
    1.55, 1.65, 1.75, 1.85, 1.95, 2.1,
    2.3, 2.5, 2.7, 2.9, 3.15, 3.45,
    3.75, 4.05, 4.35, 4.65, 5.05, 5.55, 6.3, 7.4, 9.0, 11.5, 14.75,
        18.25 };
   double p8508_d2x1y4_xerrminus[] = { 0.04999999999999999,
       0.04999999999999999, 0.04999999999999999, 0.050000000000000044,
       0.04999999999999993, 0.050000000000000044, 0.050000000000000044,
        0.04999999999999993, 0.050000000000000044, 0.04999999999999982,
        0.050000000000000044, 0.050000000000000044,
       0.050000000000000044,
    0.050000000000000044, 0.04999999999999982, 0.050000000000000044,
        0.050000000000000044, 0.050000000000000044,
        0.10000000000000009,
    0.09999999999999964, 0.10000000000000009, 0.10000000000000009,
        0.10000000000000009, 0.1499999999999999, 0.15000000000000036,
    0.1499999999999999, 0.1499999999999999, 0.14999999999999947,
        0.15000000000000036, 0.25, 0.25, 0.5, 0.6000000000000005, 1.0,
        1.5, 1.75, 1.75 };
   double p8508_d2x1y4_xerrplus[] = { 0.04999999999999999,
       0.050000000000000044, 0.04999999999999999, 0.04999999999999993,
       0.050000000000000044, 0.050000000000000044, 0.04999999999999993,
        0.050000000000000044, 0.050000000000000044,
       0.050000000000000044, 0.050000000000000044, 0.04999999999999982,
        0.050000000000000044,
    0.050000000000000044, 0.050000000000000044, 0.050000000000000044,
        0.04999999999999982, 0.050000000000000044, 0.10000000000000009,
    0.10000000000000009, 0.10000000000000009, 0.09999999999999964,
        0.10000000000000009, 0.1499999999999999, 0.1499999999999999,
    0.1499999999999999, 0.15000000000000036, 0.15000000000000036,
        0.14999999999999947, 0.25, 0.25, 0.5, 0.5999999999999996, 1.0,
        1.5, 1.75, 1.75 };
   double p8508_d2x1y4_yval[] = { 0.0304, 0.04171, 0.0535, 0.06445,
       0.0746, 0.08359, 0.09213, 0.1004, 0.1079, 0.1152, 0.1221,
```

```
    0.1287, 0.1351,
 0.1412, 0.1484, 0.1537, 0.1588, 0.165, 0.1705,
 0.1794, 0.1846, 0.1883, 0.1932, 0.1918, 0.1883,
 0.1852, 0.1753, 0.1657, 0.1606, 0.1468, 0.1268, 0.1136, 0.1094,
     0.09527, 0.07049, 0.07918, 0.1034 };
double p8508_d2x1y4_yerrminus[] = { 0.001528397853963424,
    0.002097736875778275, 0.0026782456944798774,
    0.003228203834952186, 0.003739050681657043,
    0.0041900477324250136, 0.00462109294431523,
    0.005015974481593781, 0.005414794548272353,
    0.005813776741499453, 0.006120457499239743,
    0.006419501538281613, 0.006818357573492314,
 0.007125307010929424, 0.007424284477308234, 0.007731752712031082,
     0.007940403012442128, 0.00834865258589672,
     0.005742821606144492,
 0.006067124524847005, 0.0061983868869246945, 0.006334824385884742,
      0.006573431371817919, 0.00655133574166368,
     0.006428841264178172,
 0.006419501538281614, 0.006276941930590086, 0.006217716622683925,
     0.006351377803280167, 0.005964059020499378,
     0.00608276253029822, 0.005738466694161429,
     0.006911584478250989, 0.008037269436817457,
     0.011040819715945007, 0.017469109880013923,
     0.028602797066021358 };
double p8508_d2x1y4_yerrplus[] = { 0.001528397853963424,
    0.002097736875778275, 0.0026782456944798774,
    0.003228203834952186, 0.003739050681657043,
    0.0041900477324250136, 0.00462109294431523,
    0.005015974481593781, 0.005414794548272353,
    0.005813776741499453, 0.006120457499239743,
    0.006419501538281613, 0.006818357573492314,
 0.007125307010929424, 0.007424284477308234, 0.007731752712031082,
     0.007940403012442128, 0.00834865258589672,
     0.005742821606144492,
 0.006067124524847005, 0.0061983868869246945, 0.006334824385884742,
      0.006573431371817919, 0.00655133574166368,
     0.006428841264178172,
 0.006419501538281614, 0.006276941930590086, 0.006217716622683925,
     0.006351377803280167, 0.005964059020499378,
     0.00608276253029822, 0.005738466694161429,
     0.006911584478250989, 0.008037269436817457,
     0.011040819715945007, 0.017469109880013923,
     0.028602797066021358 };
double p8508_d2x1y4_ystatminus[] = { 1.6E-4, 1.8E-4, 2.1E-4, 2.3E
    -4, 2.6E-4, 2.9E-4, 3.2E-4, 4.0E-4, 4.0E-4, 4.0E-4, 5.0E-4, 5.0E
    -4, 5.0E-4,
 6.0E-4, 6.0E-4, 7.0E-4, 8.0E-4, 9.0E-4, 7.0E-4,
 9.0E-4, 0.0011, 0.0013, 0.0015, 0.0014, 0.0017,
 0.002, 0.0024, 0.0029, 0.0035, 0.0034, 0.0044, 0.0043, 0.0059,
     0.00739, 0.01079, 0.01727, 0.0284 };
double p8508_d2x1y4_ystatplus[] = { 1.6E-4, 1.8E-4, 2.1E-4, 2.3E-4,
     2.6E-4, 2.9E-4, 3.2E-4, 4.0E-4, 4.0E-4, 4.0E-4, 5.0E-4, 5.0E-4,
     5.0E-4,
```

```
    6.0E-4, 6.0E-4, 7.0E-4, 8.0E-4, 9.0E-4, 7.0E-4,
    9.0E-4, 0.0011, 0.0013, 0.0015, 0.0014, 0.0017,
    0.002, 0.0024, 0.0029, 0.0035, 0.0034, 0.0044, 0.0043, 0.0059,
        0.00739, 0.01079, 0.01727, 0.0284 };
   int p8508_d2x1y4_numpoints = 37;
   spec = new TGraphAsymmErrors(p8508_d2x1y4_numpoints,
       p8508_d2x1y4_xval, p8508_d2x1y4_yval, p8508_d2x1y4_xerrminus,
       p8508_d2x1y4_xerrplus, p8508_d2x1y4_yerrminus,
       p8508_d2x1y4_yerrplus);
break;

  case 4:  // centrality 30-40 %
   double p8508_d2x1y5_xval[] = { 0.25, 0.35, 0.45, 0.55,
       0.6499999999999999, 0.75, 0.8500000000000001, 0.95, 1.05, 1.15,
       1.25, 1.35, 1.45,
    1.55, 1.65, 1.75, 1.85, 1.95, 2.05, 2.1500000000000004, 2.25,
        2.3499999999999996, 2.45, 2.55, 2.7, 2.9, 3.1, 3.3, 3.5,
    3.75, 4.05, 4.35, 4.7, 5.15, 5.65, 6.300000000000001, 7.1, 7.9,
    8.8, 9.8, 11.15, 13.5, 17.5 };
   double p8508_d2x1y5_xerrminus[] = { 0.04999999999999999,
       0.04999999999999999, 0.04999999999999999, 0.050000000000000044,
       0.04999999999999993, 0.050000000000000044, 0.050000000000000044,
        0.04999999999999993, 0.050000000000000044, 0.04999999999999982,
        0.050000000000000044, 0.050000000000000044,
       0.050000000000000044,
    0.050000000000000044, 0.04999999999999982, 0.050000000000000044,
        0.050000000000000044, 0.050000000000000044,
        0.04999999999999982, 0.050000000000000266, 0.04999999999999982,
         0.04999999999999982, 0.050000000000000266,
        0.04999999999999982, 0.10000000000000009, 0.10000000000000009,
        0.10000000000000009, 0.09999999999999964, 0.10000000000000009,
    0.1499999999999999, 0.1499999999999999, 0.14999999999999947,
        0.20000000000000018, 0.25, 0.25, 0.40000000000000036,
        0.39999999999999947, 0.40000000000000036,
    0.5, 0.5, 0.8499999999999996, 1.5, 2.5 };
   double p8508_d2x1y5_xerrplus[] = { 0.04999999999999999,
       0.050000000000000044, 0.04999999999999999, 0.04999999999999993,
       0.050000000000000044, 0.050000000000000044, 0.04999999999999993,
        0.050000000000000044, 0.050000000000000044,
       0.050000000000000044, 0.050000000000000044, 0.04999999999999982,
        0.050000000000000044,
    0.050000000000000044, 0.050000000000000044, 0.050000000000000044,
        0.04999999999999982, 0.050000000000000044,
        0.050000000000000266, 0.04999999999999982, 0.04999999999999982,
         0.050000000000000266, 0.04999999999999982,
        0.050000000000000266, 0.09999999999999964, 0.10000000000000009,
         0.10000000000000009, 0.10000000000000009, 0.10000000000000009,
    0.1499999999999999, 0.15000000000000036, 0.15000000000000036,
        0.20000000000000018, 0.25, 0.25, 0.39999999999999947,
        0.40000000000000036, 0.40000000000000036,
    0.5, 0.5, 0.8499999999999996, 1.5, 2.5 };
   double p8508_d2x1y5_yval[] = { 0.03401, 0.04733, 0.06098, 0.0734,
       0.08492, 0.09527, 0.1053, 0.1152, 0.1234, 0.1313, 0.1394,
```

```
    0.1469, 0.1537,
 0.1616, 0.1668, 0.1731, 0.178, 0.1836, 0.1876, 0.1914, 0.1959,
     0.1997, 0.2013, 0.2028, 0.2047, 0.2051, 0.2012, 0.1997, 0.1964,
 0.193, 0.1839, 0.1726, 0.1576, 0.1467, 0.1299, 0.1263, 0.1194,
     0.1039,
 0.09718, 0.09439, 0.1006, 0.1166, 0.04328 };
double p8508_d2x1y5_yerrminus[] = { 0.001712921480979207,
    0.002381134183535233, 0.003061061907247222,
    0.0036814399356773432, 0.004262792511957391,
    0.004773594033849129, 0.005315072906367325,
    0.005813776741499453, 0.006220128616033594,
    0.006618912297349165, 0.007025667228100119,
    0.007324616030891995, 0.007731752712031082,
 0.008139410298049852, 0.008338465086573188, 0.00875728268357257,
     0.008967719888578144, 0.009277930803794562,
     0.006334824385884742, 0.006476109943476871,
     0.006694027188471825, 0.006841052550594828,
     0.0069641941385920605, 0.007021395872616784,
     0.0069856996786291925, 0.007060453243241541,
     0.007021395872616784, 0.0070228199464317746,
     0.007038465741907109,
 0.0069079664156682175, 0.0068883960397178095, 0.00685054742338158,
      0.0065604877867426895, 0.006720863039818622,
     0.007300684899377592, 0.007488658090739621, 0.0094847245611035,
      0.011800423721205947,
 0.01350935971835823, 0.0173054124481331, 0.01771016657177453,
     0.021457399656062706, 0.026608992840767197 };
double p8508_d2x1y5_yerrplus[] = { 0.001712921480979207,
    0.002381134183535233, 0.003061061907247222,
    0.0036814399356773432, 0.004262792511957391,
    0.004773594033849129, 0.005315072906367325,
    0.005813776741499453, 0.006220128616033594,
    0.006618912297349165, 0.007025667228100119,
    0.007324616030891995, 0.007731752712031082,
 0.008139410298049852, 0.008338465086573188, 0.00875728268357257,
     0.008967719888578144, 0.009277930803794562,
     0.006334824385884742, 0.006476109943476871,
     0.006694027188471825, 0.006841052550594828,
     0.0069641941385920605, 0.007021395872616784,
     0.0069856996786291925, 0.007060453243241541,
     0.007021395872616784, 0.0070228199464317746,
     0.007038465741907109,
 0.0069079664156682175, 0.0068883960397178095, 0.00685054742338158,
      0.0065604877867426895, 0.006720863039818622,
     0.007300684899377592, 0.007488658090739621, 0.0094847245611035,
      0.011800423721205947,
 0.01350935971835823, 0.0173054124481331, 0.01771016657177453,
     0.021457399656062706, 0.026608992840767197 };
double p8508_d2x1y5_ystatminus[] = { 2.1E-4, 2.3E-4, 2.6E-4, 2.9E
    -4, 3.3E-4, 3.6E-4, 4.0E-4, 4.0E-4, 5.0E-4, 5.0E-4, 6.0E-4, 6.0E
    -4, 7.0E-4,
 8.0E-4, 8.0E-4, 0.001, 0.0011, 0.0012, 0.0013, 0.0015, 0.0016,
     0.0018, 0.0019, 0.0021, 0.0016, 0.0019, 0.0021, 0.0024, 0.0027,
```

```
    0.0026, 0.0032, 0.0038, 0.004, 0.0046, 0.0059, 0.0062, 0.0086,
        0.0113,
    0.01312, 0.01702, 0.0174, 0.0211, 0.02657 };
   double p8508_d2x1y5_ystatplus[] = { 2.1E-4, 2.3E-4, 2.6E-4, 2.9E-4,
        3.3E-4, 3.6E-4, 4.0E-4, 4.0E-4, 5.0E-4, 5.0E-4, 6.0E-4, 6.0E-4,
        7.0E-4,
    8.0E-4, 8.0E-4, 0.001, 0.0011, 0.0012, 0.0013, 0.0015, 0.0016,
        0.0018, 0.0019, 0.0021, 0.0016, 0.0019, 0.0021, 0.0024, 0.0027,
    0.0026, 0.0032, 0.0038, 0.004, 0.0046, 0.0059, 0.0062, 0.0086,
        0.0113,
    0.01312, 0.01702, 0.0174, 0.0211, 0.02657 };
   int p8508_d2x1y5_numpoints = 43;
   spec = new TGraphAsymmErrors(p8508_d2x1y5_numpoints,
       p8508_d2x1y5_xval, p8508_d2x1y5_yval, p8508_d2x1y5_xerrminus,
       p8508_d2x1y5_xerrplus, p8508_d2x1y5_yerrminus,
       p8508_d2x1y5_yerrplus);
break;

  case 5: // centrality 40-50 %
   double p8508_d2x1y6_xval[] = { 0.25, 0.35, 0.45, 0.55,
       0.6499999999999999, 0.75, 0.8500000000000001, 0.95, 1.05, 1.15,
       1.25, 1.35, 1.45,
    1.55, 1.65, 1.75, 1.85, 1.95, 2.1,
    2.3, 2.5, 2.7, 2.9, 3.15, 3.45, 3.8, 4.2, 4.7, 5.4,
        6.199999999999999,
    7.1, 8.1, 9.1, 10.45,
    12.15, 14.5, 18.0 };
   double p8508_d2x1y6_xerrminus[] = { 0.04999999999999999,
       0.04999999999999999, 0.04999999999999999, 0.050000000000000044,
       0.04999999999999993, 0.050000000000000044, 0.050000000000000044,
        0.04999999999999993, 0.050000000000000044, 0.04999999999999982,
        0.050000000000000044, 0.050000000000000044,
       0.050000000000000044,
    0.050000000000000044, 0.04999999999999982, 0.050000000000000044,
        0.050000000000000044, 0.050000000000000044,
        0.10000000000000009,
    0.09999999999999964, 0.10000000000000009, 0.10000000000000009,
        0.10000000000000009, 0.1499999999999999, 0.15000000000000036,
        0.19999999999999973, 0.20000000000000018, 0.2999999999999998,
        0.40000000000000036, 0.39999999999999947,
    0.5, 0.5, 0.5, 0.8499999999999996,
    0.8499999999999996, 1.5, 2.0 };
   double p8508_d2x1y6_xerrplus[] = { 0.04999999999999999,
       0.050000000000000044, 0.04999999999999999, 0.04999999999999993,
       0.050000000000000044, 0.050000000000000044, 0.04999999999999993,
        0.050000000000000044, 0.050000000000000044,
       0.050000000000000044, 0.050000000000000044, 0.04999999999999982,
        0.050000000000000044,
    0.050000000000000044, 0.050000000000000044, 0.050000000000000044,
        0.04999999999999982, 0.050000000000000044, 0.10000000000000009,
    0.10000000000000009, 0.10000000000000009, 0.09999999999999964,
        0.10000000000000009, 0.1499999999999999, 0.1499999999999999,
        0.20000000000000018, 0.20000000000000018, 0.2999999999999998,
```

```
    0.39999999999999947, 0.40000000000000036,
 0.5, 0.5, 0.5, 0.8500000000000014,
 0.8499999999999996, 1.5, 2.0 };
double p8508_d2x1y6_yval[] = { 0.03419, 0.04833, 0.06288, 0.07596,
    0.08822, 0.09986, 0.1105, 0.12, 0.1283, 0.1369, 0.1442, 0.1521,
    0.1584,
 0.1639, 0.171, 0.1769, 0.1804, 0.1835, 0.1897,
 0.1914, 0.1924, 0.1947, 0.1945, 0.1895, 0.1816, 0.1762, 0.1574,
    0.1447, 0.1313, 0.1233,
 0.119, 0.1057, 0.1128, 0.1163,
 0.1665, 0.1529, 0.1121 };
double p8508_d2x1y6_yerrminus[] = { 0.0017474839055052838,
    0.0024481217289996017, 0.003167964646267379,
    0.003827740848072137, 0.004440551767517185,
    0.005022449601539073, 0.005532630477449221,
    0.006040695324215582, 0.0064498061986388395,
    0.006859300255857007, 0.007269112738154499,
    0.007679192665899196, 0.007990619500389191,
 0.008302409288875129, 0.008729833904490968, 0.008962700485902673,
    0.009198369420718, 0.009436630754670864, 0.00652533524043018,
 0.006609841147864296, 0.006800735254367722, 0.007038465741907109,
    0.007158910531638176, 0.0069354163537598814,
    0.0069971422738143605, 0.00698927750200262,
    0.007283543093852057, 0.007299999999999999,
    0.008183520025025906, 0.01071540946487814,
 0.013189768762188366, 0.018238969269122638, 0.024085887984460944,
    0.025994230129011323,
 0.03631886562105155, 0.04062142291943994, 0.05113403954314581 };
double p8508_d2x1y6_yerrplus[] = { 0.0017474839055052838,
    0.0024481217289996017, 0.003167964646267379,
    0.003827740848072137, 0.004440551767517185,
    0.005022449601539073, 0.005532630477449221,
    0.006040695324215582, 0.0064498061986388395,
    0.006859300255857007, 0.007269112738154499,
    0.007679192665899196, 0.007990619500389191,
 0.008302409288875129, 0.008729833904490968, 0.008962700485902673,
    0.009198369420718, 0.009436630754670864, 0.00652533524043018,
 0.006609841147864296, 0.006800735254367722, 0.007038465741907109,
    0.007158910531638176, 0.0069354163537598814,
    0.0069971422738143605, 0.00698927750200262,
    0.007283543093852057, 0.007299999999999999,
    0.008183520025025906, 0.01071540946487814,
 0.013189768762188366, 0.018238969269122638, 0.024085887984460944,
    0.025994230129011323,
 0.03631886562105155, 0.04062142291943994, 0.05113403954314581 };
double p8508_d2x1y6_ystatminus[] = { 3.6E-4, 3.7E-4, 4.2E-4, 4.6E
    -4, 5.2E-4, 5.7E-4, 6.0E-4, 7.0E-4, 8.0E-4, 9.0E-4, 0.001,
    0.0011, 0.0012,
 0.0013, 0.0015, 0.0017, 0.0019, 0.0021, 0.0017,
 0.002, 0.0023, 0.0027, 0.003, 0.0029, 0.0036, 0.0039, 0.0051,
    0.0055, 0.0069, 0.0099,
 0.0126, 0.0179, 0.0238, 0.0257,
 0.0359, 0.0403, 0.051 };
```

```
   double p8508_d2x1y6_ystatplus[] = { 3.6E-4, 3.7E-4, 4.2E-4, 4.6E-4,
        5.2E-4, 5.7E-4, 6.0E-4, 7.0E-4, 8.0E-4, 9.0E-4, 0.001, 0.0011,
       0.0012,
    0.0013, 0.0015, 0.0017, 0.0019, 0.0021, 0.0017,
    0.002, 0.0023, 0.0027, 0.003, 0.0029, 0.0036, 0.0039, 0.0051,
        0.0055, 0.0069, 0.0099,
    0.0126, 0.0179, 0.0238, 0.0257,
    0.0359, 0.0403, 0.051 };
   int p8508_d2x1y6_numpoints = 37;
   spec = new TGraphAsymmErrors(p8508_d2x1y6_numpoints,
       p8508_d2x1y6_xval, p8508_d2x1y6_yval, p8508_d2x1y6_xerrminus,
       p8508_d2x1y6_xerrplus, p8508_d2x1y6_yerrminus,
       p8508_d2x1y6_yerrplus);
break;

 }
 return spec;
}
//=========================================================


TGraphAsymmErrors* GetDelta(Int_t choise)
{
  TGraphAsymmErrors* spec;

  double deltascale = 1;
  // Plot: p8370_d5x1y2
  double p8370_d5x1y1_xval[] = {7.5, 15.0, 25.0, 35.0, 45.0, 55.0,
      65.0 };
  double p8370_d5x1y1_xerrminus[] = {2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0
       };
  double p8370_d5x1y1_xerrplus[] = {2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0
      };
  double p8370_d5x1y1_yval[] = {deltascale*(5.61E-4), deltascale*(7.32
      E-4), deltascale*(0.001021), deltascale*(0.001432), deltascale
      *(0.002121), deltascale*(0.003306), deltascale*(0.005447) };
  double p8370_d5x1y1_yerrminus[] = {deltascale*(3.0E-6), deltascale
      *(3.0E-6), deltascale*(4.0E-6), deltascale*(6.324555320336759E-6)
      , deltascale*(1.0049875621120892E-5), deltascale
      *(1.824828759089466E-5), deltascale*(3.413209633175202E-5) };
  double p8370_d5x1y1_yerrplus[] = {deltascale*(3.0E-6), deltascale
      *(3.0E-6), deltascale*(4.0E-6), deltascale*(6.324555320336759E-6)
      , deltascale*(1.0049875621120892E-5), deltascale
      *(1.824828759089466E-5), deltascale*(3.413209633175202E-5) };
  double p8370_d5x1y1_ystatminus[] = {deltascale*(3.0E-6), deltascale
      *(3.0E-6), deltascale*(4.0E-6), deltascale*(6.0E-6), deltascale
      *(1.0E-5), deltascale*(1.8E-5), deltascale*(3.4E-5) };
  double p8370_d5x1y1_ystatplus[] = {deltascale*(3.0E-6), deltascale
      *(3.0E-6), deltascale*(4.0E-6), deltascale*(6.0E-6), deltascale
      *(1.0E-5), deltascale*(1.8E-5), deltascale*(3.4E-5) };
  int p8370_d5x1y1_numpoints = 7;

  deltascale = 1;
```

```
double p8370_d5x1y2_xval[] = {7.5, 15.0, 25.0, 35.0, 45.0, 55.0,
    65.0 };
double p8370_d5x1y2_xerrminus[] = {2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0
     };
double p8370_d5x1y2_xerrplus[] = {2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0
    };
double p8370_d5x1y2_yval[] = {deltascale*(1.78E-4), deltascale*(2.41
    E-4), deltascale*(3.57E-4), deltascale*(5.02E-4), deltascale
    *(7.57E-4), deltascale*(0.001168), deltascale*(0.002095) };
double p8370_d5x1y2_yerrminus[] = {deltascale*(2.0E-6), deltascale
    *(2.2360679774997895E-6), deltascale*(3.1622776601683796E-6),
    deltascale*(5.0E-6), deltascale*(9.899494936611665E-6),
    deltascale*(1.4422205101855957E-5), deltascale
    *(2.4738633753705964E-5) };
double p8370_d5x1y2_yerrplus[] = {deltascale*(2.0E-6), deltascale
    *(2.2360679774997895E-6), deltascale*(3.1622776601683796E-6),
    deltascale*(5.0E-6), deltascale*(9.899494936611665E-6),
    deltascale*(1.4422205101855957E-5), deltascale
    *(2.4738633753705964E-5) };
double p8370_d5x1y2_ystatminus[] = {deltascale*(2.0E-6), deltascale
    *(2.0E-6), deltascale*(3.0E-6), deltascale*(4.0E-6), deltascale
    *(7.0E-6), deltascale*(1.2E-5), deltascale*(2.4E-5) };
double p8370_d5x1y2_ystatplus[] = {deltascale*(2.0E-6), deltascale
    *(2.0E-6), deltascale*(3.0E-6), deltascale*(4.0E-6), deltascale
    *(7.0E-6), deltascale*(1.2E-5), deltascale*(2.4E-5) };
int p8370_d5x1y2_numpoints = 7;

deltascale = 1;

double d_xval[] = { 7.5, 15.0, 25.0, 35.0, 45.0, 55.0, 65.0 };
double d_xerrminus[] = { 2.5, 2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0 };
double d_xerrplus[] = { 2.5, 2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0 };
double d_yval[] = {deltascale*(5.61E-4-1.78E-4), deltascale*(7.32E
    -4-2.41E-4), deltascale*(0.001021-3.57E-4), deltascale
    *(0.001432-5.02E-4), deltascale*(0.002121-7.57E-4), deltascale
    *(0.003306-0.001168), deltascale*(0.005447-0.002095)};
double d_yerrminus[] = {deltascale*(3.0E-6+2.0E-6), deltascale*(3.0E
    -6+2.2360679774997895E-6), deltascale*(4.0E-6+3.1622776601683796E
    -6), deltascale*(6.324555320336759E-6+5.0E-6), deltascale
    *(1.0049875621120892E-5+9.899494936611665E-6), deltascale
    *(1.824828759089466E-5+1.4422205101855957E-5), deltascale
    *(3.413209633175202E-5 +2.4738633753705964E-5)};
double d_yerrplus[] = {deltascale*(3.0E-6+2.0E-6), deltascale*(3.0E
    -6+2.2360679774997895E-6), deltascale*(4.0E-6+3.1622776601683796E
    -6), deltascale*(6.324555320336759E-6+5.0E-6), deltascale
    *(1.0049875621120892E-5+9.899494936611665E-6), deltascale
    *(1.824828759089466E-5+1.4422205101855957E-5), deltascale
    *(3.413209633175202E-5 +2.4738633753705964E-5)};
double d_ystatminus[] = {deltascale*(3.0E-6+2.0E-6), deltascale*(3.0
    E-6+2.0E-6), deltascale*(4.0E-6+3.0E-6), deltascale*(6.0E-6+4.0E
    -6), deltascale*(1.0E-5+7.0E-6), deltascale*(1.8E-5+1.2E-5),
    deltascale*(3.4E-5+2.4E-5) };
```

```
  double d_ystatplus[] = {deltascale*(3.0E-6+2.0E-6), deltascale*(3.0E
      -6+2.0E-6), deltascale*(4.0E-6+3.0E-6), deltascale*(6.0E-6+4.0E
      -6), deltascale*(1.0E-5+7.0E-6), deltascale*(1.8E-5+1.2E-5),
      deltascale*(3.4E-5+2.4E-5) };
  int d_numpoints = 7;
switch (choise) {
case 0: // Opposite charge
        spec = new TGraphAsymmErrors(p8370_d5x1y1_numpoints,
            p8370_d5x1y1_xval, p8370_d5x1y1_yval,
            p8370_d5x1y1_xerrminus, p8370_d5x1y1_xerrplus,
            p8370_d5x1y1_yerrminus, p8370_d5x1y1_yerrplus);
        //spec.SetName("/HepData/8370/d5x1y1");
    //spec.SetTitle("/HepData/8370/d5x1y1");
break;
case 1: // Same Charge
        spec = new TGraphAsymmErrors(p8370_d5x1y2_numpoints,
            p8370_d5x1y2_xval, p8370_d5x1y2_yval,
            p8370_d5x1y2_xerrminus, p8370_d5x1y2_xerrplus,
            p8370_d5x1y2_yerrminus, p8370_d5x1y2_yerrplus);
        //spec.SetName("/HepData/8370/d5x1y2");
    //spec.SetTitle("/HepData/8370/d5x1y2");
break;
case 2: // delta
   spec = new TGraphAsymmErrors(d_numpoints, d_xval, d_yval,
       d_xerrminus, d_xerrplus, d_yerrminus, d_yerrplus);
   //spec.SetName("/HepData/8370/d5");
   //spec.SetTitle("/HepData/8370/d5");
break;
}
return spec;
}

//=========================================================


TGraphAsymmErrors* GetDeltascaled(Int_t choise)
{
  TGraphAsymmErrors* spec;

  double deltascale = 0.1;
  // Plot: p8370_d5x1y2
  double p8370_d5x1y1_xval[] = {7.5, 15.0, 25.0, 35.0, 45.0, 55.0,
      65.0 };
  double p8370_d5x1y1_xerrminus[] = {2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0
       };
  double p8370_d5x1y1_xerrplus[] = {2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0
      };
  double p8370_d5x1y1_yval[] = {deltascale*(5.61E-4), deltascale*(7.32
      E-4), deltascale*(0.001021), deltascale*(0.001432), deltascale
      *(0.002121), deltascale*(0.003306), deltascale*(0.005447) };
  double p8370_d5x1y1_yerrminus[] = {deltascale*(3.0E-6), deltascale
      *(3.0E-6), deltascale*(4.0E-6), deltascale*(6.324555320336759E-6)
      , deltascale*(1.0049875621120892E-5), deltascale
```

```
    *(1.824828759089466E-5), deltascale*(3.413209633175202E-5) };
double p8370_d5x1y1_yerrplus[] = {deltascale*(3.0E-6), deltascale
    *(3.0E-6), deltascale*(4.0E-6), deltascale*(6.324555320336759E-6)
    , deltascale*(1.0049875621120892E-5), deltascale
    *(1.824828759089466E-5), deltascale*(3.413209633175202E-5) };
double p8370_d5x1y1_ystatminus[] = {deltascale*(3.0E-6), deltascale
    *(3.0E-6), deltascale*(4.0E-6), deltascale*(6.0E-6), deltascale
    *(1.0E-5), deltascale*(1.8E-5), deltascale*(3.4E-5) };
double p8370_d5x1y1_ystatplus[] = {deltascale*(3.0E-6), deltascale
    *(3.0E-6), deltascale*(4.0E-6), deltascale*(6.0E-6), deltascale
    *(1.0E-5), deltascale*(1.8E-5), deltascale*(3.4E-5) };
int p8370_d5x1y1_numpoints = 7;

deltascale = 0.005;

double p8370_d5x1y2_xval[] = {7.5, 15.0, 25.0, 35.0, 45.0, 55.0,
    65.0 };
double p8370_d5x1y2_xerrminus[] = {2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0
     };
double p8370_d5x1y2_xerrplus[] = {2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0
    };
double p8370_d5x1y2_yval[] = {deltascale*(1.78E-4), deltascale*(2.41
    E-4), deltascale*(3.57E-4), deltascale*(5.02E-4), deltascale
    *(7.57E-4), deltascale*(0.001168), deltascale*(0.002095) };
double p8370_d5x1y2_yerrminus[] = {deltascale*(2.0E-6), deltascale
    *(2.2360679774997895E-6), deltascale*(3.1622776601683796E-6),
    deltascale*(5.0E-6), deltascale*(9.899494936611665E-6),
    deltascale*(1.4422205101855957E-5), deltascale
    *(2.4738633753705964E-5) };
double p8370_d5x1y2_yerrplus[] = {deltascale*(2.0E-6), deltascale
    *(2.2360679774997895E-6), deltascale*(3.1622776601683796E-6),
    deltascale*(5.0E-6), deltascale*(9.899494936611665E-6),
    deltascale*(1.4422205101855957E-5), deltascale
    *(2.4738633753705964E-5) };
double p8370_d5x1y2_ystatminus[] = {deltascale*(2.0E-6), deltascale
    *(2.0E-6), deltascale*(3.0E-6), deltascale*(4.0E-6), deltascale
    *(7.0E-6), deltascale*(1.2E-5), deltascale*(2.4E-5) };
double p8370_d5x1y2_ystatplus[] = {deltascale*(2.0E-6), deltascale
    *(2.0E-6), deltascale*(3.0E-6), deltascale*(4.0E-6), deltascale
    *(7.0E-6), deltascale*(1.2E-5), deltascale*(2.4E-5) };
int p8370_d5x1y2_numpoints = 7;

deltascale = 0.15;

double d_xval[] = { 7.5, 15.0, 25.0, 35.0, 45.0, 55.0, 65.0 };
double d_xerrminus[] = { 2.5, 2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0 };
double d_xerrplus[] = { 2.5, 2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0 };
double d_yval[] = {deltascale*(5.61E-4-1.78E-4), deltascale*(7.32E
    -4-2.41E-4), deltascale*(0.001021-3.57E-4), deltascale
    *(0.001432-5.02E-4), deltascale*(0.002121-7.57E-4), deltascale
    *(0.003306-0.001168), deltascale*(0.005447-0.002095)};
double d_yerrminus[] = {deltascale*(3.0E-6+2.0E-6), deltascale*(3.0E
    -6+2.2360679774997895E-6), deltascale*(4.0E-6+3.1622776601683796E
```

```
      -6), deltascale*(6.324555320336759E-6+5.0E-6), deltascale
      *(1.0049875621120892E-5+9.899494936611665E-6), deltascale
      *(1.824828759089466E-5+1.4422205101855957E-5), deltascale
      *(3.413209633175202E-5 +2.4738633753705964E-5)};
  double d_yerrplus[] = {deltascale*(3.0E-6+2.0E-6), deltascale*(3.0E
      -6+2.2360679774997895E-6), deltascale*(4.0E-6+3.1622776601683796E
      -6), deltascale*(6.324555320336759E-6+5.0E-6), deltascale
      *(1.0049875621120892E-5+9.899494936611665E-6), deltascale
      *(1.824828759089466E-5+1.4422205101855957E-5), deltascale
      *(3.413209633175202E-5 +2.4738633753705964E-5)};
  double d_ystatminus[] = {deltascale*(3.0E-6+2.0E-6), deltascale*(3.0
      E-6+2.0E-6), deltascale*(4.0E-6+3.0E-6), deltascale*(6.0E-6+4.0E
      -6), deltascale*(1.0E-5+7.0E-6), deltascale*(1.8E-5+1.2E-5),
      deltascale*(3.4E-5+2.4E-5) };
  double d_ystatplus[] = {deltascale*(3.0E-6+2.0E-6), deltascale*(3.0E
      -6+2.0E-6), deltascale*(4.0E-6+3.0E-6), deltascale*(6.0E-6+4.0E
      -6), deltascale*(1.0E-5+7.0E-6), deltascale*(1.8E-5+1.2E-5),
      deltascale*(3.4E-5+2.4E-5) };
  int d_numpoints = 7;
switch (choise) {
case 0: // Opposite charge
        spec = new TGraphAsymmErrors(p8370_d5x1y1_numpoints,
            p8370_d5x1y1_xval, p8370_d5x1y1_yval,
            p8370_d5x1y1_xerrminus, p8370_d5x1y1_xerrplus,
            p8370_d5x1y1_yerrminus, p8370_d5x1y1_yerrplus);
        //spec.SetName("/HepData/8370/d5x1y1");
    //spec.SetTitle("/HepData/8370/d5x1y1");
break;
case 1: // Same Charge
        spec = new TGraphAsymmErrors(p8370_d5x1y2_numpoints,
            p8370_d5x1y2_xval, p8370_d5x1y2_yval,
            p8370_d5x1y2_xerrminus, p8370_d5x1y2_xerrplus,
            p8370_d5x1y2_yerrminus, p8370_d5x1y2_yerrplus);
        //spec.SetName("/HepData/8370/d5x1y2");
    //spec.SetTitle("/HepData/8370/d5x1y2");
break;
case 2: // delta
   spec = new TGraphAsymmErrors(d_numpoints, d_xval, d_yval,
       d_xerrminus, d_xerrplus, d_yerrminus, d_yerrplus);
   //spec.SetName("/HepData/8370/d5");
   //spec.SetTitle("/HepData/8370/d5");
break;
}
return spec;
}



//=========================================================


TGraphAsymmErrors* GetGamma(Int_t choise)
{
```

```
TGraphAsymmErrors* spec;

double gammascale = 1;
// Plot: p8370_d1x1y2
double p8370_d1x1y1_xval[] = {7.5, 15.0, 25.0, 35.0, 45.0, 55.0,
    65.0, 75.0 };
double p8370_d1x1y1_xerrminus[] = {2.5, 5.0, 5.0, 5.0, 5.0, 5.0,
    5.0, 5.0 };
double p8370_d1x1y1_xerrplus[] = {2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0,
     5.0 };
double p8370_d1x1y1_yval[] = {gammascale*(4.0E-6), gammascale*(-3.0E
    -6), gammascale*(-1.4E-5), gammascale*(-2.6E-5), gammascale*(-3.9
    E-5), gammascale*(2.2E-5), gammascale*(2.49E-4), gammascale
    *(0.001059) };
double p8370_d1x1y1_yerrminus[] = {gammascale*(5.0E-6), gammascale
    *(5.0E-6), gammascale*(7.0E-6), gammascale*(1.1E-5), gammascale
    *(1.8E-5), gammascale*(3.6E-5), gammascale*(8.7E-5), gammascale
    *(2.79E-4) };
double p8370_d1x1y1_yerrplus[] = {gammascale*(5.0E-6), gammascale
    *(5.0E-6), gammascale*(7.0E-6), gammascale*(1.1E-5), gammascale
    *(1.8E-5), gammascale*(3.6E-5), gammascale*(8.7E-5), gammascale
    *(2.79E-4) };
double p8370_d1x1y1_ystatminus[] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
    0.0, 0.0 };
double p8370_d1x1y1_ystatplus[] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
    0.0, 0.0 };
int p8370_d1x1y1_numpoints = 8;

gammascale = 1;

double p8370_d1x1y2_xval[] = {7.5, 15.0, 25.0, 35.0, 45.0, 55.0,
    65.0, 75.0 };
double p8370_d1x1y2_xerrminus[] = {2.5, 5.0, 5.0, 5.0, 5.0, 5.0,
    5.0, 5.0 };
double p8370_d1x1y2_xerrplus[] = {2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0,
     5.0 };
double p8370_d1x1y2_yval[] = {gammascale*(-2.3E-5), gammascale*(-5.5
    E-5), gammascale*(-1.06E-4), gammascale*(-1.66E-4), gammascale
    *(-2.76E-4), gammascale*(-3.79E-4), gammascale*(-4.25E-4),
    gammascale*(-2.93E-4) };
double p8370_d1x1y2_yerrminus[] = {gammascale*(3.0E-6), gammascale
    *(3.0E-6), gammascale*(5.0E-6), gammascale*(8.0E-6), gammascale
    *(1.3E-5), gammascale*(2.6E-5), gammascale*(6.3E-5), gammascale
    *(1.96E-4) };
double p8370_d1x1y2_yerrplus[] = {gammascale*(3.0E-6), gammascale
    *(3.0E-6), gammascale*(5.0E-6), gammascale*(8.0E-6), gammascale
    *(1.3E-5), gammascale*(2.6E-5), gammascale*(6.3E-5), gammascale
    *(1.96E-4) };
double p8370_d1x1y2_ystatminus[] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
    0.0, 0.0 };
double p8370_d1x1y2_ystatplus[] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
    0.0, 0.0 };
int p8370_d1x1y2_numpoints = 8;
```

```
  gammascale = 1;

  double g_xval[] = {7.5, 15.0, 25.0, 35.0, 45.0, 55.0, 65.0, 75.0 };
  double g_xerrminus[] = { 2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0 };
  double g_xerrplus[] = { 2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0 };
  double g_yval[] = { gammascale*(4.0E-6+2.3E-5), gammascale*(-3.0E
      -6+5.5E-5), gammascale*(-1.4E-5+1.06E-4), gammascale*(-2.6E
      -5+1.66E-4), gammascale*(-3.9E-5+2.76E-4), gammascale*(2.2E
      -5+3.79E-4), gammascale*(2.49E-4+4.25E-4), gammascale
      *(0.001059+2.93E-4)};
  double g_yerrminus[] = {  gammascale*(5.0E-6+3.0E-6), gammascale
      *(5.0E-6+3.0E-6), gammascale*(7.0E-6+5.0E-6), gammascale*(1.1E
      -5+8.0E-6), gammascale*(1.8E-5+1.3E-5), gammascale*(3.6E-5+2.6E
      -5), gammascale*(8.7E-5+6.3E-5), gammascale*(2.79E-4+1.96E-4)};
  double g_yerrplus[] = {  gammascale*(5.0E-6+3.0E-6), gammascale*(5.0
      E-6+3.0E-6), gammascale*(7.0E-6+5.0E-6), gammascale*(1.1E-5+8.0E
      -6), gammascale*(1.8E-5+1.3E-5), gammascale*(3.6E-5+2.6E-5),
      gammascale*(8.7E-5+6.3E-5), gammascale*(2.79E-4+1.96E-4)};
  double g_ystatminus[] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
      0.0 };
  double g_ystatplus[] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
       };
  int g_numpoints = 8;

switch (choise) {
case 0: // Opposite charge
        spec = new TGraphAsymmErrors(p8370_d1x1y1_numpoints,
            p8370_d1x1y1_xval, p8370_d1x1y1_yval,
            p8370_d1x1y1_xerrminus, p8370_d1x1y1_xerrplus,
            p8370_d1x1y1_yerrminus, p8370_d1x1y1_yerrplus);
        //spec.SetName("/HepData/8370/d1x1y1");
        //spec.SetTitle("/HepData/8370/d1x1y1");
break;
case 1: // Same Charge
        spec = new TGraphAsymmErrors(p8370_d1x1y2_numpoints,
            p8370_d1x1y2_xval, p8370_d1x1y2_yval,
            p8370_d1x1y2_xerrminus, p8370_d1x1y2_xerrplus,
            p8370_d1x1y2_yerrminus, p8370_d1x1y2_yerrplus);
        //spec.SetName("/HepData/8370/d1x1y2");
        //spec.SetTitle("/HepData/8370/d1x1y2");
break;
case 2: // gamma
   spec = new TGraphAsymmErrors(g_numpoints, g_xval, g_yval,
       g_xerrminus, g_xerrplus, g_yerrminus, g_yerrplus);
        //spec.SetName("/HepData/8370/d1");
        //spec.SetTitle("/HepData/8370/d1");
break;
}
return spec;
}

//=========================================================
```

```
TGraphAsymmErrors* GetGammascaled(Int_t choise)
{
  TGraphAsymmErrors* spec;

  double gammascale = -0.75;
  // Plot: p8370_d1x1y2
  double p8370_d1x1y1_xval[] = {7.5, 15.0, 25.0, 35.0, 45.0, 55.0,
      65.0, 75.0 };
  double p8370_d1x1y1_xerrminus[] = {2.5, 5.0, 5.0, 5.0, 5.0, 5.0,
      5.0, 5.0 };
  double p8370_d1x1y1_xerrplus[] = {2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0,
       5.0 };
  double p8370_d1x1y1_yval[] = {gammascale*(4.0E-6), gammascale*(-3.0E
      -6), gammascale*(-1.4E-5), gammascale*(-2.6E-5), gammascale*(-3.9
      E-5), gammascale*(2.2E-5), gammascale*(2.49E-4), gammascale
      *(0.001059) };
  double p8370_d1x1y1_yerrminus[] = {gammascale*(5.0E-6), gammascale
      *(5.0E-6), gammascale*(7.0E-6), gammascale*(1.1E-5), gammascale
      *(1.8E-5), gammascale*(3.6E-5), gammascale*(8.7E-5), gammascale
      *(2.79E-4) };
  double p8370_d1x1y1_yerrplus[] = {gammascale*(5.0E-6), gammascale
      *(5.0E-6), gammascale*(7.0E-6), gammascale*(1.1E-5), gammascale
      *(1.8E-5), gammascale*(3.6E-5), gammascale*(8.7E-5), gammascale
      *(2.79E-4) };
  double p8370_d1x1y1_ystatminus[] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
      0.0, 0.0 };
  double p8370_d1x1y1_ystatplus[] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
      0.0, 0.0 };
  int p8370_d1x1y1_numpoints = 8;

  gammascale = -0.05;

  double p8370_d1x1y2_xval[] = {7.5, 15.0, 25.0, 35.0, 45.0, 55.0,
      65.0, 75.0 };
  double p8370_d1x1y2_xerrminus[] = {2.5, 5.0, 5.0, 5.0, 5.0, 5.0,
      5.0, 5.0 };
  double p8370_d1x1y2_xerrplus[] = {2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0,
       5.0 };
  double p8370_d1x1y2_yval[] = {gammascale*(-2.3E-5), gammascale*(-5.5
      E-5), gammascale*(-1.06E-4), gammascale*(-1.66E-4), gammascale
      *(-2.76E-4), gammascale*(-3.79E-4), gammascale*(-4.25E-4),
      gammascale*(-2.93E-4) };
  double p8370_d1x1y2_yerrminus[] = {gammascale*(3.0E-6), gammascale
      *(3.0E-6), gammascale*(5.0E-6), gammascale*(8.0E-6), gammascale
      *(1.3E-5), gammascale*(2.6E-5), gammascale*(6.3E-5), gammascale
      *(1.96E-4) };
  double p8370_d1x1y2_yerrplus[] = {gammascale*(3.0E-6), gammascale
      *(3.0E-6), gammascale*(5.0E-6), gammascale*(8.0E-6), gammascale
      *(1.3E-5), gammascale*(2.6E-5), gammascale*(6.3E-5), gammascale
      *(1.96E-4) };
  double p8370_d1x1y2_ystatminus[] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
      0.0, 0.0 };
```

```
  double p8370_d1x1y2_ystatplus[] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
      0.0, 0.0 };
  int p8370_d1x1y2_numpoints = 8;

  gammascale = 0.1;

  double g_xval[] = {7.5, 15.0, 25.0, 35.0, 45.0, 55.0, 65.0, 75.0 };
  double g_xerrminus[] = { 2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0 };
  double g_xerrplus[] = { 2.5, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0 };
  double g_yval[] = { gammascale*(4.0E-6+2.3E-5), gammascale*(-3.0E
      -6+5.5E-5), gammascale*(-1.4E-5+1.06E-4), gammascale*(-2.6E
      -5+1.66E-4), gammascale*(-3.9E-5+2.76E-4), gammascale*(2.2E
      -5+3.79E-4), gammascale*(2.49E-4+4.25E-4), gammascale
      *(0.001059+2.93E-4)};
  double g_yerrminus[] = {  gammascale*(5.0E-6+3.0E-6), gammascale
      *(5.0E-6+3.0E-6), gammascale*(7.0E-6+5.0E-6), gammascale*(1.1E
      -5+8.0E-6), gammascale*(1.8E-5+1.3E-5), gammascale*(3.6E-5+2.6E
      -5), gammascale*(8.7E-5+6.3E-5), gammascale*(2.79E-4+1.96E-4)};
  double g_yerrplus[] = {  gammascale*(5.0E-6+3.0E-6), gammascale*(5.0
      E-6+3.0E-6), gammascale*(7.0E-6+5.0E-6), gammascale*(1.1E-5+8.0E
      -6), gammascale*(1.8E-5+1.3E-5), gammascale*(3.6E-5+2.6E-5),
      gammascale*(8.7E-5+6.3E-5), gammascale*(2.79E-4+1.96E-4)};
  double g_ystatminus[] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
      0.0 };
  double g_ystatplus[] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
       };
  int g_numpoints = 8;

switch (choise) {
case 0: // Opposite charge
        spec = new TGraphAsymmErrors(p8370_d1x1y1_numpoints,
            p8370_d1x1y1_xval, p8370_d1x1y1_yval,
            p8370_d1x1y1_xerrminus, p8370_d1x1y1_xerrplus,
            p8370_d1x1y1_yerrminus, p8370_d1x1y1_yerrplus);
        //spec.SetName("/HepData/8370/d1x1y1");
        //spec.SetTitle("/HepData/8370/d1x1y1");
break;
case 1: // Same Charge
        spec = new TGraphAsymmErrors(p8370_d1x1y2_numpoints,
            p8370_d1x1y2_xval, p8370_d1x1y2_yval,
            p8370_d1x1y2_xerrminus, p8370_d1x1y2_xerrplus,
            p8370_d1x1y2_yerrminus, p8370_d1x1y2_yerrplus);
        //spec.SetName("/HepData/8370/d1x1y2");
        //spec.SetTitle("/HepData/8370/d1x1y2");
break;
case 2: // gamma
   spec = new TGraphAsymmErrors(g_numpoints, g_xval, g_yval,
       g_xerrminus, g_xerrplus, g_yerrminus, g_yerrplus);
        //spec.SetName("/HepData/8370/d1");
        //spec.SetTitle("/HepData/8370/d1");
break;
}
return spec;
```

```
}

//=========================================================

Double_t BosPhi(Double_t phi, Double_t Rx)
{
 Double_t phis = TMath::ATan(Rx*Rx*TMath::Tan(phi));
 if(phi>PI/2. && phi<PI*3./2.)     phis += PI;
 else if(phi<2*PI && phi>PI*3./2.) phis += 2*PI;
 return phis;
}

//=========================================================

Double_t RadSys(Double_t rad, Double_t phi, Double_t Rx)
{
 return sqrt(pow(rad*cos(phi)/Rx,2.)+pow(rad*sin(phi),2.));
}
```