$\phi(1020)$ and $K^{*0}(892)$ production in proton-proton collisions at $\sqrt{s} = 7$ TeV

A thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of

MASTER OF SCIENCE

by

Himangshu Neog

to the

School of Physical Sciences

National Institute of Science Education and Research

DECLARATION

I hereby declare that I am the sole author of this thesis in partial fulfillment of the requirements for a postgraduate degree from National Institute of Science Education and Research (NISER). I authorize NISER to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Signature of the Student
Date:

The thesis work reported in the thesis entitled $\phi(1020)$ and $K^{*0}(892)$ production in proton- proton collisions at $\sqrt{s} = 7$ TeV was carried out under my supervision, in the school of Physical Sciences at NISER, Bhubaneswar, India.

Signature of the thesis supervisor
School:
Date:

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deep sense of gratitude to my thesis supervisor Dr. Bedangadas Mohanty for his invaluable guidance, encouragement and his constant support quiteessential for the successful completion of this thesis.I am grateful to him for giving me the opportunity to work on this interesting topic for my M.Sc thesis.

I would also like to express my thanks to Ranbir Singh (Scientific Officer, NISER), Ajay Kumar Dash (Research Associate, NISER) and Sourav Kundu (Phd, NISER) for providing healthy physics discussions and helping me on each and every step of my analysis work.

I gratefully acknowledge Prof. V Chandrashekar (Director of NISER) for providing the necessary infrastructure to carry out research and INSPIRE for my fellowship at NISER.

Last but most importantly, I would like to thank my parents and all other family members for the understanding and support for the entire period.

Himangshu Neog

April, 2016

National Institute of Science Education and Research

ABSTRACT

The differential yield ($d^2N/dydp_T$), mean p_T and p_T-integrated (dN/dy) yield of $\phi(1020)$ and K^{*0} (892) at midrapidity($\mid y \mid < 0.5$) in pp collisions at $\sqrt{s} = 7$ TeV is presented in this work. The measured p_T spectra for both the resonances are compared with QCD-based event generator, PYTHIA 6.4. Invariant mass and width values for each transverse momentum bin from the two data sets (experimental measurement and event generator) are compared in each case.

Contents

List of Figures

Chapter 1

Introduction

The elementary particles constituting matter and their fundamental interactions are well represented by the *Standard Model* of particle physics. In the Standard Model, the strong force is described by Quantum Chromodynamics(QCD). QCD describes the interactions between quarks and gluons, which bind together to form hadrons. The two main features of QCD are confinement and asymptotic freedom, and it has one parameter, the strong interaction coupling constant, α_s, given by

$$\alpha_s(Q^2) \approx \frac{12\pi}{(33 - (2N_f)ln(Q^2/\bigwedge^2_{QCD}))} \tag{1.1}$$

where Q^2 is the momentum transfer, N_f is the number of quark flavors and Λ_{QCD} is the scale parameter.

For small momentum transfer or at large distance, the value of α_s is very high and increases with increase in distance between quarks. This property, known as confinement, is responsible for binding of quarks inside the hadrons. On the other hand, when momentum transfers are large i.e distance between quarks are very small, the coupling between quarks are very small and quarks behaves like nearly free or weakly interacting particles. This is known as asymptotic freedom [1, 2].

1.1 Quark-Gluon Plasma: An Overview

In 1975, Collins and Perry suggested that the dense nuclear matter at the center of neutron stars could consist of deconfined quarks and gluons [3]. The lattice QCD calculations later predicted a phase transition from a confined hadronic matter to a

deconfined state of quarks and gluons at a transition temperature $T_c \sim 150MeV$ and energy density $\epsilon \sim 1GeV/fm^3$. Such a state of matter containing deconfined quarks and gluon is called the Quark-Gluon Plasma (QGP), which is believed to have existed a few microseconds after Big Bang.

1.2 Proton-proton collisions and resonances

The primary motivation for heavy-ion collisions at ultrarelativistic energies is because it is believed that it is possible to create macroscopic volumes of nuclear matter at such extreme conditions of temperature and/or energy density that a phase transition will occur from a confined hadronic matter to a plasma of deconfined quarks and gluons. But, in pp collisions, production of strange particle is strongly forbidden due to higher mass of the s quark and small system size. So the yield of strange mesons and baryons has been predicted to be strongly enhanced in presence of QGP as compared to the purely hadronic nature at the same temperature. In pp collisions, we don't expect any QGP medium to be formed due to small system size, so pp collisions are used as a baseline for heavy ion collisions.

Protons consist of quarks bound by gluons, and in a head-on collision between two protons, it is the constituent quarks and gluons that collide. When protons collide with such large energies as at the LHC, the collision results in a shower of all types of particles, the ones usual matter is made of, and others that only existed just after the Big Bang. The strategy for figuring out whats going on in a collision is to stand back and observe all products that comes out of the collision vertex and then we work backwards given what we see coming out, what must have been produced during the collision.

Resonances are extremely short lived particles. The lifetime of these particles is of

the order of 10^{-23} seconds. Traveling at the speed of light, these particles could only travel about 10^{-15} meters, or about the diameter of a proton, before decaying into other particles. Then, How do we detect them or infer their existence? The answer lies in the fact that the cross-section of the colliding particles changes as a function of the total energy in the collision. Most collisions have several possible outcomes, and each possibility has peak cross-sections at certain energies. In one view, the peaks themselves are resonant states or resonances. In another view, the peaks are evidence for actual particles that form as intermediate steps in the collision.

1.3 ϕ meson properties

ϕ is a vector meson made up of a strange quark(s) and a strange antiquark ($\overline{s}$). ϕ meson is a resonant particle which provides a probe for the study of strangeness production. Since ϕ meson is unstable, it can only be detected from its decay product of either the kaon-antikaon pair or the lepton pair. In this work, we used the kaon-antikaon pair productions for reconstructing the ϕ meson signal.

ϕ Meson Properties:

Mass: 1019.455 ± 0.020 MeV

Width: 4.26 ± 0.04 MeV

Decay Mode:(Studied in this work) K^+K^- $(49.2 \pm 0.6\%)$

1.4 K^{*0} properties

K^{*0} is another vector meson made up of a down quark(d) and a strange antiquark ($\overline{s}$). It is another useful probe for strangeness production study as it has a mass similar

to ϕ but differ by one unit of strangeness quantum number. In this work, we used the kaon-pion pair productions for reconstructing the K^{*0} signal.

K^{*0} **Properties:**

Mass: 896.00 ± 0.25 MeV

Width: 47.79 ± 0.86 MeV

Decay Mode:(Studied in this work) $K^{\pm}\pi^{\mp}$ (66.66%)

1.5 p_T spectra

In p-p collisions, the transverse momentum (p_T) spectra of identified particles is very useful in extracting initial conditions of the collision. Transverse momentum is defined as:

$$p_T = \sqrt{{p_x}^2 + {p_y}^2} \tag{1.2}$$

where p_x and p_y are the x and y component of momentum p. The p_T spectra is used to fine tuning of QCD based event generators. The mean p_T values give an indirect information about enhancement of hard or soft processes at a given center of mass energy.

Probabilities of nuclear reactions is expressed by the effective area of collision called as cross-section. Cross-sections are used to describe total yields of reactions regardless of energies of emitted particles or of their spatial distribution. It turns out that the product of energy, E and triple differential cross-section $d^3\sigma/dP^3$ is a Lorentz invariant quantity. Now, this product can be rewritten as

$$\sigma_{inv} = \frac{d^3\sigma}{dP_x dP_y dP_z/E} = \frac{d^3\sigma}{dP_x dP_y dy} \tag{1.3}$$

The p_T spectra is studied from the cross section defined as $E\frac{d^3N}{dp^3}$ which is invariant under Lorentz transformations.

In polar co-ordinates of the P_x, P_y plane, this can be written as

$$\sigma_{inv} = \frac{d^3\sigma}{p_T dp_T dy d\phi} \tag{1.4}$$

For an azimuthally isotropic system, integration over ϕ angles leads to

$$\sigma_{inv} = \frac{d^2\sigma}{2\pi p_T dp_T dy} \tag{1.5}$$

In practice, this quantity is determined by measuring the number of particles, d^2N emitted per unit p_T per unit rapidity (y) which is calculated as:

$$E\frac{d^2N}{dp^3} = \frac{1}{2\pi p_T}\frac{d^2N}{dp_T dy} \tag{1.6}$$

where rapidity is defined as

$$y = \frac{1}{2}ln(\frac{E+p_z}{E-p_z}) \tag{1.7}$$

where E and p_z are the total energy and z-component of momentum of the particle. In the situation where we do not know the mass of the particle, usually pseudorapidity (η) is used and is defined as

$$\eta = -ln(\tan(\frac{\theta}{2})) \tag{1.8}$$

where θ is the angle between the particle three-momentum p and the positive direction of the beam axis.

Chapter 2

ALICE Experiment

2.1 Experimental setup

In the ALICE (A Large Ion Collider Experiment) experiment, particles are identified by the characteristic signatures they leave in the detector. The experiment is divided into a few main components and each component tests a specific set of particle properties. These components are stacked in layers and the particles go through the layers sequentially from the collision point outwards: first a tracking system, then an electromagnetic (EM) and a hadronic calorimeter and finally a muon system. The detectors are embedded in a magnetic field in order to bend the tracks of charged particles for momentum and charge determination. This method for particle identification works well only for certain particles, and is used for example by the large LHC experiments ATLAS (A Toroidal LHC Apparatus) and CMS (Compact Muon Solenoid).

The central-barrel detectors (ITS, TPC, TRD, TOF, PHOS, EMCal, and HMPID; some of them discussed later) are embedded in a solenoid and address particle production at midrapidity. For the analyses described in this work, the ITS, the TPC, and the TOF detectors are used. These detectors are set inside a large solenoidal magnet providing a magnetic field B= 0.5T, and have a common pseudorapidity coverage of $\mid \eta \mid < 0.9$. Two forward scintillator hodoscopes (VZERO) placed along the beam direction at -0.9 m and 3.3 m on either side of the interaction point, cover the pseudorapidity regions -3.7 $< \eta <$ -1.7 and 2.8 $< \eta <$ 5.1. The three detectors are described below.

2.1.1 The Inner Tracking System (ITS)

The ITS [5] aims at identifying the short-living heavy particles by measuring the location where they occur with a precision of a tenth of millimetre. The two innermost layers of ITS detector is located at radii of 3.9 cm and 43 cm with pseudorapidity coverage of $\mid \eta \mid < 2.0$ and $\mid \eta \mid < 1.4$ respectively. It is made of six cylindrical layers of silicon detectors (two layers of pixels, two of silicon drift, and two of silicon strips), with a total material budget of 7.66% of the radiation length X_0 where radiation length of a material is defined as the mean length traversed by the high energy electrons to reduce its energy by a factor of 1/e.

2.1.2 The Time Projection Chamber (TPC)

The Time Projection Chamber [6] is a large volume filled with a gas as detection medium and considered to be the main particle tracking and particle identification device in ALICE. It is a high-granularity, cylindrical drift detector with a length of 5.1m and inner and outer radii of 0.85 m and 2.47 m, respectively. It covers the pseudorapidity range $\mid \eta \mid < 0.9$ with a full azimuthal acceptance. The drift volume is filled with 90 m^3 of Ne/CO_2 /N_2 . The maximum drift time is 94 μs. A total of 72 multi-wire proportional chambers with cathode pad readout instrument the two end plates, which are segmented into 18 sectors and include a total of over $550,000$ readout pads. The ionization electrons drift for up to 2.5 m and are measured on 159 pad rows. The ALICE TPC ReadOut (ALTRO) chip, employs a 10 bit ADC at 10 MHz sampling rate and digital filtering circuits which allows precise position and linear energy loss measurements with a gas gain of the order of 10^4. Measuring the deflection in the magnetic field, the ITS and TPC are able to determine the

momentum of charged particles.

2.1.3 The Time Of Flight (TOF)

Charged particles are identified in ALICE by Time-Of-Flight measurements. The TOF [7, 8] is a cylindrical assembly of Multi-gap Resistive Plate Chambers (MRPC) with an inner radius of 370 cm and an outer radius of 399 cm. MRPCs are parallel-plate detectors built of thin sheets of standard window glass to create narrow gas gaps with high electric fields. These plates are separated using fishing lines to provide the desired spacing; 10 gas gaps per MRPC are needed to arrive at a detection efficiency close to 100%. The ALICE TOF has a pseudorapidity coverage of $\mid \eta \mid < 0.9$ and full azimuthal acceptance.

2.2 ALICE data specifications

The data set used for the analysis are from pp collisions at $\sqrt{s} = 7$ TeV collected in 2010. This data set is taken with a minimum bias trigger, which requires a single hit in the SPD detector or in one of the two VZERO counters, i.e. at least one charged particle anywhere within the pseudorapidity covered by these detectors. A coincidence is also required with signals from two beam pick-up counters, one on each side of the interaction region.

During the data-taking period, the luminosity at the ALICE interaction point was kept in the range $0.6 - 1.2$ x 10^{29} cm^2 s^{-1} . Runs with a mean pile-up probability per event larger than 5% were excluded from the analysis [10].

2.3 Pythia 6.4

Pythia program [4] is used for event generation in high-energy physics. The emphasis is on hard interactions as in e^+e^-, pp and ep colliders, although also other applications

are envisaged. The program is intended to generate complete events comparable to experimentally observed ones, within the bounds of our current understanding of the underlying physics.

Many versions of Pythia are available. The analysis in this work is done with Pythia 6.4. 20 million events are generated with initialization parameter as Center of mass energy $\sqrt{s}$ =7 TeV. The generated ascii file is analysed with ROOT Data Analysis software package [9].

Chapter 3

Data Set And Quality Assurance

For the analysis of data from ALICE, several cuts are applied to achieve a high track quality in the analyzed sample as discussed below.

3.1 Event selection

An event refers to a outcome of an interaction between two colliding particles, occuring in a very short time span. Accepted events are required to have a reconstructed primary vertex. Its position can be calculated by using the tracks reconstructed by TPC and ITS.

3.1.1 Charged particle multiplicity

The charged-particle multiplicity is a key observable for the understanding of multi-particle production in collisions of hadrons at high-energy. The probability for producing multiple charged particles in the final state is related to the production mechanism of the particles. For a particle source without any correlations the multiplicity distribution follows a Poisson distribution. Any kind of deviation is manifested as a deviation from Poisson distribution.

The charge particle multiplicities for the analysed data is plotted in Fig. 3.1 to get an idea of any embedded correlations.

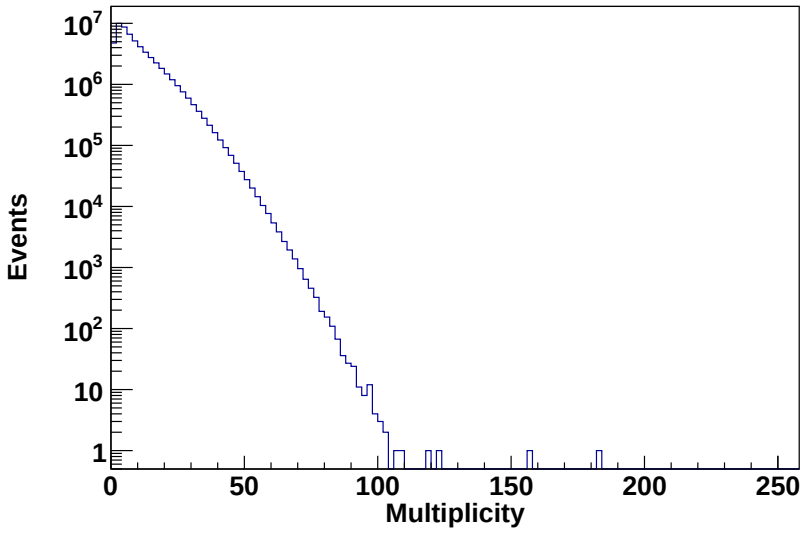

Figure 3.1: Charged particle multiplicity in pp collision at $\sqrt{s} = 7$ TeV for pseudorapidity range $\mid \eta \mid < 0.8$

3.1.2 Vertex distribution along z-direction

To minimize acceptance and efficiency biases for tracks at the edge of the TPC detection volume, events are accepted (standard 2010 cut) only when their primary vertex is within ± 10 cm from the geometrical centre of the ALICE barrel. The z-vertex distribution is shown in Fig. 3.2.

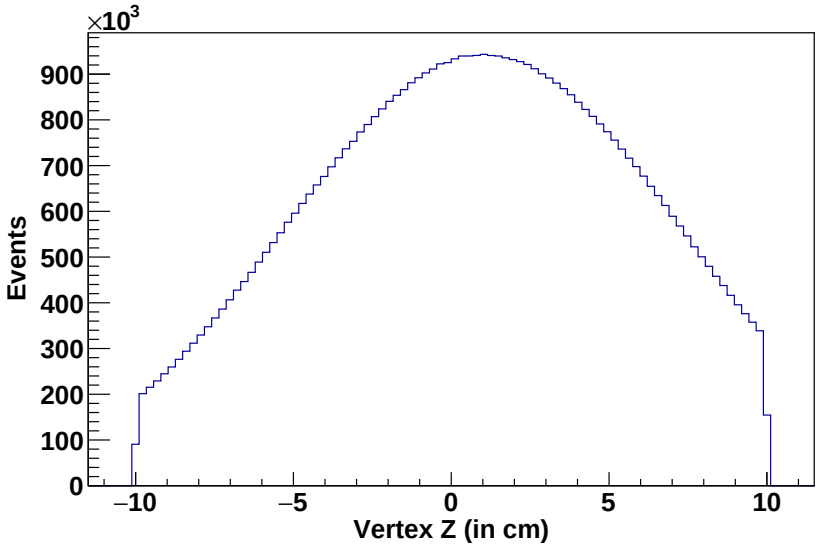

Figure 3.2: Vertex distribution along z direction of selected kaons after standard cut.

3.2 Track Selection

In order to ensure optimal particle identification and momentum resolution, various quality cuts are given to the tracks before they are considered for analysis. The track quality cuts for ϕ and K^{*0} study are as follows:

3.2.1 DCA_Z distribution

The Distance of Closest Approach (DCA) to the primary vertex is used to discriminate between primary and secondary particles. Primary charged particles are those produced directly in the interaction and all decay products from particles with a proper decay length $c\tau < 1$ cm. In order to reduce secondary particles, tracks are selected with DCA in the z-direction to the primary vertex less than 2 cm along the beam direction as shown in Fig. 3.3.

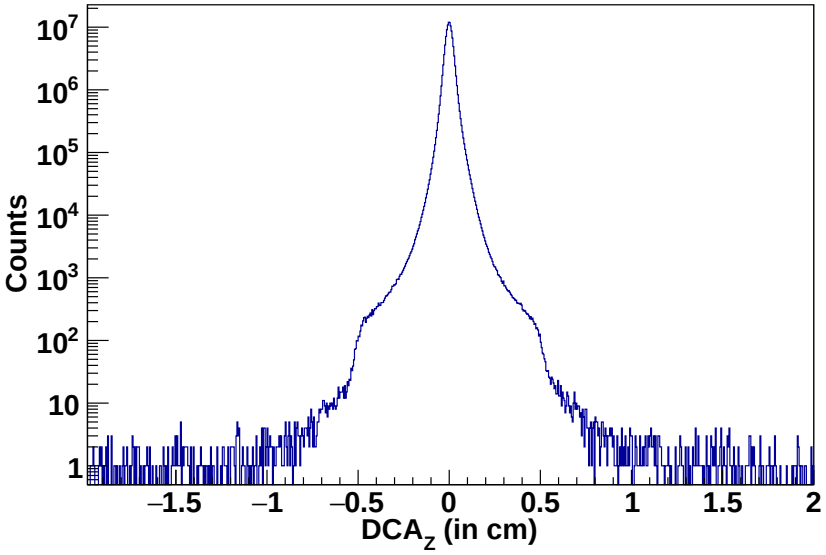

Figure 3.3: The selected tracks DCA distribution in longitudinal(z) direction.

3.2.2 DCA_{XY} distribution

The DCA in the transverse plane is selected to be smaller than 7 σ where 1 σ = $0.0026 + 0.005 * (1/{p_T}^{1.01})$ where p_T is the transverse momentum (Fig. 3.4).

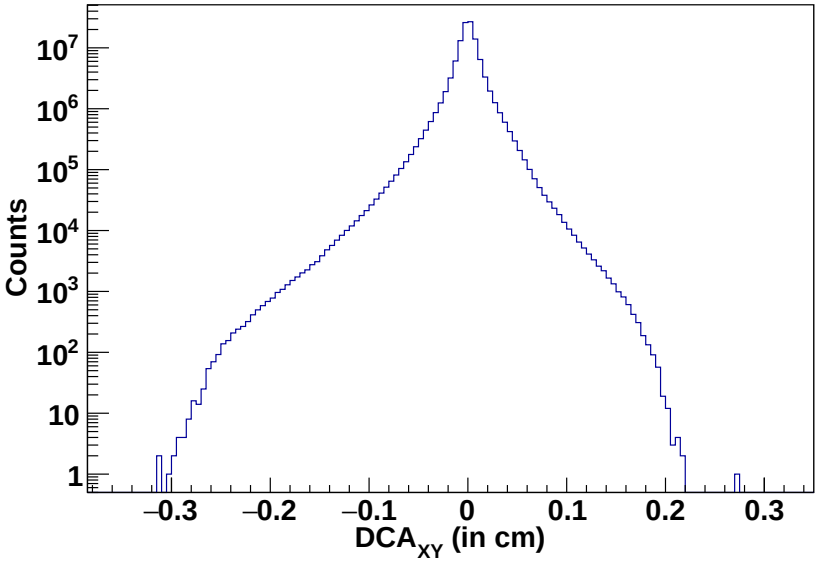

Figure 3.4: The selected tracks DCA distribution in transverse(xy) direction.

3.2.3 φ distribution

The φ distribution gives an idea about the acceptance of the detectors. Fig. 3.5 shows the azimuthal angle distribution of selected kaons an pions.

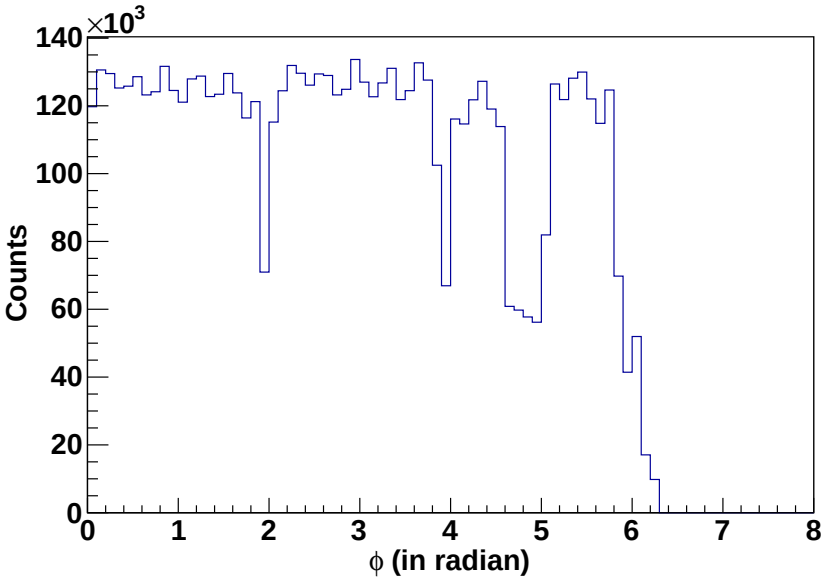

Figure 3.5: Azimuthal angle (φ) distribution of selected kaons and pions.

As evident from Fig. 3.5 , the dip in φ value in the range of $4-5$ radians implies a lower acceptance of the detector at these φ range.

3.2.4 η distribution

Pseudorapidity(η) is a mathematical function of the polar angle of the particle with the beam axis. It is essentially the rapidity in the limit of massless particles. Fig. 3.6 shows the η distribution for selected kaons and pions in the ranges from $-0.8-0.8$ which means that we are considering particles which move perpendicular to the beam. The dip at $\eta = 0$ refers to the transformation of η to y. The kaons and pions are selected with pseudorapidity cut $\mid \eta \mid < 0.8$ because TPC has full tracking in this range.

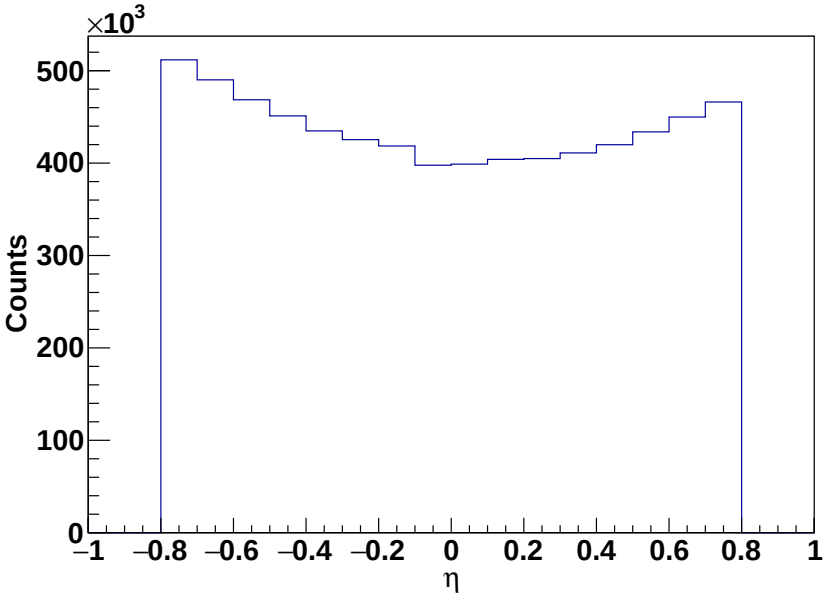

Figure 3.6: Pseudorapidity (η) distribution of selected tracks.

3.2.5 TPC crossed rows

Tracks were selected which have at least 70 reconstructed clusters in the TPC out of the maximum 159 available. This is done by the TPC Crossed rows cut of 70. A typical TPC crossed row distribution is shown in Fig 3.7.

3.2.6 TPC crossed rows ratio with findable cluster

This variable can be viewed proportional to the effectively sampled track length of a particle in the TPC. TPC Crossed rows ratio with findable cluster is set to 0.8 as shown in Fig. 3.8.

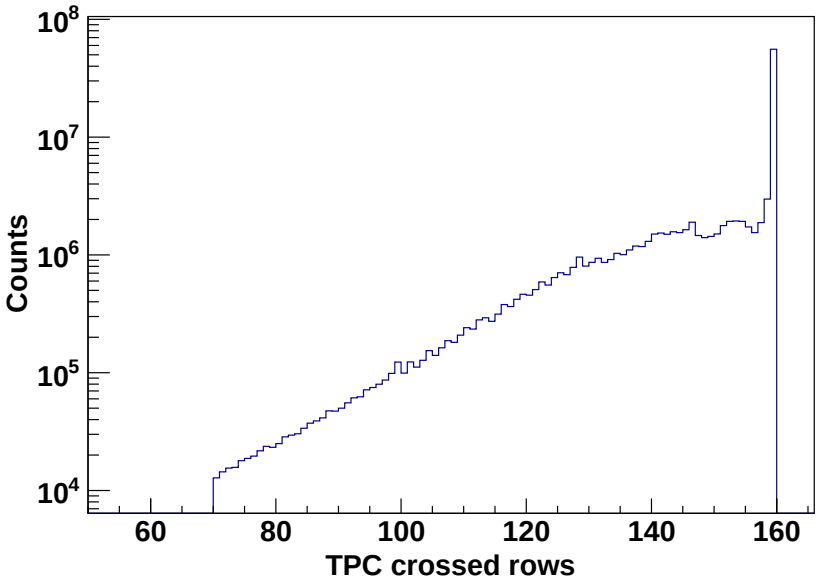

Figure 3.7: The distribution of TPC crossed rows after track selection

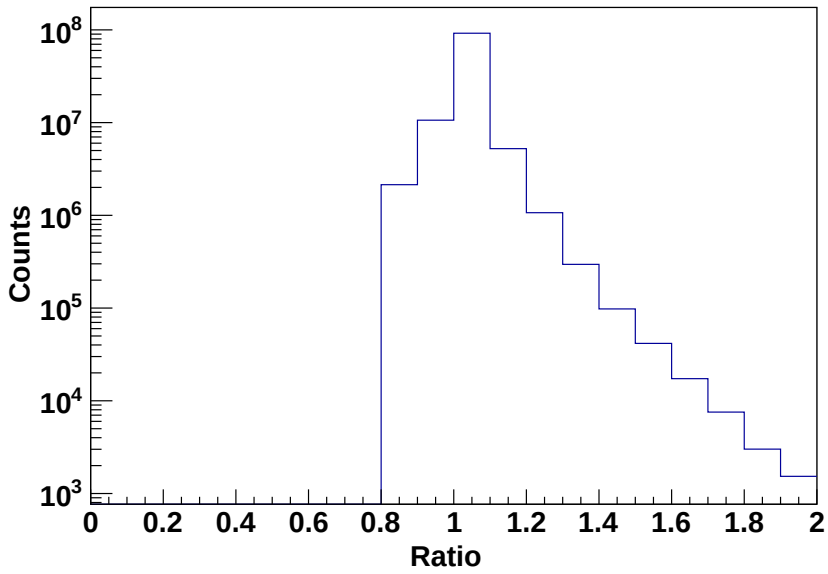

Figure 3.8: The distribution of TPC crossed rows ratio with findable cluster after track selection

3.2.7 TPC dE/dx distribution

Particles are identified in the TPC based on the energy they deposit in the drift gas, compared with the expected value computed using a parameterized Bethe-Bloch function.The specific ionization energy loss dE/dx as a function of momentum of the tracks for the selected kaon and pion tracks(using Nσ_K and Nσ_K cut)is shown in Fig. 3.9.

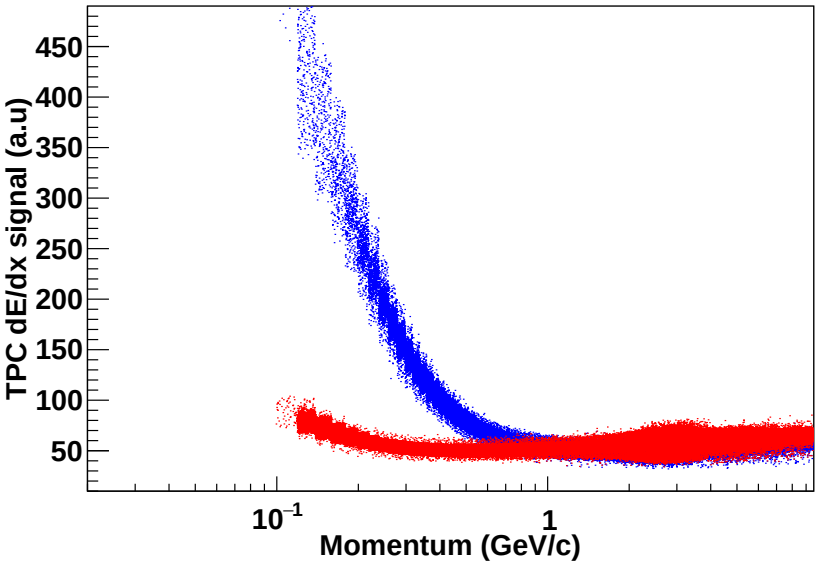

Figure 3.9: dE/dx as a function of momentum for charged particles in TPC. Red and blue bands are shown for selected pion and kaon candidates, respectively.

3.2.8 Nσ distribution

Particles are identified in the TPC via the difference between the measured energy loss and the value expected for different mass hypotheses. The Nσ_K and Nσ_π cut takes care of the limit for the difference between the measured and expected values. The Nσ distribution for charged kaons and pions is shown in Fig. 3.10 and Fig. 3.11 respectively.

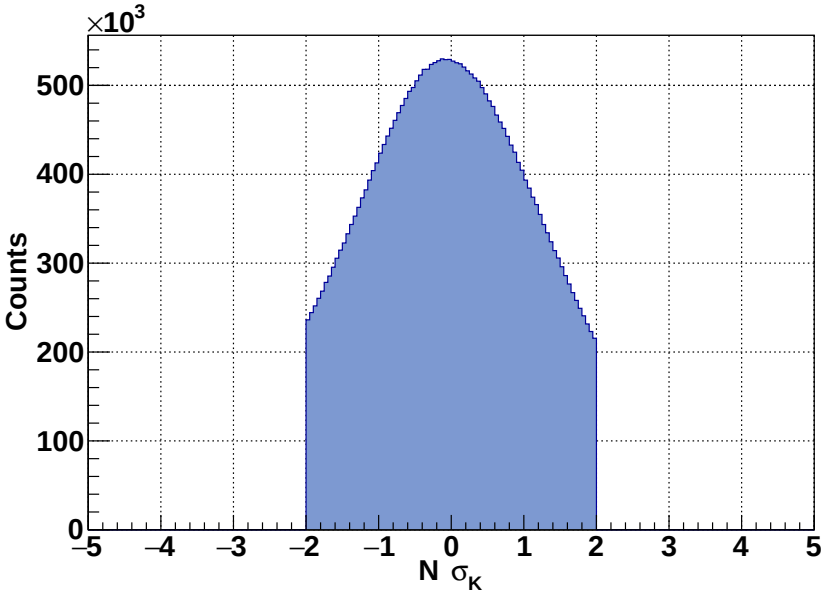

Figure 3.10: Nσ_K distribution of selected kaons.

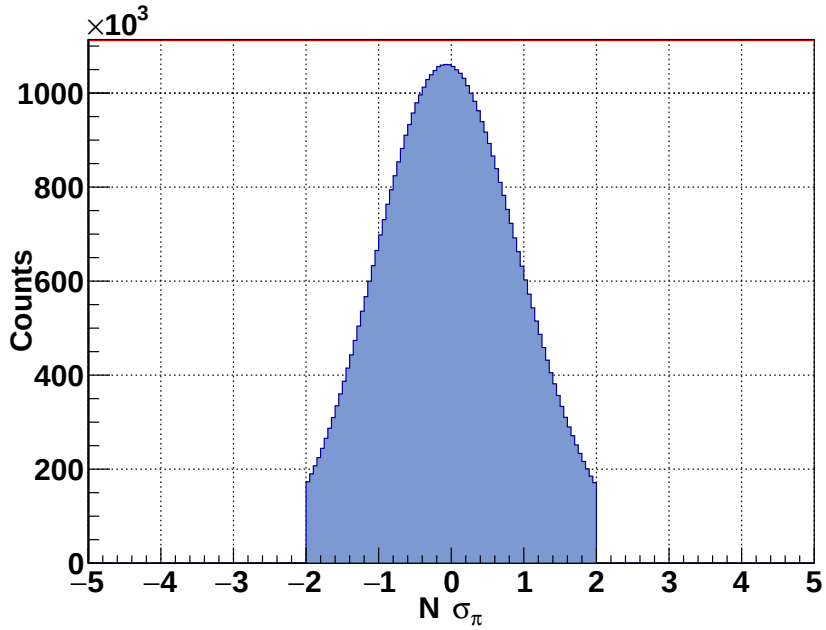

Figure 3.11: Nσ_π distribution of selected pions.

Chapter 4

Analysis Technique

4.1 Resonance reconstruction

In this analysis technique, the signal of a particular resonance is fully reconstructed from the daughter particles to which the resonance has decayed. From the decay particles the invariant mass is reconstructed, and this invariant mass distribution gives a peak at the mass position of mother particle, which sits on a combinatorial background.. The reconstruction of the invariant mass works well in hadronic decays for which all final state particles are measured as charged particles in the tracking detectors.

Resonance reconstruction is used for both the data generated from Pythia 6.4 and data from ALICE.

4.2 Signal extraction

In principle the signal can be extracted by fitting the invariant mass (signal + combinatorial background) distribution with appropriate functions chosen in order to provide a good description of the overall distribution. This technique, however, does not work if the signal and background have a similar shape. In order to extract the signal, we subtract the combinatorial background which is reconstructed by using standard techniques like Like-sign method and Mixed event method .

4.3 Data set and Analysis cuts

4.3.1 ALICE data set

The data set used for the analysis are from pp collisions at $\sqrt{s} = 7$ TeV collected in 2010 with 53.14 million total events. This data set is taken with a minimum bias trigger, which requires a single hit in the SPD detector or in one of the two VZERO counters, i.e. at least one charged particle anywhere in the pseudorapidity range covered by these detectors. In addition, a coincidence is required with signals from two beam pick-up counters, one on each side of the interaction region. Beam-induced background is reduced by a cut on the position of the primary vertex reconstructed in SPD. To minimize acceptance and efficiency biases for tracks at the edge of the TPC detection volume,events are accepted only when their primary vertex was within ±10 cm from the geometrical centre of the ALICE barrel.

To ensure optimal particle identification and momentum resolution, various quality cuts are given to the tracks as summarized below:

Track Quality Cuts	
$\mid DCA_Z \mid$	< 2 cm
DCA_{XY}	$< 7\sigma_{DCA}$ cm
TPC crossed rows	≥ 70
TPC crossed rows ratio with findable clusters	≥ 0.8
$\mid \eta \mid$	< 0.8
Nσ_K	± 3
Nσ_π	± 3

4.3.2 Simulated data generation

Simulated data set is generated with Pythia 6.4 with initial parameters as follows:

Number of events : 20 million

Centre of Mass energy: 7 TeV Type: p - p collisions

For analysis, we take a pair rapidity cut of $\mid y \mid \leq 0.5$ for mixing the K^+ and K^-

in case of ϕ and $K^{\pm}$ and $\pi^{\mp}$ in case of $K^{*0}\overline{K^{*0}}$ produced in a given event for both ALICE and Simulated data.

4.4 Invariant mass reconstruction

For ϕ reconstruction, we select the K^+ and K^- samples from the data by applying the various cuts as discussed above. The signal for ϕ meson is reconstructed by calculating the invariant mass for all possible K^+K^- pairs in a given event. Since not all charged kaons are originating from the ϕ meson decays, the signal of ϕ meson will be superimposed on a broad combinatorial background.

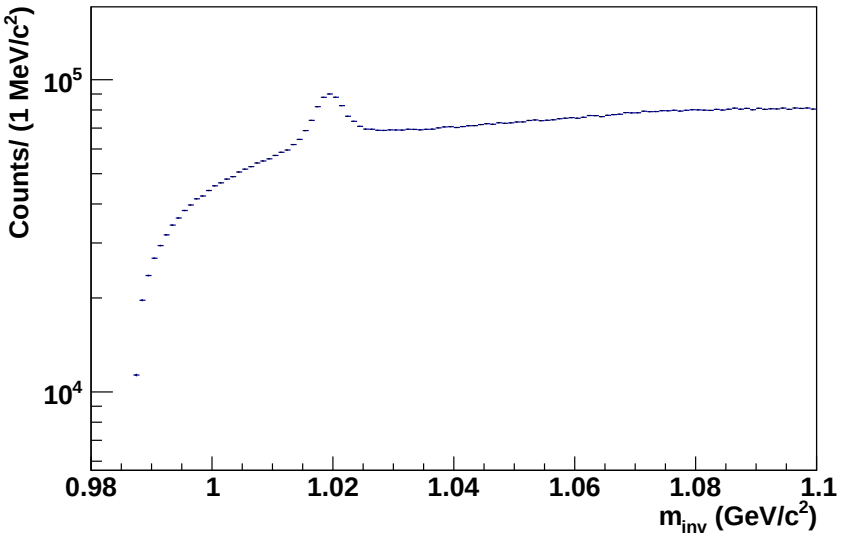

Figure 4.1: ALICE: Invariant mass distribution of K^+ and K^- in same event for $0.4 \leq p_T < 0.6$ range.

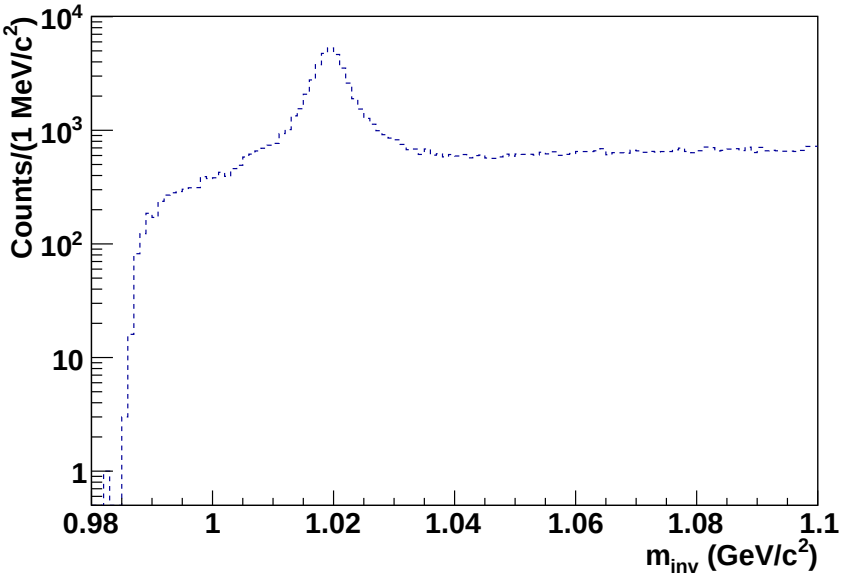

Figure 4.2: Pythia: Invariant mass distribution of K^+ and K^- in same event for $0.4 \leq p_T < 0.6$ range.

For K^{*0} reconstruction from their hadronic decay channel $K^{*0}(\overline{K^{*0}}) \rightarrow K^+\pi^-(K^-\pi^+)$, we select the $K^{\pm}$ and $\pi^{\pm}$ samples from the data by applying the various cuts as discussed above. The signal is reconstructed by calculating the invariant mass for all possible $K^{\pm}\pi^{\mp}$ pairs in an event. Throughout this report, the results for K^{*0} and $\overline{K^{*0}}$ are averaged and denoted by the symbol K^{*0} unless specified otherwise. Since not all charged kaons and pions are originating from the K^{*0} meson decays, the signal of K^{*0} meson will again be superimposed on a broad combinatorial background.

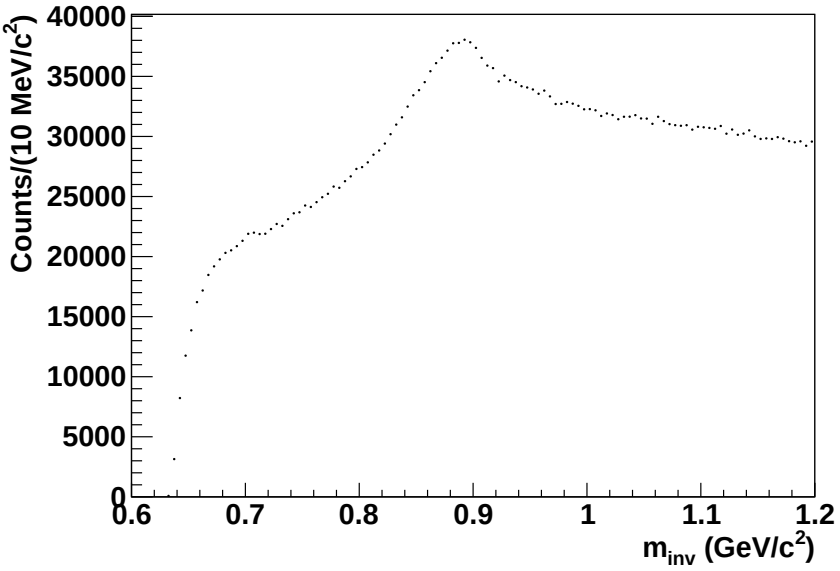

Figure 4.3: ALICE: Invariant mass distribution of $K^+(K^-)$ and $\pi^-(\pi^+)$ in same event for $0.4 \leq p_T < 0.6$ range.

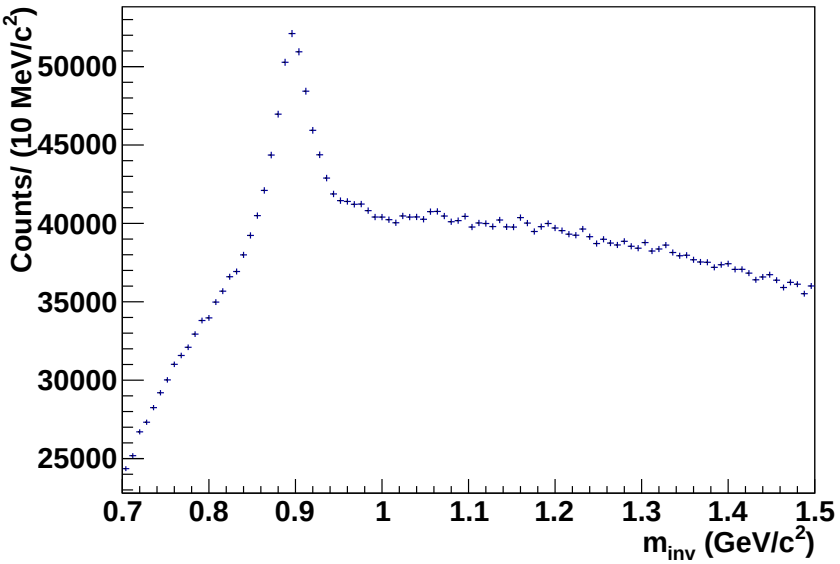

Figure 4.4: Pythia: Invariant mass distribution of $K^+(K^-)$ and $\pi^-(\pi^+)$ in same event for $0.4 \leq p_T < 0.6$ range.

4.5 Combinatorial background

Since we are constructing an unlike-sign two particle invariant mass spectrum, there are mainly two ways to determine and subtract the combinatorial background: Like Sign Pair Method and Mixing Event Method. For the case of ϕ and K^{*0} data generated from Pythia, we use Like-Sign Pair method. While for the ALICE data, we use the Event Mixing technique.

4.5.1 Event mixing method

In case of ϕ, the uncorrelated background is given by an unlike-sign combination of K^+K^- for which the two particles of a pair are taken from two different events, shown in Fig. 4.5. This offers a better statistical precision than the Like-Sign method since one can mix each event with several other events. To avoid mismatch due to different acceptances and to assure a similar event structure, tracks from events with similar vertex positions z (Δ z < 1 cm) and track multiplicities < 5 are mixed. To reduce statistical uncertainties each event is mixed with 5 other events.

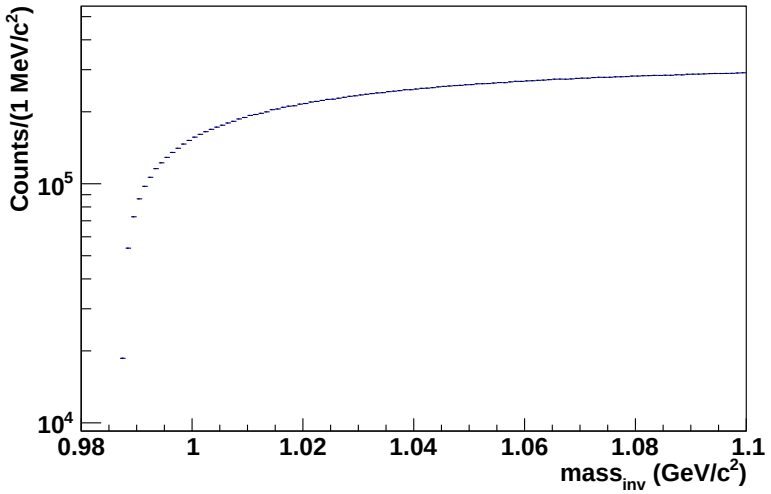

Figure 4.5: ALICE: Invariant mass distribution of K^+ K^- from different events for $0.4 \leq p_T < 0.6$ range.

In case of K^{*0}, the uncorrelated background is given by an unlike-sign combination of $K^+\pi^-(K^-\pi^+)$ for which the two particles of a pair are taken from two different events, shown in Fig. 4.6. To assure a similar event structure, tracks from events with similar vertex positions z (Δ z < 2 cm) and track multiplicities < 5 are mixed. To reduce statistical uncertainties each event is mixed with 5 other events.

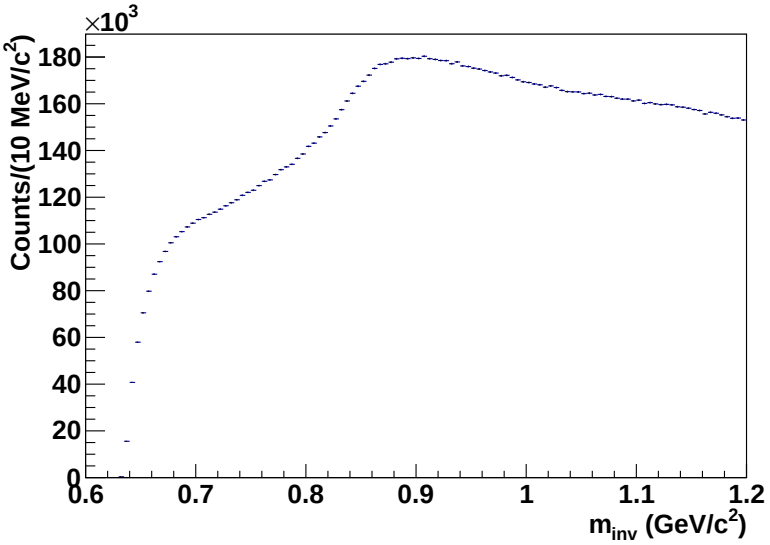

Figure 4.6: ALICE: Invariant mass distribution of $K^+(K^-)$ and $\pi^-(\pi^+)$ from different events $0.4 \leq p_T < 0.6$ range.

4.5.2 Like sign method

For the case of ϕ, we reconstructed the invariant mass for K^+ and K^+ pairs and K^- and K^- pairs from the same event, since they cannot give signal for ϕ which is shown in Fig. 4.7. Similarly, for K^{*0}, we combine like sign $(++, --)$ pairs to construct the combinatorial background as shown in Fig. 4.8. This technique has an advantage since the number of unlike-sign pairs and like-sign pairs are calculated within the same event, the normalization of the determined background to the signal+background spectrum is straightforward.

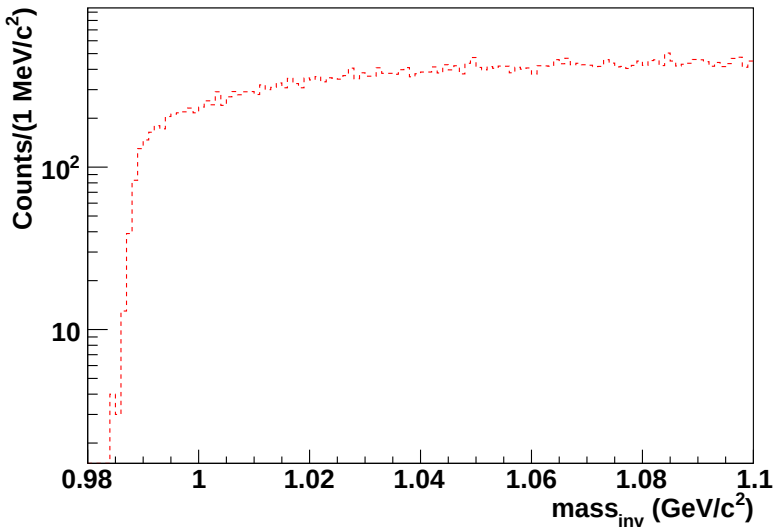

Figure 4.7: Pythia: Invariant mass distribution of K^+ K^+ and K^- K^- in same event for $0.4 \leq p_T < 0.6$ range.

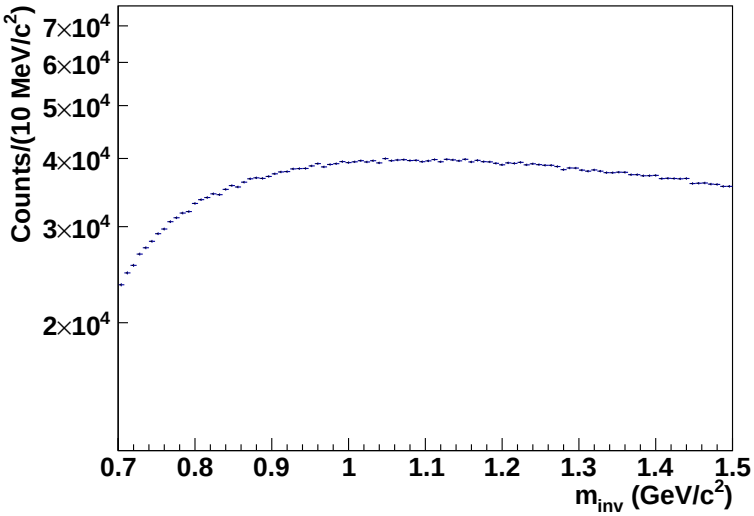

Figure 4.8: Pythia: Invariant mass distribution of K^+ π^+ and K^- π^- in same event for $0.4 \leq p_T < 0.6$ range.

4.6 Subtraction of normalized background

The background calculated with the help of above standard techniques has to be normalized to be able to be subtracted from the signal+background. The background distribution is scaled by the ratio of the integral of the signal+background to the integral of the background distribution in a fixed invariant mass region excluding the signal region. We have taken 1.04 GeV to 1.06 GeV for ϕ and 1.1 GeV to 1.5 GeV for K^{*0}as shown in Fig. 4.9, 4.10 and Fig. 4.11, 4.12 for ALICE data and simulated Pythia data respectively.

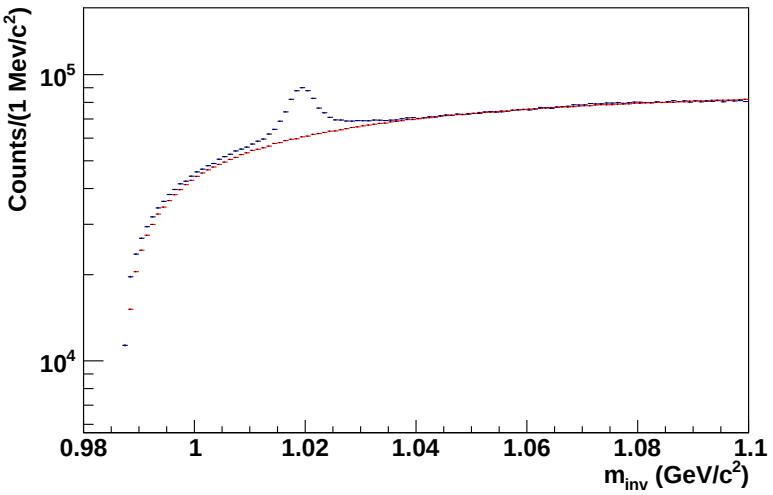

Figure 4.9: ALICE: Invariant mass distribution $m_{K^+K^-}$ (blue dots) with normalized mixed event background (red dots) for integrated p_T range.

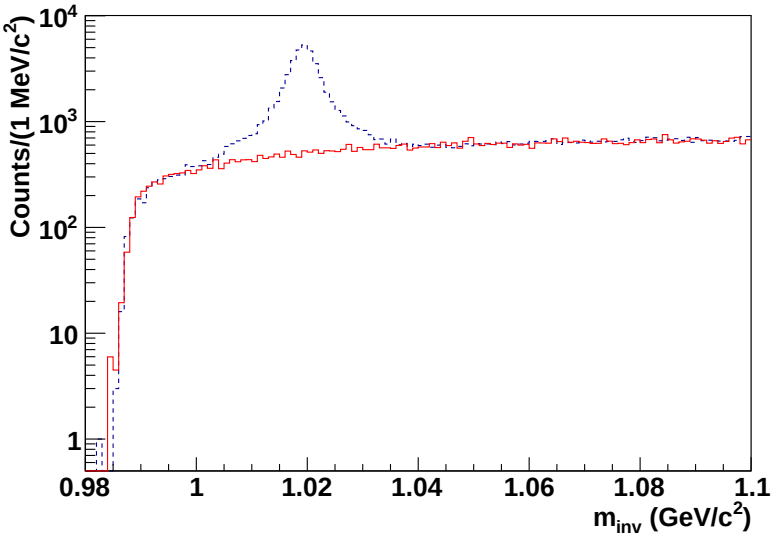

Figure 4.10: Pythia: Invariant mass distribution $m_{K^+K^-}$ (blue dots) with normalized like sign background (red dots) for integrated p_T range.

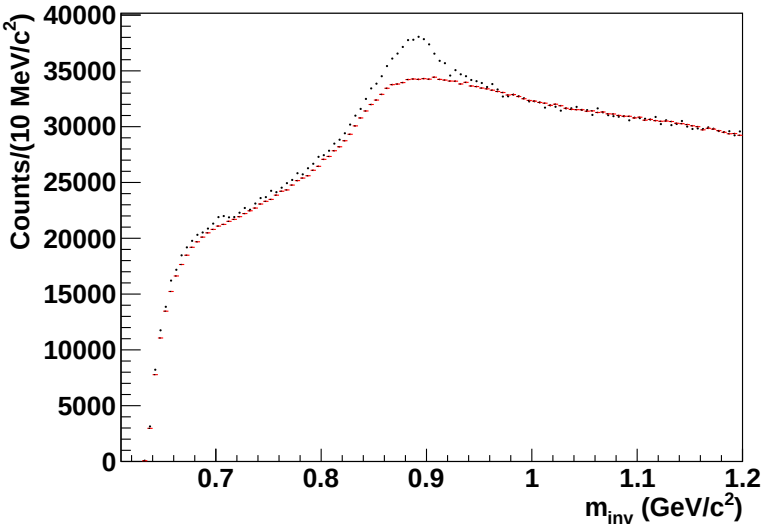

Figure 4.11: ALICE: Invariant mass distribution $m_{K^+\pi^-(K^-\pi^+)}$ with normalized mixed event background for integrated p_T range.

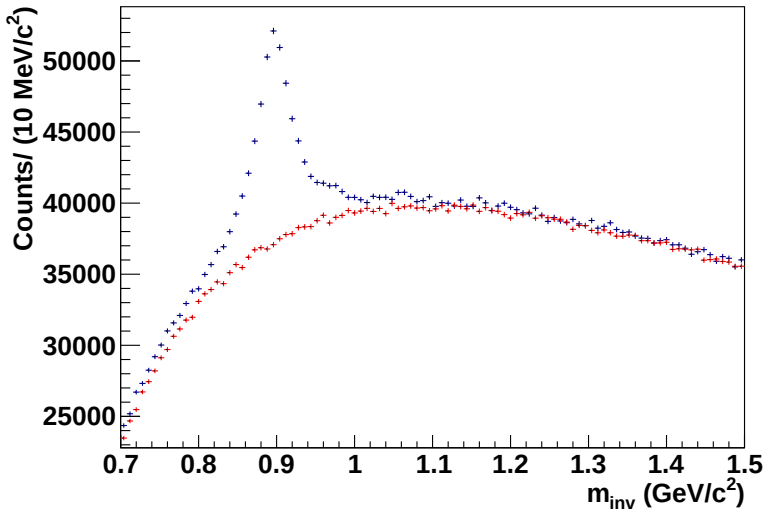

Figure 4.12: Pythia: Invariant mass distribution $m_{K^+\pi^-(K^-\pi^+)}$ with normalized like sign background for integrated p_T range.

We, now subtract the scaled background from the signal+background distribution. The plots obtained are shown in Fig. 4.13, 4.14 and Fig. 4.15, 4.16 for ALICE data and simulated Pythia data respectively.

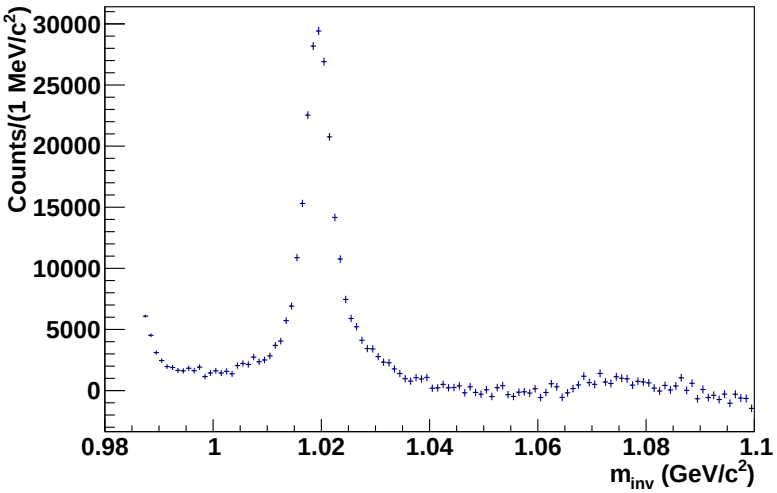

Figure 4.13: ALICE: Invariant mass distribution $m_{K^+K^-}$ after mixed event background subtraction

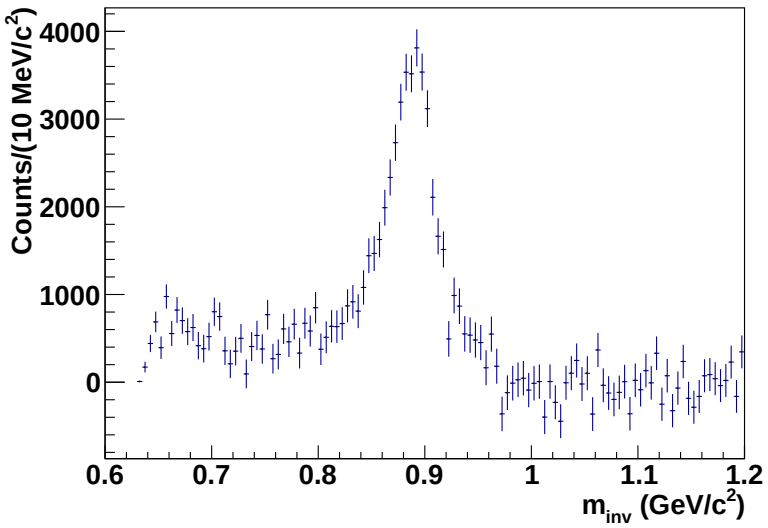

Figure 4.14: ALICE: Invariant mass distribution $m_{K^+\pi^-(K^-\pi^+)}$ after mixed event background subtraction

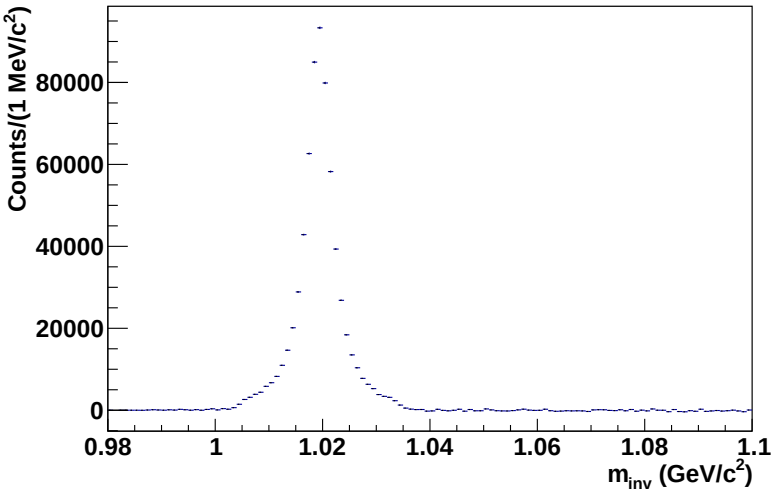

Figure 4.15: Pythia: Invariant mass distribution $m_{K^+K^-}$ after like sign background subtraction

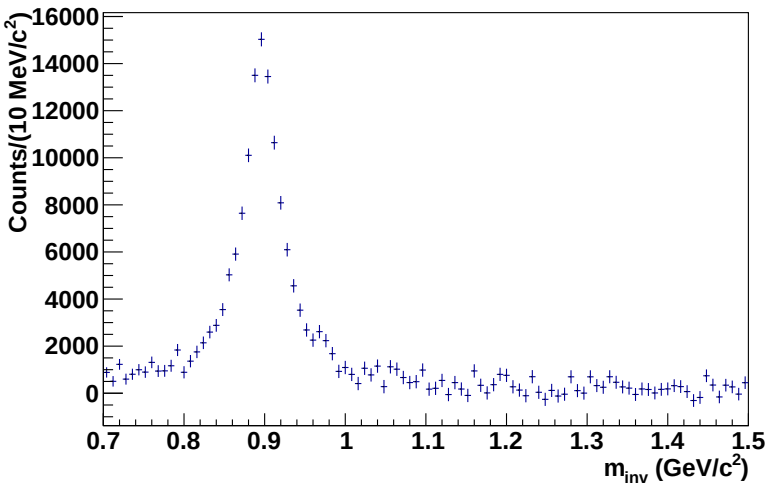

Figure 4.16: Pythia: Invariant mass distribution $m_{K^+\pi^-(K^-\pi^+)}$ after like sign background subtraction

4.7 Residual background and fitting functions

The plot obtained after above steps has the signal and a small residual background. This residual background is due to an imperfect description of the combinatorial background. The obtained signal+residual background is fitted with a Breit-Wigner function and a two degree polynomial to measure raw yield.

4.7.1 Briet-Wigner for signal

To extract the raw yield, we are using Briet-Wigner (BW) function of the form

$$\frac{dN}{dM} = \frac{1}{2\pi}\frac{A\Gamma}{(M - M_0)^2 + \Gamma^2/4} \tag{4.1}$$

where A is the area under the peak corresponding to the number of ϕ(or K^{*0}) mesons, Γ is the full width at half maximum and M_0 is the resonance mass. The parameters A, M_0 and Γ are free parameters in this fitting.

4.7.2 Polynomial function for background

The residual background (RB) is fitted with a two degree polynomial for data from ALICE and one degree polynomial for pythia generated data. It is of the form

$$B(M) = A_0 x^2 + B_0 x + C_0 \tag{4.2}$$

where A_0, B_0 and C_0 are the free parameters of the two degree polynomial and $A_0 = 0$ for one degree polynomial. The variable x here is the invariant mass of the particle.

4.8 Raw yield extraction

For ϕ meson, raw yield is extracted from the K^+K^- invariant mass distribution in 24 p_T bins ranging from 0.4 to 8 GeV/c. The plots obtained from both ALICE data and Pythia data are shown in Fig. 4.17, Fig. 4.18, Fig. 4.19, Fig. 4.20 and Fig. 4.21 for ALICE data and Fig. 4.22, Fig. 4.23, Fig. 4.24 and Fig. 4.25 for Pythia data for different p_T bins. The red line corresponds to the total fit function (BW + RB) and the blue line shows only the parameterized residual background.

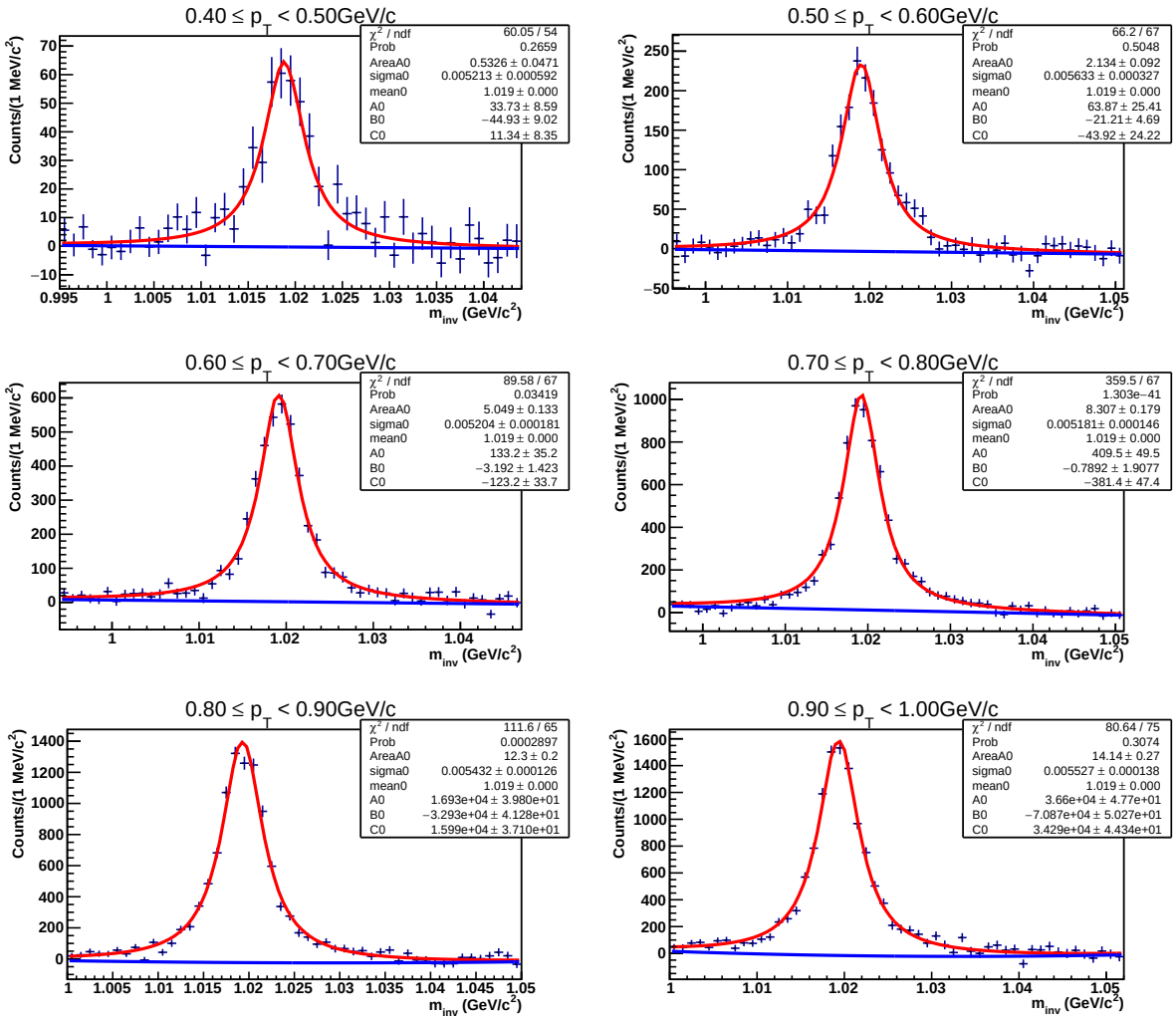

Figure 4.17: ALICE: Invariant mass distribution $m_{K^+K^-}$ fitted with BW+Poly2 (red line) in the p_T range $0.4 - 1.0$ GeV/c. The blue line shows the residual background described by Poly2.

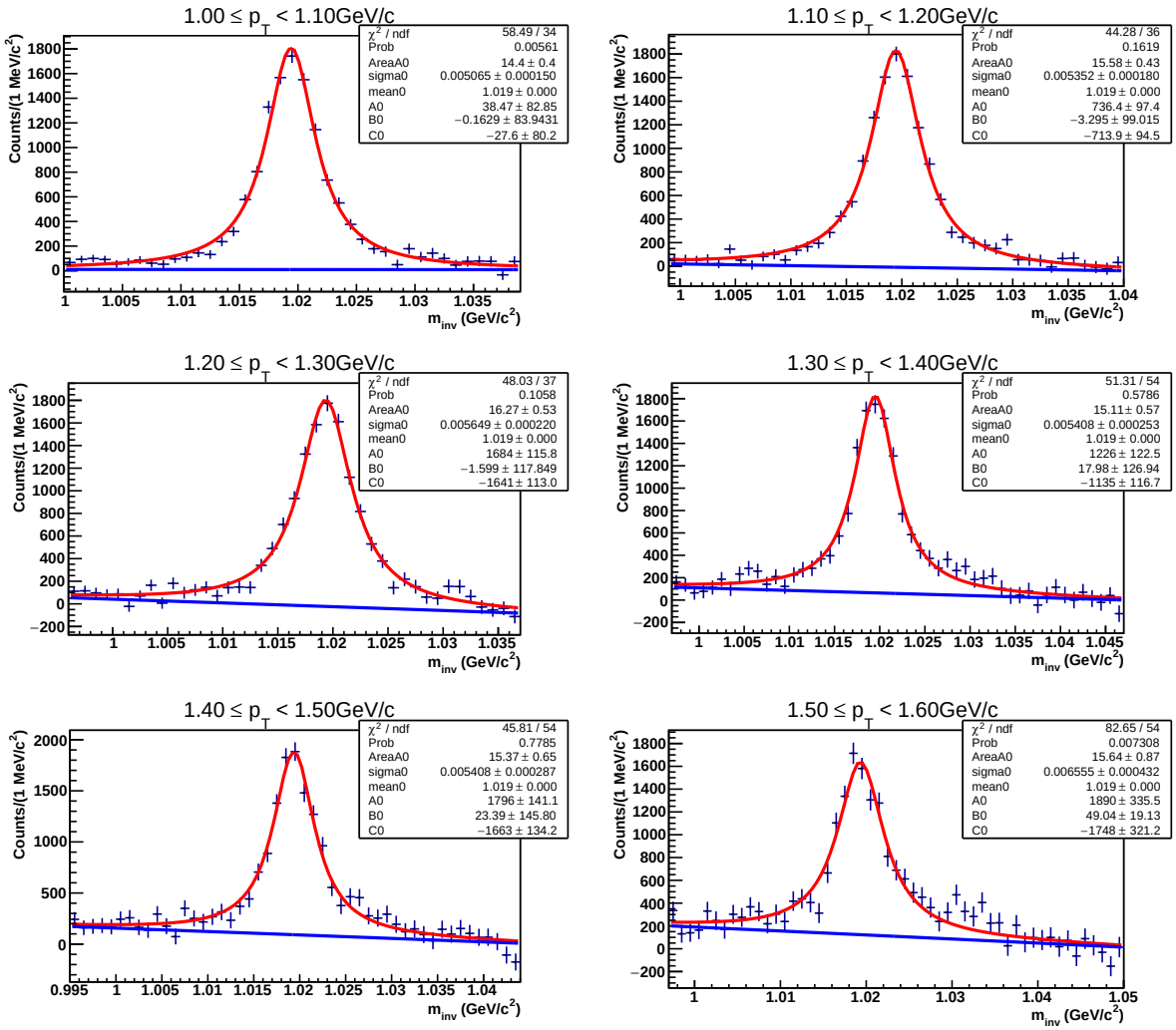

Figure 4.18: ALICE: Invariant mass distribution $m_{K^+K^-}$ fitted with BW+Poly2 (red line) in the p_T range $1.0 - 1.6$ GeV/c. The blue line shows the residual background described by Poly2.

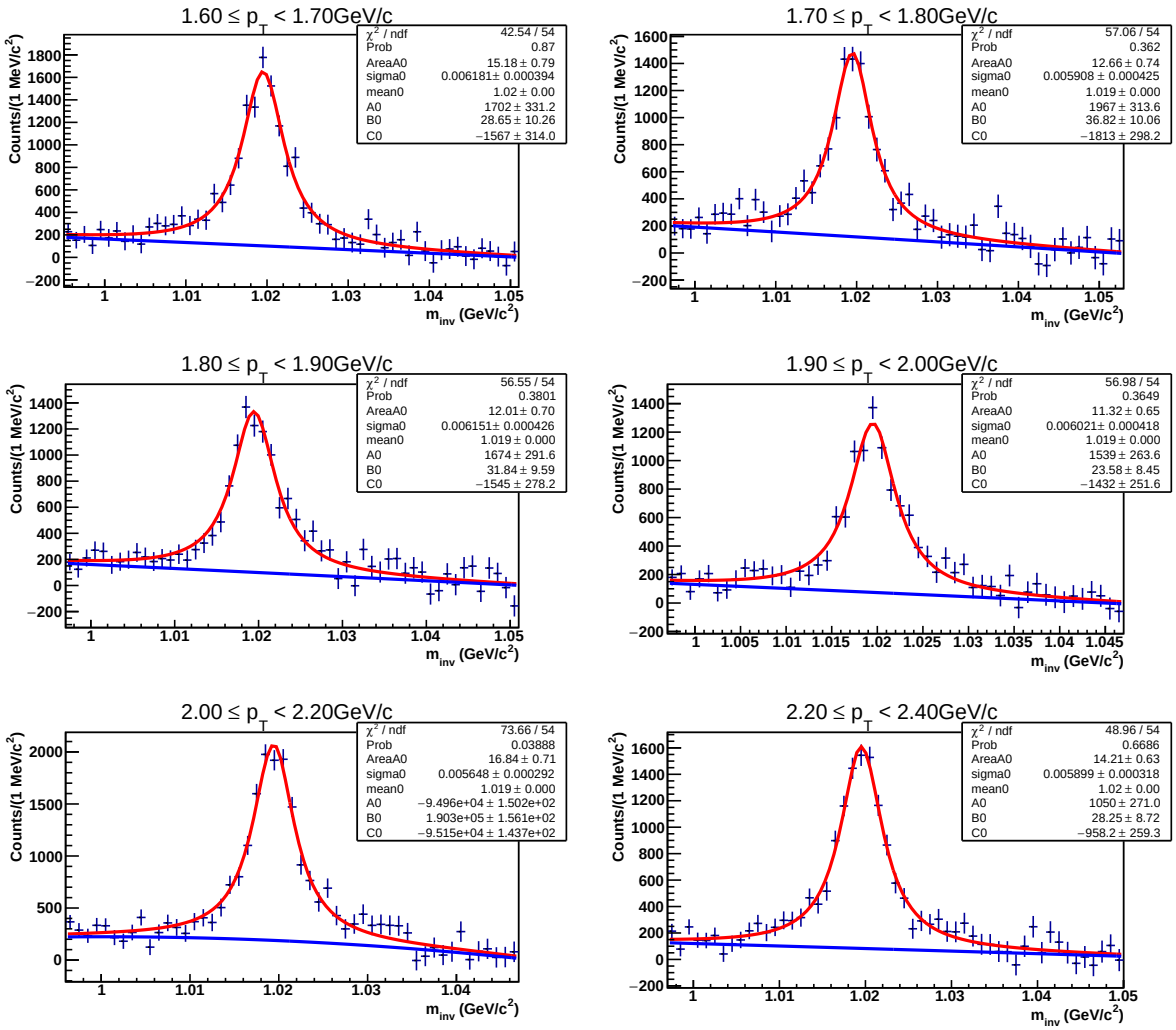

Figure 4.19: ALICE: Invariant mass distribution $m_{K^+K^-}$ fitted with BW+Poly2 (red line) in the p_T range$1.6 - 2.4$ GeV/c. The blue line shows the residual background described by Poly2.

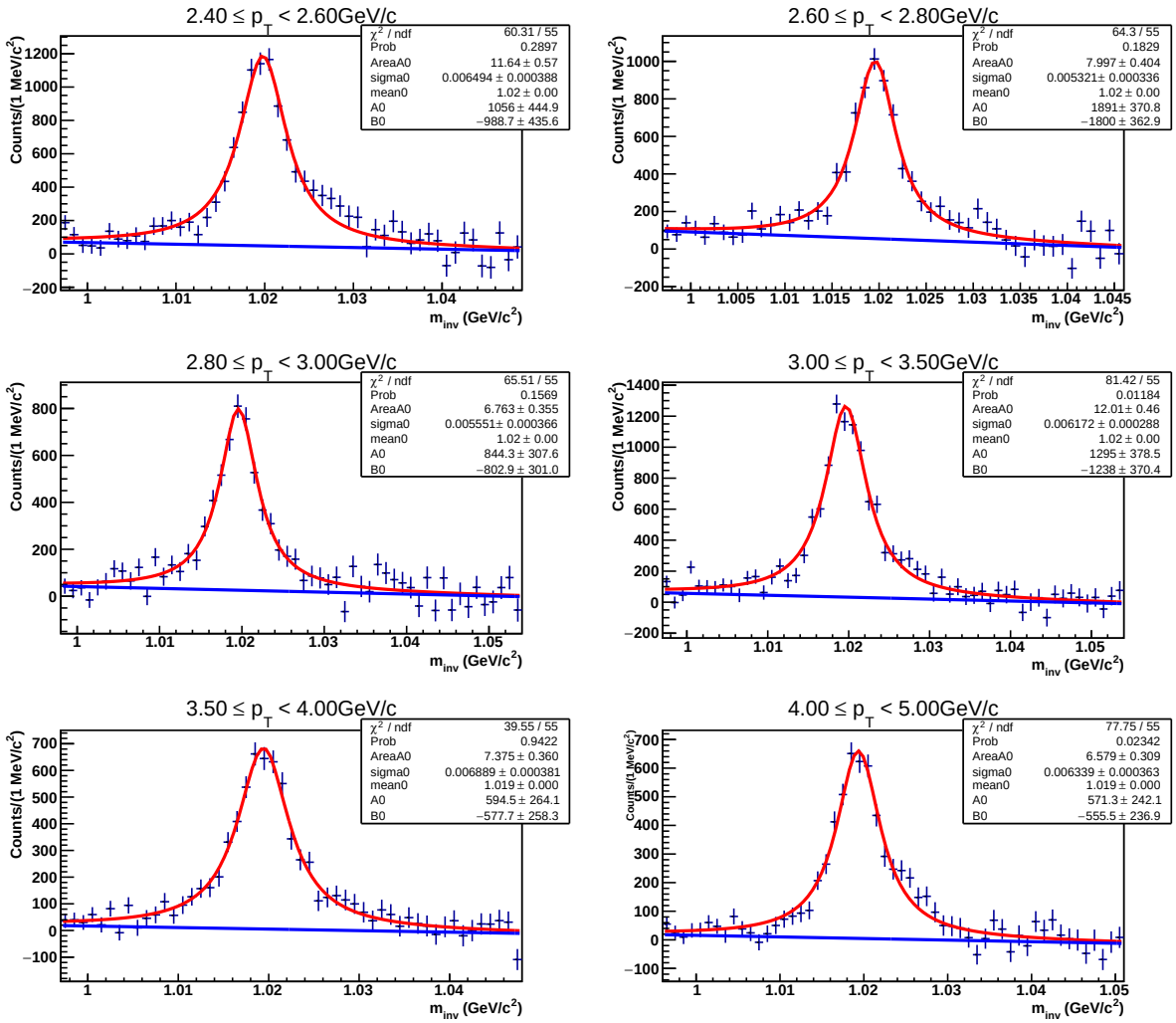

Figure 4.20: ALICE: Invariant mass distribution $m_{K^+K^-}$ fitted with BW+Poly2 (red line) in the p_T range $2.4 - 5.0$ GeV/c. The blue line shows the residual background described by Poly2.

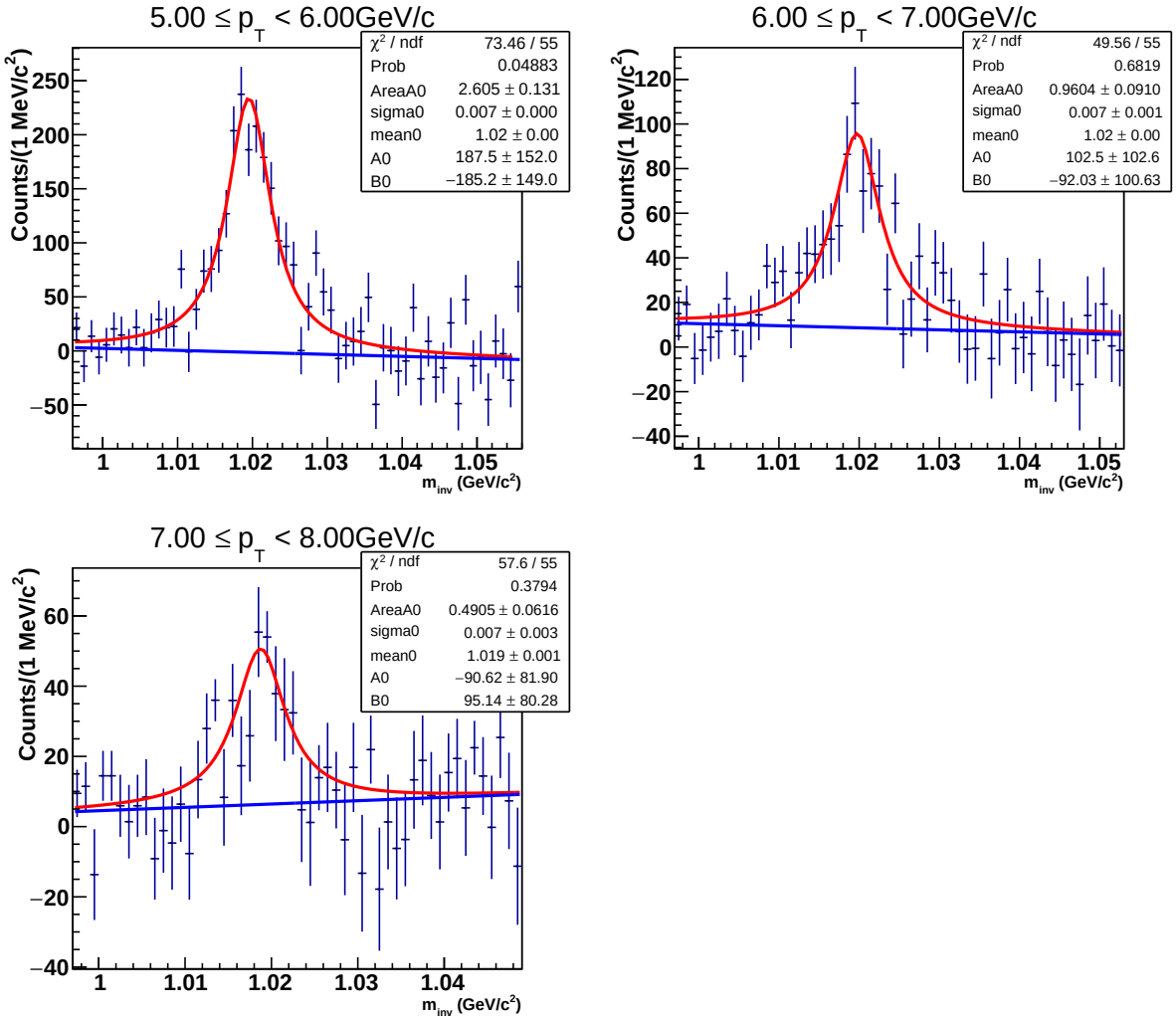

Figure 4.21: ALICE: Invariant mass distribution $m_{K^+K^-}$ fitted with BW+Poly2 (red line) in the p_T range $5.0 - 8.0$ GeV/c. The blue line shows the residual background described by Poly2.

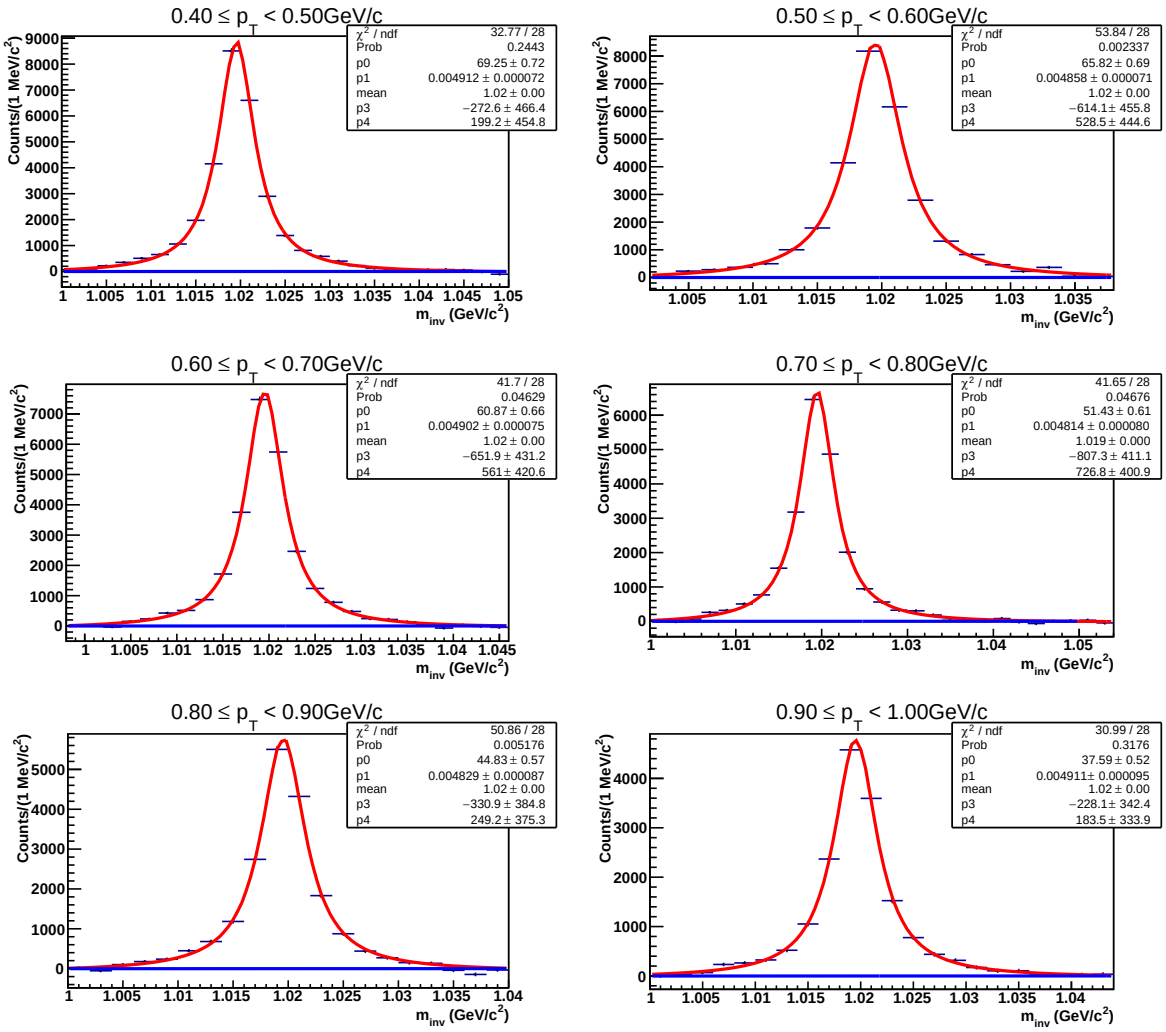

Figure 4.22: Pythia: Invariant mass distribution $m_{K^+K^-}$ fitted with BW+Poly1 (red line) in the p_T range $0.4 - 1.0$ GeV/c. The blue line shows the residual background described by Poly1.

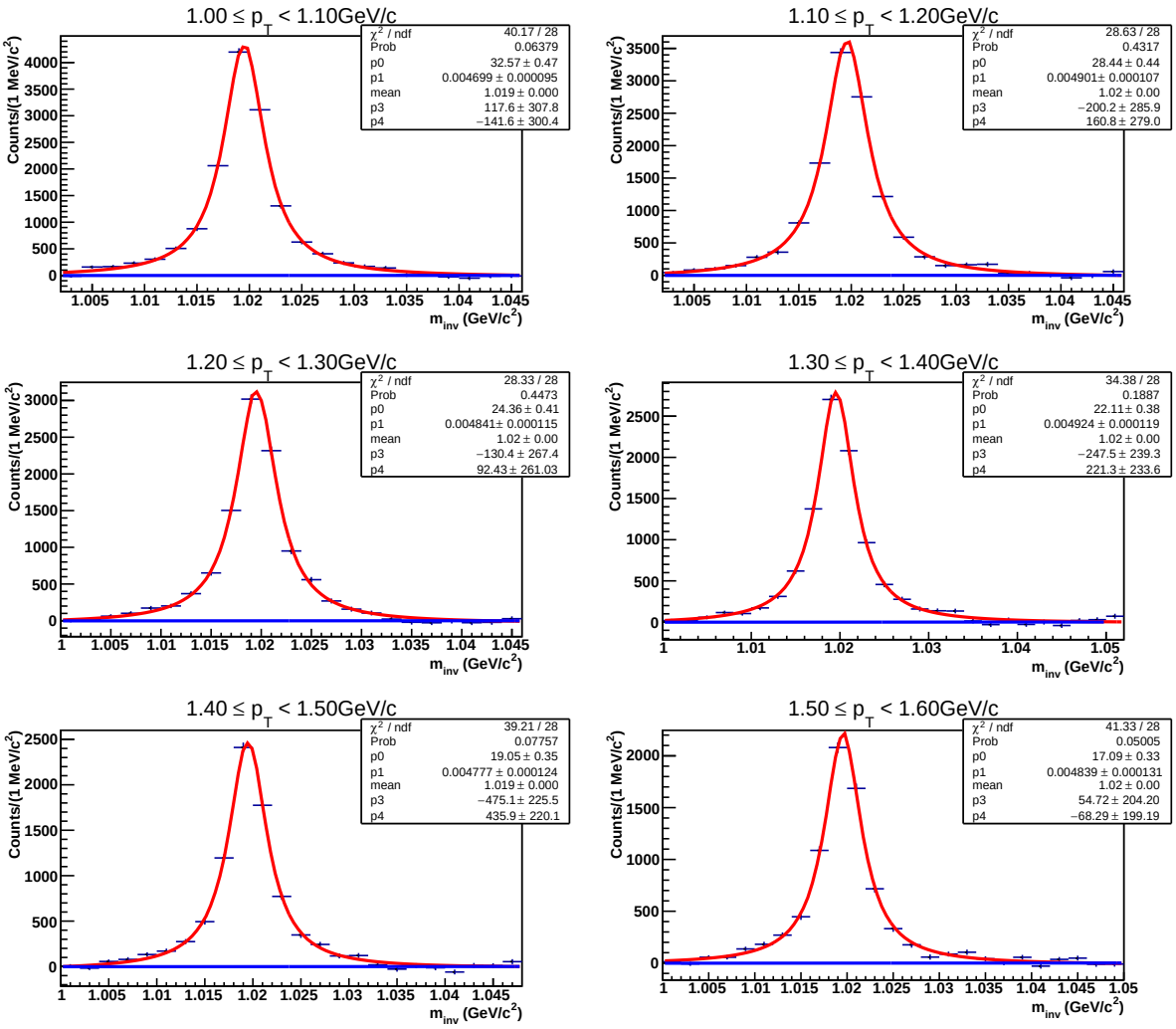

Figure 4.23: Pythia: Invariant mass distribution $m_{K^+K^-}$ fitted with BW+Poly1 (red line) in the p_T range $1.0 - 1.6$ GeV/c. The blue line shows the residual background described by Poly1.

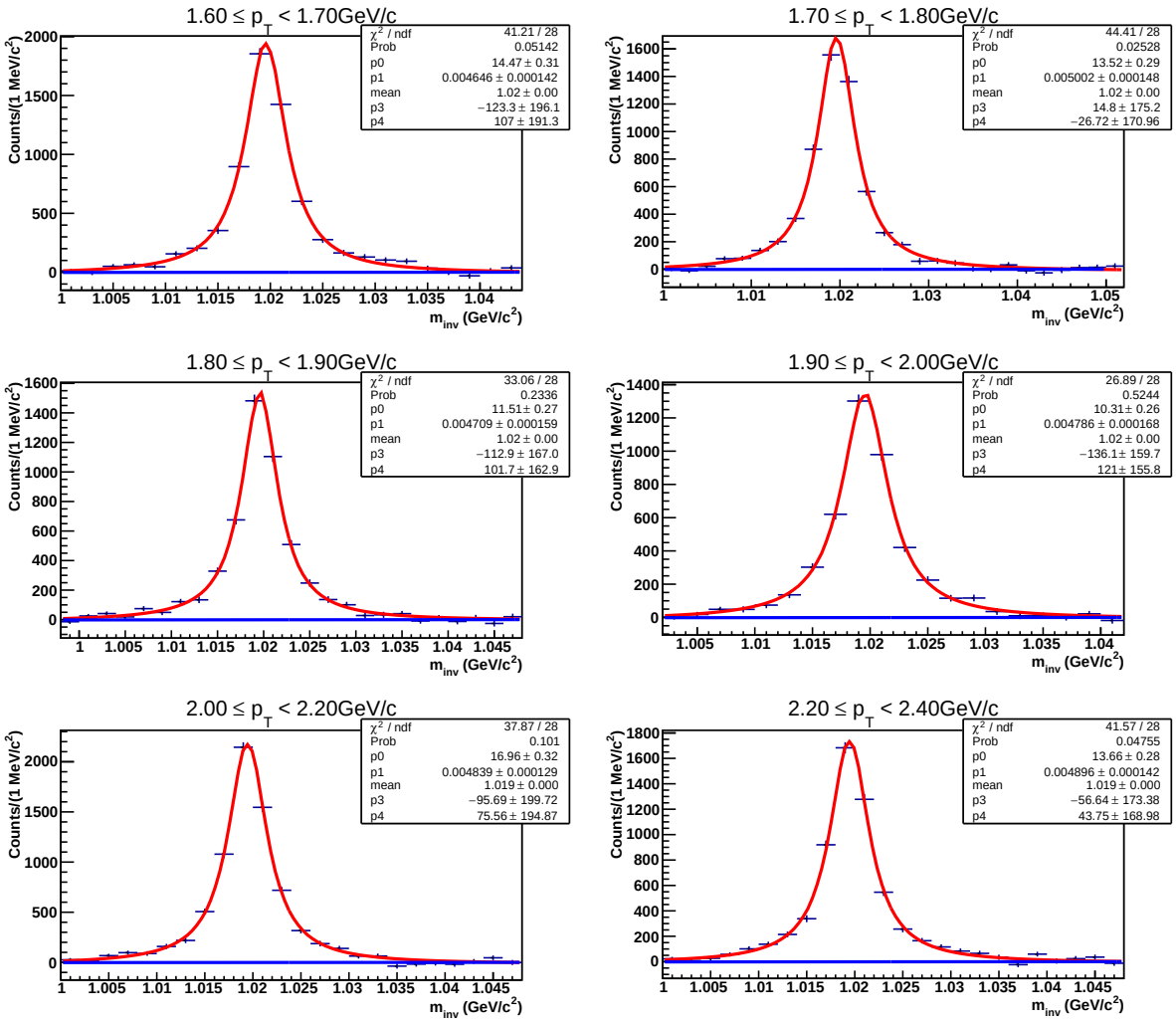

Figure 4.24: Pythia: Invariant mass distribution $m_{K^+K^-}$ fitted with BW+Poly1 (red line) in the p_T range $1.6 - 2.4$ GeV/c. The blue line shows the residual background described by Poly1.

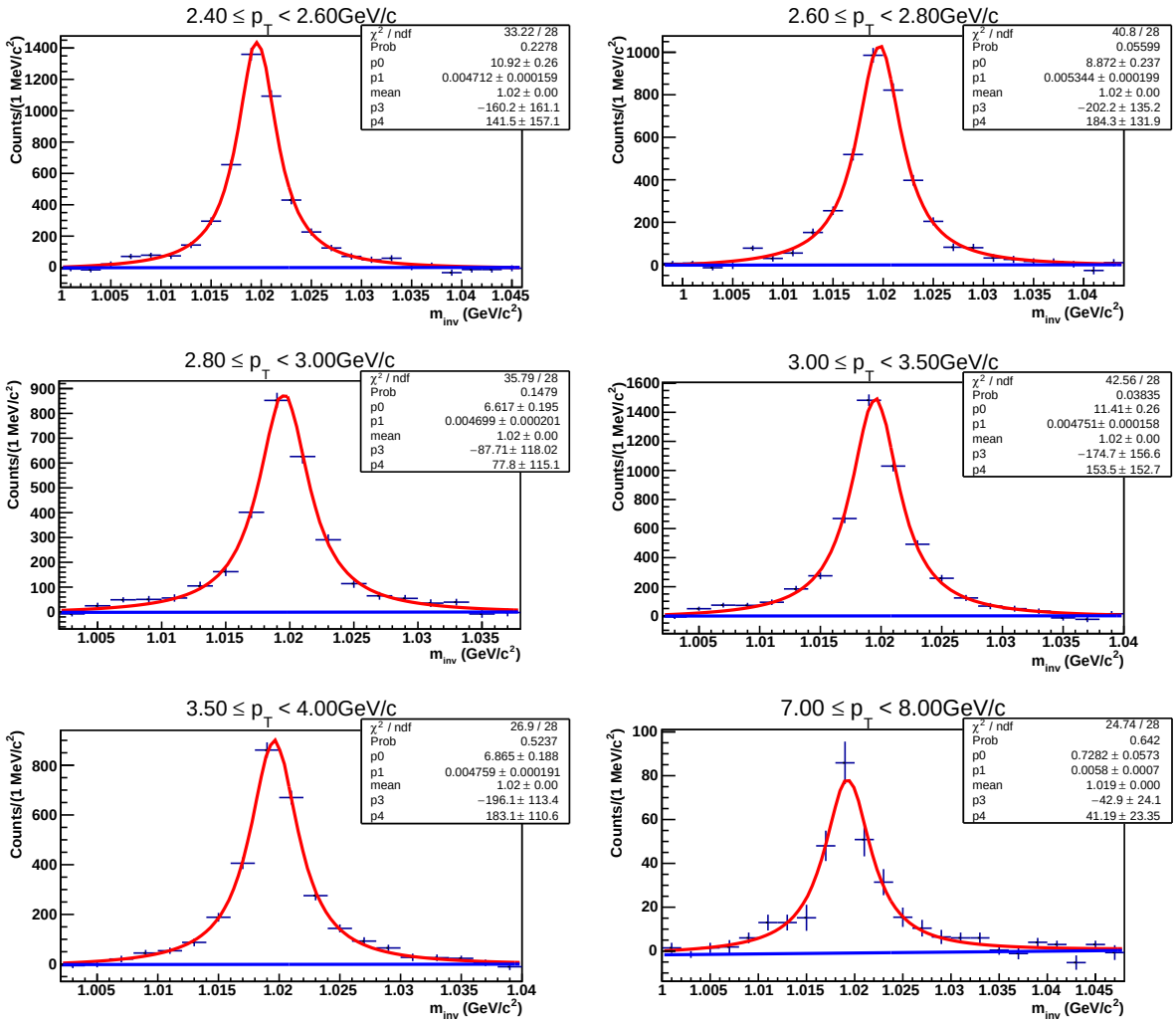

Figure 4.25: Pythia: Invariant mass distribution $m_{K^+K^-}$ fitted with BW+Poly1 (red line) in the p_T range $2.4 - 8.0$ GeV/c. The blue line shows the residual background described by Poly1.

For K^{*0} meson, raw yield is extracted from the $K^+\pi^-(K^-\pi^+)$ invariant mass distribution in 29 p_T bins ranging from 0.0 to 16 GeV/c in case of ALICE data and 30 p_T bins ranging from 0.0 to 18 GeV/c in case of Pythia data. The plots obtained from both ALICE data and Pythia data are shown in Fig. 4.26, Fig. 4.27, Fig. 4.28, Fig. 4.29 and Fig. 4.30 for ALICE data and Fig. 4.31, Fig. 4.32, Fig. 4.33, Fig. 4.34 and Fig. 4.35 for Pythia data for different p_T bins. The red line corresponds

to the total fit function (BW + RB) and the blue line shows only the parameterized residual background.

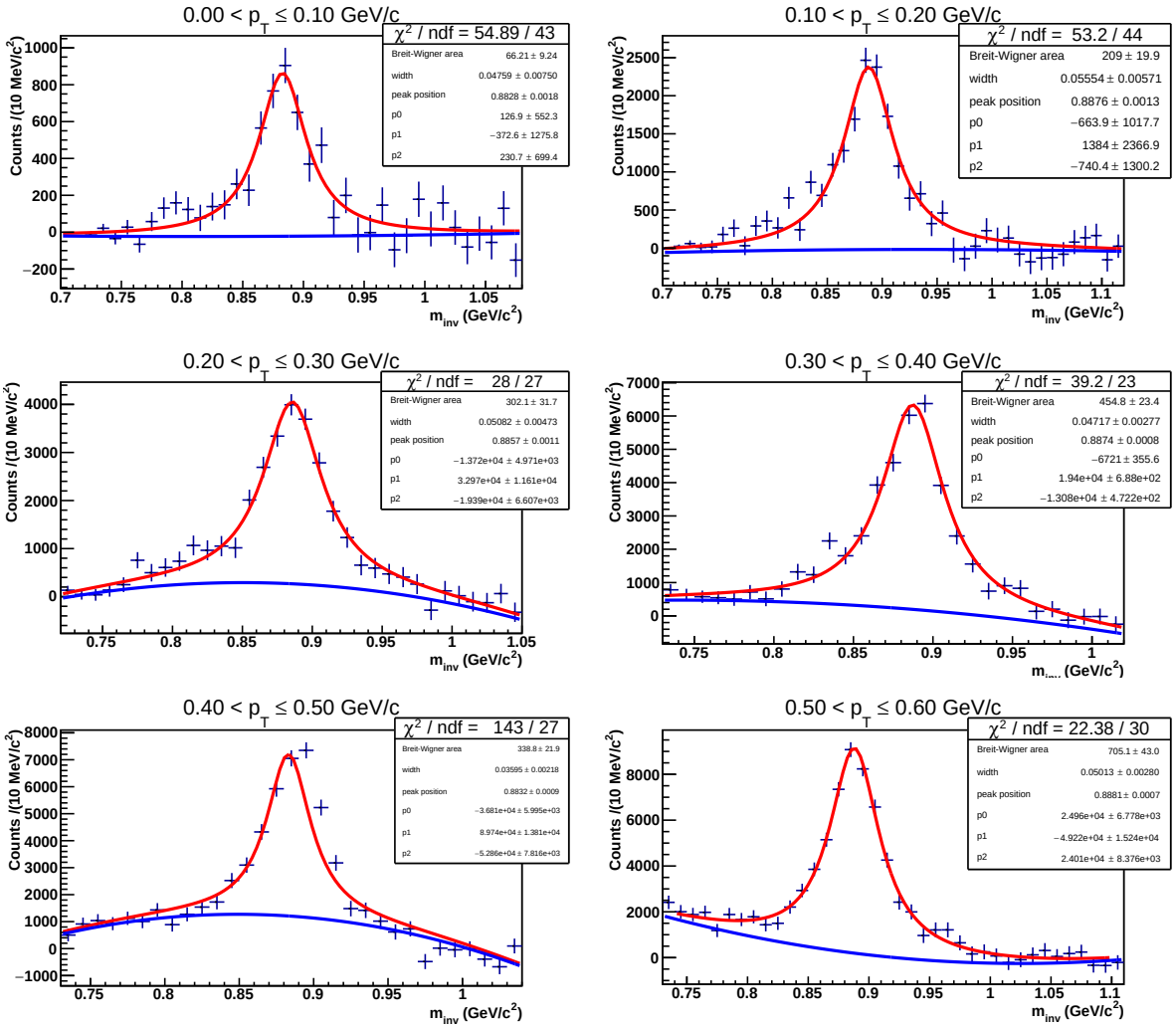

Figure 4.26: ALICE: Invariant mass distribution $m_{K^+\pi^-(K^-\pi^+)}$ fitted with BW+Poly2 (red line) in the p_T range 0.0−0.6 GeV/c. The blue line shows the residual background described by Poly2.

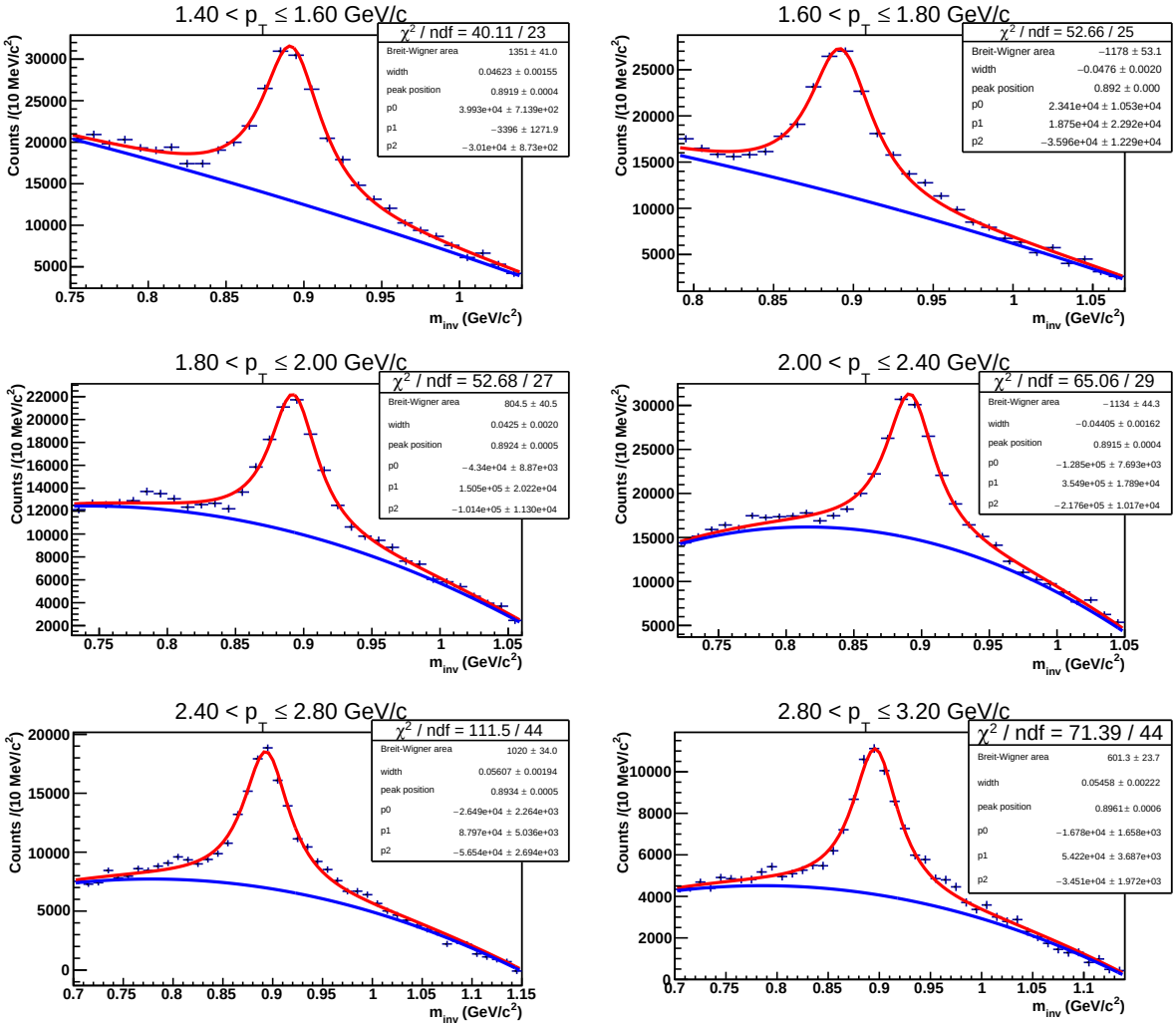

Figure 4.27: ALICE: Invariant mass distribution $m_{K^+\pi^-(K^-\pi^+)}$ fitted with BW+Poly2 (red line) in the p_T range 0.6−1.4 GeV/c. The blue line shows the residual background described by Poly2.

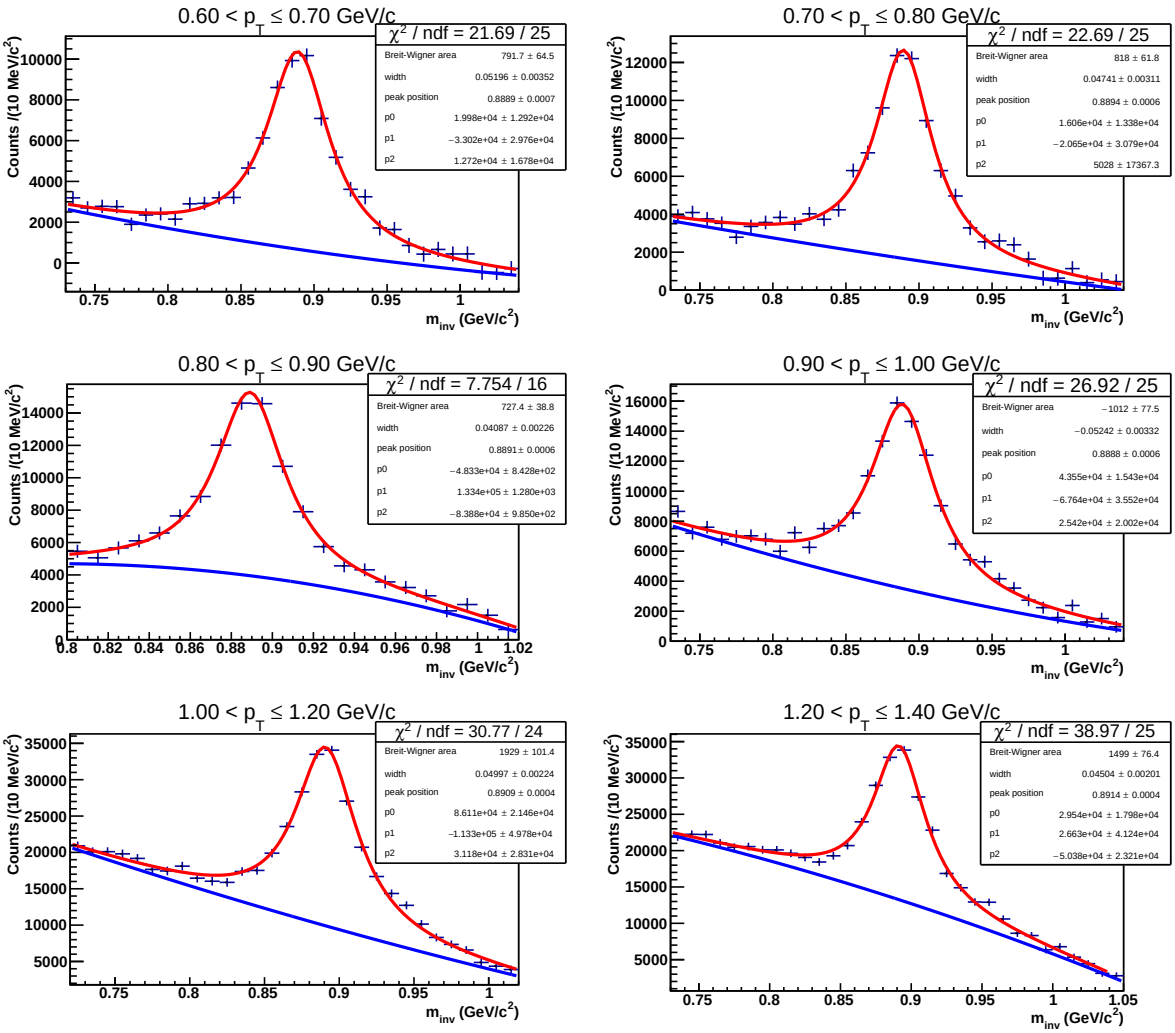

Figure 4.28: ALICE: Invariant mass distribution $m_{K^+\pi^-(K^-\pi^+)}$ fitted with BW+Poly2 (red line) in the p_T range1.4−3.2 GeV/c. The blue line shows the residual background described by Poly2.

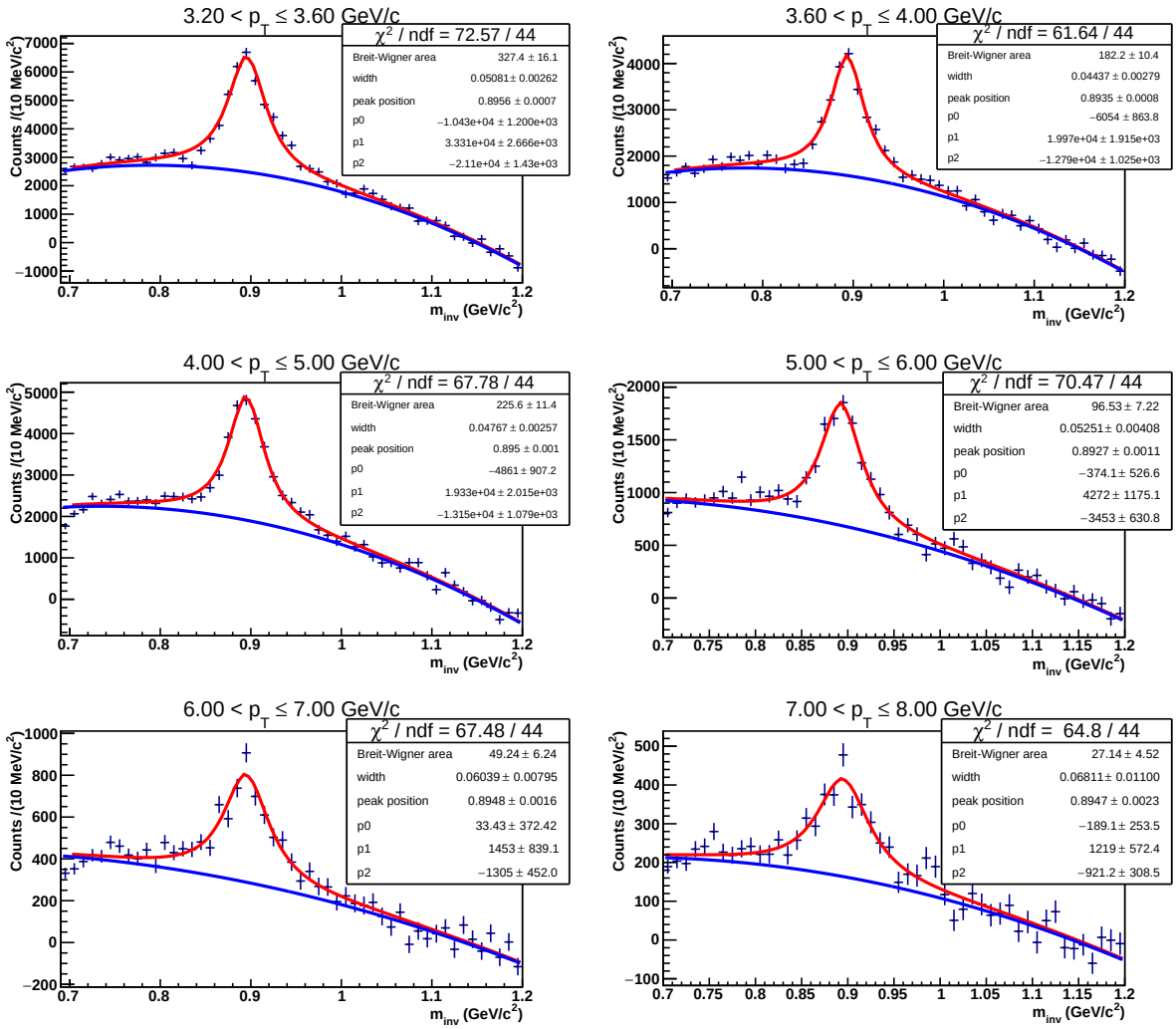

Figure 4.29: ALICE: Invariant mass distribution $m_{K^+\pi^-(K^-\pi^+)}$ fitted with BW+Poly2 (red line) in the p_T range 3.2−8.0 GeV/c. The blue line shows the residual background described by Poly2.

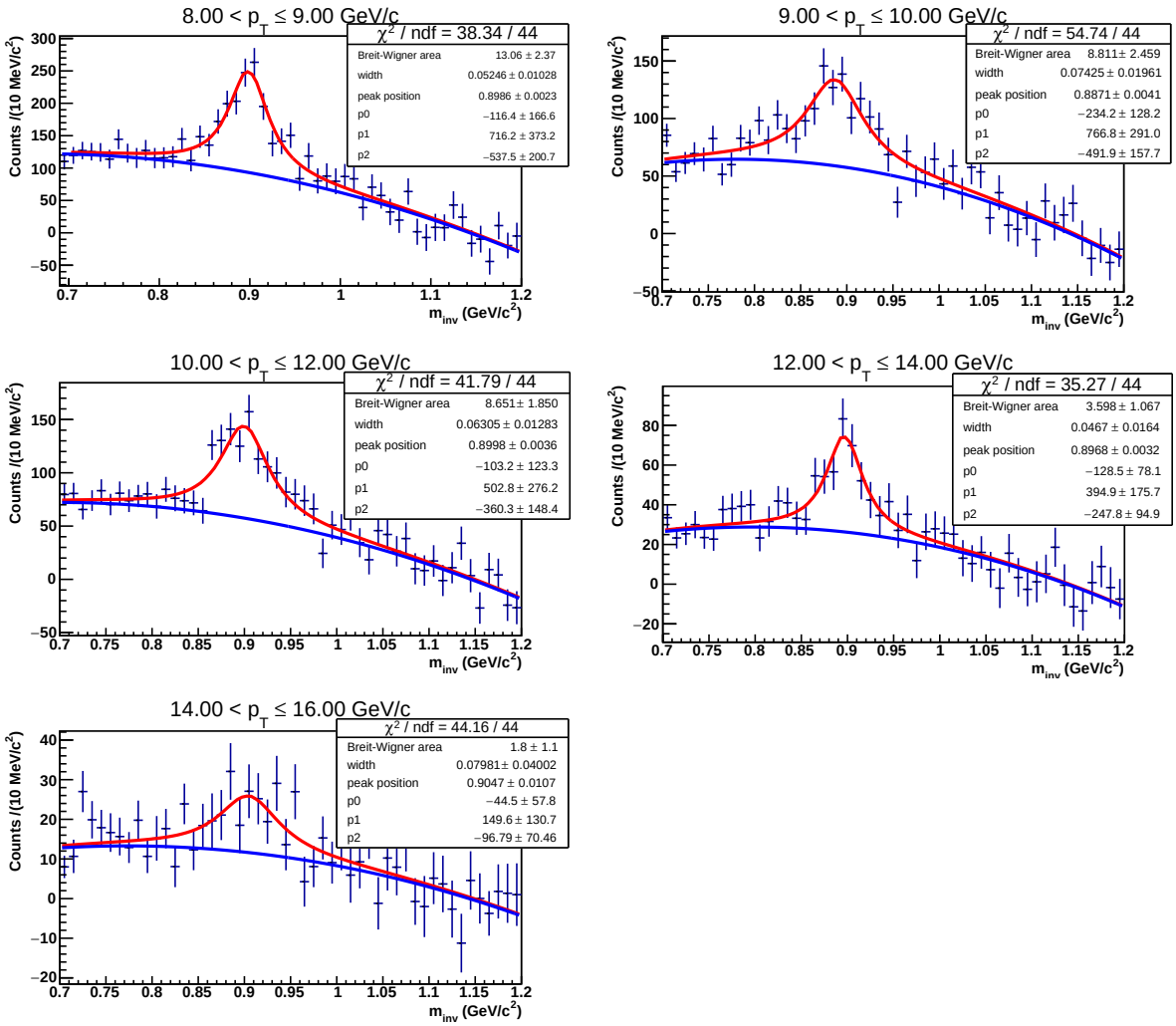

Figure 4.30: ALICE: Invariant mass distribution $m_{K^+\pi^-(K^-\pi^+)}$ fitted with BW+Poly2 (red line) in the p_T range $8.0 - 16.0$ GeV/c. The blue line shows the residual background described by Poly2.

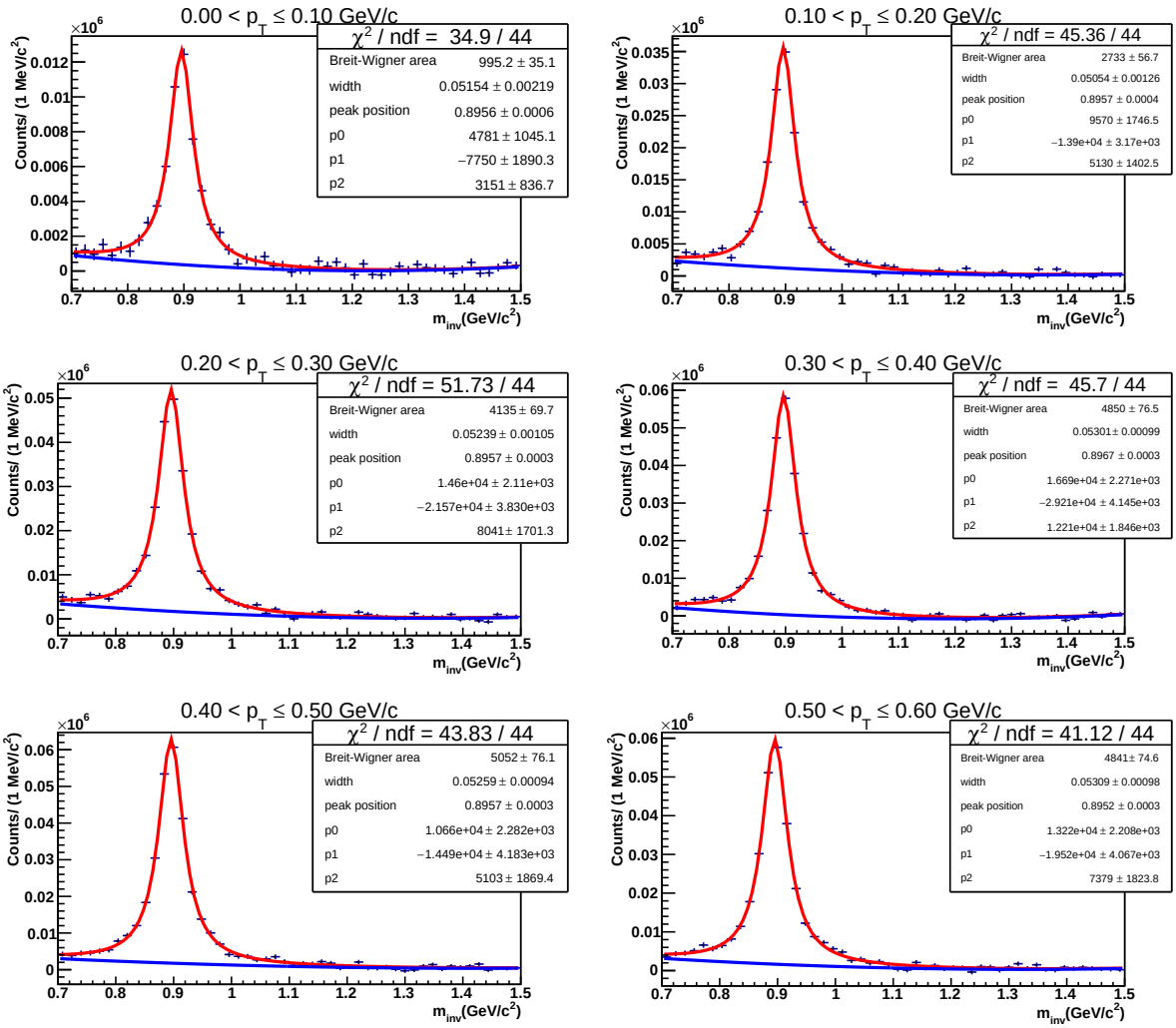

Figure 4.31: Pythia: Invariant mass distribution $m_{K^+K^-}$ fitted with BW+Poly1 (red line) in the p_T range $0.0 - 0.6$ GeV/c. The blue line shows the residual background described by Poly1.

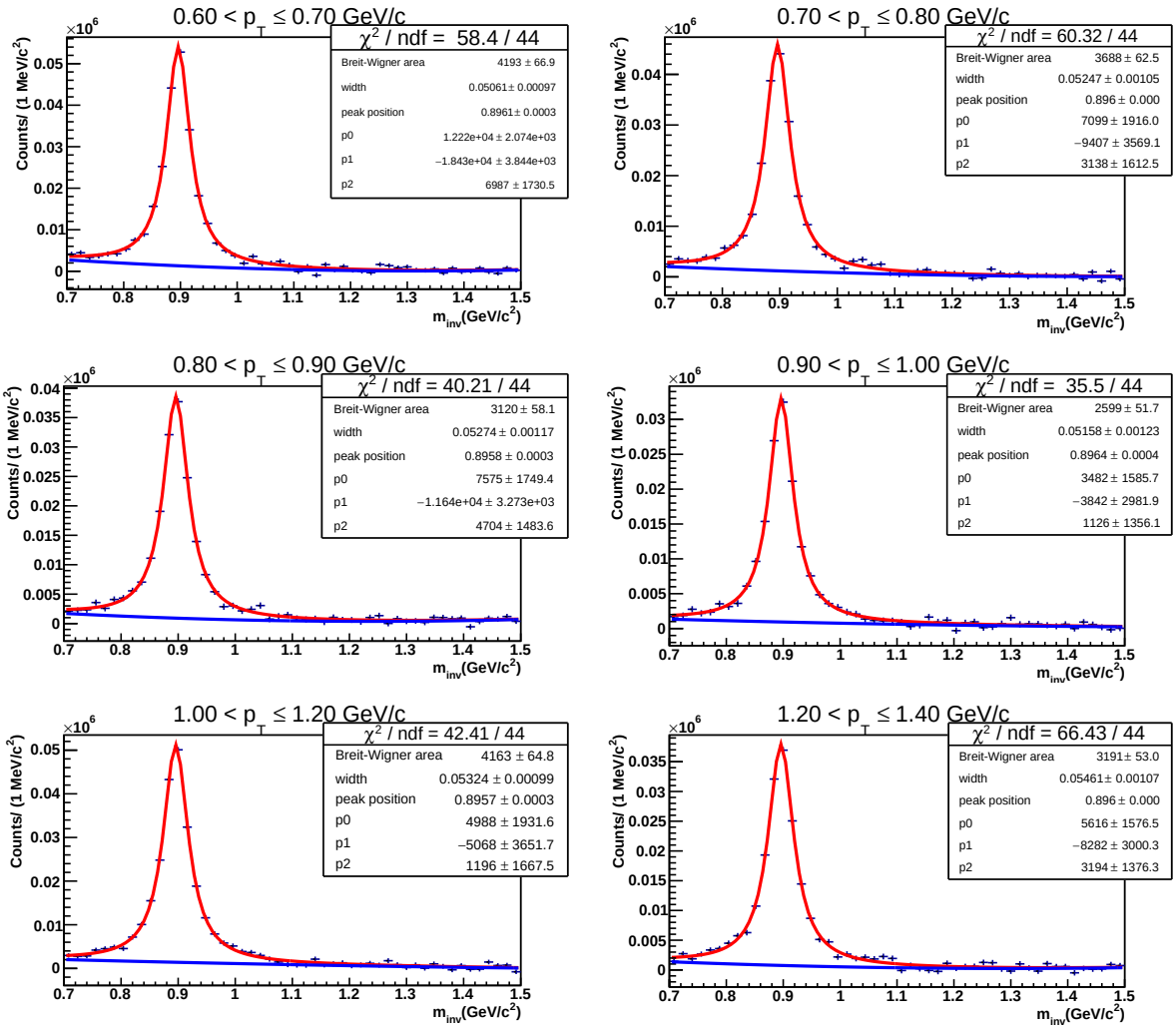

Figure 4.32: Pythia: Invariant mass distribution $m_{K^+K^-}$ fitted with BW+Poly1 (red line) in the p_T range $0.6 - 1.4$ GeV/c. The blue line shows the residual background described by Poly1.

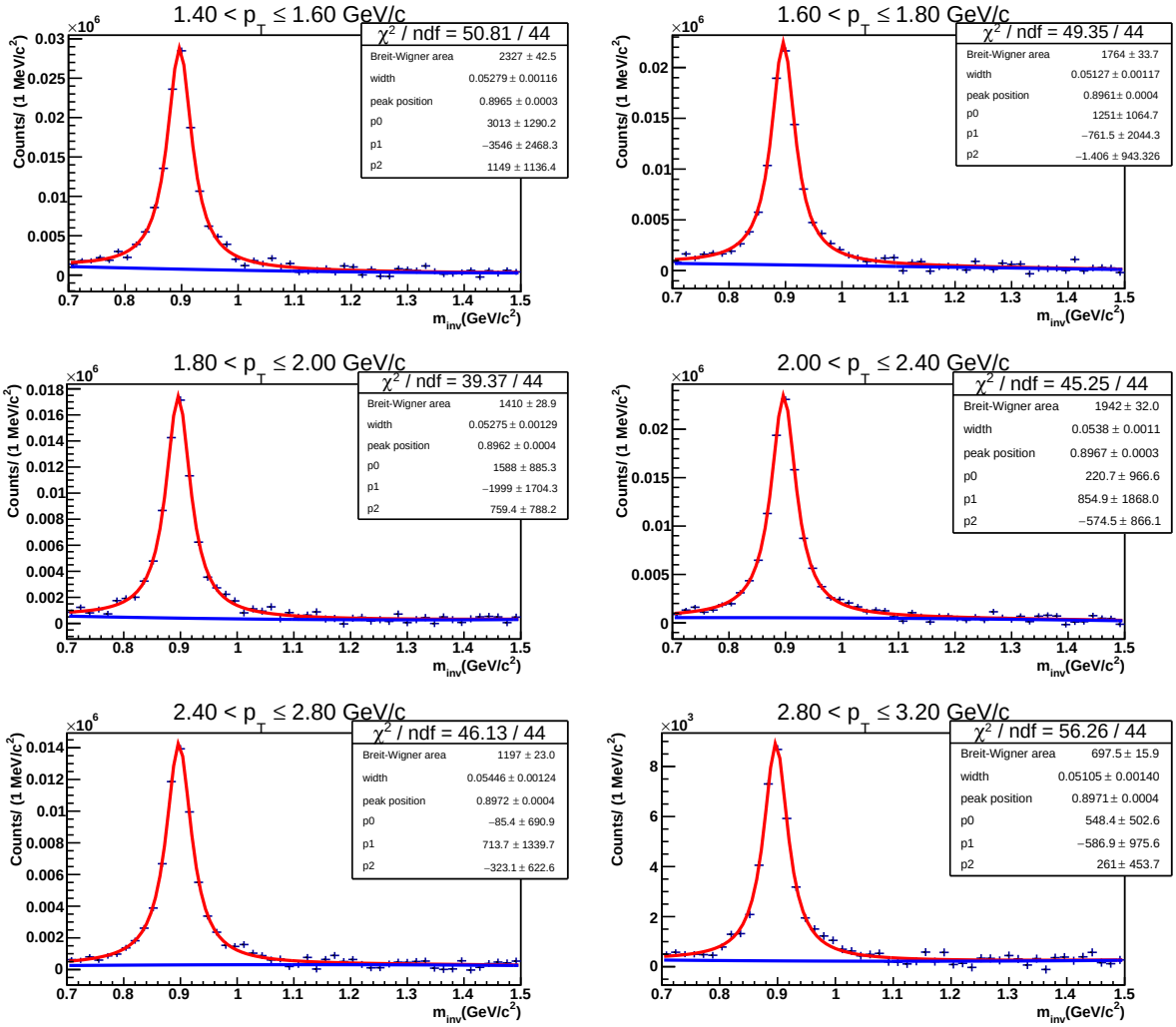

Figure 4.33: Pythia: Invariant mass distribution $m_{K^+K^-}$ fitted with BW+Poly1 (red line) in the p_T range $1.4 - 3.2$ GeV/c. The blue line shows the residual background described by Poly1.

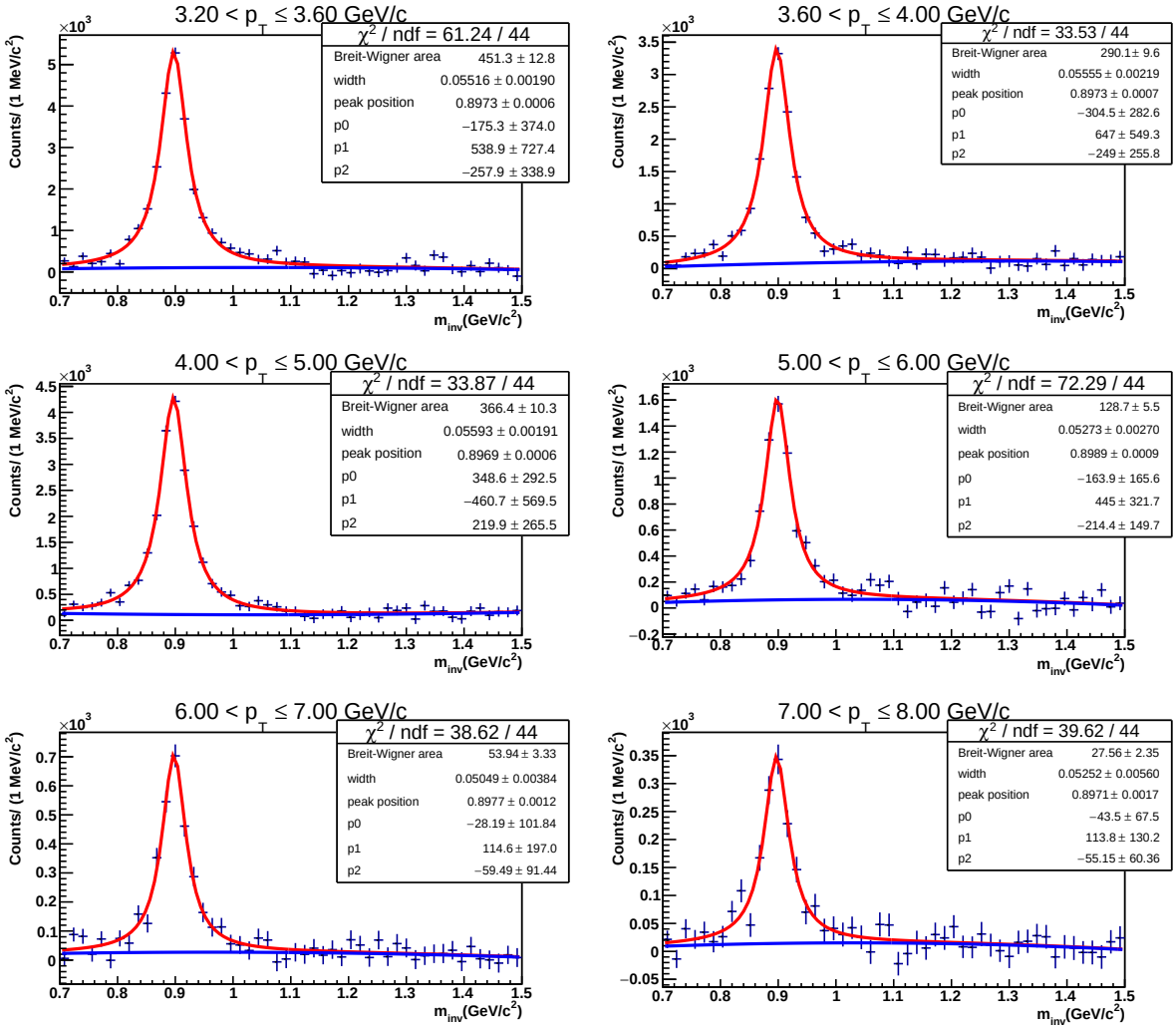

Figure 4.34: Pythia: Invariant mass distribution $m_{K^+K^-}$ fitted with BW+Poly1 (red line) in the p_T range $3.2 - 8.0$ GeV/c. The blue line shows the residual background described by Poly1.

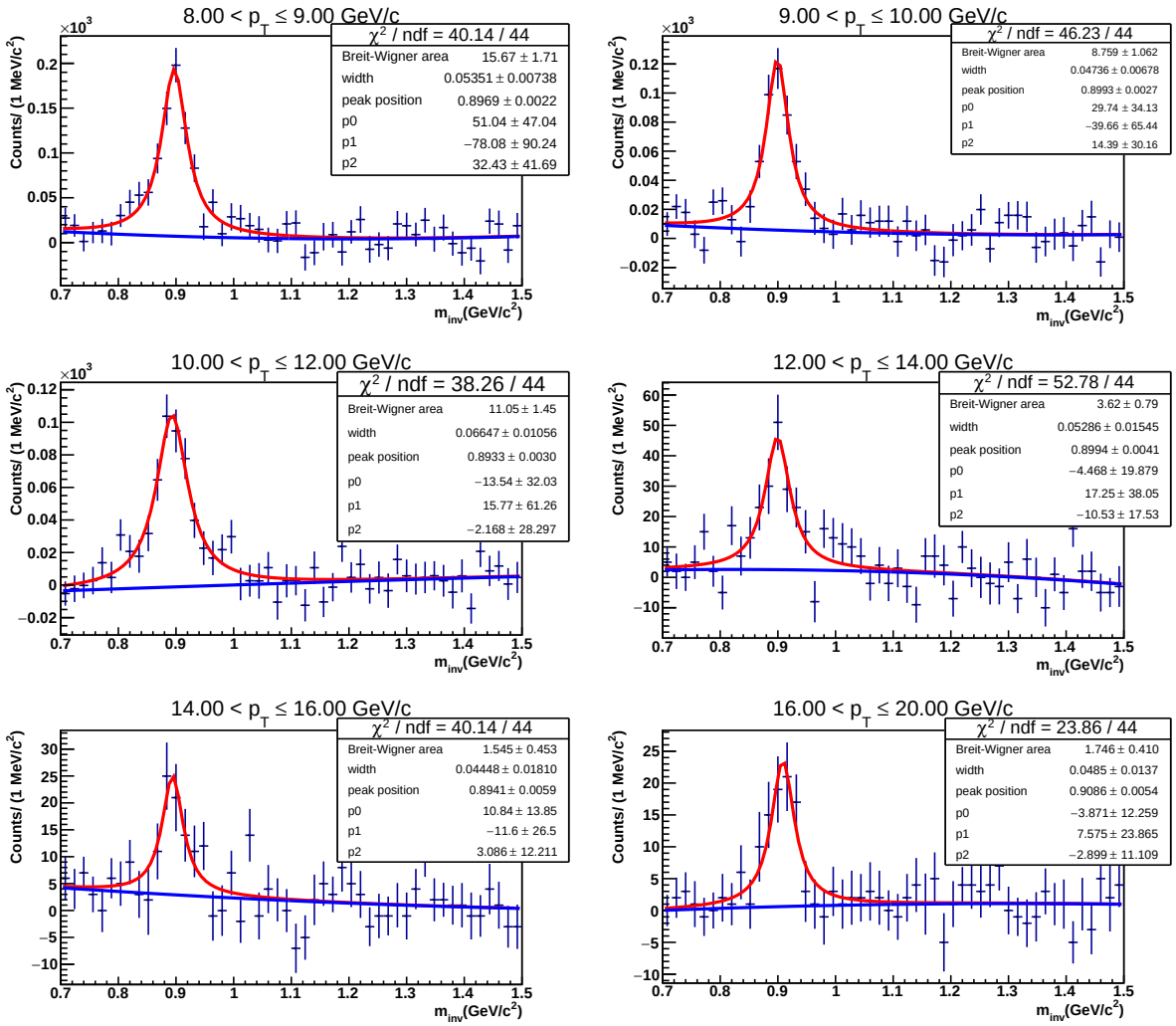

Figure 4.35: Pythia: Invariant mass distribution $m_{K^+K^-}$ fitted with BW+Poly1 (red line) in the p_T range $8.0 - 20.0$ GeV/c. The blue line shows the residual background described by Poly1.

From the fitting parameters of the above plots, we got the area under the signal, which is the yield value for each p_T bin. This measured yield is then plotted as a function of p_T. This plot is known as p_T spectra shown in Fig. 4.36 and Fig. 4.37 for ϕ and K^{*0} ALICE data.

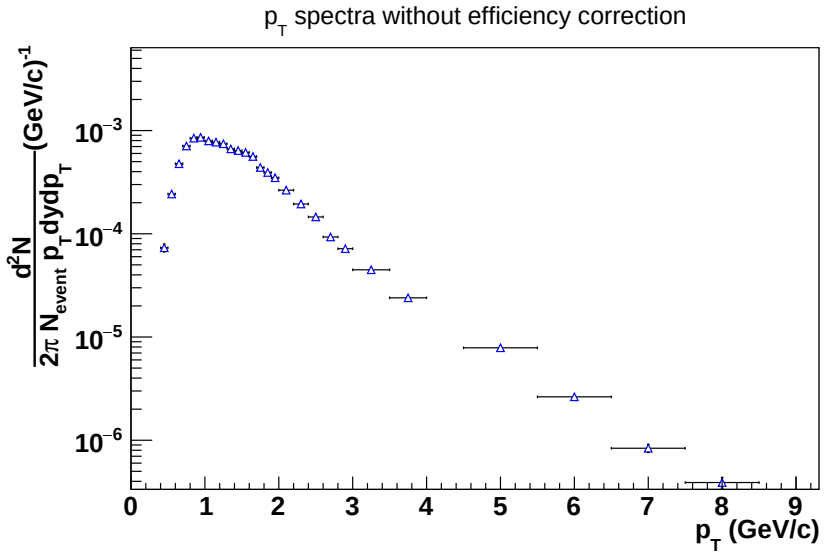

Figure 4.36: ALICE: Raw transverse momentum (p_T) spectra of ϕ meson in pp collisions at $\sqrt{s} = 7$ TeV.

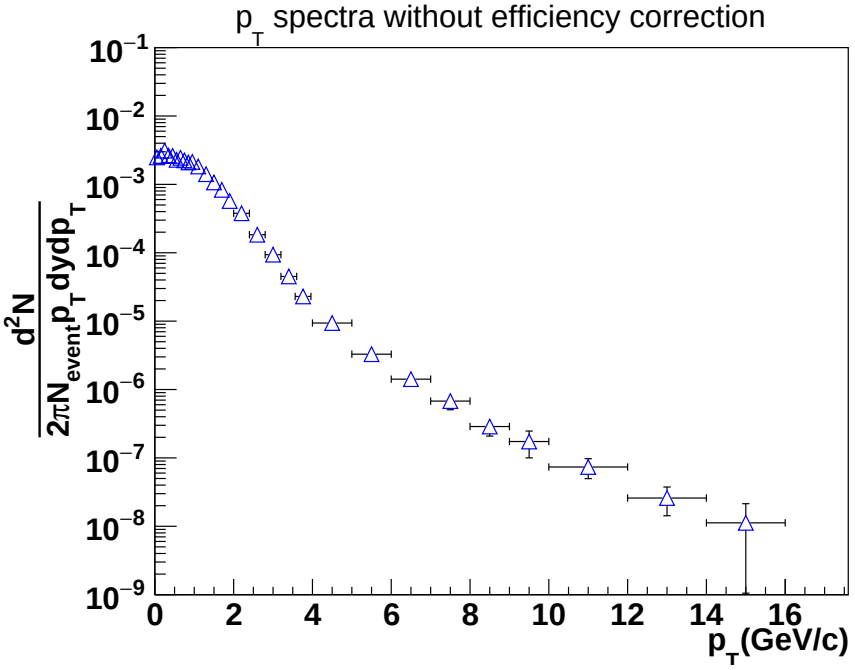

Figure 4.37: ALICE: Raw transverse momentum (p_T) spectra of K^{*0} meson in pp collisions at $\sqrt{s} = 7$ TeV.

4.9 Correction in data

In case of ALICE data, the yield obtained for ϕ and K^{*0} for different p_T bins have to be corrected for detector acceptance and tracking efficiency. The acceptance and reconstruction efficiency are calculated using simulation using the expression

$$(efficiency\, x\, acceptance) = \frac{N_{\phi/K^{*0}}\, reconstructed}{N_{\phi/K^{*0}}\, generated} \tag{4.3}$$

The (efficiency x acceptance) for same p_T scheme as used for yield calculation is obtained as shown in Fig. 4.38 and Fig. 4.39 for ϕ and K^{*0} respectively.

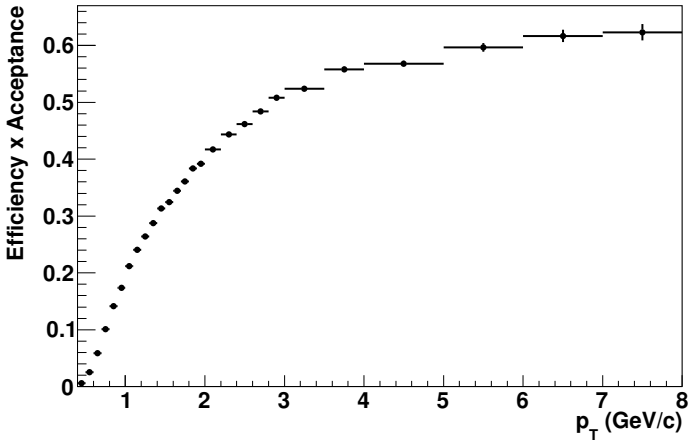

Figure 4.38: Acceptance x efficiency vs p_T of ϕ meson in pp collision at $\sqrt{s} = 7$ TeV.

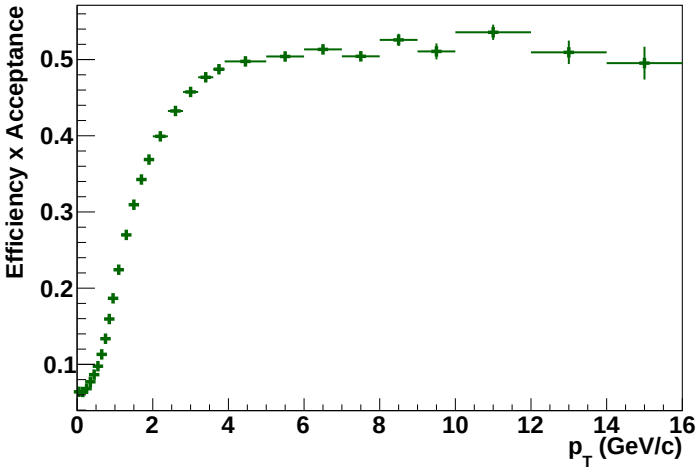

Figure 4.39: Acceptance x efficiency vs p_T of K^{*0} meson in pp collision at $\sqrt{s} = 7$ TeV.

The yield values for each p_T bin is divided by the (efficiency x acceptance) and branching ratio to get the corrected yield values.

4.10 Levy-Tsallis function for p_T spectra

Various functions can be fitted to the p_T spectra as obtained above. We are using Levy-Tsallis function which is of the form

$$\frac{1}{2\pi p_T}\frac{d^2N}{dp_T dy} = \frac{(dN/dy)(n-1)(n-2)}{2\pi nT(nT+m_0(n-2))}(1-\frac{m_T-m_0}{nT})^{-n} \tag{4.4}$$

where $m_T = \sqrt{p_T^2+m_0^2}$ and dN/dy, n and T are free parameters. This function has the advantage that it describes both the exponential shape of the spectrum at low p_T and the power-law distribution at large p_T , quantified by the inverse slope parameter T and the exponent parameter n, respectively.

Chapter 5

Results and Discussion

5.1 p_T spectra from ALICE data

The p_T spectra after efficiency correction for data from ALICE for ϕ and K^{*0} is presented below. The corrected p_T spectra for ϕ and K^{*0} are shown in Fig. 5.1 and Fig. 5.2 respectively. The red line shows the Levy-Tsallis fit.

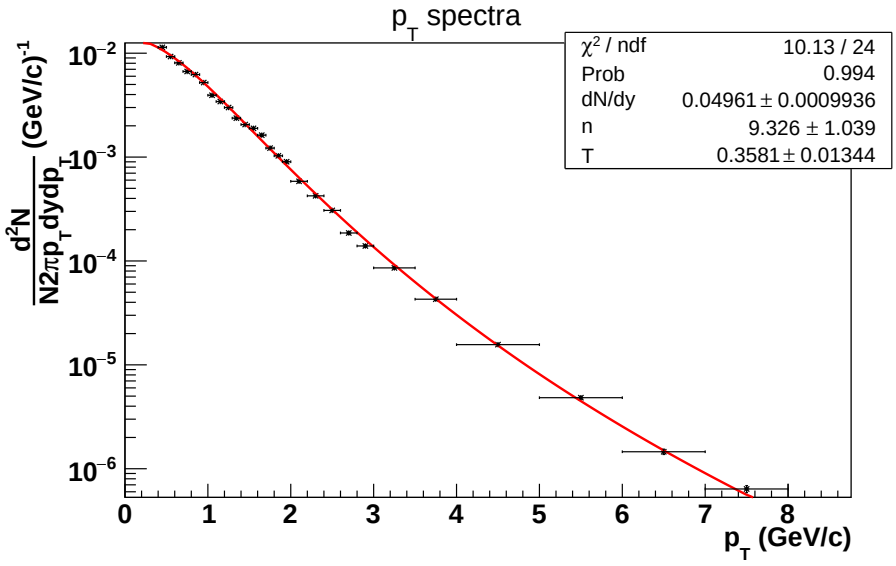

Figure 5.1: ALICE: Corrected transverse momentum (p_T) spectra of ϕ meson in pp collision at $\sqrt{s} = 7$ TeV. The red line shows Levy- Tsallis fit to the spectra.

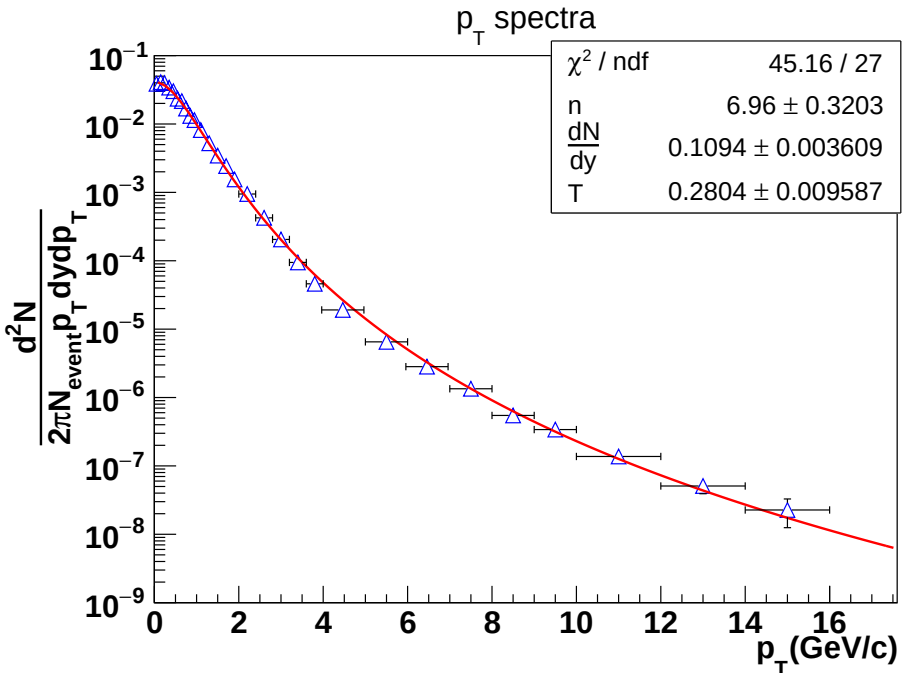

Figure 5.2: ALICE: Corrected transverse momentum (p_T) spectra of K^{*0} meson in pp collision at $\sqrt{s} = 7$ TeV. The red line shows Levy- Tsallis fit to the spectra.

5.2 p_T spectra from Pythia

No correction is done for data from Pythia as Pythia is an event generator which doesn't take into account of the detector efficiency and particle loss in the detector medium. The p_T spectra for ϕ and K^{*0} are shown in Fig. 5.3 and Fig. 5.4 where the red line shows the Levy-Tsallis fit.

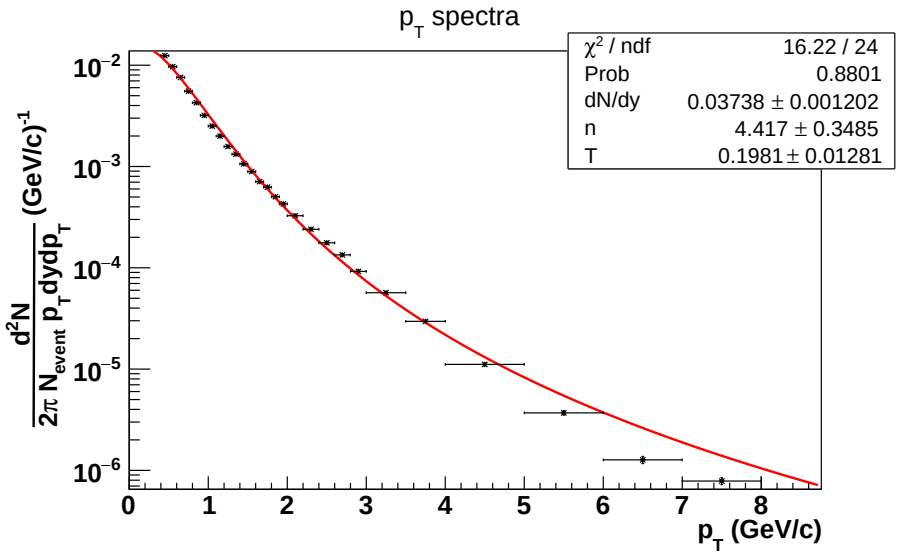

Figure 5.3: Pythia: Transverse momentum (p_T) spectra of ϕ meson in pp collision at $\sqrt{s} = 7$ TeV. The red line shows Levy- Tsallis fit to the spectra.

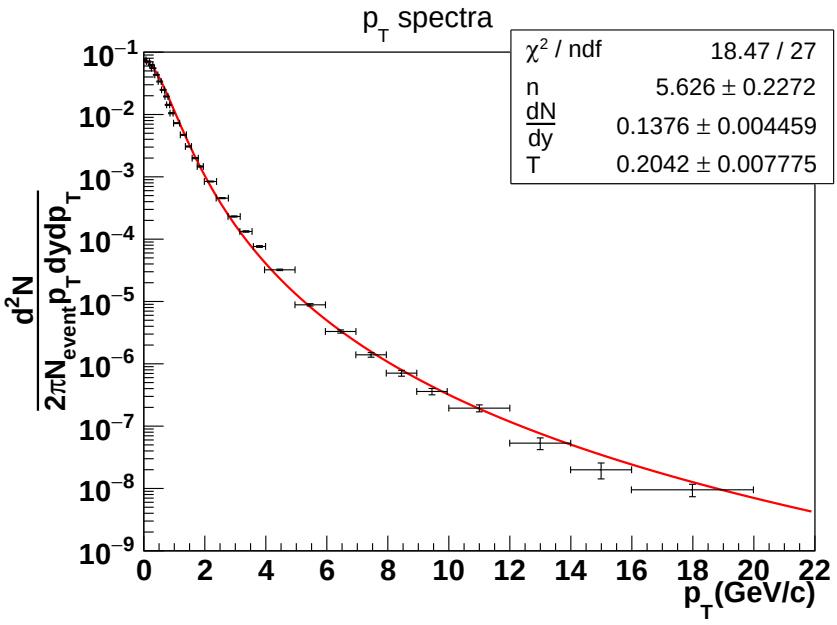

Figure 5.4: Pythia: Transverse momentum (p_T) spectra of K^{*0} meson in pp collision at $\sqrt{s} = 7$ TeV. The red line shows Levy- Tsallis fit to the spectra.

5.3 Comparison of data and model

5.3.1 Invariant mass and Width Γ as a function of p_T

Invariant mass and width Γ for ϕ as a function of p_T is plotted for both the data sets which are shown in Fig.5.3 for ALICE data and Fig. 5.4 Pythia data.

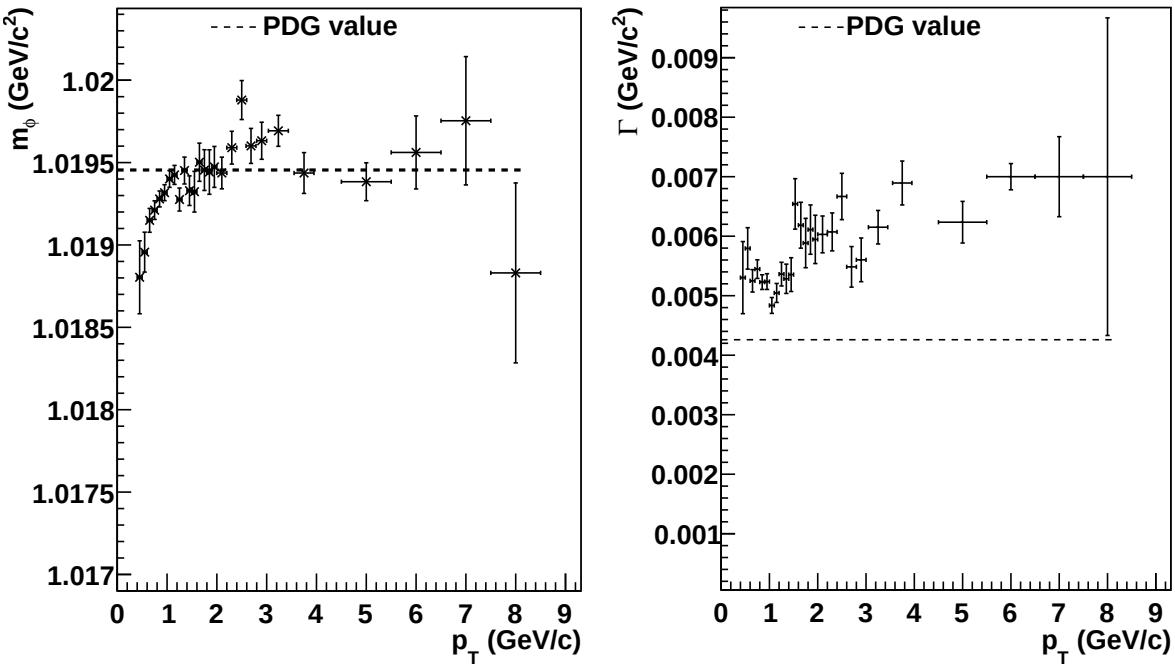

Figure 5.5: ALICE: ϕ meson mass and width Γ as a function of p_T. The dotted line shows the PDG value.

It can be inferred from the mass vs p_T plot that the Invariant mass is low at low p_T compared to the PDG values (shown in dotted lines) but it almost saturates as we increase the p_T value. The Γ values are above the theoretical values(show in dotted line). High values of Γ can be attributed to the resolution of the detector.

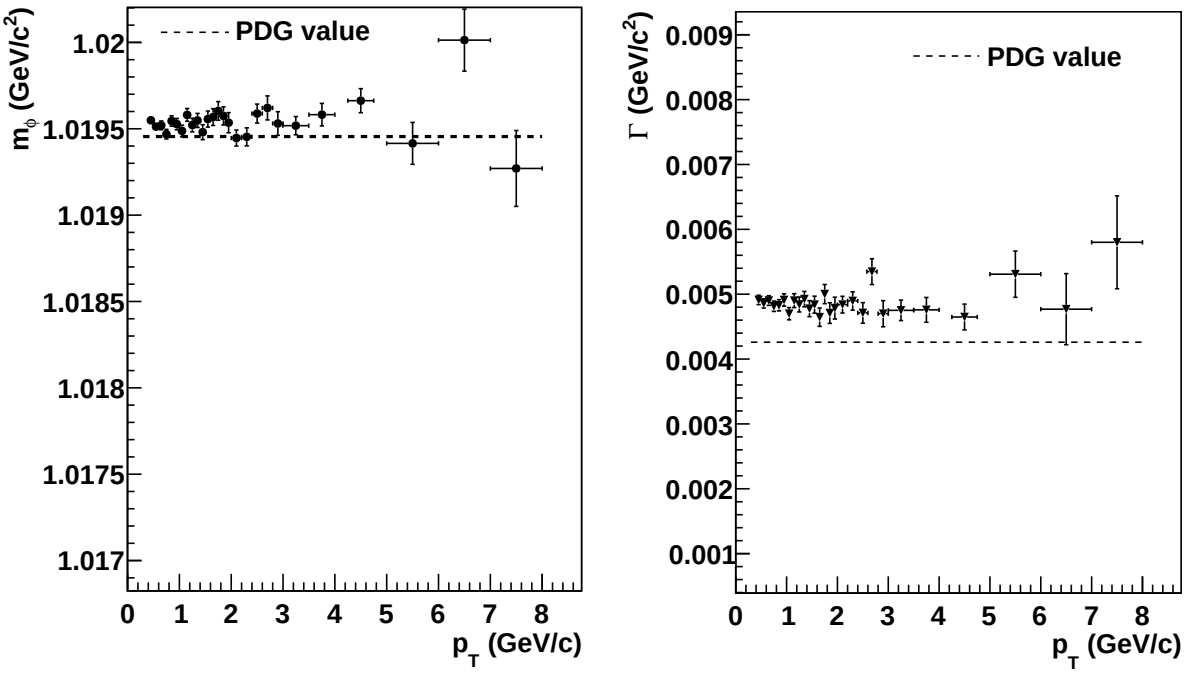

Figure 5.6: Pythia: ϕ meson mass and width Γ as a function of p_T. The dotted line shows the PDG value.

The mass is observed to be consistent for a wide p_T range. The Γ value is above the theoretical value(shown in dotted line) but is almost constant.
Mass and width Γ for K^{*0} as a function of p_T is plotted for both the data sets which are shown in Fig.5.7 for ALICE data and Fig. 5.8 Pythia data. In low p_T, we get a mass shift from PDG value, and that is may be due to energy loss.

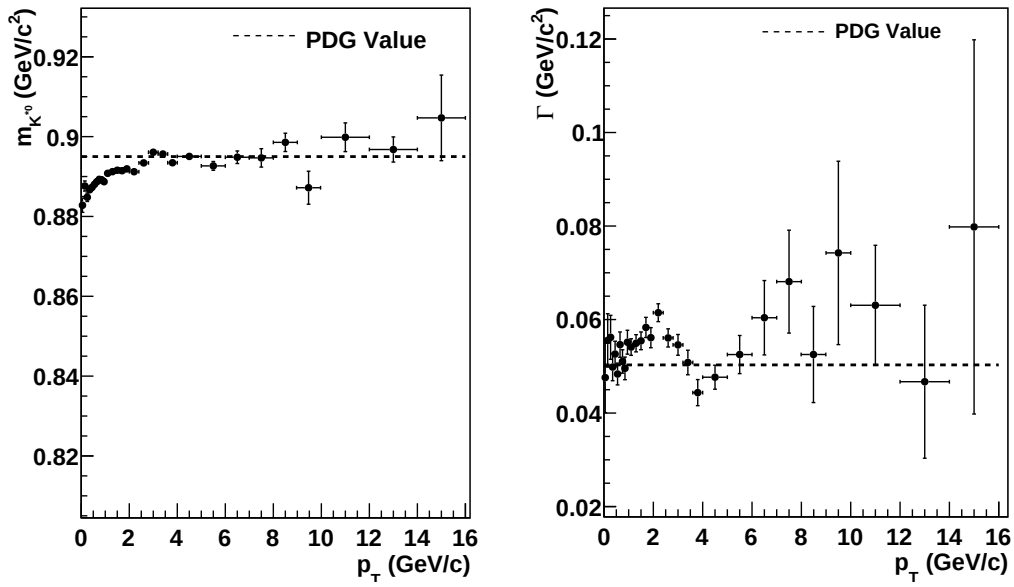

Figure 5.7: ALICE: K^{*0} mass and width Γ as a function of p_T. The dotted line shows the PDG value.

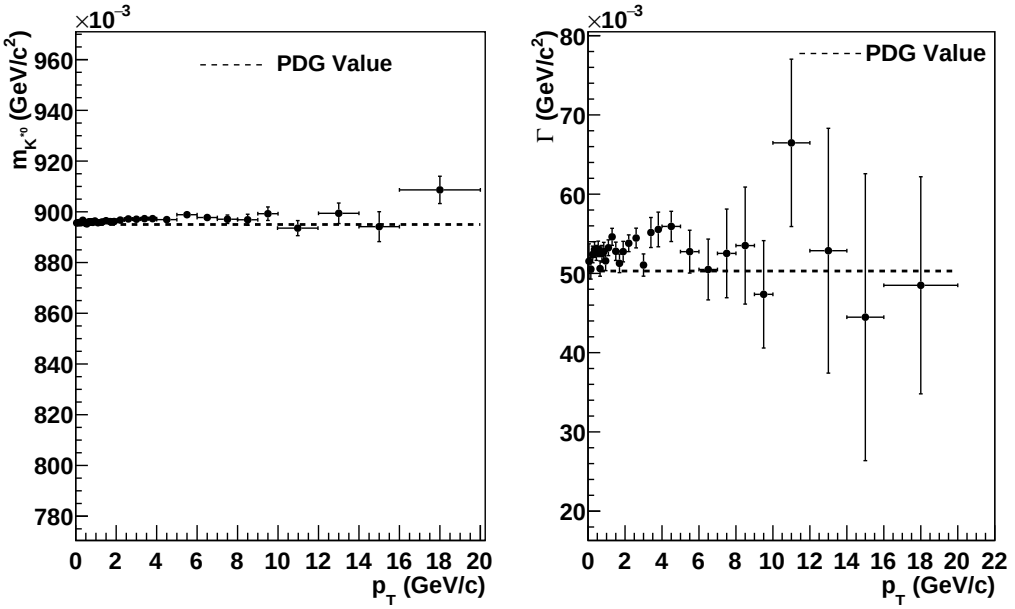

Figure 5.8: Pythia: K^{*0} meson mass and width Γ as a function of p_T. The dotted line shows the PDG value.

5.4 dN/dy and mean p_T ($< p_T >$)

The values of dN/dy and $< p_T >$ are obtained from the yield values by using

$$dN/dy = \Sigma 2\pi p_T f(p_T, y) dp_T$$
$$< p_T > = \Sigma 2\pi {p_T}^2 f(p_T, y) dp_T / \Sigma 2\pi p_T f(p_T, y) dp_T$$

where $f(p_T, y)$ are the invariant yield.

The total p_T integrated yield and mean transverse momentum $< p_T >$ for ϕ along with the Levi-Tsallis parameters are listed below:

ϕ	ALICE data	Pythia 6.4
dN/dy	0.0519 ± 0.0009	0.037 ± 0.001
n	9.326 ± 1	4.4 ± 0.3
T	0.35 ± 0.01	0.19 ± 0.01
$< p_T >$	1.21 ± 0.007	1.219 ± 0.004

Similarly, the total p_T integrated yield and mean transverse momentum $< p_T >$ for K^{*0} along with the Levi-Tsallis parameters are listed below:

K^{*0}	ALICE data	Pythia 6.4
dN/dy	0.1130 ± 0.0009	0.1284 ± 0.0005
n	6.9 ± 0.1	5.6 ± 0.2
T	0.280 ± 0.009	0.204 ± 0.007
$< p_T >$	1.0530 ± 0.0002	0.9602 ± 0.0004

The values of dN/dy are obtained by integrating the spectra in the measured range and extrapolating to zero p_T with the fitted Levy-Tsallis function where we do not have the measurement.

The calculated integrated yield and mean p_T values are compared with published results [11, 12] as shown in Fig. 5.9, Fig. 5.10 for ϕ and Fig. 5.11, Fig. 5.12 for K^{*0} respectively.

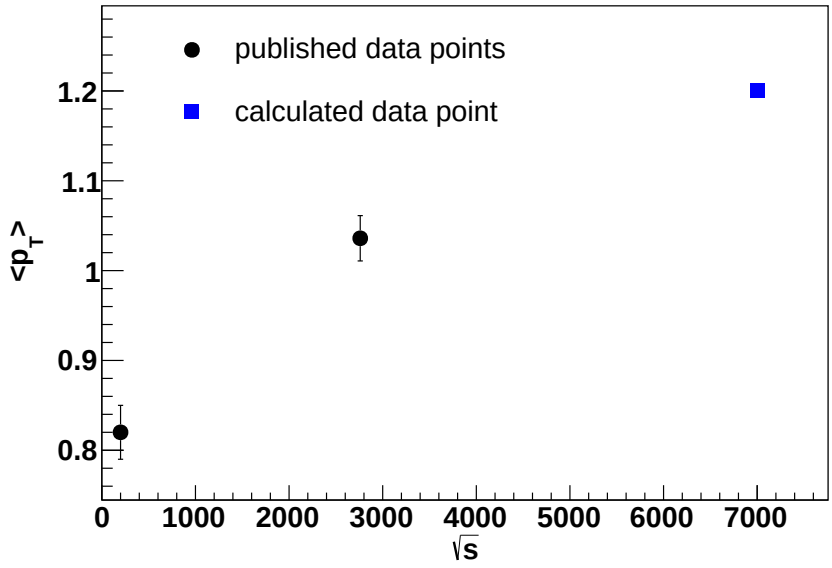

Figure 5.9: Mean p_T for ϕ as a function of $\sqrt{s}$

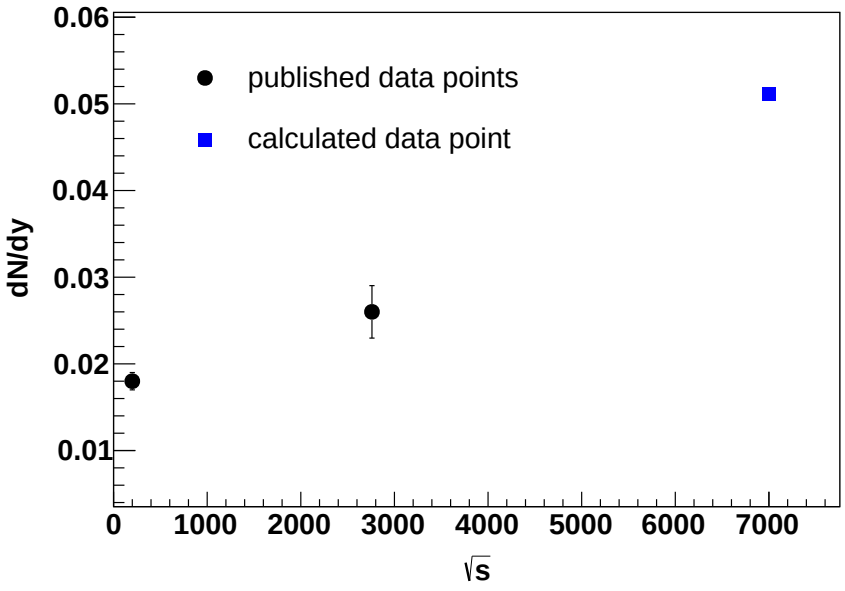

Figure 5.10: dN/dy for ϕ as a function of $\sqrt{s}$

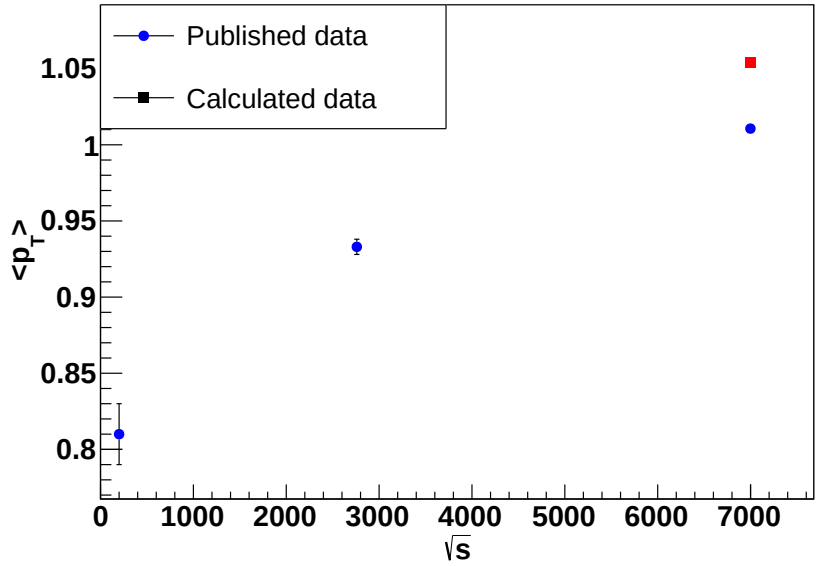

Figure 5.11: Mean p_T for K^{*0} as a function of $\sqrt{s}$

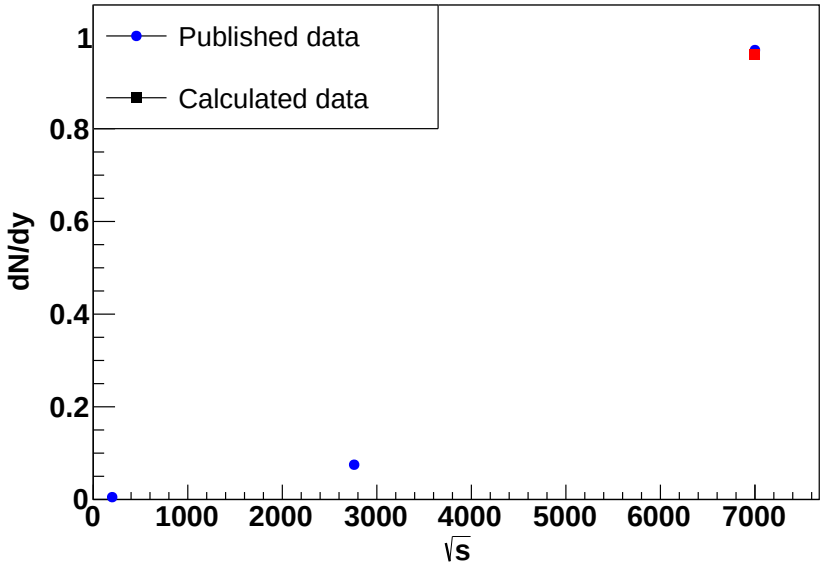

Figure 5.12: dN/dy for K^{*0} as a function of $\sqrt{s}$

The discrepancy between the published data and our data points can be attributed to the fact that the published data is for Events with INEL = 0 but our event

selection is INEL > 0. This means besides the VZERO trigger, we are demanding at least one charged particle in the ITS detector. So in that case, our observed yield is expected to be higher. We observe that the experimental results on yields and average transverse momentum are different from Pythia. Thus, the new LHC data allows for scope to further improve the existing models on particle production in proton-proton collisions.

References

[1] D.J. Gross, F. Wilczek , "Ultraviolet behavior of non-abelian gauge theories", Phys. Rev. Lett. **30**, 1343- 1346 (1973)

[2] H.D. Politzer, "Reliable perturbative results for strong interactions". Phys. Rev. Lett. **30** (26), 1346- 1349 (1973)

[3] J. C. Collins and M. J. Perry, Phys. Rev. Lett. **34**, 1353 (1975)

[4] T. Sjostrand, S. Mrenna and P. Skands, J. High Energy Phys. **05** (2006) 026

[5] K. Aamodt et al. (ALICE Collaboration), J. Instrum. **5** (2010) P03003

[6] J. Alme et al., Nucl. Instr. Meth. in Phys. Res. A **622** (2010) 316

[7] A. Akindinov et al., Nuovo Cim. B **124** (2009) 235

[8] A. Akindinov et al., Eur. Phys. J. C **68** (2010) 601

[9] https://root.cern.ch

[10] The ALICE Collaboration, Production of $K*^0(892)$ and ϕ (1020) in pp collisions at sqrt(s)=7 TeV

[11] J. Adams et al (STAR collaboration), Phi meson production in Au+Au and p+p collisions at sqrt (s)=200 GeV, Phys. Letters B **612**, 181(2005)

[12] CMS Collaboration, Study of high-pT charged particle suppression in PbPb compared to pp collisions at sqrt(sNN)=2.76 TeV, Eur. Phys. J. C (2012) 72:218

[13] ALICE preliminary

[14] W. R. Leo, Techniques for Nuclear and Particle Physics Experiments, Springer

Appendix A

(Analysis Macros)

A.1 Macro for ALICE data

frame

```
1  #include "THnSparse.h"
2  #include "TVector3.h"
3  #include "TH1.h"
4  #include "TH2.h"
5  #include "TMath.h"
6  #include "TLorentzVector.h"
7  #include "TExMap.h"
8  #include "TTree.h"
9  //#include "AliAnalysisPhiPPMultDep.h"
10 #include <vector>
11 #include<iostream>
12 #include<fstream>
13 #include<math.h>
14 //Breit-Wigner function
15 Double_t mybw(Double_t* x, Double_t* par)
16 {
17   Double_t arg1 = 14.0/22.0; // 2 over pi
18   Double_t arg2 = par[1]*par[1]*par[2]*par[2]; //Gamma=par[1]  M=par[2]
19   Double_t arg3 = ((x[0]*x[0]) - (par[2]*par[2]))*((x[0]*x[0]) - (par[2]*par[2]));
20   Double_t arg4 = x[0]*x[0]*x[0]*x[0]*((par[1]*par[1])/(par[2]*par[2]));
21   return par[0]*arg1*arg2/(arg3 + arg4)+(par[3]+par[4]*x[0]+par[5]*x[0]*x[0]);
22 }
23 Double_t mybwn(Double_t* x, Double_t* par)
24 {
25   Double_t arg1 = 7.0/44.0; // 1 over 2*pi
26   Double_t arg2 = par[1]; //Gamma=par[1]  A=par[0] M=par[2]
27   Double_t arg3 = (x[0] - par[2])*(x[0] - par[2]);
28   Double_t arg4 = (par[1]*par[1])/4;
29   return (par[0]*arg1*arg2)/(arg3 + arg4)+(par[3]+par[4]*x[0]+par[5]*x[0]*x[0]);
30
31 }
32 //Levy-Tsallis Function
33 Double_t ltf(Double_t* x, Double_t* par)
34 {
35   //par[0]=dN/dy par[1]=n par[2]=T x[0]=pT
36   Double_t pi=22/7,mt,mzero=1.019;
37   mt=sqrt(x[0]*x[0]+mzero*mzero);
38   Double_t arg1= par[0]*(par[1]-1)*(par[1]-2);
39   Double_t arg2= 2*pi*par[1]*par[2];
40   Double_t arg3= par[1]*par[2] + mzero*(par[1]-2);
41   Double_t arg4= pow(1+((mt-mzero)/(par[1]*par[2])), -1*par[1]);
42   return (arg1*arg4)/(arg2*arg3);
43
44 }
45 Double_t poly(Double_t* x, Double_t* par)
```

```
46  {
47    return (par[0]+par[1]*x[0]+par[2]*x[0]*x[0]);
48  }
49  Double_t ltfm(Double_t* x, Double_t* par)
50  {
51    //par[0]=dN/dy par[1]=n par[2]=T x[0]=pT
52    Double_t pi=22/7,mt,mzero=1.019;
53    mt=sqrt(x[0]*x[0]+mzero*mzero);
54    Double_t arg1= par[0]*(par[1]-1)*(par[1]-2);
55    Double_t arg2= 2*pi*par[1]*par[2];
56    Double_t arg3= par[1]*par[2] + mzero*(par[1]-2);
57    Double_t arg4= pow(1+((mt-mzero)/(par[1]*par[2])), -1*par[1]);
58    return (x[0]*arg1*arg4)/(arg2*arg3);
59
60  }
61  Double_t ltfd(Double_t* x, Double_t* par)
62  {
63    //par[0]=dN/dy par[1]=n par[2]=T x[0]=pT
64    Double_t pi=22/7,mt,mzero=1.019;
65    mt=sqrt(x[0]*x[0]+mzero*mzero);
66    Double_t arg1= par[0]*(par[1]-1)*(par[1]-2);
67    Double_t arg2= 2*pi*par[1]*par[2];
68    Double_t arg3= par[1]*par[2] + mzero*(par[1]-2);
69    Double_t arg4= pow(1+((mt-mzero)/(par[1]*par[2])), -1*par[1]);
70    return (2*pi*x[0]*arg1*arg4)/(arg2*arg3);
71
72  }void dataone()
73  {
74    gStyle->SetOptStat(0);
75    gStyle->SetOptFit(1111);
76    gStyle->SetStatColor(0);
77    gStyle->SetCanvasColor(0);
78    gStyle->SetPadColor(0);
79    gStyle->SetPadBorderMode(0);
80    gStyle->SetCanvasBorderMode(0);
81    gStyle->SetFrameBorderMode(0);
82    gStyle->SetFillColor(0);
83    gStyle->SetLineColor(1);
84    gStyle->SetTitleTextColor(1);
85    gStyle->SetTitleColor(1);
86    TFile *f = new TFile("resultsn.root");
87    TFile *ef = new TFile("HimangshuPhiEfficiencyMinBias.root");
88    ofstream fout("outputdata.txt");
89    fHistTotal  = new TH1D("fHistTotal","Inv Mass Dist" ,120,0.98,1.1);
90    fHistMix  = new TH1D("fHistMix","Inv Mass Dist mix" ,120,0.98,1.1);
91    TH1I *fHistCent=(TH1I*)f->Get("fHistCent");
92    fTHnSV0MCentSig  = (THnSparseF *) f->Get("fTHnSV0MCentSig");
93    fTHnSV0MCentMix  = (THnSparseF *) f->Get("fTHnSV0MCentMix");
94    TH2F *hmpt = new TH2F("hmpt","Mass vs pT",100,0.,10.,120,0.98,1.1);
95    TH2F *hgpt = new TH2F("hgpt","Gamma vs pT",10,0.,10.,100,0.,0.01);
96
97    hmpt->SetXTitle("p_{T} (GeV/c^{1})");
98    hmpt->SetYTitle("m_{inv} (GeV/c^{2})");
99    hgpt->SetXTitle("p_{T} (GeV/c^{1})");
100   hgpt->SetYTitle("Gamma");
101   const int nDims = 3;
102   const int nBins[nDims] =   {200, 120,100};
103   const double xMin[nDims] = {0, 0.98,0};
104   const double xMax[nDims] = {20, 1.1,100};
105   int bmin,bmax,binnmbr;
106   TF1 *funcn[27];
107   double integral,integral1,ratio,ntot=fHistCent->GetEntries(),mzero[30],gamma[30],
          mzeroe[30],gammae[30],ptn[30],efc[30];
108   TLine *line = new TLine(0.,1.019455,8.1,1.019455);
```

```
109    TLine *line1 = new TLine(0.,0.00426,8.1,0.00426);
110    Float_t par[30],para[30];
111
112    int k=0,i,up=5,down=5;
113    double integ[30],j=0.45,j2=1.9,j3=2.75,j4=3,k2=19,k3=26,k4=21,fpt[30],fpte[30],upp,
           downn;
114
115    TF1 *polyn[27];
116
117    TCanvas *c = new TCanvas("c","", 800, 1200);
118    TCanvas *c11 = new TCanvas("c11","", 800, 1200);
119    TCanvas *c22 = new TCanvas("c22","", 800, 1200);
120    TCanvas *c33 = new TCanvas("c33","", 800, 1200);
121    TCanvas *d = new TCanvas("d","", 800, 800);
122    c->Divide(2,3);
123    c11->Divide(2,3);
124    c22->Divide(2,3);
125    c33->Divide(2,3);
126    d->Divide(2,2);
127    for(i=0;i<27;i++)
128      {
129          funcn[i]= new TF1(Form("mybwn%d",0),mybwn,0.98,1.1,6);
130          funcn[i]->SetParameter(0,124);    funcn[i]->SetParName(0,Form("AreaA%d",0));
131        funcn[i]->SetParameter(1,0.004);      funcn[i]->SetParName(1,Form("sigma%d",0));
132        funcn[i]->SetParameter(2,1.019);     funcn[i]->SetParName(2,Form("mean%d",0));
133        funcn[i]->SetParameter(3,3);     funcn[i]->SetParName(3,Form("A%d",0));
134        funcn[i]->SetParameter(4,7);     funcn[i]->SetParName(4,Form("B%d",0));
135        funcn[i]->SetParameter(5,9);     funcn[i]->SetParName(5,Form("C%d",0));
136        funcn[i]->SetParLimits(2,1.015,1.025);funcn[i]->SetParLimits(1,0.0032,0.007);
137        funcn[i]->FixParameter(5,0);
138
139        if(i<6)
140        c->cd(i+1);
141        else if(i<12)
142        c11->cd(i-5);
143        else if(i<18)
144        c22->cd(i-11);
145        else if(i<24)
146        c33->cd(i-17);
147        else
148      {
149        d->cd(i-15);
150      }
151            upp=up;
152            upp=upp/10;
153            downn=down;
154            downn=(downn-1)/10;
155        fTHnSV0MCentSig->GetAxis(0)->SetRange(down, up);
156            fTHnSV0MCentSig->GetAxis(0)->SetBit(TAxis::kAxisRange);//Pt
157        fTHnSV0MCentSig->GetAxis(2)->SetRange(0, 100);
158            fTHnSV0MCentSig->GetAxis(2)->SetBit(TAxis::kAxisRange);
159        fTHnSV0MCentMix->GetAxis(0)->SetRange(down, up);
160            fTHnSV0MCentMix->GetAxis(0)->SetBit(TAxis::kAxisRange);
161        fTHnSV0MCentMix->GetAxis(2)->SetRange(0, 100);
162            fTHnSV0MCentMix->GetAxis(2)->SetBit(TAxis::kAxisRange);//Pt mix
163        fHistTotal = fTHnSV0MCentSig->Projection(1,"E");//projection x
164        fHistMix = fTHnSV0MCentMix->Projection(1,"E");//projection x
165        TAxis *axis = fHistTotal->GetXaxis();
166
167        bmin=axis->FindBin(1.04);
168        bmax=axis->FindBin(1.06);
169        integral = fHistTotal->Integral(bmin,bmax);
170        integral1 = fHistMix->Integral(bmin,bmax);
171        ratio= integral/integral1;
```

```
172         fHistMix->Scale(ratio);
173         fHistTotal->Add(fHistMix,-1);
174         if(i==0)
175         {
176            fHistTotal->Fit(funcn[i],"","",0.99,1.05);polyn[i] = new TF1(Form("poly%d"
                ,0),poly,0.99,1.05,3);
177         }
178         else if(i==1)
179         {
180            fHistTotal->Fit(funcn[i],"","",0.98,1.06);polyn[i] = new TF1(Form("poly%d"
                ,0),poly,0.994,1.06,3);
181         }
182         else if(i==2)
183         {
184            fHistTotal->Fit(funcn[i],"","",0.995,1.06);polyn[i] = new TF1(Form("poly%d"
                ,0),poly,0.994,1.06,3);
185         }
186         else if(i==3)
187         {
188            fHistTotal->Fit(funcn[i],"","",0.995,1.06);polyn[i] = new TF1(Form("poly%d"
                ,0),poly,0.994,1.06,3);
189         }
190         else if(i==4)
191         {
192            fHistTotal->Fit(funcn[i],"","",0.999,1.07);polyn[i] = new TF1(Form("poly%d"
                ,0),poly,0.999,1.06,3);
193         }
194         else if(i==5)
195         {
196            fHistTotal->Fit(funcn[i],"","",0.999,1.08);polyn[i] = new TF1(Form("poly%d"
                ,0),poly,0.999,1.06,3);
197         }
198         else if(i==6)
199         {
200            fHistTotal->Fit(funcn[i],"","",1.005,1.05);polyn[i] = new TF1(Form("poly%d"
                ,0),poly,1.005,1.05,3);
201         }
202         else if(i==7)
203         {
204            fHistTotal->Fit(funcn[i],"","",0.998,1.04);polyn[i] = new TF1(Form("poly%d"
                ,0),poly,0.998,1.06,3);
205         }
206         else if(i==8)
207         {
208            fHistTotal->Fit(funcn[i],"","",0.995,1.038);polyn[i] = new TF1(Form("poly%d"
                ,0),poly,0.995,1.04,3);
209         }
210         else
211         {fHistTotal->Fit(funcn[i],"","",0.995,1.055);polyn[i] = new TF1(Form("poly%d"
              ,0),poly,0.994,1.06,3);
212          }
213         polyn[i]->SetParameter(0,funcn[i]->GetParameter(3));
214         polyn[i]->SetParameter(1,funcn[i]->GetParameter(4));
215         polyn[i]->SetParameter(2,funcn[i]->GetParameter(5));
216         par[i]=(funcn[i]->GetParameter(0))*1000/0.492;
217         para[i]=(funcn[i]->GetParError(0))*1000/0.492;
218         mzero[i]=funcn[i]->GetParameter(2);
219
220         gamma[i]=funcn[i]->GetParameter(1);
221         mzeroe[i]=funcn[i]->GetParError(2);
222         gammae[i]=funcn[i]->GetParError(1);
223         hmpt->Fill(j,mzero[i]);
224         hgpt->Fill(j,gamma[i]);
225         ptn[i]=j;
```

```
226
227         binnmbr=hEfficiencyB1_100->GetXaxis()->FindBin(j);
228         efc[i]=hEfficiencyB1_100->GetBinContent(binnmbr);
229         bmin=axis->FindBin(1.01);
230         bmax=axis->FindBin(1.03);
231         integ[i]=fHistTotal->Integral(bmin,bmax);
232         fHistTotal->SetTitle(Form("%.2f #leq p_{T} < %.2fGeV/c",downn,upp));
233         fHistTotal->SetXTitle("m_{inv} (GeV/c^{2})");
234         fHistTotal->SetYTitle("Counts/(1 MeV/c^{2}");
235         if(i<16)
236         {
237         fpt[i]=(par[i]*7)/(44*j*ntot*0.1*efc[i]); //values for mean pT
238         fpte[i]= (para[i]*7)/(44*j*ntot*0.1);
239         fout<<j<<"\t"<<(par[i]*7)/(44*j*ntot*0.1*efc[i])<<"\t"<<0.05<<"\t"<<(para[i]*7)
                /(44*j*ntot*0.1)<<endl;
240         j=j+0.1; k=1;down=down+k;up=up+k;
241            if(i==15)
242            {j=2.1;down=21;up=22;}
243         }
244         else if(i<21)
245         {
246         fpt[i]=(par[i]*7)/(44*j*ntot*0.2*efc[i]); //values for mean pT
247         fpte[i]= (para[i]*7)/(44*j*ntot*0.2);
248         fout<<j<<"\t"<<(par[i]*7)/(44*j*ntot*0.2*efc[i])<<"\t"<<0.1<<"\t"<<(para[i]*7)
                /(44*j*ntot*0.2)<<endl;
249         j=j+0.2;k=2;down=down+k;up=up+k;
250            if(i==20)
251            {j=3.25;down=31;up=35;}
252         }
253         else if(i<23)
254         {
255            fpt[i]=(par[i]*7)/(44*j*ntot*0.5*efc[i]); //values for mean pT
256            fpte[i]= (para[i]*7)/(44*j*ntot*0.5);
257            fout<<j<<"\t"<<(par[i]*7)/(44*j*ntot*0.5*efc[i])<<"\t"<<0.25<<"\t"<<(para[i
                ]*7)/(44*j*ntot*0.5)<<endl;
258            j=j+0.5;k=5;down=down+k;up=up+k;
259            if(i==22)
260            {j=4.5;down=41;up=50;}
261
262         }
263         else if(i<29)
264         {
265            fpt[i]=(par[i]*7)/(44*j*ntot*1*efc[i]);   //values for mean pT
266            fpte[i]= (para[i]*7)/(44*j*ntot*1);
267            fout<<j<<"\t"<<(par[i]*7)/(44*j*ntot*1*efc[i])<<"\t"<<0.5<<"\t"<<(para[i]*7)
                /(44*j*ntot*1)<<endl;
268            j=j+1;k=10;down=down+k;up=up+k;
269
270
271         }
272
273            fHistTotal->Draw();
274            polyn[i]->Draw("same");
275            polyn[i]->SetLineColor(4);
276
277
278
279
280
281
282
283
284
285
```

```
286     }
287
288
289     double ex[30],dndpt=0,dndp=0,meanpt,meanpte,frace,dndpte=0,dndpe=0,meanfitte,dndy
            =0;
290     for(i=0;i<27;i++)
291     {  if(i<16)
292        ex[i]=0.05;
293        else if(i<21)
294        ex[i]=0.1;
295        else if(i<23)
296        ex[i]=0.2;
297        else if(i<29)
298        ex[i]=0.5;      }
299
300
301     TGraphErrors *MyGraph = new TGraphErrors(i,ptn,mzero,ex,mzeroe);
302     TGraphErrors *MyGraph2 = new TGraphErrors(i,ptn,gamma,ex,gammae);
303     MyGraph->GetXaxis()->SetTitle("p_{T} (GeV/c)");
304     MyGraph->GetYaxis()->SetTitle("m_{inv} (GeV/c^{2})");
305     MyGraph2->GetXaxis()->SetTitle("p_{T} (GeV/c)");
306     MyGraph2->GetYaxis()->SetTitle("Gamma (GeV/c^{2})");
307
308 //Mean pT loop
309     j=0.45;
310     for(i=0;i<16;i++)
311     {
312        dndpt=dndpt+j*j*fpt[i]*0.1;
313        dndpte=sqrt(dndpte*dndpte+(j*j*fpte[i]*0.1)*(j*j*fpte[i]*0.1));
314        dndp=dndp+j*fpt[i]*0.1;
315        dndy=dndy+(par[i]/ntot);
316        dndpe=sqrt(dndpe*dndpe+(fpte[i]*0.1)*(fpte[i]*0.1));
317        j=j+0.1;
318
319     }
320     j=2.1;
321     for(i=16;i<21;i++)
322     {
323        dndpt=dndpt+j*j*fpt[i]*0.2;
324        dndpte=sqrt(dndpte*dndpte+(j*j*fpte[i]*0.2)*(j*j*fpte[i]*0.2));
325        dndp=dndp+j*fpt[i]*0.2;
326        dndy=dndy+(par[i]/ntot);
327        dndpe=sqrt(dndpe*dndpe+(fpte[i]*0.2)*(fpte[i]*0.2));
328        j=j+0.2;
329
330     }
331     j=3.25;
332     for(i=21;i<23;i++)
333     {
334        dndpt=dndpt+j*j*fpt[i]*0.5;
335        dndpte=sqrt(dndpte*dndpte+(j*j*fpte[i]*0.5)*(j*j*fpte[i]*0.5));
336        dndp=dndp+j*fpt[i]*0.5;
337        dndy=dndy+(par[i]/ntot);
338        dndpe=sqrt(dndpe*dndpe+(fpte[i]*0.5)*(fpte[i]*0.5));
339        j=j+0.5;
340
341     }
342     j=4.5;
343     for(i=23;i<27;i++)
344     {
345        dndpt=dndpt+j*j*fpt[i]*1;
346        dndpte=sqrt(dndpte*dndpte+(j*j*fpte[i]*1)*(j*j*fpte[i]*1));
347        dndp=dndp+j*fpt[i]*1;
348        dndy=dndy+(par[i]/ntot);
```

```
349         dndpe=sqrt(dndpe*dndpe+(fpte[i]*1)*(fpte[i]*1));
350         j=j+1;
351
352     }
353     meanpt=dndpt/dndp;
354     meanpte=meanpt*sqrt((dndpte/dndpt)*(dndpte/dndpt)+(dndpe/dndp)*(dndpe/dndp));
355     frace=meanpte/meanpt;
356     cout<<"\nMean pT(only within data range) is= "<<meanpt<<" with error "<<meanpte<<
            endl;
357     TGraphErrors *MyGraph3 = new TGraphErrors("outputdata.txt");
358     MyGraph3->GetXaxis()->SetTitle("p_{T} (GeV/c)");
359     MyGraph3->GetYaxis()->SetTitle("d^{2}N/N2pip_{T}dydp_{T} (GeV/c)^{-1}");
360
361     TF1 *func = new TF1("ltf",ltf,0.,100,3);
362     TF1 *fitt = new TF1("ltf1",ltf,0.,100,3);
363     TCanvas *cn = new TCanvas("cn","", 400, 400);
364     cn->cd();
365     func->SetParameter(0,0.024);   func->SetParName(0,"dN/dy");
366     func->SetParameter(1,6);     func->SetParName(1,"n");
367     func->SetParameter(2,0.2);    func->SetParName(2,"T");
368     func->SetParameter(3,1.019);    func->SetParName(3,"Mzero");
369     MyGraph3->Fit(ltf,"","",0.,9);
370     fitt->SetParameter(0,func->GetParameter(0));
371     fitt->SetParameter(1,func->GetParameter(1));
372     fitt->SetParameter(2,func->GetParameter(2));
373     fitt->SetParameter(3,func->GetParameter(3));
374     TAxis *axisn = MyGraph3->GetXaxis();
375     bmin=axisn->FindBin(0.);
376     bmax=axisn->FindBin(0.4);
377     double meanfitt=fitt->Integral(bmin,bmax);
378     bmin=axisn->FindBin(8);
379     bmax=axisn->FindBin(10);
380     meanfitt+=fitt->Integral(bmin,bmax);
381     meanfitte=frace*meanfitt;
382     cout<<"\nMean pT(only within function range) is= "<<meanfitt<<" with error "<<
            meanfitte<<endl;
383     cout<<"\nMean pT is= "<<meanpt+meanfitt<<" with error "<<meanpte+meanfitte<<endl;
384     MyGraph3->Draw("A*");
385     //TFile *fFile = new TFile("ptplotdata.root","RECREATE");
386     //MyGraph3->Write();
387     MyGraph2->SetTitle("ptdata");
388     MyGraph2->SetName("ptdata");
389     TCanvas *c2 = new TCanvas("c2","", 800, 400);
390     c2->Divide(2,1);
391     c2->cd(1);
392     MyGraph->Draw("AP");
393
394
395     line->SetLineWidth(2);
396     line->Draw("same");
397     c2->cd(2);
398     MyGraph2->Draw("AP");
399     line1->Draw("same");
400
401
402
403
404
405 }
```

A.2 Macro for Pythia data

```
   frame
1  #include "TH1.h"
2  #include "TH2.h"
3  #include "TF1.h"
4  #include "TF2.h"
5  #include "TMath.h"
6  #include<math.h>
7  #include <fstream.h>
8  //Breit-Wigner function
9  Double_t mybw(Double_t* x, Double_t* par)
10 {
11   Double_t arg1 = 14.0/22.0; // 2 over pi
12   Double_t arg2 = par[1]*par[1]*par[2]*par[2]; //Gamma=par[1]  M=par[2]
13   Double_t arg3 = ((x[0]*x[0]) - (par[2]*par[2]))*((x[0]*x[0]) - (par[2]*par[2]));
14   Double_t arg4 = x[0]*x[0]*x[0]*x[0]*((par[1]*par[1])/(par[2]*par[2]));
15   return par[0]*arg1*arg2/(arg3 + arg4)+(par[3]+par[4]*x[0]);
16 }
17 Double_t mybwn(Double_t* x, Double_t* par)
18 {
19   Double_t arg1 = 7.0/44.0; // 1 over 2*pi
20   Double_t arg2 = par[1]; //Gamma=par[1]  A=par[0] M=par[2]
21   Double_t arg3 = (x[0] - par[2])*(x[0] - par[2]);
22   Double_t arg4 = (par[1]*par[1])/4;
23   return (par[0]*arg1*arg2)/(arg3 + arg4)+(par[3]+par[4]*x[0]+par[5]*x[0]*x[0]);
24   //return (par[0]*arg1*arg2)/(arg3 + arg4)+(par[3]+par[4]*x[0]);
25 }
26 Double_t poly(Double_t* x, Double_t* par)
27 {
28    return (par[0]+par[1]*x[0]+par[2]*x[0]*x[0]);
29 }
30 //Levy-Tsallis Function
31 Double_t ltf(Double_t* x, Double_t* par)
32 {
33   //par[0]=dN/dy par[1]=n par[2]=T x[0]=pT
34   Double_t pi=22/7,mt,mzero=1.019;
35   mt=sqrt(x[0]*x[0]+mzero*mzero);
36   Double_t arg1= par[0]*(par[1]-1)*(par[1]-2);
37   Double_t arg2= 2*pi*par[1]*par[2];
38   Double_t arg3= par[1]*par[2] + mzero*(par[1]-2);
39   Double_t arg4= pow(1+((mt-mzero)/(par[1]*par[2])), -1*par[1]);
40   return (arg1*arg4)/(arg2*arg3);
41
42 }
43 Double_t ltfm(Double_t* x, Double_t* par)
44 {
45   //par[0]=dN/dy par[1]=n par[2]=T x[0]=pT
46   Double_t pi=22/7,mt,mzero=1.019;
47   mt=sqrt(x[0]*x[0]+mzero*mzero);
48   Double_t arg1= par[0]*(par[1]-1)*(par[1]-2);
49   Double_t arg2= 2*pi*par[1]*par[2];
50   Double_t arg3= par[1]*par[2] + mzero*(par[1]-2);
51   Double_t arg4= pow(1+((mt-mzero)/(par[1]*par[2])), -1*par[1]);
52   return (x[0]*arg1*arg4)/(arg2*arg3);
53
54 }
55 Double_t ltfd(Double_t* x, Double_t* par)
56 {
57   //par[0]=dN/dy par[1]=n par[2]=T x[0]=pT
58   Double_t pi=22/7,mt,mzero=1.019;
59   mt=sqrt(x[0]*x[0]+mzero*mzero);
```

```
60    Double_t arg1= par[0]*(par[1]-1)*(par[1]-2);
61    Double_t arg2= 2*pi*par[1]*par[2];
62    Double_t arg3= par[1]*par[2] + mzero*(par[1]-2);
63    Double_t arg4= pow(1+((mt-mzero)/(par[1]*par[2])), -1*par[1]);
64    return (2*pi*x[0]*arg1*arg4)/(arg2*arg3);
65
66  }
67  void set(int num,double &p1,double &p2,double &p3,double &p4,double &p5)
68  {
69     if(num<27)
70     {
71        if(num==0)
72        {p1=10 ;p2=0.004 ;p3=10 ;p4=6 ;p5=-5 ;}
73        if(num==1)
74        {p1=133 ;p2= 0.004;p3=-3 ;p4=4 ;p5=-3 ;}
75        if(num==2)
76        {p1=123 ;p2=0.004 ;p3= -3 ;p4=2 ;p5=-1 ;}
77        if(num==3)
78        {p1=132;p2=0.004;p3=-4;p4=5;p5=-5;}
79        if(num==4)
80        {p1=123;p2=0.004;p3=-3;p4=6;p5=-3;}
81        if(num==5)
82        {p1=132;p2=0.004;p3=-3;p4=5;p5=-4;}
83        if(num==6)
84        {p1=123;p2=0.004;p3=-2;p4=4;p5=-2;}
85        if(num==7)
86        {p1=132;p2=0.004;p3=-3;p4=5;p5=-4;}
87        if(num==8)
88        {p1=132;p2=0.004;p3=-2;p4=4;p5=-2;}
89        if(num==9)
90        {p1=132;p2=0.004;p3=-3;p4=5;p5=-4;}
91        if(num==10)
92        {p1=124;p2=0.005;p3=-1;p4=2;p5=-1;}
93        if(num==11)
94        {p1=125;p2=0.004;p3=5;p4=-5;p5=4;}
95        if(num==12)
96        {p1=124;p2=0.004;p3=-20;p4=10;p5=-5;}
97        if(num==13)
98        {p1=134;p2=0.004;p3=-5;p4=4;p5=-3;}
99        if(num==14)
100       {p1=125;p2=0.004;p3=3;p4=1;p5=-4;}
101       if(num==15)
102       {p1=134;p2=0.004;p3=-6;p4=1;p5=2;}
103       if(num==16)
104       {p1=125;p2=0.004;p3=1;p4=2.5;p5=-3;}
105       if(num==17)
106       {p1=132;p2=0.004;p3=-4;p4=5;p5=1;}
107       if(num==18)
108       {p1=123;p2=0.004;p3=-4;p4=9;p5=8;}
109       if(num==19)
110       {p1=132;p2=0.004;p3=-12;p4=-13;p5=9;}
111       if(num==20)
112       {p1=124;p2=0.004;p3=-4;p4=5.2;p5=9.8;}
113       if(num==21)
114       {p1=26.9;p2=0.004;p3=4;p4=-1.4;p5=4;}
115       if(num==22)
116       {p1=125;p2=0.004;p3=11;p4=2;p5=1;}
117       if(num==23)
118       {p1=125;p2=0.004;p3=11;p4=2;p5=1;}
119       if(num==24)
120       {p1=134;p2=0.004;p3=11;p4=2;p5=-1;}
121       if(num==25)
122       {p1=124;p2=0.005;p3=11;p4=6;p5=-6;}
123       if(num==26)
```

```
124         {p1=127;p2=0.005;p3=11;p4=6;p5=-6;}
125 
126            }
127     else
128     {
129        cout<<"\nEnter Area, Sigma, A, B, C: "<<endl;
130        cin>>p1>>p2>>p3>>p4>>p5;
131 
132     }
133 }
134 void twodhist()
135 {
136     gStyle->SetOptStat(0);
137     gStyle->SetOptFit(1111);
138     gStyle->SetStatColor(0);
139     gStyle->SetCanvasColor(0);
140     gStyle->SetPadColor(0);
141     gStyle->SetPadBorderMode(0);
142     gStyle->SetCanvasBorderMode(0);
143     gStyle->SetFrameBorderMode(0);
144     gStyle->SetFillColor(0);
145     gStyle->SetLineColor(1);
146     gStyle->SetTitleTextColor(1);
147     gStyle->SetTitleColor(1);
148     TFile *f = new TFile("TwoDHistnew.root");
149     ofstream fout("output.txt");
150     TH2F *hmpt = new TH2F("hmpt","Mass vs pT ",120,0.98,1.1,100,0.,10.);
151     TH2F *hgpt = new TH2F("hgpt","Gamma vs pT ",100,0.,0.01,100,0.,10.);
152     TF1 *func1 = new TF1("mybw",mybw,0.98,1.1,5);
153     TF1 *funcn = new TF1("mybwn",mybwn,0.98,1.1,6);
154     TF1 *polyn = new TF1("poly",poly,0.995,1.05,3);
155     hmpt->SetXTitle("p_{T} (GeV/c^{1})");
156     hmpt->SetYTitle("m_{inv} (GeV/c^{2})");
157     hgpt->SetXTitle("p_{T} (GeV/c^{1})");
158     hgpt->SetYTitle("Gamma");
159     TLine *line = new TLine(0.3,1.019455,8,1.019455);
160     TLine *line1 = new TLine(0.3,0.00426,8,0.00426);
161     Float_t par[30],para[30];
162     int bmin,bmax;
163     double integral,integral1,ratio,ntot=20000000,mzero[30],gamma[30],mzeroe[30],
            gammae[30],ptn[30];
164     //Settings for funcn
165     funcn->SetParameter(2,1.019);    funcn->SetParName(2,"mean");
166     funcn->SetParLimits(2,1.015,1.025);
167     funcn->SetParLimits(1,0.0032,0.01);
168     TH2D *h = (TH2D*)f->Get("fTHnSSig");
169     TH2D *hbg = (TH2D*)f->Get("fTHnSLikeSignPP");
170     TH2D *hbgn = (TH2D*)f->Get("fTHnSLikeSignMM");
171     hbg->Add(hbgn,1);
172     int k=0,plot=0,i,up=5,down=5,iti=27;
173     double integ[27],j=0.45,fpt[27],fpte[27];
174     TH1D *pt[27];
175     TH1D *ptbg[27];
176     TCanvas *c = new TCanvas("c","", 800, 1200);
177     TCanvas *c11 = new TCanvas("c11","", 800, 1200);
178     TCanvas *c22 = new TCanvas("c22","", 800, 1200);
179     TCanvas *c33 = new TCanvas("c33","", 800, 1200);
180     TCanvas *d = new TCanvas("d","", 800, 800);
181     c->Divide(2,3);
182     c11->Divide(2,3);
183     c22->Divide(2,3);
184     c33->Divide(2,3);
185     d->Divide(2,2);
186     float upp,downn;
```

```
187     double para1,para2,para3,para4,para5;
188          for(i=0;i<iti;i++)
189     {  if(i<6)
190        c->cd(i+1);
191        else if(i<12)
192        c11->cd(i-5);
193        else if(i<18)
194        c22->cd(i-11);
195        else if(i<24)
196        c33->cd(i-17);
197        else
198        {d->cd(i-15);
199        }
200        upp=up;
201        upp=upp/10;
202        downn=down;
203        downn=(downn-1)/10;
204        pt[i]= h->ProjectionX(Form("%f #leq p_{T} < %f GeV/c",downn,upp),down,up);
205        pt[i]->SetYTitle("Counts/(1 Mev/c^{2}");
206        pt[i]->SetXTitle("m_{inv} (GeV/c^{2})");
207        pt[i]->Sumw2();
208        pt[i]->Rebin(2);
209        ptbg[i]= hbg->ProjectionX(Form("Bg_%d0.4<=pt<0.6",i),down,up);
210        ptbg[i]->Sumw2();
211        ptbg[i]->Rebin(2);
212        TAxis *axis = pt[i]->GetXaxis();
213        bmin=axis->FindBin(1.04);
214        bmax=axis->FindBin(1.06);
215        integral = pt[i]->Integral(bmin,bmax);
216        integral1 = ptbg[i]->Integral(bmin,bmax);
217        ratio= integral/integral1;
218        ptbg[i]->Scale(ratio);
219        pt[i]->Add(ptbg[i],-1);
220        set(i,para1,para2,para3,para4,para5);
221        funcn->SetParameter(0,para1);
222        funcn->SetParameter(1,para2);
223        funcn->SetParameter(3,para3);
224        funcn->SetParameter(4,para4);
225        funcn->FixParameter(5,0);
226        pt[i]->Fit(funcn,"","",0.994,1.06);
227        polyn->SetParameter(0,funcn->GetParameter(3));
228        polyn->SetParameter(1,funcn->GetParameter(4));
229        polyn->SetParameter(2,funcn->GetParameter(5));
230        par[i]=(funcn->GetParameter(0))*1000/(2*0.492);
231        para[i]=(funcn->GetParError(0))*1000/(2*0.492);
232        mzero[i]=funcn->GetParameter(2);
233        gamma[i]=funcn->GetParameter(1);
234        mzeroe[i]=funcn->GetParError(2);
235        gammae[i]=funcn->GetParError(1);
236        ptn[i]=j;
237        hmpt->Fill(ptn[i],mzero[i]);
238        hgpt->Fill(ptn[i],gamma[i]);
239        bmin=axis->FindBin(1.01);
240        bmax=axis->FindBin(1.03);
241        integ[i]=pt[i]->Integral(bmin,bmax);
242        pt[i]->SetTitle(Form("%.2f #leq p_{T} < %.2fGeV/c",downn,upp));
243        pt[i]->SetXTitle("m_{inv} (GeV/c^{2})");
244        pt[i]->SetYTitle("Counts/(1 MeV/c^{2}");
245        if(i<16)
246        {
247        fpt[i]=(par[i]*7)/(44*j*ntot*0.1);  //values for mean pT
248        fpte[i]= (para[i]*7)/(44*j*ntot*0.1);
249        fout<<j<<"\t"<<(par[i]*7)/(44*j*ntot*0.1)<<"\t"<<0.05<<"\t"<<(para[i]*7)/(44*j*
               ntot*0.1)<<endl;
```

```
250         j=j+0.1; k=1;down=down+k;up=up+k;
251            if(i==15)
252            {j=2.1;down=21;up=22;}
253         }
254         else if(i<21)
255         {
256         fpt[i]=(par[i]*7)/(44*j*ntot*0.2);  //values for mean pT
257         fpte[i]= (para[i]*7)/(44*j*ntot*0.2);
258         fout<<j<<"\t"<<(par[i]*7)/(44*j*ntot*0.2)<<"\t"<<0.1<<"\t"<<(para[i]*7)/(44*j*
                ntot*0.2)<<endl;
259         j=j+0.2;k=2;down=down+k;up=up+k;
260            if(i==20)
261            {j=3.25;down=31;up=35;}
262         }
263         else if(i<23)
264         {  fpt[i]=(par[i]*7)/(44*j*ntot*0.5);  //values for mean pT
265            fpte[i]= (para[i]*7)/(44*j*ntot*0.5);
266            fout<<j<<"\t"<<(par[i]*7)/(44*j*ntot*0.5)<<"\t"<<0.25<<"\t"<<(para[i]*7)
                   /(44*j*ntot*0.5)<<endl;
267            j=j+0.5;k=5;down=down+k;up=up+k;
268            if(i==22)
269            {j=4.5;down=41;up=50;}
270
271         }
272         else if(i<29)
273         {  fpt[i]=(par[i]*7)/(44*j*ntot*1); //values for mean pT
274            fpte[i]= (para[i]*7)/(44*j*ntot*1);
275            fout<<j<<"\t"<<(par[i]*7)/(44*j*ntot*1)<<"\t"<<0.5<<"\t"<<(para[i]*7)/(44*j*
                   ntot*1)<<endl;
276            j=j+1;k=10;down=down+k;up=up+k;
277
278
279         }
280            polyn->SetLineColor(4);
281            pt[i]->Draw();
282            polyn->Draw("same");
283
284
285      }
286      TCanvas *coon = new TCanvas("coon","", 800, 400);
287      coon->Divide(2,1);
288      coon->cd(1);
289      hmpt->Draw("AP");
290      coon->cd(2);
291      hgpt->Draw("AP");
292
293      double ex[27],dndpt=0,dndp=0,meanpt,meanpte,frace,dndpte=0,dndpe=0,meanfitte,dndy
            =0;
294      for(i=0;i<iti;i++)
295      {
296         if(i<16)
297         ex[i]=0.05;
298         else if(i<21)
299         ex[i]=0.1;
300         else if(i<24)
301         ex[i]=0.25;
302         else
303         ex[i]=0.5;
304      }
305      TGraphErrors *MyGraph = new TGraphErrors(i,ptn,mzero,ex,mzeroe);
306      TGraphErrors *MyGraph2 = new TGraphErrors(i,ptn,gamma,ex,gammae);
307      MyGraph->GetXaxis()->SetTitle("p_{T} (GeV/c)");
308      MyGraph->GetYaxis()->SetTitle("m_{inv} (GeV/c^{2})");
309      MyGraph2->GetXaxis()->SetTitle("p_{T} (GeV/c)");
```

```
310     MyGraph2->GetYaxis()->SetTitle("Gamma");
311
312  //Mean pT loop
313     j=0.45;
314     for(i=0;i<16;i++)
315     {
316        dndpt=dndpt+j*j*fpt[i]*0.1;
317        dndpte=sqrt(dndpte*dndpte+(j*j*fpte[i]*0.1)*(j*j*fpte[i]*0.1));
318        dndp=dndp+j*fpt[i]*0.1;
319        dndy=dndy+(par[i]/ntot);
320        dndpe=sqrt(dndpe*dndpe+(fpte[i]*0.1)*(fpte[i]*0.1));
321        j=j+0.1;
322
323     }
324     j=2.1;
325     for(i=16;i<21;i++)
326     {
327        dndpt=dndpt+j*j*fpt[i]*0.2;
328        dndpte=sqrt(dndpte*dndpte+(j*j*fpte[i]*0.2)*(j*j*fpte[i]*0.2));
329        dndp=dndp+j*fpt[i]*0.2;
330        dndy=dndy+(par[i]/ntot);
331        dndpe=sqrt(dndpe*dndpe+(fpte[i]*0.2)*(fpte[i]*0.2));
332        j=j+0.2;
333
334     }
335     j=3.25;
336     for(i=21;i<23;i++)
337     {
338        dndpt=dndpt+j*j*fpt[i]*0.5;
339        dndpte=sqrt(dndpte*dndpte+(j*j*fpte[i]*0.5)*(j*j*fpte[i]*0.5));
340        dndp=dndp+j*fpt[i]*0.5;
341        dndy=dndy+(par[i]/ntot);
342        dndpe=sqrt(dndpe*dndpe+(fpte[i]*0.5)*(fpte[i]*0.5));
343        j=j+0.5;
344
345     }
346     j=4.5;
347     for(i=23;i<26;i++)
348     {
349        dndpt=dndpt+j*j*fpt[i]*1;
350        dndpte=sqrt(dndpte*dndpte+(j*j*fpte[i]*1)*(j*j*fpte[i]*1));
351        dndp=dndp+j*fpt[i]*1;
352        dndy=dndy+(par[i]/ntot);
353        dndpe=sqrt(dndpe*dndpe+(fpte[i]*1)*(fpte[i]*1));
354        j=j+1;
355
356     }
357     meanpt=dndpt/dndp;
358     meanpte=meanpt*sqrt((dndpte/dndpt)*(dndpte/dndpt)+(dndpe/dndp)*(dndpe/dndp));
359     cout<<"\nMean pT is= "<<meanpt<<" with error "<<meanpte<<endl;
360     TGraphErrors *MyGraph3 = new TGraphErrors("output.txt");
361     MyGraph3->GetXaxis()->SetTitle("p_{T} (GeV/c)");
362     MyGraph3->GetYaxis()->SetTitle("d^{2}N/N2pip_{T}dydp_{T} (GeV/c)^{-1}");
363
364     TF1 *func = new TF1("ltf",ltf,0.,8,3);
365     TF1 *fitt = new TF1("ltf1",ltf,0.,8,3);
366     TF1 *fitm = new TF1("ltfm",ltfm,0.,100,3);
367     TF1 *fitd = new TF1("ltfd",ltfd,0.,100,3);
368     func->SetParameter(0,0.07);    func->SetParName(0,"dN/dy");
369     func->SetParameter(1,6);      func->SetParName(1,"n");
370     func->SetParameter(2,0.17);     func->SetParName(2,"T");
371     func->SetParameter(3,1.019);     func->SetParName(3,"Mzero");
372     func->SetParameter(4,0.1);     func->SetParName(4,"B");
373     MyGraph3->Fit(ltf,"","",0.,9);
```

```
374     fitt->SetParameter(0,func->GetParameter(0));
375     fitt->SetParameter(1,func->GetParameter(1));
376     fitt->SetParameter(2,func->GetParameter(2));
377     fitt->SetParameter(3,func->GetParameter(3));
378     fitm->SetParameter(0,func->GetParameter(0));    fitm->SetParName(0,"dN/dy");
379     fitm->SetParameter(1,func->GetParameter(1));    fitm->SetParName(1,"n");
380     fitm->SetParameter(2,func->GetParameter(2));    fitm->SetParName(2,"T");
381     fitm->SetParameter(3,func->GetParameter(3));    fitm->SetParName(3,"Mzero");
382     fitd->SetParameter(0,func->GetParameter(0));
383     fitd->SetParameter(1,func->GetParameter(1));
384     fitd->SetParameter(2,func->GetParameter(2));
385     fitd->SetParameter(3,func->GetParameter(3));
386     dndy=dndy+fitd->Integral(0.1,0.39)+fitd->Integral(8,10);
387     cout<<"\nMean pT is= "<<meanpt+((fitm->Integral(0.01,0.39)+fitm->Integral(8,10))/(
            fitt->Integral(0.01,0.39)+fitt->Integral(8,10)))<<endl;
388     cout<<"\nDnDy is= "<<dndy<<endl;
389     TCanvas *c1 = new TCanvas("c1","", 400, 400);
390     c1->cd();
391     TFile *fFile = new TFile("ptpythia.root","RECREATE");
392     MyGraph3->Draw("A*");
393
394     MyGraph3->SetTitle("ptpythia");
395     MyGraph3->SetName("ptpythia");
396     MyGraph3->Write();
397
398     TCanvas *c2 = new TCanvas("c2","", 800, 400);
399     c2->Divide(2,1);
400     c2->cd(1);
401     //hmpt->Draw();
402     MyGraph->Draw("AP");
403
404     line->SetLineWidth(2);
405     line->Draw("same");
406     c2->cd(2);
407     //hgpt->Draw();
408     MyGraph2->Draw("AP");
409     line1->Draw("same");
410     //TCanvas *c2 = new TCanvas("c2","", 400, 400);
411     //c2->cd(0);
412
413     //MyGraph2->Write();
414
415
416 }
```