Study of Elliptic Flow of Identified Charge Particles using Relativistic Heavy-ion Collision Models using Different flow Methods

Summer Training Project Report

Session : 2010 - 2011

Submitted By :- **Vipul Bairathi**

University of Rajasthan,Jaipur

Name of The Supervisor :- **Dr. Bedangadas Mohanty**

Variable Energy Cyclotron Centre,Kolkata

Signature of Supervisor :

Signature of Student :

July 29, 2011

Index

Introduction to Elliptic Flow The Elliptic Flow is described as the azimuthal momentum space anisotropy of produced particles from non-central heavy ion collisions in the plane transverse to the beam direction. Anisotropic flow leads to the azimuthal distribution of particles produced in non-central collisions, measured with respect to the reaction plane being not flat. The particle azimuthal distributions relative to the reaction plane can be written in the form of Fourier series as

$$\frac{dN}{d\phi} = \frac{1}{2\pi}\left(1 + \sum_{n=1}^{\infty} 2v_n \cos\left(n(\phi - \psi_{rp})\right)\right)$$

where ψ_{rp} is the reaction plane angle defined by the impact parameter vector in the transverse plane.

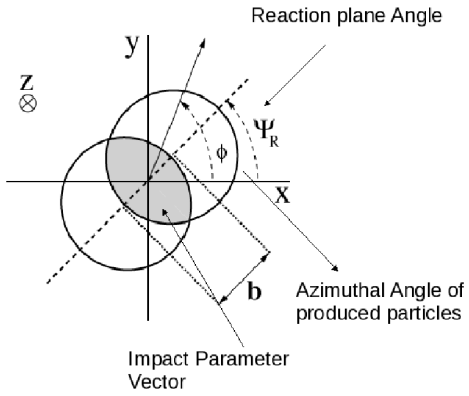


Figure 1: Non central nucleus-nucleus collision with Azimuthal angle ϕ, Reaction plane angle ψ_R and Impact parameter vector

v_n coefficients are given by

$$v_n = < \cos(n(\phi - \psi_{rp})) >$$

The first two coefficients v_1 and v_2 are called Directed and Elliptic Flow respectively. The second harmonics of this Fourier series is termed as Elliptic Flow and is denoted by v_2 is given as

$$v_2 = < \cos(2(\phi_i - \psi_{rp})) >$$

ϕ_i = the angle of azimuth of the emitted particle

average <> is over all particles and for all events.

The elliptic anisotropy present in relativistic heavy ion collisions leads to a collective motion(flow)

of the particles produced. For non- central collisions, the geometrical anisotropy of the almond shaped overlap region of the colliding nuclei causes a larger pressure gradient in the reaction plane than in the direction orthogonal to it. This inital spatial anisotropy is converted into a momentum anisotropy through interaction among the constituents, which can be observed in the particle azimuthal distribution. This momentum anisotropy leads to the elliptic flow of produced particles.

different methods are used for the analysis of elliptic flow:

1. **Standard Event Plane Method**
2. **Scalar Product Method**
3. **Cumulant Method**
4. **Lee Yang Zeros Method**

In my project work, I have used Standard Event Plane Method and Q-Cumulant Method which is discussed in section 2 and 3 respectively. In section 4, I present conclusions from above methods and why elliptic flow is very important in heavy ion collisions at high energies as well as low energies. Finally,Comparison in above methods is discussed in section 5. I have used AMPT Model and UrQMD Model to analyze the Elliptic flow, which is introduced in Appendix A, Appendix B respectively.

Standard Event Plane Method

Description of Method

Reaction Plane is the plane spanned by the impact parameter vector in transverse plane with beam direction (Z- axis).

Practically it is impossible to determined true reaction plane as we don't know impact parameter vector. So we use azimuthal distribution of produced particles to estimate the reaction plane. This estimated reaction plane is called event plane as we have finite number of detected particles in limited resolution of the angle measured. So the method use to determine the anisotropic flow from the determination of event plane is know as Event plane Method. The method is summarize as follows:

1. The dependence on the particle emission azimuthal angle measured with respect to the reaction plane is given by

$$E\frac{d^3N}{d^3\phi} = \frac{1}{2\pi}\frac{d^2N}{P_T dP_T dy}\left(1 + \sum_{n=1}^{\infty} 2v_n \cos\left(n(\phi - \psi_{rp})\right)\right)$$

2. The reaction plane ψ for each event can be estimated using this azimuthal distribution of particle, which is given by (for nth harmonics)

$$\psi_n = \arctan\left(\frac{\sum_i \sin(n\phi_i)}{\sum_i \cos(n\phi_i)}\right)/n$$

   where sum goes over all i particles used in the event plane determination and the weight w_i are chosen in such a way to make the reaction plane resolution the best that is possible, it could be $\phi, P_T or \eta$. This event plane angle ψ_n is in the range $0 \leq \psi_n \leq \frac{2\pi}{n}$. For n = 2 it is $0 \leq \psi_n \leq \pi$.

3. After determination of event plane angle for each event, we calculate second harmonic of Fourier expansion of azimuthal distribution of produced particles with respect to this event plane.

4. Now average this second harmonics over all events and for all particles for each P_T(Transverse Momentum).

5. Plot $< v_2 >$ as a function of P_T to analyze the flow.

In my project for study of elliptic flow From Event Plane Method using simulated data, I use AMPT Model and UrQMd as event generator which is discussed in Appendix A and Appendix B, respectively.

Results from Event Plane Method (Along with plots) Various Plots From Standard Event Plane Method for different Model at Au+Au $\sqrt{S_{NN}} = 200$ A GeV

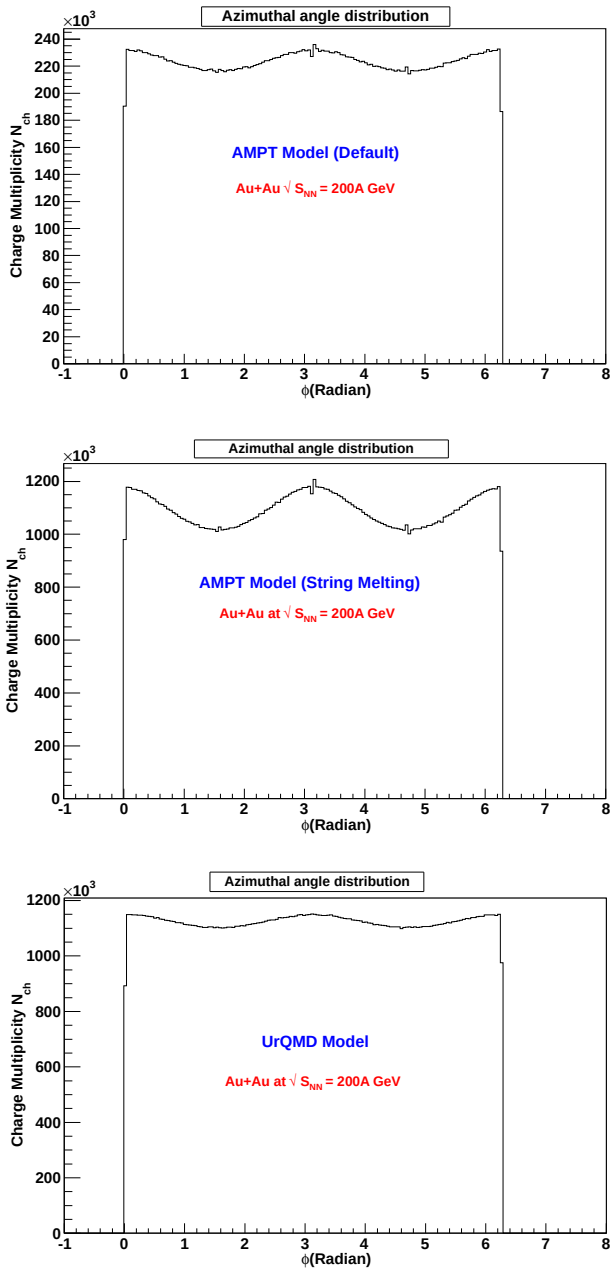
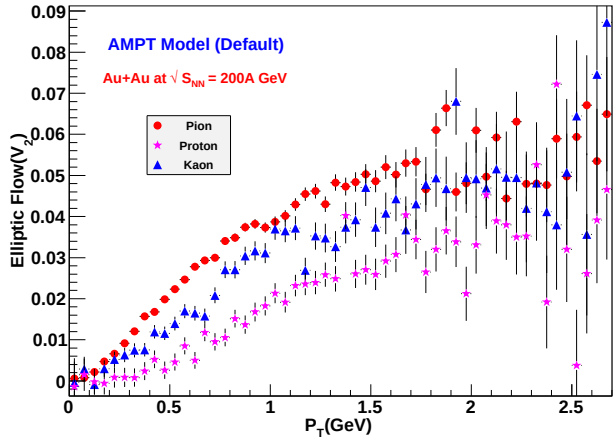
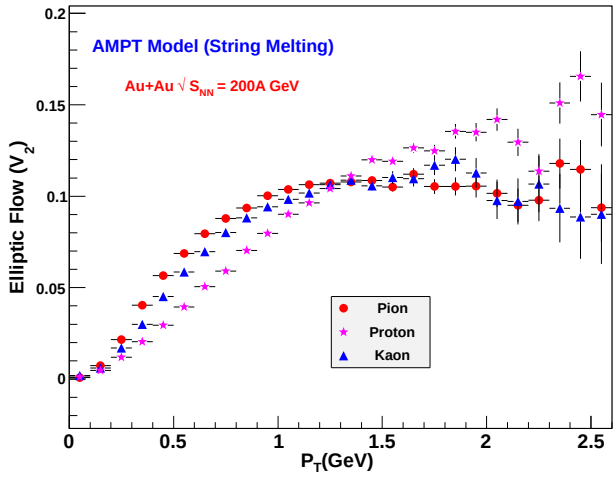


Figure 2: Azimuthal distribution of observed charge particles for Au+Au collision at $\sqrt{S_{NN}} = 200$ GeV for AMPT Model(Default), AMPT Model(String Melting) and UrQMD Model with X axis as ϕ in radian and Y axis as Number of charge particles





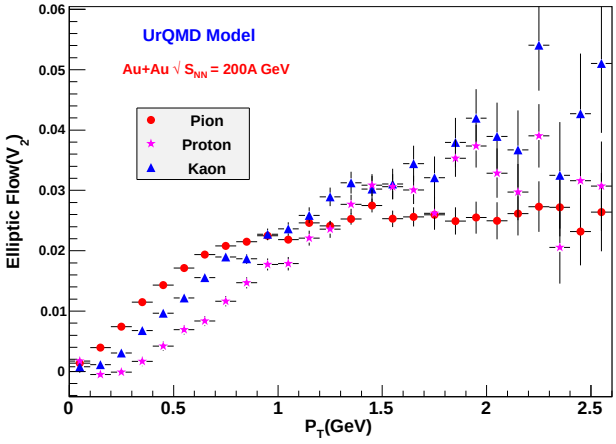


Figure 3: Elliptic Flow as a function of P_T for charge Pions,Kaons and Protons for Au+Au collision at $\sqrt{S_{NN}} = 200$ GeV for AMPT Model(Default), AMPT Model(String Melting) and UrQMD Model with X axis as Transverse Momentum P_T in GeV and Y axis as Average Elliptic Flow v_2

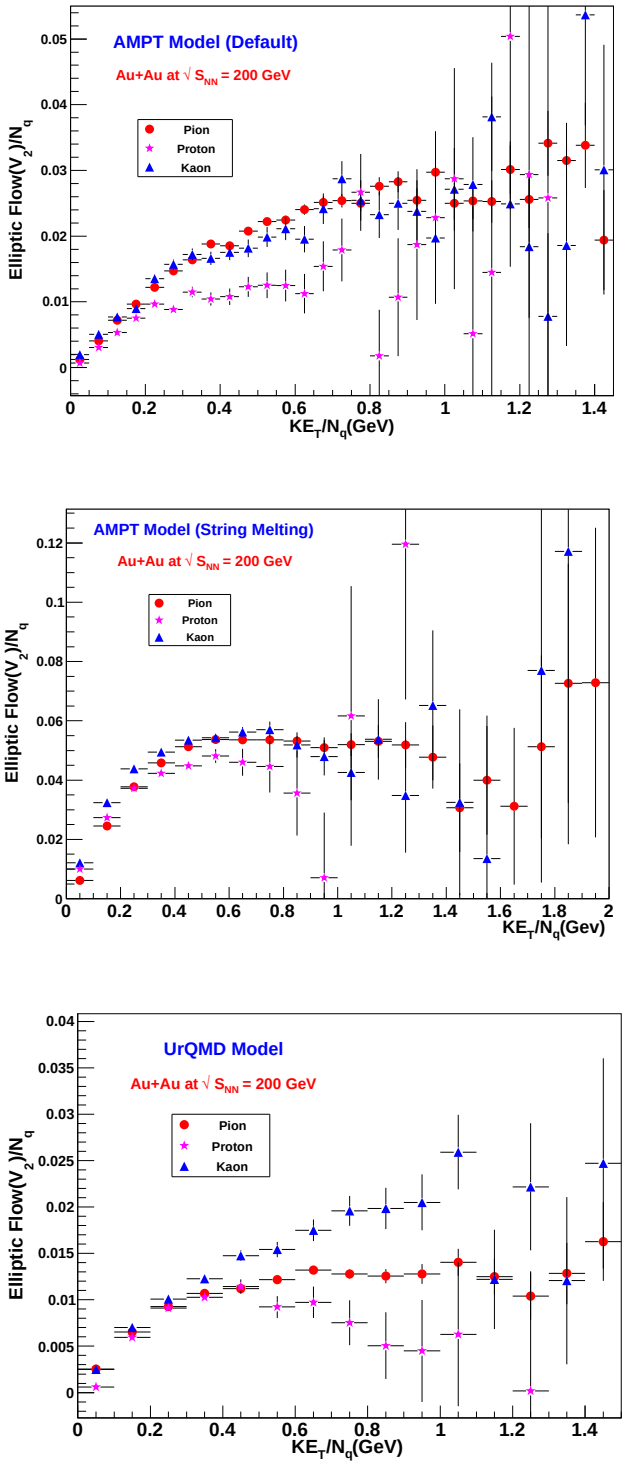


Figure 4: Elliptic Flow as a function of KE_T with number of constituents Quark (N_Q) for charge Pions,Kaons and Protons for Au+Au collision at $\sqrt{S}_{NN} = 200$ GeV for AMPT Model(Default), AMPT Model(String Melting) and UrQMD Model with X axis as Transverse Kinetic Energy $KE_T = M_T - M_0$ in GeV and Y axis as Average Elliptic Flow v_2

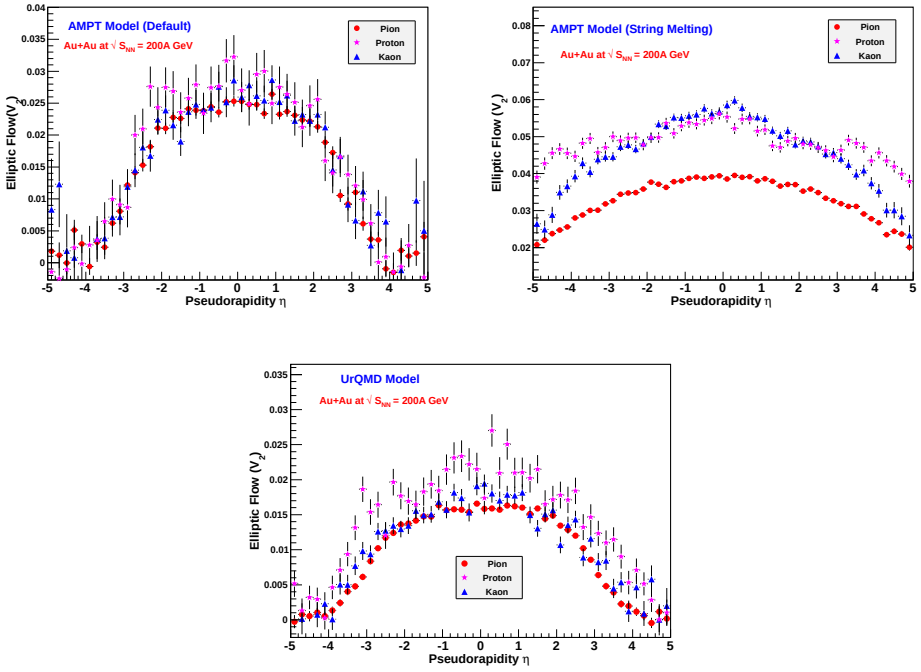


Figure 5: Elliptic Flow as a function of η for charge Pions,Kaons and Protons for Au+Au collision at $\sqrt{S_{NN}} = 200$ GeV for AMPT Model(Default), AMPT Model(String Melting) and UrQMD Model with X axis as Pseudorapidity η and Y axis as Average Elliptic Flow v_2

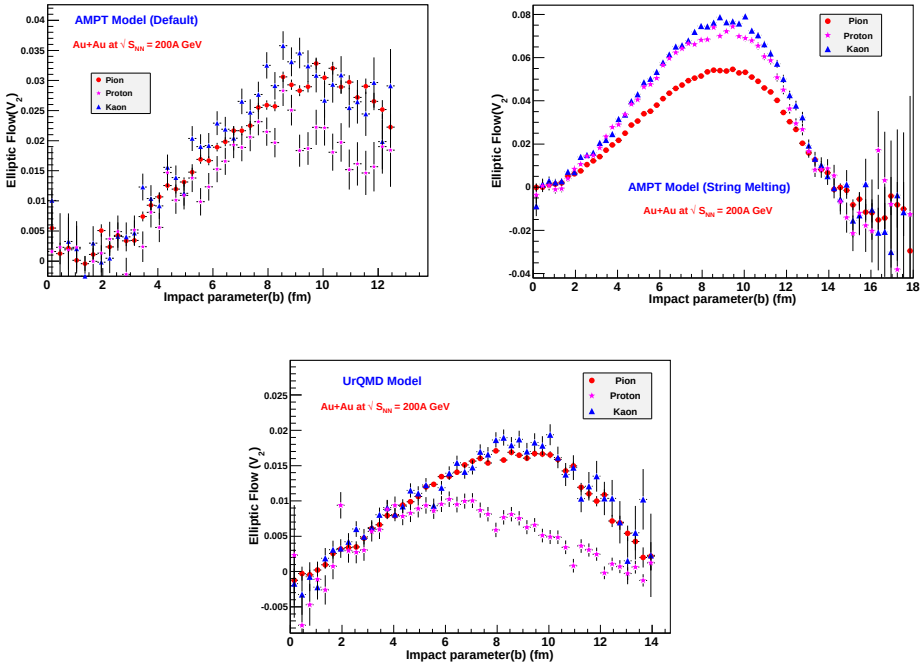


Figure 6: Elliptic Flow as a function of Impact Parameter for charge Pions,Kaons and Protons for Au+Au collision at $\sqrt{S_{NN}} = 200$ GeV for AMPT Model(Default), AMPT Model(String Melting) and UrQMD Model with X axis as Impact Parameter (b) in fm and Y axis as Average Elliptic Flow v_2

From the plots we see that:

1. Figure 2: Azimuthal distribution of produced particles

    - From figure we can see that the azimuthal distribution of produced particles is not flat and it looks like a cosine function. Amplitude of this function is more For AMPT String Melting Model as the value v_2 is more compare to AMPT Default and UrQMD Model.

2. Figure 3: Elliptic flow as a function transverse momentum

    - For AMPT Default Model v_2 as a function of P_T for Range (0 to 2.5 GeV) shows mass ordering i.e. flow of baryons (Protons) is low compare to measons(Pions and Kaons)
    - For AMPT String Melting Model v_2 has mass dependence for low $P_T(< 1.5)$ GeV but beyond > 1.5 GeV flow of baryons is more than that of measons.
    - For UrQMD Model same result is found as AMPT String Melting but the difference is that after $P_T > 1.5$ GeV flow of Kaons and protons are same, which is more than the flow of Pions.

3. Figure 4: Number of constituents quark (NCQ) scaling of Elliptic flow with transverse kinetic energy

    - From figure it can be observed that For AMPT Default Model has no NCQ scaling and flow of baryons and measons are different.
    - For AMPT String Melting Model scaling is observed therefore flow of baryons and measons are same.
    - For UrQMD Model same result is found as AMPT String Melting but scaling up to to $KE_T = .5$ GeV and after that there in no scaling.

4. Figure 5: Elliptic flow as a function of Pseudorapidity

    - In figure we see that peaks appears at Pseudorapidity range -1 to +1 i.e. in this centrality region flow is maximum.

5. Figure 6: Elliptic flow as a function of Impact parameter

    - From figure it can be observed that in central Au + Au collisions the charge multiplicity reaches the maximum at zero impact parameter, where the nuclear overlap region is nearly symmetric thus v2 approaches zero .In the middle peripheral collisions, although the charge multiplicity is going down, the v2 is large because of the strong asymmetry of the nuclear overlap region (large impact parameter). In the most peripheral collisions, the asymmetry of the nuclear overlap region is very strong, while the v2 is going down because the multiplicity is too low to generate a pressure gradient. When the impact parameter is around the radius of the colliding nucleus, charge multiplicity is not so small, and the asymmetry of the nuclear overlap region is considerably strong, so v2 approaches its maximum.

Q-Cumulant Method

Description of Method An another method for the analysis of anisotropic flow is cumulant method. Since the reaction plane is not known experimentally, the anisotropic flow is estimated using multiparticle azimuthal correlations between the observed particles. In this method cumulants are expressed in terms of Q-Vector(Flow Vector) evaluated for different harmonics, so it is known as Q-Cumulants Method.

Here I use only two particle and four particle azimuthal correlation for the analysis of elliptic flow. In the case of 2-particle azimuthal correlations the correlator is proportional to $< v_n^2 >$. This can be seen by:

$$\langle\langle e^{in(\phi_1-\phi_2)}\rangle\rangle = \langle\langle e^{in(\phi_1-\psi_R-(\phi_2-\psi_R))}\rangle\rangle$$

$$= \langle\langle e^{in(\phi_1-\psi_R)}\rangle\langle e^{-in(\phi_2-\psi_R)}\rangle\rangle$$

$$= \langle v_n^2\rangle$$

Where double brackets denotes a statistical average over all particles and all events. The Q-Vector (Flow Vector) evaluated in harmonic n is defined as:

$$Q_n \equiv \sum_{i=1}^{M} e^{in\phi_i}$$

Where M is the total number of reference particle in particular event.

Estimating Reference Flow from Q-Cumulants

To obtain 2nd order Q-Cumulant:

$$|Q_n|^2 = \sum_{i,j=1}^{M} e^{in(\phi_i-\phi_j)}$$

The two summing indices i and j can be either the same or different in the above relation.When the indices are the same then it is auto correlation between particle itself. So it follows that in the decomposition of $|Q_n|^2$ we have 2-particle adn 1-particle contributions.hence

$$|Q_n|^2 = \langle 2\rangle M_{C_2} 2! + 1.M$$

From this we obtain the single-event average 2- particle azimuthal correlation i.e.

$$< 2 >= \frac{|Q_n|^2 - M}{M(M-1)}$$

To get all event average 2- particle azimuthal correlation then average $< 2 >$ for all events thus the 2nd order Q-Cumulant is than simply written as :

$$c_n\{2\} = \langle\langle 2\rangle\rangle = \frac{\sum_{i=1}^{N}\left(|Q_n|_i^2 - M_i\right)}{\sum_{i=1}^{N} M_i(M_i-1)}$$

Here M_i is the number of reference particles in i-th event and N is total number of events.

To obtain 4nd order Q-Cumulant:

$$|Q_n|^4 = \sum_{i,j,k,l=1}^{M} e^{in(\phi_i+\phi_j-\phi_k-\phi_l)}$$

The four summing indices are the same leads to auto correlation and with all different indices i, j, k, l leads to 4- particle correlation. The single-event average 4- particle correlation defined as:

$$< 4 >= \frac{|Q_n|^4 + |Q_{2n}|^2 - 2.Re\{Q_{2n}Q_n^*Q_n^*\}}{M(M-1)(M-2)(M-3)} - 2\frac{2(M-2).\,|Q_n|^2 - M(M-3)}{M(M-1)(M-2)(M-3)}$$

The final event average 4- particle azimuthal correlation called the 4th order Q-Cumulant is than written as :

$$c_n\{4\} = \langle\langle 4\rangle\rangle - 2.\langle\langle 2\rangle\rangle^2 = \frac{\sum_{i=1}^{N}\left(|Q_n|_i^4 + |Q_{2n}|_i^2 - 2.Re\{Q_{2n}Q_n^*Q_n^*\}_i\right)}{\sum_{i=1}^{N} M_i(M_i-1)(M_i-2)(M_i-3)}$$

$$-2\frac{\sum_{i=1}^{N}\left(2(M_i-2).\,|Q_n|_i^2 - M_i(M_i-3)\right)}{\sum_{i=1}^{N} M_i(M_i-1)(M_i-2)(M_i-3)}$$

$$-2\left[\frac{\sum_{i=1}^{N}\left(|Q_n|_i^2 - M_i\right)}{\sum_{i=1}^{N} M_i(M_i-1)}\right]^2$$

Here $|Q_n|_i$ and $|Q_{2n}|_i$ are moduli of Q-vector evaluated in harmonics n and 2n, respectively, in the i-th event. M_i is the number of reference particles in i-th event and N is total number of events.

For the case of detector with uniform acceptance and when only flow correlations are present then the flow harmonics can be written as:

$$v_n\{2, QC\} = \sqrt{c_n\{2\}}$$

$$v_n\{4, QC\} = \sqrt[4]{-c_n\{4\}}$$

Results from Q-Cumulant Method (Along with plots)

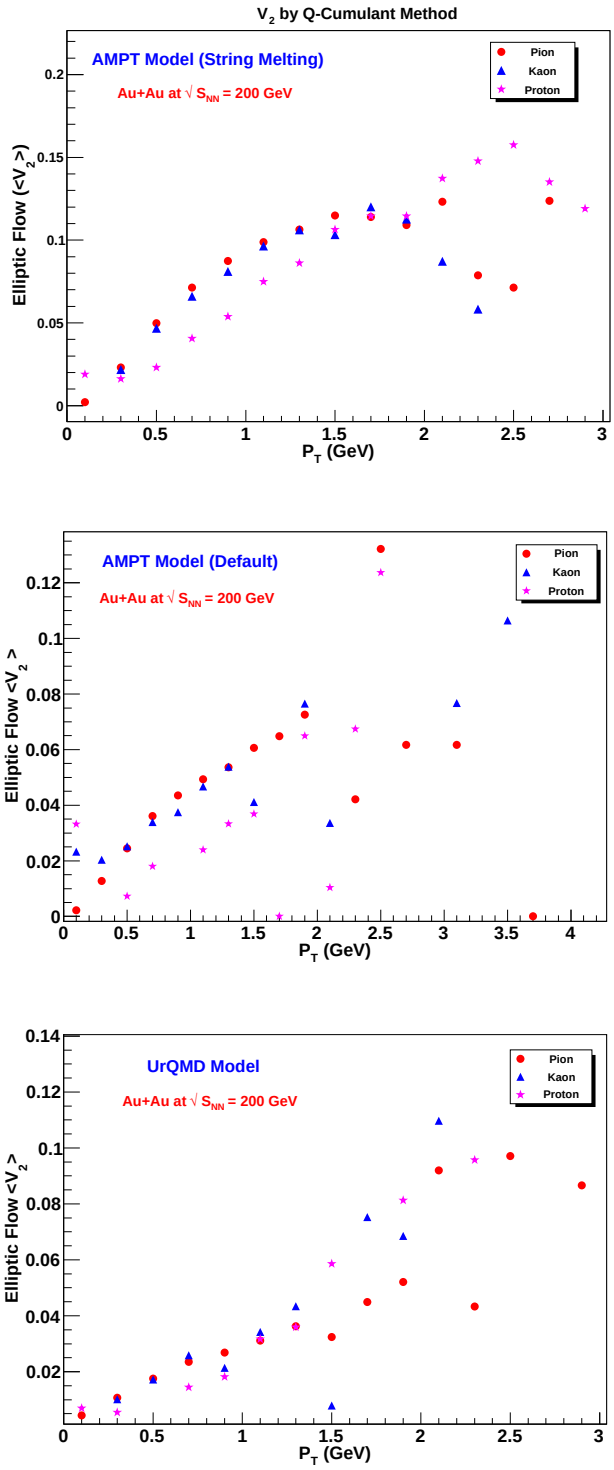


Figure 7: Elliptic Flow as a function of P_T by Q-Cumulant Method for Charge particles Pions, Kaons and Protons for Au+Au collision at $\sqrt{S_{NN}} = 200$ GeV for AMPT Model(Default), AMPT Model(String Melting) and UrQMD Model with X axis as Transverse Momentum P_T in GeV and Y axis as Average Elliptic Flow v_2

Conclusions

Need of Elliptic Flow: The elliptic flow in particular is a powerful probe to investigate the matter produced in relativistic heavy ion collisions at high energies (in CERN SPS, BNL RHIC, CERN LHC). The following are the important point for measurement of elliptic flow –

1. The elliptic anisotropy is attributed to the presence of a collective motion (flow) of the particle produce in the collisions.
2. The elliptic flow brings information about
   - Initial conditions of system created in collisions.
   - Equation of state.
   - Hadronization by coalescence or recombination of deconfined quarks and gluons.
   - Degree of thermalization.
   - On the time needed to reach the equilibrium.
   - On the viscosity of the system created in the nuclear collisions.
3. Helps in understanding the process of thermalization, creation of quark-gulon plasma, phase transition etc.
4. For measurements of identical and nonidentical two particle correlation.
5. For development of new technique suitable for flow studies at high energies.

Conclusions from methods :

In this report, I have studied the anisotropic flow v_2 for identified charge particles ($\pi^{+-}, K^{+-}, p, \overline{p}$) for Au+Au collisions at $\sqrt{S}_{NN} = 200$ GeV using different model (AMPT,UrQMD). Hadron mass dependence at low P_T region and particle type dependence (baryons and mesons) at the intermediate P_T region are observed in different model.

The absolute value of hadron v_2 demonstrate the need of early dense partionic interaction in heavy ion collisions.The measurements of collective motion of hadrons from high-energy nuclear collisions can provide information on the dynamical equation of the system. Specifically, the strange hadron flow results show the partonic collectivity and heavy-quarks flow will test early thermalization in such collisions.

At RHIC energies (almost 200 GeV), the measurements of elliptic flow v_2 has lead to the conclusion that hadrons were formed via the recombination of massive quarks. This conclusion is directly related to the deconfinement in high energy nuclear collisions. It also helps in understanding the process of hardronization in high energies collisions.

P_T dependence of the elliptic flow v_2:

from Plots we can see that at the lower $P_T \leq 1.5 GeV/c$, heavier hadrons have smaller values of v_2, such mass dependence is observed in experimental data. At higher P_T region, the mass ordering seems disappearing and $v_2(P_T)$ depends on particle type either meason or baryon. but we can see that for AMPT Default Model their is mass dependence for whole P_T region. This results shows that hadrons are created by fragmentation process instead of recombination of constituents quarks.

Fig. 4 shows the scaled hadron v_2. The scaling factor, according to the quark coalescence approach, is the number of constituent quarks (NCQ). For mesons and bayrons $n_q = 2$ and $n_q = 3$, respectively. In UrQMD Model, Expect pions, the NCQ scaling is observed in model calculations and it does not work for kaons in $.5 < KE_T/n_q < 1 GeV/C$. In AMPT String Melting where flow is more it shows good scaling comparing to the UrQMD model. But we see that for AMPT Default Model there is no scaling. In high energy collisions the parton density is much higher than in elementary(e^+e^-) collisions. Hadrons can be formed, via 'coalescence' of quarks from different strings. The partonic collectivity,therefore will be observed in hadron momentum distribution.

From the Plots of Elliptic Flow with Transverse Momentum we conclude that there can be three qualitatively different phase space regions:

- At very low $P_T(< 1 GeV)$ flow decreases with increase in particle mass i.e. it follows as $V_2 \propto P_T^2$. Hence flow dominated by mass dependence of particles. For baryon, mesons and quarks it is $V_{2B} < V_{2M} < V_{2q}$.
- At intermediate $P_T(> 1 GeV and < 4 GeV)$ where elliptic flow increase linearly with P_T thus $V_{2B} \approx V_{2M}$.
- At very large $P_T(> 4 GeV)$ where particle production is dominated by independent of parton fragmentation and thus parton's elliptic flow decrease as expected i.e. $V_{2B} > V_{2M} > V_{2q}$.(not shown in plots due to low statistics)

This shows that particle production is dominated by quark coalescence.

Appendix A:

AMPT Model

AMPT Model A Multiphase transport Model

Introduction : The AMPT Model is constructed to describe nuclear collisions ranging from p+A to A+A systems at center of mass energies from about = 5 to 5500 GeV at LHC, where strings and minijets dominate the inital energy production and effects from final state interactions are important. For initial conditions the AMPT Model uses the hard minijets partons and soft strings from HIJING Model. ZPC for describe parton's scattering among parton which is followed by a hadronization model or by quark coalescence model. An extended AMPT model with string melting in which hadrons, which would have been produced from string fragmentation are converted instead of their valance quark and anti quark is used. Scattering among the resulting hadrons are described by a relativistic model called ART model.

Physics of AMPT Model: AMPT Model consists of four main components:

1. Initial conditions : HIJING (Heavy Ion Jet Interaction Generator) Model for generating the initial conditions.
2. Partonic interactions : ZPC (Zhang's Parton Cascade) model for modeling partonic scatterings.
3. conversion from partonic to hardronic matter : LSF (Lund String Fragmentation) model or Quark coalescence model for Hydronization.
4. hardronic interactions: ART(A Relativistic Transport) model for treating hardronic scatterings.

The initial conditions which include the spacial and momentum distributions of minijet partons and soft string excitations are obtained from the HIJING Model. Its current version is 1.383 which do not include baryon junctions.

Scattering among partons are modeled by ZPC which at presents include only two body scattering with cross section obtained from the pQCD with screening masses.

In The Default AMPT Model, partions are recombined with their parent string when they stop interacting and the resulting strings are converted into hadrons using Lund String Fragmentation model.

In The AMPT String melting model, a Quark Coalescence model is used instead to combine partons into hardons.

Final results from AMPT Model are obtained after hadronic interactions are terminated at a cutoff time tcut when observables under study are considered to be stable.

To run the AMPT program :

1. Set the initial parameters in 'input.ampt'. If one prefers to use run-time random number seed, set 'ihjsed=11', In this way, every run is different even with the same 'input.ampt' file.

2. Type ' sh exec ' to compile and run the executable 'ampt' with some general information written in 'nohup.out'.

Key initial parameters in 'input.ampt' are		
Parameter	**Value**	**Description**
EFRM	200	$\sqrt{(}s_{NN})$ in GeV
FRAME	CMS	Frame of reference (center of Mass frame, Lab frame)
PROJ	A	Projectile Nucleus
TARG	A	Target Nucleus
IAP	197	Projectile Atomic Mass number
IZP	79	Projectile Atomic number
IAT	197	Target Atomic Mass number
IZT	79	Target Atomic number
NEVNT	500	The total number of events.
BMIN	0.5	Minimum impact parameter in fm
BMAX	25.0	Maximum impact parameter in fm
ISOFT		choice of parton-hadron conversion scenario.
	1	Default AMPT model (version 1.x);
	4	The AMPT model with string melting (version 2.y).
	2	A string is decomposed into q+dq+minijet partons instead of using the Lund fragmentation;
	3	A baryon is decomposed into q+qq instead of 3 quarks;
	5	Same as 4 but partons freeze out according to local energy density.
NTMAX	150	Number of time-steps for hadron cascade.(default = 150)
	1000	Use large value for HBT studies in heavy ion collisions.
	3	Use effectively turns off hadronic cascade.
DT	0.2	Timestep in fm (hadron cascade time= DT*NTMAX) (D=0.2)
PARJ(41)	2.5	Parameter a in Lund symmetric splitting function.
PARJ(42)	0.9	Parameter b in Lund symmetric splitting function.
PM flag	1,0	flag for popcorn mechanism (netbaryon stopping)(1= yes, 0= no)

shadowing flag	1,0	(Default=1,yes; 0,no)
quenching flag	1,0	(Default=0,no; 1,yes)
quenching parameter	1.5	-dE/dx (GeV/fm) in case quenching flag=1
p0 cutoff	2.0	For minijet productions in HIJING (Default=2.0)
parton screening mass	3.2264d0	In fm^{-1} (Default)
Ks0decays	1,0	flag for Kshort Decay (Default=0,no; 1,yes)

Key output file are: ana/ampt.dat: It contains particle records at hadron kinetic freeze-out, i.e., at the last interaction point. For each event, the first line gives: event number, test number(=1), number of particles in the event, impact parameter, total number of participant nucleons in projectile, total number of participant nucleons in target, number of participant nucleons in projectile due to elastic collisions, number of participant nucleons in projectile due to inelastic collisions, and corresponding numbers in target.Participant nucleon numbers include nucleons participating in both elastic and inelastic collisions. Each of the following lines gives: PYTHIA particle ID number, three-momentum (Px,Py,Pz), mass, and space-time coordinates(x,y,z,t) of one final particle at freeze-out. Momenta are in units of GeV/c, mass in $\frac{GeV}{c^2}$, space in fm, and time in fm/c. If a particle comes from the decay of a resonance which still exists at the termination time of hadron cascade, then its space-time corresponds to the decay point of the parent resonance. The x-axis in AMPT is defined as the direction along the impact parameter, and the z-axis is defined as the beam direction.

ana/zpc.dat: similar to 'ana/ampt.dat' but for partons at freeze-out. The first line of each event gives: event number, number of partons in the event, impact-parameter, number of participant nucleons in projectile due to elastic collisions, number of participant nucleons in projectile due to inelastic collisions, and corresponding numbers in target. Each of the following lines gives: PYTHIA particle ID number, three-momentum(P_x, P_y, P_z), mass, and space-time coordinates (x,y,z,t) of one final parton at freeze-out.

Appendix B:

UrQMD Model

UrQMD Model Ultrarelativistic Quantum Molecular Dynamics model

Introduction :

The Ultra relativistic Quantum Molecular Dynamics model is a microscopic model used to simulate (ultra)relativistic heavy ion collisions in the energy range from Bevalac and SIS up to AGS, SPS and RHIC. Main goals are to gain understanding about the following physical phenomena within a single transport model:

- Creation of dense hadronic matter at high temperatures
- Properties of nuclear matter, Delta and Resonance matter
- Creation of mesonic matter and of anti-matter
- Creation and transport of rare particles in hadronic matter
- Creation, modification and destruction of strangeness in matter
- Emission of electromagnetic probes

Physics of UrQMD Model :

The Ultra-relativistic Quantum Molecular Dynamic model (UrQMD) is a micro-scopic model based on a phase space description of nuclear reactions. It describes the phenomenology of hadronic interactions at low and intermediate energies ($\sqrt{s} < 5$GeV) in terms of interactions between known hadrons and their resonances. At higher energies, $\sqrt{s} > 5$ GeV, the excitation of color strings and their subsequent fragmentation into hadrons are taking into account in the UrQMD model. The model was proposed mainly for a description of nucleus-nucleus interactions. Since,up to now there is no unique theoretical description of the underlying hadronhadron interactions, with their vastly different characteristics at different incident energies and in different kinematic intervals. Perturbative quantum chromodynamics (pQCD) can be applied to describe hard processes, i.e. processes with large four-momentum, Q^2 , transfer. But pQCD is formally inappropriate for the description of the soft interactions because of the absence of the large Q^2scale. Therefore, low P_T collisions are described in terms of phenomenological models. A vast variety of models for hadronic- and nuclear collisions have been developed.

Compiling and running :

1. To compile UrQMD one needs a FORTRAN77 compiler and GNU-make.
2. Compilation is initiated by issuing the make command at the command-prompt in the UrQMD subdirectory

3. After successful compilation the binary has the name urqmd.ARCH where ARCH is the machine type as given by uname -m.

4. In order to run UrQMD dene the running parameters with an input le.The input le is made accessible to UrQMD by attaching its name to the environment-variable ftn09.The output les are attached in the same fashion via the environment variables ftn14 and ftn15.

   - export ftn09=inputfile
   - export ftn13=outputfile with freezeout
   - export ftn14=outputfile
   - export ftn15=collisionfile
   - export ftn16=outputfile with decaying resonances
   - export ftn19=outputfile for OSCAR97
   - export ftn20=outputfile for OSCAR99

5. To run UrQMD model start the runqmd.bash file.

6. Output is written to the files test.f14.

Key output file are: The UrQMD program has several different output les. The standard output les (file13 and file14) contain all particles of a given event at a certain time-step. The collision history le (file15) contains information on all collisions/decays of a given event. The decay le (file16) contains information on all particle decays as well as information on all stable particles after the nal timestep. The OSC les (file19, file20) generates output compliant with the Open Standards And Codes (OSCAR) format. Consecutive timesteps (only file13 and file14) and events are added sequentially to the les.Each event consists of a header and a body. The standard header is identical for file13, file14 and file16 and file15 use an abbreviated header and the format of file19 and file20 is xed by the OSCAR requirements.

Key initial parameters in 'inputfile' are		
Label	**Arguments**	**Description**
#		comment line
xxx		last line of inputle
pro	Ap Zp	dene projectile
PRO	ityp iso3	dene special projectile
tar	At Zt	dene target
TAR	ityp iso3	dene special target
nev	nevents	number of events to calculate
tim	tottime outtime	dene time of calculation and output
ene	ebeam	incident kinetic beam energy (lab frame)
elb	ebeam	incident kinetic beam energy (lab frame)
plb	pbeam	incident beam momentum (lab frame)
PLB	pmin pmax npbin	incident (min/max) beam momentum for excitation function
PLG	pmin pmax npbin	like PLB, log-weighted
ecm	srt	$\sqrt{s}$ for two particle collision
ENE	srtmin srtmax nsrt	incident min/max $\sqrt{s}$ for excitation function
ELG	srtmin srtmax nsrt	incident min/max $\sqrt{s}$ for excitation function (logweighted)
imp	bmax	dene impact parameter (bmin=0)
IMP	bmin bmax	dene impact parameter
eos	EoS	dene equation of state
box	dim edens solid para	dene box for innite matter calculation
bpt	ityp iso3 npart pmax	dene particle population for box-mode
bpe	ityp iso3 npart	like bpt, for given energy density
rsd	seed	seed for random number generator
stb	ityp	keep particle stable
cdt	deltat	Δt between full collision load
f13		suppress output to unit 13
f14		suppress output to unit 14
f15		suppress output to unit 15
f16		suppress output to unit 16
f19		suppress output to unit 19
f20		suppress output to unit 20
ctp	index value	set optional parameter in CTParam array
cto	index value	set option in CTOption array

Appendix C:

Program for Event Plane Method

```
#define ampt200GeV6mb_cxx
#include "ampt200GeV6mb.h"
#include <TH2.h>
#include <TStyle.h>
#include <TCanvas.h>

Float_t GetPt(Float_t,Float_t);
Float_t GetTotalMomentum(Float_t,Float_t,Float_t);
Float_t GetEnergy(Float_t,Float_t);
Float_t GetTransMass(Float_t,Float_t);
Float_t GetPolarAngle(Float_t,Float_t);
Float_t GetRapidity(Float_t,Float_t);
Float_t GetPseudoRapidity(Float_t, Float_t);
Float_t GetPhi(Float_t,Float_t);
Float_t GetV2(Float_t,Float_t);

void ampt200GeV6mb::Loop()
{

  gROOT->SetStyle("Plain");
  gStyle->SetCanvasBorderMode(0);
  gStyle->SetFrameBorderMode(0);

  Float_t pT,TotalP,E,theta,Y,eta,mT,phi;
  Float_t sum1L=0,sum2L=0,sum1R=0,sum2R=0,si=0,shi2L=0,shi2R=0,v2Pion,v2Proton,v2Kaon, TKineticE;
  Int_t count=0,npion=0,nkaon=0,nproton=0,Nch=0;


  TH1F *hpt = new TH1F("hpt","Transverse Momentum distribution; P_{T}(GeV);Counts(N)",200,-1,8);
  TH1F *hp = new TH1F("hp","Total Momentum distribution;P(GeV) ;Counts(N)",200,-1,30);
  TH1F *hE = new TH1F("hE","Mass Energy distribution; Mass energy(GeV);Counts(N)",200,-1,30);
  TH1F *hmT= new TH1F("hmT","Transverse Mass distribution; M_{T}(GeV);Counts(N)",200,-1,8);
  TH1F *htheta = new TH1F("htheta","Polar Angle distribution; #theta(Radian);Counts(N)",200,-1,5);
  TH1F *hy = new TH1F("hy","Rapidity distribution; Rapidity(y);Counts(N)",200,-10,10);
  TH1F *heta = new TH1F("heta","Pseudorapidity distribution; #eta ;Counts(N)",200,-10,10);
  TH1F *hphi = new TH1F("hphi","Azimuthal angle distribution; #phi(Radian);Counts(N)",200,-1,8);
  TH1F *hshi2 = new TH1F("hshi2","Reaciton Plane Angle distribution;#psi(Radian); Counts(N)",200,-1,8);

  TProfile *hb = new TProfile("hb","N_{ch} Vs b; Imapact Parameter b(fm); Charge Multiplicity N_{ch}",2000,0,20,0,10000);
  TProfile *hv2Pion = new TProfile("v2pi","V2 Vs pt  ;P_{T}(Gev) ;Elliptic Flow(V_{2})",30,0,3,-1,1);
  TProfile *hv2Kaon = new TProfile("v2K","V2 Vs pt   ;P_{T}(Gev) ;Elliptic Flow(V_{2})",30,0,3,-1,1);
  TProfile *hv2Proton = new TProfile("v2p","V2 Vs pt ;P_{T}(Gev) ;Elliptic Flow(V_{2})",30,0,3,-1,1);

  TProfile *hKEPion = new TProfile("KEpi","V2 Vs KE_{T}  ;KE_{T}(Gev) ;Elliptic Flow(V_{2})",30,0,3,-1,1);
  TProfile *hKEKaon = new TProfile("KEk","V2 Vs KE_{T}  ;KE_{T}(Gev) ;Elliptic Flow(V_{2})",30,0,3,-1,1);
  TProfile *hKEProton = new TProfile("KEp","V2 Vs KE_{T};  KE_{T}(Gev) ;Elliptic Flow(V_{2})",30,0,3,-1,1);


  TProfile *hv2nqPion = new TProfile("v2nqpi","V2 Vs pt  ;P_{T}/N_{q}(Gev) ;Elliptic Flow(V_{2})/N_{q}",20,0,2,-1,1);
  TProfile *hv2nqKaon = new TProfile("v2nqK","V2 Vs pt   ;P_{T}/N_{q}(Gev) ;Elliptic Flow(V_{2})/N_{q}",20,0,2,-1,1);
  TProfile *hv2nqProton = new TProfile("v2nqp","V2 Vs pt ;P_{T}/N_{q}(Gev) ;Elliptic Flow(V_{2})/N_{q}",20,0,2,-1,1);

  TProfile *hKEnqPion = new TProfile("KEnqpi","V2 Vs KE_{T}  ;KE_{T}/N_{q}(Gev) ;Elliptic Flow(V_{2})/N_{q}",20,0,2,-1,1);
  TProfile *hKEnqKaon = new TProfile("KEnqk","V2 Vs KE_{T}  ;KE_{T}/N_{q}(Gev) ;Elliptic Flow(V_{2})/N_{q}",20,0,2,-1,1);
  TProfile *hKEnqProton = new TProfile("KEnqp","V2 Vs KE_{T}  KE_{T}/N_{q}(Gev) ;Elliptic Flow(V_{2})/N_{q}",20,0,2,-1,1);

  TProfile *hv2massPion = new TProfile("v2masspi","V2 Vs Mass  ;M_{0}(Gev) ;Elliptic Flow(V_{2})",30,0,3,-1,1);
  TProfile *hv2massKaon = new TProfile("v2massK","V2 Vs Mass   ;M_{0}(Gev) ;Elliptic Flow(V_{2})",30,0,3,-1,1);
  TProfile *hv2massProton = new TProfile("v2massp","V2 Vs Mass ;M_{0}(Gev) ;Elliptic Flow(V_{2})",30,0,3,-1,1);

  TProfile *hv2etaPion = new TProfile("v2etapi","V2 Vs #eta  ;#eta ;Elliptic Flow(V_{2})",50,-5,5,-1,1);
  TProfile *hv2etaKaon = new TProfile("v2etaK","V2 Vs #eta   ;#eta ;Elliptic Flow(V_{2})",50,-5,5,-1,1);
  TProfile *hv2etaProton = new TProfile("v2etap","V2 Vs #eta ;#eta ;Elliptic Flow(V_{2})",50,-5,5,-1,1);

  TProfile *hv2impPion = new TProfile("v2imppi","V2 Vs b  ;Impact parameter(b) ;Elliptic Flow(V_{2})",60,0,18,-1,1);
  TProfile *hv2impKaon = new TProfile("v2impK","V2 Vs b   ;Impact parameter(b) ;Elliptic Flow(V_{2})",60,0,18,-1,1);
  TProfile *hv2impProton = new TProfile("v2impp","V2 Vs b ;Impact parameter(b) ;Elliptic Flow(V_{2})",60,0,18,-1,1);


  if (fChain == 0) return;

  Long64_t nentries = fChain->GetEntriesFast();

  Long64_t nbytes = 0, nb = 0;
  for (Long64_t jentry=0; jentry<nentries;jentry++) {
```

```
    Long64_t ientry = LoadTree(jentry);
    if (ientry < 0) break;
    nb = fChain->GetEntry(jentry);   nbytes += nb;
    // if (Cut(ientry) < 0) continue;
        //if(jentry>5000) break;
    cout<<"event number --->  "<<jentry<<endl;


    // for reaction plane angle
    for (int jparticle=0; jparticle<fParticles_; jparticle++)
      {

if( (fParticles_PID[jparticle]== 211 ||  fParticles_PID[jparticle]== -211) ||
( (fParticles_PID[jparticle]== 321 || fParticles_PID[jparticle] == -321) ||
(fParticles_PID[jparticle] == 2212 ||fParticles_PID[jparticle] == -2212) ) )
  {

    if(fParticles_px[jparticle]== 0 && fParticles_py[jparticle]== 0)  continue; //exclusion of non participating nucleon

    TotalP = GetTotalMomentum(fParticles_px[jparticle],fParticles_py[jparticle],fParticles_pz[jparticle]);

    eta= GetPseudoRapidity(TotalP, fParticles_pz[jparticle]);

    phi = GetPhi(fParticles_px[jparticle],fParticles_py[jparticle]);
    if(eta>0)
      {
sum1R = sum1R + sin(2*phi);
sum2R = sum2R + cos(2*phi);
      }
    if(eta<0)
      {
sum1L = sum1L + sin(2*phi);
sum2L = sum2L + cos(2*phi);
      }
  }

      }


    if(sum2L!=0)
      {
si =sum1L/sum2L;
if(si!=0)  shi2L = TMath::ATan(si)/2.0;
hshi2->Fill(shi2L);
      }

    if(sum2R!=0)
      {
si =sum1R/sum2R;
if(si!=0)  shi2R = TMath::ATan(si)/2.0;
hshi2->Fill(shi2R);
      }


Nch =0;
    //Particle loop starts here
    for (int jparticle=0; jparticle<fParticles_; jparticle++)
      {

if( (fParticles_PID[jparticle]== 211 ||  fParticles_PID[jparticle]== -211) ||
( (fParticles_PID[jparticle]== 321 || fParticles_PID[jparticle] == -321) ||
(fParticles_PID[jparticle] == 2212 ||fParticles_PID[jparticle] == -2212) ) )
  {
    if(fParticles_px[jparticle]==0 && fParticles_py[jparticle]==0)    continue; //exclusion of sepectator nucleons

    count = count +1;                                          // counting of charge particles
    Nch = Nch +1;
    //Impact parameter

    //Transverse Momentum
    pT = GetPt(fParticles_px[jparticle],fParticles_py[jparticle]);
    hpt->Fill(pT);

    //Total Momentum
    TotalP = GetTotalMomentum(fParticles_px[jparticle],fParticles_py[jparticle],fParticles_pz[jparticle]);
    hp->Fill(TotalP);

    //Energy
    E = GetEnergy(TotalP, fParticles_m[jparticle]);
    hE->Fill(E);
```

```
    //Rapidity
    Y=GetRapidity(E,fParticles_pz[jparticle]);
    hy->Fill(Y);

    //PseudoRapidity
    eta= GetPseudoRapidity(TotalP, fParticles_pz[jparticle]);
    heta->Fill(eta);

    //Polar angle
    theta = GetPolarAngle(fParticles_pz[jparticle],TotalP);
    htheta->Fill(theta);

    //Azimuthal angle
    //  phi = GetPhi(fParticles_px[jparticle],fParticles_py[jparticle]);



    //eta cut start here
    if(eta > 0)
      {
phi = GetPhi(fParticles_px[jparticle],fParticles_py[jparticle]);
hphi->Fill(phi);

//for Pion V2 and v2/Nq
if(fParticles_PID[jparticle]==211 || fParticles_PID[jparticle]==-211)
  {
    v2Pion = GetV2(phi,shi2L);

    mT = GetTransMass(E,fParticles_pz[jparticle]);

    TKineticE = (mT - fParticles_m[jparticle]);

    hv2Pion->Fill(pT,v2Pion);
    hKEPion->Fill(TKineticE,v2Pion);

    hv2nqPion->Fill(pT/2.0,v2Pion/2.0);
    hKEnqPion->Fill(TKineticE/2.0,v2Pion/2.0);

hv2massPion->Fill(fParticles_m[jparticle], v2Pion);
hv2etaPion->Fill(eta, v2Pion);
hv2impPion->Fill(bb,v2Pion);

    npion = npion+1;


  }
// V2 and v2/Nq for Kaon
if(fParticles_PID[jparticle]==321 || fParticles_PID[jparticle]==-321)
  {
    v2Kaon = GetV2(phi,shi2L);

    mT = GetTransMass(E,fParticles_pz[jparticle]);

    TKineticE = (mT - fParticles_m[jparticle]);

    hv2Kaon->Fill(pT,v2Kaon);
    hKEKaon->Fill(TKineticE,v2Kaon);

    hv2nqKaon->Fill(pT/2.0,v2Kaon/2.0);
    hKEnqKaon->Fill(TKineticE/2.0,v2Kaon/2.0);

hv2massKaon->Fill(fParticles_m[jparticle], v2Kaon);
hv2etaKaon->Fill(eta, v2Kaon);
hv2impKaon->Fill(bb,v2Kaon);

    nkaon = nkaon +1;


  }

// v2 and v2/Nq for Proton
if(fParticles_PID[jparticle]==2212 || fParticles_PID[jparticle]==-2212)
  {

    v2Proton = GetV2(phi,shi2L);

    mT = GetTransMass(E,fParticles_pz[jparticle]);

    TKineticE = (mT - fParticles_m[jparticle]);

    hv2Proton->Fill(pT,v2Proton);
```

```
    hKEProton->Fill(TKineticE,v2Proton);

    hv2nqProton->Fill(pT/3.0,v2Proton/3.0);
    hKEnqProton->Fill(TKineticE/3.0,v2Proton/3.0);

hv2massProton->Fill(fParticles_m[jparticle], v2Proton);
hv2etaProton->Fill(eta, v2Proton);
hv2impProton->Fill(bb,v2Proton);
    nproton = nproton +1;
  }


      }//eta cut if ends


    if(eta< 0)
      {
phi = GetPhi(fParticles_px[jparticle],fParticles_py[jparticle]);
hphi->Fill(phi);

//for Pion V2 and v2/Nq
if(fParticles_PID[jparticle]==211 || fParticles_PID[jparticle]==-211)
  {
    v2Pion = GetV2(phi,shi2R);

    mT = GetTransMass(E,fParticles_pz[jparticle]);

    TKineticE = (mT - fParticles_m[jparticle]);

    hv2Pion->Fill(pT,v2Pion);
    hKEPion->Fill(TKineticE,v2Pion);

    hv2nqPion->Fill(pT/2.0,v2Pion/2.0);
    hKEnqPion->Fill(TKineticE/2.0,v2Pion/2.0);

hv2massPion->Fill(fParticles_m[jparticle], v2Pion);
hv2etaPion->Fill(eta, v2Pion);
hv2impPion->Fill(bb,v2Pion);

    npion= npion +1;

  }
// V2 and v2/Nq for Kaon
if(fParticles_PID[jparticle]==321 || fParticles_PID[jparticle]==-321)
  {
    v2Kaon = GetV2(phi,shi2R);

    mT = GetTransMass(E,fParticles_pz[jparticle]);

    TKineticE = (mT - fParticles_m[jparticle]);

    hv2Kaon->Fill(pT,v2Kaon);
    hKEKaon->Fill(TKineticE,v2Kaon);

    hv2nqKaon->Fill(pT/2.0,v2Kaon/2.0);
    hKEnqKaon->Fill(TKineticE/2.0,v2Kaon/2.0);

hv2massKaon->Fill(fParticles_m[jparticle], v2Kaon);
hv2etaKaon->Fill(eta, v2Kaon);
hv2impKaon->Fill(bb,v2Kaon);

    nkaon= nkaon +1;

  }

// v2 and v2/Nq for Proton
if(fParticles_PID[jparticle]==2212 || fParticles_PID[jparticle]==-2212)
  {

    v2Proton = GetV2(phi,shi2R);

    mT = GetTransMass(E,fParticles_pz[jparticle]);

    TKineticE = (mT - fParticles_m[jparticle]);

    hv2Proton->Fill(pT,v2Proton);
    hKEProton->Fill(TKineticE,v2Proton);


    hv2nqProton->Fill(pT/3.0,v2Proton/3.0);
    hKEnqProton->Fill(TKineticE/3.0,v2Proton/3.0);
```

```
hv2massProton->Fill(fParticles_m[jparticle], v2Proton);
hv2etaProton->Fill(eta, v2Proton);
hv2impProton->Fill(bb,v2Proton);

    nproton = nproton +1;
  }

      }//eta cut if ends

  }   // end of charge partice if block

      } // end of Particle loop
     hb->Fill(bb,Nch);    //Impact Parameter vs Charge Multiplicity

  }// event loop ends here

  cout<<"Total no. of Charge particle are---------->"<<count<<endl;
  cout<<"Total no. of pion are ------->"<<npion<<endl;
  cout<<"Total no. of proton are ------->"<<nproton<<endl;
  cout<<"Total no. of Kaon are ------->"<<nkaon<<endl;

  TCanvas *cphi = new TCanvas("cphi", "Azimuthal angle  Distributions",800,600);
  TCanvas *cshi2 = new TCanvas("cshi2", "reaction plane angle Distributions",800,600);
  TCanvas *cv2 = new TCanvas("cv2", "V_{2} Vs P_{T} Plot",800,600);
  TCanvas *cKE = new TCanvas("cKE", "V_{2} Vs KE_{T} Plot",800,600);
  TCanvas *cv2nq = new TCanvas("cv2nq", "V_{2}/N_{q} Vs P_{T}/N_{q} Plot",800,600);
  TCanvas *cKEnq = new TCanvas("cKEnq", "V_{2}/N_{q} Vs KE_{T}/N_{q} Plot",800,600);
  TCanvas *cv2mass = new TCanvas("cv2mass","V_{2} Vs Mass",800,600);
  TCanvas *cv2eta = new TCanvas("cv2eta","V_{2} Vs #eta", 800,600);
  TCanvas *cv2imp = new TCanvas("cv2imp","V_{2} Vs b",800,600);

  cphi->cd();
  hphi->Draw();

  cshi2->cd();
  hshi2->Draw();

  cv2->cd();
  gStyle->SetOptStat(kFALSE);
  hv2Pion->SetMarkerColor(2);
  hv2Pion->SetMarkerStyle(20);
  hv2Pion->SetMarkerSize(1);
  hv2Pion->Draw();

  hv2Kaon->SetMarkerColor(4);
  hv2Kaon->SetMarkerStyle(22);
  hv2Kaon->Draw("same");

  hv2Proton->SetMarkerColor(6);
  hv2Proton->SetMarkerStyle(29);
  hv2Proton->Draw("same");

  TLegend *leg1 = new TLegend(.6,.6,.8,.8);
  leg1->AddEntry(hv2Pion,"Pion","p");
  leg1->AddEntry(hv2Proton,"Proton","p");
  leg1->AddEntry(hv2Kaon,"Kaon","p");
  leg1->Draw();

  cKE->cd();
  gStyle->SetOptStat(kFALSE);
  hKEPion->SetMarkerColor(2);
  hKEPion->SetMarkerStyle(20);
  hKEPion->SetMarkerSize(1);
  hKEPion->Draw();

  hKEKaon->SetMarkerColor(4);
  hKEKaon->SetMarkerStyle(22);
  hKEKaon->Draw("same");

  hKEProton->SetMarkerColor(6);
  hKEProton->SetMarkerStyle(29);
  hKEProton->Draw("same");

  TLegend *leg2 = new TLegend(.6,.6,.8,.8);
  leg2->AddEntry(hKEPion,"Pion","p");
  leg2->AddEntry(hKEProton,"Proton","p");
  leg2->AddEntry(hKEKaon,"Kaon","p");
  leg2->Draw();

  cv2nq->cd();
```

```
gStyle->SetOptStat(kFALSE);
hv2nqPion->SetMarkerColor(2);
hv2nqPion->SetMarkerStyle(20);
hv2nqPion->SetMarkerSize(1);
hv2nqPion->Draw();

hv2nqKaon->SetMarkerColor(4);
hv2nqKaon->SetMarkerStyle(22);
hv2nqKaon->Draw("same");

hv2nqProton->SetMarkerColor(6);
hv2nqProton->SetMarkerStyle(29);
hv2nqProton->Draw("same");

TLegend *leg3 = new TLegend(.6,.6,.8,.8);
leg3->AddEntry(hv2nqPion,"Pion","p");
leg3->AddEntry(hv2nqProton,"Proton","p");
leg3->AddEntry(hv2nqKaon,"Kaon","p");
leg3->Draw();

cKEnq->cd();
gStyle->SetOptStat(kFALSE);
hKEnqPion->SetMarkerColor(2);
hKEnqPion->SetMarkerStyle(20);
hKEnqPion->SetMarkerSize(1);
hKEnqPion->Draw();

hKEnqKaon->SetMarkerColor(4);
hKEnqKaon->SetMarkerStyle(22);
hKEnqKaon->Draw("same");

hKEnqProton->SetMarkerColor(6);
hKEnqProton->SetMarkerStyle(29);
hKEnqProton->Draw("same");

TLegend *leg4 = new TLegend(.6,.6,.8,.8);
leg4->AddEntry(hKEnqPion,"Pion","p");
leg4->AddEntry(hKEnqProton,"Proton","p");
leg4->AddEntry(hKEnqKaon,"Kaon","p");
leg4->Draw();

cv2mass->cd();
hv2massProton->SetMarkerSize(1.3);
hv2massProton->SetMarkerColor(6);
hv2massProton->SetMarkerStyle(29);
hv2massProton->Draw();

hv2massPion->SetMarkerColor(2);
hv2massPion->SetMarkerStyle(20);
hv2massPion->Draw("same");

hv2massKaon->SetMarkerColor(4);
hv2massKaon->SetMarkerStyle(22);
hv2massKaon->Draw("same");

TLegend *leg5 = new TLegend(.6,.6,.8,.8);
leg5->AddEntry(hv2massPion,"Pion","p");
leg5->AddEntry(hv2massProton,"Proton","p");
leg5->AddEntry(hv2massKaon,"Kaon","p");
leg5->Draw();

cv2eta->cd();
hv2etaPion->SetMarkerColor(2);
hv2etaPion->SetMarkerStyle(20);
hv2etaPion->SetMarkerSize(1);
hv2etaPion->Draw();

hv2etaKaon->SetMarkerColor(4);
hv2etaKaon->SetMarkerStyle(22);
hv2etaKaon->Draw("same");

hv2etaProton->SetMarkerColor(6);
hv2etaProton->SetMarkerStyle(29);
hv2etaProton->Draw("same");

TLegend *leg6 = new TLegend(.6,.6,.8,.8);
leg6->AddEntry(hv2etaPion,"Pion","p");
leg6->AddEntry(hv2etaProton,"Proton","p");
leg6->AddEntry(hv2etaKaon,"Kaon","p");
leg6->Draw();
```

```
  cv2imp->cd();
  hv2impPion->SetMarkerColor(2);
  hv2impPion->SetMarkerStyle(20);
  hv2impPion->SetMarkerSize(1);
  hv2impPion->Draw();

  hv2impKaon->SetMarkerColor(4);
  hv2impKaon->SetMarkerStyle(22);
  hv2impKaon->Draw("same");

  hv2impProton->SetMarkerColor(6);
  hv2impProton->SetMarkerStyle(29);
  hv2impProton->Draw("same");

  TLegend *leg7 = new TLegend(.6,.6,.8,.8);
  leg7->AddEntry(hv2impPion,"Pion","p");
  leg7->AddEntry(hv2impProton,"Proton","p");
  leg7->AddEntry(hv2impKaon,"Kaon","p");
  leg7->Draw();

  TFile *f =new TFile("ampt200GeV6mb_rt.root","recreate");
  f->cd();
  hb->Write();
  hpt->Write();
  hp->Write();
  hE->Write();
  hmT->Write();
  htheta->Write();
  hy->Write();
  heta->Write();
  hphi->Write();
  hshi2->Write();

  cv2->Write();
  hv2Pion->Write();
  hv2Kaon->Write();
  hv2Proton->Write();
  cKE->Write();
  hKEPion->Write();
  hKEKaon->Write();
  hKEProton->Write();
  cv2nq->Write();
  hv2nqPion->Write();
  hv2nqKaon->Write();
  hv2nqProton->Write();
  cKEnq->Write();
  hKEnqPion->Write();
  hKEnqKaon->Write();
  hKEnqProton->Write();

  cv2mass->Write();
  hv2massPion->Write();
  hv2massKaon->Write();
  hv2massProton->Write();

  cv2eta->Write();
  hv2etaPion->Write();
  hv2etaKaon->Write();
  hv2etaProton->Write();

  cv2imp->Write();
  hv2impPion->Write();
  hv2impKaon->Write();
  hv2impProton->Write();

  f->Close();
   // if (Cut(ientry) < 0) continue;
} // end of program Loop()


Float_t GetPt(Float_t fpx, Float_t fpy)
{
  Float_t sqpt;
  sqpt = (fpx*fpx) + (fpy*fpy);
  return sqrt(sqpt);
}


Float_t  GetTotalMomentum(Float_t fpx, Float_t fpy, Float_t fpz)
{
  return  sqrt( (fpx*fpx) + (fpy*fpy)+ (fpz*fpz) );
```

```
}

Float_t GetEnergy(Float_t  fp ,Float_t Invm)
{
  return sqrt( (fp*fp) + (Invm*Invm));

}


Float_t GetPolarAngle(Float_t  fpz ,Float_t fp)
{
  const Float_t pi =3.14159;
  if(fp == 0)
    return pi/2.0;
  else
    return TMath::ACos(fpz/fp);
}


Float_t GetPhi(Float_t fpx,Float_t fpy)
{
  const double pi = 3.14159;
  double phi=0, phi1=0;
  if(fpx == 0 && fpy > 0) phi = pi/2.0;
  if(fpx == 0 && fpy < 0) phi = 3*pi/2.0;
  if(fpy== 0 && fpx > 0) phi = 0;
  if(fpy ==0 && fpx < 0) phi = pi;
  phi1=TMath::ATan(TMath::Abs(fpy/fpx));
  if(fpx > 0 && fpy > 0) phi = phi1;
  if(fpx < 0 && fpy > 0) phi = pi -phi1;
  if(fpx < 0 && fpy < 0) phi = pi + phi1;
  if(fpx > 0 && fpy < 0) phi = 2*pi - phi1;

  return phi;
}


Float_t GetRapidity(Float_t fE, Float_t fpz)
{
  double temp;
  temp  = (fE + fpz) /(fE-fpz);
  if(temp != 0)
    return  (TMath::Log(temp) )/2.0;
}



Float_t GetPseudoRapidity(Float_t fp ,Float_t fpz)
{
  double temp;
  temp =(TMath::Abs(fp)+fpz)/(TMath::Abs(fp)-fpz);
  if(temp != 0)
    return  ( TMath::Log(temp) )/2.0;

}

Float_t GetTransMass(Float_t fE,Float_t fpz)
{
  return sqrt( (fE*fE) -(fpz*fpz) );
}


Float_t GetV2(Float_t fphi,Float_t fshi2)
{
  return TMath::Cos( 2*(fphi-fshi2) );
}
```

Appendix D:

Program for Q-Cumulant Method

```
#define ampt200Gev_cxx
#include "ampt200Gev.h"
#include <TH2.h>
#include <TStyle.h>
#include <TCanvas.h>

Float_t GetPhi(Float_t,Float_t);
Float_t GetPt(Float_t,Float_t);

void ampt200Gev::Loop()
{
  const Int_t M=10000;
  const Int_t ptbin=20;
  Int_t bin;
  Float_t pT,Phi;
  Float_t pxPion[ptbin][M],pyPion[ptbin][M],pzPion[ptbin][M],ptPion[ptbin][M],phiPion[ptbin][M];
  Float_t pxKaon[ptbin][M],pyKaon[ptbin][M],pzKaon[ptbin][M],ptKaon[ptbin][M],phiKaon[ptbin][M];
  Float_t pxProton[ptbin][M],pyProton[ptbin][M],pzProton[ptbin][M],ptProton[ptbin][M],phiProton[ptbin][M];
  Float_t AvgV2Pion[ptbin],AvgQ2sqPion[ptbin],CombPion[ptbin];
  Float_t AvgV2Kaon[ptbin],AvgQ2sqKaon[ptbin],CombKaon[ptbin];
  Float_t AvgV2Proton[ptbin],AvgQ2sqProton[ptbin],CombProton[ptbin];

  Float_t temp,AvgpT[ptbin];

 TH1F *hpt = new TH1F("hpt","Transverse Momentum distribution; P_{T}(GeV) ; Charge Multiplicity N_{ch}",200,0,8);
 TH1F *hphi = new TH1F("hphi","Azimuthal Angle distribution; #phi(Radian); Charge Multiplicity N_{ch}",200,-1,7);

  if (fChain == 0) return;
  Long64_t nentries = fChain->GetEntriesFast();
  Long64_t nbytes = 0, nb = 0;

  //Event Loop start here
  for (Long64_t jentry=0; jentry<nentries;jentry++)
    {
      Long64_t ientry = LoadTree(jentry);
      if (ientry < 0) break;
      nb = fChain->GetEntry(jentry);   nbytes += nb;
      // if(jentry >=1000) break;
      cout<<"event number --->  "<<jentry<<endl;
      //cout<<"multiplicity---->"<<Event_multi<<endl;
      //cout<<"------------------------------------------- "<<endl;

      Int_t npion[M] ={0.};
      Int_t nkaon[M] ={0.};
      Int_t nproton[M] ={0.};

      //Particle loop start
      // for(Int_t mul=0; mul<50;mul++)
      for(Int_t mul=0; mul<Event_multi; mul++)
{
  if(Px[mul] ==0 && Py[mul] ==0 ) continue;
  //For Pions
  if(ID[mul]==211 || ID[mul]==-211)
    {
      pT = GetPt(Px[mul],Py[mul]);
      Phi = GetPhi(Px[mul],Py[mul]);
hpt->Fill(pT);
hphi->Fill(Phi);
      for(Int_t i=0; i< ptbin ;i++)
{
  if(pT > .2*i && pT < =.2+(.2*i))
    {
      //cout<<"i am in for i="<<i<<endl;
      bin =i;
      //pxPion[bin][npion[bin]] = Px[mul];
      //pyPion[bin][npion[bin]] = Py[mul];
      //pzPion[bin][npion[bin]] = Pz[mul];
      //ptPion[bin][npion[bin]] = pT;
      phiPion[bin][npion[bin]] = Phi;
      npion[bin] = npion[bin]+1;
      break;
    }

}
```

```
    }


  // for Kaons
  if(ID[mul]==321 || ID[mul]==-321)
    {
      pT = GetPt(Px[mul],Py[mul]);
      Phi = GetPhi(Px[mul],Py[mul]);
hpt->Fill(pT);
hphi->Fill(Phi);
      for(Int_t i=0; i< ptbin ;i++)
{

  if(pT > .2*i && pT < =.2+(.2*i))
    {
      //cout<<"i am in for i="<<i<<endl;
      bin =i;
      //pxKaon[bin][nkaon[bin]] = Px[mul];
      //pyKaon[bin][nkaon[bin]] = Py[mul];
      //pzKaon[bin][nkaon[bin]] = Pz[mul];
      //ptKaon[bin][nkaon[bin]] = pT;
      phiKaon[bin][nkaon[bin]] = Phi;
      nkaon[bin] = nkaon[bin]+1;
      break;
    }

}

    }


  //For Protons
  if(ID[mul]==2212 || ID[mul]==-2212)
    {
      pT = GetPt(Px[mul],Py[mul]);
      Phi = GetPhi(Px[mul],Py[mul]);
hpt->Fill(pT);
hphi->Fill(Phi);
      for(Int_t i=0; i< ptbin ;i++)
{

  if(pT > .2*i && pT < =.2+(.2*i))
    {
      //cout<<"i am in for i="<<i<<endl;
      bin =i;
      //pxProton[bin][nproton[bin]] = Px[mul];
      //pyProton[bin][nproton[bin]] = Py[mul];
      //pzProton[bin][nproton[bin]] = Pz[mul];
      //ptProton[bin][nproton[bin]] = pT;
      phiProton[bin][nproton[bin]] = Phi;
      nproton[bin] = nproton[bin]+1;
      break;
    }

}

    }

}  // particle loop ends

      Float_t Q2SqPion[ptbin] ={ 0.};
      Float_t Q2SqKaon[ptbin] ={ 0.};
      Float_t Q2SqProton[ptbin] ={ 0.};

      for(Int_t b=0; b<ptbin; b++)
{
  for(Int_t i=0; i<npion[b]; i++)
    {
      for(Int_t j=0; j<npion[b]; j++)
{

  Q2SqPion[b] = Q2SqPion[b] + ( cos(2*(phiPion[b][i] - phiPion[b][j])) );
}

    }
  Q2SqPion[b] = Q2SqPion[b] - npion[b];


  for(Int_t i=0; i<nkaon[b]; i++)
    {
      for(Int_t j=0; j<nkaon[b]; j++)
```

```
{
  Q2SqKaon[b] = Q2SqKaon[b] + ( cos(2*(phiKaon[b][i] - phiKaon[b][j])) );
}
    }
  Q2SqKaon[b] = Q2SqKaon[b] - nkaon[b];


  for(Int_t i=0; i<nproton[b]; i++)
    {
      for(Int_t j=0; j<nproton[b]; j++)
{
  Q2SqProton[b] = Q2SqProton[b] + ( cos(2*(phiProton[b][i] - phiProton[b][j])) );
}
    }
  Q2SqProton[b] = Q2SqProton[b] - nproton[b];

  AvgQ2sqPion[b] = AvgQ2sqPion[b] +Q2SqPion[b];
  CombPion[b] = CombPion[b] + npion[b]*(npion[b]-1);

  AvgQ2sqKaon[b] = AvgQ2sqKaon[b] + Q2SqKaon[b];
  CombKaon[b] = CombKaon[b] + nkaon[b]*(nkaon[b]-1);

  AvgQ2sqProton[b] = AvgQ2sqProton[b] + Q2SqProton[b];
  CombProton[b] = CombProton[b] + nproton[b]*(nproton[b]-1);
}

      //  cout<<"---------------------------------------------------- "<<endl;

  } // end of event loop

  for(Int_t bbin=0; bbin<ptbin; bbin++)
    {
      AvgpT[bbin] = .1 +(.2*bbin);

      AvgV2Pion[bbin] = sqrt(AvgQ2sqPion[bbin]/CombPion[bbin]);
      hv2byQcumPion->Fill(pT,AvgV2Pion[bbin]);

      AvgV2Kaon[bbin] = sqrt(AvgQ2sqKaon[bbin]/CombKaon[bbin]);
      hv2byQcumKaon->Fill(pT,AvgV2Kaon[bbin]);

      AvgV2Proton[bbin] = sqrt(AvgQ2sqProton[bbin]/CombProton[bbin]);
      hv2byQcumProton->Fill(pT,AvgV2Proton[bbin]);

    }

 TGraph *grPion = new TGraph(ptbin,AvgpT,AvgV2Pion);
 TGraph *grKaon = new TGraph(ptbin,AvgpT,AvgV2Kaon);
 TGraph *grProton = new TGraph(ptbin,AvgpT,AvgV2Proton);

 TCanvas *cpt = new TCanvas("cpt","Transverse Momentum Distribution",800,600);
 TCanvas *cphi = new TCanvas("cphi","Azimuthal angle Distribution",800,600);
 TCanvas *cv2byQcum = new TCanvas("cv2byQcum","V2 by Q-Cumulant Method",800,600);
 cpt->cd();
 hpt->Draw();

 cphi->cd();
 hphi->Draw();

 cv2byQcum->cd();

 grPion->SetLineColor(2);
 grPion->SetMarkerSize(1.2);
 grPion->SetMarkerStyle(20);
 grPion->SetMarkerColor(2);
 grPion->Draw("AP");

 grKaon->SetLineColor(4);
 grKaon->SetMarkerSize(1.2);
 grKaon->SetMarkerStyle(22);
 grKaon->SetMarkerColor(4);
 grKaon->Draw("P");

 grProton->SetLineColor(3);
 grProton->SetMarkerSize(1.3);
 grProton->SetMarkerStyle(29);
 grProton->SetMarkerColor(6);
 grProton->Draw("P");


 TLegend *l= new TLegend(.1,.2,.2,.3);
 l->AddEntry(grPion,"Pion","p");
```

```
 l->AddEntry(grKaon,"Kaon","p");
 l->AddEntry(grProton,"Proton","p");
 l->Draw();


 TFile *f = new TFile("Q_cumulant_corr2.root","recreate");
 f->cd();
   hpt->Write();
 hphi->Write();

 cv2byQcum->Write();
 grPion->Write();
 grKaon->Write();
 grProton->Write();

 f->Close();



}
// end of program

Float_t GetPt(Float_t fpx,Float_t fpy)
{
  return sqrt(fpx*fpx + fpy*fpy);

}


Float_t GetPhi(Float_t fpx,Float_t fpy)
{
  const double pi = 3.14159;
  double phi=0, phi1=0;
  if(fpx == 0 && fpy > 0) phi = pi/2.0;
  if(fpx == 0 && fpy < 0) phi = 3*pi/2.0;
  if(fpy== 0 && fpx > 0) phi = 0;
  if(fpy ==0 && fpx < 0) phi = pi;
  phi1=TMath::ATan(TMath::Abs(fpy/fpx));
  if(fpx > 0 && fpy > 0) phi = phi1;
  if(fpx < 0 && fpy > 0) phi = pi -phi1;
  if(fpx < 0 && fpy < 0) phi = pi + phi1;
  if(fpx > 0 && fpy < 0) phi = 2*pi - phi1;

  return phi;
}
```

Appendix E:

Visit to various facilities at VECC

Compressed Barionic Matter (CBM) Experiment :

In CBM experiment scientists are looking towards low temperature and high density region of Quantum Chromodynamic (QCD) phase transition diagram. In this region density is 6 to 12 times the nucleus density and E_{lab} = 10- 40 GeV per Nucleon. The focused particle is J/ψ meson as suppuration of it is expected during quark gluon plasma formation. Indian group is preparing detector MUCH (muon chambers) to detect muons coming from the J/ψ decay in FAIR.

J/ψ has comparable decay channel in electrons also; to detect electrons a rich detector is going to be prepared by German collaborators of the CBM experiment. Thick muon absorber is necessary to detect muons due to their high penetration depth and high energy. An iron absorber of 225cm length kept in front of the detector. Most of other particles are stopped inside the absorber. At the detector two straight lines of muon and anti-muon are observed for a single J/ψ. The detector is kept outside the magnetic field. It is an Ag + CO_2 gas detector. Muons interact with the Ag gas and ionize it. The energy of muons is calculated via dE/dx vs total Momentum plot. The CO_2 gas is used for quenching purpose. In simulations the head on collision of Au-Au at 25GeV per nucleon energy is the reference. Via invariant mass relation production of J/ψ can be estimated and theoretical predictions can be checked. This CBM muons system design, development and fabrication of the entire electronics for muon detectors will involve Indian industry as well as scientific community in a big way.

The photon multiplicity detector (PMD) :

The PMD is used to detect number of photons in an event. PMD made by VECC are used in STAR, ALICE and LHC experiments. The basic structure of PMD is: there is a Pb absorber in between two gas detectors with honeycomb structure. The gas detector is used to detect charged particles falling on PMD. The gas used in detector is a mixture of Ar+ CO_2 in the ratio of 70 : 30. At Pb plate photons interact with the atoms and generate electron-positron pairs. These pairs are highly energetic and they emit photons and at the end shower of e^+e^- is detected in more than one hexagonal detector. The hexagonal shape helps to create nearly uniform magnetic field along with usage of whole space. Each hexagonal unit cell consist of an anode made up of gold plated tungsten W wire and hexagons works as cathode having -1400 Volt made up of copper(Cu).The size of hexagons varies according to the experimental need of precision and particle production, higher the number of produced particles smaller the size of hexagonal cells. In ALICE it is almost 5mm thick and almost same depth.

Electronics (concerned with detectors) :

To do a particle physics experiment and get interesting results, we need to accelerate particles at high velocities and collide them and detect or measure their properties and then analyze them. A detector, as the name suggests, is used to detect the particles and then produce electronic signals which can be used for analysis. The interactions of particles with matter play a significant role in their detection. The different types of interactions are electromagnetic and hadronic interaction. Electro-magnetic interactions include excitation, ionization, Cerenkov radiation, bremsstrahlung, transmission radiation while hadronic interactions include hadronic showers. The efficiency of a detector is determined by its electronics, the time lag between two consecutive event detection and transformation of the information. Presently VECC electronics lab is busy in developing ultrafast and small electronic chips which has time lag between two consecutive detection of generated particles is in the nanoseconds. Here in VECC a IC named MANAS(Multiplex Analog Signal Processor) is built for PMD detector which is a completely Indian project. Now they are looking for fast electronics as High Energy Experiment requires.

Grid Computing :

The scientific community especially High Energy Physics community needs a large storage of information, ultrafast data transfer and processing. The major data generating hub is CERN where high energy experiments produce data of around 1pB/year. These data are used by various scientists around the world. Hence to provide access worldwide, store these data and process a technique is developed which is called as grid computing. The main hub of this storage and processing centre is called as Tire0 which is at CERN. A copy of these data is sent to 11 other centres which are at different places of the world. These all are called as Tire1. Each of these 11 Tire1, have approx. 200 Tire2s. VECC is a LHC TIER-2 Institute in India.

To access data from grid one has to connect with grid system via provided certificate to enter in the grid wtih the help of software called 'Middleware' . This software provide connection between hardware and software of different users. The efficiency of grid functioning is several times faster than any individual computer. Here hundreds of processors run simultaneously over the same job and decrease time duration of completion of the job. In this way grid computing provides advantage store huge amount of data, process and transfer information faster. In short, it is an infrastructure that enables flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions and resources. For Monitoring such a huge data acquisition and for security purpose there is a software called 'MONALISA' to monitor the jobs submitted by different Tier2 institutions in a week, a month, a year and more.

Cyclotron :

A cyclotron is a type of particle accelerator. Cyclotrons accelerate charged particles using a high-frequency, alternating voltage (potential difference). A perpendicular magnetic field causes the particles to spiral almost in a circle so that they re-encounter the accelerating voltage many times. A cyclotron has two dees separated by a small gap which has a uniform electric field pointing from one dee to the other. When the particle is released from rest it gets accelerated by the electric field to enter one of the Dee's. Inside the Dee's there is magnetic field perpendicular to the plane of the Dee's so it makes the particle move in a circular path till it comes out of the Dee's and again into the gap. Since the direction of the electric field is now reversed in the gap, the particle now moves to the other Dee with a greater velocity and hence a larger radius, hence eventually by the time the particle moves out of the cyclotron it has very high speed. VECC has two cyclotrons - superconducting (the iron magnet which produces magnetic field is placed at -269 degree Celsius) and room temperature.

India based neutrino observatory (INO) :

Neutrinos are neutral particles. They are produced in reactions in which neutrons change into protons. Initially they were thought of as mass-less particles. However observations indicate that it has some mass. Neutrinos are of three types or flavours

(a) Electron neutrino

(b) Muon neutrino

(c) Tau neutrino

Corresponding to every neutrino there is an anti-particle called anti-neutrino. Neutrinos travel almost at the speed of light and pass easily through matter which makes their detection difficult. It has been observed that when neutrino passes through matter there is a change in its flavour, which is possible only if it has mass. Various detectors have been installed at various locations to study in details about neutrinos, prominent being Super kamiokande ,SNO, MINOS. A neutrino detector (INO) in India has been proposed to be set up. However the location has not been finalized. One of the biggest puzzles that have been solved in neutrinos is the solar neutrino puzzle. The amount of electron neutrinos coming from sun which is detected at the detectors is 1/3 of the number proposed by the standard model. The puzzle was solved once it was found that there is neutrino oscillation. This assumption was found to be true when the total number of neutrinos (all flavours) detected were compared with the standard model values.

Acknowledgments :

After successful completion of my Summer Training Project I would like to thank the many people who have helped me on the path towards my project *"Study of Elliptic Flow of Identied Charge Particles using Relativistic Heavy-ion Collision Models using different flow Methods"*.

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

First of all I am grateful to Indian Academy of Sciences (IAS), Indian National Science Academy (INSA) and National Academy of Sciences (NAS) for giving me opportunity for Research in High Energy Physics as Summer Research Fellow.

I am thankful to Dr. R. K. Bhandari, Director VECC and Dr. Y.P.Viyogi, Head Experimental High Energy Physics group.

I am extremely grateful for the opportunity to have Dr. Bedangadas Mohanty, VECC as an advisor during my two month project. he has been an inspiration to me for heavy-ion physics. What I value most about this project was the opportunity to absorb not only his insights into the specifics of my work, but his fundamental approach to research that emphasizes rigor and first principles. He behaved as one of my friends and taught a lot many lessons of life in the physics.

My sincere thanks to Dr. Ajay Kumar Dash for his involvements and discussions at many stages throughout my project. Thanks to Md. Nasim for giving me valuable data and study material for my project. Many more thanks to Nihar Sahoo for his help in Latex writing and fruitful discussions.My thanks and appreciations also go to my colleague and friend Harsh shah, who willingly helped me out with his abilities.

I am also grateful to several other people of VECC who interacted with me during my visits of different labs.

At last but not least I would like to express my gratitude towards my parents and Department of Physics (University of Rajasthan) for their kind support and encouragement which help me in completion of this project.

References:

1. Methods for analyzing anisotropic flow in relativistic nuclear collisions. by *A.M. Poskanzer and S.A. Voloshin*

2. Flow study in Relativistic Nuclear collisions by Fourier expansion of azimuthal particle distributions. by *S.A. Voloshin and Y. Zhang*

3. Anisotropic transverse Flow introduction in Monte carlo generators for heavy ion collisions. by *M.Mosera, G.Ortonam, M.G.poghsyan, F.prino*

4. Elliptic Flow at Large Transverse Momenta from quark coalescence. by *De'nes Molna'r and S.A. Voloshin*

5. Scaling properties of Azimuthal anisotropy in Au+Au and Cu+Cu collisions at $\sqrt{S}_{NN} = 200$ GeV. *Physical Review Letter (Phenix Collaboration)*

6. Multiphase Transport Model for relativistic heavy ion collisions. by *Zi-Wei Lin, Che Ming ko, Bao- AnLi and Bin Zhang, Subrata pal*

7. Flow Analysis with Q-Cumulants. by *Ante Bilandzic, Raimond Snellings and S.A. Voloshin*