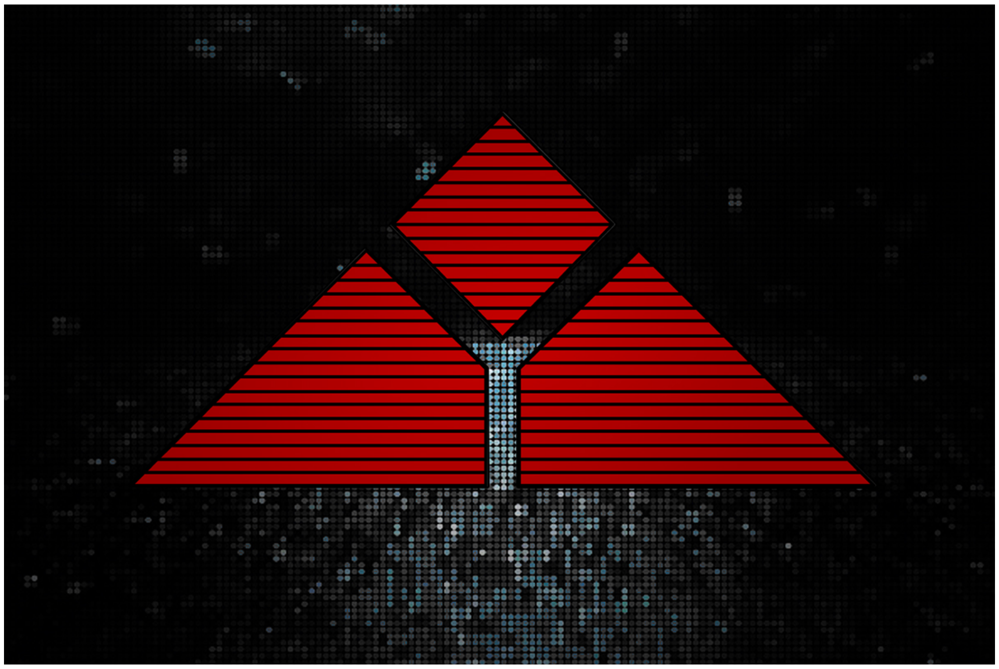
Universiteit Utrecht

Opleiding Natuur- en Sterrenkunde

HBT analysis in simulated heavy-ion collisions

Bachelor Thesis

Hidde Rinsema



Supervisors:

MSc. M.H.P.A. Sas
Subatomic Physics (SAP)

Prof. Dr.T Peitzmann
Subatomic Physics (SAP)

June 12, 2019

Abstract

A way to research heavy ion collisions is to use femtoscopic correlations with probing particles. This gives information on the space-time state of the emmision source. In this thesis the possibity of doing this with photons as probing particles is research by using the event generator THERMINATOR to create date for analysis. This will be inconclusive because the main source of photons, neutral pions, don't decay in THERMINATOR, some solutions are posed to fix this.

Contents

1 Introduction

On the border between France and Switzerland the Large Hadron collider accelerates particles to almost that of the speed of light. Most of the time it uses protons but only for short amounts of time they run the machine with lead ions to form Quark-Gluon-Plasma, or QGP for short, this is a state of matter so hot that the building blocks of protons and neutrons, quarks, while normally bound by the gluons, carriers of the strong force, can move freely between each other. Cosmology predicts that the first microsecond after the big bang the QGP has existed, there is also evidence that QGP is present in the cores of neutron stars [8]. To study the QGP heavy ion's collisions are used (It should be possible to generate a QGP with proton-proton collisions [2]). At the ALICE detector at CERN they observe collisions with lead ions, another place where heavy-ion collisions happen is at the Relativistic Heavy Ion Collider at Brookhaven(RHIC) in the STAR detector where gold ions are used. These collisions produce the QGP only for very short amounts of time(It might be stable in neutron stars [8]) such that we can only observe it by analyzing decay particles. This analysis is done by observing certain probing particles, these particles come from the quark gluon plasma itself and hold information about it. One the the ways of extracting information about the QGP is through second order interferometry of photons(using the intensity of light instead of it's amplitude) which was observer by R. Hanbury-Brown and R.Q. Twiss in 1954 [1] and proved to work in 1957 [3]. This second order interferometry effect is thus called the Hanbury-Brown Twiss effect or HBT-effect for short. In this thesis this effect is used on photons produced in the event generator THERMINATOR to test whether or not it is possible to use photons in heavy ion collisions to obtain information about the QGP.
First the theory behind the HBT-effect is presented, second the THERMINATOR event generator is discussed, what it does, and how it works, briefly, then the method on how the analysis is carried out after that the results are presented and discussed. These results will prove to be inconclusive and will need more research whether if the HBT effect of photons in heavy ion collisions can be observed.

2 Theory

In this section we will go through what the HBT effect is and in what ways it is used in this thesis, then the event generator, THERMINATOR, briefly explained.

2.1 Hanbury-Brown Twiss effect

In 1956 R. Hanbury Brown and R.Q. Twiss demonstrated how a second order interferometry effect could be used to determine the angular size of Sirius, later in 1957 they showed how this effect works in theory [3]. While intensity inteferometry is nothing new, previously it had only be used in radio astronomy, it was new for optical observations.
In 1959 Goldhaber et. al. found corrolations between emitted pions from proton-antiproton annihilation events [4]. Since then it has been used in particle physics to determine space-time dimensions.

A short qualitative quantum mechanical explanation of the effect, given in [5] and [9] goes as follows: Take two points a and b on a particle source, doesn't really matter which particles as long as both a and b send out the same particles, which send out particles to two detectors, as seen in fig. 1.

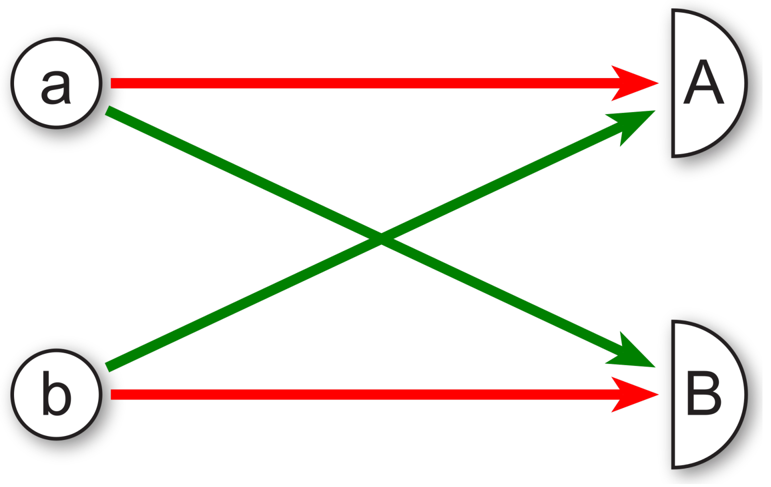


Figure 1: Two source points, a and b, emit particles, detected by A and B, the colors represent the two ways particles can be detected indistinguishable from each other.

In this example there are 4 options:
1. a sends out a particle to A and B
2. b sends out a particle to A and B
3. a sends out a particle to A and b sends out a particle to B
4. a sends out a particle to B and b sends out a particle to A
The first two are distinguishable and do not produce interferometry, however process 3 and 4 are indistinguishable, and these amplitudes interfere constructively to give a greater detection probability.
However, if said particles are charged it will produce something that is called nati-bunching and will make it so that the wave amplitdues interfere destructively, which is called anti-bunching(see ref [6]).

2.1.1 Classical description of the HBT-effect

A more quantitative, all be it classical, description of the HBT effect goes as follows. Looking again at fig. 1 with it's two source points, a and b, imagine that source a produces an electromagnetical wave of amplitude $\alpha e^{ik|\vec{r}-\vec{r}_a|+i\phi_a}/|\vec{r}-\vec{r}_a|$ and that source b produces a wave $\beta e^{ik|\vec{r}-\vec{r}_b|+i\phi_b}/|\vec{r}-\vec{r}_b|$ where both ϕ are random phases, so the amplitude at detector A becomes

$$A_A = \frac{1}{L}\left(\alpha e^{ikr_{Aa}+i\phi_a} + \beta e^{ikr_{Ab}+i\phi_b}\right) \tag{1}$$

With r_{Aa} being the distance from source a to detector A and r_{Ab} similarly the distance from source b to detector A. And L being the average distance between the source points and the detectors. The intensity at detector A becomes then

$$I_A = \frac{1}{L^2}\left(|\alpha|^2 + |\beta|^2 + \alpha^*\beta e^{i(k(r_{Ab}-r_{Aa})+\phi_b-\phi_a)} + \alpha\beta^* e^{-i(k(r_{Ab}-r_{Aa})+\phi_b-\phi_a)}\right) \tag{2}$$

For I_B follows something similar, when averaged over random phases the average intensity is

$$\langle I_A\rangle = \langle I_B\rangle = \frac{1}{L^2}\left(\langle|\alpha|^2\rangle + \langle|\beta|^2\rangle\right) \tag{3}$$

However, if multiplied before averaging an extra term appears. Then after averaging over the phases it is

$$\langle I_A I_B\rangle = \langle I_A\rangle\langle I_B\rangle + \frac{2}{L^4}|\alpha|^2|\beta|^2 cos(k(r_{Aa} - r_{Ba} - r_{Ab} + r_{Ba})) \tag{4}$$

When looking at the correlation function it becomes clear that the *cos* term is the source of the HBT effect here.

$$C(\vec{d}) = \frac{\langle I_A I_B\rangle}{\langle I_A\rangle\langle I_B\rangle} = 1 + 2\frac{\langle|\alpha|^2\rangle\langle|\beta|^2\rangle}{(\langle|\alpha|^2\rangle + \langle|\beta|^2\rangle)^2} cos(k(r_{Aa} - r_{Ba} - r_{Ab} + r_{Ba})) \tag{5}$$

In astronomy this formula works, since distances are great, and wave amplitudes are easily measured, however for use in particle physics one has to start with a more quantum mechanical description, in the next section a start from probabilities is made and a short derivation is made.

2.1.2 Short derivation of Corrolation functions

For analysis of the generated data I will use the corrolation between the particles, for this I will shortly derive the useful form of the corrolation function. This short derivation will mostly follow section 5 and 6 of [9]. Following up on the description given in section 2.1, consider the probability of detection of detecting a particle in state ϕ_i at A

$$P_{\vec{k}}(A; i) = \int dx dx' e^{ikx} \phi_i^* s_A(x, x') e^{-ikx'} \phi_i(x'). \tag{6}$$

Where s is the detector response function, in general to detect a pion at A, the probability to detect it is given by

$$P_{\vec{k}}(A) = \int dx dx' e^{ik(x-x')} s_A(x, x') \langle\pi^\dagger(x)\pi(x')\rangle. \tag{7}$$

Where $\langle\pi^\dagger(x)\pi(x')\rangle$ is the single particle, in this case pion, correlation function. Now consider the detection of two particles, then the question becomes, if a pion is detected at A, do other nearby detectors have a higher probability of detection?
The wave function for two particles is $\phi(\vec{r}, \vec{r}', t) = (\phi_i(\vec{r}, t)\phi_j(\vec{r}', t) + \phi_i(\vec{r}', t)\phi_j(\vec{r}, t))/\sqrt{2}$. Then the detection of a pion with momentum $\vec{k}$ at detector A and a detection at a detector B of another pion with momentum $\vec{k}'$ is

$$P_{\vec{k},\vec{k}'}(A, B; i, j) = \int dx dx'' e^{ik(x-x'')} s_A(x, x'') \int dx' dx''' e^{ik(x'-x''')} s_B(x', x''')$$

$$\begin{aligned}
&\times(\phi_i(x)\phi_j(x') + \phi_i(x')\phi_j(x)) * (\phi_i(x'')\phi_j(x''') + \phi_i(x''')\phi_j(x'')) \\
&= P_{\vec{k}}(A;i)P_{\vec{k}'}(B;j) + P_{\vec{k}}(A;j)P_{\vec{k}'}(B;i) \\
&+ \mathcal{A}_{\vec{k}}(A;i,j)\mathcal{A}_{\vec{k}'}(B;j,i) + \mathcal{A}_{\vec{k}}(A;j,i)\mathcal{A}_{\vec{k}'}(B;i,j)
\end{aligned} \tag{8}$$

where

$$\mathcal{A}_{\vec{k}}(C;i,j) = \int dxdx' e^{ik(x-x'')} s_C(x,x'')\phi_i^*(x)\phi_j(x''). \tag{9}$$

The first two terms are normal terms, the last terms with $\mathcal{A}$ enhance the detection and are the HBT terms. Much like the extra terms in the classical description
The corrolation of two particles in terms of its momentum, using these probabilities is defined as

$$C(\vec{p_1},\vec{p_2}) = \frac{P(\vec{p_1},\vec{p_2})}{P(\vec{p_1}P(\vec{p_2})} \tag{10}$$

where $P(\vec{p_1})$ is the probability of observing a single particle with momentum $\vec{p_1}$ this is the 4-momentum of a particle,

$$\vec{p} = \left(E/c, p_1, p_2, p_3\right)$$

and $P(\vec{p_1},\vec{p_2})$ the probability observing a pair of particles with momenta $\vec{p_1}$ and $\vec{p_2}$.
This can be understood as that $P(\vec{p_1})$ is the probability that a given particle has momentum $\vec{p_1}$, then $P(\vec{p_1})$ is defined as

$$P(\vec{p_1}) = \frac{1}{\langle n\rangle}\frac{d^3n}{d^3p_1} \tag{11}$$

The definition of the corrolation function then becomes

$$C(\vec{p_1},\vec{p_2}) = \frac{\langle n\rangle^2}{\langle n(n-1)\rangle}\frac{\frac{d^6n}{d^3p_1d^3p_2}}{\frac{d^3n}{d^3p_1}\frac{d^3n}{d^3p_2}} \tag{12}$$

Where $\frac{d^6n}{d^3p_1d^3p_2}$ is the pair yield per event and $\frac{d^3n}{d^3p}$ the particle yield per event [7][?] and is normalized.
The particle yield per event is multiplied with other particles, and the pair yield is devided by it. experimentally this is written as:

$$C(Q_{inv}) = \frac{A(Q_{inv})}{B(Q_{inv})} \times \xi(Q_{inv}) \tag{13}$$

Where Q_{inv} is the invariant momentum difference between two particle pairs $[(p_1-p_2)^2]^{1/2}$, $A(Q_{inv})$ is the yield of particle pairs of the same event and $B(Q_{inv})$ the yield of particle pairs of different events, the ξ term is a correction term used to deal with the unpreciseness of detectors, but since this thesis covers a simulation it will not be used. In data analysis this becomes simply the devision of two histograms, one with Q_{inv} of the same event, and the other of Q_{inv} of different events
To analyse this function parametrization is used on the correlation, C, the way to parametrize this function correction is properly explained in [9] here the simpelest one is used.

$$C(q) = 1 + \lambda e^{-Q_{inv}^2 R^2} \tag{14}$$

Where λ is the chaosity parameter, $\lambda = 1$ for a chaotic blackbody source,$\lambda = 0$ for a non-chaotic source, such as a laser. R is the radius of the particle emission source and thus possibly says something about the QGP, and is the object of interest.

2.2 THERMINATOR

THERMINATOR is a monte carlo event generator and stands for THERMal heavy IoN generATOR, one that uses randomness to produce it's events, that offers a simulation with thermal particle production. It is written in C++ and uses CERN's Root enviroment.
Each particle is created on what is called the freeze-out hypersurface, a surface and point in time from which hadronization will take place. From this "initial" state THERMINATOR resonances of particles are created which quickly decay to more stable particles, THERMINATOR gives us all the data of every particle, space-time positions, momentum, which particle it is, the particle from which it orininated and the particle that started from the freeze-out surface and began the cascade.
In fig. 2 it is shown how the event generation works in a block chain. scheme.
In the configurator it applies the settings and how it should be configurated, the particle database contains the decay tables and channels and will determine how the cascades behave.
The event generator generates said events, and takes into account what freeze-out model is used. This generates the event files, each of which is a ROOT file containing a Tree structure.
In short ROOT is a software toolkit developed by CERN for the analysis of big data, it is written in C++. THERMINATOR produces .root files, these files have a special structure called a Tree which consists of brances of a type of data and the leaves making up the actual data, it allows for quick access, and small storing space. For a more information, see [14] and [12].
Another program that was considered, due to some difficulties with THERMINATOR, was HYDJET++ which has similar thermal particle production as THERMINATOR [13]

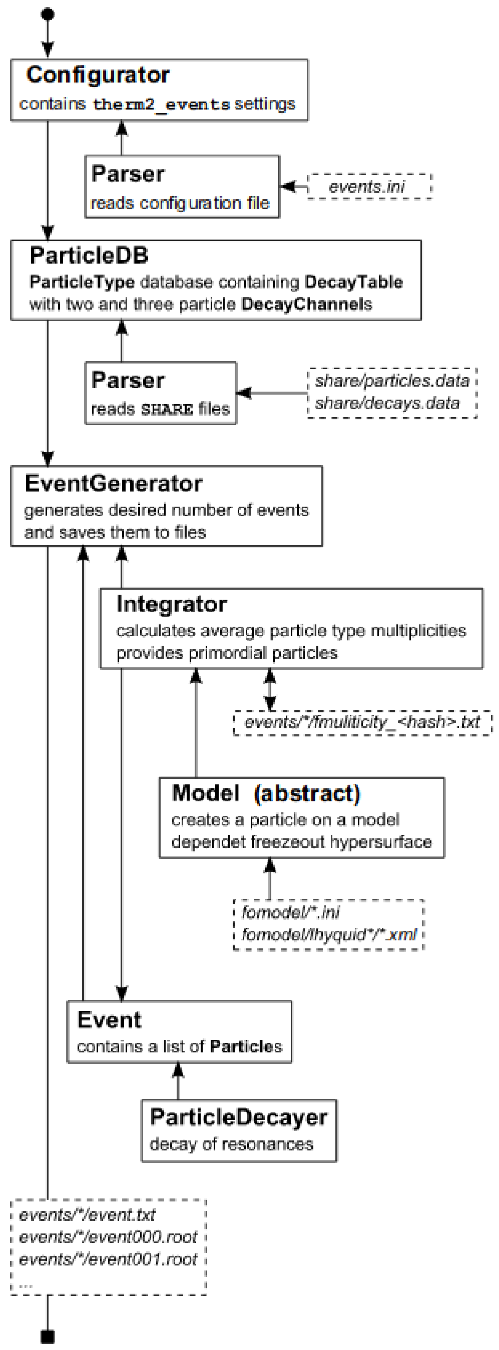

Configurator
contains therm2_events settings
Parser
reads configuration file
events.ini
ParticleDB
ParticleType database containing DecayTable
with two and three particle DecayChannels
Parser
reads SHARE files
share/particles.data
share/decays.data
EventGenerator
generates desired number of events
and saves them to files
Integrator
calculates average particle type multiplicities
provides primordial particles
events/*/fmuliticity_<hash>.txt
Model (abstract)
creates a particle on a model
dependet freezeout hypersurface
fomodel/*.ini
fomodel/lhyquid*/*.xml
Event
contains a list of Particles
ParticleDecayer
decay of resonances
events/*/event.txt
events/*/event000.root
events/*/event001.root
...

Figure 2: Block chain scheme on how THERMINATOR generates events, it creates a TTree object, the way of storing data in .root files, with particles in it. source

3 Methodology

In the Methodology the analysis process will be explained, this process will be conducted with the raw .root files generated by the THERMINATOR program. First the setup of the code is explained then the corrolation plots themselves.

3.1 Code

The root files obtained from the THERMINATOR simulation consist of 3 TTree objects, particle, event and parameter. The particle tree has the information conserning the particles themselves, things such as momentum, position, but also what the original particle was and whether or not it has decayed. The event tree stores the information on the events, their identification and the number of entries per event. Then there is the parameter tree which holds the information on the parameters used to create the events.
The method that was first used was to use 2 for loops over all the entries in the particle Tree but the total amount of particles in the files is about 10^7 using the for loop in the for loop method the amount of particles that had to be read into the file was 10^{14} particles, this is no option to continiously run for it takes to long, and even uses double the amount of data needed.
A single for loop was to be used to reduce runtime, it runs over one event and picks out the useful particles, in this case photons, then it does do a double for loop to calculate the corrolations, then stores the particles to get the corrolations between the particles of one event and another. It does this for all events, which results in two histograms, one of the invariant momentum difference between particles of the same event and one with invariant momentum difference between all particles of all events, these are then devided to get the corrolation plots.

3.2 Corrolation Plots

Now that A(Q_{inv}) and B(Q_{inv}) have been calculated dividing A(Q_{inv}) by B(Q_{inv}) for one event this is shown in fig. 3. Fig. 3 shows the corrolation of photons from one event, normally there are not enough photons to get a correct fit, but it will serve well for use as an example. For use in analysis it is necessary to have it normalized. By considering a part of the histogram where there are no HBT-effects and multiplying the corrolation plot with a factor that brings it in that range to 1 we get the normalization. In this case it is done by C(Q_{inv}) with the ratio of entries of B(Q_{inv}) and A(Q_{inv}), which gives, as can be seen in fig. 4.

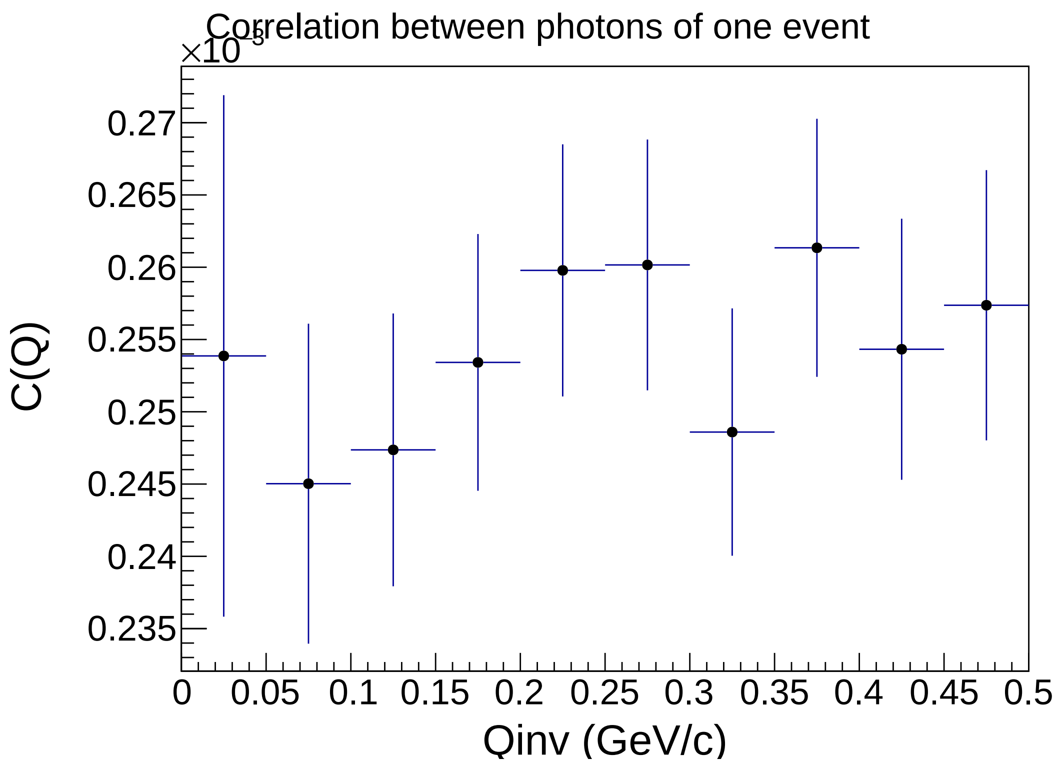


Figure 3: C(Q_{inv}) of one event(eventID 13419671, see Appendix A) without any normalization no corrolation can be seen

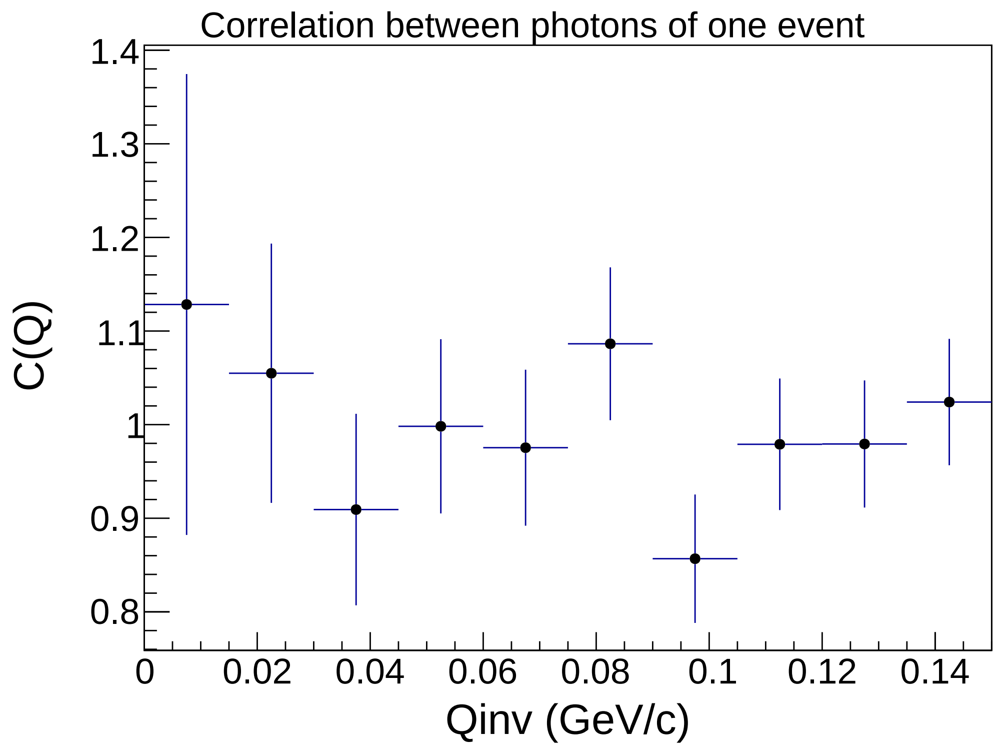


Figure 4: The same as in figure 4 but now it's nicely centered around 1, suited for analysis

3.2.1 Fit of the corrolation function

The data on itself doesn't tell much only the deviations of the normalization, to get usefull data out of the corrolation function a fit needs to be applied and the parameters need to be read.
The fit that is used is a simple gaussian one. The derivation of which is found in section 6 of [9]

$$C(q) = 1 + \lambda e^{-Q_{inv}^2 R^2} \tag{15}$$

. The parameters are R and λ, where R is the radius of the emission source and λ is the chaoticity parameter. This is a fit that neglects any interactions happening after the particles have been released from their source-points. The normalization factor is usually considered in the fit themselves, here it is used before the fit is applied.

4 Results

From the histogram we obtain the corrolation plots of the photons, for all produced photons, regardless of source.
As can be seen in fig. 5 there is no corrolation to be seen for all photons, so I checked for photons of different sources. Particularly those with lower lifetimes than the main source of the photons in the simulation, eta mesons.

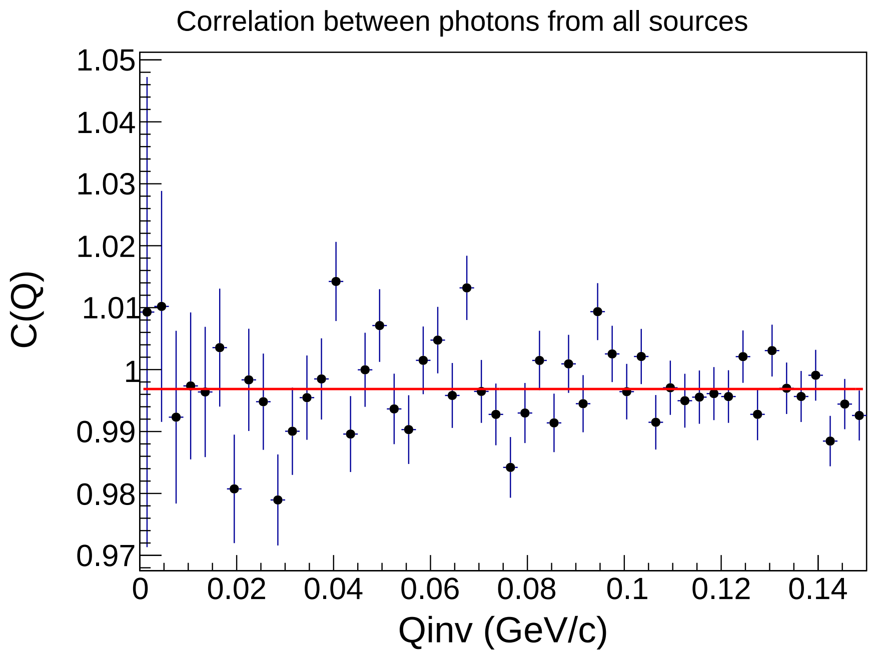


Figure 5: Corrolation of photons from between all the 1001 events, a fit function, based on HBT analysis has been added of the form $C(q) = 1 + \lambda e^{-Q_{inv}^2 R^2}$ the values of the two parameters are $\lambda = -3.14 * 10^{-3} \pm 7.32 * 10^{-4}$ and $R = 2.44 * 10^{-4} \pm 4.37$.

So instead the following particles are used for photon sources, the ω meson, with a mean lifetime of 10^{22} seconds, the neutral Σ baryon, with a lifetime of around 10^{19} and for reference the η meson as well 10^{19}. Both in fig 6 ,7 and 8 there are also no significant corrolations to be found, and the uncertanties given for the fit are too large for this data to hold significance(same as the plot for all photons).

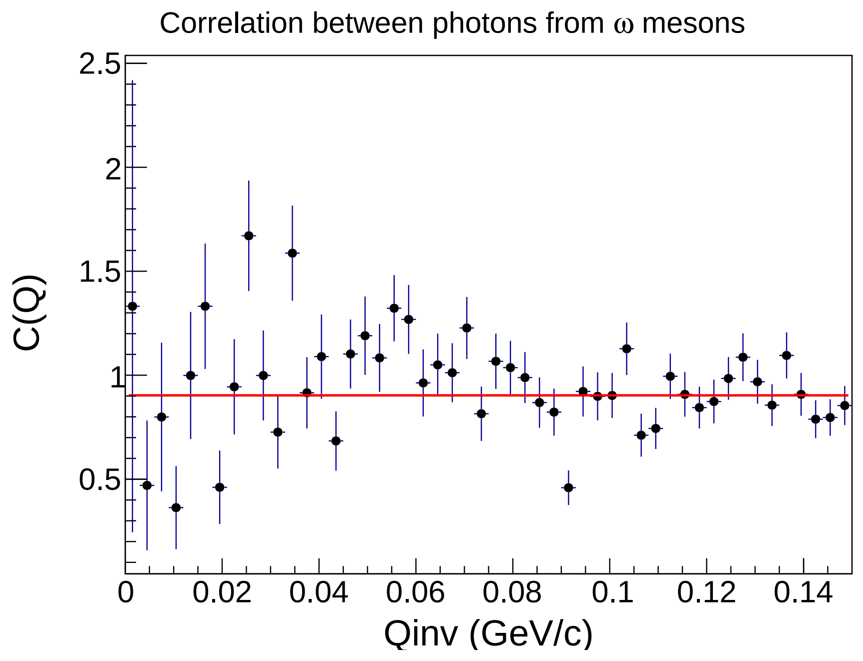


Figure 6: Corrolation of photons from between all the 1001 events, but limiting photon sources to just ω mesons, a fit function, based on HBT analysis has been added of the form $C(q) = 1 + \lambda e^{-Q^2_{inv}R^2}$ the values of the two parameters are $\lambda = -9.69 * 10^{-2} \pm 1.84 * 10^{-2}$ and $R = -4.49 * 10^{-4} \pm 3.22$.

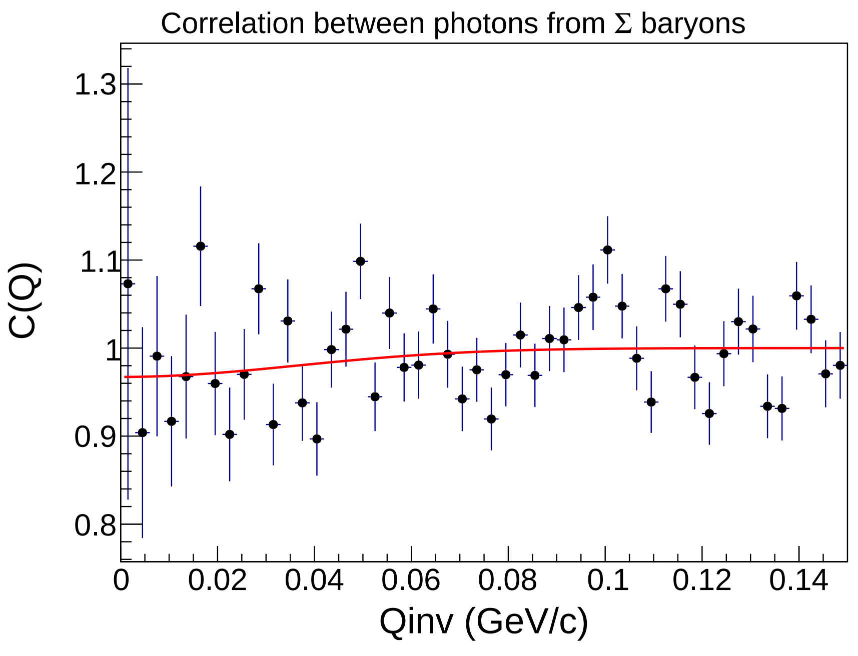


Figure 7: Corrolation of photons from between all the 1001 events, but limiting photon sources to just Σ mesons, a fit function, based on HBT analysis has been added of the form $C(q) = 1 + \lambda e^{-Q^2_{inv}R^2}$ the values of the two parameters are $\lambda = -3.28 * 10^{-2} \pm 2.31 * 10^{-2}$ and $R = 1.94 * 10^{-1} \pm 9.57$.

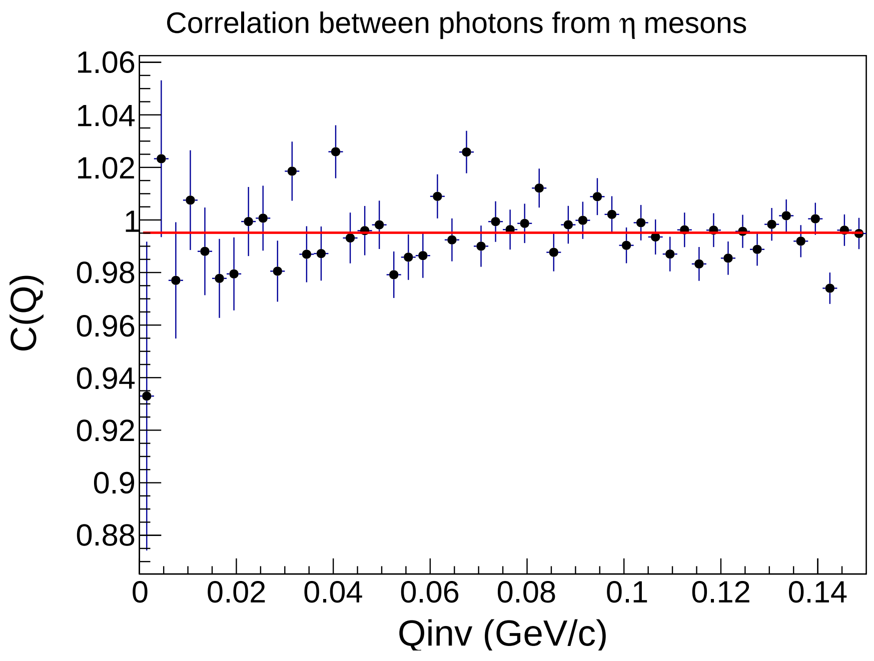


Figure 8: Corrolation of photons from between all the 1001 events, but limiting photon sources to just η mesons, a fit function, based on HBT analysis has been added of the form $C(q) = 1 + \lambda e^{-Q_{inv}^2 R^2}$ the values of the two parameters are $\lambda = -4.87 * 10^{-3} \pm 1.11 * 10^{-3}$ and $R = -4.52 * 10^{-1} \pm 3.28$.

5 Conclusion

The results show that there is no corrolation present in the interference of photons. To check this plots have been made of photons but from different sources. This begs the question whether or not there even is a possible HBT effect in this data, to check this femtoscopic corrolations where tested on pions (π^+) and kaons (K^+) since these are the particles mostly used in the literature (see for example [4]).

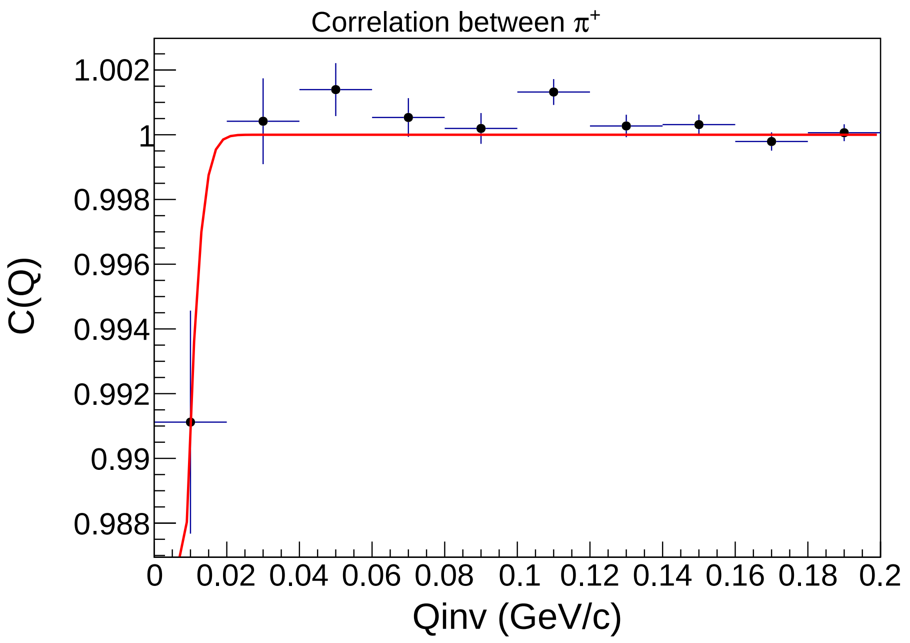


Figure 9: Corrolation of neutral pions, with a fit function plotted over it, a clearer, but negative, corrolation, can be seen all be it not a too great one. It shows that the THERMINATOR simulation can produce HBT effects.

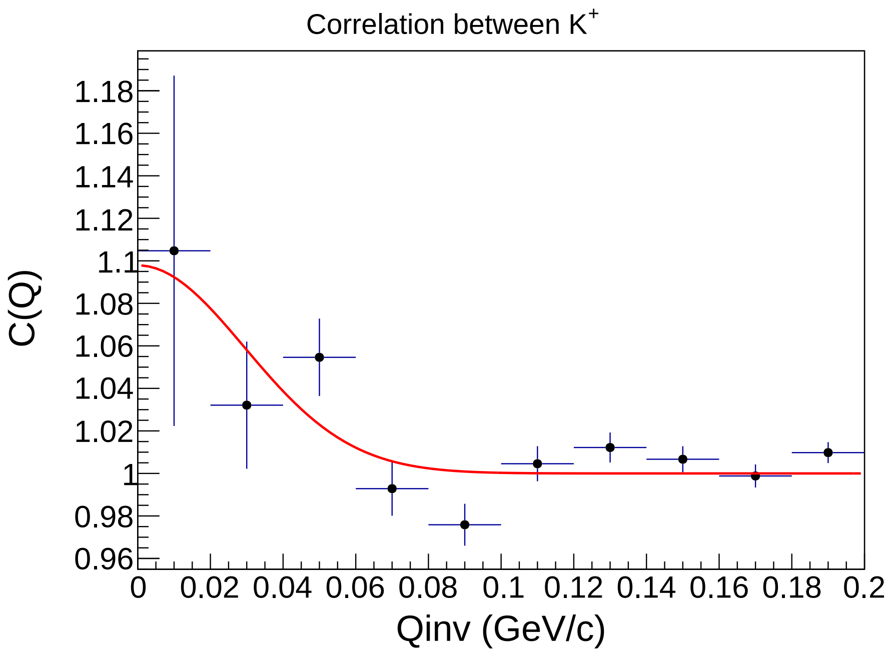


Figure 10: Corrolation of positvely charged kaons with a fit function plotted over it, a clear sign of a HBT-effect is seen.

Fig 9 and 10 show that there can be a corrolation effect between particles producing a HBT effect, however, this does not happen with photons.

6 Discussion

In this thesis we tried to show whether or not it was possible to use photons in conjunction with the HBT-effect to obtain the system-size of the freeze-out surface of heavy-ion collisions, and possibly obtain information about the underlying Quark Gluon Plasma, from simulated data using the THERMINATOR event generator.
Unfortunately it was not possible to do this since there weren't any signs of the HBT effect being present in the data, for photons, neutral pions and positve kaons however, do produce a HBT-effect.
It showed that there is something wrong with the data for photons. The problem is that in the simulation pions do not decay into photons by $\pi^0 \rightarrow \gamma\gamma$ which should be the biggest source of photons (see [15]). This lack of decay is coming from one of the settings in the THERMINATOR program (see 2.2 for a brief overview, [14] for a detailed explanation). There are two likely possibilities for this, either the pion can't decay, because it's decay isn't in the decay files, or the time for which THERMINATOR simulates the collisions is too short for pion decay to happen. Due to time constraints this could not be explored further
The outlook of such a research as this would be that, to find a way to make the simulation decay the pions and then try to find a HBT-effect in the data. After that one can look at possible applications on real data, however one now deals with final resolutions, mis-identified particles and background deficiencies, another problem is margins of error, in this analysis there are no margins of error in the data of the particles, there are errors in the applied fits.

To deal with that one would need to make corrections in the data. In 2.1.2 we discussed briefly formula 13 which has a extra term in it that signifies these corrections that would be made if you apply the experimental formula for the corrolations to real data.

References

[1] R. Hanbury Brown , R.Q. Twiss
Correlation between Photons in two Coherent Beams of Light
`https://www.nature.com/articles/177027a0` last accessed 08-06-2019

[2] M. T. AlFiky, O. T. ElSherif, A. M. Hamed
Quark Gluon Plasma Formation in Proton-Proton Collisions Using PYTHIA
`https://arxiv.org/abs/1902.05114` last accessed 08-06-2019

[3] R Hanbury Brown, R.Q. Twiss, 1957
Interferometry of the Intensity Fluctuations in Light. I. Basic Theory: The Correlation between Photons in Coherent Beams of Radiation
`https://web.archive.org/web/20050124042837/http://www.strw.leidenuniv.nl/~tubbs/classic_papers/hanbury_brown_et_twiss_1957.pdf`

[4] G. Goldhaber, W.B. Fowler, S. Goldhaber, T.F. Hoang, T.E Kalogeropoulos, W.M. Powell, 1959
Pion-Pion Correlations in Antiproton annihilation events
`https://escholarship.org/uc/item/7nw6p1br` last accessed 31-05-2019

[5] U. Fano, 1961 *Quantum Theory of Interference Effects in the Mixing of Light from Phase-Independent Sources*
`http://adsabs.harvard.edu/abs/1961AmJPh..29..539F` last accessed 31-05-2019

[6]

[7] U. Heinz, 1998 *HANBURY BROWN TWISS INTERFEROMETRYIN HIGH ENERGY NUCLEAR AND PARTICLE PHYSICS*
`https://arxiv.org/pdf/hep-ph/9806512.pdf` last accessed 09-06-2019 M. Henny, S. Oberholzer, C. Strunk, T. Heinzel, K. Ensslin, H. Holland, C. Schonenberger. 1999 *The Fermionic Hanbury Brown and Twiss Experiment*
`http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.205.4817&rep=rep1&type=pdf` last accessed 31-05-2019

[8] E. Annala, T. Gorda, A. Kurkela, J. Nattila, A. Vuorinen. 2019 *Quark-matter cores in neutron stars*
`https://arxiv.org/pdf/1903.09121.pdf` last accessed 31-05-2019

[9] G. Baym, 1997 *The physics of Hanbury Brown?Twiss intensity interferometry:from stars to nuclear collisions.*
`https://www.phenix.bnl.gov/WWW/publish/seto/wonyong/zakopane.pdf` last accessed 31-05-2019

[10] M.A. Lisa, S. Pratt, R. Soltz, U. Wiedermann, 2005 *Femtoscopy in Relativistic Heavy IonCollisions: Two Decades of Progress*
`https://arxiv.org/pdf/nucl-ex/0505014.pdf` last accessed 31-05-2019

[11] E. Frodermannm U. Heinz, 2018 *Photon HBT interferometry for non-central heavy-ion collisions*
`https://arxiv.org/pdf/0907.1292.pdf` last accessed 31-05-2019

[12] *Root users guide* `https://root.cern.ch/root/htmldoc/guides/users-guide/ROOTUsersGuide.html` last accessed 31-05-2019

[13] I.P. Lokhtin, L.V. Malinina, S.V. Petrushanko, A.M. Snigirev, I. Arsene, K. Tywoniuk 2008 *Heavy ion event generator HYDJET++(HYDrodynamics plus JETs)* `https://arxiv.org/pdf/0809.2708.pdf` last accessed 31-05-2019

[14] M. Chojnacki, A. Kisiel, W. Florkowski, W. Broniowski, 2011 *THERMINATOR 2:THERMal heavyIoNgenerATOR2* `https://arxiv.org/pdf/1102.0273.pdf` last accessed 31-05-2019

[15] O. Linnyk, V.P. Konchakovski, W. Cassing, E. L. Bratkovskaya, 2013 *Photon elliptic flow in relativistic heavy-ion collisions: hadronic versus partonic sources*
`https://arxiv.org/abs/1304.7030` last accessed 07-06-2019

[16] D. Miskowiec, S. Voloshin, 1998 *On the normalization of the HBT correlation function* `https://arxiv.org/pdf/nucl-ex/9704006.pdf` last accessed 09-06-2019

A Appendix

```
#include <TH1D.h>
#include <TTree.h>
#include <TCanvas.h>
#include <TF1.h>
#include <TMath.h>
#include <TStyle.h>
#include "events2chain.C"
#include "model2legend.C"
#include "hist2xml.C"

#define  _FIGURE_NAME_ "HBT2"
#define  _N_HISTOGRAMS_ 6

void HBT2(TString aEventDir = "./events/", Int_t aEventFiles = 3)
{
  static ParticleCoor Particle;
  Int_t    Events;
  TChain* Chain = events2chain(aEventDir, aEventFiles, &Particle,
     &Events);
  Int_t    XBins  = 100;
  Float_t XMin    = 0.0;
  Float_t XMax    = 3.0;
  Float_t dX      = (XMax - XMin) / XBins;
  TH1D*    H1D[_N_HISTOGRAMS_];
  Float_t piid;
  cout<<"Please enter Particle ID";
  cin>>piid;
// Create histograms
  H1D[0] = new TH1D("H0", "#pi^{+}", XBins, XMin, XMax);
  H1D[0]->GetXaxis()->SetTitle("p_{T} [GeV/c]");
  H1D[0]->GetYaxis()->SetTitle("dN/(2 #pi p_{T} dp_{T} dy)");
  H1D[0]->Sumw2();
  H1D[1] = (TH1D*) H1D[0]->Clone("H1");   H1D[1]->SetTitle("K^{+}")
     ;
  H1D[2] = (TH1D*) H1D[0]->Clone("H2");   H1D[2]->SetTitle("p");
  H1D[3] = (TH1D*) H1D[0]->Clone("H3");   H1D[3]->SetTitle("#gamma"
     );
  H1D[4] = (TH1D*) H1D[0]->Clone("H4");   H1D[4]->SetTitle("#pi^{0}
     ");
  H1D[5] = (TH1D*) H1D[0]->Clone("H5");   H1D[5]->SetTitle("#eta");

  TH1D* CorrA = new TH1D("H1","CorrA", XBins*0.1,0,0.2);
  CorrA->GetXaxis()->SetTitle("Qinv (GeV/c)");
  CorrA->GetYaxis()->SetTitle("N");
  CorrA->Sumw2();

  TH1D* CorrB = new TH1D("H1","CorrB", XBins*0.1,0,0.2);
  CorrB->GetXaxis()->SetTitle("Qinv (GeV/c)");
  CorrB->GetYaxis()->SetTitle("N");
  CorrB->Sumw2();


  TH1D* OpeningAngleSame = new TH1D("H1","OpeningAngleSame", XBins
     *40,0.0,4.0);
  OpeningAngleSame->GetXaxis()->SetTitle("OpeningAngleSame");
  OpeningAngleSame->GetYaxis()->SetTitle("");
  OpeningAngleSame->Sumw2();

  TH1D* OpeningAngleMixed = new TH1D("H1","OpeningAngleMixed",
     XBins*40,0.0,4.0);
  OpeningAngleMixed->GetXaxis()->SetTitle("OpeningAngleMixed");
  OpeningAngleMixed->GetYaxis()->SetTitle("");
  OpeningAngleMixed->Sumw2();

// Fill histograms
  Float_t Rap, Pt;
  Int_t   pid;
  Int_t   nEta = 0; Int_t nPhotons = 0; Int_t nPhotonsFromEta = 0;
      Int_t nPionsneutral = 0; Int_t nPionsplus = 0; Int_t
     nPhotonsFromPi0 = 0; Int_t nThermalPhotons =0;

  Double_t Qinv = 0;
  Double_t Mass = 0;
  Double_t OpeningAngle = 0;
  Double_t Phi = 0;
  Double_t PhiMax = 0;

  vector<ParticleCoor> ParticleVector;
  deque<vector<ParticleCoor>> BackgroundBuffer;
  vector<ParticleCoor> ParticleVectorBck;
  TVector3 vecPart1, vecPart2;

  Chain->GetEntry(0);
  Int_t eventID = Particle.eventid;
  std:: cout << "the chain has " << Chain->GetEntries() << "
     entries " << endl;
  std::cout << "Found a new event with ID: " << Particle.eventid
     << endl;
  for(Int_t i=0; i<Chain->GetEntries(); i++) {
    Chain->GetEntry(i);
    if( eventID != Particle.eventid ){
      std::cout << "Concluding event" << endl;
      std::cout << "nEta = " << nEta << endl;
      std::cout << "nPhotons = " << nPhotons << endl;
      std::cout << "nPhotonsFromEta = " << nPhotonsFromEta << endl
         ;
      std::cout << "nPionsneutral = " << nPionsneutral << endl;
      std::cout << "nPhotonsFromPi0 = " << nPhotonsFromPi0 << endl
         ;
      std::cout << "nPionsplus = " << nPionsplus << endl;
      std::cout << "nThermalPhotons = " << nThermalPhotons << endl
         ;
      std::cout << "Size of vector = "<<ParticleVector.size()<<
         endl;

      nEta = 0;
      nPhotons = 0;
      nPhotonsFromEta = 0;
      nPionsplus = 0;
      nPionsneutral = 0;
      nPhotonsFromPi0 = 0;
      nThermalPhotons = 0;
      eventID = Particle.eventid;
//      // same event mixing
      for(Int_t i = 0; i < ParticleVector.size(); i++){
        vecPart1.SetXYZ(ParticleVector.at(i).px,ParticleVector.at(
           i).py,ParticleVector.at(i).pz);
        Phi = vecPart1.Phi();
        for(Int_t j = i + 1; j < ParticleVector.size(); j++){
          vecPart2.SetXYZ(ParticleVector.at(j).px,ParticleVector.
             at(j).py,ParticleVector.at(j).pz);
          Phi = vecPart2.Phi();
          OpeningAngle = vecPart1.Angle(vecPart2);
         Qinv =TMath::Sqrt(pow(ParticleVector.at(i).px -
            ParticleVector.at(j).px,2) + pow(ParticleVector.at(i).
            py - ParticleVector.at(j).py,2) + pow(ParticleVector.at
            (i).pz - ParticleVector.at(j).pz,2)  - pow(
            ParticleVector.at(i).e - ParticleVector.at(j).e,2) );
          CorrA->Fill(Qinv);

          OpeningAngleSame->Fill(OpeningAngle);
        }
      }
      // different event mixing
      for(Int_t i = 0; i < BackgroundBuffer.size(); i++){
        vector<ParticleCoor> BackgroundVector = BackgroundBuffer.
           at(i);
        for(Int_t j = 0; j < ParticleVector.size(); j++){
          vecPart1.SetXYZ(ParticleVector.at(j).px,ParticleVector.
             at(j).py,ParticleVector.at(j).pz);
          Phi = vecPart1.Phi();
          for(Int_t k = 0; k < BackgroundVector.size(); k++){
            vecPart2.SetXYZ(BackgroundVector.at(k).px,
               BackgroundVector.at(k).py,BackgroundVector.at(k).pz)
               ;
            Phi = vecPart2.Phi();
            OpeningAngle = vecPart1.Angle(vecPart2);
           Qinv =TMath::Sqrt(  pow( ParticleVector.at(j).px -
              BackgroundVector.at(k).px ,2) + pow( ParticleVector.
              at(j).py - BackgroundVector.at(k).py ,2) + pow(
              ParticleVector.at(j).pz - BackgroundVector.at(k).pz
              ,2) - pow( ParticleVector.at(j).e - BackgroundVector.
              at(k).e ,2) );

             CorrB->Fill(Qinv);


            OpeningAngleMixed->Fill(OpeningAngle);
          }
        }
      }

      BackgroundBuffer.push_back(ParticleVector);
      if(BackgroundBuffer.size() == 3){
          BackgroundBuffer.pop_front();
      }


      ParticleVector.clear();

      std::cout << endl << "Found␣a␣new␣event␣with␣ID:␣" <<
         Particle.eventid << endl;
    }
    if(Particle.e == Particle.pz)
      continue;
    Rap = 0.5 * TMath::Log((Particle.e+Particle.pz) / (Particle.e-
       Particle.pz));
    if( TMath::Abs(Rap) >= 1.0 )
```

```
      continue;
    pid = Particle.pid;

    Pt  = TMath::Sqrt(Particle.px*Particle.px + Particle.py*
       Particle.py);
    if(pid == 211) {
      nPionsplus++;
      H1D[0]->Fill(Pt, 1.0/Pt);
      }
    if(pid == 321)
      H1D[1]->Fill(Pt, 1.0/Pt);
    if((pid == 2212) && (Particle.fatherpid != 3122))
      H1D[2]->Fill(Pt, 1.0/Pt);
    if(pid == 22){
      H1D[3]->Fill(Pt, 1.0/Pt);
      nPhotons++;
    }
    if(pid == 111) {
      nPionsneutral++;
      H1D[4]->Fill(Pt, 1.0/Pt);
      }
    if(pid == 221){
      H1D[5]->Fill(Pt, 1.0/Pt);
      nEta++;
    }
    if(pid == 22 && Particle.fatherpid == 111){
      nPhotonsFromPi0++;
  }
    if(pid == 22 && Particle.fatherpid == 221){
      nPhotonsFromEta++;
  }
    if(pid ==22 && Particle.fatherpid != 221){
      nThermalPhotons++ ;
  }
    if(pid ==piid){
      ParticleVector.push_back(Particle);
    }

  }

    std::cout << "Concluding event" << endl;
    std::cout << "nEta = " << nEta << endl;
    std::cout << "nPhotons = " << nPhotons << endl;
    std::cout << "nPhotonsFromEta = " << nPhotonsFromEta << endl;
    std::cout << "nPionsneutral = " << nPionsneutral << endl;
    std::cout << "nPhotonsFromPi0␣=␣" << nPhotonsFromPi0 << endl;
    std::cout << "nPionsplus␣=␣" << nPionsplus << endl;
    std::cout << "nThermalPhotons␣=␣" << nThermalPhotons <<endl;

// Rescale histograms
  for(Int_t i=0; i<N_HISTOGRAMS_; i++)
    H1D[i]->Scale(1.0 / (Events * 2*1.0 * 2.0*TMath::Pi() * dX ));

  TH1D* CorrC = (TH1D*)CorrA->Clone("CorrC");
  CorrC->Divide(CorrA,CorrB,1,1,"B");

  TH1D* OpeningAngleRatio = (TH1D*)OpeningAngleSame->Clone("
     OpeningAngleRatio");
  OpeningAngleRatio->Divide(OpeningAngleSame,OpeningAngleMixed
     ,1,1,"B");
  Double_t entA = CorrA->GetEntries();
  Double_t entB = CorrB->GetEntries();
  CorrC->Scale(entB/entA);
//Fit
  TF1 *fitcor = new TF1("fitcor","1+[0]*exp(-x*x*[1]*[1])",0,0.2);
  fitcor->SetParameter(0,0.1);
  fitcor->SetParameter(1,20);



//Plot histograms
  Int_t tMinBin, tMaxBin;
  gStyle->SetOptStat("");
// Canvas
  TCanvas* Canvas         = new TCanvas("canvas", H1D[0]->GetYaxis()
     ->GetTitle(), 800, 600);
  gPad->SetFillColor(0);
  gPad->SetFillStyle(4000);
  gPad->SetMargin(0.14, 0.02, 0.14, 0.08);
  gPad->SetLogy();
// Histogram legend
  TLegend* LegendPart    = new TLegend(0.17, 0.18, 0.25, 0.40, "",
             "brNDC");
  LegendPart->SetFillColor(0);
  LegendPart->SetFillStyle(4000);
  LegendPart->SetTextSize(0.05);
  for(Int_t i=0; i<N_HISTOGRAMS_; i++)
    LegendPart->AddEntry(H1D[i], H1D[i]->GetTitle());
// Model legend
  TLegend* LegendModel   = new TLegend(0.70, 0.40, 0.98, 0.92, "
     model", "brNDC");
  LegendModel->SetFillColor(0);
  LegendModel->SetFillStyle(4000);
  LegendModel->SetTextSize(0.035);
  model2legend(aEventDir, aEventFiles, LegendModel);
// Histograms
  H1D[0]->GetXaxis()->SetTitleSize(0.06);
  H1D[0]->GetXaxis()->CenterTitle(kTRUE);
  H1D[0]->GetXaxis()->SetLabelSize(0.05);
  H1D[0]->GetYaxis()->SetTitleSize(0.06);
  H1D[0]->GetYaxis()->CenterTitle(kTRUE);
  H1D[0]->GetYaxis()->SetLabelSize(0.05);

  tMaxBin = H1D[0]->GetMaximumBin();     H1D[0]->SetMaximum((H1D
     [0]->GetBinContent(tMaxBin) + H1D[0]->GetBinError(tMaxBin)) *
     2.0);

  H1D[0]->SetMarkerColor(2);     H1D[0]->SetMarkerStyle(20);
  H1D[1]->SetMarkerColor(4);     H1D[1]->SetMarkerStyle(21);
  H1D[2]->SetMarkerColor(1);     H1D[2]->SetMarkerStyle(22);
  H1D[3]->SetMarkerColor(3);     H1D[3]->SetMarkerStyle(33);
  H1D[4]->SetMarkerColor(6);     H1D[4]->SetMarkerStyle(34);
  H1D[5]->SetMarkerColor(5);     H1D[5]->SetMarkerStyle(29);
// Plot
  H1D[0]->SetTitle("p_{T}␣distribution");
  H1D[0]->Draw();
  H1D[1]->Draw("SAME");
  H1D[2]->Draw("SAME");
  H1D[3]->Draw("SAME");
  H1D[4]->Draw("SAME");
  H1D[5]->Draw("SAME");
  LegendPart ->Draw();
  LegendModel->Draw();
// Save to files
  Canvas->SaveAs(aEventDir + _FIGURE_NAME_ + ".eps");




  // Canvas
  TCanvas* Canvas_2      = new TCanvas("InvMass", "InvMass", 800,
     600);
  gPad->SetFillColor(0);
  gPad->SetFillStyle(4000);
  gPad->SetMargin(0.14, 0.02, 0.14, 0.08);
// Histograms
// Plot


  // Canvas

  TCanvas* Canvas_4      = new TCanvas("OpeningAngle", "
     OpeningAngle", 800, 600);
  gPad->SetFillColor(0);
  gPad->SetFillStyle(4000);
  gPad->SetMargin(0.14, 0.02, 0.14, 0.08);

  OpeningAngleSame->Draw("SAME");

  TCanvas* Canvas_5      = new TCanvas("OpeningAngleRatio", "
     OpeningAngleRatio", 800, 600);
  gPad->SetFillColor(0);
  gPad->SetFillStyle(4000);
  gPad->SetMargin(0.14, 0.02, 0.14, 0.08);
  OpeningAngleRatio->Draw();


// Canvas
  TCanvas* Canvas_6      = new TCanvas("CorrA", "CorrA", 800, 600);
  CorrA->SetMarkerStyle(kOpenCircle);
// Histograms
  CorrA->GetXaxis()->SetTitleSize(0.06);
  CorrA->GetXaxis()->CenterTitle(kTRUE);
  CorrA->GetXaxis()->SetLabelSize(0.05);
  CorrA->GetYaxis()->SetTitleSize(0.06);
  CorrA->GetYaxis()->CenterTitle(kTRUE);
  CorrA->GetYaxis()->SetLabelSize(0.05);
// Plot
  CorrA->SetTitle("InvMomentum distribution same events");
  CorrA->Draw();

// Canvas
  TCanvas* Canvas_7      = new TCanvas("CorrB", "CorrB", 800, 600);
  CorrB->SetMarkerStyle(kOpenCircle);
// Histograms
  CorrB->GetXaxis()->SetTitleSize(0.06);
  CorrB->GetXaxis()->CenterTitle(kTRUE);
  CorrB->GetXaxis()->SetLabelSize(0.05);
  CorrB->GetYaxis()->SetTitleSize(0.06);
  CorrB->GetYaxis()->CenterTitle(kTRUE);
  CorrB->GetYaxis()->SetLabelSize(0.05);
// Plot
  CorrB->SetTitle("InvMomentum distribution different events");
  CorrB->Draw();

// Canvas
  TCanvas* Canvas_8       = new TCanvas("CorrC", "CorrC", 800, 600);
  CorrC->SetMarkerStyle(kFullCircle);



// Histograms
  CorrC->GetXaxis()->SetTitleSize(0.06);
  CorrC->GetXaxis()->CenterTitle(kTRUE);
  CorrC->GetXaxis()->SetLabelSize(0.05);
  CorrC->GetYaxis()->SetTitleSize(0.06);
  CorrC->GetYaxis()->CenterTitle(kTRUE);
  CorrC->GetYaxis()->SetLabelSize(0.05);
  CorrC->GetXaxis()->SetTitle("Qinv (GeV/c)");
  CorrC->GetYaxis()->SetTitle("C(Q)");
// Plot
  CorrC->SetTitle("Correlation between #pi^{+}");
  CorrC->Fit("fitcor");
  CorrC->Draw();
  TFile *outputFile = new TFile("ThermInatorEventMixing.root","
     RECREATE");
  CorrA->Write("CorrA");
  CorrB->Write("CorrB");
  CorrC->Write("CorrC");
  OpeningAngleSame->Write("OpeningAngleSame");
  OpeningAngleMixed->Write("OpeningAngleMixed");
  OpeningAngleRatio->Write("OpeningAngleRatio");
  outputFile->Close();




}
```