Computer exercise about elliptic flow

P. Christiansen (Lunds University)

November 30, 2014

Abstract

The goal of this exercise is to develop a toy simulation that can test various ways of measuring the elliptic flow.

1 The quark gluon plasma and elliptic flow

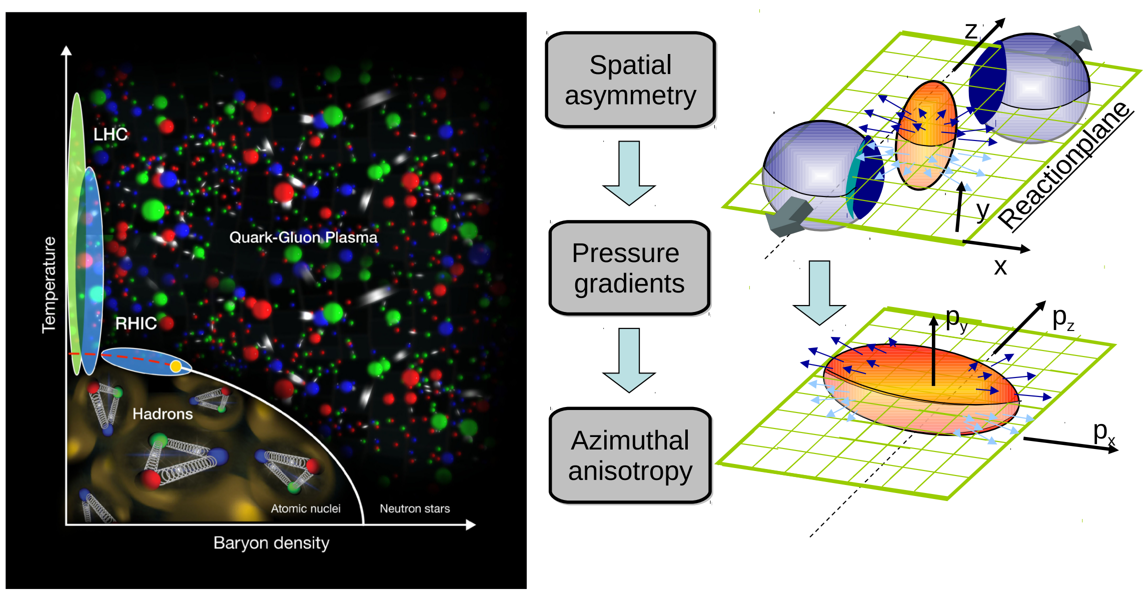

Figure 1: Left: QCD phase diagram [1]. Right: illustration of how elliptic flow arises in non-central heavy ion collisions.

Elliptic flow is a special phenomenon that occurs in collisions of heavy ions, e.g., at LHC. In central high energy heavy ion collisions a phase of deconfined quarks and gluons called the Quark Gluon Plasma (QGP) is produced, see Fig. 1. After production the QGP expands and cools off before it finally hadronizes. It is by studying the produced hadrons that we attempt to understand the properties of the QGP.

Surprisingly it has been shown that the expansion of the QGP medium is the same as expected for an almost ideal fluid. Because in most collisions the initial geometry is not symmetric, but is rather elliptic, see

Fig. 1, the expansion is asymmetric and gives rise to so-called elliptic flow: the distribution of particles in the final state is asymmetric in the azimuthal plane. In general this effect also depends on the transverse momentum of the particles and peaks at $p_T \approx 3$ GeV/c. It also has a mass dependence such that heavier particles are more affected by the flow. This can be understood if one considers a particle produced at rest ($p_T = 0$) and then boosts it with the fluid flow velocity β and looks at the mass dependence of the final p_T.

2 How to measure elliptic flow

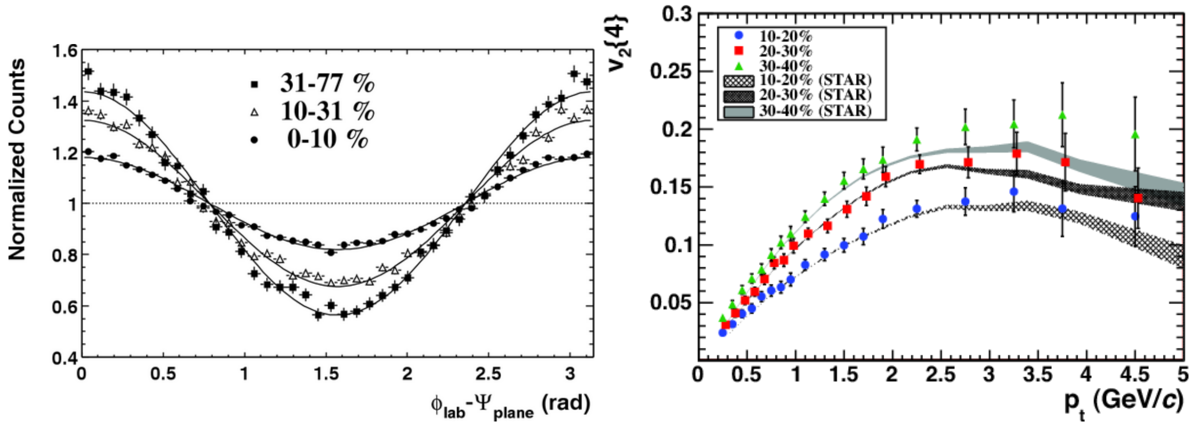

Figure 2: Left: STAR data used to estimate v_2 using the event plane method ($2 < p_{\mathrm{T}} < 6\,\mathrm{GeV}/c$). The elliptic flow v_2 is ≈0.1 for the most central collisions (0-10%) and ≈0.2 for the most peripheral collisions (31-77%) [2]. Right: Results from ALICE at LHC showing v_2 as a function of p_{T} extracted using 4-particle correlations [3]. The grey bands show comparable results from STAR at a beam energy that is more than 10 times smaller indicating that indeed we have a Quark Gluon Plasma with similar properties at both energies.

There are several ways to measure the elliptic flow. In this exercise we will test some of the different ways.

2.1 Event Plane method

In the event plane method we estimate the reaction plane in each collision and then align the particles relative to this plane. In this way we can average over many events.

If the particles in an event is distributed according to:

$$f(\varphi) \propto 1 + 2v_2 \cos[2(\varphi - \Psi_2)], \tag{1}$$

then the event plane we want to determine is Ψ_2.

It can be shown (good optional exercise) that Ψ_2 can be estimated as:

$$\Psi_2 = \frac{1}{2} \tan^{-1}\left(\frac{\langle \sin(2\varphi)\rangle}{\langle \cos(2\varphi)\rangle}\right), \tag{2}$$

where it is smart to use the ROOT function `TMath::ATan2(y, x)`.

One then histograms all tracks relative to this plane in each event. After measuring over many events one fits the final histogram using a constant (normalization) times Eq. 15, see Fig. 2 left.

The elliptic flow measured this way is often denoted $v_2\{\mathrm{EP}\}$. It is in this authors opinion the most easy method to understand and one can easily visually check that one did not make a grave mistake, but is is not as precise as the other methods.

2.2 2-Particle correlations

The Event Plane method is criticized for the need to first determine the event plane and then measure v_2 since the statistical precision with which we can determine the event plane event-by-event will affect the result (even one typically corrects for this using a resolution function). One can avoid this by studying 2-particle correlations [1]:

$$\begin{aligned}
\langle \cos[2(\varphi_1 - \varphi_2)] \rangle &= Re\langle e^{i2(\varphi_1-\varphi_2)} \rangle \\
&= Re\langle e^{i2(\varphi_1-\Psi_2-\varphi_2+\Psi_2)} \rangle \\
&\approx Re\left[\langle e^{i2(\varphi_1-\Psi_2)} \rangle \langle e^{i2(\varphi_2-\Psi_2)} \rangle \right] \\
&= \langle \cos[2(\varphi_1 - \Psi_2)] \rangle \langle \cos[2(\varphi_2 - \Psi_2)] \rangle \\
&= v_2^2,
\end{aligned} \tag{3}$$

where the average is over all pairs, and the assumption in line 3 is that there are no direct correlations between particle 1 and 2, but only indirect correlations through the common event plane Ψ_2.

In this way we determine the 2-particle correlation factor $\langle 2 \rangle = \langle \cos[2(\varphi_1 - \varphi_2)]$ and we write $v_2\{2\} = \sqrt{\langle 2 \rangle}$.

2.3 2-Particle correlations using the Q-vector

To determine the 2-particle correlations we need a nested double loop, e.g, if we have 1000 tracks we loop for track 1 over the remaining 999 particles, for track 2 over the remaining 998 particles, and so on. Especially when one goes to 4-particle correlations and higher this becomes impossible due to the time it takes.

It turns out that one can determine $Q_n = \sum_{\text{tracks}} e^{in\varphi}$ with which it is easy to show that:

$$\langle 2 \rangle = \frac{|Q_2|^2 - M}{M(M-1)} \tag{4}$$

In this way one just have to loop one time over all tracks to calculate $\langle 2 \rangle$.

3 Different Ways to Generate Random Numbers According to a Distribution

To be able to test the different methods we want to be able to generate random numbers that are distributed like Eq. 15. Here we shall not discuss how one can generate a so-called flat distribution of random numbers between 0 and 1, but show how one can use these to generate any distribution.

In the following example we always wants to generate numbers distributed according to $\sin x$ and we will show plots from the macro `generate_sinx.C` that is also made available to you.

3.1 The box method

This is a simple method that works as long as one generates random numbers in a restricted range, but is not always very efficient. We generate two random numbers x and y. We scale x so that it gives a random point in the restricted range we want to generate random numbers in. Now we scale y so it matches the range from 0 to the maximum value of the function $f(x)$ we want t generate. Now we accept x if $y < f(x)$ and reject x otherwise.

Figure 3 gives an example of this method.

[1]The biggest gain is when we go to higher order correlations, see Advanced Topic 2.

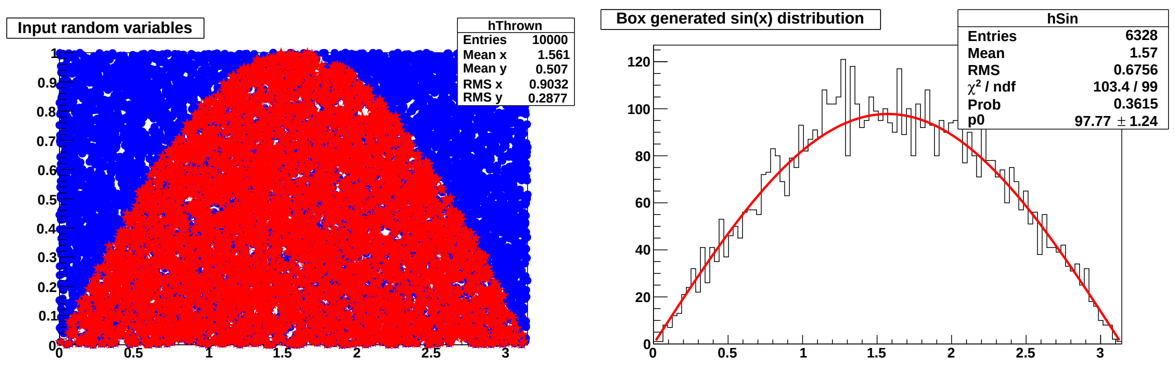

Figure 3: The box method. Left: 2-dimensional histograms showing all attempted (x, y) blue, and those accepted in red. Right: The distribution of accepted x values. The red function is a fit to the generated data.

3.2 The analytical

In some cases one can integrate and invert the probability density distribution. In this case, e.g.:

$$
\begin{aligned}
&P(x) = \frac{1}{2}\sin x \\
&\int_0^{\pi} P(x')dx' = 1 \\
&y = \int_0^{x} P('x)dx' = \frac{1}{2}[-\cos x']_0^x \\
&= \frac{1}{2}(1-\cos x), \qquad \text{so that} \\
&x = \cos^{-1}(1-2y)
\end{aligned}
\tag{5}
$$

This means that if we generate a random number y between 0 and 1 then we should just apply the transformation $x = \cos^{-1}(1-2y)$ to get random numbers distributed according to $\sin x$.

From this method one will get a similar distribution as Figure 3 right, but one has a 100% efficiency.

This method cannot always be used, because it is not always possible to invert a function. Sometimes it can be used even when x can take unbounded values, e.g., if we want to distribute according to an exponential distribution $\exp^{-x/k}$. There are also unbounded cases where one can combine this with the box method.

4 The histogram method

This method can be used to approximate any function in a restricted range and can also be used in the case where one only has a histogram, e.g., with real data and want to generate random numbers according to this.

We first choose a binning and then fill in each bin the cumulative function value in this bin. Then we normalize it to the total integral so that in the last bin we have the value 1.

Now it works in some sense exactly like a numerical approximation of the analytical method. We generate a random value z and we find the corresponding lowest bin where z is smaller than the histogram value. Now we use as random number x of this bin. Figure 4 gives an example of this method.

This is the least elegant method, but it is quite simple to implement and can easily be generalized to 2d and 3d distributions, but one will of course have binning effects even if one can smooth out this.

Here it is meant to illustrate how we understand the basic principles of ROOT's generic function as explained in the ROOT class description [2]

[2]See `http://root.cern.ch/root/html/TF1.html`.

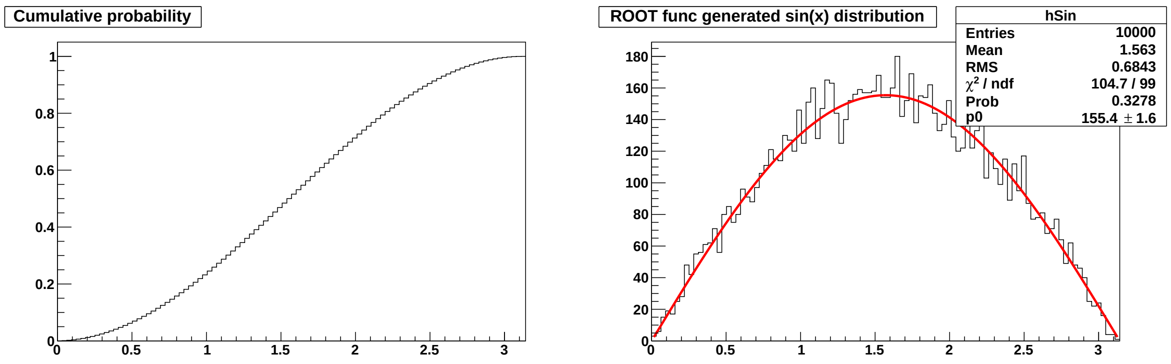

Figure 4: The histogram method. Left: Cumulative probability for 100 bins. Right: The distribution of generated x values. The red function is a fit to the generated data.

```
Double_t TF1::GetRandom(Double_t xmin, Double_t xmax)

Return a random number following this function shape in [xmin,xmax]

The distribution contained in the function fname (TF1) is integrated
over the channel contents.
It is normalized to 1.
For each bin the integral is approximated by a parabola.
The parabola coefficients are stored as non persistent data members
Getting one random number implies:
- Generating a random number between 0 and 1 (say r1)
- Look in which bin in the normalized integral r1 corresponds to
- Evaluate the parabolic curve in the selected bin to find
the corresponding X value.
The parabolic approximation is very good as soon as the number
of bins is greater than 50.

IMPORTANT NOTE
The integral of the function is computed at fNpx points. If the function
has sharp peaks, you should increase the number of points (SetNpx)
such that the peak is correctly tabulated at several points.
```

5 Compulsory Exercises

This is the list of exercises that each student has to do. In addition there are 3 advanced topics that the student can optionally solve.

5.1 Exercise 1

Familiarize yourself with the example `generate_sinx.C`. Run the 3 different methods (box, analytical, histogram), and understand the differences and advantages.

5.2 Exercise 2

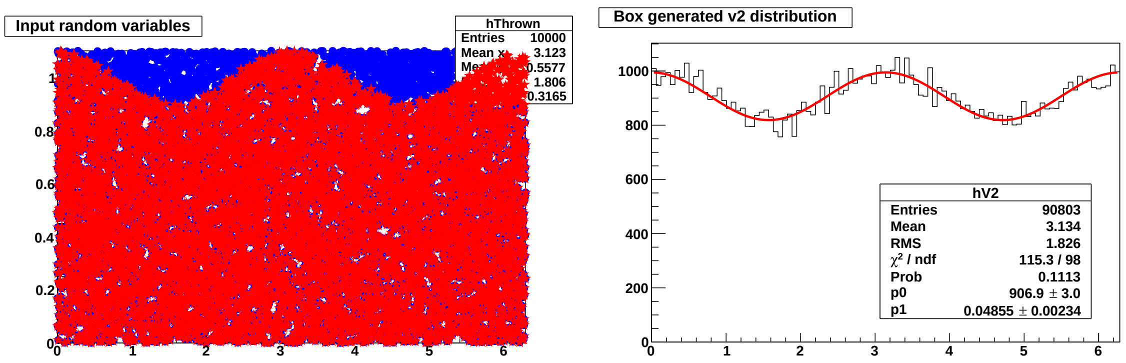

Figure 5: Generating φ according to $v_2 = 0.05$. Left: 2-dimensional histograms showing all attempted (φ, y) blue, and those accepted in red. Right: The distribution of accepted φ values. The red function is a fit to the generated data. Note that in the left we only used 10,000 events while in the right we used 100,000 events.

Extend the box method to be able to generate random numbers according to Eq. 15. In all cases we set $\Psi_2 = 0$. Figure 5 gives an example of how this will work.
Technical help:

- Copy `generate_sinx.C` to a new file: `generate_v2.C`
- Remove all methods except the box method works
- Rename all places sinx to v2.
- Change the code to work for v2. Be careful to also change ranges to match the new situation. The function should also take the new argument v2.

Make sure after each step that the code still works before going to the next step.

Note that in real data the total transverse momentum will have to be 0. This constrain we do not have here and so the data we generate does not necessary have to match even idealized collision data. For larger number of tracks this is likely not an issue.

5.3 Exercise 3

Extend this code to implement all 3 methods.

First we need to convert this to a generator that now should take 3 arguments: Nevents, Ntracks (in each event), and v2.

- Copy `generate_v2.C` to a new file: `calculate_v2.C`

- To speed up visualization only draw hAccepted and change the draw option to COLZ (you can even remove hThrown).
- Add the Ntrack option and make a temporary storage for the phi angle, e.g.:

```
  Double_t phi[nTracks];

  Int_t nt = 0;
  while (nt < nTracks) {

    ....
    if(....) {
      // for each accepted track
      phi[nt] = x;
      nt++;
    }
  }
```

  To test this we fill hV2 in another loop

```
    for(Int_t i = 0; i < nTracks; i++) {

      hV2->Fill(phi[nt]);
    }
```

Make sure after each step that the code still works before going to the next step.

Now you have the generator part and should implement the 3 different methods instead of filling the hV2 histogram. It is maybe easier to create one method for the event plane and one for the 2-particle correlation methods. For the event plane, , you need to loop two times: firstly, to estimate the event-plane, Ψ_2 and secondly to fill the histogram with $\varphi - \Psi_2$. Make sure that $\varphi - \Psi_2$ is in the actual range of your histogram (add or subtract 2π if it is outside this range).

To help with the 2-particle correlation function part I give here an example of how the loop can be done:

```
// Q vector
Double_t sum_cos2 = 0;
Double_t sum_sin2 = 0;
// 2-particle
Double_t sum_cos2_diff = 0;
for(Int_t i = 0; i < nTracks; i++) {

  sum_cos2 += TMath::Cos(2*phi[i]);
  sum_sin2 += TMath::Sin(2*phi[i]);

  for(Int_t j = i+1; j < nTracks; j++) {

    sum_cos2_diff += 2*TMath::Cos(2*(phi[i]-phi[j]));
  }
}
```

And then one can compare the methods event-by-event and see they give exactly the same, e.g., by printing out:

```
v2 (event) = 0.0393162
```

```
v2 (average of 99 events) = 0.0494526
Q: v2 (event) = 0.0393162
Q: v2 (average of 99 events) = 0.0494526
v2 (event) = 0.0383989
v2 (average of 100 events) = 0.0493544
Q: v2 (event) = 0.0383989
Q: v2 (average of 100 events) = 0.0493544
```

Note that one can only calculate the $v_2\{2\}$ for an event when $\langle 2 \rangle > 0$.

5.4 Exercise 4

Which method does the best job of calculating v_2?

How does this depend on Ntracks and Nevents?

Bonus question: in the event plane method one has trivial correlations between each article and the event plane. Since the event plane in each event just tries to maximize the v_2 one in general overestimates v_2. One can calculate the event plane for each track where one ignores the track itself. If one does this one instead always finds a too small v_2. Can you understand why? (Hint: study the event plane resolution in your code and think about what it does to v_2).

6 Advanced Topic 1: Triangular flow

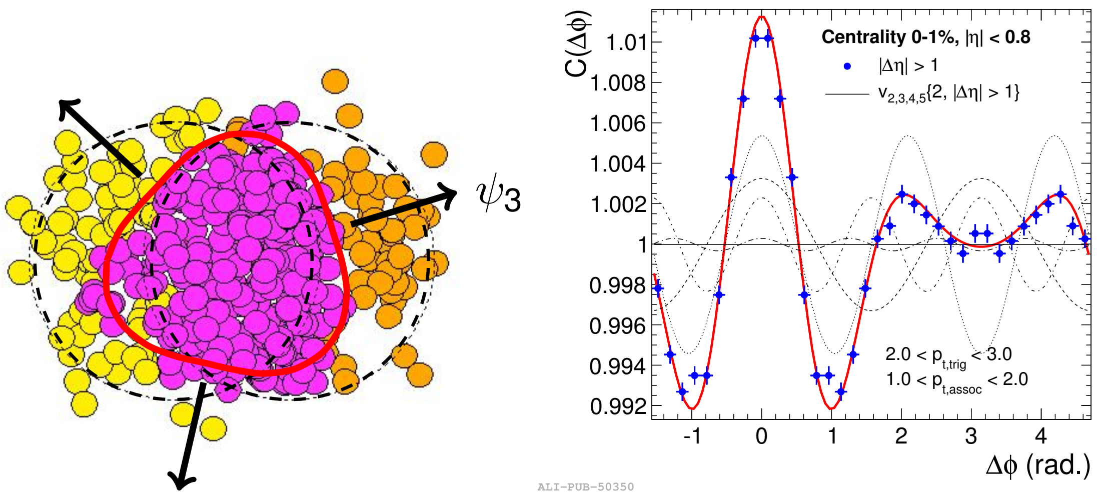

Figure 6: Triangular flow. Left: fluctuations can give rise to a triangularity in the created medium. Right: by going to the 1% most central collisions one can suppress v_2 and directly observe the triangular correlations using 2-particle correlation techniques as maxima at 0, $2/3\pi$ and $4/3\pi$ radians (0, 120, and 240 degrees). Notice that the red curve is *calculated* from the 2-particle flow coefficients v_2, v_3, v_4, v_5 [5].

In 2005 the PHOBOS experiment at RHIC have shown that to understand the flow fluctuations observed in Au-Au and Cu-Cu collisions one needs to take into account fluctuations in the positions of nucleon-nucleon collisions. This means that the impact parameter plane is not always the best symmetry plane to use. In 2010 two experimental physicists, Alver and Roland [4], showed that the same fluctuations could give rise to triangular flow, v_3, see Fig. 6 Left:

$$f(\varphi) \propto 1 + 2v_3 \cos[3(\varphi - \Psi_3)], \tag{6}$$

The ALICE experiment at LHC confirmed this prediction by going to the most central collisions where the geometric elliptic flow is suppressed, see Fig. 6.

6.1 Step 1: extend your code to handle v_3 instead of v_2

In this part assume that $\Psi_3 = 0$.

This should be straight forward, but recall that v_3 has 3 peaks and in general one therefore needs $\cos 3\varphi$ instead of $\cos 2\varphi$ and so on.

6.2 Step 2: extend your code to handle v_2 and v_3 at the same time

In detailed simulations and confirmed by measurements the symmetry angles Ψ_2 and Ψ_3 are not correlated. Ψ_2 is geometrical and close to the impact parameter angle, while Ψ_3 is related to fluctuations. This might be necessary to implement to get the methods to work.

7 Advanced topic 2: 4-particle correlations

It is known that in addition to correlations through the event plane Ψ_2 there are also correlations from decays, e.g.: $\phi \to K^+ + K^-$. These correlations are called non-flow and can be removed by studying higher order correlation functions which is the goal of this advanced exercise.

7.1 Step 1: implement non-flow

An easy way to implement non-flow is for each particle (or some fraction of particles) to add one extra particle with exactly the same φ angle [3].

Do this in your code and check that you now estimate a wrong $v_2\{2\}$.

7.2 Step 2: implement the 4-particle correlation function

In the same way as we defined the 2-particle correlation factor we can define the 4-particle correlation factor:

$$\langle 4 \rangle = \langle \cos[2(\varphi_1 + \varphi_2 - \varphi_3 - \varphi_4)] \rangle. \tag{7}$$

It turns out that by using the 4-particle cumulant one can subtract the 2-particle correlations to v_2 so one only are left with genuine 4-particle correlations:

$$v_2\{4\} = \sqrt[4]{-\langle 4 \rangle + 2\langle 2 \rangle^2}. \tag{8}$$

In this way we expect to be insensitive to the non-flow we just added before.

The problem with $\langle 4 \rangle$ is that if you have e.g. 1000 tracks in each event then it will take forever to calculate the correlation as a nested loop (over i, j, k, l). The trick is to use the Q-vector. It can be shown that:

$$\begin{aligned} \langle 4 \rangle = & \frac{|Q_2|^4 + |Q_4|^2 - 2 \cdot \mathrm{Re}[Q_4 Q_2^* Q_2^*]}{M(M-1)(M-2)(M-3)} \\ & - 2\frac{2(M-2) \cdot |Q_2|^2 - M(M-3)}{M(M-1)(M-2)(M-3)}. \end{aligned} \tag{9}$$

This analytic result and many more can be found here [6].

Implement this estimate and show that now you obtain the correct v_2 independent of non-flow. While $v_2\{2\}$ converges fast and works well even for a few particles in general $v_2\{4\}$ converges much slower and one therefore needs to run with a lot of tracks in each event [4].

7.3 Final remarks about flow fluctuations

Due to fluctuations the v_2 varies event-by-event. As the 2- and 4-particle correlation functions measures v_2 to some power they are biased by these fluctuations. The easiest case to understand is the 2-particle correlation function. If $\sigma_{v2}^2 = \langle v_2^2 \rangle - \langle v_2 \rangle^2$, then it is clear that one in general with $v_2\{2\}$ measures (neglecting non-flow):

$$v_2\{2\}^2 = \langle v_2 \rangle^2 + \sigma_{v2}^2, \tag{10}$$

so that $v_2\{2\} \geq \langle v_2 \rangle$.

For 4-particle correlations one finds that when $\sigma_{v2} \ll \langle v_2 \rangle$ then:

$$v_2\{4\}^2 = \langle v_2 \rangle^2 - \sigma_{v2}^2, \tag{11}$$

so that $v_2\{4\} \leq \langle v_2 \rangle$.

This means that we can estimate the fluctuations as:

$$2\sigma_{v2}^2 = v_2\{2\}^2 - v_2\{4\}^2 \tag{12}$$

Use your code to study this relation (remove non-flow).

[3] The advantage of this method is not only that it is easy, but one can in fact calculate the exact effect on the 2-particle correlation analytic. If you do this, then try to understand the nTrack dependence (Hint: consider the number of pairs).

[4] The number of 4-particle associations grows approximately as $(\text{Ntracks})^4$ so when calculating higher order correlations one therefore also has to be careful that this can impose a significant bias towards the subsample of the events with the most tracks

8 Advanced topic 3: the statistical uncertainty

It can sometimes be difficult to calculate the statistical uncertainty on a quantity that is measured in a complex way like $v_2\{2\}$. Here we will illustrate how it can be done by subdividing the total sample into smaller samples and then from the spread of those assign an uncertainty for the average.

8.1 Step 1: extend your code to make 1000s of analyzes

In this part you want to know the answer for a certain configuration of Ntracks and Nevents. To find this you can simply just change your code so it will make $N_{\text{experiments}}$. Then you can make a histogram with the results of e.g. $v_2\{2\}$ for each experiment and then in the end you will have a histogram which is approximately Gaussian and where the mean should be very close to the input v_2 and where the σ is the statistical uncertainty.

8.2 Step 2: implement an unbiased estimator for the mean

In this part we now want to divide the original Nevents sample into 10 smaller samples (Nevents/10) that we then analyze independently ($N_{\text{experiments}} = 10$). In this way we can obtain:

$$\sigma_{1/10} = \frac{1}{N_{\text{experiments}} - 1} \sum_{i=1}^{N_{\text{experiments}}} (v_{2,i} - \langle v_2 \rangle)^2, \tag{13}$$

and then we estimate the statistical uncertainty for the full sample to be:

$$\sigma = \frac{\sigma_{1/10}}{\sqrt{N_{\text{experiments}}}}. \tag{14}$$

The important thing to notice is that one should divide by $N_{\text{experiments}} - 1$ and not $N_{\text{experiments}}$ in Eq. 13. This estimator for $\sigma_{1/10}$ is called the unbiased estimator because it takes into account that we use 1 degree of freedom to estimate also $\langle v_2 \rangle$. If you would use the generator v_2 in Eq. 13 then you should in fact use $N_{\text{experiments}}$ and not $N_{\text{experiments}} - 1$.

Test the performance of this method, i.e., compare to the result found in step 1.

Note that since we simulate only few events here (vs millions in real experiments) then the method is not so good. This is because $v_2\{2\}$ depends non-linearly on $\langle 2 \rangle$. The fluctuations in $\langle 2 \rangle$ must be rather symmetric and so if they are large this gives an asymmetric influence on $v_2\{2\}$, e.g. if it fluctuates down to 0 then $v_2\{2\}$ decreases by 100% while if it fluctuates up a factor 2 $v_2\{2\}$ only increases by 40% [5]. Due to the 10 times smaller statistics one also easier gets negative values for $\langle 2 \rangle$ that one has to decide how to handle. For this reason it is recommended to have a large number of events and a large v_2, e.g., 1000 events with 100 tracks and $v_2 = 0.1$.

Bonus question: what is the motivation to subdivide into 10 subsamples and not 3 or 50?

8.3 Step 3: implement fitter

In the event plane method one actually obtains also a statistical uncertainty from the ROOT fitter.

In this step we want to implement a simple fit program and use that to fit the event plane data and estimate the statistical uncertainty.

If we normalize the event plane histogram properly we only need to fit the v_2:

$$f(\varphi) = 1 + 2v_2 \cos[2\varphi]. \tag{15}$$

The idea is to make a 1 dimensional histogram where on the x-axis we vary v_2 and on the y-axis we calculate χ^2:

$$\chi^2(v_2) = \sum_{\text{bins}} \frac{(f(\varphi) - (\text{hist value in bin}))^2}{(\text{hist uncertainty in bin})^2}. \tag{16}$$

[5]One can actually do the σ analysis for $\langle 2 \rangle$ instead to reduce these problems.

ROOT will keep track of the statistical uncertainty for a histogram when you call the method `Sumw2()` just after creating it. The statistical uncertainty in each of the event plane histogram bins is in this case just: $\sqrt{N_{\text{entries}}}$/Normalization.

If you select a reasonable range of parameters for the guessed v_2 then you should now see a minimum in your histogram. This is the best estimate of v_2 given the current binning. Write a loop into your code that finds this minimum. Now it turns out that if the fit has good quality then the so-called reduced χ^2 should be close to 1. The reduced χ^2 is defines as χ^2/N_{dof}, where the N_{dof} is the number of degrees of freedoms given as the number of bins minus the number of fit parameters, so in this case it is number of bins minus 1.
The reduced χ^2 quantifies the actual deviation between the fitted curve and the data points and gives a measure of how similar they are compared to the actual statistical uncertainty of your measurements.
If the reduced χ^2 is much less than 1 then it means that your statistical uncertainty is too large or you have too many parameters in your fit. If the reduced χ^2 is much larger than 1 it means that your statistical uncertainty is too small and likely there are systematic uncertainties that you need to take into account.

Test if the reduced χ^2 is close to 1 in your case when you vary the number of histogram bins, the number of events, and/or the number of tracks.

The statistical uncertainty on the fitted v_2 is approximately the absolute distance in v_2 you have to step away from the minimum for the χ^2 to be larger by 1 (this step is in χ^2 and not in the reduced χ^2). Estimate this in your program and compare to what ROOT gets for the fit.

In more advanced fitters the fitting routine estimates a local derivative in χ^2 space around the current value and then uses that to find where to go next, in this way stepping towards the minimum. For situations where there are more than one minimum more advanced methods are needed.
In general things gets even more complicated (unstable and time consuming) the more parameters one has in the fit.

References

[1] B. V. Jacak and B. Muller, Science **337**, 310 (2012).

[2] C. Adler *et al.* [STAR Collaboration], Phys. Rev. Lett. **90**, 032301 (2003).

[3] K. Aamodt *et al.* [ALICE Collaboration], Phys. Rev. Lett. **105**, 252302 (2010).

[4] B. Alver and G. Roland, Phys. Rev. C **81**, 054905 (2010).

[5] K. Aamodt *et al.* [ALICE Collaboration], Phys. Rev. Lett. **107**, 032301 (2011).

[6] A. Bilandzic, R. Snellings and S. Voloshin, Phys. Rev. C **83**, 044913 (2011).