Flow analysis 'on the fly'

January 7, 2014

1 Introduction

Flow analysis 'on the fly' is a feature in the *ALICE flow package*[1] which can serve both as a demo for the potential users of the package and as an important debugging tool for the core flow code developers. Underlying idea is very simple: To simulate events of interest for flow analysis (in what follows we shall refer to such events as *flow events*) in the computers memory and than pass them 'on the fly' to the implemented methods for flow analysis. Benefits of this approach include:

1. No need to store data on disk (storing only the output files with the final results and not the simulated events themselves);
2. Enormous gain in statistics;
3. Speed (no need to open the files from disk to read the events);
4. Random generators initialized with the same and random seed (if the same seed is used simulations are reproducible) .

In Section 2 we indicate how the user can immediately in a few simple steps start flow analysis 'on the fly' with the default settings both within AliRoot and Root. In Section 3 we explain how the user can modify the default settings and create 'on the fly' different flow events by following the guidance of his own taste.

[1] `http://alisoft.cern.ch/viewvc/trunk/PWG/FLOW/?root=AliRoot` .

2 Kickstart

We divide the potential users of ALICE flow package into two groups, namely the users which are using AliRoot (default) and the users which are using only Root.

2.1 AliRoot users

To run flow analysis 'on the fly' with the default settings within AliRoot and to see the final results obtained from various implemented methods for flow analysis, the user should execute the following steps:

Step 1: Turn off the lights ...

Step 2: ... take a deep breath ...

Step 3: ... start to copy macros `runFlowAnalysisOnTheFly.C` and `compareFlowResults.C` from `AliRoot/PWG/FLOW/macros` to your favorite directory slowly.

Step 4: Once you have copied those macros in your favorite directory simply go to that directory and type

```
aliroot runFlowAnalysisOnTheFly.C
```

Step 5: If you have a healthy AliRoot version the flow analysis 'on the fly' will start. Once it is finished in your directory you should have the following files:

```
runFlowAnalysisOnTheFly.C
compareFlowResults.C
outputLYZ1PRODanalysis.root
outputQCanalysis.root
outputFQDanalysis.root
outputLYZ1SUManalysis.root
outputSPanalysis.root
outputGFCanalysis.root
outputMCEPanalysis.root
```

Each implemented method for flow analysis produced its own output file holding various output histograms. The final flow results are stored in the common histogram structure implemented in the class `AliFlowCommonHistResults`.

Step 6: To access and compare those final flow results automatically there is a dedicated macro available, so execute

```
> aliroot
root [0] .x compareFlowResults.C("")
```

Step 7: If you want to rerun and get larger statistics modify

```
Int_t nEvts=440
```

in the macro `runFlowAnalysisOnTheFly.C`.

Step 8: Have fun!

In the next section we outline the steps for the Root users.

2.2 Root users

To be written at Nikhef...

3 Making your own flow events

This section is common both for AliRoot and Roor users. In this section we outline the procedure the user should follow in order to simulate 'on the fly' the events with his own settings by making use of the available setters. Those setters are implemented in the class `AliFlowEventSimpleMakerOnTheFly` and user shall use them in the macro `runFlowAnalysisOnTheFly.C`.

3.1 p_T spectra

Transverse momentum of particles is sampled from the predefined Boltzmann distribution

$$\frac{dN}{dp_T} = Mp_T \exp\left(-\frac{\sqrt{m^2+p_T^2}}{T}\right), \tag{1}$$

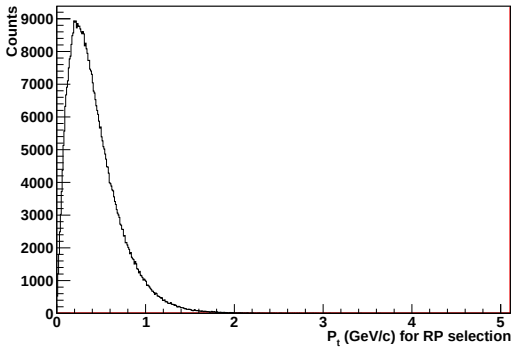

Figure 1: $T = 0.2$ GeV/c.

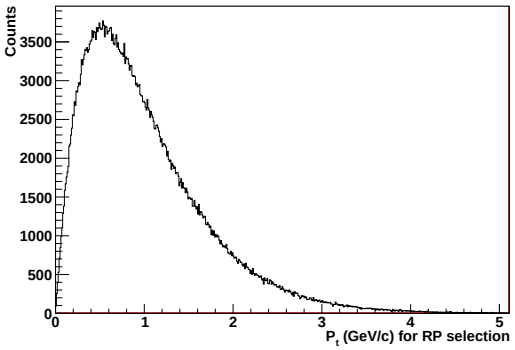

Figure 2: $T = 0.5$ GeV/c.

where M is the multiplicity of the event, T is "temperature" and m is the mass of the particle. By increasing the parameter T one is increasing the number of high p_T particles and this parameter is the same for all events. On the other hand, multiplicity M will in general vary from event to event. In the macro `runFlowAnalysisOnTheFly.C` one can modify distribution (1) by using setter for "temperature" T and various setters for multiplicity M.

Example: *If one wants to increase/decrease the number of high p_T particles, one should modify the line*

```
Double_t dTemperatureOfRP = 0.44;
```

Examples of p_T spectra for two different values of T are shown in Figures 1 and 2.

What is shown in Figures 1 and 2 is only one example of the so called *common control histograms*. They are the histograms organized in the same structure and implemented in the class `AliFlowCommonHist`. In output file of each method one can access those histograms with `TBrowser`.

When it comes to multiplicity M, one has a choice to sample it event-by-event from two different distributions before plugging its value into Eq. (1) which than will be used to sample transverse momenta of M particles in that event.

Example: *If one wants to sample multiplicity event-by-event from Gaussian*

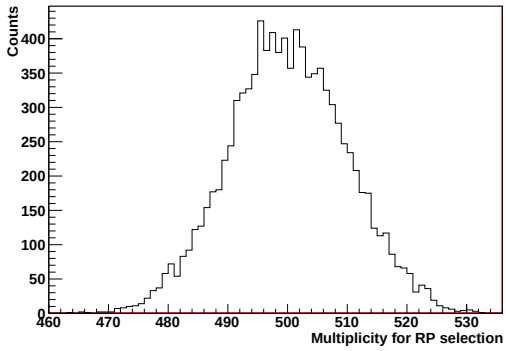

Figure 3: Gaussian multiplicity distribution.

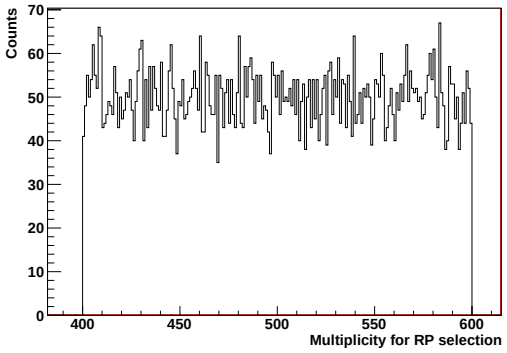

Figure 4: Uniform multiplicity distribution.

distribution with mean 500 and spread 10, one should have the following relevant settings

```
Bool_t bMultDistrOfRPsIsGauss = kTRUE;
Int_t iMultiplicityOfRP = 500;
Double_t dMultiplicitySpreadOfRP = 10;
```

Example plot for multiplicity distribution with these settings is shown in Figure 3.

Another way to sample multiplicity event-by-event is by using uniform distribution.

Example: *If one wants to sample multiplicity event-by-event from uniform distribution in the interval [400,600], one must have the following relevant settings*

```
Bool_t bMultDistrOfRPsIsGauss = kFALSE;
Int_t iMinMultOfRP = 400;
Int_t iMaxMultOfRP = 600;
```

Example plot for multiplicity distribution with these settings is shown in Figure 4.

One can also fix multiplicity to be the same for each event.

Example: *If one wants to have the same fixed multiplicity of 500 for each event one can use the following settings:*

```
Bool_t bMultDistrOfRPsIsGauss = kTRUE;
Int_t iMultiplicityOfRP = 500;
Double_t dMultiplicitySpreadOfRP = 0;
```

These are all manipulations available at the moment with p_T spectra given in Eq. (1).

3.2 Azimuthal distribution

If the anisotropic flow exists, it will manifest itself in the anisotropic azimuthal distribution of outgoing particles measured with respect to the reaction plane:

$$E\frac{d^3N}{d^3\vec{p}} = \frac{1}{2\pi}\frac{d^2N}{p_T dp_T d\eta}\left(1+\sum_{n=1}^{\infty} 2v_n(p_T,\eta)\cos\left(n\left(\phi-\Psi_{\mathrm{RP}}\right)\right)\right). \qquad (2)$$

Flow harmonics v_n quantify anisotropic flow and are in general function of transverse momentum p_T and pseudorapidity η. Orientation of reaction plane Ψ_{RP} fluctuates randomly event-by-event and cannot be measured directly. In the implementation 'on the fly' reaction plane is sampled uniformly event-by-event from the interval $[0^{\mathrm{o}}, 360^{\mathrm{o}}]$. When it comes to flow harmonics, there are two modes which we outline next.

3.2.1 Constant flow harmonics

In this mode all flow harmonics are treated as a constant, event-wise quantities, meaning that for a particular event azimuthal angles of all particles will be sampled from the same azimuthal distribution in which flow harmonics appear just as fixed parameters. The implemented most general azimuthal distribution for this mode reads

$$\frac{dN}{d\phi} = 1+2v_1\cos(\phi-\Psi_{\mathrm{RP}})+2v_2\cos(2(\phi-\Psi_{\mathrm{RP}}))+2v_4\cos(4(\phi-\Psi_{\mathrm{RP}}))\,. \quad (3)$$

In the macro `runFlowAnalysisOnTheFly.C` one can use the dedicated setters and have handle on the flow harmonics v_1, v_2 and v_4. The most important harmonic is v_2, the so called *elliptic flow*, so we start with it first.

Example: *If one wants to sample particle azimuthal angles from azimuthal distribution parameterized only with constant elliptic flow of 5%, namely*

$$\frac{dN}{d\phi} = 1 + 2 \cdot 0.05 \cdot \cos(2(\phi - \Psi_{\mathrm{RP}}))\,, \tag{4}$$

then one should use the following settings

```
Bool_t bConstantHarmonics = kTRUE;
Bool_t bV2DistrOfRPsIsGauss = kTRUE;
Double_t dV2RP = 0.05;
Double_t dV2SpreadRP = 0.0;
Double_t dV1RP = 0.0;
Double_t dV1SpreadRP = 0.0;
Double_t dV4RP = 0.0;
Double_t dV4SpreadRP = 0.0;
```

In this mode the flow coefficients are constant for all particles within particular event, but still the flow coefficients can fluctuate event-by-event.

Example: *If one wants to sample particle azimuthal angles from azimuthal distribution parameterized only with elliptic flow which fluctuates event-by-event according to Gaussian distribution with mean 5% and spread 1%, than one should use the following settings*

```
Bool_t bConstantHarmonics = kTRUE;
Bool_t bV2DistrOfRPsIsGauss = kTRUE;
Double_t dV2RP = 0.05;
Double_t dV2SpreadRP = 0.01;
Double_t dV1RP = 0.0;
Double_t dV1SpreadRP = 0.0;
Double_t dV4RP = 0.0;
Double_t dV4SpreadRP = 0.0;
```

On can also study uniform flow fluctuations.

Example: *If one wants to sample particle azimuthal angles from azimuthal distribution parameterized only with elliptic flow which fluctuates event-by-event according to uniform distribution in interval [4%,6%], than one should use the following settings*

```
Bool_t bConstantHarmonics = kTRUE;
Bool_t bV2DistrOfRPsIsGauss = kFALSE;
Double_t dMinV2RP = 0.04;
Double_t dMinV2RP = 0.06;
Double_t dV1RP = 0.0;
Double_t dV1SpreadRP = 0.0;
Double_t dV4RP = 0.0;
Double_t dV4SpreadRP = 0.0;
```

It is of course possible to simulate simultanously nonvanishing v_1, v_2 and v_4.

Example: *If one wants to sample particle azimuthal angles from azimuthal distribution parameterized by harmonics* $v_1 = 2\%$, $v_2 = 5\%$ *and* $v_4 = 1\%$, *namely*

$$\begin{aligned} \frac{dN}{d\phi} &= 1 + 2 \cdot 0.02 \cdot \cos(\phi - \Psi_{\mathrm{RP}}) + 2 \cdot 0.05 \cdot \cos(2(\phi - \Psi_{\mathrm{RP}})) \\ &+ 2 \cdot 0.01 \cdot \cos(4(\phi - \Psi_{\mathrm{RP}})) \end{aligned} \tag{5}$$

then one should use the following settings

```
Bool_t bConstantHarmonics = kTRUE;
Bool_t bV2DistrOfRPsIsGauss = kTRUE;
Double_t dV2RP = 0.05;
Double_t dV2SpreadRP = 0.0;
Double_t dV1RP = 0.02;
Double_t dV1SpreadRP = 0.0;
Double_t dV4RP = 0.01;
Double_t dV4SpreadRP = 0.0;
```

In the next section we outline the procedure for simulating flow events with p_T dependent flow harmonics.

3.2.2 p_T dependent flow harmonics

In this mode the functional dependence of flow harmonics on transverse momentum is treated as an event-wise quantity, while within the particular event the flow harmonics will change from particle to particle depending on

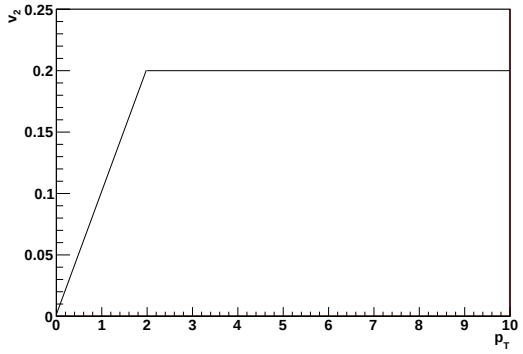

Figure 5: p_T dependent elliptic flow.

its transverse momentum. The implemented azimuthal distribution for this case reads

$$\frac{dN}{d\phi} = 1 + 2v_2(p_T)\cos(2(\phi - \Psi_{\mathrm{RP}}))\,, \tag{6}$$

and the functional dependence $v_2(p_T)$ is implemented as follows:

$$v_2(p_T) = \begin{cases} v_{\max}(p_T/p_{\mathrm{cutoff}}) & p_T < p_{\mathrm{cutoff}}\,, \\ v_{\max} & p_T \geq p_{\mathrm{cutoff}}\,. \end{cases} \tag{7}$$

In the macro `runFlowAnalysisOnTheFly.C` one can have the handle on the parameters $v_{\max}$ and p_{cutoff}.

Example: *If one wants to set* $v_{\max} = 0.2$ *and* $p_{\mathrm{cutoff}} = 2$ *GeV/c, than one should use the following settings:*

```
Bool_t bConstantHarmonics = kFALSE;
Double_t dV2RPMax = 0.20;
Double_t dPtCutOff = 2.0;
```

Example plot is given in Figure 5.

(Remark: Add further explanation here.)

3.3 Nonflow

One can simply simulate strong 2-particle nonflow correlations by taking each particle twice.

Example: *If one wants to simulate strong 2-particle nonflow correlations one should simply set*

```
Int_t iLoops = 2;
```

3.4 Detector inefficiencies

In reality we never deal with a detector with uniform azimuthal coverage, hence a need for a thorough studies of the systematic bias originating from the non-uniform acceptance.

Example: *One wants to simulate a detector whose acceptance is uniform except for the sector which spans the azimuthal interval* $[60^{\mathrm{o}}, 120^{\mathrm{o}}]$. *In this sector there are some issues, so only half of the particles are reconstructed. To simulate this acceptance one should use the following settings:*

```
Bool_t uniformAcceptance = kFALSE;
Double_t phimin1 = 60;
Double_t phimax1 = 120;
Double_t p1 = 1/2.;
Double_t phimin2 = 0.0;
Double_t phimax2 = 0.0;
Double_t p2 = 0.0;
```

The resulting azimuthal profile is shown in Figure (6).

One can also simulate two problematic sectors.

Example: *One wants to simulate a detector whose acceptance is uniform except for the two sectors which span azimuth* $[60^{\mathrm{o}}, 120^{\mathrm{o}}]$ *and* $[270^{\mathrm{o}}, 330^{\mathrm{o}}]$, *respectively. In the first sector only 1/2 of the particles are reconstructed and only 1/3 of the particles are reconstructed in the second. To simulate this acceptance one should use the following settings:*

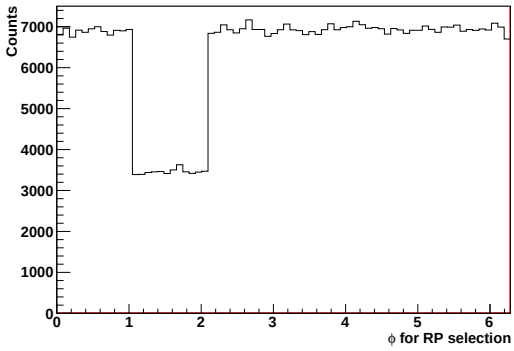

Figure 6: Non-uniform acceptance.

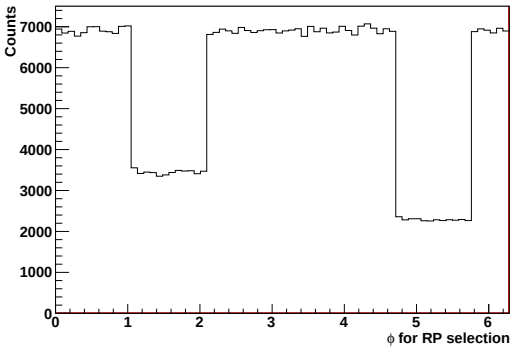

Figure 7: Non-uniform acceptance.

```
Bool_t uniformAcceptance = kFALSE;
Double_t phimin1 = 60;
Double_t phimax1 = 120;
Double_t p1 = 1/2.;
Double_t phimin2 = 270.0;
Double_t phimax2 = 330.0;
Double_t p2 = 1/3.;
```

The resulting azimuthal profile is shown in Figure (7).

3.5 Selecting the methods

3.6 Particle weights

4 Extra

4.1 Running on Stoomboot