

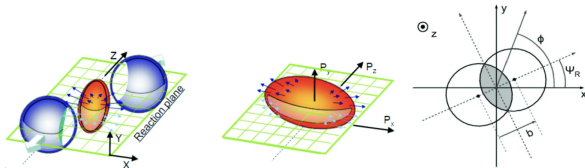
Event plane reconstruction in O2Physics

Junlee Kim¹

November 13, 2025
O2 Tutorial 5.0

1. CERN

Event plane reconstruction



- Flow coefficients can be extracted using the reaction plane of ψ_n
 - $v_n = \langle \cos[n(\varphi - \psi_n)] \rangle$
 - Not possible to access ψ_n in experiment.
- Alternatively, the event plane (Ψ_n) is reconstructed for n -th modulations using the Q -vector
 - $Q_{n,x} = \sum_i \omega_i \cos(n\varphi_i)$ and $Q_{n,y} = \sum_i \omega_i \sin(n\varphi_i)$
 - ω_i : weight factor of each component, detector amplitude or p_T ($v_n \propto p_T$)
 - Gain equalization for FIT detector with a factor of $(M_{\text{tot}}^{\text{avg}}/M_i^{\text{avg}})$
 - $\Psi_n = (1/n) \arctan(Q_{n,y}/Q_{n,x})$
 - $v_n^{\text{obs}} = \langle \cos[n(\varphi - \Psi_n)] \rangle$
- The resolution of the event plane can be calculated with 3-sub event method
 - $$\mathcal{R}_n = \langle \cos(n(\Psi_n^A - \psi_n)) \rangle \approx \sqrt{\frac{\langle \cos(n(\Psi_n^A - \Psi_n^B)) \rangle \langle \cos(n(\Psi_n^A - \Psi_n^C)) \rangle}{\langle \cos(n(\Psi_n^B - \Psi_n^C)) \rangle}}$$
 - Individual booking for $\langle \cos(n(\Psi_n^A - \Psi_n^B)) \rangle$, $\langle \cos(n(\Psi_n^A - \Psi_n^C)) \rangle$, and $\langle \cos(n(\Psi_n^B - \Psi_n^C)) \rangle$.
 - $v_n^{\text{cor}} = v_n^{\text{obs}}/\mathcal{R}_n$

- Data Model: `Common/DataModel/Qvectors.h`
 - `DECLARE_SOA_COLUMN(QvecDetRe(Im)Vec, qvecDetRe(Im)Vec, std::vector<float>);`
 - `DECLARE_SOA_TABLE(QvectorFTOCVecs, "AOD", "QVECTORSFTOCVEC", qvec::IsCalibrated, qvec::QvecFTOCReVec, qvec::QvecFTOCImVec, qvec::SumAmplFTOC);`
 - Support to save multiple harmonic orders for different subsystems
- Table Producer: `Common/TableProducer/qVectorsTable.cxx`
 - Main piece to calculate and correct Q -Vectors.
 - Gain equalization and Q -Vector calibration
 - A few key configurable parameters to be emphasized.
- Helper: `Common/Core/EventPlaneHelper.h`
 - Provide relevant functions
- Validation Task: `Common/Tasks/qVectorsCorrection.cxx`
- Example: `PWGLF/Tasks/Strangeness/lambdapolarization.cxx`

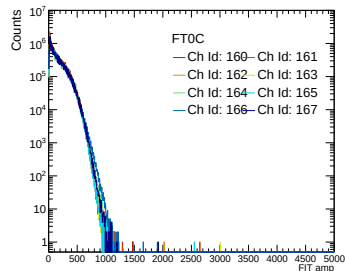
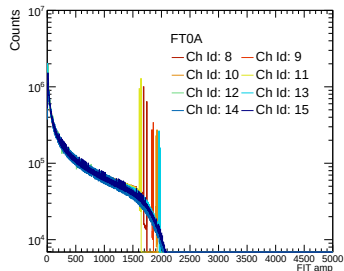
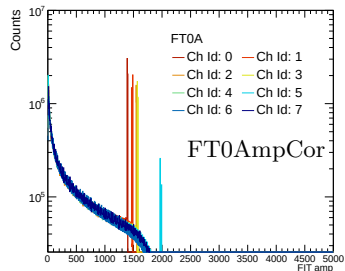
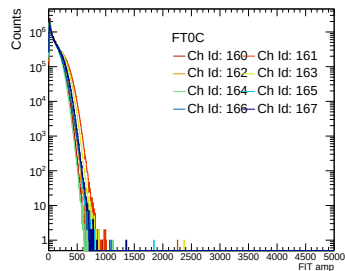
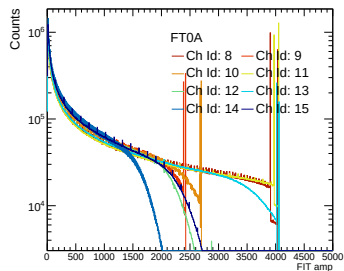
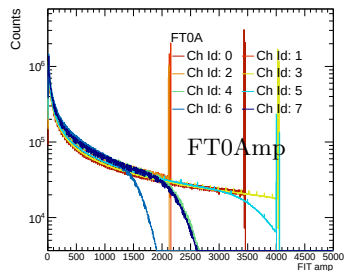
Preparation

- Gain equalization is required to ensure that every channel provides the same signal amplitude for a traversing charged particle.
- Require to have amplitude distribution for each channel
`histosQA.fill(HIST("FVOAmp"), ampl, FVOAchId);` (L.350, L.395)
- Average amplitudes of each channel and average amplitude of all channels to calculate ($M_{\text{tot}}^{\text{avg}}/M_i^{\text{avg}}$)
- Upload via `ccdb.storeAsTFileAny(VECTOR, ccdbInternalPath.c_str() metadata, sor, eor);`
- `cfgGainEqPath` in Table Producer: CCDB path for accessing gain equalization constants

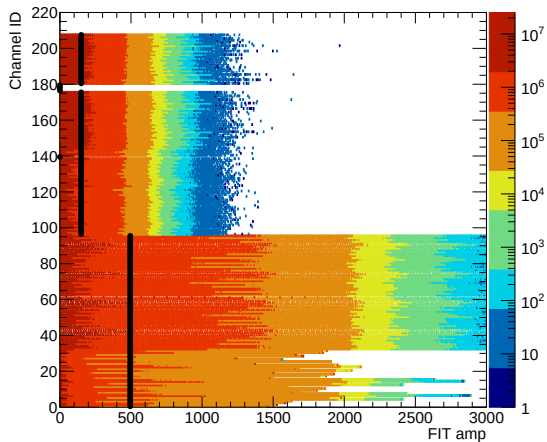
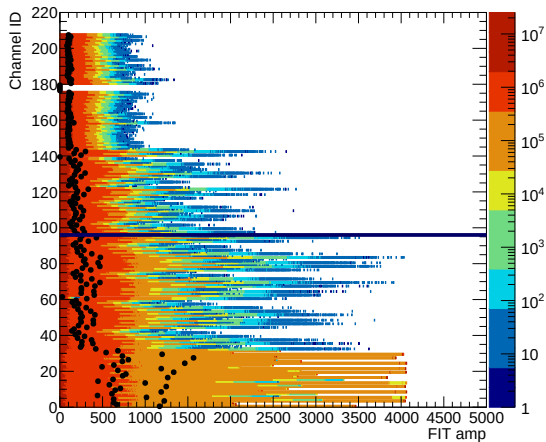
Equalization

- Load gain via `auto objft0Gain = ccdb->getForTimeStamp<std::vector<float>>(fullPath, timestamp);`
- Validate the process with `histosQA.fill(HIST("FT0AmpCor"), ampl / FT0RelGainConst[FT0CchId], FT0CchId);`

Equalized amplitude distribution



Raw amplitude distribution



- Gain equalization has been done for FT0C and FT0A (the same treatment for V0A and V0C in Run2)

Preparation

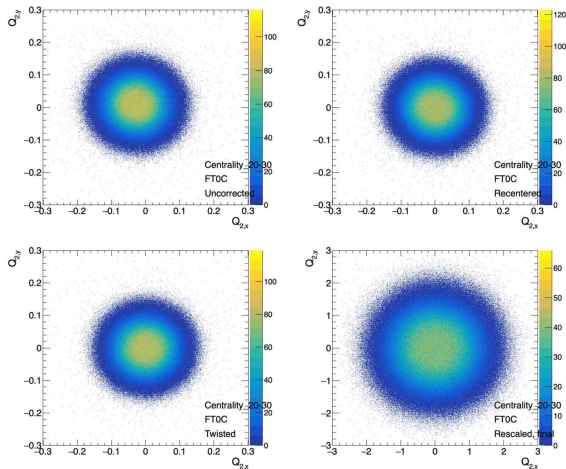
- Uncorrected 2-D scatter plots of $Q_{n,x}$ and $Q_{n,y}$.
- Generate correction constants from the scatter plot based on functions in `Common/Core/EventPlaneHelper.h`
 - TH3 with 6 (calibration step) $\times$ 7 (subsystems) $\times$ 90 (centrality intervals of 1%) bin size
- Upload constants in the CCDB (L.246)

Correction

- Load gain via `auto objqvec = ccdb->getForTimeStamp<TH3F>(fullPath, timestamp);`
- Validate the process with `Common/Tasks/qVectorsCorrection.cxx`

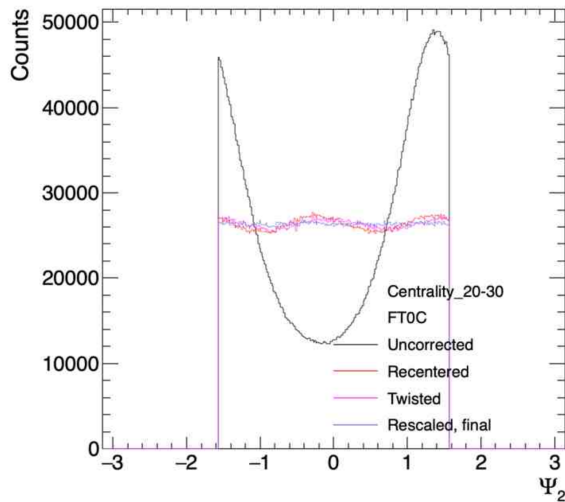
Q-Vector correction

- Scatter plot of $Q_{n,x}$ and $Q_{n,y}$ would be isotropic around the origin.
 - Uniform event plane distribution
- In reality, $(Q_{n,x}, Q_{n,y})$ distribution is not centered and elongated.
 - **Recentering** corrects the center position
 - **Twist** corrects the possible rotation
 - **Rescaling** corrects the possible deformation
- Prerequisite: gain equalization



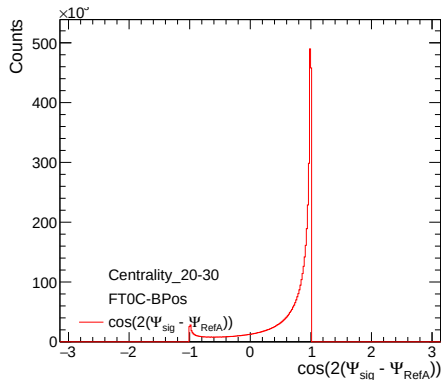
Event plane distribution

- $\Psi_2 = (1/2) \arctan(Q_{2,y}/Q_{2,x})$
- Fully corrected distribution is uniform as expected.



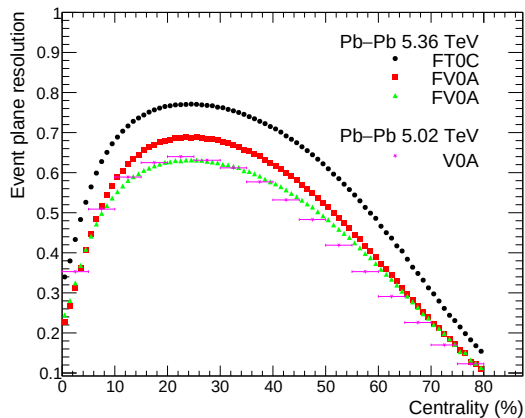
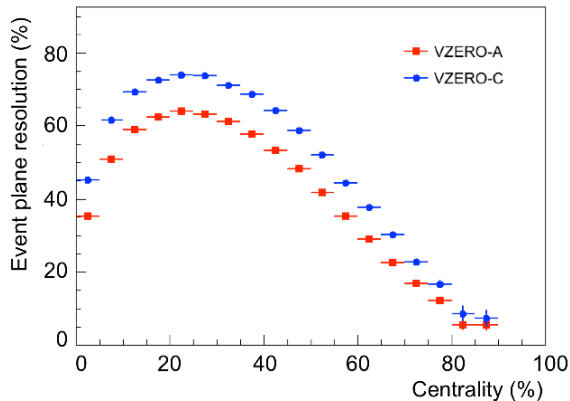
The resolution of the event plane

- $\mathcal{R}_n = \langle \cos(n(\Psi_n^A - \psi_n)) \rangle \approx \sqrt{\frac{\langle \cos(n(\Psi_n^A - \Psi_n^B)) \rangle \langle \cos(n(\Psi_n^A - \Psi_n^C)) \rangle}{\langle \cos(n(\Psi_n^B - \Psi_n^C)) \rangle}}$
- $v_n^{\text{cor}} = v_n^{\text{obs}} / \mathcal{R}_n$
- Individual booking for $\langle \cos(n(\Psi_n^A - \Psi_n^B)) \rangle$, $\langle \cos(n(\Psi_n^A - \Psi_n^C)) \rangle$, and $\langle \cos(n(\Psi_n^B - \Psi_n^C)) \rangle$
 - The figure shows $\langle \cos(n(\Psi_n^A - \Psi_n^B)) \rangle$
- The average value of distributions is used to evaluate the resolution
- histogram generation with `Common/Tasks/qVectorsCorrection.cxx`



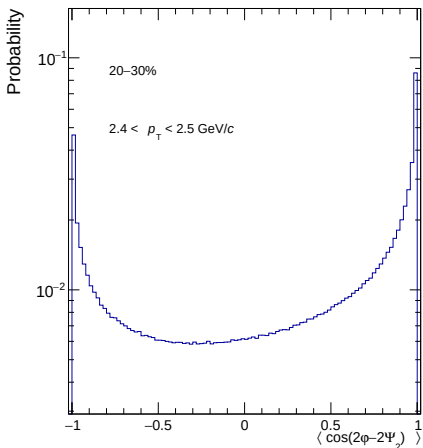
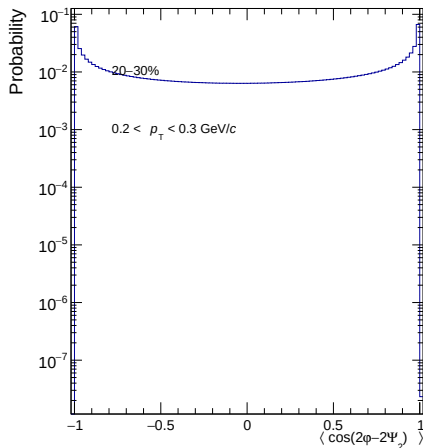
Event plane resolution

2013 JINST 8 P10016



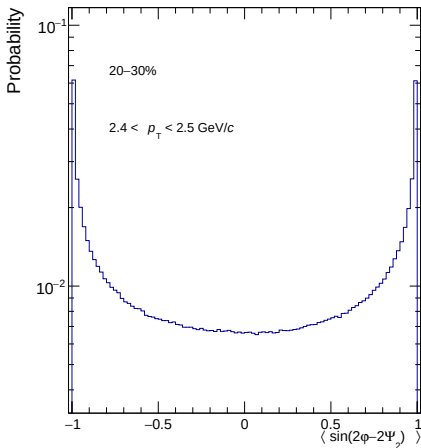
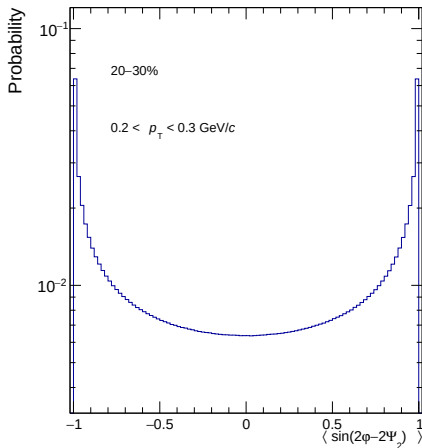
- Run 2 data with plot digitizer (no public data)
- Comparable event plane resolution for V0A (bit higher in peripheral collisions)

$$\langle \cos [n(\varphi - \Psi_n)] \rangle$$



- Average of the histogram to be signal

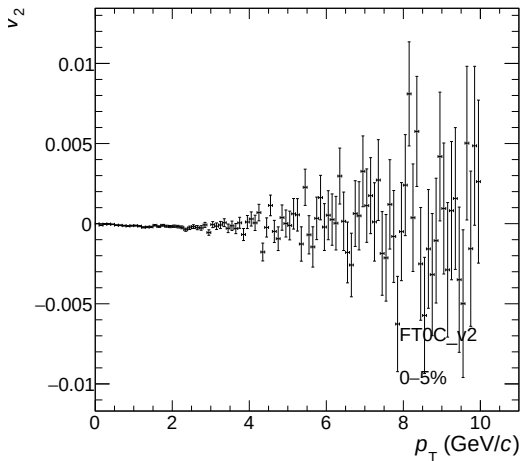
$$\langle \sin [n(\varphi - \Psi_n)] \rangle$$



- Average of the histogram to examine possible bias
- $\langle \sin [n(\varphi - \Psi_n)] \rangle = 0 \rightarrow$ well decomposed to cosine

$$\langle \sin [n(\varphi - \Psi_n)] \rangle$$

- $\langle \sin [n(\varphi - \Psi_n)] \rangle = 0$ throughout all measured p_T
 $\rightarrow$ no bias is expected



Goal of the present tutorial

Implementation of shift correction in OO collisions

- Need to call input parameters from CCDB
- One can utilize this for online efficiency correction

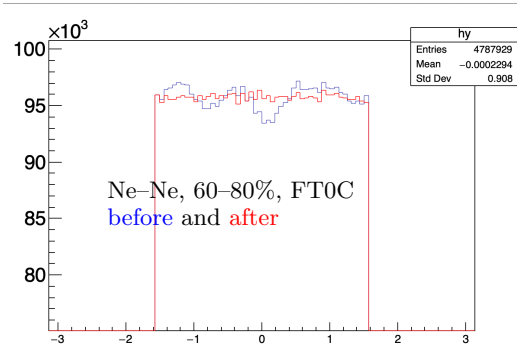
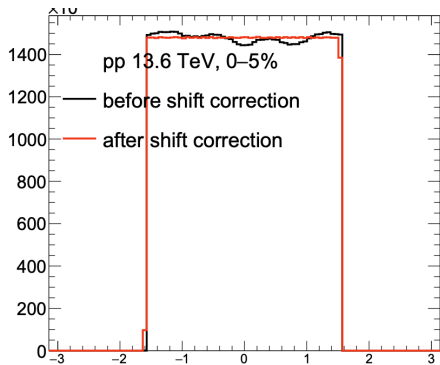
Flow measurement in light-ion collisions

- One of the most unambiguous evidence of QGP formation
- Booking histogram, post projection of THnSparse, scalar product method...

When you develop the code

- The location of the code `02Physics/Tutorials/PWGCF/EventPlane/src/qVectorstutorial.cxx`
- `alienv enter ninja::latest → cd alice/sw/BUILD/02Physics-latest/02Physics → ninja Tutorials/install`
- Task chain: `o2-analysis-ft0-corrected-table -b --configuration json://configuration.json | o2-analysis-cf-qvectors-tutorial -b --configuration json://configuration.json | o2-analysis-multcenttable -b --configuration json://configuration.json | o2-analysis-event-selection-service -b --configuration json://configuration.json | o2-analysis-propagationservice -b --configuration json://configuration.json | o2-analysis-qvector-table -b --configuration json://configuration.json | o2-analysis-trackselection -b --configuration json://configuration.json --aod-file @input_data.txt`

Shift correction for small systems



- Additional shift correction applied (Phys.Rev.C56:3254-3264,1997)
- Working for both small systems

$$v_n\{\text{EP}\} = \frac{\langle Q_n Q_{n,A}^* / |Q_{n,A}| \rangle}{\sqrt{\langle Q_{n,A} Q_{n,B}^* / |Q_{n,A}| |Q_{n,B}| \rangle}} = \frac{\langle v_n R_n \rangle}{\langle R^2 \rangle}$$

- Not a simple weighting $\rightarrow$ biases from event-by-event fluctuation of R
- $v_n\{\text{EP}\} \sim v_n$ when ($R \sim 1$), $v_n\{\text{EP}\} \sim \sqrt{v_n^2}$ when ($R \ll 1$)

$$v_n\{\text{SP}\} = \frac{\langle Q_n Q_{n,A}^* \rangle}{\langle Q_{n,A} Q_{n,B}^* \rangle} = \frac{\langle v_n v_{n,A} \rangle}{\sqrt{\langle v_{n,A}^2 \rangle}}$$

- Always $v_n\{\text{SP}\} = \sqrt{v_n^2}$

- Preparation of CCDB connection
 - L40-, L59-, L77-. L238
- Loading input parameters for the shift correction
 - L83-, L244-,
- Implementation of the shift correction
 - L157-
- Need to update configurable.json

Flow measurement with the scalar product method (after shift correction)

- Booking histograms
- Direct calculations of inner product of Q -vectors

BACKUP

Previous exercises

Before doing this make sure that you are in the O2Physics with `alienv enter O2Physics::latest`

Main code to work in this time: `Tutorials/PWGCF/EventPlane/src/qVectortutorial.cxx`

Constants for gain equalization and Q -Vector corrections are prepared at `Users/j/junlee/Qvector/Pass4`

You need to update tutorial code with `ninja Tutorials/PWGCF/EventPlane/install`

Get the [A02D.root](#), [macro](#), and [configuration](#) for running the task

- Checking gain equalization
 - See <http://alice-ccdb.cern.ch/browse/Users/j/junlee/Qvector/Pass4> and insert path to conf.
 - Plot and compare FT0Amp and FT0AmpCor
- Checking Q -Vector corrections
 - Q -Vectors with each calibration step can be obtained with `cfgCorrLevel`
 - Plot 2-D Q -Vectors for each step from 1 to 4
(`cfgCorrLevel=0` corresponds to the Q -Vector before gain eq.)
- Checking event plane distribution
 - Plot event plane distribution and check if it is uniform
- Measuring v_2 and correct it for event plane resolution
 - Event plane resolution with 3-sub event method: need to book more histograms
 - Book another histogram for sine term to see if any acceptance bias
 - Correct v_2 with the event plane resolution
- Draw higher-order such like v_3 and v_4