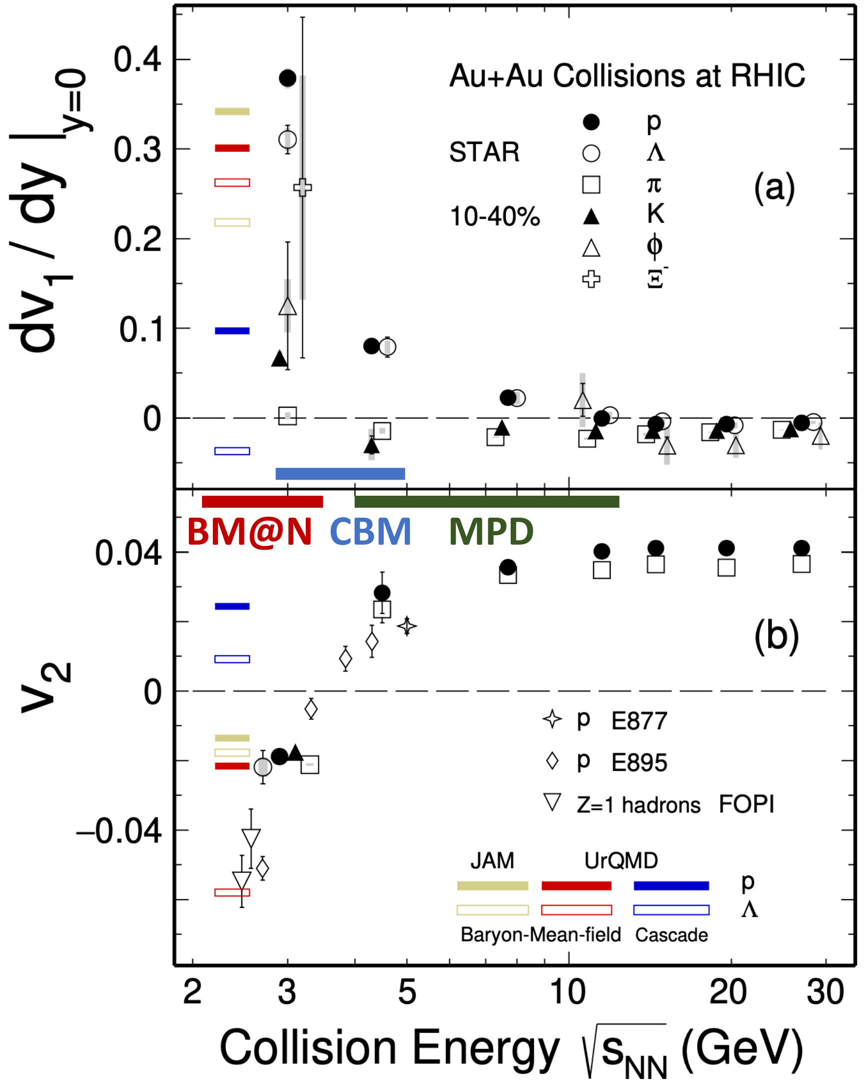
Implementation of QnAnalysis framework for flow measurements in MPD

Parfenov Petr, Evgeny Kashirin, Mikhail Mamaev, Oleg Golosov, Valerii Troshin
NRNU MEPhI

Cross-PWG meeting in MPD
12.04.2022

Anisotropic flow in heavy-ion collisions at Nuclotron-NICA energies

M. Abdallah et al. [STAR Collaboration] 2108.00908 [nucl-ex]



$$\frac{dN}{d\phi} \propto 1 + 2\sum_{n=1} \boldsymbol{v_n} \cos[n(\phi - \Psi_{RP})], \qquad v_n = \langle \cos[n(\phi - \Psi_{RP})] \rangle$$

v_1 – directed flow, v_2 – elliptic flow, v_3 – triangular flow, etc.

Strong energy dependence of dv_1/dy and v_2 at $\sqrt{s_{NN}}$=2-11 GeV

Anisotropic flow at FAIR/NICA energies is a delicate balance between:

I. The ability of pressure developed early in the reaction zone and

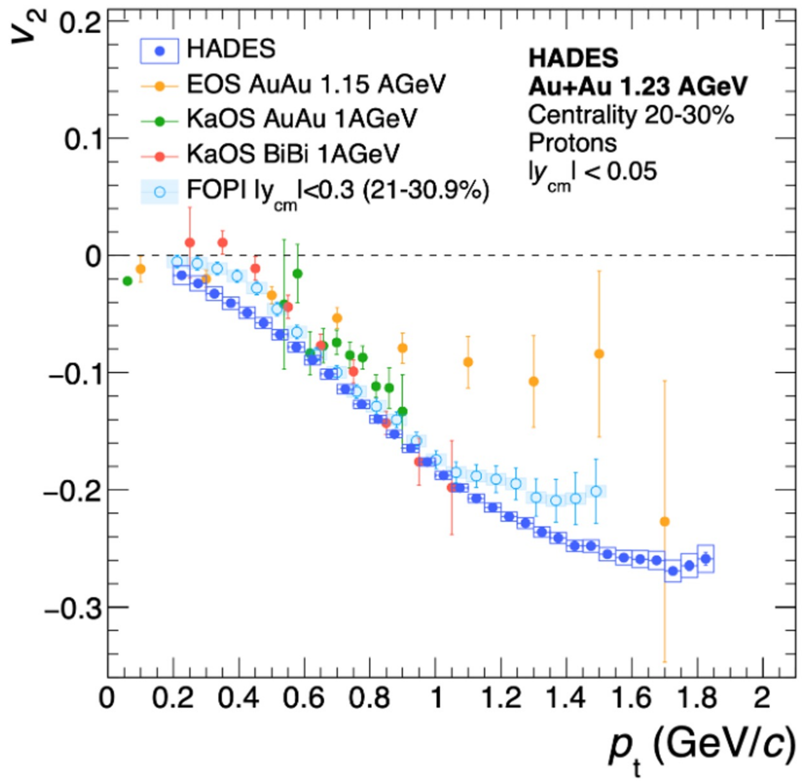
II. Long passage time (strong shadowing by spectators)

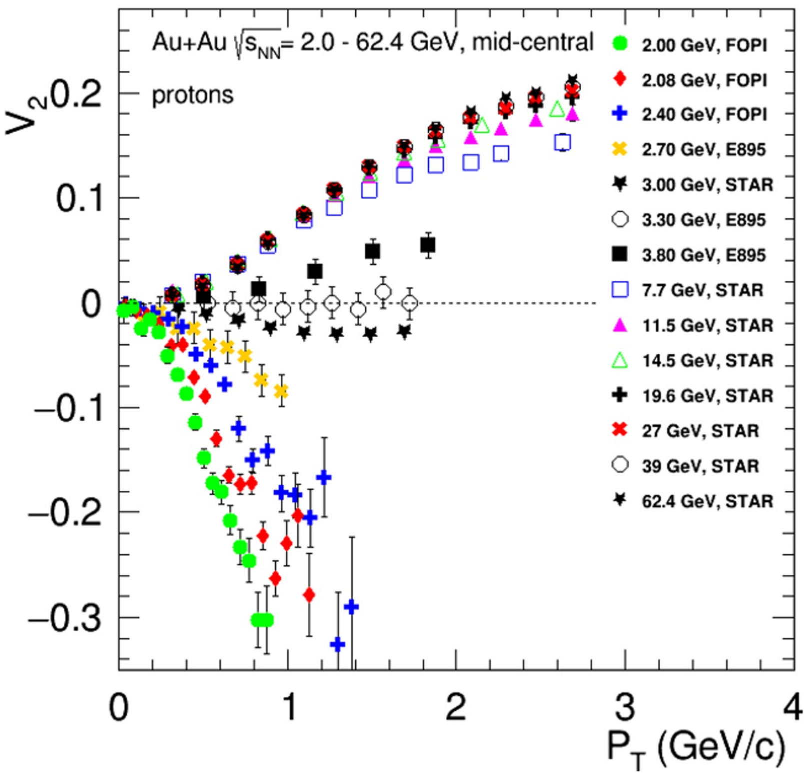
Differential flow measurements $v_n(\sqrt{s_{NN}}, \text{centrality}, \text{pid}, p_T, y)$ will help to study:

- effects of collective (radial) expansion on anisotropic flow
- interaction between collision spectators and produced matter
- baryon number transport

Several experiments (MPD, BM@N, STAR FXT, CBM, HADES, NA61/SHINE) aim to study properties of the strongly-interacted matter in this energy region

Why do we need unified package for flow analysis?





- Biggest systematics – difference between experiments (for example, FOPI vs. HADES)
- Problem with correction for detector acceptance

u_n, Q_n vectors formalism for flow measurements

- Unit vector of a particle $u_n(\mathrm{centrality}, \mathrm{pid}, p_T, y)$:

$$u_n = e^{in\varphi} = \begin{cases} u_{n,x} \equiv x_n = \cos n\varphi \\ u_{n,y} \equiv y_n = \sin n\varphi \end{cases}$$

- Event flow vector $Q_n(\mathrm{centrality})$:

$$Q_n = \sum_{k=1}^{M} \omega_n^k u_n^k \equiv |Q_n| e^{in\Psi_n} = \begin{cases} Q_{n,x} \equiv X_n = |Q_n| \cos n\Psi_n \\ Q_{n,y} \equiv Y_n = |Q_n| \sin n\Psi_n \end{cases}$$

  - φ – azimuthal angle of the produced particle
  - ω – weight of the Q_n vector (for example, $\omega = 1$ for participant plane and $\omega = E$ for spectator plane)
  - Ψ_n – event plane angle

u_n, Q_n vectors formalism for flow measurements

Flow can be measured using Q_n, u_n vectors:

$$v_n = \frac{\langle u_n^{\pm} Q_n^{\mp *} \rangle}{2\sqrt{\langle Q_n^{+} Q_n^{-*} \rangle}}, v_{n,xx} = \frac{\langle x_n^{\pm} X_n^{\mp *} \rangle}{\sqrt{2\langle X_n^{+} X_n^{-*} \rangle}}, v_{n,yy} = \frac{\langle y_n^{\pm} Y_n^{\mp *} \rangle}{\sqrt{2\langle Y_n^{+} Y_n^{-*} \rangle}}$$

Where " $\pm$ " – different subevents

Normalizations of Q_n vector:

- $|Q_n|$ (event plane method)
- 1 (scalar product method)

Corrections for non-uniform acceptance

Recentering:

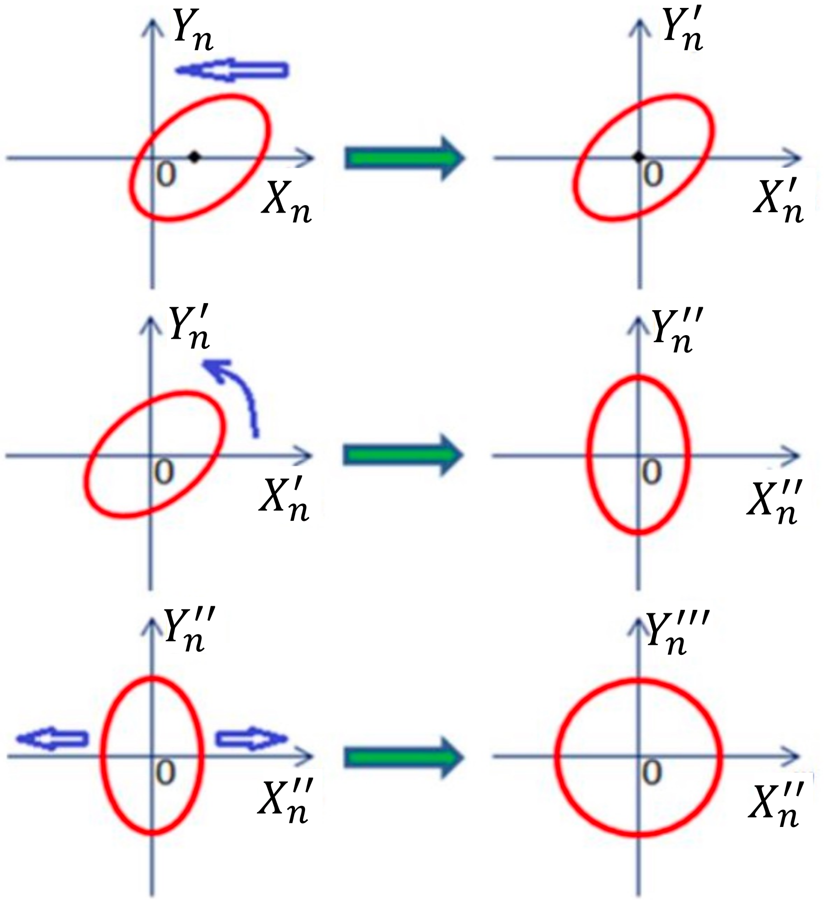
$$X_n' = X_n - \langle X_n \rangle, \qquad Y_n' = Y_n - \langle Y_n \rangle$$

Twist:

$$X_n'' = \frac{X_n' - \lambda_{2n}^{s-} Y_n'}{1 - \lambda_{2n}^{s-}\lambda_{2n}^{s+}}, \qquad Y_n'' = \frac{Y_n' - \lambda_{2n}^{s-} X_n'}{1 - \lambda_{2n}^{s-}\lambda_{2n}^{s+}}$$

Rescale:

$$X_n''' = \frac{X_n''}{a_{2n}^{+}}, \qquad Y_n''' = \frac{Y_n''}{a_{2n}^{-}}$$



Where $a_{2n}^{\pm} = 1 \pm \langle X_{2n} \rangle, \lambda_{m\mp n}^{s\pm} = \frac{v_m}{v_n}\frac{\langle Y_{m\mp n}\rangle}{a_{2n}^{\pm}}$

Corrections are based on method in:
I. Selyuzhenkov and S. Voloshin PRC77, 034904 (2008)

Corrections applicable for both Q_n and u_n vectors

The QnAnalysis package

Motivation:

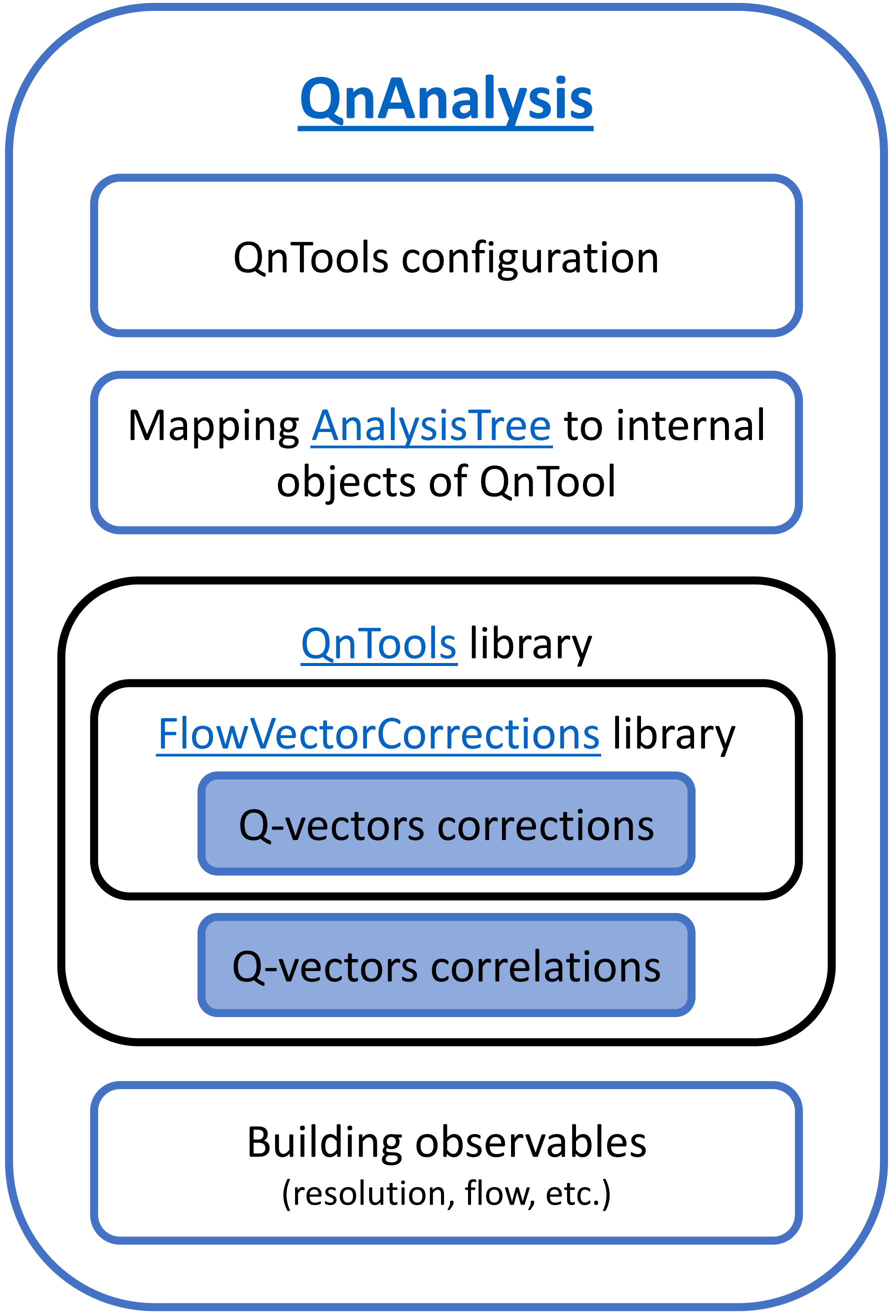
- Decoupling configuration from implementation
- Persistency of analysis setup
- Co-existence of different setups (easy systematics study)
- Unification of analysis methods
- Self-descriptiveness of the analysis results

QnAnalysis requirements:

- ROOT ver. ≧ 6.20 (with MathMore library)
- C++17 compatible compiler
- CMake ver. ≧ 3.13

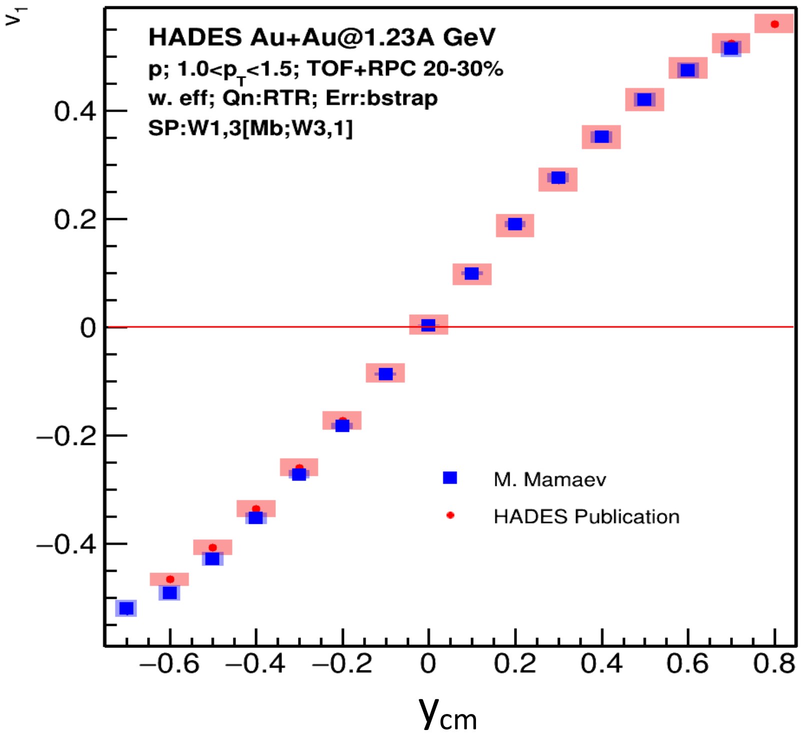
Can be easily installed on NICA cluster using ROOT and CMake modules

Git repository: https://github.com/HeavyIonAnalysis/QnAnalysis

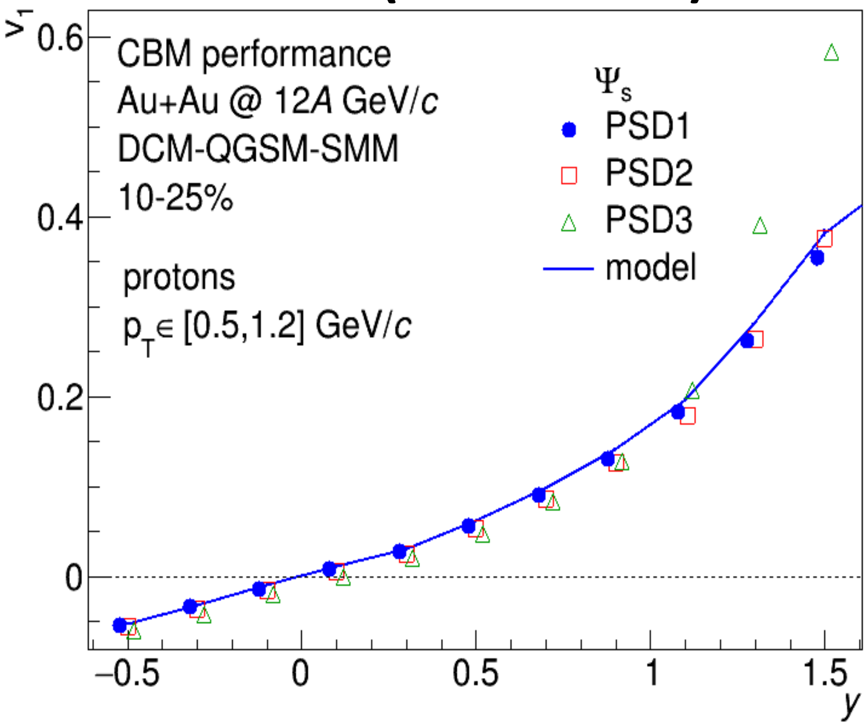


Examples of QnAnalysis usage

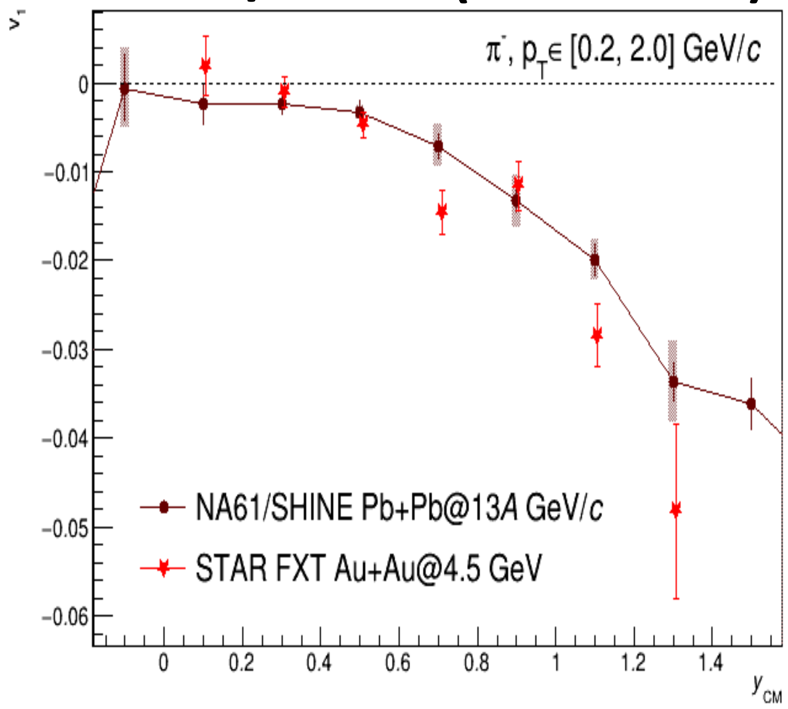
HADES (M. Mamaev)



CBM (O. Golosov)



NA61/SHINE (E. Kashirin)



QnAnalysis is already used in the current (HADES, ALICE) and future (CBM) experiments

Now it is available in MPD

AnalysisTree format for MPD data

AnalysisTree:

A framework and experimentally independent, lightweight and flexible data format that stores information in configurable basic objects:

- **EventHeader** – information about general event properties
- **Track** – reconstructed track parameters
- **Particle** – Monte Carlo track parameters
- **Module** – information about module in a module-type detector (FHCal)
- **Hit** – information about hit in a hit-type detector

Each object can contain any number of custom integer, floating or boolean fields

AnalysisTree can store information from any experiment and/or model

AnalysisTree

Core library

Data formats:

- EventHeader
- Track
- Particle
- Module
- Hit

Infra library

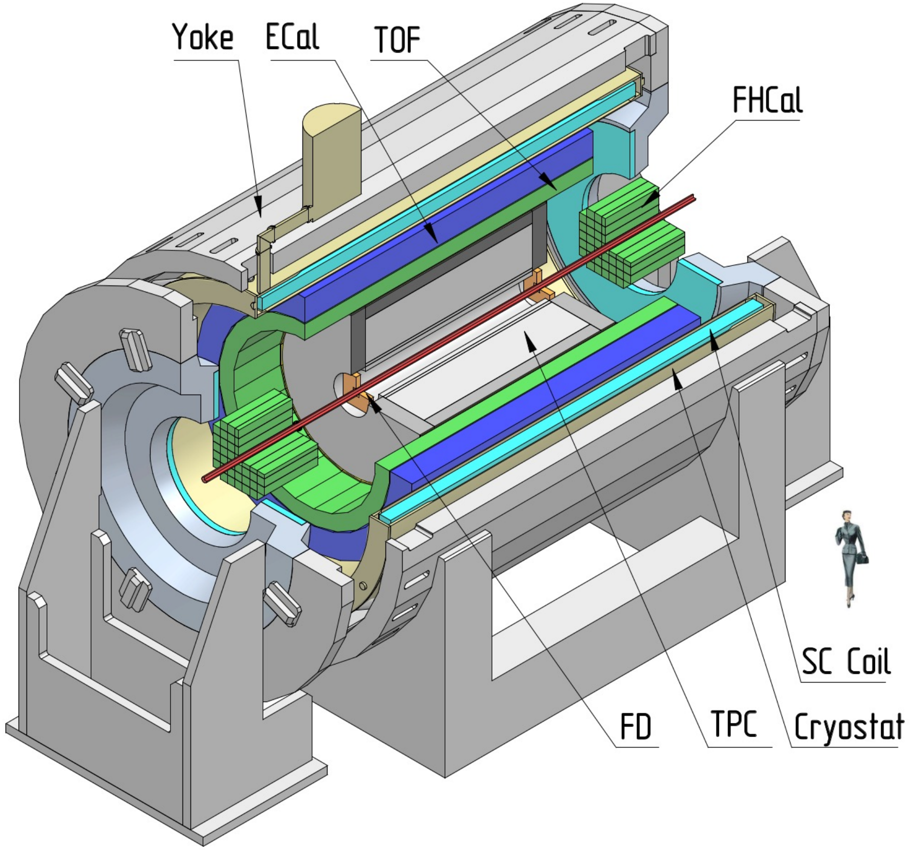
Ifrastructure for AnalysisTree:

- AnalysisTree reader
- AnalysisTree writer

AnalysisTree data format:
https://github.com/HeavyIonAnalysis/AnalysisTree

MPD experiment at NICA

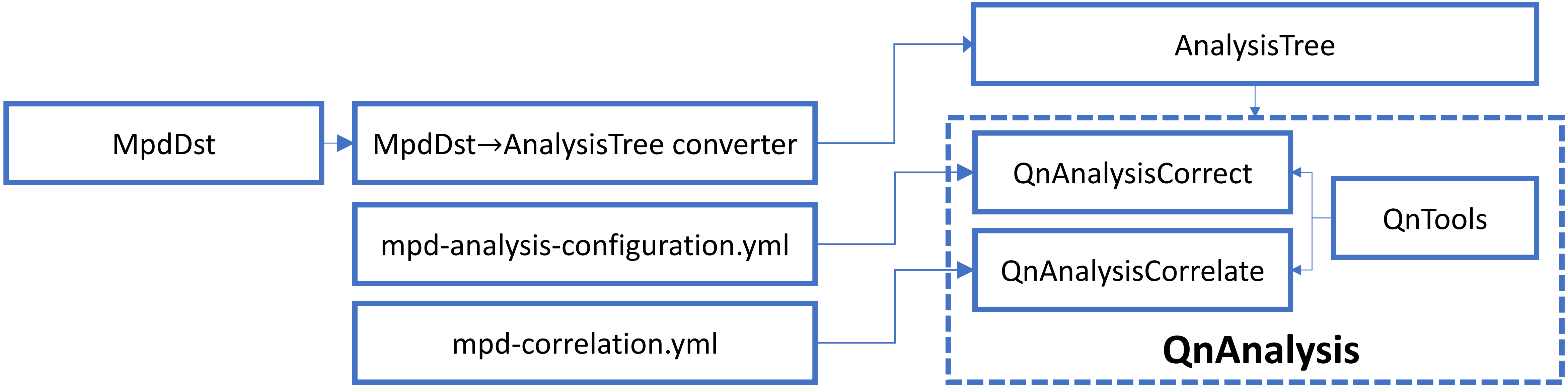
Multi Purpose Detector (MPD) Stage 1



- **Data set – official production (request 9):**
  - Au+Au at $\sqrt{s_{NN}}$ = 7.7 GeV (10M events)
- **Centrality determination:**
  - b based on MC-Glauber method
  - **Event plane determination:** TPC (for v_2), FHCal (for v_1)
- **Track selection:**
  - Primary tracks
  - $N_{hits}^{\mathrm{TPC}} > 16$
  - $|\eta| < 1.5$
  - $p_{\mathrm{T}} < 3.0\ \mathrm{GeV}/c$
  - PID based on PDG

UrQMD → GEANT4 → Reconstruction → Flow analysis

QnAnalysis implementation in MPD experiment



MPD-specific interface:

- **MpdDst→AnalysisTree converter:** converter from MpdDst to AnalysisTree format
- YAML configuration files for QnAnalysis:
  - **mpd-analysis-configuration.yml**: sets up Q_n, u_n vectors to collect (cuts, correction steps, …)
  - **mpd-correlation.yml:** sets up correlations between previously collected Q_n, u_n vectors

General interface:

- **AnalysisTree:** A framework-independent, lightweight and flexible data format
- **QnTools:** set of tools for multidimentional Q-vector-based corrections and correlations:
  - **QnAnalysisCorrect**: collects Q_n, u_n vectors
  - **QnAnalysisCorrelate**: make correction between collected Q_n, u_n vectors

Joint development with FAIR (CBM for NICA)

QnAnalysis is already used in the current (HADES, ALICE) and future (CBM) experiments – now available for MPD

QnAnalysis git link: https://github.com/HeavyIonAnalysis/QnAnalysis

AnalysisTree git link: https://github.com/HeavyIonAnalysis/AnalysisTree

Flow measurement procedure in MPD with QnAnalysis

The whole procedure can be divided into 3 main steps:

Preprocessing → Analysis → Postprocessing

Preprocessing

- Conversion to AnalysisTree
- Necessary calibrations (centrality determination, pid, etc.)
- Creating additional AnalysisTree with extended information

Analysis

- Configure Q-vectors
- Run **QnAnalysisCorrect** to measure and correct Q-vectors
- Configure correlations between Q-vectors
- Run **QnAnalysisCorrelate** to collect correlations

Postprocessing

- Obtain v_n from correlations
- Get TGraphErrors for v_n

Working templates for all steps can be found here:
https://devel.mephi.ru/PEParfenov/QnAnalysisMPD_scripts

Preprocessing stage

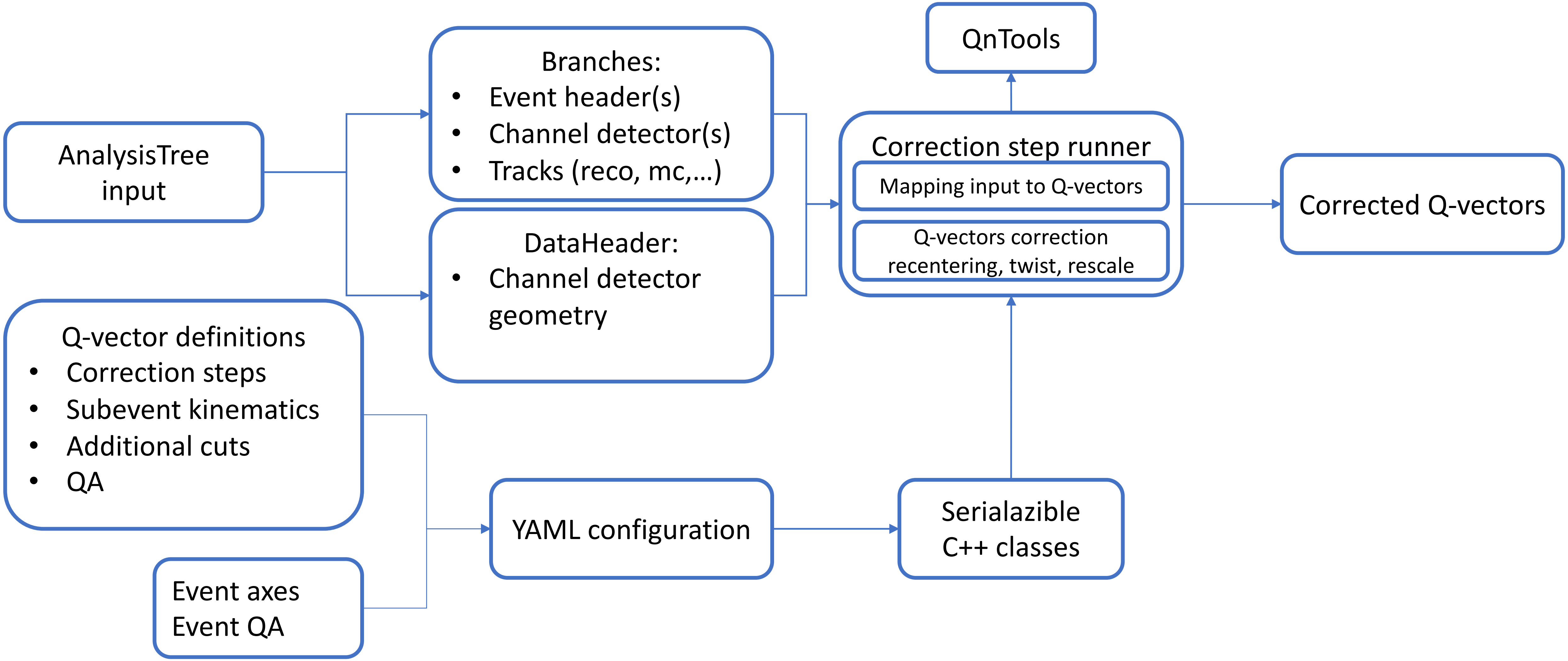
- Converter from MpdDst to base AnalysisTree – a simple ROOT macro:

root -l -b -q MpdDst2AT.C'("mpddst.root","AnalysisTree.root","System",$\sqrt{s_{NN}}$)'

- Run additional calibration (centrality determination, pid, candidates from KFParticleFinder, etc.) if needed
- Create an additional AnalysisTree with extended information from previous step using run_write_task.cpp and UserTaskWrite class (needed to be compiled)

Prepared setup examples for MPD are available here:
https://devel.mephi.ru/PEParfenov/QnAnalysisMPD_scripts

Analysis stage: QnAnalysisCorrect



Configuration example: Q-vector definition

AnalysisTree variable
<branch>/<field>

```yaml
# --- u(centrality, pT, y) vector for protons in TPC L
- name: reco_protons_L
  type: track
  phi:  TpcTracksExt/phi
  weight: Ones
  norm: m
  corrections:
    - recentering
    - twist-and-rescale
  axes:
  - *axis_reco_pT
  - *axis_reco_rapidity
  cuts:
    TpcTracksExt/eta: { range: [-1.5, -0.05] }
    TpcTracksExt/mc_mother_id: { equals: -1 }
    TpcTracksExt/mc_pid: { equals: 2212 }
    TpcTracksExt/nhits: { range: [16, 100]}
  qa:
    - {name: TpcTracksExt/phi, nb: 100, lo: -4., hi: 4.}
    - *axis_reco_pT
    - *axis_reco_rapidity
    - [*axis_reco_rapidity, *axis_reco_pT]
```

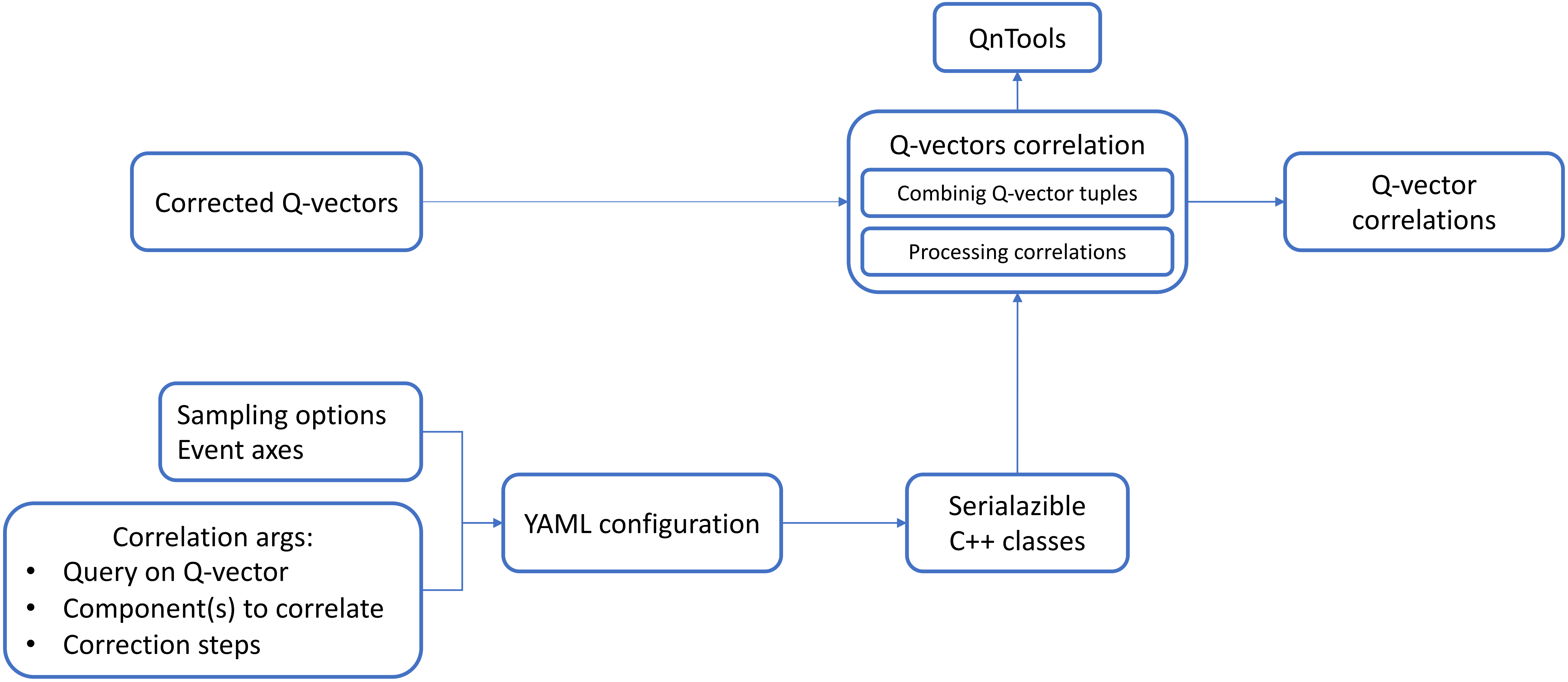
Reusable elements using YAML substitution

Here, the Q-vector was defined with the following cuts:

- $-1.5 < \eta < -0.05$ (TPC L)
- $\mathrm{motherId} = 1$ (primary track)
- $\mathrm{pdg} = 2212$ (protons)
- $16 < N_{hits} < 100$ (track quality)

Prepared setup examples for MPD are available here:
https://devel.mephi.ru/PEParfenov/QnAnalysisMPD_scripts

Analysis stage: QnAnalysisCorrelate



Configuration example: Correlation setup

```
# --- Set up u,Q vectors from QnAnalysisCorrect
_detectors: &detectors_reco
  - name: reco_hadrons_L
    tags: [ un_vector ]
    correction-step: plain
  - name: reco_hadrons_R
    tags: [ un_vector ]
```

```
  - name: reco_TPC_EP_L
    tags: [ qn_vector ]
    correction-step: plain
  - name: reco_TPC_EP_R
    tags: [ qn_vector ]
    correction-step: plain
```

Here, the correlations $\langle u_2(\text{centrality}, p_T, y) * Q_2(\text{centrality})\rangle$ are defined with tags "un_vector" and "qn_vector" correspondingly

```
# <u2 x Q2> with scalar product method
- args:
    - query: { tags: { any-in: [ un_vector ] } }
      query-list: *detectors_reco
      components: *v2_sp_components
      correction-steps: [ plain ]
      weight: sumw
    - query: { tags: { any-in: [ qn_vector ] } }
      query-list: *detectors_reco
      components: *v2_sp_components
      correction-steps: [ plain ]
      weight: ones
  n-samples: 50
  weights-type: observable
  folder: "/v2/uQ/SP"
  axes: [ *centrality ]
```

Prepared setup examples for MPD are available here:
https://devel.mephi.ru/PEParfenov/QnAnalysisMPD_scripts

Postprocessing stage

- Calculate v_n from correlations and save them as TGraphErrors – a simple ROOT macro:

root -l -b -q Draw_graphs.C'("correlation_out.root","graphs.root")'

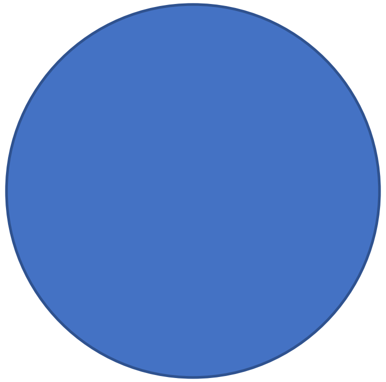
One can do simple arithmetic operations with correlations (+,-,*,/,sqrt(),…). That way, it is easy to calculate v_n. For example, for scalar product one can get $\langle u_n^{\pm} Q_n^{\mp *}\rangle$ and $\langle Q_n^{+} Q_n^{-*}\rangle$ and construct $v_n, v_{n,xx}, v_{n,yy}$.

QnAnalysis allows to contruct differential v_n signal for any component of $u_n = (x_n, y_n)$ and $Q_n = (X_n, Y_n)$ separately

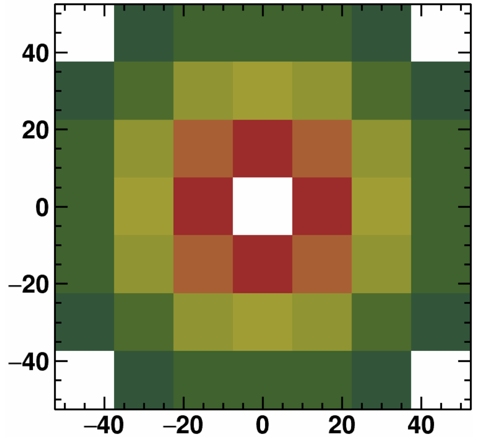
Prepared setup examples for MPD are available here:
https://devel.mephi.ru/PEParfenov/QnAnalysisMPD_scripts

Non-uniform acceptance corrections

TPC

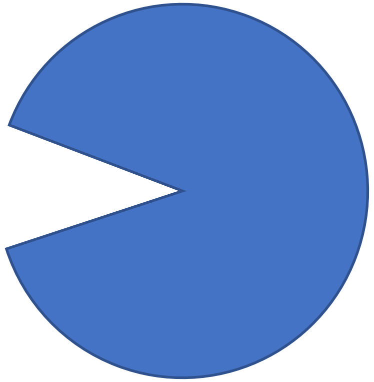


FHCal

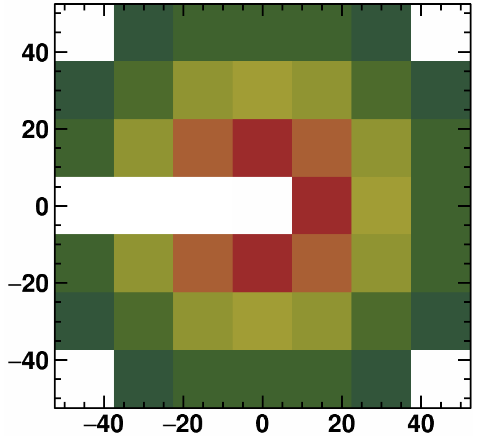


Acceptance filter

TPC



FHCal

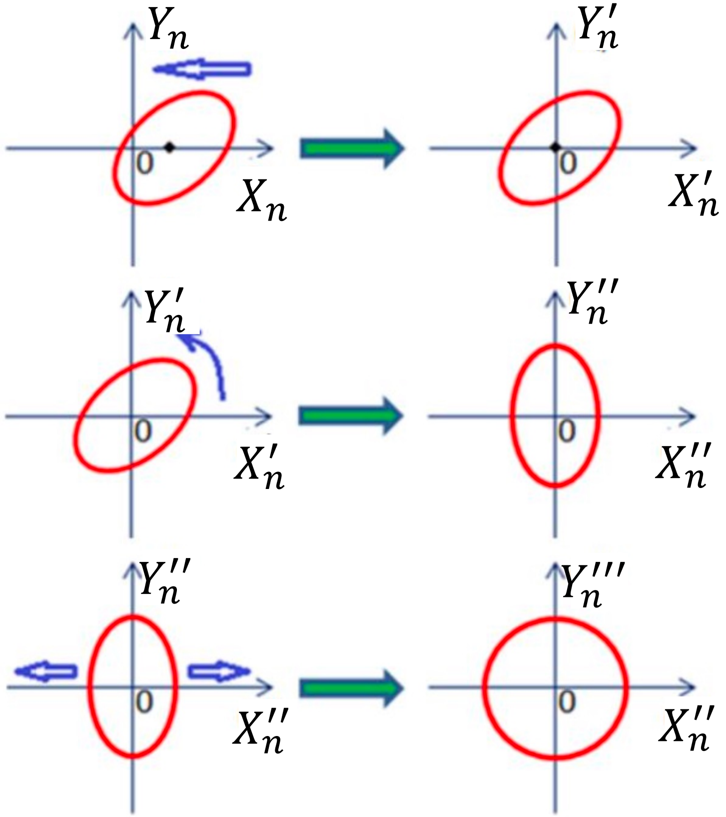


$-165° < \varphi < 165°$

Modules 23, 24, 25 (L) and 19, 20, 21 (R) are off

Correction for non-uniform azimuthal acceptance

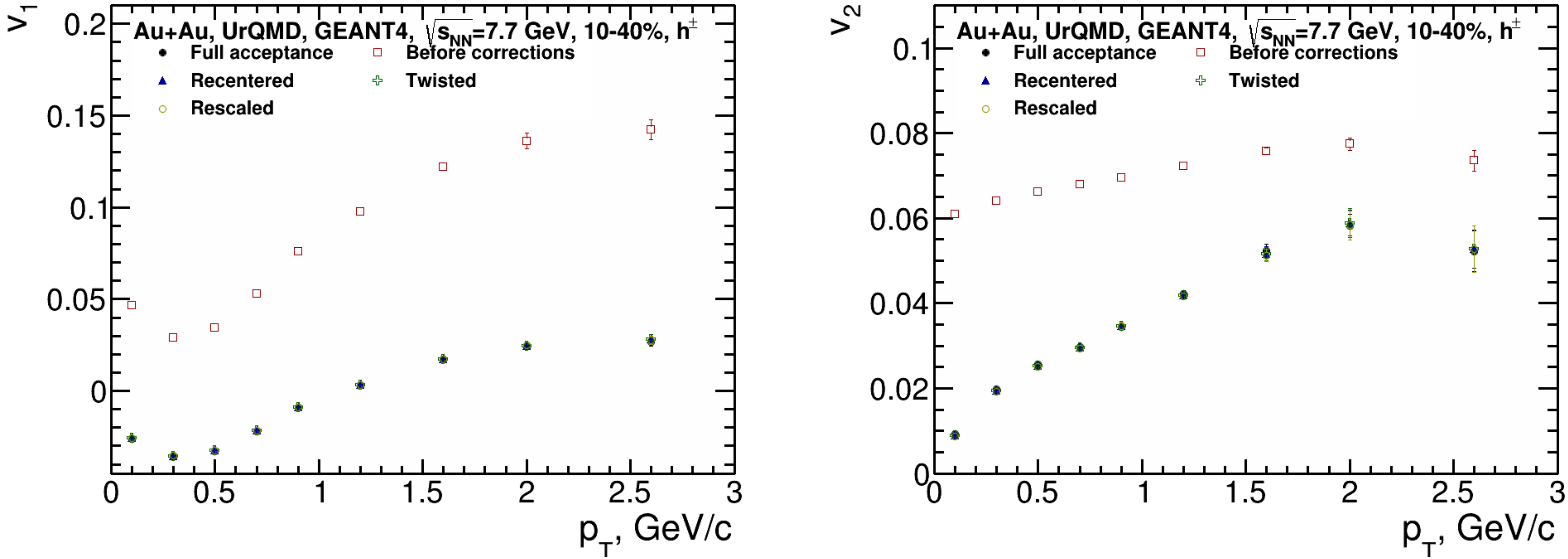
1. Recentering
2. Twist
3. Rescaling



Corrections are based on method in:

I. Selyuzhenkov and S. Voloshin PRC77, 034904 (2008)

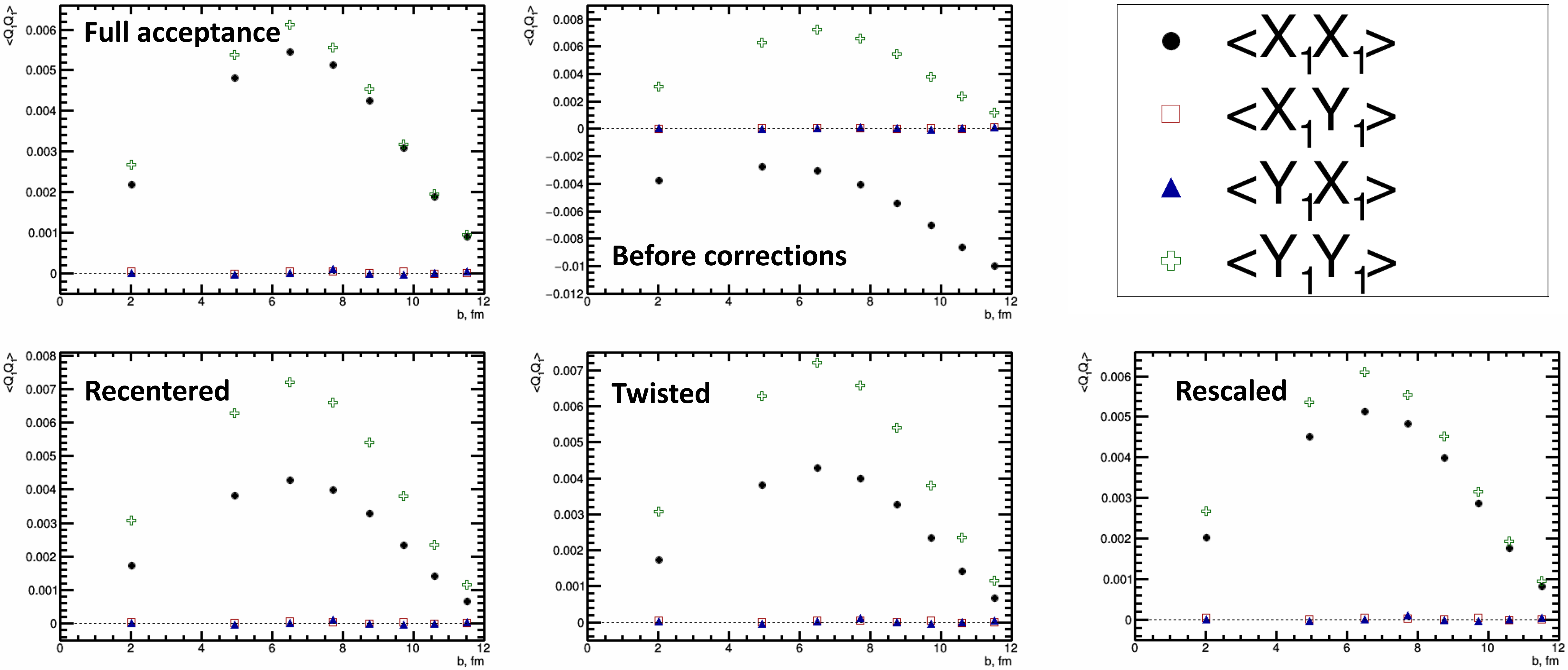
Effects of non-uniformity corrections



Q-vector	Q_n weight	Correction axes	Correction steps	Error calculation	Q_n normalization
Spectators (FHCal)	Module energy	b [0,12], 9 bins	Recentering Twist Rescaling	Bootstrapping, 50 samples	Sum of weights
Charged hadrons (TPC)	1	pT [0,3], 9 bins b [0,12], 9 bins			

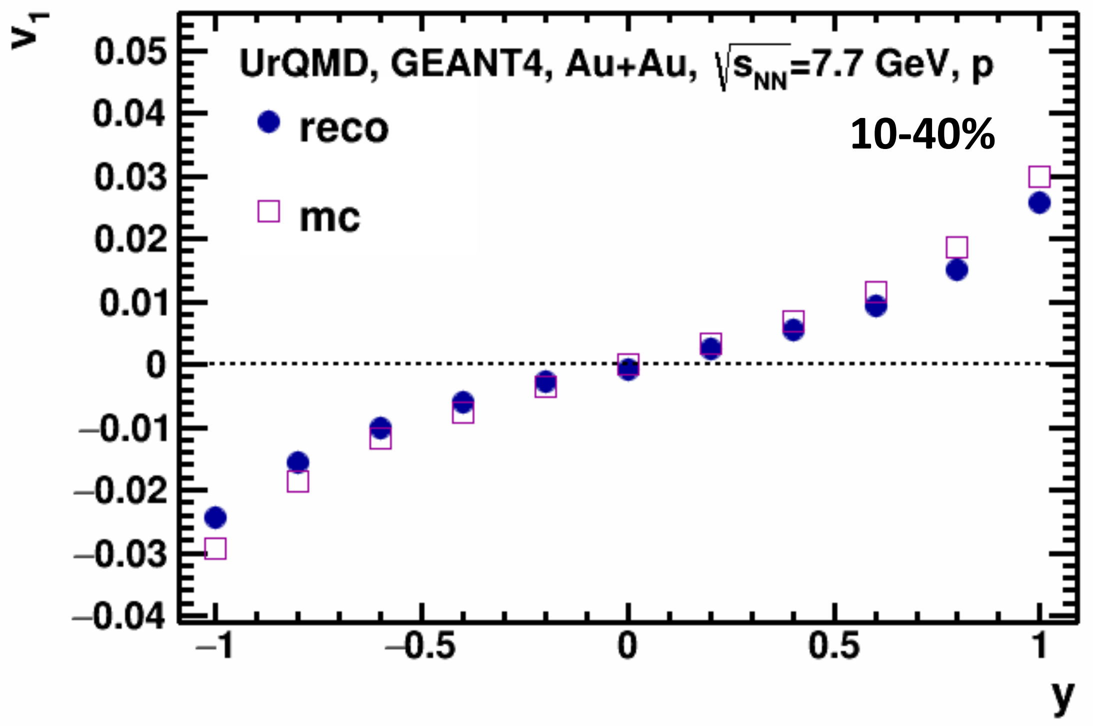
Good agreement between v_n with acceptance non-uniformity corrections and full acceptance

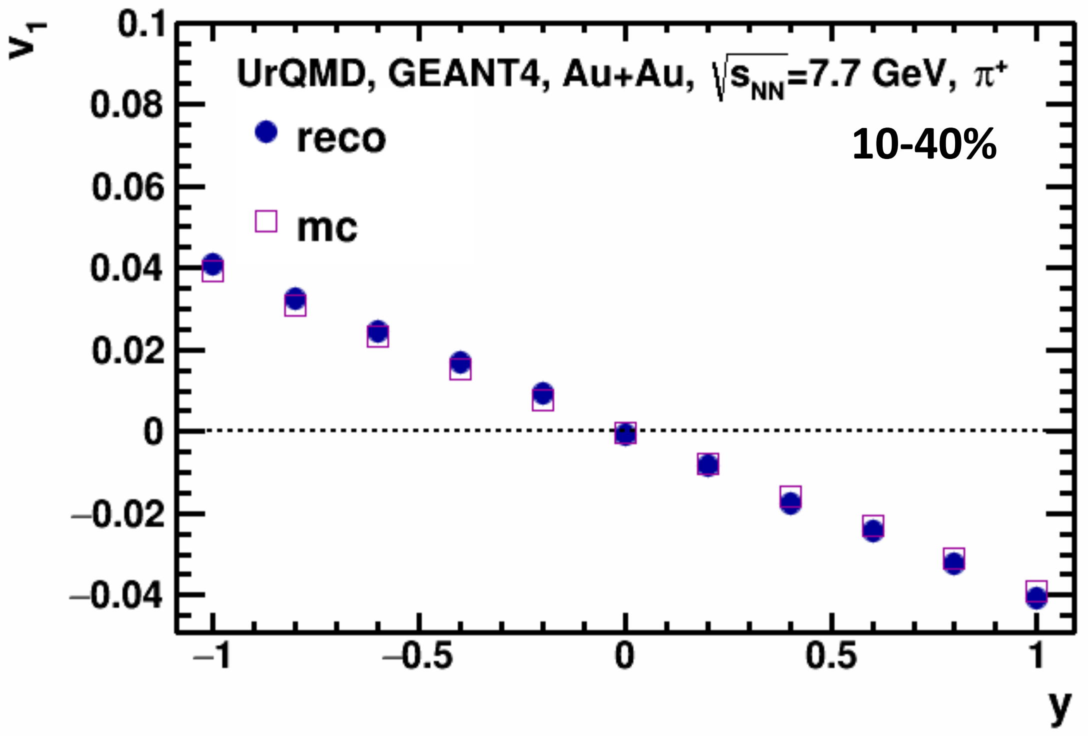
$\langle Q_n Q_n \rangle$ components (FHCal)



Looking at different components of $\langle Q_n Q_n \rangle$ provides more detailed information about effects of non-uniform acceptance

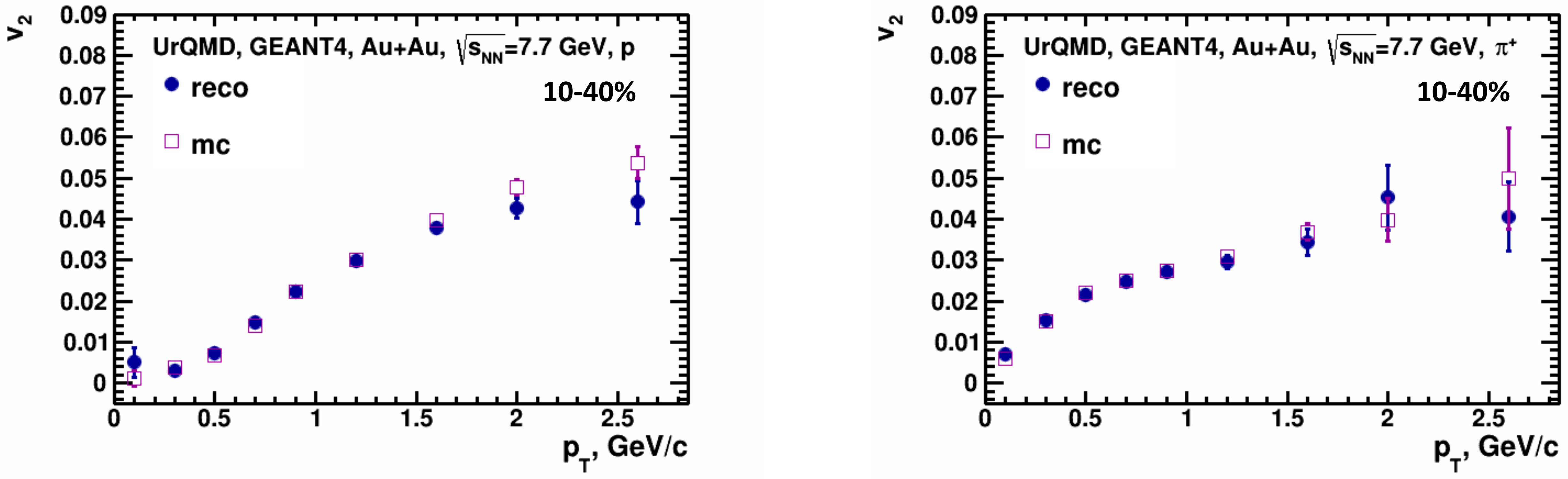
Reco vs. mc: v_1





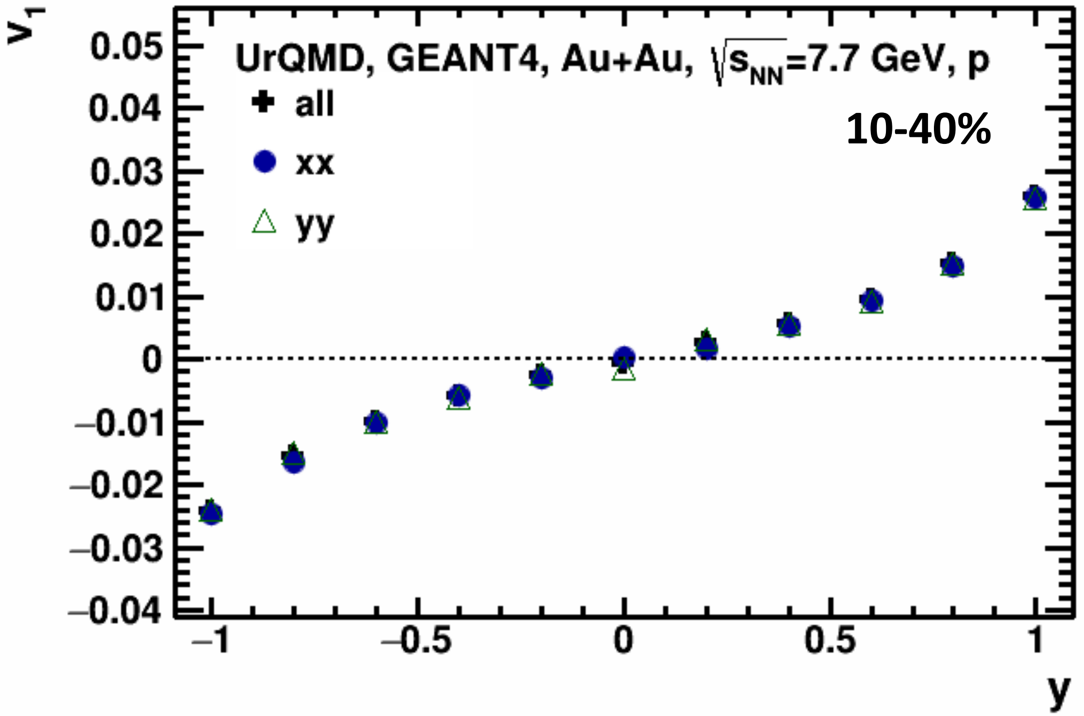
v_1 from reconstructed tracks is in agreement with v_1 from model

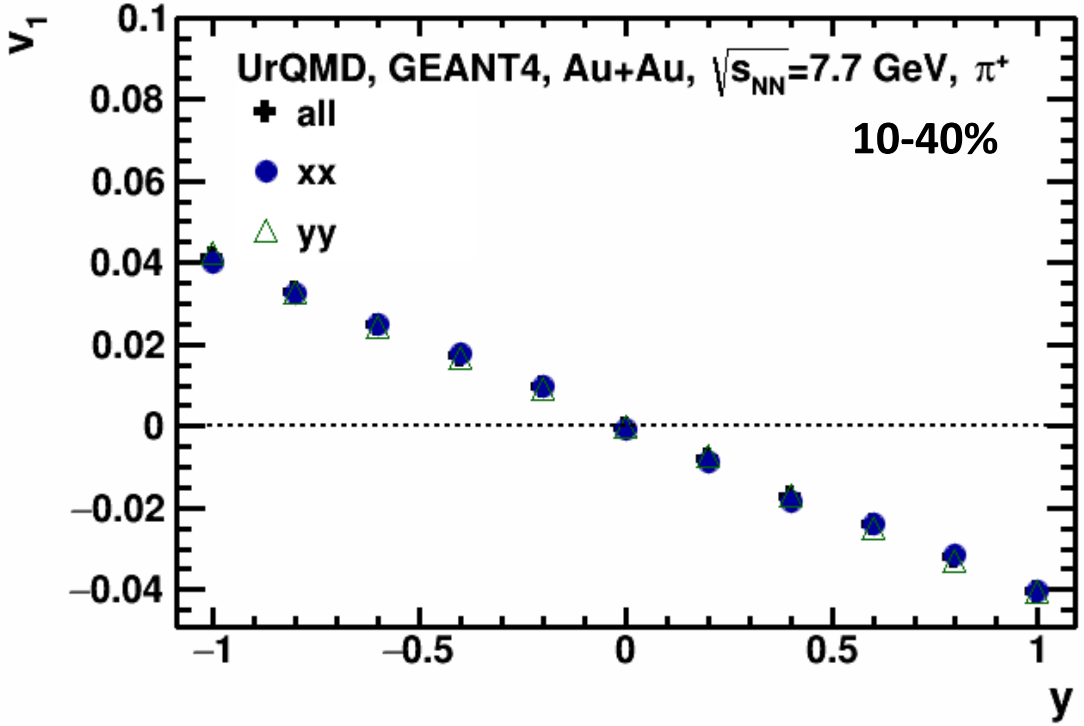
Reco vs. mc: v_2



v_2 from reconstructed tracks is in agreement with v_2 from model

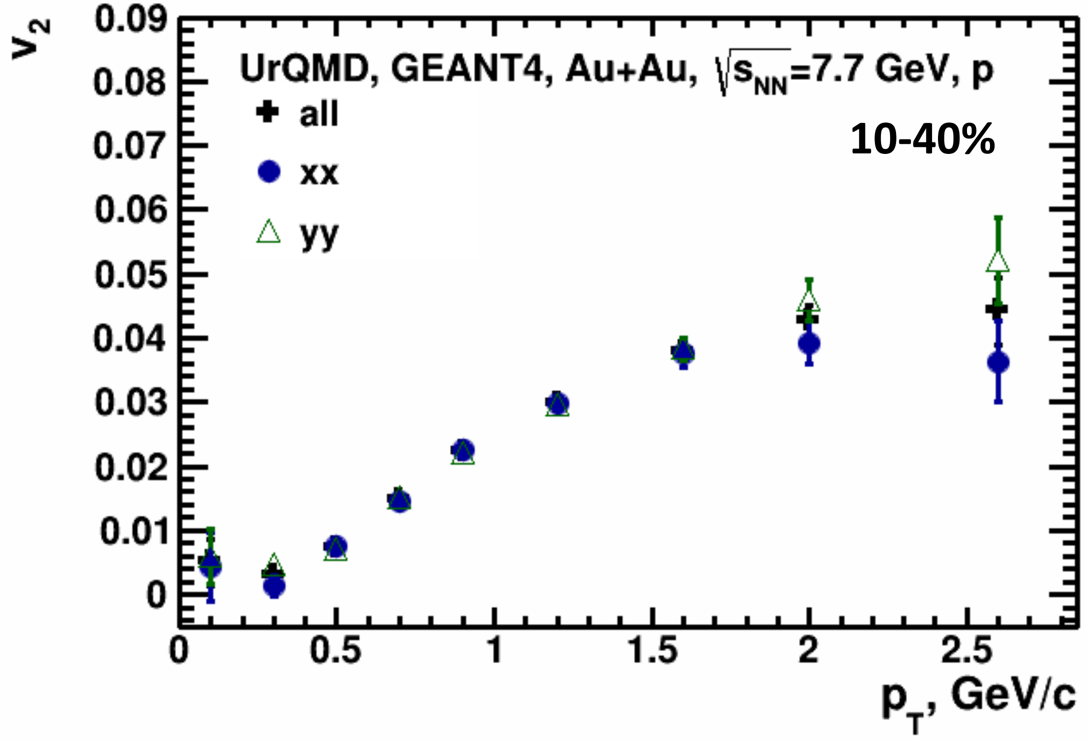
Components comparison: v_1

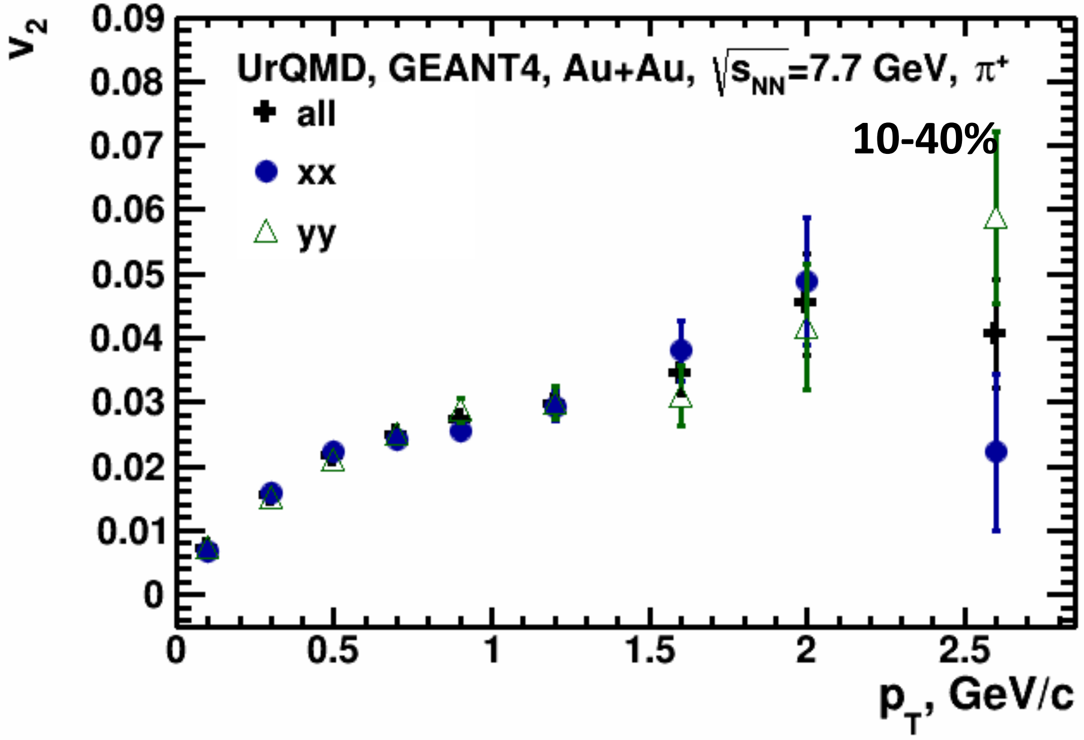




$v_{1,XX}$, $v_{1,YY}$ and $v_{1,all}$ are consistent with each other

Components comparison: v_2





$v_{2,XX}$, $v_{2,YY}$ and $v_{2,all}$ are consistent with each other

Summary and outlook

- QnAnalysis framework is ready for use in MPD experiment
  - Basic setup templates for all stages of flow measurement are available here: https://devel.mephi.ru/PEParfenov/QnAnalysisMPD_scripts
- Using acceptance filter it was shown that corrections of the Q-vector employed by the QnAnalysis framework can suppress contribution from non-uniform acceptance of the detector
- Simple validation of the results were done by comparing flow coefficients measured from reconstructed and model data
  - Both directed and elliptic flow show good agreement between reconstructed and simulated (model) data
- Flow coefficients obtained using only certain Q-vector components were compared with the averaged value
  - Both directed and elliptic flow from different components of Q-vectors show consistent results
- **ToDo**: implementation of direct cumulant method for flow measurements is in progress; more detailed study of non-uniform acceptance effects in flow measurements

Thank you for your attention!

Backup slides

AnalysisTree format for MPD data

AnalysisTree:

A framework and experimentally independent, lightweight and flexible data format that stores information in configurable basic objects:

- **EventHeader** – information about general event properties
- **Track** – reconstructed track parameters
- **Particle** – Monte Carlo track parameters
- **Module** – information about module in a module-type detector (FHCal)
- **Hit** – information about hit in a hit-type detector

Each object can contain any number of custom integer, floating or boolean fields

AnalysisTree data format:
https://github.com/HeavyIonAnalysis/AnalysisTree

Main structure of the AnalysisTree format in MPD:
(mandatory default information, added custom information)

- RecoEvent (**EventHeader**):
  - Vertex (x,y,z)
- McEvent (**EventHeader**):
  - Vertex (x,y,z)
  - Impact parameter B
  - Reaction plane PhiRP
- TpcTracks (**Track**):
  - Momentum (p_x, p_y, p_z or p_T, ϕ, η)
  - Track quality (N_{hits})
  - DCA (x,y,z)
  - PID-related information (charge, dE/dx, m^2, tof_flag, pid_probability)
- McTracks (**Particle**):
  - Momentum (p_x, p_y, p_z or p_T, ϕ, η)
  - PID-related information (pdg, y)
  - mother_id
- FHCalModules (**Module**):
  - Module information (Energy, number)
- TpcTracks->McTracks matching