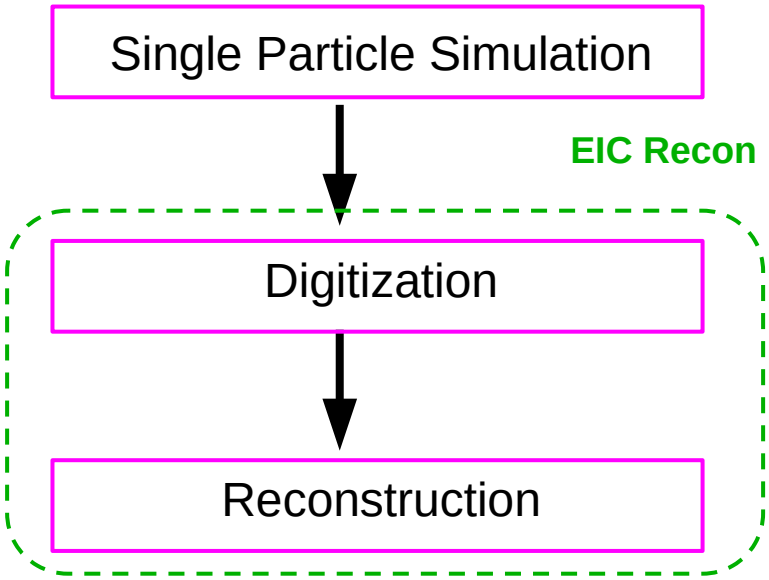
Benchmarking Tracking Performance

Shyam Kumar*, Annalisa Mastroserio, Domenico Elia
INFN Bari, Italy

Simulation and Reconstruction in ePIC experiment

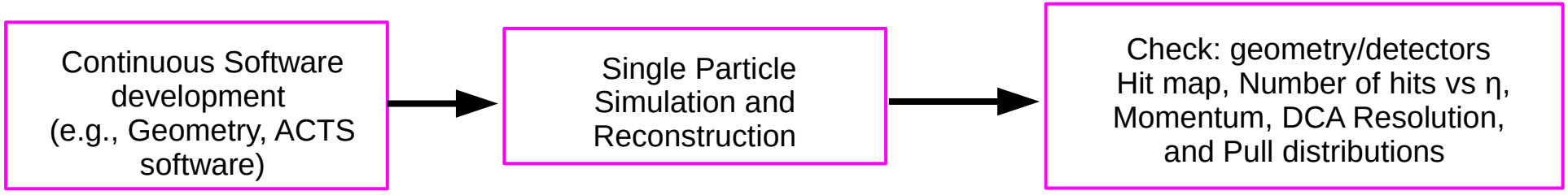


DD4HEP and GEANT4 (Particle Gun): Single particle generation and their propagation using G4ParticleGun in GEANT4

Digitization (detector response): For silicon detectors hits are smeared (Gaussian) according to spatial resolutions after applying an energy threshold

Tracking in EIC Recon using ACTS (A Common Tracking Software): Digitized hits are used for the track reconstruction in EICRecon framework using ACTS software

Why single particle tracking benchmarks?



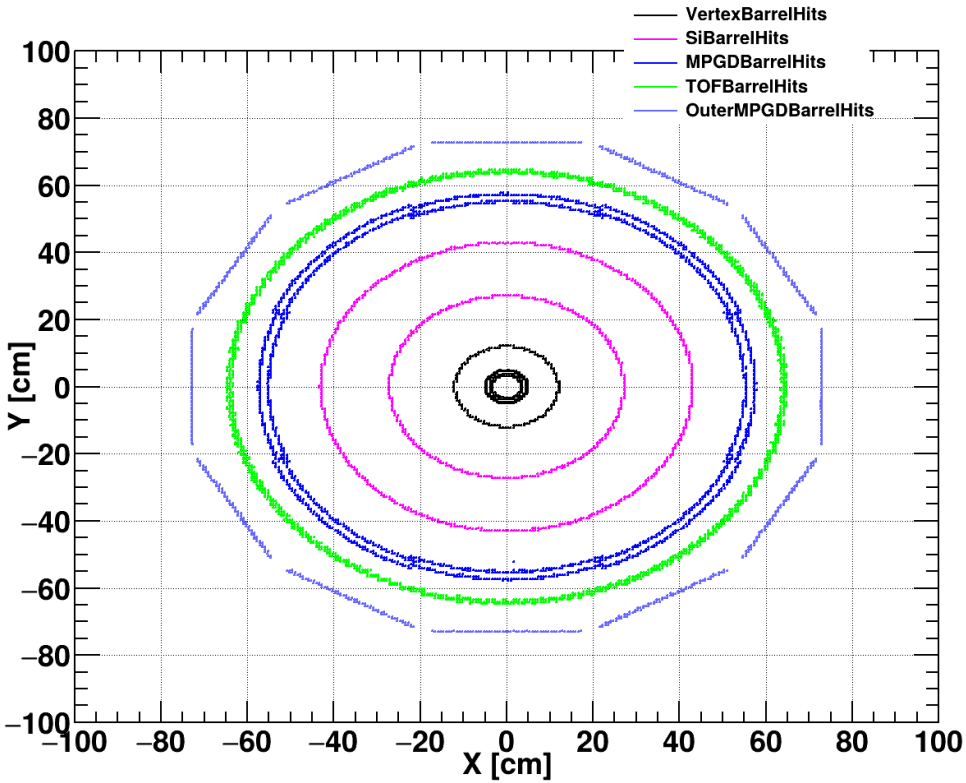
Hit Maps (Simulation Level)

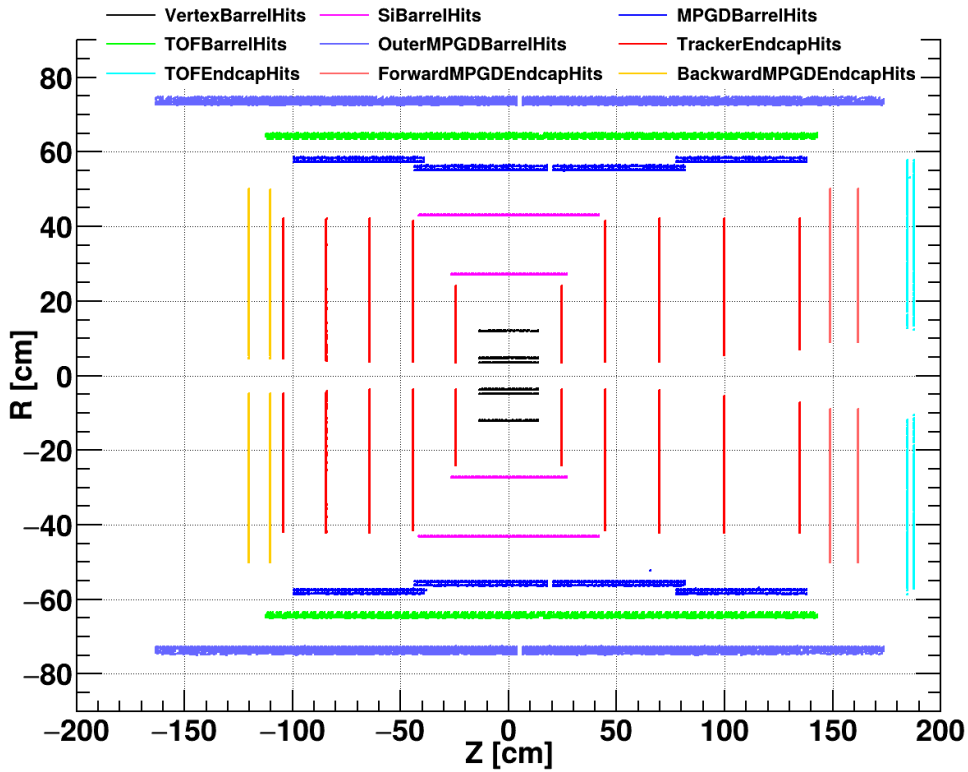
10k Events (pi-)

Geometry

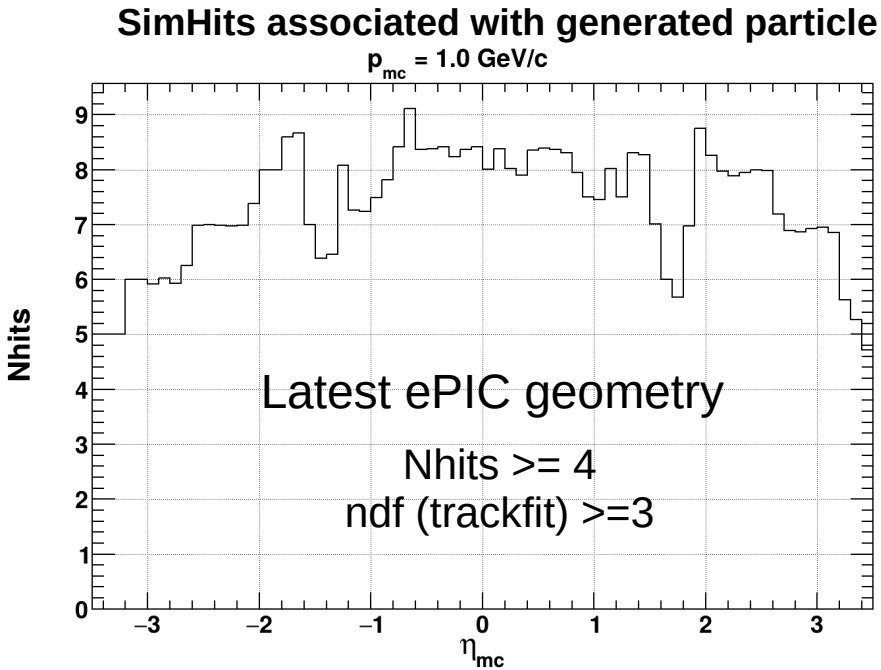
MOMENTUM=[0.5, 1.0, 2.0, 5.0, 10.0, 15.0]

SimHits associated with generated particle





Average Number of Hits vs η



Instructions

Shyam Kumar; INFN Bari, Italy; shyam055119@gmail.com Method to produce the tracking performances with ePIC tracker The scripts can be used to create the debug plots for the momentum resolutions.

To run a full simulation-reconstruction-analysis chain, start in the top-level directory of the detector_benchmarks repository (`cd ../..` relative to this one) and do:

```
snakemake -c2 results/tracking_performances/local
```

or, referencing it by the rule name:

```
snakemake --cores 1 tracking_performance_local
```

To process an individual campaign do:

```
snakemake -c2 results/tracking_performances/24.04.0
```

It will produce the results with truth/realistic seeding.

If Nhits = 3 results are biased because of ndf = 1

Similar plots for other momentum ranges are produced

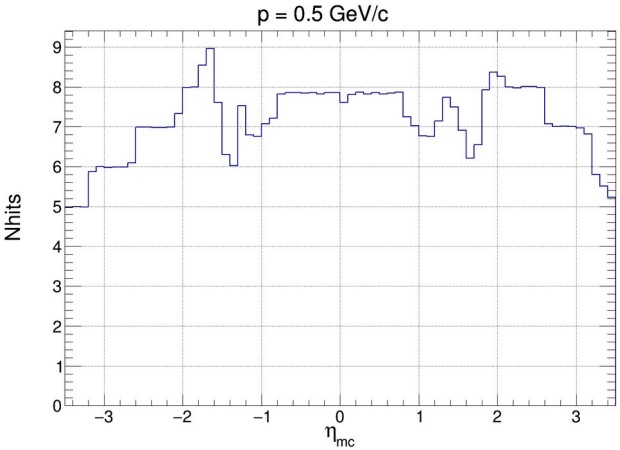
My Slides from Tracking Meeting

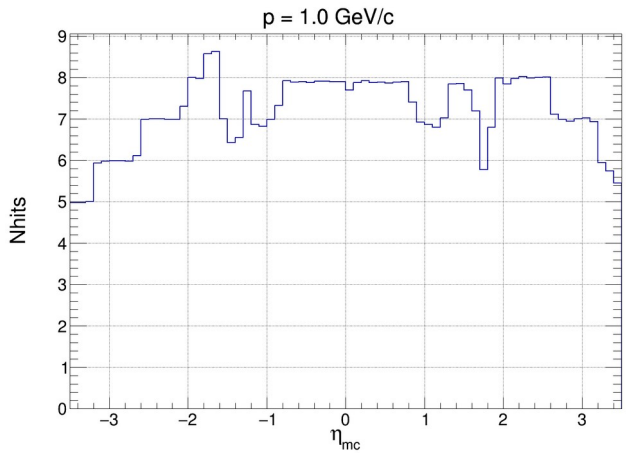
Nhits vs Eta

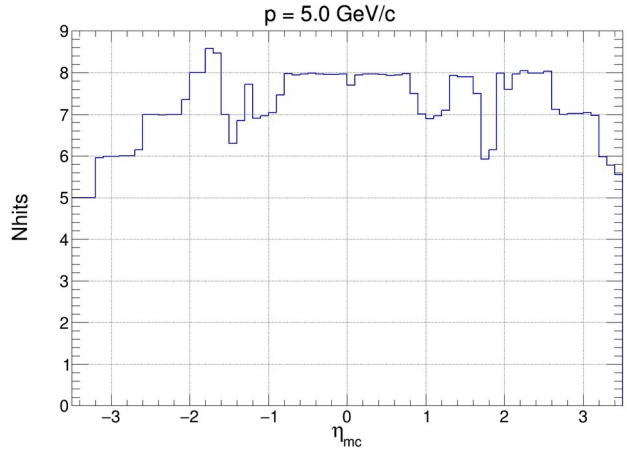
ePIC: 24.04.0

Nhits>=5 as reported

Previous Studies



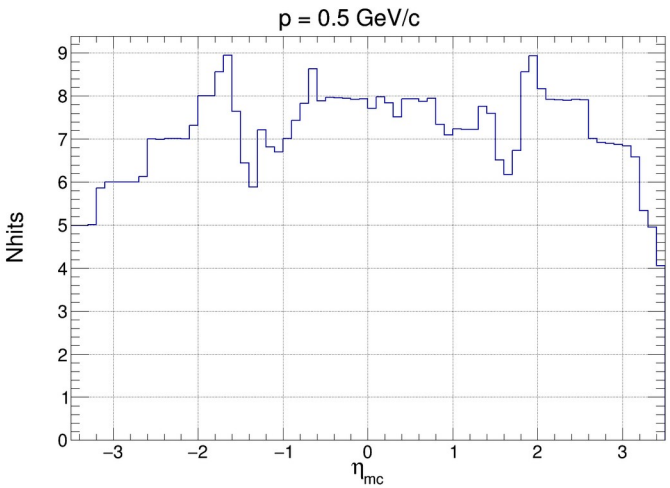


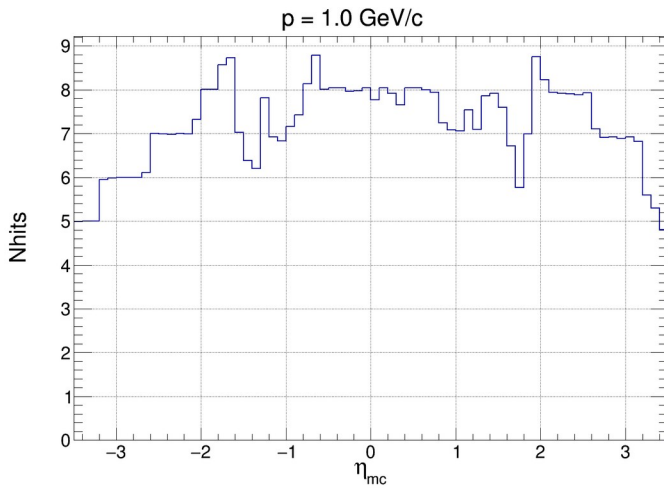


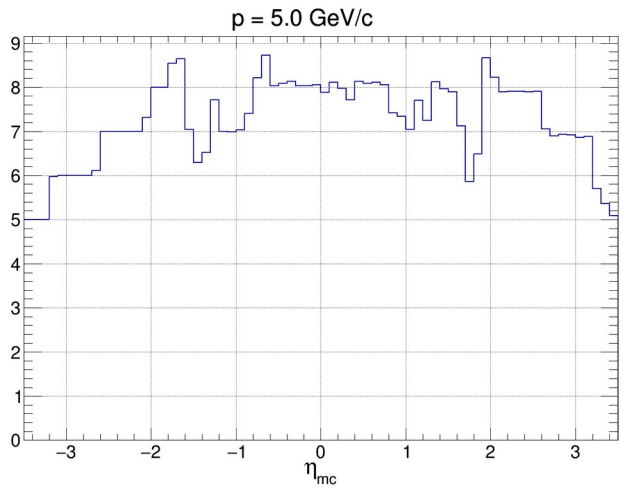
ePIC: 24.12.0

Nhits>=4

Forward beampipe radius is increased

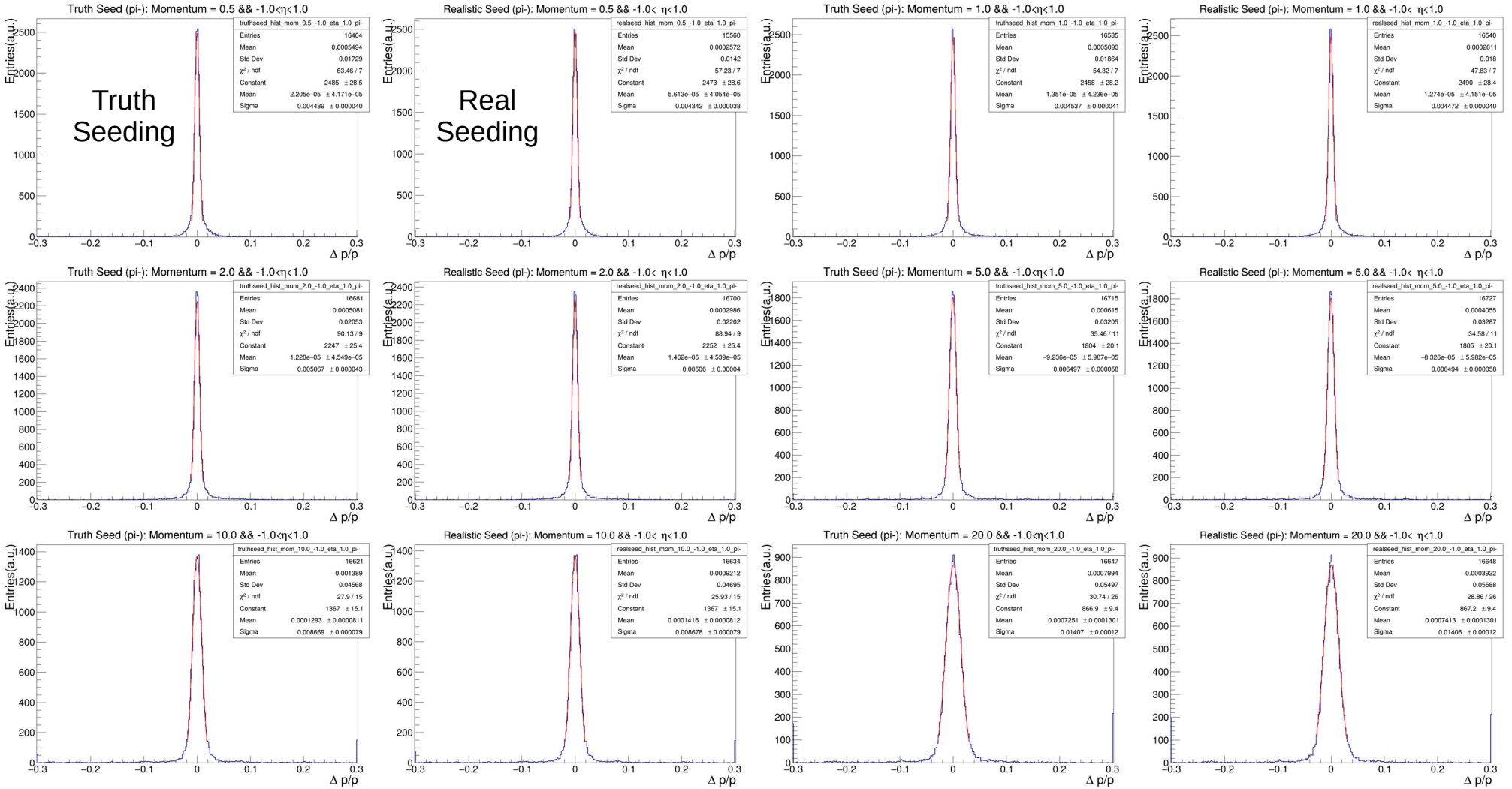






Momentum Resolutions

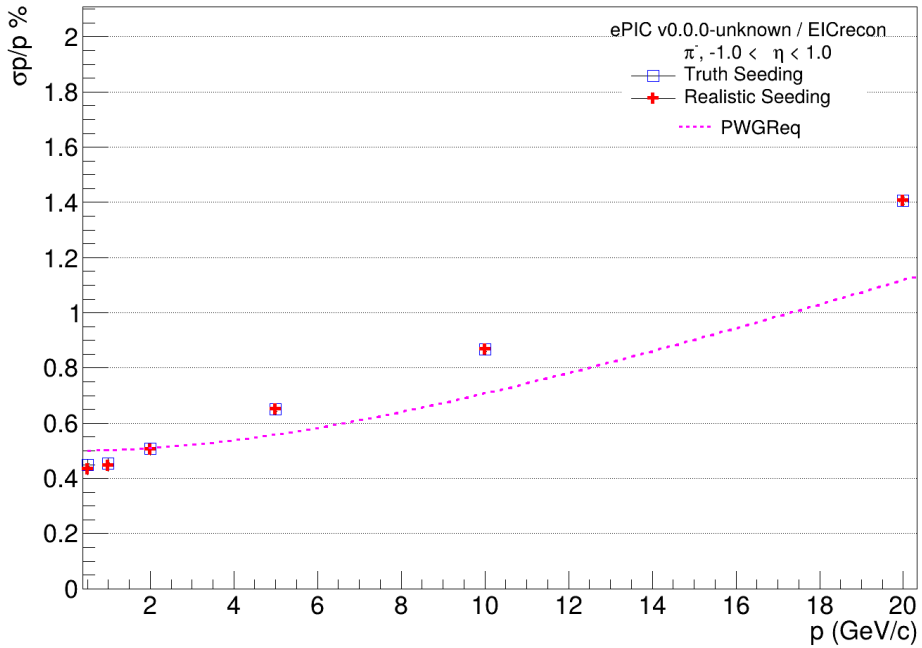
The plots are important to crosscheck the material map and performing ACTS upgrade

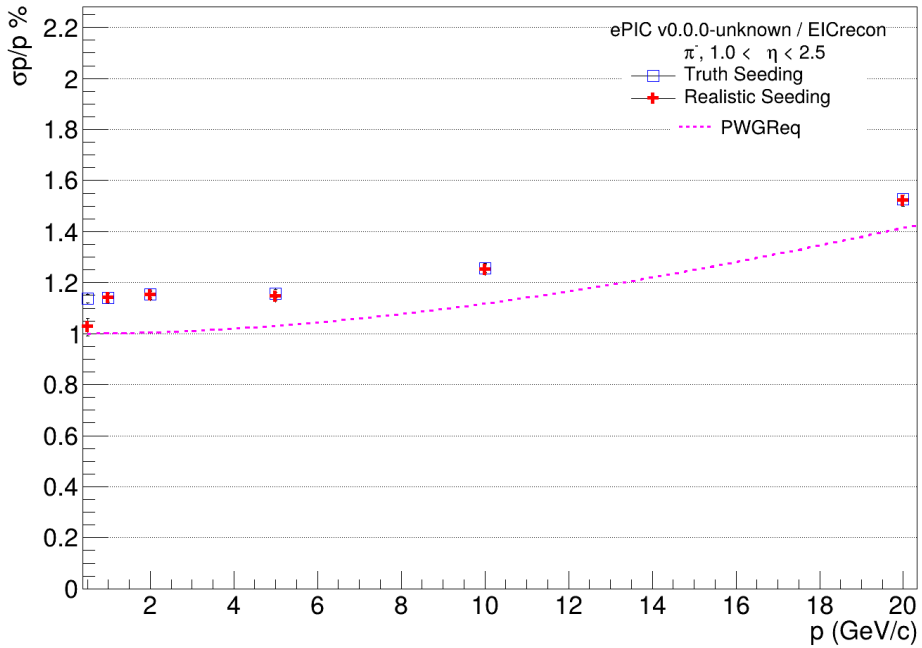


Momentum Resolutions

Latest ePIC geometry

The plots are important to crosscheck the material map and performing ACTS upgrade

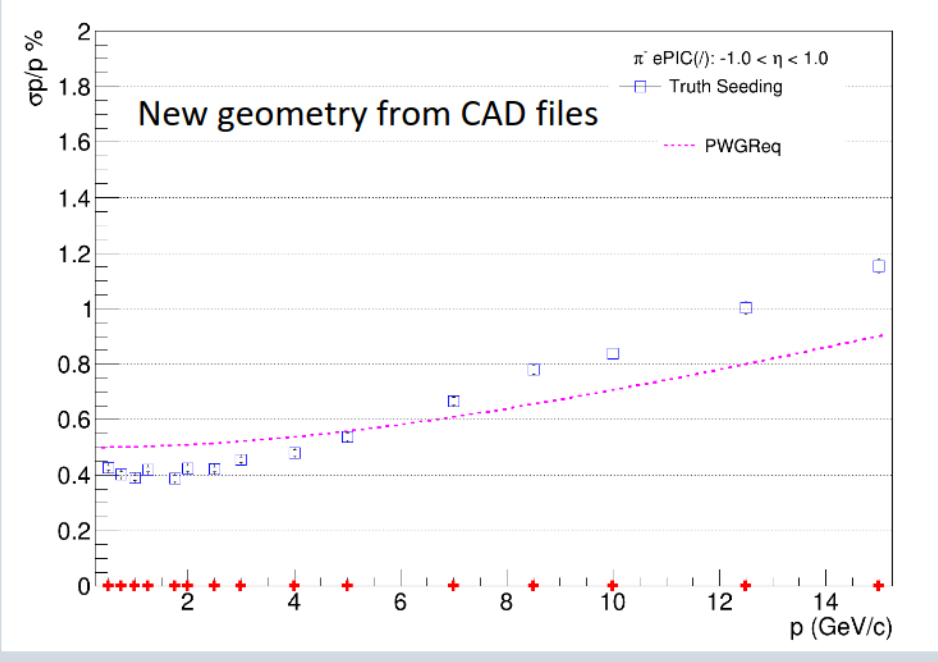


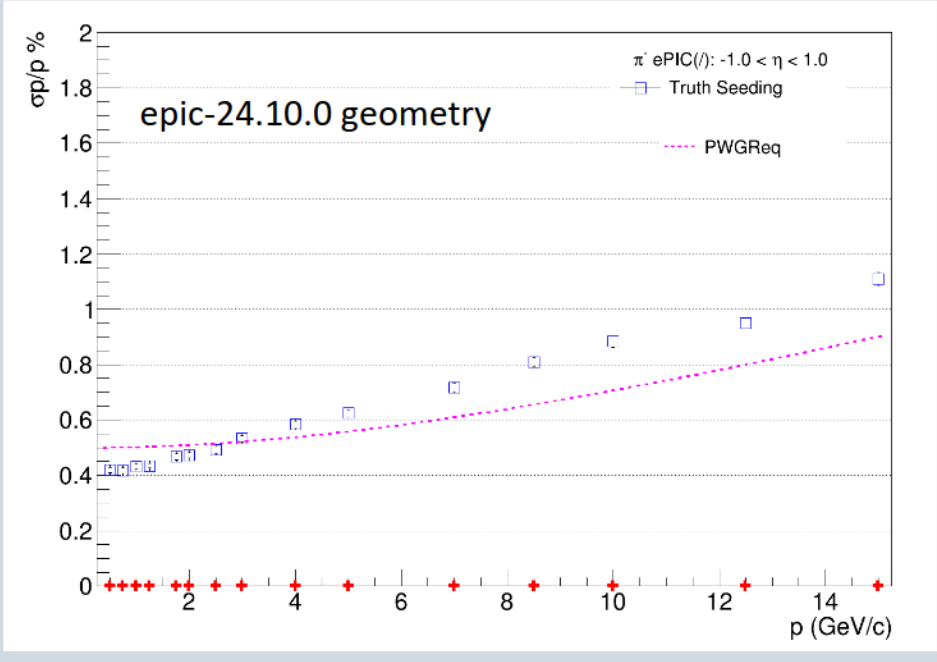


Slides by Sam Henry, University of Oxford

Tracking momentum resolution

Using tracking performance benchmark script by Shyam Kumar, epic_craterlake_tracking_only.xml
https://github.com/eic/detector_benchmarks/tree/master/benchmarks/tracking_performances





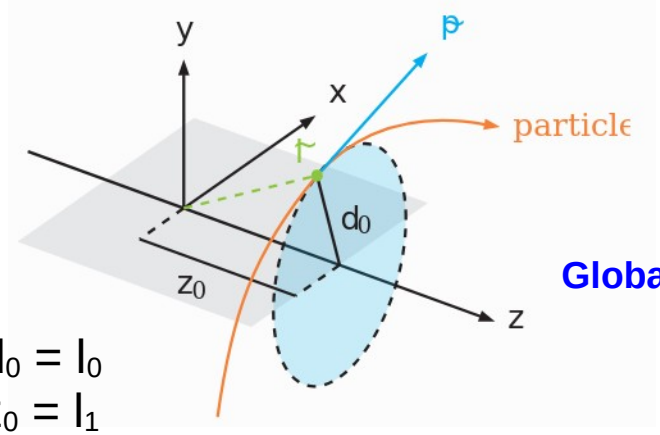
Run time: 92±25 minutes

Run time: 10±3 minutes

Running ddsim and eicrecon on 10,000 events

Track Parameters in ACTS

Track Parameters $(l_0, l_1, \phi, \theta, q/p, time)$ If tracking algorithm is working fine Pull must be consistent with unity



$d_0 = l_0$

$z_0 = l_1$

At Point of closest approach

$$(l_0, l_1, \phi, \theta, q/p)$$

Global (Lab frame)

$$(x, y, z, p_x, p_y, p_z, q)$$

$$x = -l_0 \sin\phi, \quad y = l_0 \cos\phi, \quad z = l_1$$

$$p_x = p\cos\phi\sin\theta, \quad p_y = p\sin\phi\sin\theta, \quad p_z = p\cos\theta$$

$$charge = sign(q/p)$$

Plan to add it to the benchmarks

$$\text{Pull } l_0 = \frac{(l_{0\,rec} - l_{0\,gen})}{\sigma_{l0}} \qquad \text{Pull } \phi = \frac{(\phi_{rec} - \phi_{gen})}{\sigma_{\phi}} \qquad \text{Pull } q/p = \frac{(q/p_{rec} - q/p_{gen})}{\sigma_{q/p}}$$

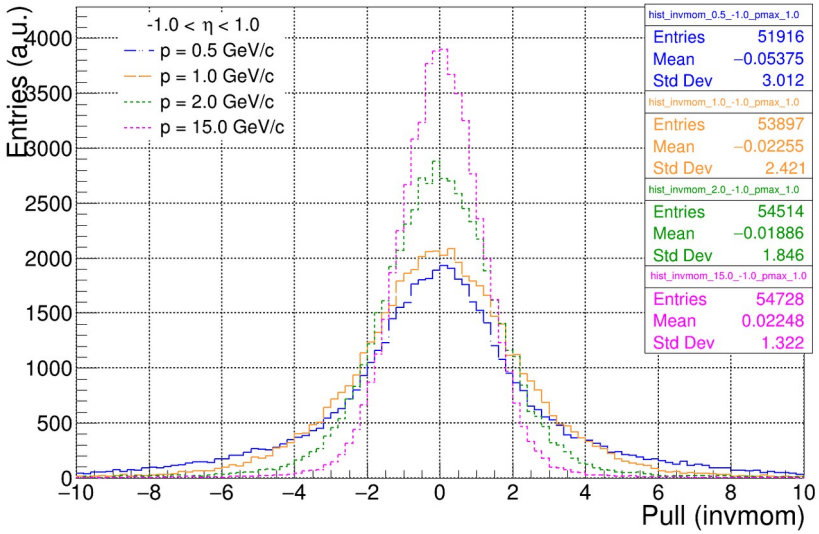
$$\text{Pull } l_1 = \frac{(l_{1\,rec} - l_{1\,gen})}{\sigma_{l1}} \qquad \text{Pull } \theta = \frac{(\theta_{rec} - \theta_{gen})}{\sigma_{\theta}} \qquad \text{Pull } p = \frac{(p_{rec} - p_{gen})}{\sigma_{p}}$$

Simulation of 200K pi- for momentum 0.5, 1.0, 2,0, 15.0 GeV/c

ePIC: 24.12.0

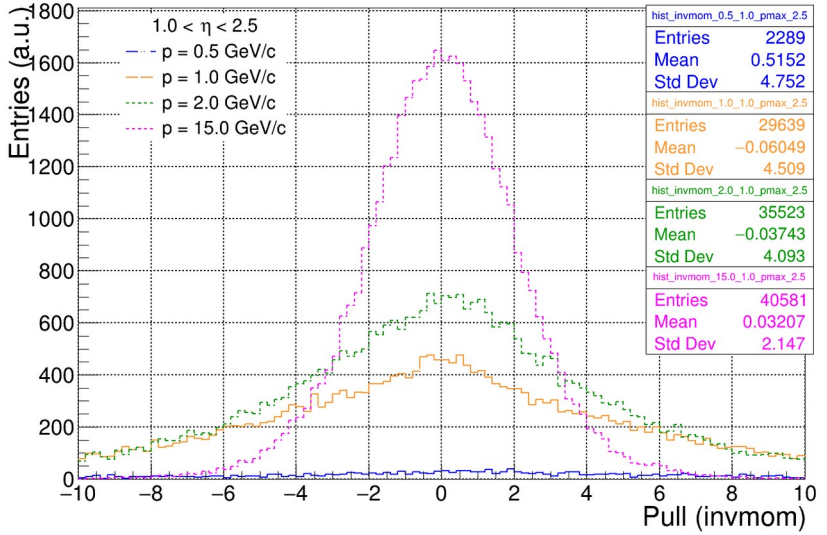
EICRecon: v1.20.0

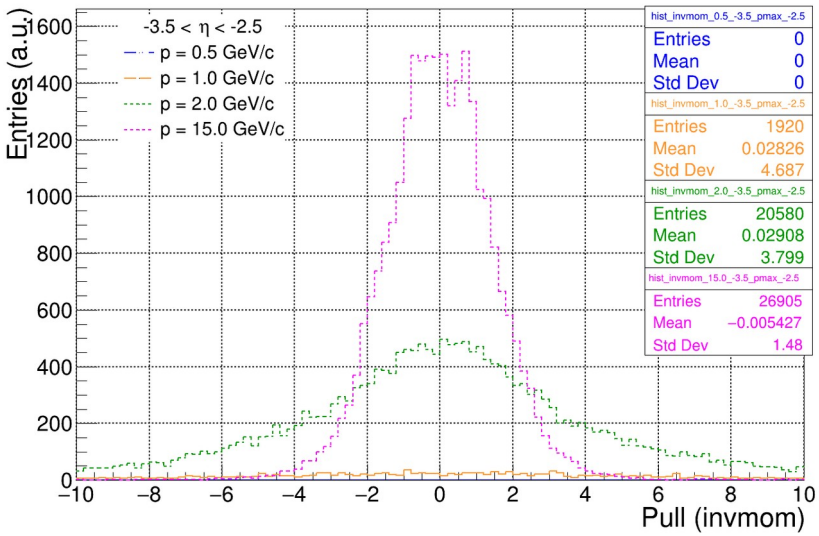
Pull distributions (Inverse Momentum)



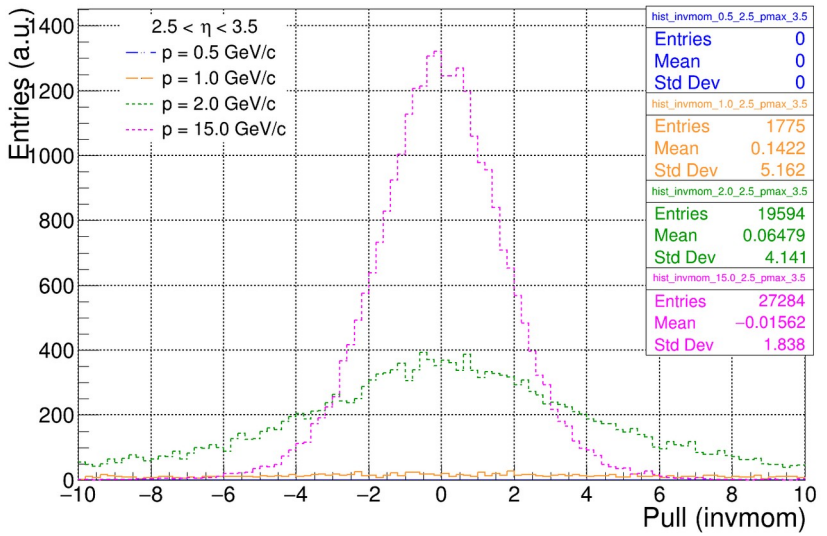
ePIC: 24.12.0

EICRecon: v1.20.0

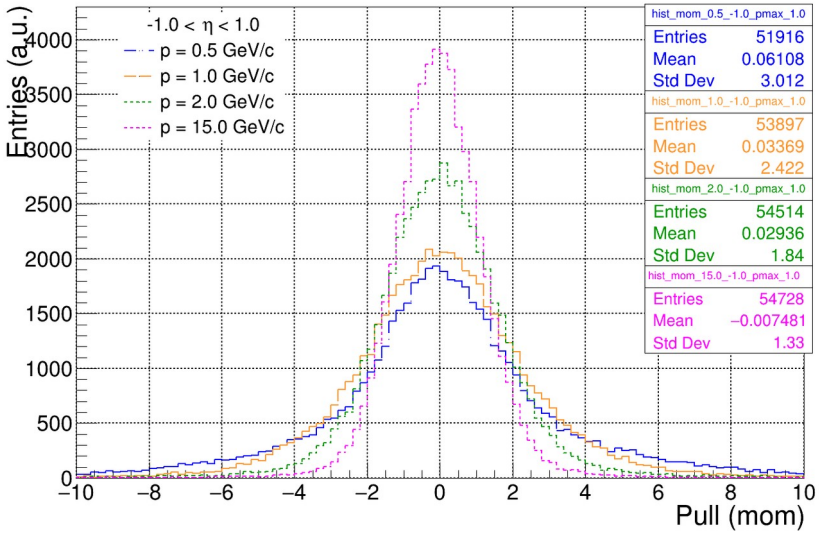




Inverse Momentum = 1/p

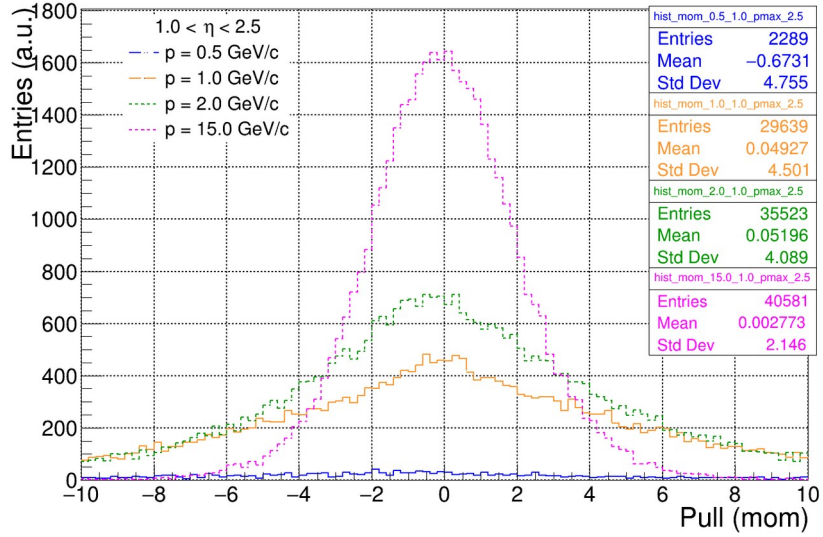


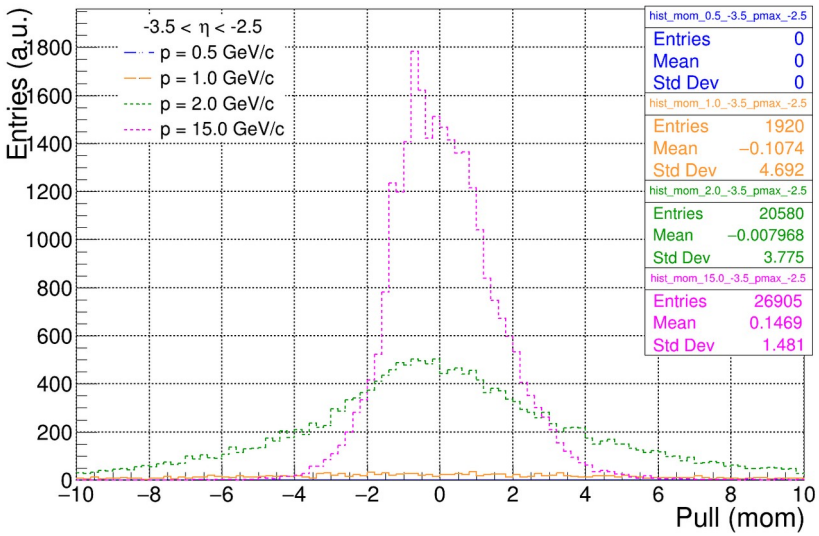
Pull distributions (Momentum)

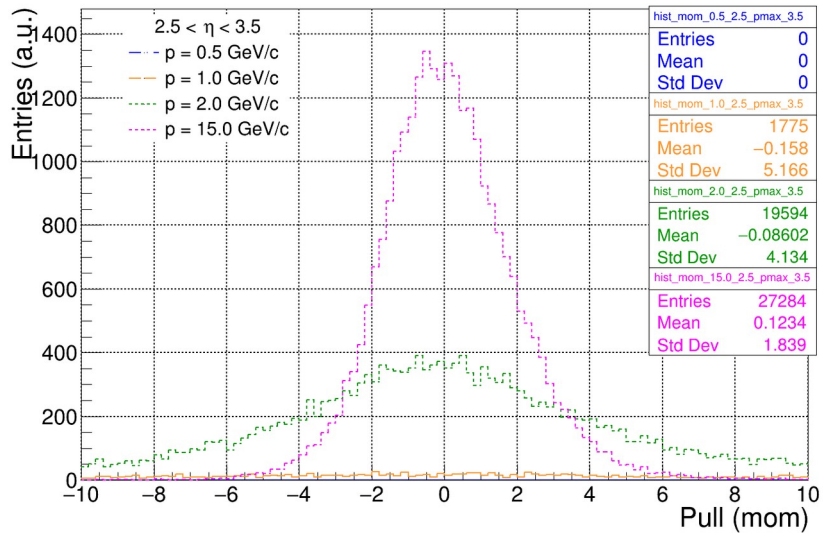


ePIC: 24.12.0

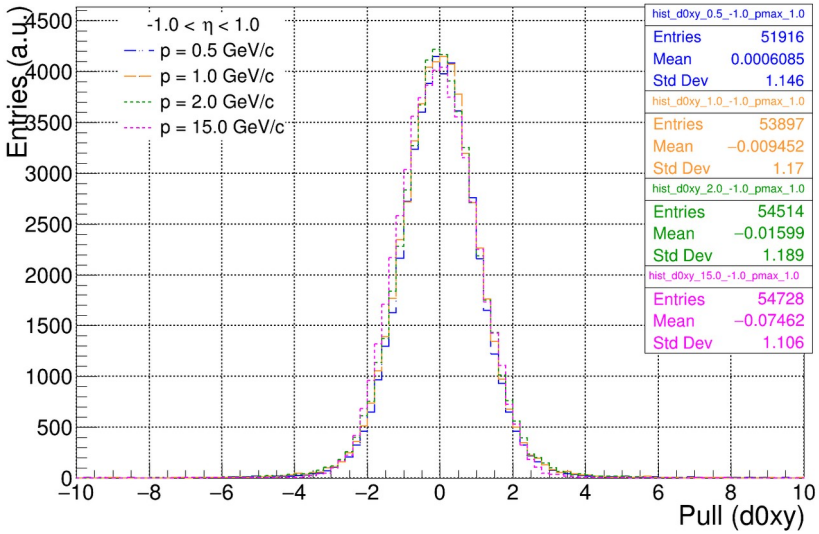
EICRecon: v1.20.0

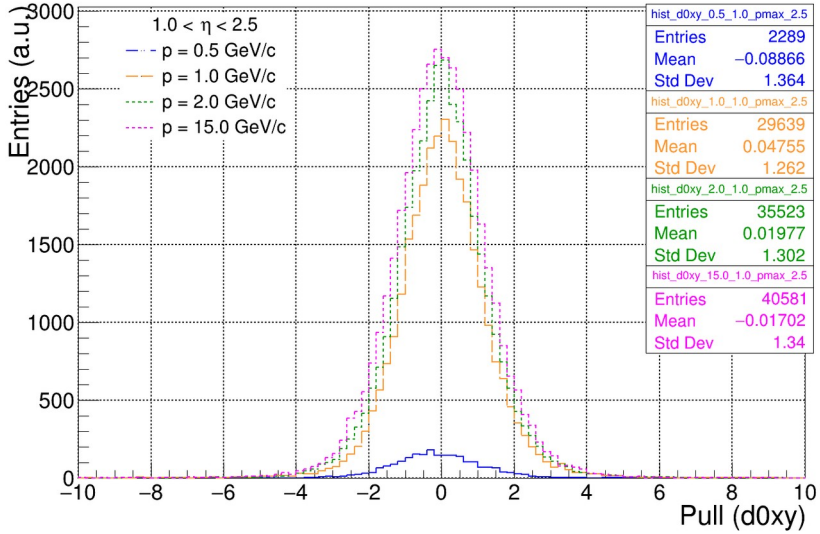


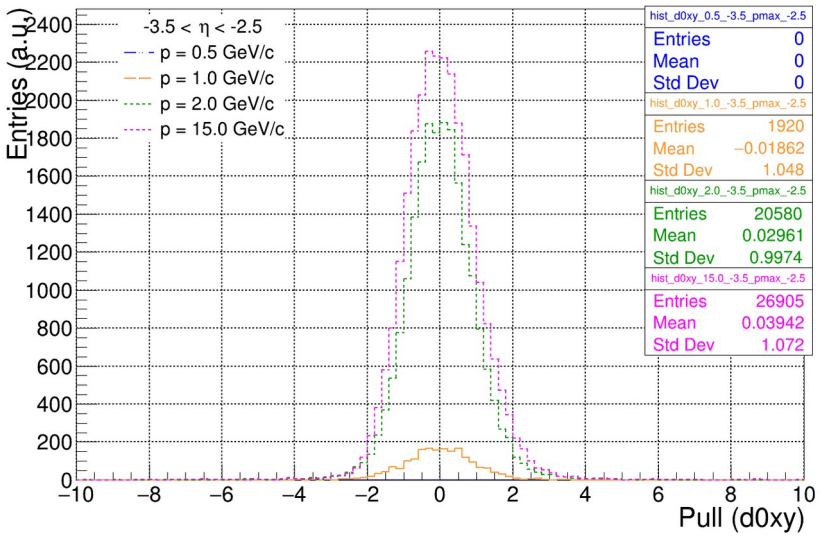


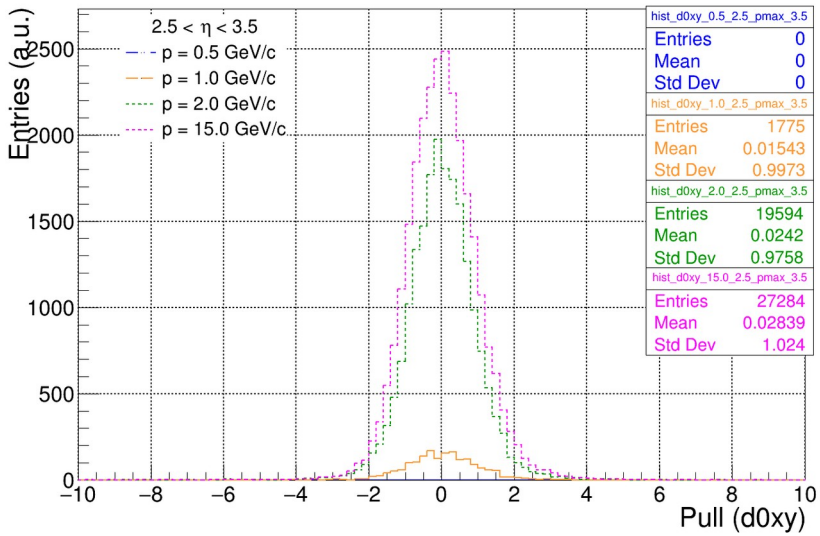


Pull distributions (l_0 or d_{0xy})

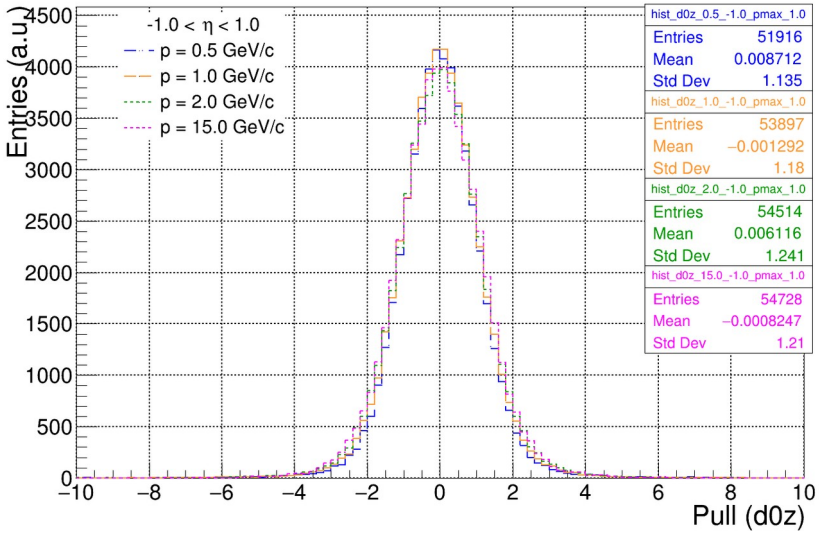


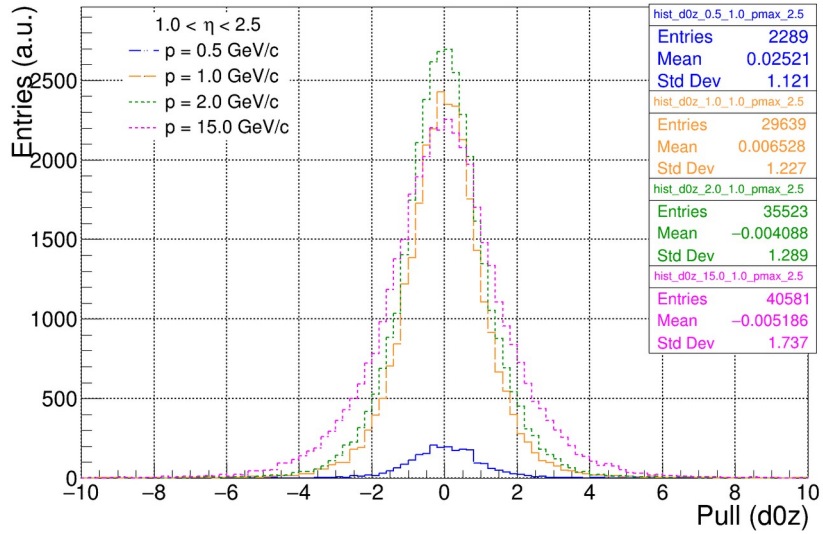


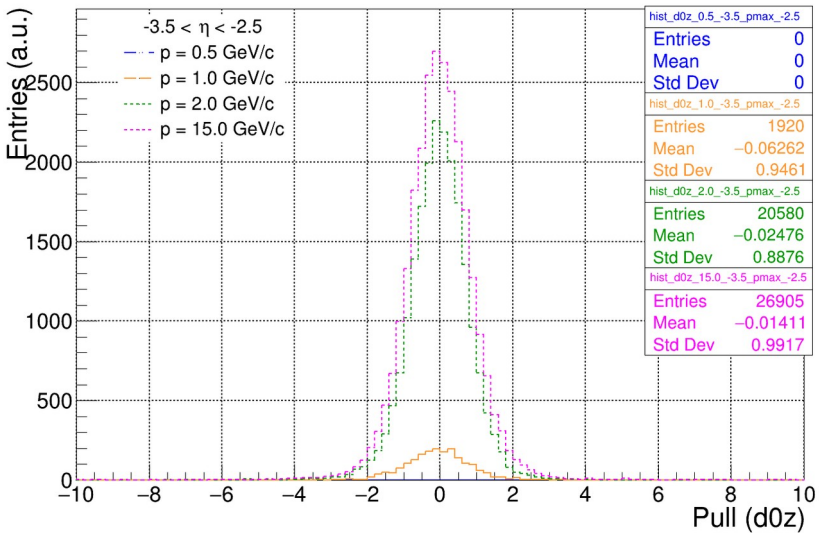


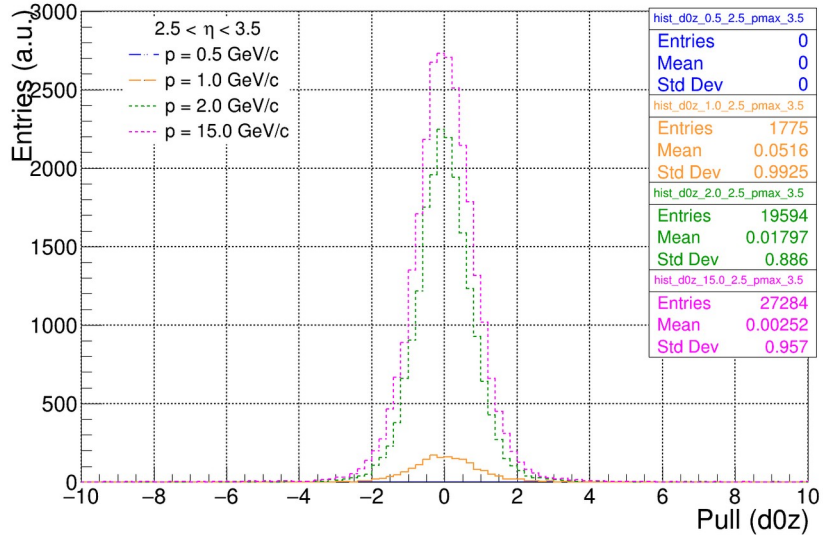


Pull distributions (l_1 or d_{0z})

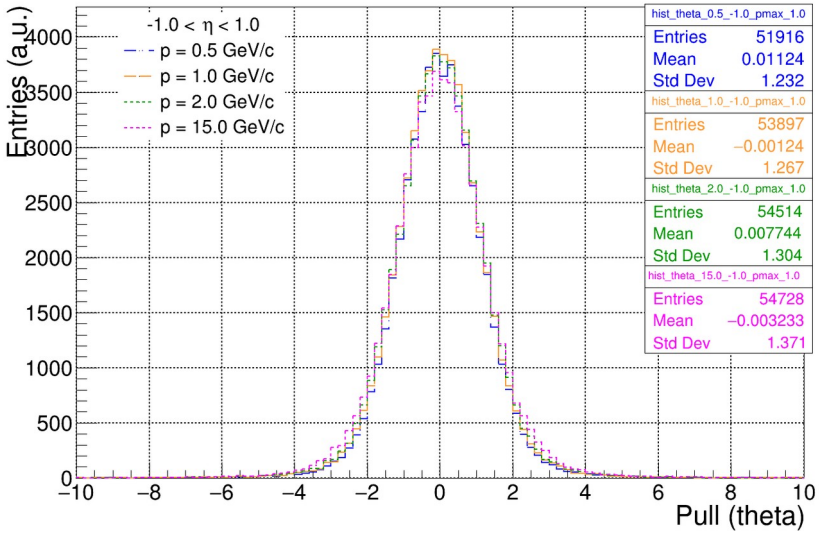


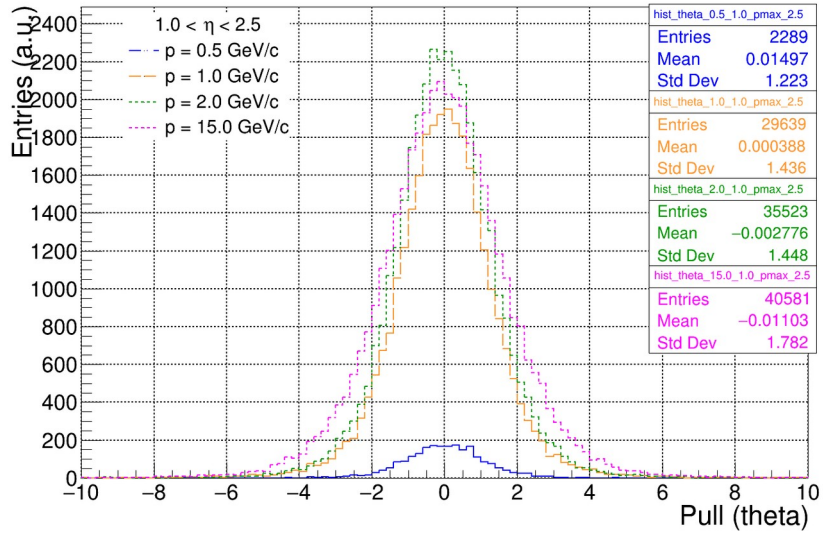


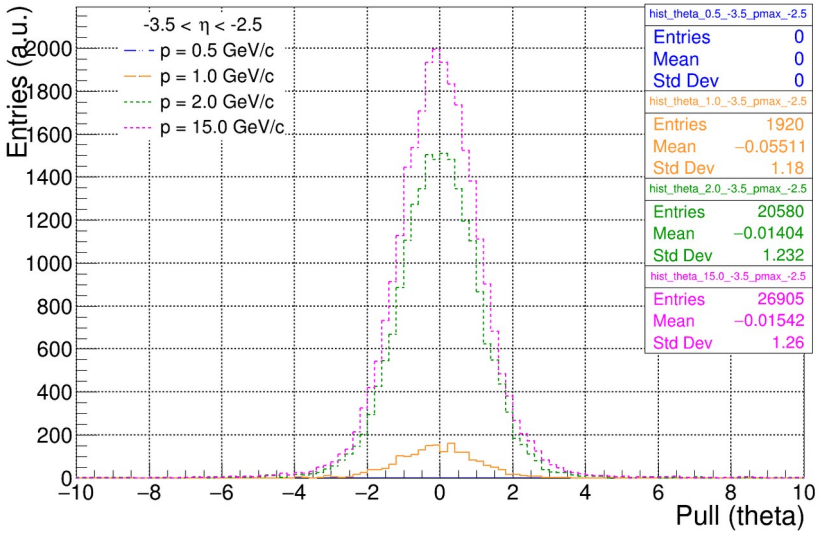


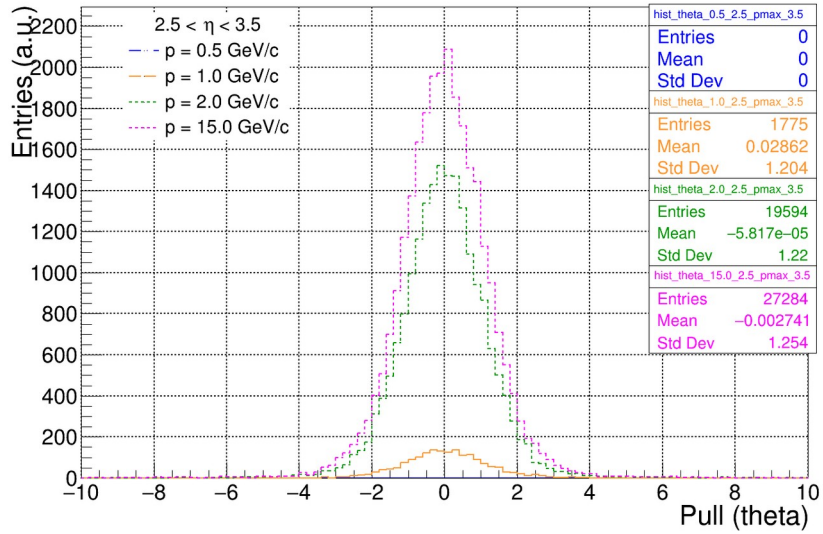


Pull distributions (Theta)

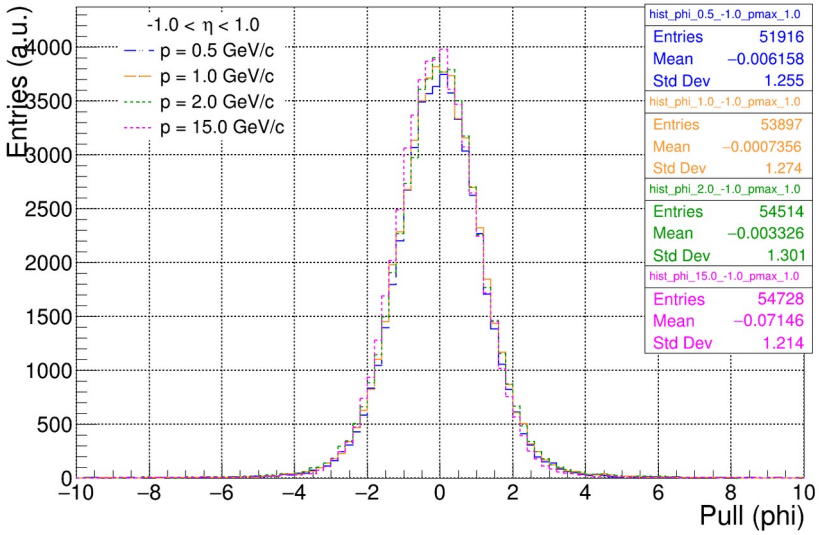


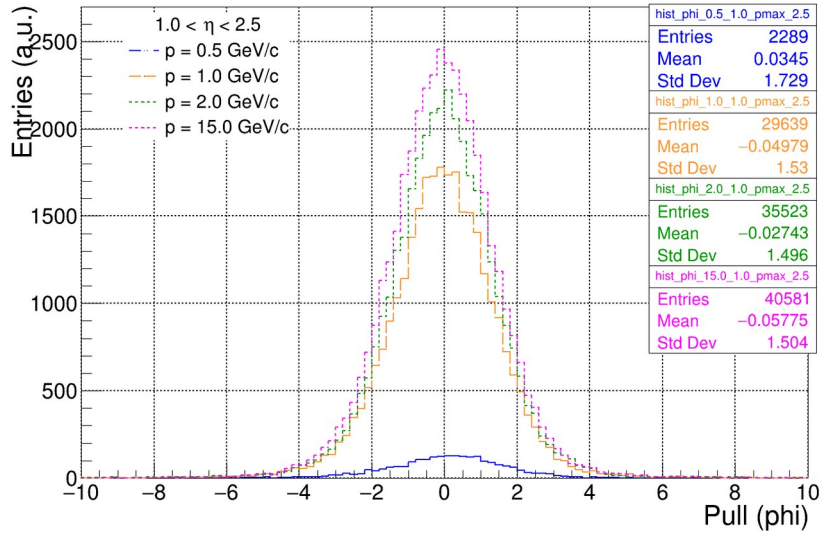


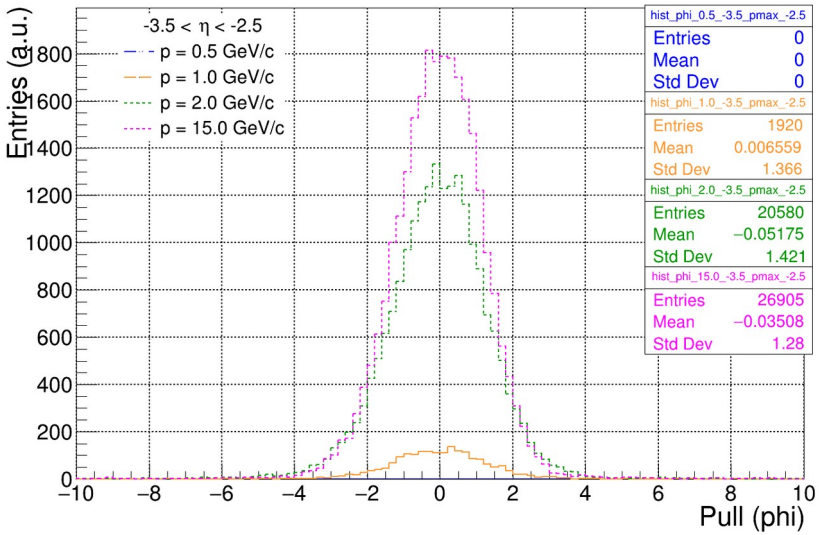


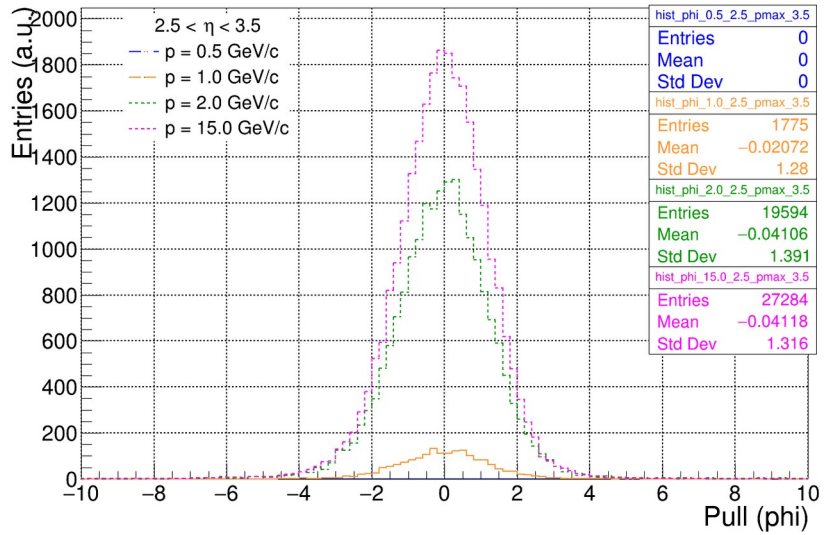


Pull distributions (Phi)









Summary and Future Plan

- Tracking Performance is included to the benchmarks
- Included hit maps and the average number of hits per track at the simulation level in the benchmarks
- I also checked the pull distributions of the track parameters
- Future Plan
  - Include hit maps and average number of hits per track for each momentum
  - Need to understand why pull distributions for momentum are not consistent with unity especially for higher η
  - Implement pull distributions to the benchmarks

Thank you !!!

Track Parametrization (Local to Global)

Helical Track model: $(l_0, l_1, \phi, \theta, q/p)$

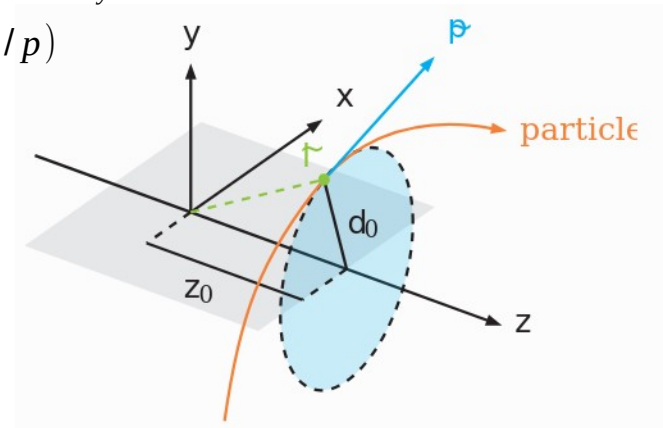
$$x = -l_0 \sin\phi, \quad y = l_0 \cos\phi, \quad z = l_1$$

$$p_x = p\cos\phi\sin\theta, \quad p_y = p\sin\phi\sin\theta, \quad p_z = p\cos\theta$$

$$charge = sign(q/p)$$

$d_0 = l_0$

$z_0 = l_1$



At Point of closest approach (perigee surface)

$(l_0, l_1, \phi, \theta, q/p)$

↓ **Global (Lab frame)**

$(x, y, z, p_x, p_y, p_z, q)$

```
Vector3 LineSurface::localToGlobal(const GeometryContext& gctx, const Vector2&
lposition, const Vector3& direction) const

{

 Vector3 unitZ0 = lineDirection(gctx);

// get the vector perpendicular to the momentum direction and the straw axis

Vector3 radiusAxisGlobal = unitZ0.cross(direction);

Vector3 locZinGlobal = transform(gctx) * Vector3(0., 0., lposition[1]);

// add loc0 * radiusAxis

return Vector3(locZinGlobal + lposition[0] * radiusAxisGlobal.normalized());

}
```

Calculation

UnitZ0: is (0,0,1) vector along the z-axis for cylinder and disks.

direction: (p Cos(phi) Sin(theta), p Sin(phi) Sin(theta), p Cos(theta))

radiusAxisGlobal = UnitZ0 Cross product direction = (-p Sin(phi) Sin(theta), p Cos(phi) Sin(theta), 0)

radiusAxisGlobal.Normalized = (-Sin(phi), Cos(phi), 0)locZinGlobal = (0,0,l1) (is same as global)

Global position = locZinGlobal + lposition[0] * radiusAxisGlobal.normalized() = (0,0,l1) + l0(-Sin(phi), Cos(phi), 0)Global Position = (-l0 Sin(phi), l0 Cos(phi), l1)

Returns the components, which we are using in HF analysis.

$x = -l_0 \sin\phi$, $y = l_0 \cos\phi$, $z = l_1$

Covariance Matrix in ACTS

Track Parameters $(l_0, l_1, \phi, \theta, q/p, time)$

For each fitted track we get track parameters and covariance matrix

$$\begin{array}{c} \\ l_0 \\ l_1 \\ \phi \\ \theta \\ q/p \\ time \end{array} \begin{array}{c} \begin{array}{cccccc} l_0 & l_1 & \phi & \theta & q/p & time \end{array} \\ \begin{bmatrix} \sigma^2(l_0) & \mathrm{cov}(l_0,l_1) & \mathrm{cov}(l_0,\phi) & \mathrm{cov}(l_0,\theta) & \mathrm{cov}(l_0,q/p) & \mathrm{cov}(l_0,t) \\ . & \sigma^2(l_1) & \mathrm{cov}(l_1,\phi) & \mathrm{cov}(l_1,\theta) & \mathrm{cov}(l_1,q/p) & \mathrm{cov}(l_1,t) \\ . & . & \sigma^2(\phi) & \mathrm{cov}(\phi,\theta) & \mathrm{cov}(\phi,q/p) & \mathrm{cov}(\phi,t) \\ . & . & . & \sigma^2(\theta) & \mathrm{cov}(\theta,q/p) & \mathrm{cov}(\theta,t) \\ . & . & . & . & \sigma^2(q/p) & \mathrm{cov}(q/p,t) \\ . & . & . & . & . & \sigma^2(t) \end{bmatrix} \end{array}$$

Symmetric matrix: Independent entries = n(n+1)/2 = 6*7/2 = 21

```
Processing ReadCovarianceArray_new.C...
Event 0, number of tracks: 8
  Track 0 covariance:
    cov[0] = 0.0104456
    cov[1] = 2.366e-06
    cov[2] = 0.0103324
    cov[3] = -0.000289634
    cov[4] = 7.52669e-07
    cov[5] = 8.04972e-06
    cov[6] = -8.12589e-07
    cov[7] = 0.000284166
    cov[8] = 4.50984e-08
    cov[9] = 7.82997e-06
    cov[10] = 0.000153944
    cov[11] = 7.02629e-07
    cov[12] = -4.94218e-06
    cov[13] = 5.14138e-09
    cov[14] = 0.00011838
    cov[15] = -1.41347e-06
    cov[16] = -3.20263e-06
    cov[17] = 3.89044e-08
    cov[18] = -8.82041e-08
    cov[19] = 2.242e-09
    cov[20] = 0.000333566
```

Cov[0] = cov(l0, l0)
Cov[1] = cov(l0, l1)
Cov[2] = cov(l1, l1)
Cov[3] = cov(l0, φ)
cov[4] = cov(l1, φ)
cov[5] = cov(φ, φ)
cov[6] = cov(l0, θ)
cov[7] = cov(l1, θ)
cov[8] = cov(φ, θ)
cov[9] = cov(θ, θ)
cov[10] = cov(l0, q/p)

Cov[11] = cov(l1, q/p)
Cov[12] = cov(φ, q/p)
Cov[13] = cov(θ, q/p)
Cov[14] = cov(q/p, q/p)
Cov[15] = cov(l0, time)
Cov[16] = cov(l1, time)
Cov[17] = cov(φ, time)
Cov[18] = cov(θ, time)
Cov[19] = cov(q/p, time)
Cov[20] = cov(time, time)

$\sigma_{l0} = \sqrt{(Cov[0])}$ $\sigma_{l1} = \sqrt{(Cov[2])}$

$\sigma_{\phi} = \sqrt{(Cov[5])}$ $\sigma_{\theta} = \sqrt{(Cov[9])}$

$\sigma_{q/p} = \sqrt{(Cov[14])}$