AstroPix Clustering

(https://github.com/eic/EICrecon/pull/1999)

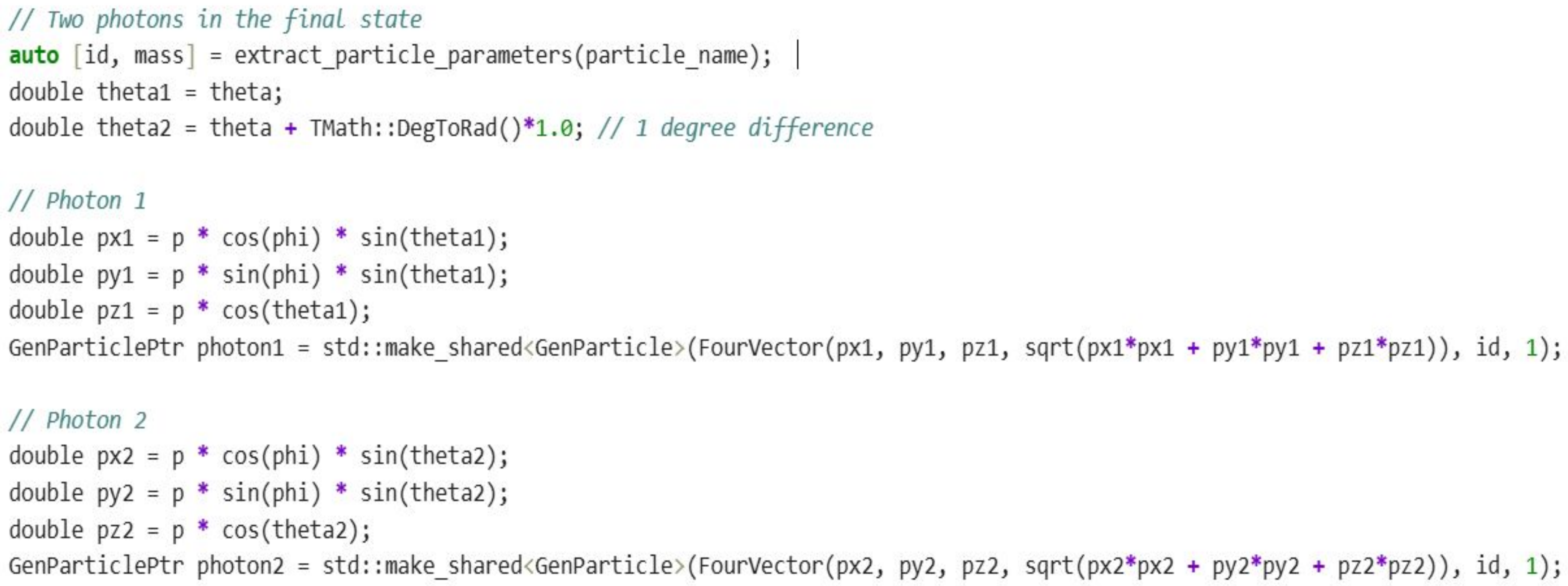
Akshaya Vijay
Dr Wouter Deconinck
University of Manitoba

August 26,2025

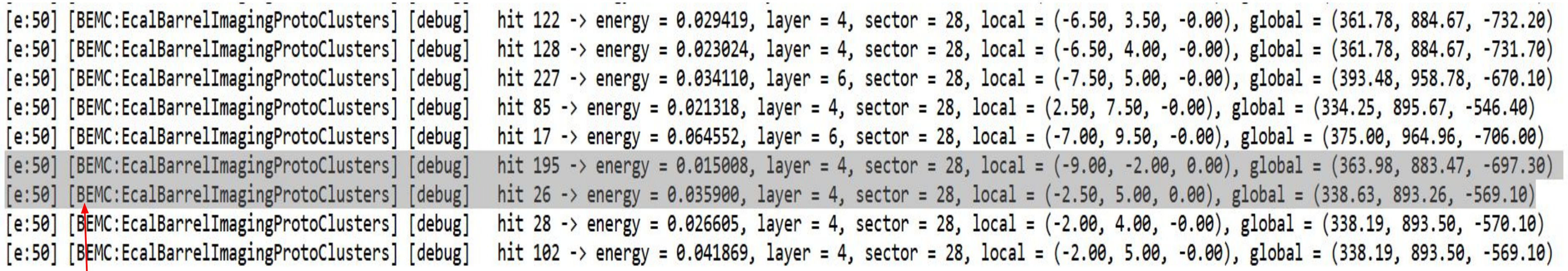
```cpp
// layer check
int ldiff = std::abs(h1.getLayer() - h2.getLayer());
// same layer, check local positions
if (ldiff == 0) {
  return (std::abs(h1.getLocal().x - h2.getLocal().x) <= localDistXY[0]) &&
         (std::abs(h1.getLocal().y - h2.getLocal().y) <= localDistXY[1]);
} else if (ldiff <= m_cfg.neighbourLayersRange) {
  switch (m_cfg.layerMode) {
  case eicrecon::ImagingTopoClusterConfig::ELayerMode::etaphi:
    return (std::abs(edm4hep::utils::eta(h1.getPosition()) -
                     edm4hep::utils::eta(h2.getPosition())) <= layerDistEtaPhi[0]) &&
           (std::abs(edm4hep::utils::angleAzimuthal(h1.getPosition()) -
                     edm4hep::utils::angleAzimuthal(h2.getPosition())) <= layerDistEtaPhi[1]);
  case eicrecon::ImagingTopoClusterConfig::ELayerMode::xy:
    return (std::abs(h1.getPosition().x - h2.getPosition().x) <= layerDistXY[0]) &&
           (std::abs(h1.getPosition().y - h2.getPosition().y) <= layerDistXY[1]);
  }
}
```

2 photon simulation to check the clustering in AstroPix :

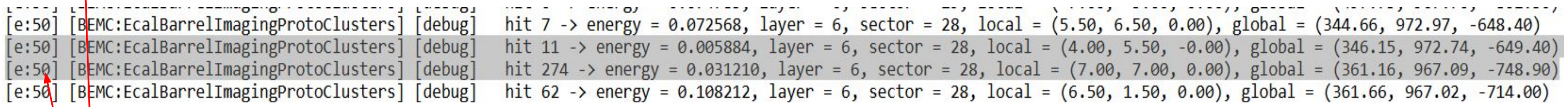
(with uniformly random energy between 0 and 20 GeV and different emission angle, θ)



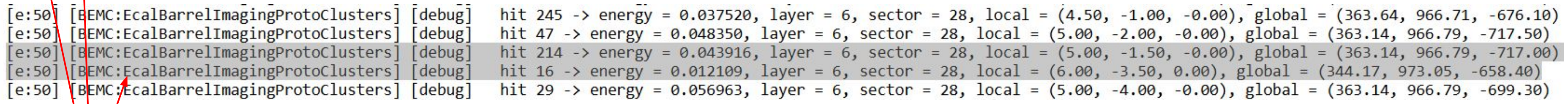
```
// Two photons in the final state
auto [id, mass] = extract_particle_parameters(particle_name);
double theta1 = theta;
double theta2 = theta + TMath::DegToRad()*1.0; // 1 degree difference

// Photon 1
double px1 = p * cos(phi) * sin(theta1);
double py1 = p * sin(phi) * sin(theta1);
double pz1 = p * cos(theta1);
GenParticlePtr photon1 = std::make_shared<GenParticle>(FourVector(px1, py1, pz1, sqrt(px1*px1 + py1*py1 + pz1*pz1)), id, 1);

// Photon 2
double px2 = p * cos(phi) * sin(theta2);
double py2 = p * sin(phi) * sin(theta2);
double pz2 = p * cos(theta2);
GenParticlePtr photon2 = std::make_shared<GenParticle>(FourVector(px2, py2, pz2, sqrt(px2*px2 + py2*py2 + pz2*pz2)), id, 1);
```



```
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 122 -> energy = 0.029419, layer = 4, sector = 28, local = (-6.50, 3.50, -0.00), global = (361.78, 884.67, -732.20)
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 128 -> energy = 0.023024, layer = 4, sector = 28, local = (-6.50, 4.00, -0.00), global = (361.78, 884.67, -731.70)
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 227 -> energy = 0.034110, layer = 6, sector = 28, local = (-7.50, 5.00, -0.00), global = (393.48, 958.78, -670.10)
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 85 -> energy = 0.021318, layer = 4, sector = 28, local = (2.50, 7.50, -0.00), global = (334.25, 895.67, -546.40)
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 17 -> energy = 0.064552, layer = 6, sector = 28, local = (-7.00, 9.50, -0.00), global = (375.00, 964.96, -706.00)
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 195 -> energy = 0.015008, layer = 4, sector = 28, local = (-9.00, -2.00, 0.00), global = (363.98, 883.47, -697.30)
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 26 -> energy = 0.035900, layer = 4, sector = 28, local = (-2.50, 5.00, 0.00), global = (338.63, 893.26, -569.10)
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 28 -> energy = 0.026605, layer = 4, sector = 28, local = (-2.00, 4.00, -0.00), global = (338.19, 893.50, -570.10)
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 102 -> energy = 0.041869, layer = 4, sector = 28, local = (-2.00, 5.00, -0.00), global = (338.19, 893.50, -569.10)
```



```
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 7 -> energy = 0.072568, layer = 6, sector = 28, local = (5.50, 6.50, 0.00), global = (344.66, 972.97, -648.40)
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 11 -> energy = 0.005884, layer = 6, sector = 28, local = (4.00, 5.50, -0.00), global = (346.15, 972.74, -649.40)
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 274 -> energy = 0.031210, layer = 6, sector = 28, local = (7.00, 7.00, 0.00), global = (361.16, 967.09, -748.90)
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 62 -> energy = 0.108212, layer = 6, sector = 28, local = (6.50, 1.50, 0.00), global = (361.66, 967.02, -714.00)
```



```
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 245 -> energy = 0.037520, layer = 6, sector = 28, local = (4.50, -1.00, -0.00), global = (363.64, 966.71, -676.10)
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 47 -> energy = 0.048350, layer = 6, sector = 28, local = (5.00, -2.00, -0.00), global = (363.14, 966.79, -717.50)
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 214 -> energy = 0.043916, layer = 6, sector = 28, local = (5.00, -1.50, -0.00), global = (363.14, 966.79, -717.00)
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 16 -> energy = 0.012109, layer = 6, sector = 28, local = (6.00, -3.50, 0.00), global = (344.17, 973.05, -658.40)
[e:50] [BEMC:EcalBarrelImagingProtoClusters] [debug]   hit 29 -> energy = 0.056963, layer = 6, sector = 28, local = (5.00, -4.00, -0.00), global = (363.14, 966.79, -699.30)
```

Imaging hits in the same layer which are far in global distance are grouped

Right now, we use the variables:

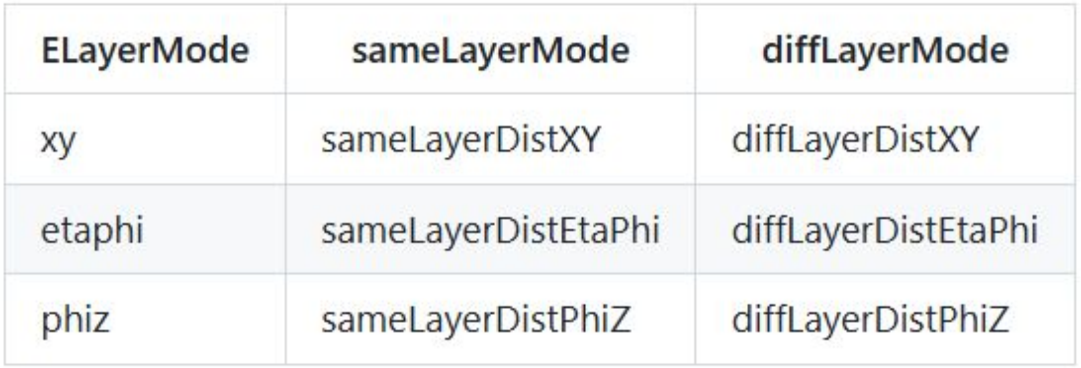
same layer check

- localDistXY

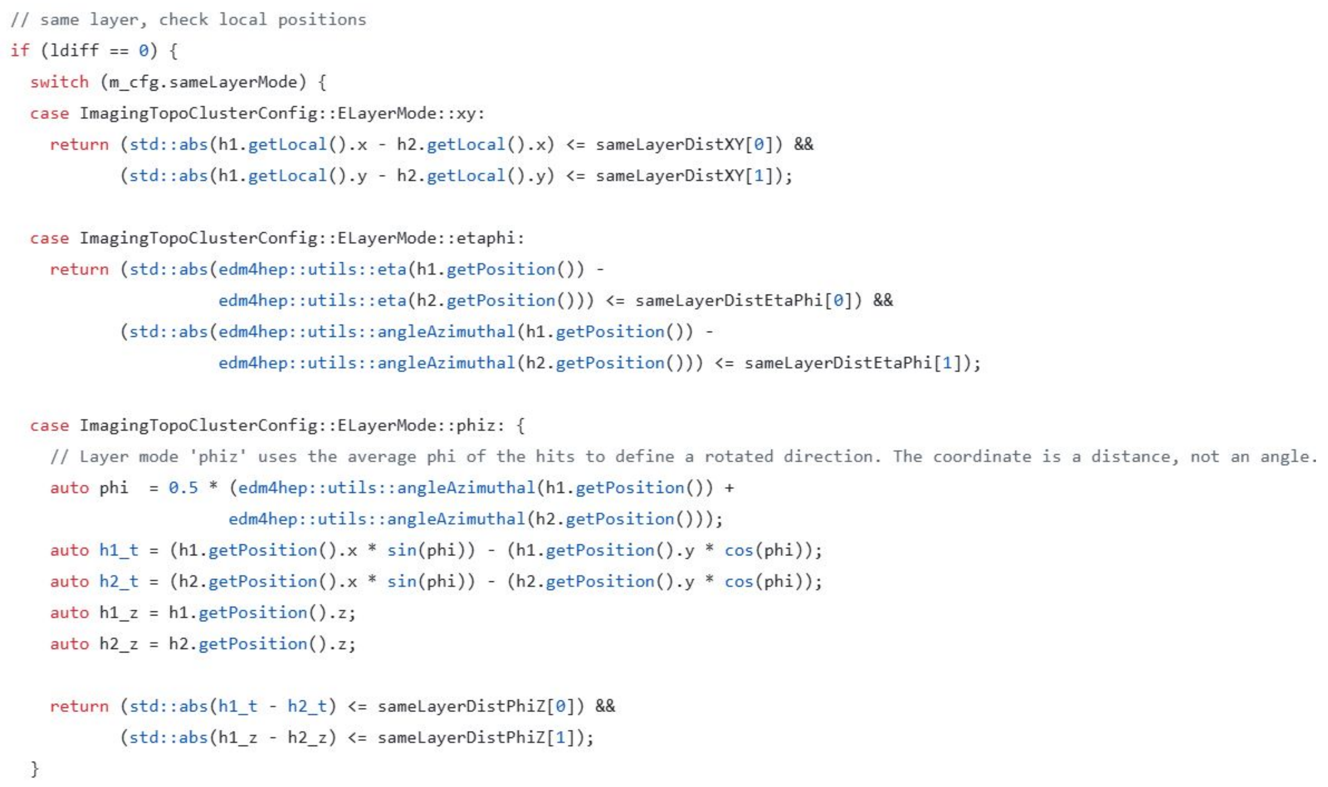
Different Layer Check

- layerDistEtaPhi
- layerDistXY

With the new Layer modes we have (https://github.com/eic/EICrecon/pull/1999):



ELayerMode	sameLayerMode	diffLayerMode
xy	sameLayerDistXY	diffLayerDistXY
etaphi	sameLayerDistEtaPhi	diffLayerDistEtaPhi
phiz	sameLayerDistPhiZ	diffLayerDistPhiZ



```cpp
// same layer, check local positions
if (ldiff == 0) {
  switch (m_cfg.sameLayerMode) {
  case ImagingTopoClusterConfig::ELayerMode::xy:
    return (std::abs(h1.getLocal().x - h2.getLocal().x) <= sameLayerDistXY[0]) &&
           (std::abs(h1.getLocal().y - h2.getLocal().y) <= sameLayerDistXY[1]);

  case ImagingTopoClusterConfig::ELayerMode::etaphi:
    return (std::abs(edm4hep::utils::eta(h1.getPosition()) -
                     edm4hep::utils::eta(h2.getPosition())) <= sameLayerDistEtaPhi[0]) &&
           (std::abs(edm4hep::utils::angleAzimuthal(h1.getPosition()) -
                     edm4hep::utils::angleAzimuthal(h2.getPosition())) <= sameLayerDistEtaPhi[1]);

  case ImagingTopoClusterConfig::ELayerMode::phiz: {
    // Layer mode 'phiz' uses the average phi of the hits to define a rotated direction. The coordinate is a distance, not an angle.
    auto phi  = 0.5 * (edm4hep::utils::angleAzimuthal(h1.getPosition()) +
                      edm4hep::utils::angleAzimuthal(h2.getPosition()));
    auto h1_t = (h1.getPosition().x * sin(phi)) - (h1.getPosition().y * cos(phi));
    auto h2_t = (h2.getPosition().x * sin(phi)) - (h2.getPosition().y * cos(phi));
    auto h1_z = h1.getPosition().z;
    auto h2_z = h2.getPosition().z;

    return (std::abs(h1_t - h2_t) <= sameLayerDistPhiZ[0]) &&
           (std::abs(h1_z - h2_z) <= sameLayerDistPhiZ[1]);
  }
```

Introduced different Layer mode :

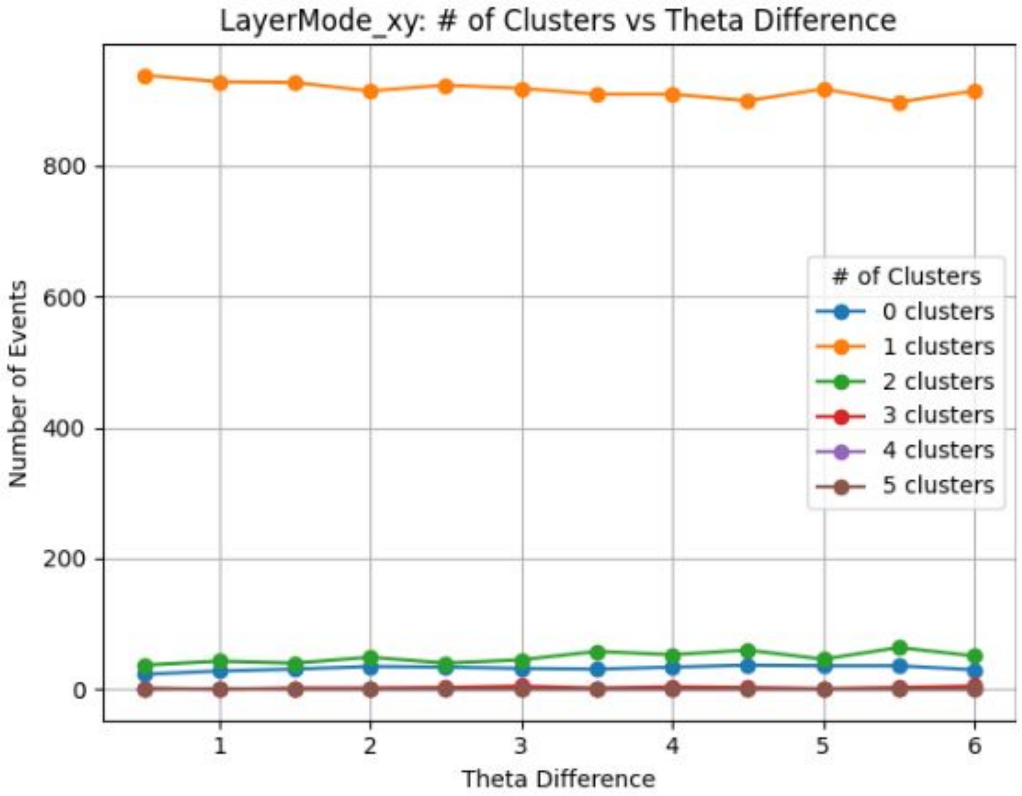
xy mode: Use local coordinates (x, y) to check how close hits are.

etaphi mode: Use pseudorapidity (η) and azimuthal angle (φ) and thus gives the angular distance between the hits

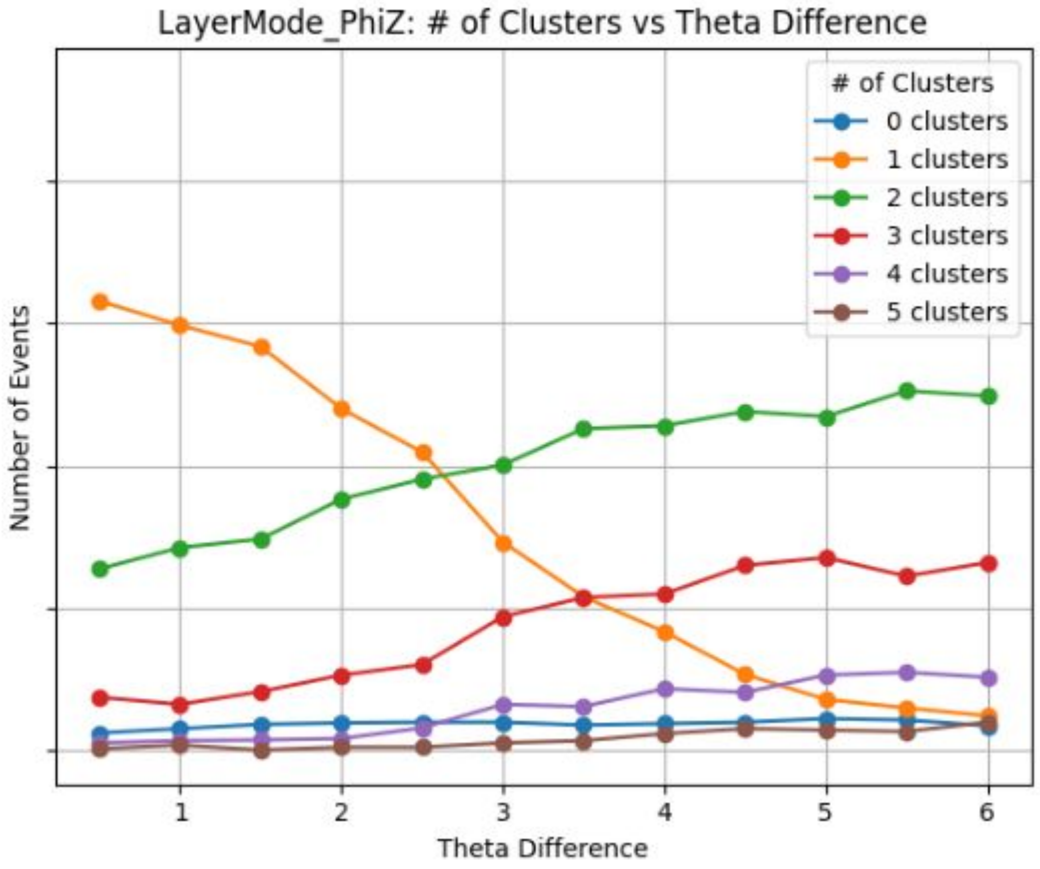
phiz mode: Use azimuthal angle (φ) and the longitudinal coordinate (z). Converts the azimuthal (φ) separation between hits into a tangential distance (in millimeters) by projecting hit positions relative to the average phi of the two hits.)

```cpp
} else if (ldiff <= m_cfg.neighbourLayersRange) {
  switch (m_cfg.diffLayerMode) {
  case eicrecon::ImagingTopoClusterConfig::ELayerMode::etaphi:
    return (std::abs(edm4hep::utils::eta(h1.getPosition()) -
                     edm4hep::utils::eta(h2.getPosition())) <= diffLayerDistEtaPhi[0]) &&
           (std::abs(edm4hep::utils::angleAzimuthal(h1.getPosition()) -
                     edm4hep::utils::angleAzimuthal(h2.getPosition())) <= diffLayerDistEtaPhi[1]);

  case eicrecon::ImagingTopoClusterConfig::ELayerMode::xy:
    // Here, the xy layer mode is based on global XY positions rather than local XY positions, and thus it only works for endcap detectors.
    return (std::abs(h1.getPosition().x - h2.getPosition().x) <= diffLayerDistXY[0]) &&
           (std::abs(h1.getPosition().y - h2.getPosition().y) <= diffLayerDistXY[1]);

  case eicrecon::ImagingTopoClusterConfig::ELayerMode::phiz: {
    auto phi  = 0.5 * (edm4hep::utils::angleAzimuthal(h1.getPosition()) +
                      edm4hep::utils::angleAzimuthal(h2.getPosition()));
    auto h1_t = (h1.getPosition().x * sin(phi)) - (h1.getPosition().y * cos(phi));
    auto h2_t = (h2.getPosition().x * sin(phi)) - (h2.getPosition().y * cos(phi));
    auto h1_z = h1.getPosition().z;
    auto h2_z = h2.getPosition().z;

    return (std::abs(h1_t - h2_t) <= diffLayerDistPhiZ[0]) &&
           (std::abs(h1_z - h2_z) <= diffLayerDistPhiZ[1]);
  }
```

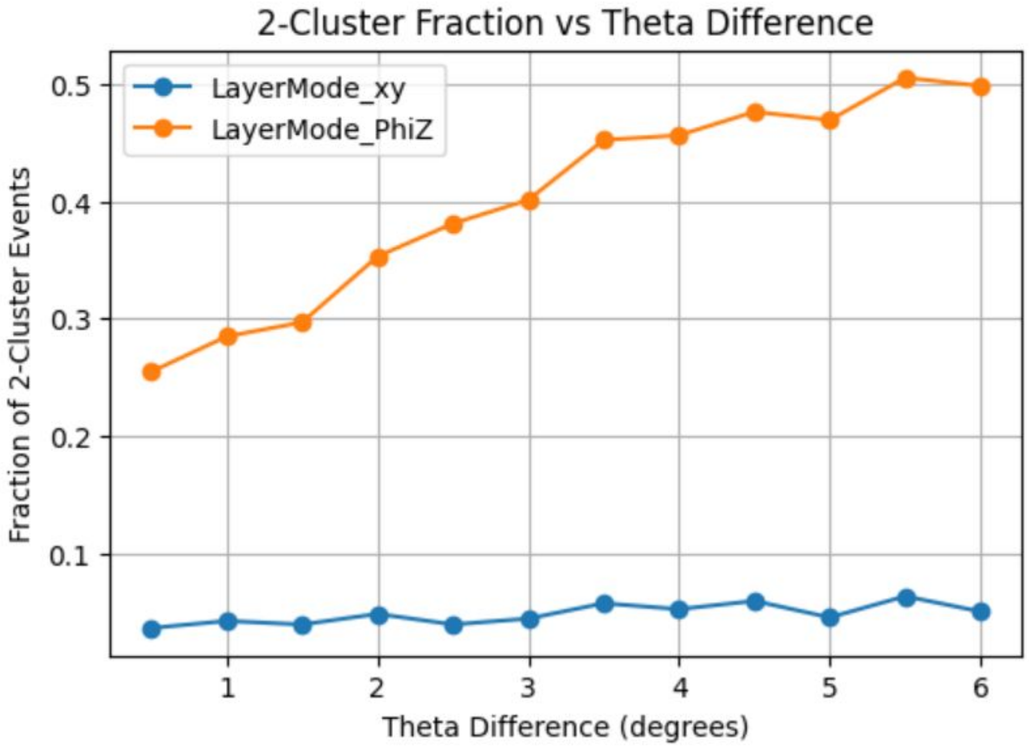
Layermode xy (based on local x position x and y) and the Layermode phiz (based on phi direction converted to mm and z) are compared for same Layer Check



Here, The orange curve (1 cluster) dominates across all θ differences, meaning the algorithm mostly merges everything into a single cluster even when the angular separation between showers increases. The number of events with 2 or more clusters stays very low and almost flat with θ difference thus the mode is not very sensitive to angular separation in this configuration.



Here, a clear trend appears as θ difference increases, the case of “1 cluster” events (orange) drops, while “2 clusters” (green) and higher cluster counts increase. This shows that phiz mode is actually resolving separate showers as their angular difference grows, rather than over-merging them.



The blue curve (XY mode) stays less than 10% across all θ differences, meaning XY mode rarely finds exactly two separate clusters, regardless of how far apart the showers are.
The orange curve (phiz mode) starts around 25% for small θ differences and rises to ~50% for θ difference of 5.5° showing that phiz mode identifies the expected two clusters as the events become more separable.

Summary:

- The existing local xy-based clustering tends to over-merge hits, even when showers are far apart.

- The new phiz mode shows sensitivity to angular differences and thus can separate the two showers.