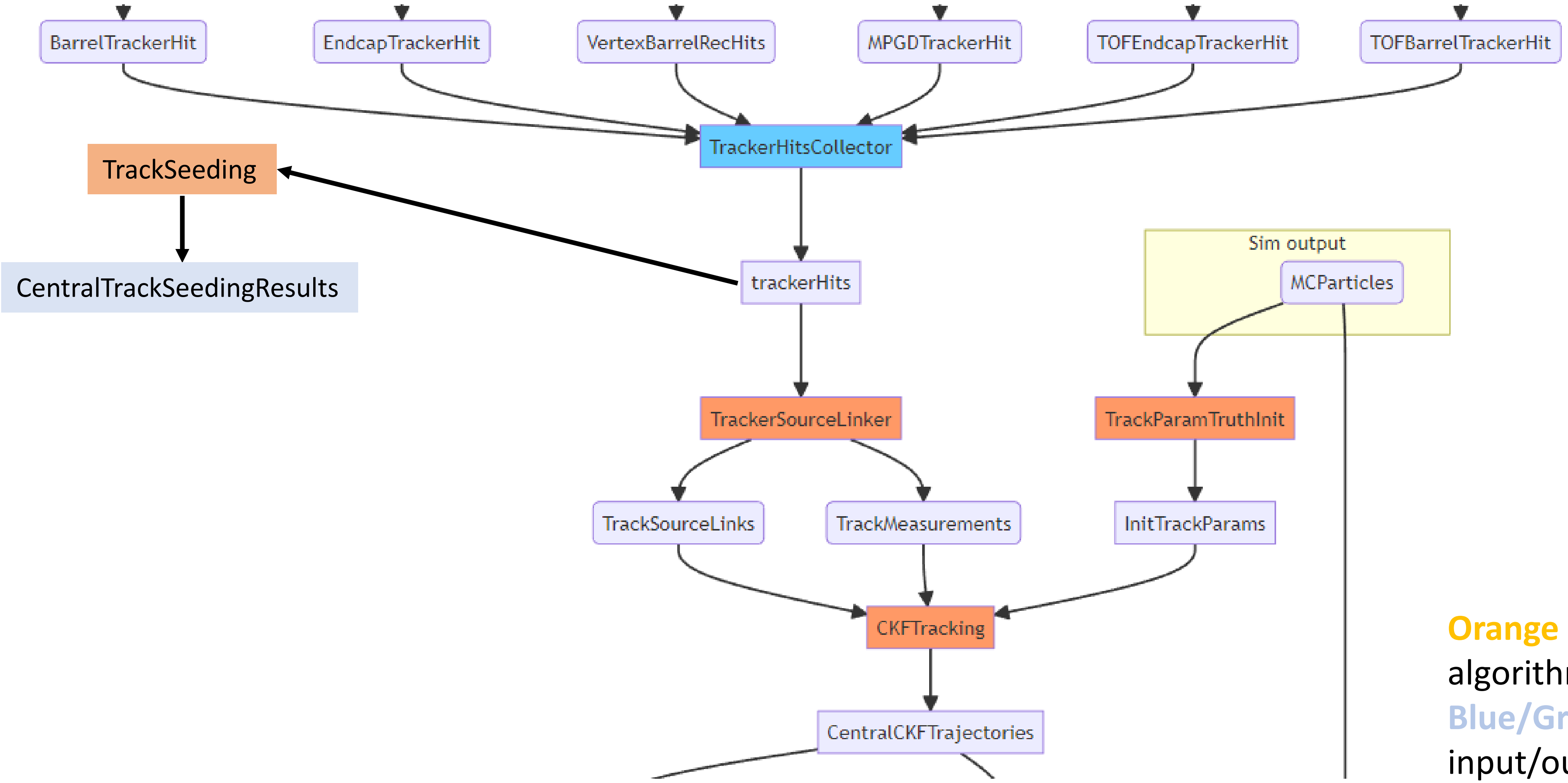
# Track fitting in *EICRecon* using the ACTS orthogonal seeder

Beatrice Liang-Gilman,

Barak Schmookler,

Reynier Cruz Torres

# Track reconstruction logic in *EICRecon*



BarrelTrackerHit

EndcapTrackerHit

VertexBarrelRecHits

MPGDTrackerHit

TOFEndcapTrackerHit

TOFBarrelTrackerHit

TrackerHitsCollector

TrackSeeding

CentralTrackSeedingResults

trackerHits

Sim output

MCParticles

TrackerSourceLinker

TrackParamTruthInit

TrackSourceLinks

TrackMeasurements

InitTrackParams

CKFTracking

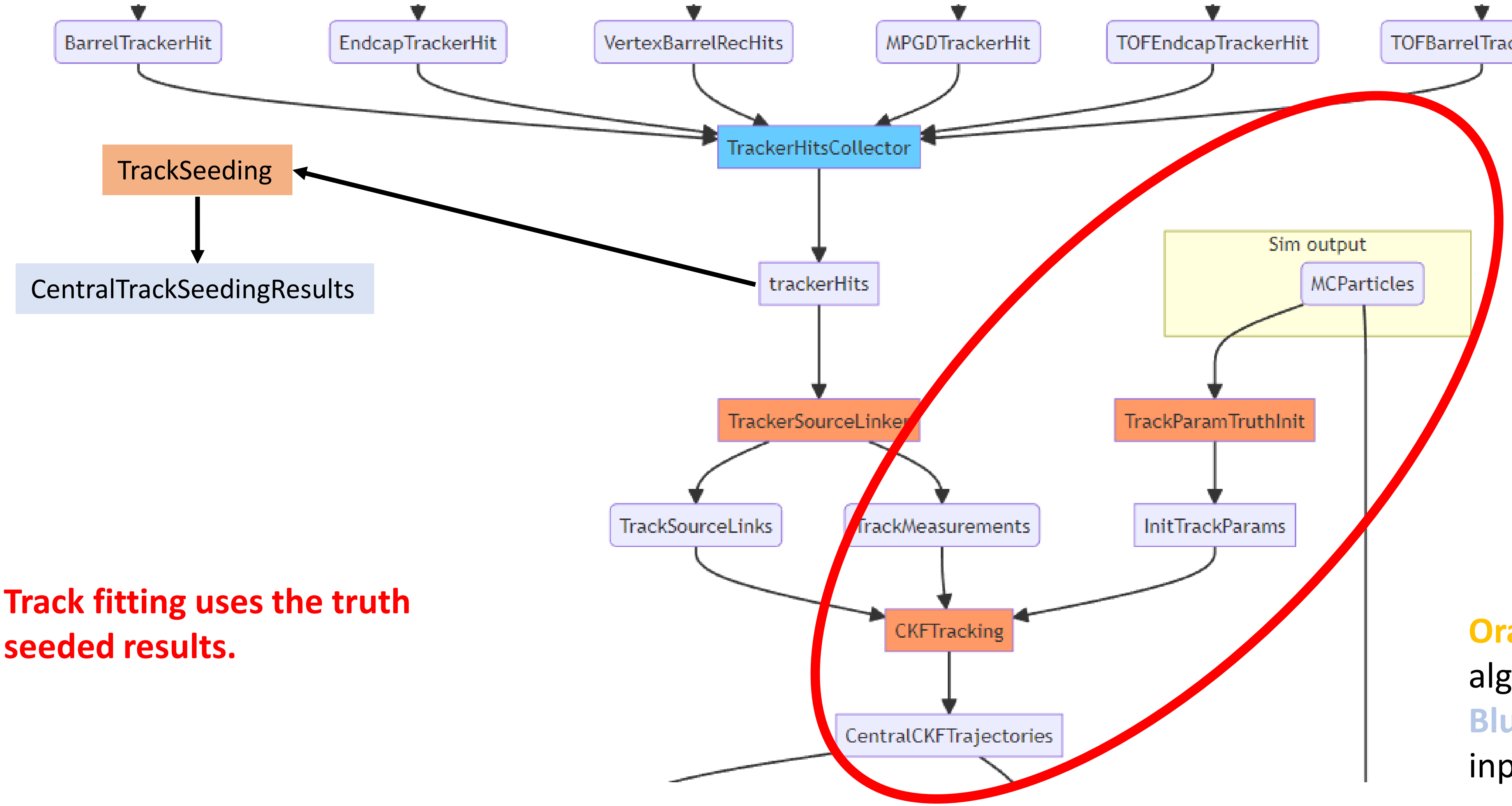
CentralCKFTrajectories

**Orange** boxes are algorithms.

**Blue/Grey** boxes are input/output data.

https://eic.github.io/EICrecon/#/design/tracking

# Track reconstruction logic in *EICRecon*



BarrelTrackerHit

EndcapTrackerHit

VertexBarrelRecHits

MPGDTrackerHit

TOFEndcapTrackerHit

TOFBarrelTrackerHit

TrackerHitsCollector

TrackSeeding

CentralTrackSeedingResults

trackerHits

Sim output

MCParticles

TrackerSourceLinker

TrackParamTruthInit

TrackSourceLinks

TrackMeasurements

InitTrackParams

CKFTracking

CentralCKFTrajectories

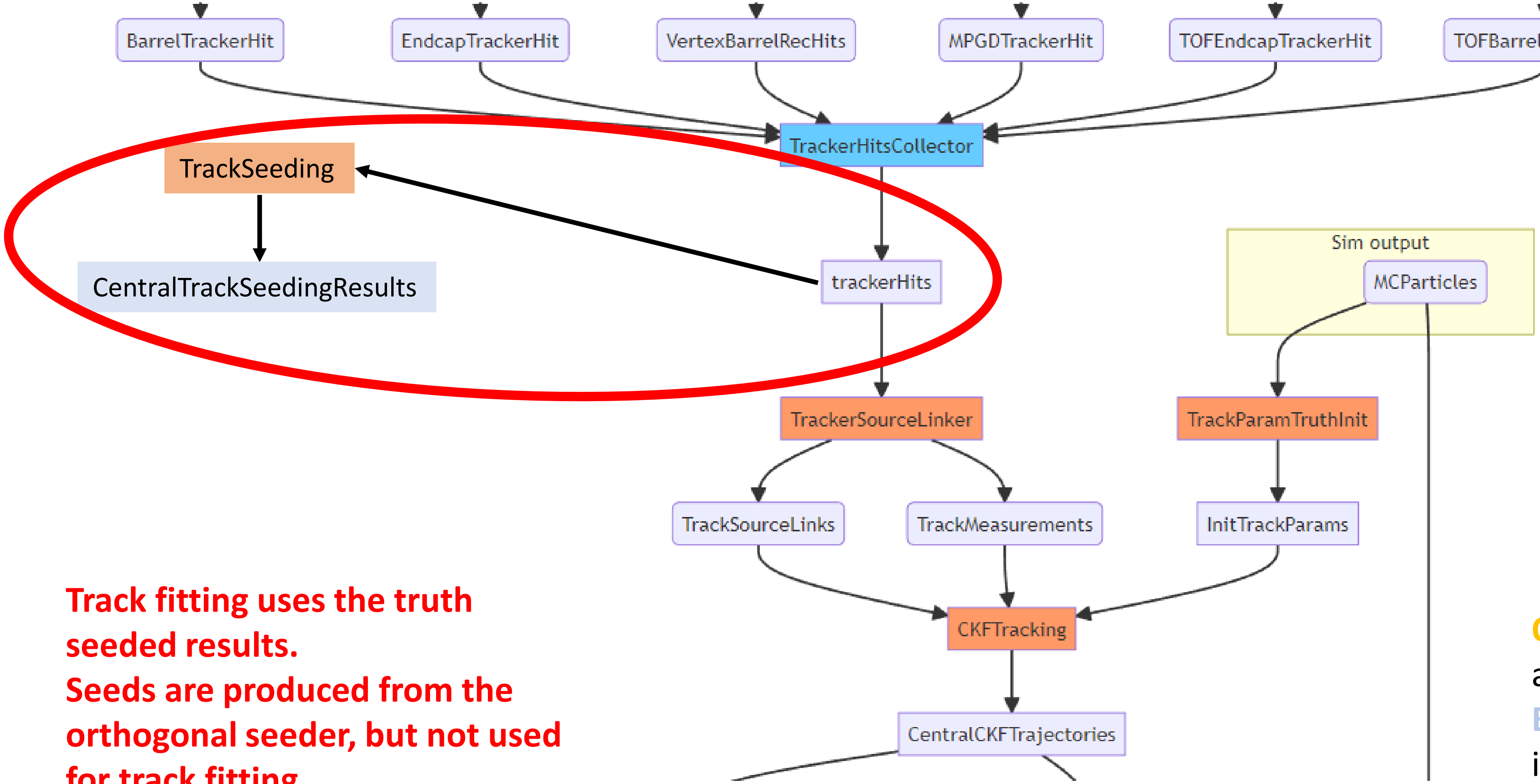
**Track fitting uses the truth seeded results.**

Orange boxes are algorithms.

Blue/Grey boxes are input/output data.

https://eic.github.io/EICrecon/#/design/tracking

# Track reconstruction logic in *EICRecon*

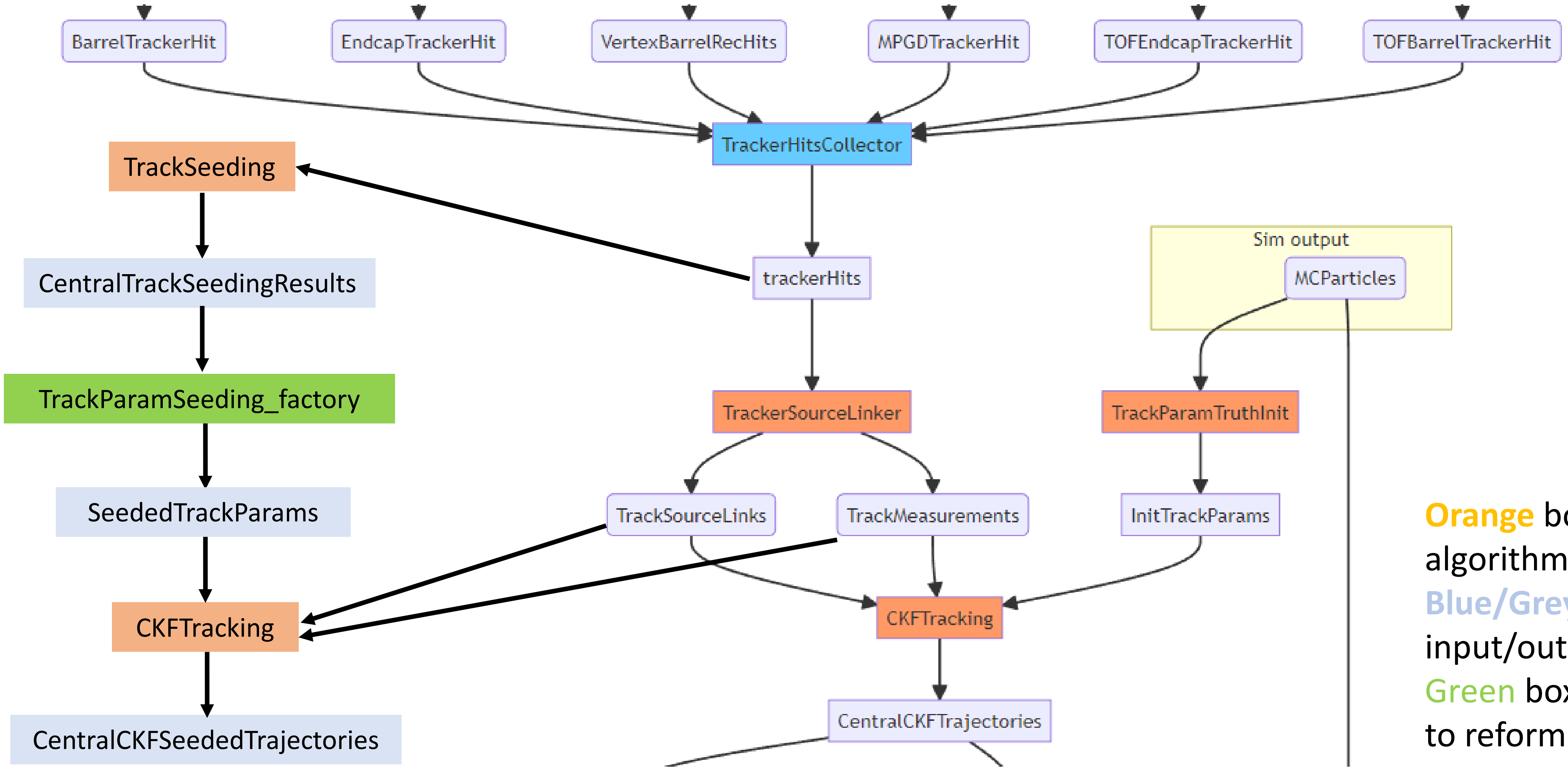


**Track fitting uses the truth seeded results.**
**Seeds are produced from the orthogonal seeder, but not used for track fitting.**

Orange boxes are algorithms.
Blue/Grey boxes are input/output data.

https://eic.github.io/EICrecon/#/design/tracking

# Update to use real seed for tracking



Orange boxes are algorithms.
Blue/Grey boxes are input/output data.
Green box is a factory to reformat data type.

https://eic.github.io/EICrecon/#/design/tracking

# How this can be used

- ➢The code lives in our track-QA branch:

  https://github.com/eic/EICrecon/tree/track-qa-barak

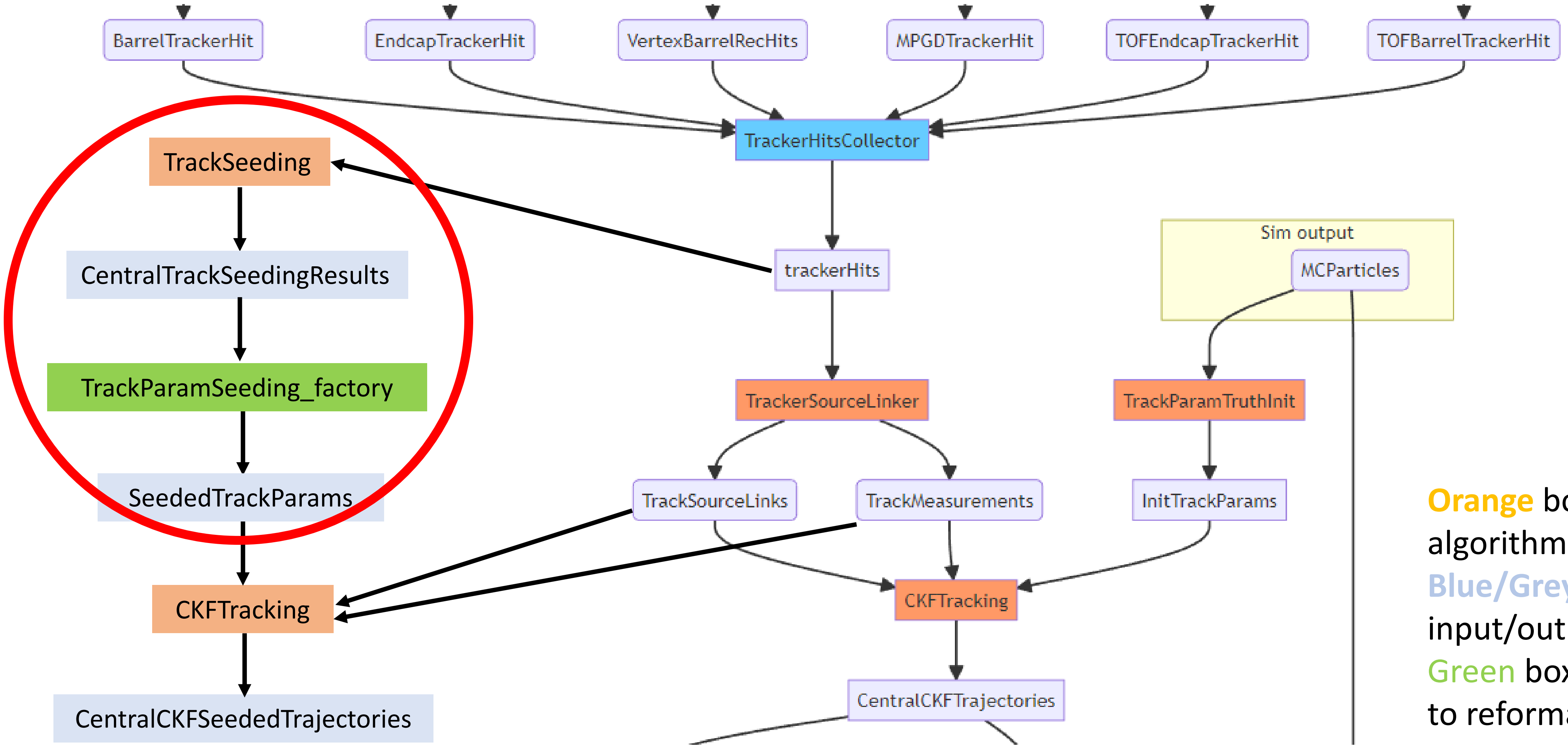
- ➢The new datatypes and factories shown above do not affect the previous workflow. So, they can be merged into the main branch without causing any changes to the standard output ROOT file.
- ➢A user can access the tracks which use the realistic seeding by using the following in a Plugin:

*auto trajectories = event->Get<eicrecon::TrackingResultTrajectory>("CentralCKFSeededTrajectories");*

instead of (for truth seeded tracks):

*auto trajectories = event->Get<eicrecon::TrackingResultTrajectory>("CentralCKFTrajectories");*

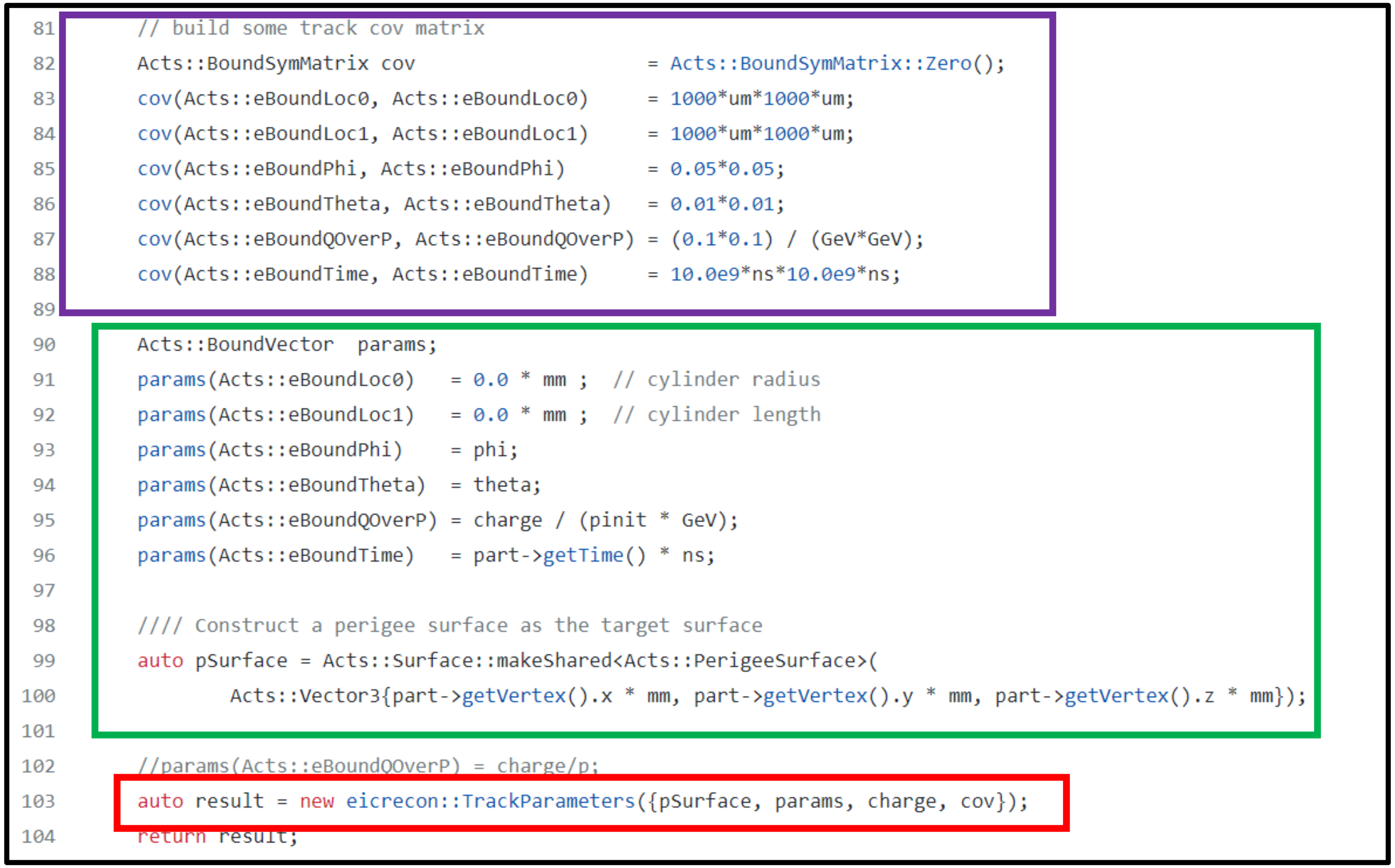
# Seeding parameter format



Orange boxes are algorithms.
Blue/Grey boxes are input/output data.
Green box is a factory to reformat data type.

https://eic.github.io/EICrecon/#/design/tracking

# Truth seeded parameters



```
    // build some track cov matrix
    Acts::BoundSymMatrix cov                    = Acts::BoundSymMatrix::Zero();
    cov(Acts::eBoundLoc0, Acts::eBoundLoc0)     = 1000*um*1000*um;
    cov(Acts::eBoundLoc1, Acts::eBoundLoc1)     = 1000*um*1000*um;
    cov(Acts::eBoundPhi, Acts::eBoundPhi)       = 0.05*0.05;
    cov(Acts::eBoundTheta, Acts::eBoundTheta)   = 0.01*0.01;
    cov(Acts::eBoundQOverP, Acts::eBoundQOverP) = (0.1*0.1) / (GeV*GeV);
    cov(Acts::eBoundTime, Acts::eBoundTime)     = 10.0e9*ns*10.0e9*ns;

    Acts::BoundVector  params;
    params(Acts::eBoundLoc0)   = 0.0 * mm ;  // cylinder radius
    params(Acts::eBoundLoc1)   = 0.0 * mm ;  // cylinder length
    params(Acts::eBoundPhi)    = phi;
    params(Acts::eBoundTheta)  = theta;
    params(Acts::eBoundQOverP) = charge / (pinit * GeV);
    params(Acts::eBoundTime)   = part->getTime() * ns;

    //// Construct a perigee surface as the target surface
    auto pSurface = Acts::Surface::makeShared<Acts::PerigeeSurface>(
            Acts::Vector3{part->getVertex().x * mm, part->getVertex().y * mm, part->getVertex().z * mm});

    //params(Acts::eBoundQOverP) = charge/p;
    auto result = new eicrecon::TrackParameters({pSurface, params, charge, cov});
    return result;
```

**This is what gets passed to the tracking algorithm.**

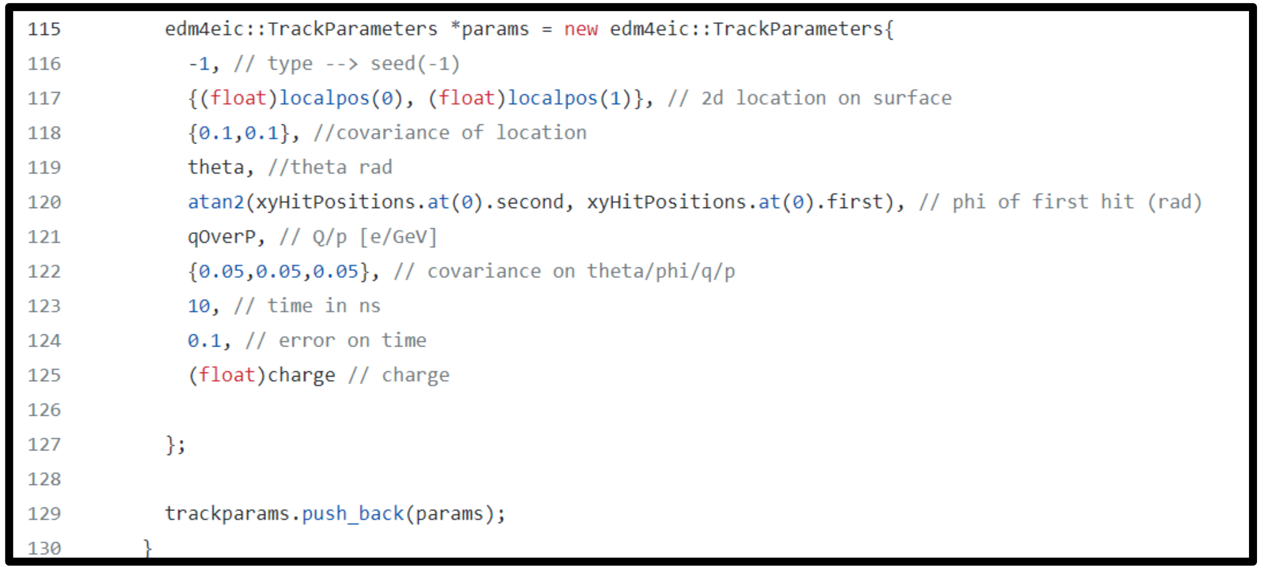
**These parameters and surface come from the generated particle.**

**The covariance matrix of the seed is hardcoded.**

https://github.com/eic/EICrecon/blob/main/src/algorithms/tracking/TrackParamTruthInit.cc

# Put orthogonal seeding output into same format

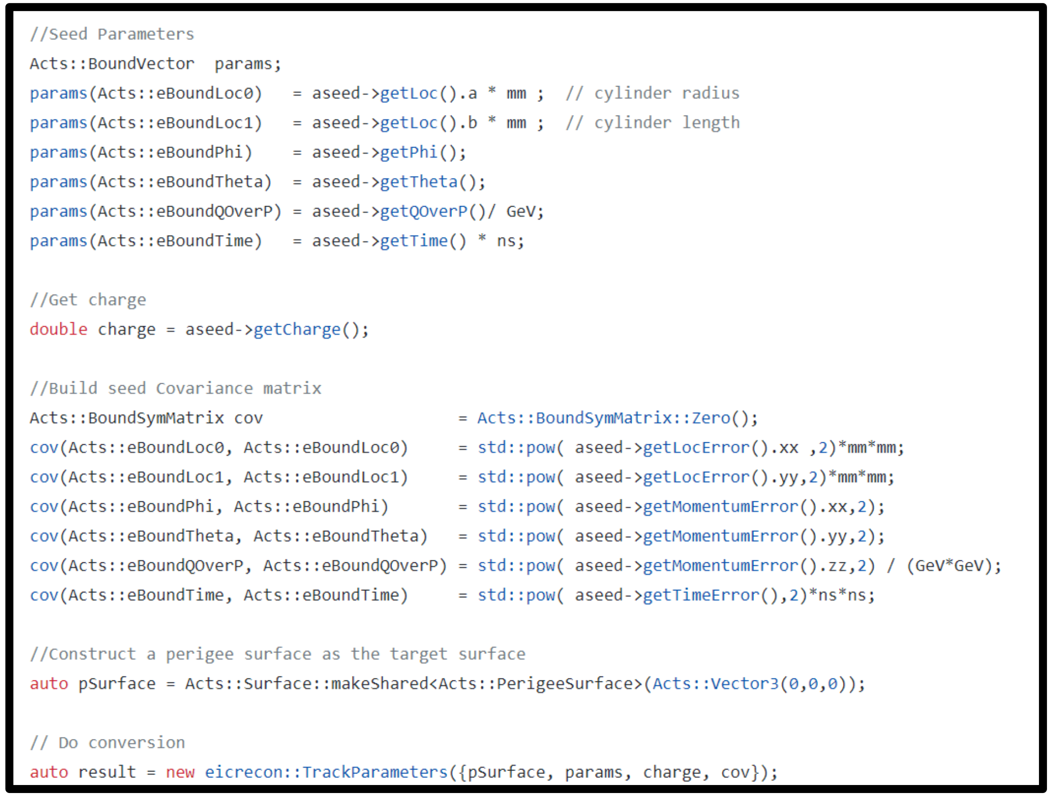
## Seeder writes out an edm4eic::TrackParameters data type which can be accessed in ROOT file



```
   edm4eic::TrackParameters *params = new edm4eic::TrackParameters{
    -1, // type --> seed(-1)
    {(float)localpos(0), (float)localpos(1)}, // 2d location on surface
    {0.1,0.1}, //covariance of location
    theta, //theta rad
    atan2(xyHitPositions.at(0).second, xyHitPositions.at(0).first), // phi of first hit (rad)
    qOverP, // Q/p [e/GeV]
    {0.05,0.05,0.05}, // covariance on theta/phi/q/p
    10, // time in ns
    0.1, // error on time
    (float)charge // charge

   };

   trackparams.push_back(params);
  }
```

https://github.com/eic/EICrecon/blob/main/src/algorithms/tracking/TrackSeeding.cc

## Convert to eicrecon::TrackParameters data type as input for track fitting

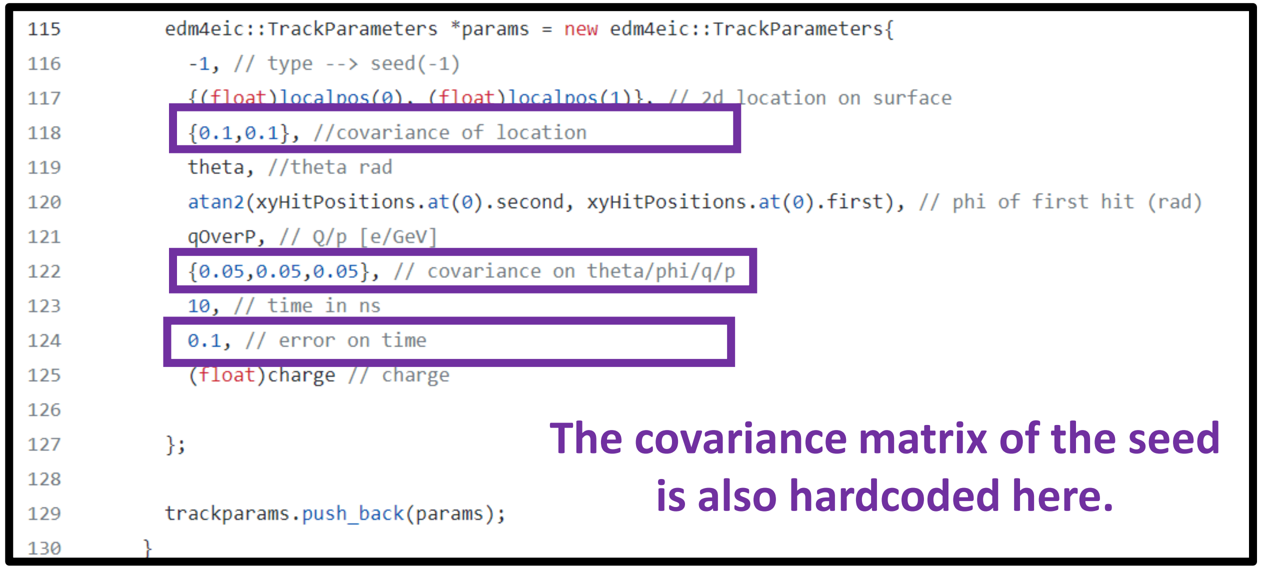


```
//Seed Parameters
Acts::BoundVector  params;
params(Acts::eBoundLoc0)   = aseed->getLoc().a * mm ;  // cylinder radius
params(Acts::eBoundLoc1)   = aseed->getLoc().b * mm ;  // cylinder length
params(Acts::eBoundPhi)    = aseed->getPhi();
params(Acts::eBoundTheta)  = aseed->getTheta();
params(Acts::eBoundQOverP) = aseed->getQOverP()/ GeV;
params(Acts::eBoundTime)   = aseed->getTime() * ns;

//Get charge
double charge = aseed->getCharge();

//Build seed Covariance matrix
Acts::BoundSymMatrix cov                    = Acts::BoundSymMatrix::Zero();
cov(Acts::eBoundLoc0, Acts::eBoundLoc0)     = std::pow( aseed->getLocError().xx ,2)*mm*mm;
cov(Acts::eBoundLoc1, Acts::eBoundLoc1)     = std::pow( aseed->getLocError().yy,2)*mm*mm;
cov(Acts::eBoundPhi, Acts::eBoundPhi)       = std::pow( aseed->getMomentumError().xx,2);
cov(Acts::eBoundTheta, Acts::eBoundTheta)   = std::pow( aseed->getMomentumError().yy,2);
cov(Acts::eBoundQOverP, Acts::eBoundQOverP) = std::pow( aseed->getMomentumError().zz,2) / (GeV*GeV);
cov(Acts::eBoundTime, Acts::eBoundTime)     = std::pow( aseed->getTimeError(),2)*ns*ns;

//Construct a perigee surface as the target surface
auto pSurface = Acts::Surface::makeShared<Acts::PerigeeSurface>(Acts::Vector3(0,0,0));

// Do conversion
auto result = new eicrecon::TrackParameters({pSurface, params, charge, cov});
```

https://github.com/eic/EICrecon/blob/track-qa-barak/src/global/tracking/TrackParamSeeding_factory.cc

# Put orthogonal seeding output into same format

**Seeder writes out an edm4eic::TrackParameters data type which can be accessed in ROOT file**

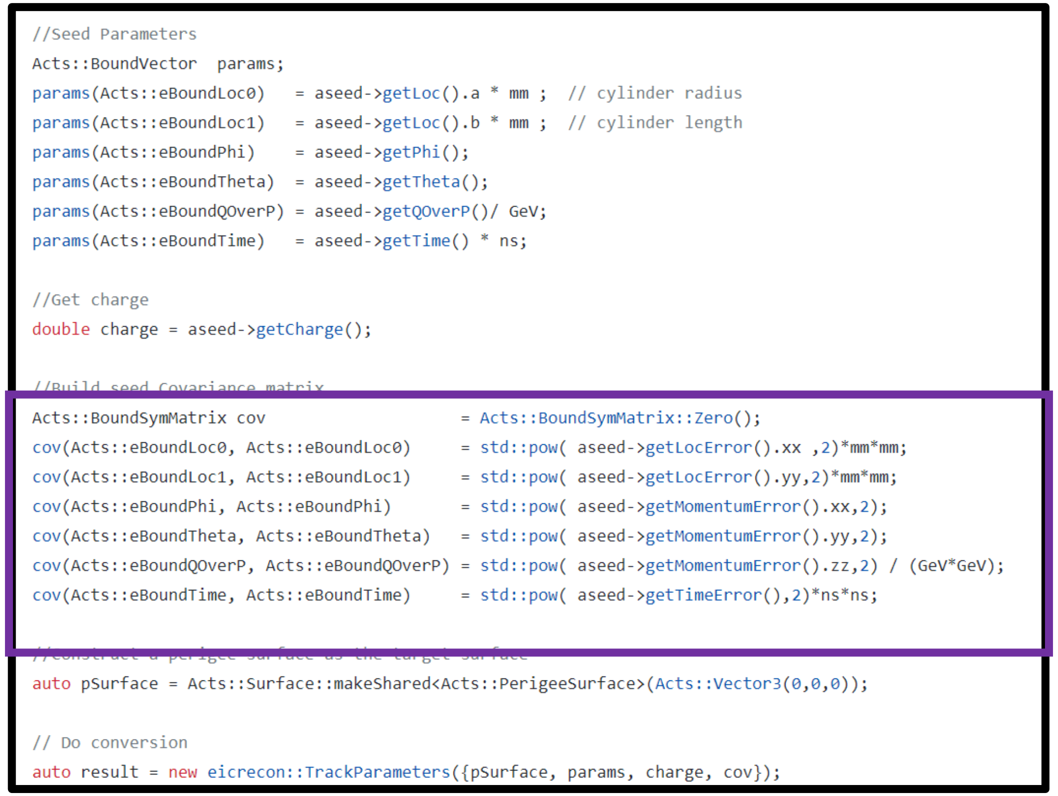


```
  edm4eic::TrackParameters *params = new edm4eic::TrackParameters{
    -1, // type --> seed(-1)
    {(float)localpos(0), (float)localpos(1)}, // 2d location on surface
    {0.1,0.1}, //covariance of location
    theta, //theta rad
    atan2(xyHitPositions.at(0).second, xyHitPositions.at(0).first), // phi of first hit (rad)
    qOverP, // Q/p [e/GeV]
    {0.05,0.05,0.05}, // covariance on theta/phi/q/p
    10, // time in ns
    0.1, // error on time
    (float)charge // charge

  };

  trackparams.push_back(params);
}
```

**The covariance matrix of the seed is also hardcoded here.**

https://github.com/eic/EICrecon/blob/main/src/algorithms/tracking/TrackSeeding.cc

**Convert to eicrecon::TrackParameters data type as input for track fitting**

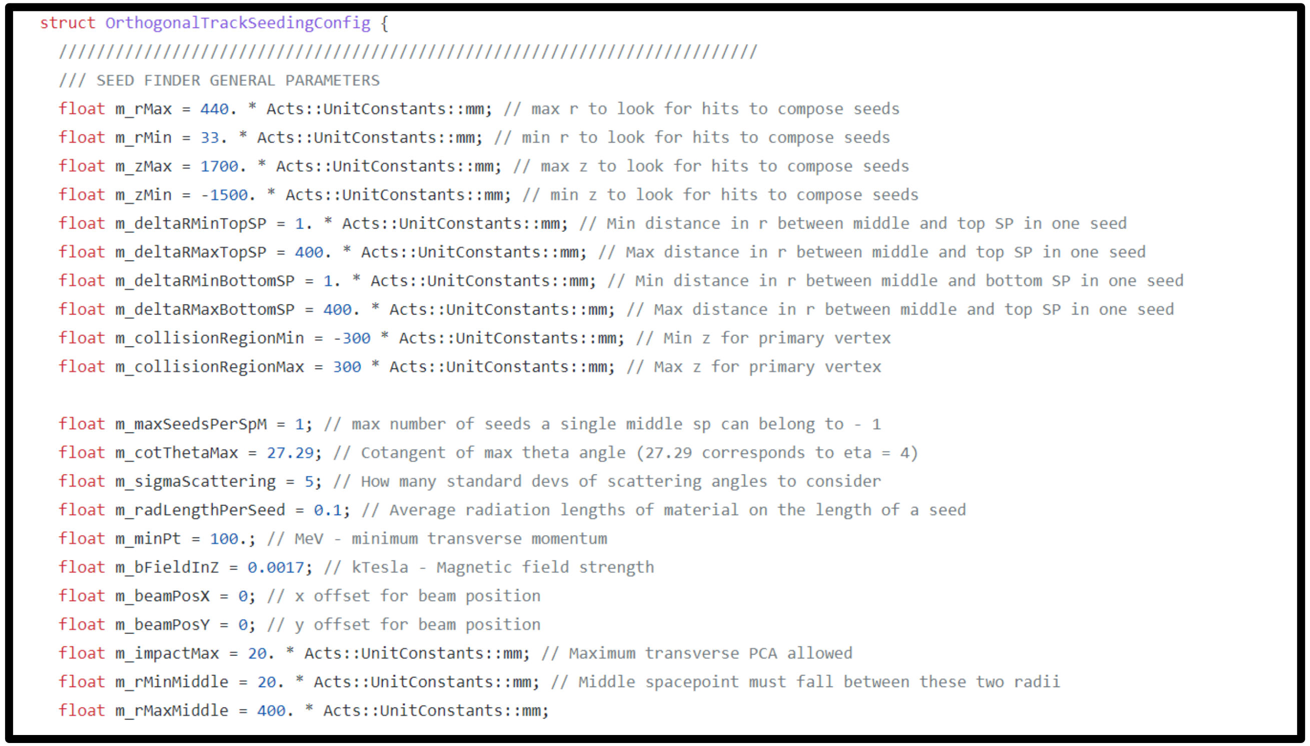


```
//Seed Parameters
Acts::BoundVector  params;
params(Acts::eBoundLoc0)    = aseed->getLoc().a * mm ;  // cylinder radius
params(Acts::eBoundLoc1)    = aseed->getLoc().b * mm ;  // cylinder length
params(Acts::eBoundPhi)     = aseed->getPhi();
params(Acts::eBoundTheta)   = aseed->getTheta();
params(Acts::eBoundQOverP)  = aseed->getQOverP()/ GeV;
params(Acts::eBoundTime)    = aseed->getTime() * ns;

//Get charge
double charge = aseed->getCharge();

//Build seed Covariance matrix
Acts::BoundSymMatrix cov                    = Acts::BoundSymMatrix::Zero();
cov(Acts::eBoundLoc0, Acts::eBoundLoc0)     = std::pow( aseed->getLocError().xx ,2)*mm*mm;
cov(Acts::eBoundLoc1, Acts::eBoundLoc1)     = std::pow( aseed->getLocError().yy,2)*mm*mm;
cov(Acts::eBoundPhi, Acts::eBoundPhi)       = std::pow( aseed->getMomentumError().xx,2);
cov(Acts::eBoundTheta, Acts::eBoundTheta)   = std::pow( aseed->getMomentumError().yy,2);
cov(Acts::eBoundQOverP, Acts::eBoundQOverP) = std::pow( aseed->getMomentumError().zz,2) / (GeV*GeV);
cov(Acts::eBoundTime, Acts::eBoundTime)     = std::pow( aseed->getTimeError(),2)*ns*ns;

//Construct a perigee surface as the target surface
auto pSurface = Acts::Surface::makeShared<Acts::PerigeeSurface>(Acts::Vector3(0,0,0));

// Do conversion
auto result = new eicrecon::TrackParameters({pSurface, params, charge, cov});
```

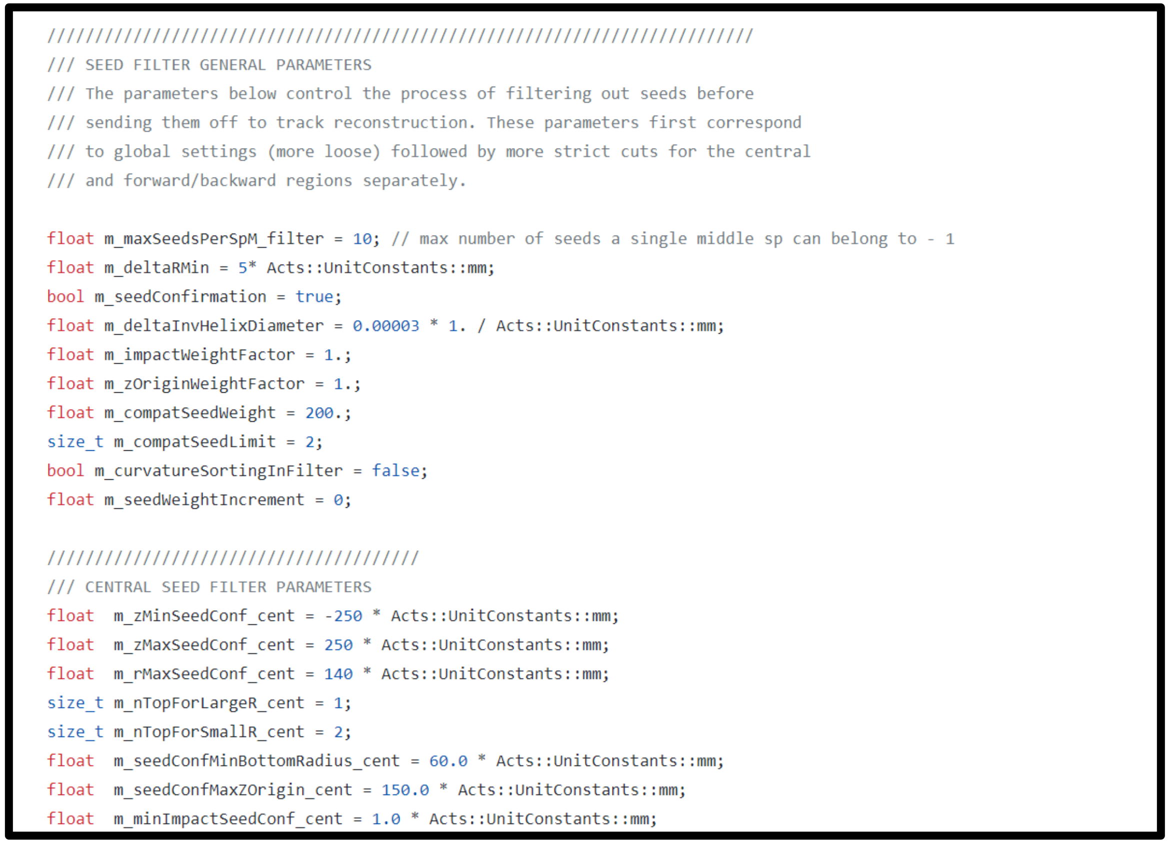
https://github.com/eic/EICrecon/blob/track-qa-barak/src/global/tracking/TrackParamSeeding_factory.cc

# Seeder configuration file

Parameters can be changed on command line during *EICRecon* running



```
struct OrthogonalTrackSeedingConfig {
  //////////////////////////////////////////////////////////////////////////
  /// SEED FINDER GENERAL PARAMETERS
  float m_rMax = 440. * Acts::UnitConstants::mm; // max r to look for hits to compose seeds
  float m_rMin = 33. * Acts::UnitConstants::mm; // min r to look for hits to compose seeds
  float m_zMax = 1700. * Acts::UnitConstants::mm; // max z to look for hits to compose seeds
  float m_zMin = -1500. * Acts::UnitConstants::mm; // min z to look for hits to compose seeds
  float m_deltaRMinTopSP = 1. * Acts::UnitConstants::mm; // Min distance in r between middle and top SP in one seed
  float m_deltaRMaxTopSP = 400. * Acts::UnitConstants::mm; // Max distance in r between middle and top SP in one seed
  float m_deltaRMinBottomSP = 1. * Acts::UnitConstants::mm; // Min distance in r between middle and bottom SP in one seed
  float m_deltaRMaxBottomSP = 400. * Acts::UnitConstants::mm; // Max distance in r between middle and top SP in one seed
  float m_collisionRegionMin = -300 * Acts::UnitConstants::mm; // Min z for primary vertex
  float m_collisionRegionMax = 300 * Acts::UnitConstants::mm; // Max z for primary vertex

  float m_maxSeedsPerSpM = 1; // max number of seeds a single middle sp can belong to - 1
  float m_cotThetaMax = 27.29; // Cotangent of max theta angle (27.29 corresponds to eta = 4)
  float m_sigmaScattering = 5; // How many standard devs of scattering angles to consider
  float m_radLengthPerSeed = 0.1; // Average radiation lengths of material on the length of a seed
  float m_minPt = 100.; // MeV - minimum transverse momentum
  float m_bFieldInZ = 0.0017; // kTesla - Magnetic field strength
  float m_beamPosX = 0; // x offset for beam position
  float m_beamPosY = 0; // y offset for beam position
  float m_impactMax = 20. * Acts::UnitConstants::mm; // Maximum transverse PCA allowed
  float m_rMinMiddle = 20. * Acts::UnitConstants::mm; // Middle spacepoint must fall between these two radii
  float m_rMaxMiddle = 400. * Acts::UnitConstants::mm;
```

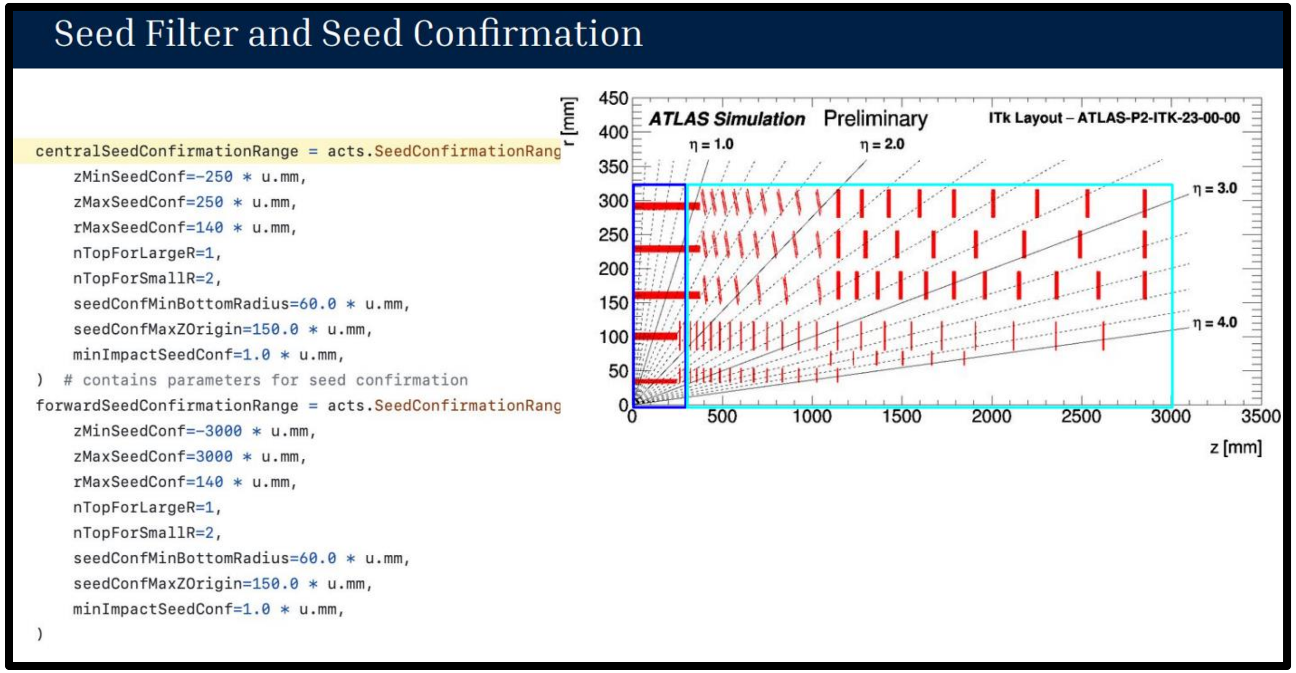


```
//////////////////////////////////////////////////////////////////////////
/// SEED FILTER GENERAL PARAMETERS
/// The parameters below control the process of filtering out seeds before
/// sending them off to track reconstruction. These parameters first correspond
/// to global settings (more loose) followed by more strict cuts for the central
/// and forward/backward regions separately.

float m_maxSeedsPerSpM_filter = 10; // max number of seeds a single middle sp can belong to - 1
float m_deltaRMin = 5* Acts::UnitConstants::mm;
bool m_seedConfirmation = true;
float m_deltaInvHelixDiameter = 0.00003 * 1. / Acts::UnitConstants::mm;
float m_impactWeightFactor = 1.;
float m_zOriginWeightFactor = 1.;
float m_compatSeedWeight = 200.;
size_t m_compatSeedLimit = 2;
bool m_curvatureSortingInFilter = false;
float m_seedWeightIncrement = 0;

///////////////////////////////////////
/// CENTRAL SEED FILTER PARAMETERS
float  m_zMinSeedConf_cent = -250 * Acts::UnitConstants::mm;
float  m_zMaxSeedConf_cent = 250 * Acts::UnitConstants::mm;
float  m_rMaxSeedConf_cent = 140 * Acts::UnitConstants::mm;
size_t m_nTopForLargeR_cent = 1;
size_t m_nTopForSmallR_cent = 2;
float  m_seedConfMinBottomRadius_cent = 60.0 * Acts::UnitConstants::mm;
float  m_seedConfMaxZOrigin_cent = 150.0 * Acts::UnitConstants::mm;
float  m_minImpactSeedConf_cent = 1.0 * Acts::UnitConstants::mm;
```

https://github.com/eic/EICrecon/blob/track-qa-barak/src/algorithms/tracking/OrthogonalTrackSeedingConfig.h

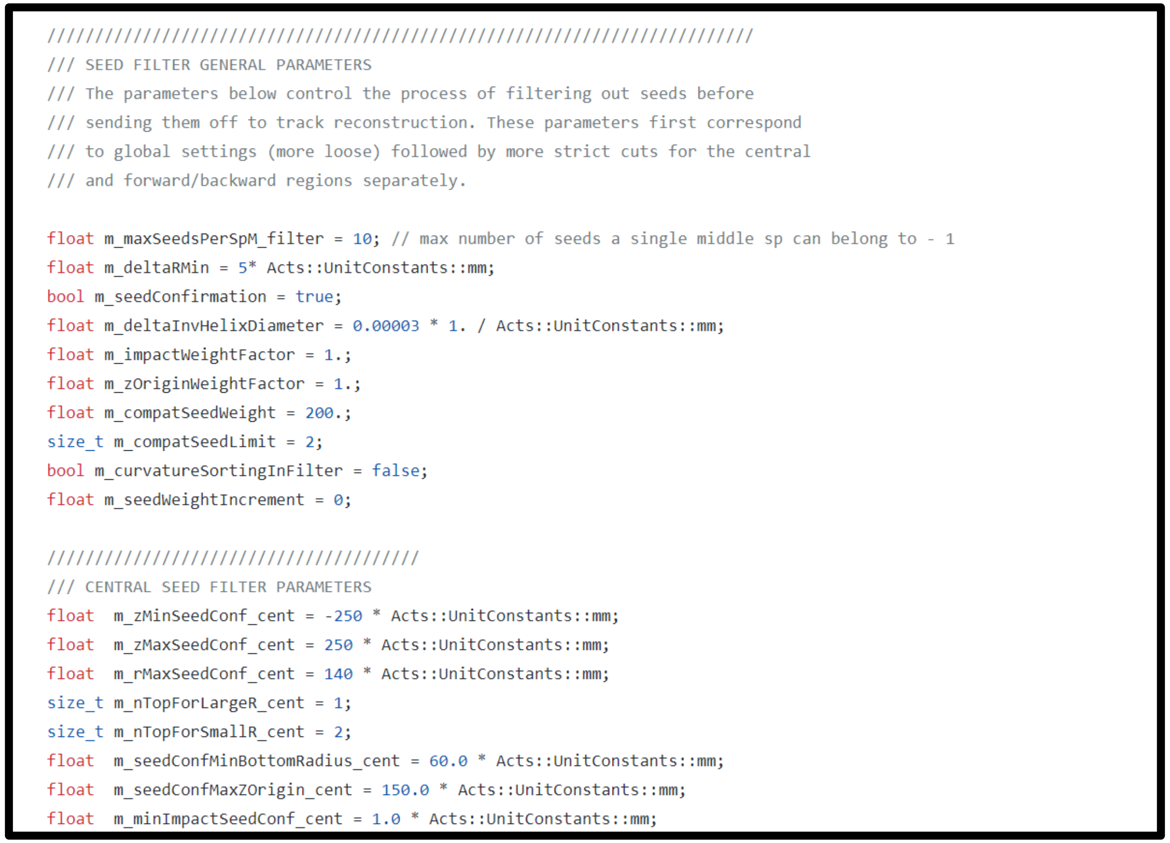
# Seeder configuration file

Implemented seed filter/confirmation options based on suggestion of ACTS expert.



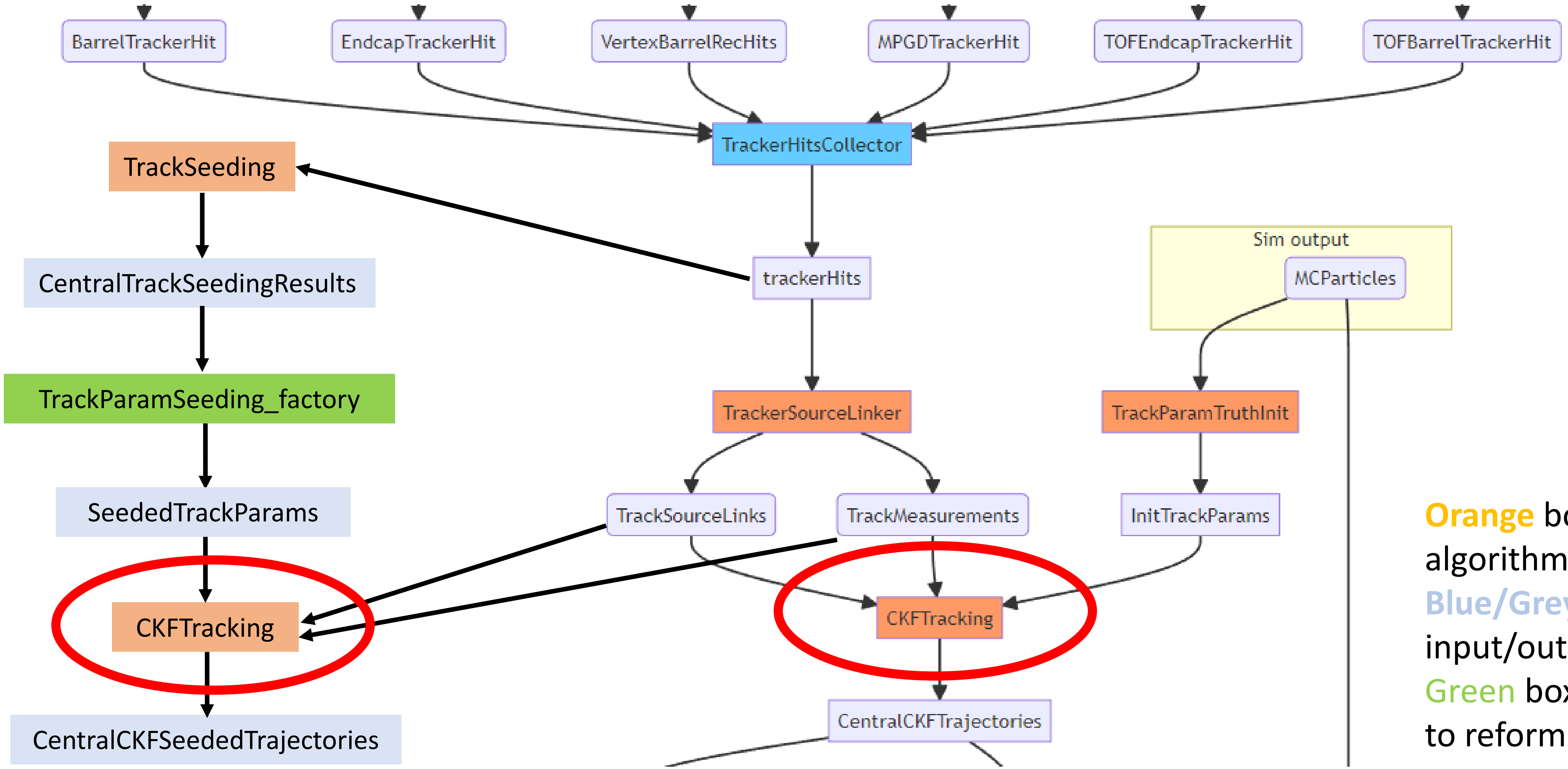
## Seed Filter and Seed Confirmation

```
centralSeedConfirmationRange = acts.SeedConfirmationRang
    zMinSeedConf=-250 * u.mm,
    zMaxSeedConf=250 * u.mm,
    rMaxSeedConf=140 * u.mm,
    nTopForLargeR=1,
    nTopForSmallR=2,
    seedConfMinBottomRadius=60.0 * u.mm,
    seedConfMaxZOrigin=150.0 * u.mm,
    minImpactSeedConf=1.0 * u.mm,
)  # contains parameters for seed confirmation
forwardSeedConfirmationRange = acts.SeedConfirmationRang
    zMinSeedConf=-3000 * u.mm,
    zMaxSeedConf=3000 * u.mm,
    rMaxSeedConf=140 * u.mm,
    nTopForLargeR=1,
    nTopForSmallR=2,
    seedConfMinBottomRadius=60.0 * u.mm,
    seedConfMaxZOrigin=150.0 * u.mm,
    minImpactSeedConf=1.0 * u.mm,
)
```



```
/////////////////////////////////////////////////////////////////////////
/// SEED FILTER GENERAL PARAMETERS
/// The parameters below control the process of filtering out seeds before
/// sending them off to track reconstruction. These parameters first correspond
/// to global settings (more loose) followed by more strict cuts for the central
/// and forward/backward regions separately.

float m_maxSeedsPerSpM_filter = 10; // max number of seeds a single middle sp can belong to - 1
float m_deltaRMin = 5* Acts::UnitConstants::mm;
bool m_seedConfirmation = true;
float m_deltaInvHelixDiameter = 0.00003 * 1. / Acts::UnitConstants::mm;
float m_impactWeightFactor = 1.;
float m_zOriginWeightFactor = 1.;
float m_compatSeedWeight = 200.;
size_t m_compatSeedLimit = 2;
bool m_curvatureSortingInFilter = false;
float m_seedWeightIncrement = 0;

//////////////////////////////////////
/// CENTRAL SEED FILTER PARAMETERS
float  m_zMinSeedConf_cent = -250 * Acts::UnitConstants::mm;
float  m_zMaxSeedConf_cent = 250 * Acts::UnitConstants::mm;
float  m_rMaxSeedConf_cent = 140 * Acts::UnitConstants::mm;
size_t m_nTopForLargeR_cent = 1;
size_t m_nTopForSmallR_cent = 2;
float  m_seedConfMinBottomRadius_cent = 60.0 * Acts::UnitConstants::mm;
float  m_seedConfMaxZOrigin_cent = 150.0 * Acts::UnitConstants::mm;
float  m_minImpactSeedConf_cent = 1.0 * Acts::UnitConstants::mm;
```

# Track fitting

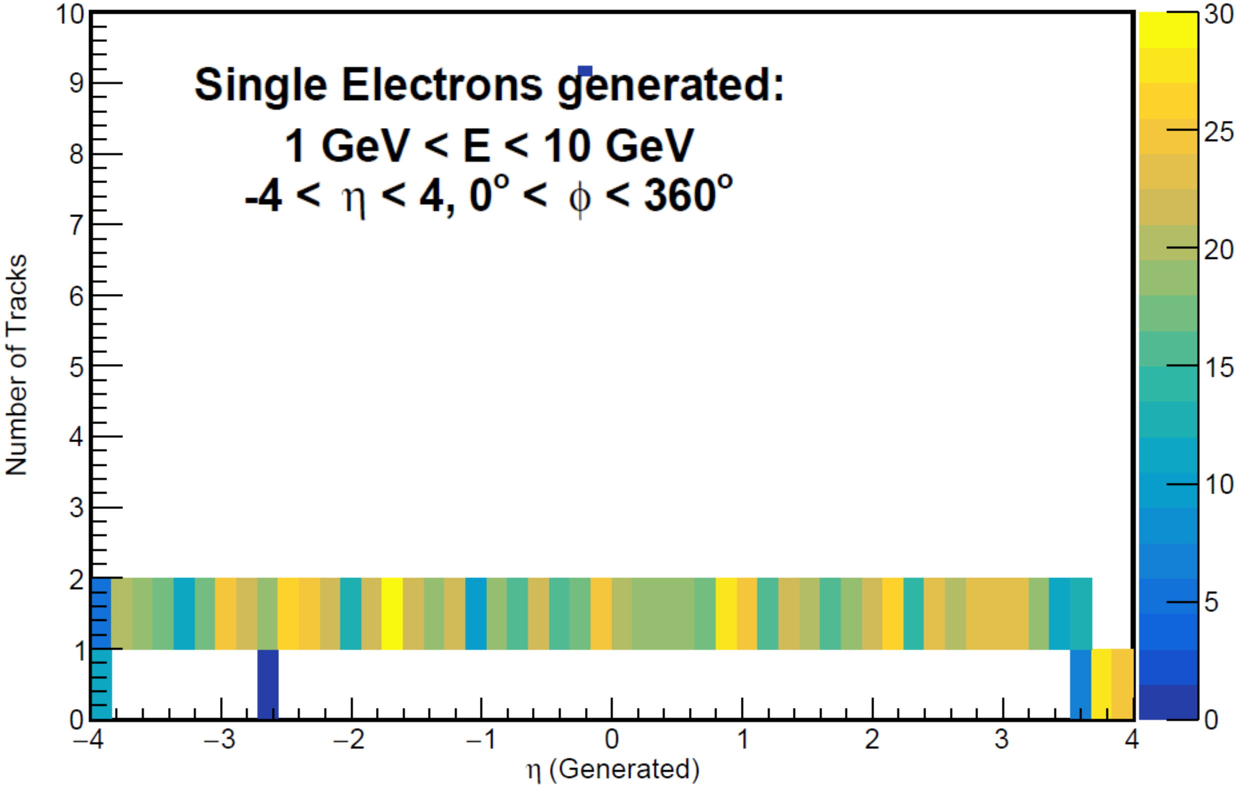


Orange boxes are algorithms.
Blue/Grey boxes are input/output data.
Green box is a factory to reformat data type.

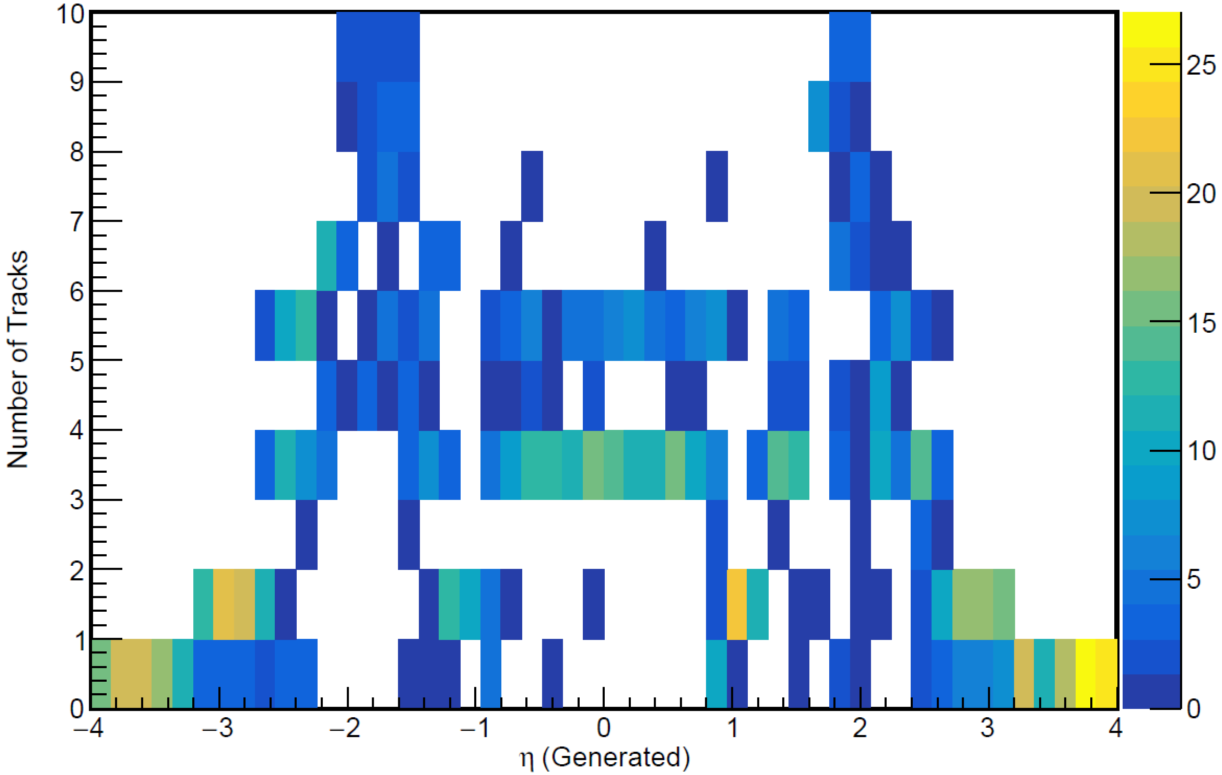
https://eic.github.io/EICrecon/#/design/tracking

# Number of reconstructed tracks

## Truth Seeded

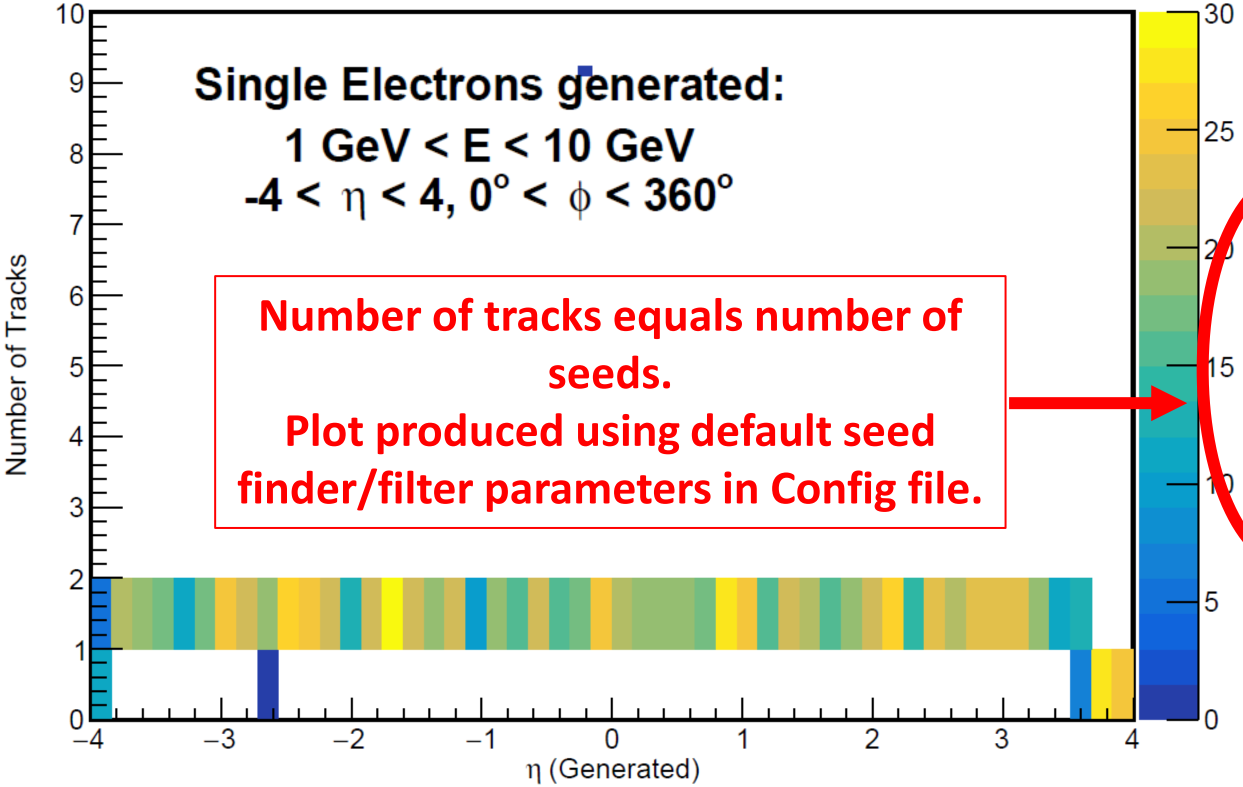


## Real Seeded



# Number of reconstructed tracks

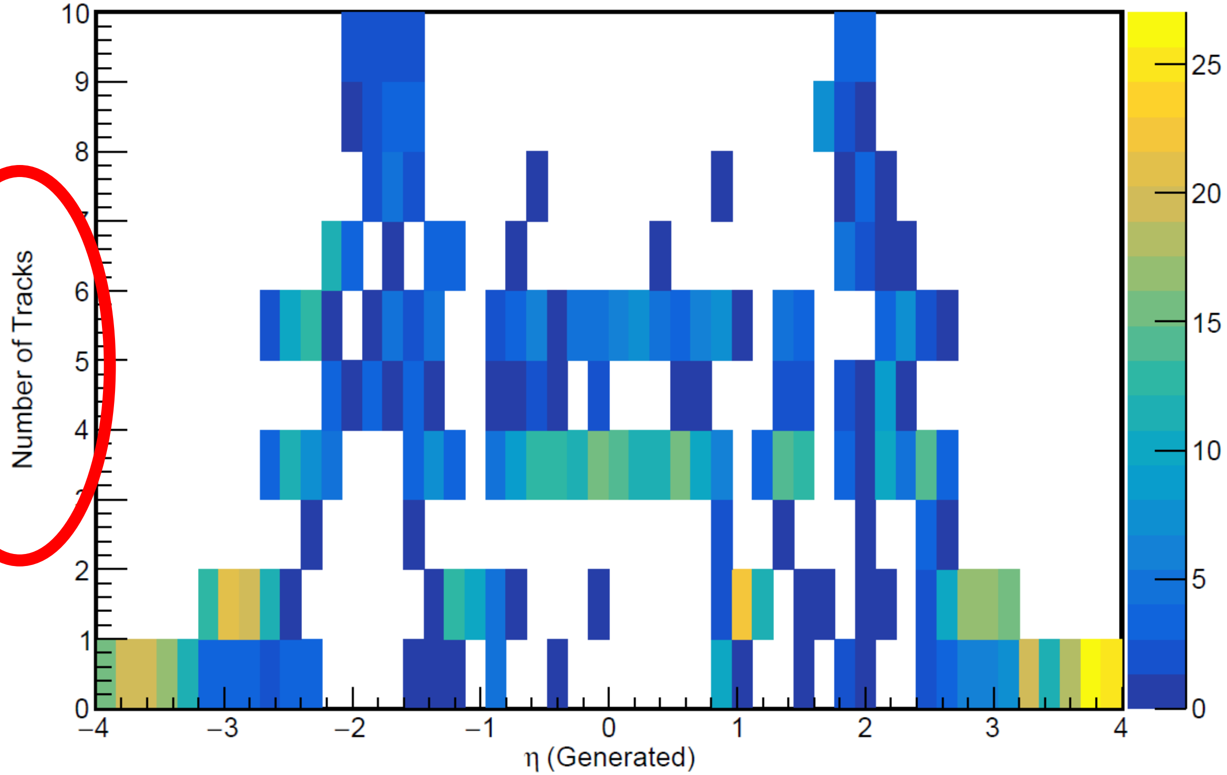
## Truth Seeded



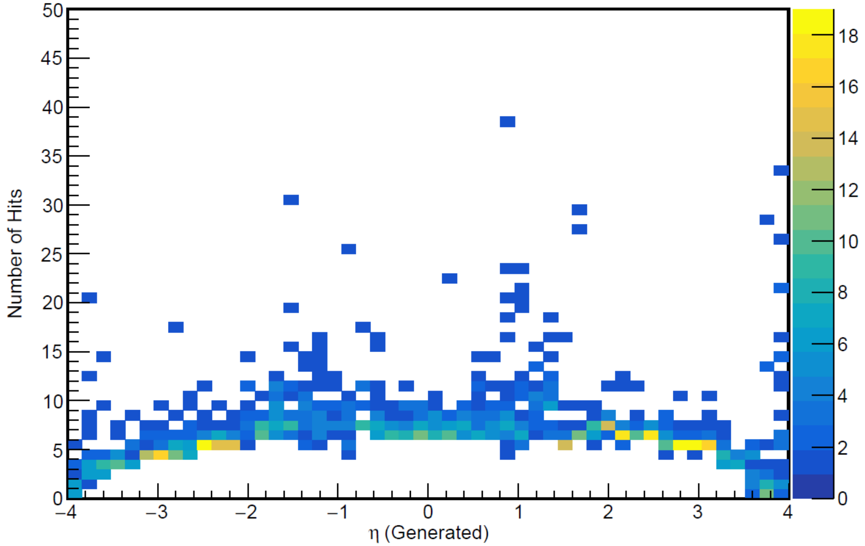
**Single Electrons generated:**
**1 GeV < E < 10 GeV**
**-4 < η < 4, 0° < ϕ < 360°**

**Number of tracks equals number of seeds.**
**Plot produced using default seed finder/filter parameters in Config file.**

## Real Seeded



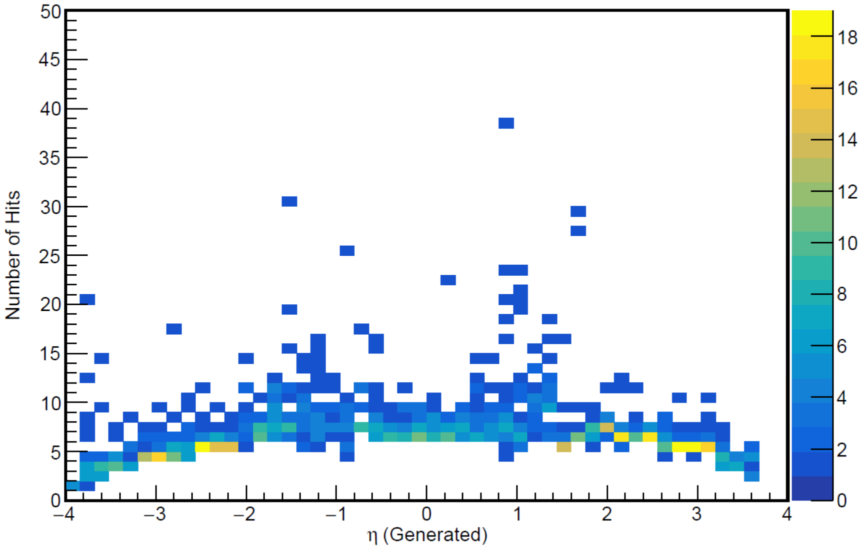
# Hits in tracking detector and track reconstruction



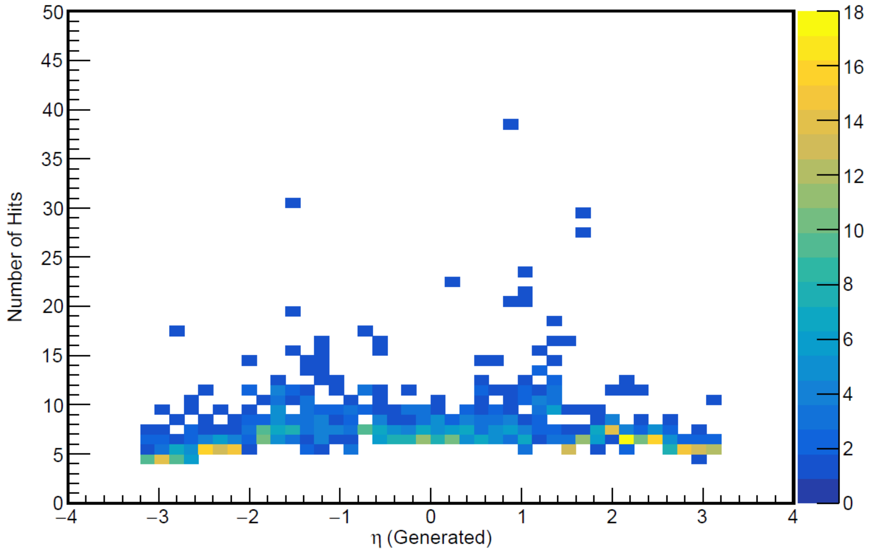
Truth Seeded

Real Seeded

At least one track reconstructed

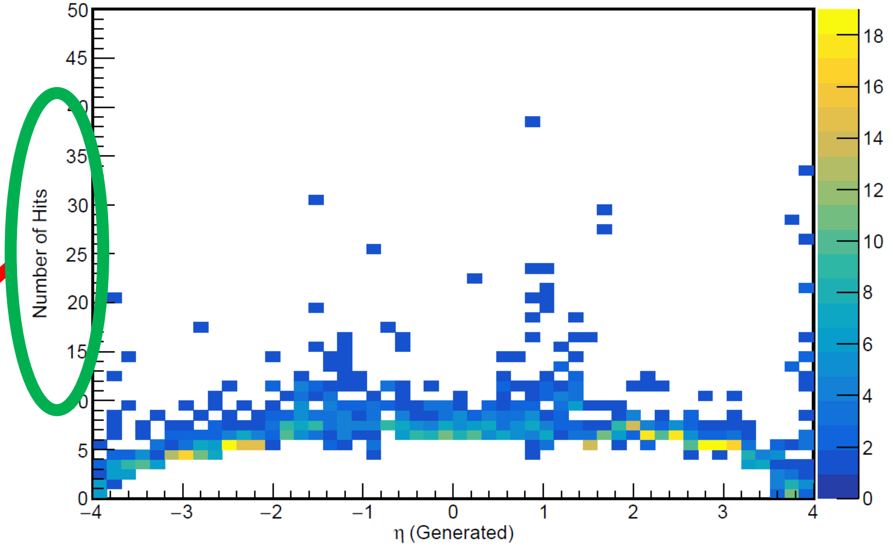


At least one track reconstructed



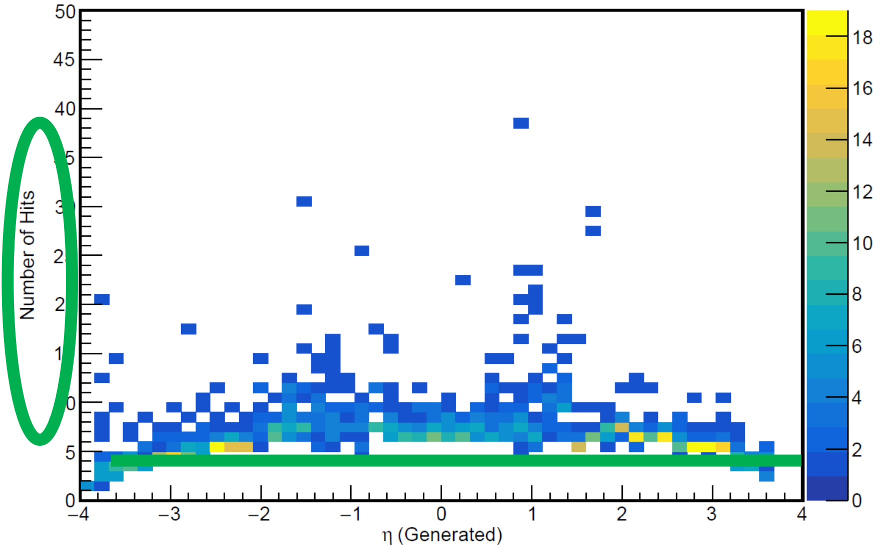
# Hits in tracking detector and track reconstruction

**Truth Seeded**



**Real Seeded**

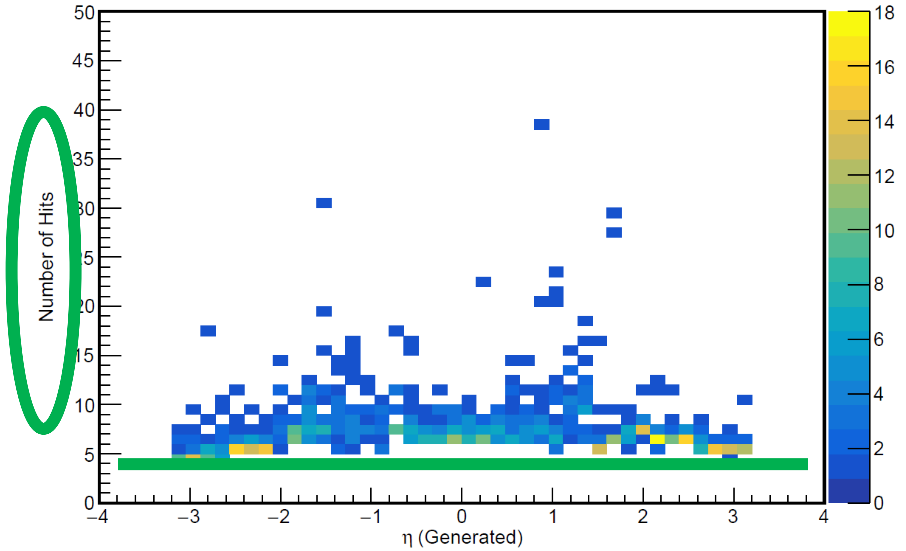
At least one track reconstructed



Total number of digitized hits in tracking detector – not necessarily equal to number of hit used in track fit.

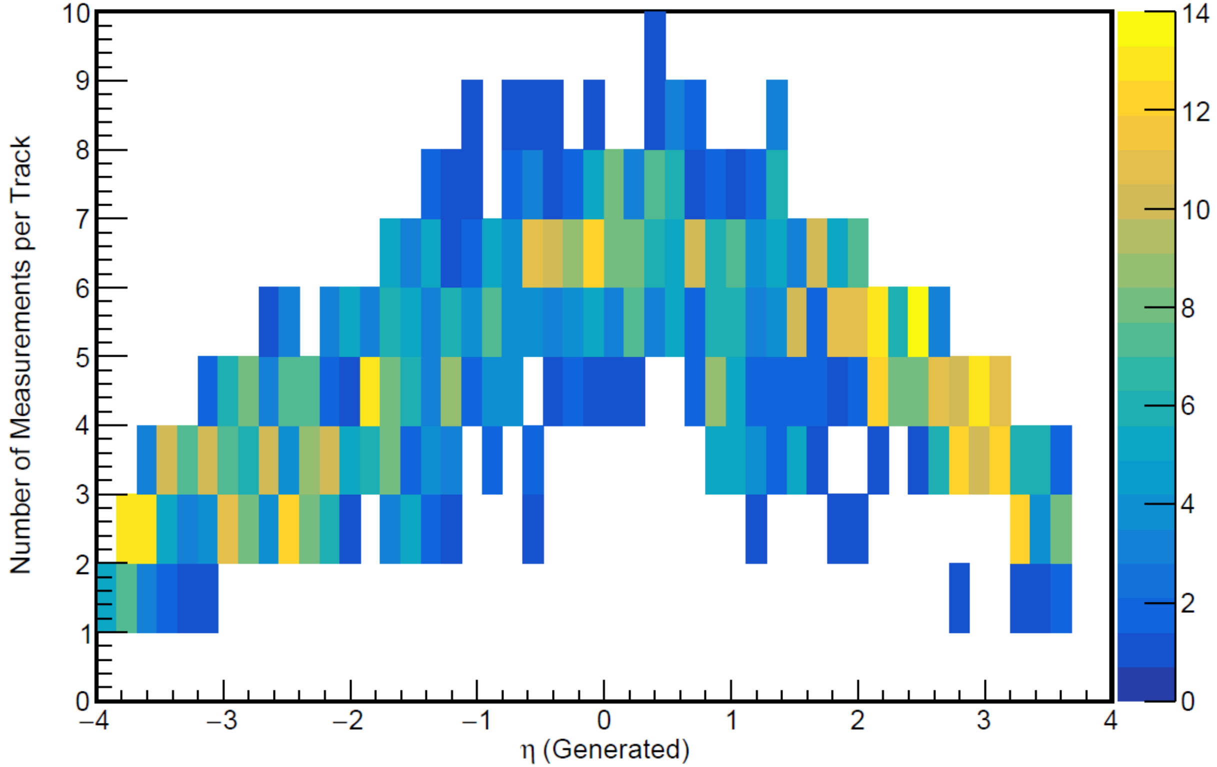
Minimum of 4 hits seen when a real seeded track is reconstructed

At least one track reconstructed

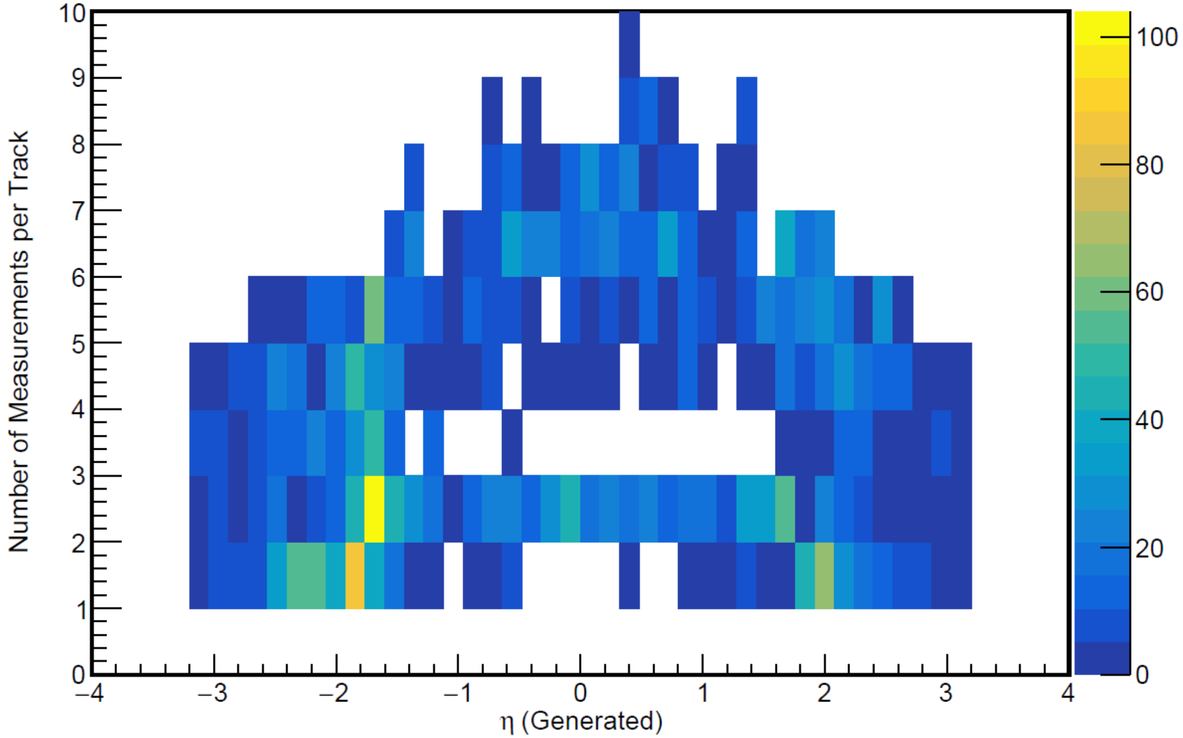


# Number of measurements used per track

## Truth Seeded



## Real Seeded



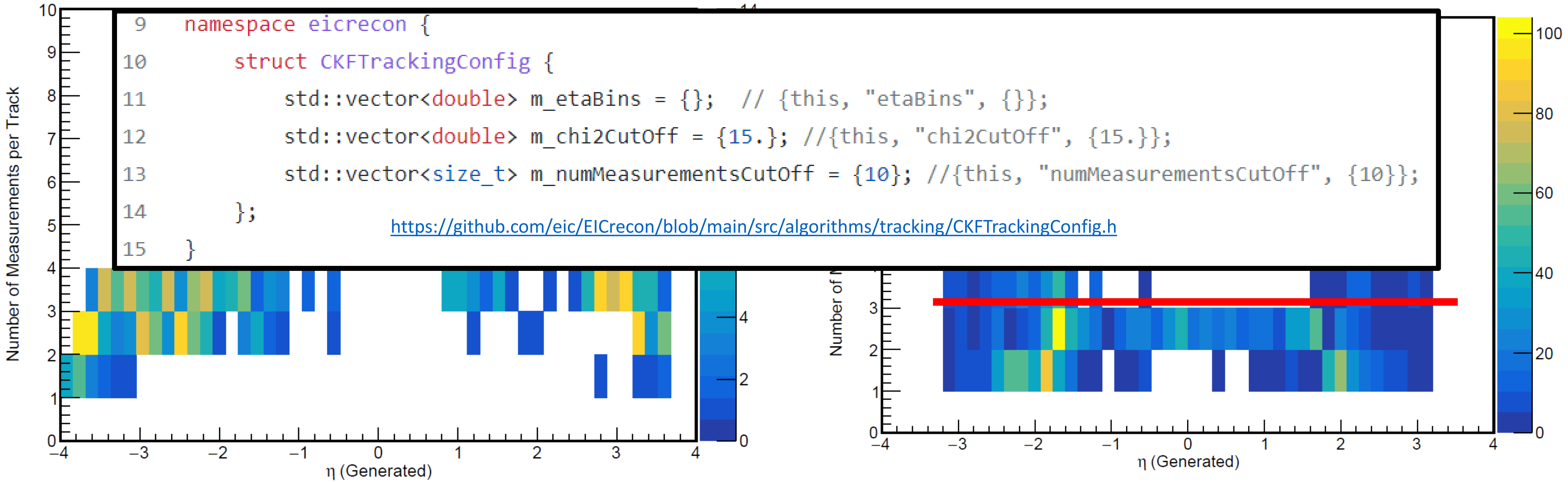
# Number of measurements used per track

**How can the reconstructed tracks have less than 3 measurements, when a seed requiring a triplet has been found?**

**Truth Seeded**

**May need to consider number of outliers.**

**Real Seeded**

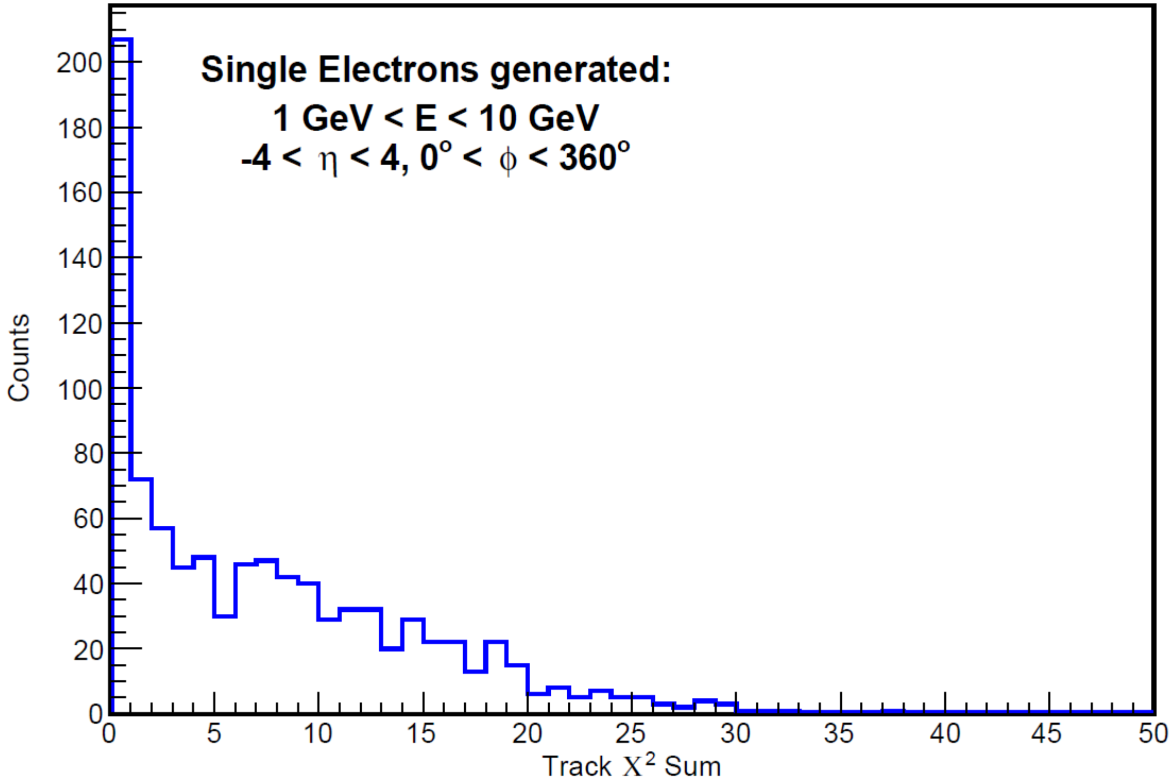


```
namespace eicrecon {
    struct CKFTrackingConfig {
        std::vector<double> m_etaBins = {};  // {this, "etaBins", {}};
        std::vector<double> m_chi2CutOff = {15.}; //{this, "chi2CutOff", {15.}};
        std::vector<size_t> m_numMeasurementsCutOff = {10}; //{this, "numMeasurementsCutOff", {10}};
    };
}
```

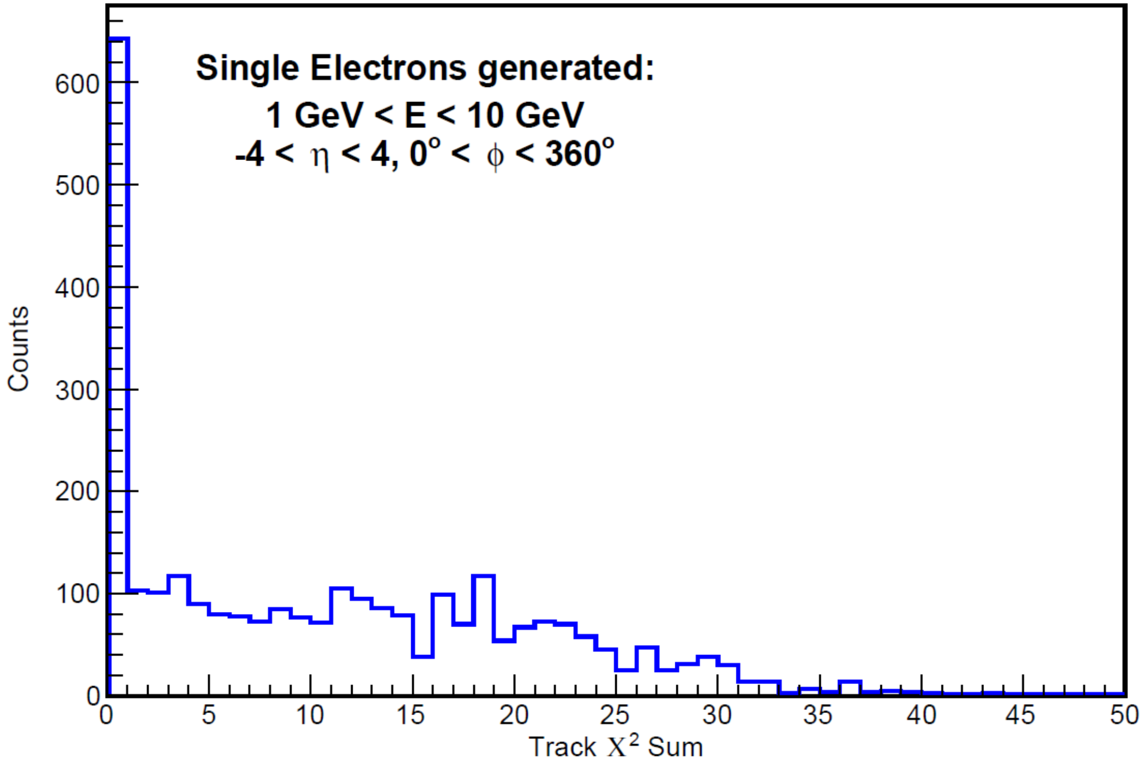
https://github.com/eic/EICrecon/blob/main/src/algorithms/tracking/CKFTrackingConfig.h

# Additional track information: Track $\chi^2$

## Truth Seeded

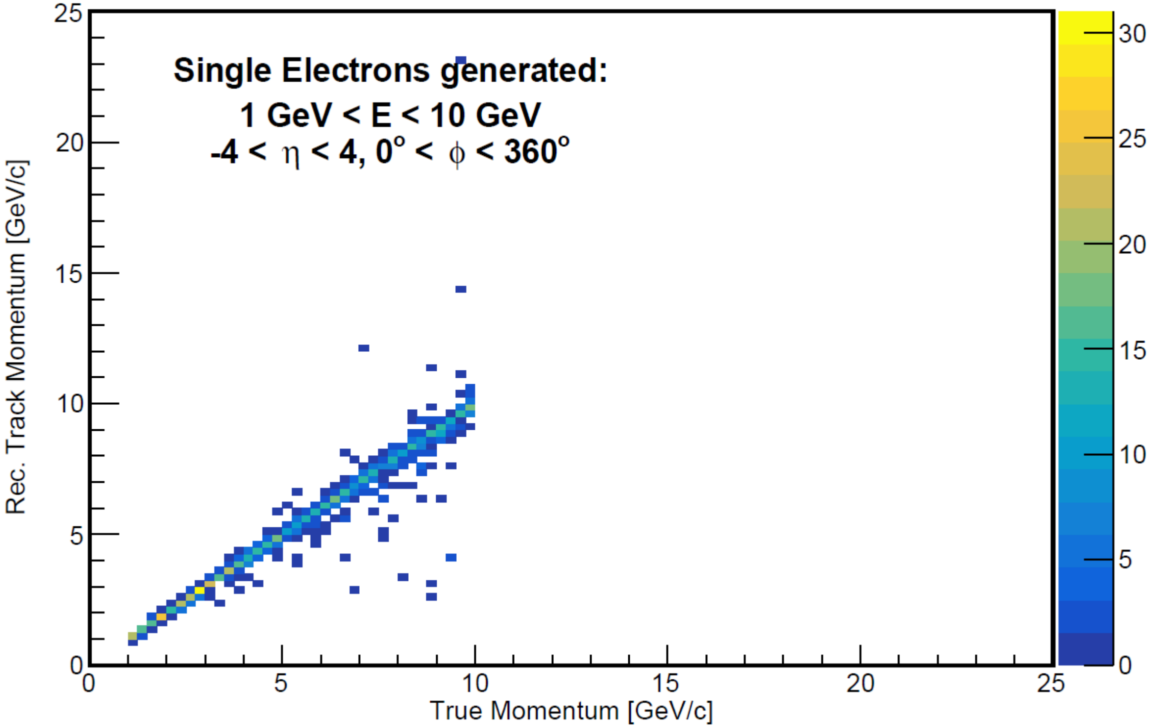


## Real Seeded

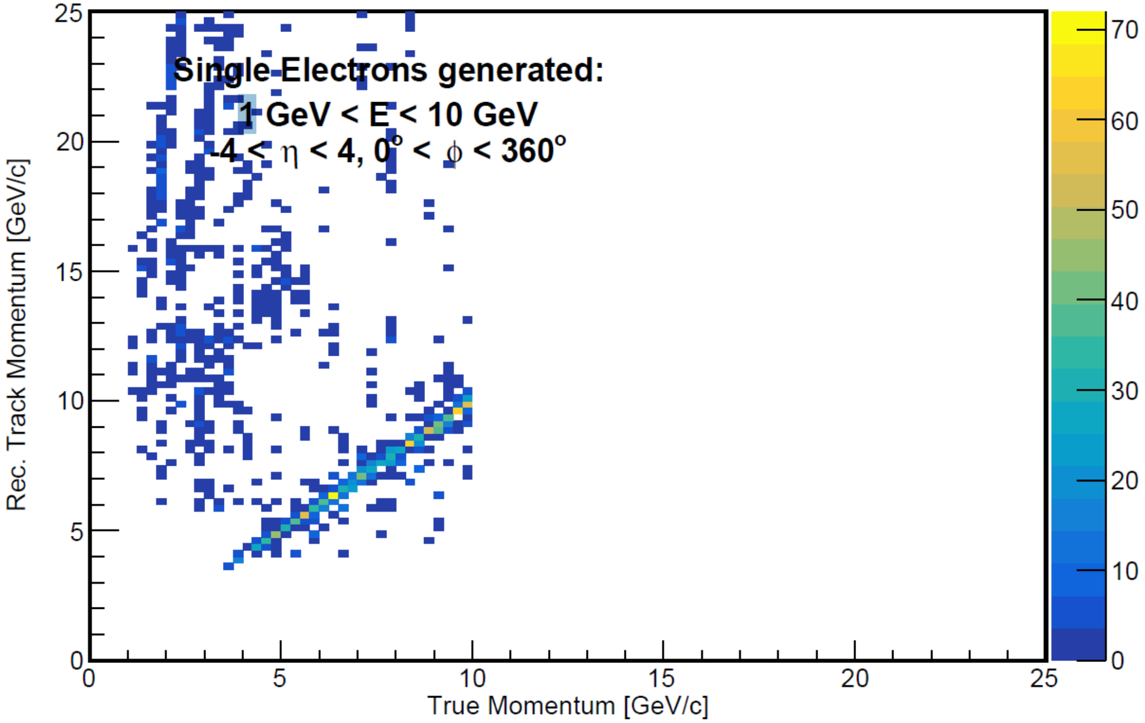


# Additional track information: Track momentum resolution

## Truth Seeded



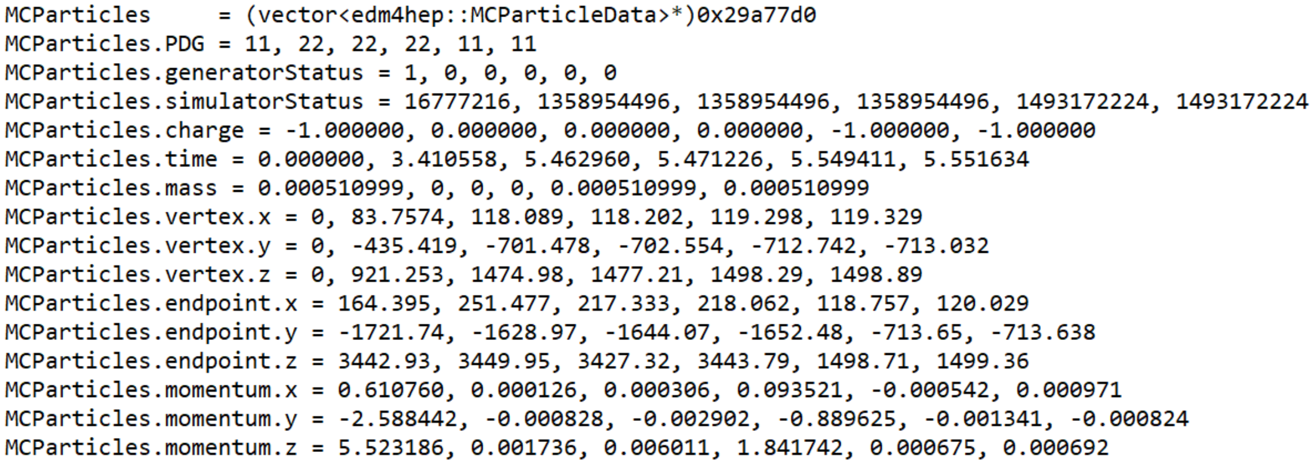
## Real Seeded



# Track hit and generated particle association

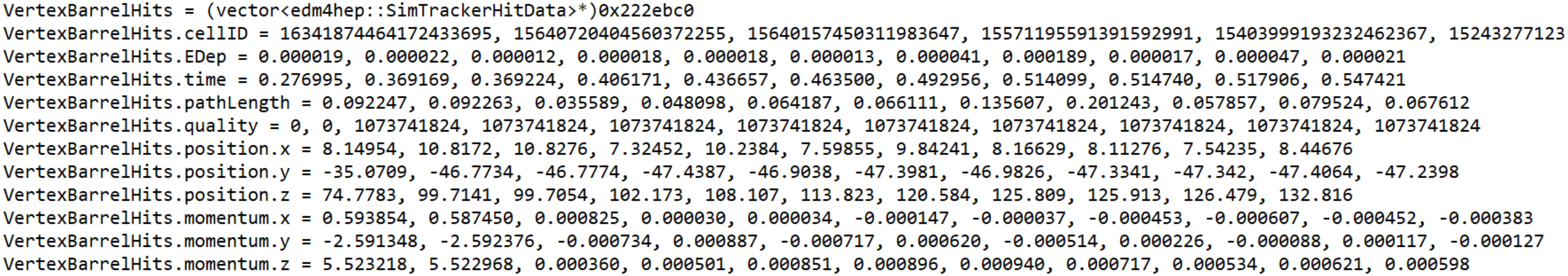
- For the ongoing seeding studies/parameter optimization, it would be useful to have and easy way to associate the digitized hit with the MC particle(s) which caused the hit.
- This can be done to some extent now for single particle events, but it is not very user-friendly. Shyam is developing a code that would allow us to do this more easily.

# Track hit and generated particle association



```
MCParticles      = (vector<edm4hep::MCParticleData>*)0x29a77d0
MCParticles.PDG = 11, 22, 22, 22, 11, 11
MCParticles.generatorStatus = 1, 0, 0, 0, 0, 0
MCParticles.simulatorStatus = 16777216, 1358954496, 1358954496, 1358954496, 1493172224, 1493172224
MCParticles.charge = -1.000000, 0.000000, 0.000000, 0.000000, -1.000000, -1.000000
MCParticles.time = 0.000000, 3.410558, 5.462960, 5.471226, 5.549411, 5.551634
MCParticles.mass = 0.000510999, 0, 0, 0, 0.000510999, 0.000510999
MCParticles.vertex.x = 0, 83.7574, 118.089, 118.202, 119.298, 119.329
MCParticles.vertex.y = 0, -435.419, -701.478, -702.554, -712.742, -713.032
MCParticles.vertex.z = 0, 921.253, 1474.98, 1477.21, 1498.29, 1498.89
MCParticles.endpoint.x = 164.395, 251.477, 217.333, 218.062, 118.757, 120.029
MCParticles.endpoint.y = -1721.74, -1628.97, -1644.07, -1652.48, -713.65, -713.638
MCParticles.endpoint.z = 3442.93, 3449.95, 3427.32, 3443.79, 1498.71, 1499.36
MCParticles.momentum.x = 0.610760, 0.000126, 0.000306, 0.093521, -0.000542, 0.000971
MCParticles.momentum.y = -2.588442, -0.000828, -0.002902, -0.889625, -0.001341, -0.000824
MCParticles.momentum.z = 5.523186, 0.001736, 0.006011, 1.841742, 0.000675, 0.000692
```

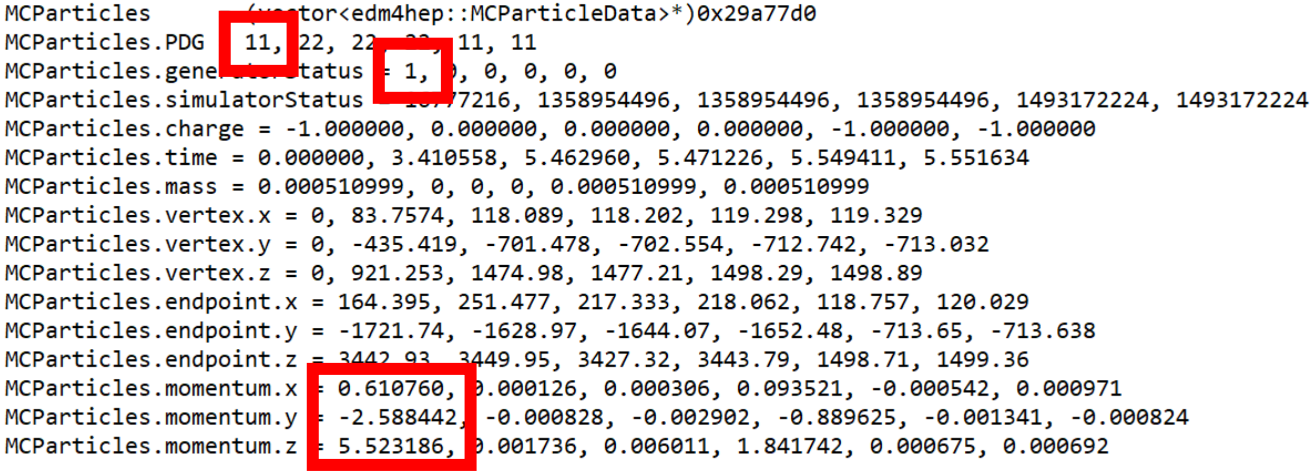
**Monte Carlo particles – both primary and saved secondaries**



```
VertexBarrelHits = (vector<edm4hep::SimTrackerHitData>*)0x222ebc0
VertexBarrelHits.cellID = 16341874464172433695, 15640720404560372255, 15640157450311983647, 15571195591391592991, 15403999193232462367, 15243277123
VertexBarrelHits.EDep = 0.000019, 0.000022, 0.000012, 0.000018, 0.000018, 0.000013, 0.000041, 0.000189, 0.000017, 0.000047, 0.000021
VertexBarrelHits.time = 0.276995, 0.369169, 0.369224, 0.406171, 0.436657, 0.463500, 0.492956, 0.514099, 0.514740, 0.517906, 0.547421
VertexBarrelHits.pathLength = 0.092247, 0.092263, 0.035589, 0.048098, 0.064187, 0.066111, 0.135607, 0.201243, 0.057857, 0.079524, 0.067612
VertexBarrelHits.quality = 0, 0, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824
VertexBarrelHits.position.x = 8.14954, 10.8172, 10.8276, 7.32452, 10.2384, 7.59855, 9.84241, 8.16629, 8.11276, 7.54235, 8.44676
VertexBarrelHits.position.y = -35.0709, -46.7734, -46.7774, -47.4387, -46.9038, -47.3981, -46.9826, -47.3341, -47.342, -47.4064, -47.2398
VertexBarrelHits.position.z = 74.7783, 99.7141, 99.7054, 102.173, 108.107, 113.823, 120.584, 125.809, 125.913, 126.479, 132.816
VertexBarrelHits.momentum.x = 0.593854, 0.587450, 0.000825, 0.000030, 0.000034, -0.000147, -0.000037, -0.000453, -0.000607, -0.000452, -0.000383
VertexBarrelHits.momentum.y = -2.591348, -2.592376, -0.000734, 0.000887, -0.000717, 0.000620, -0.000514, 0.000226, -0.000088, 0.000117, -0.000127
VertexBarrelHits.momentum.z = 5.523218, 5.522968, 0.000360, 0.000501, 0.000851, 0.000896, 0.000940, 0.000717, 0.000534, 0.000621, 0.000598
```

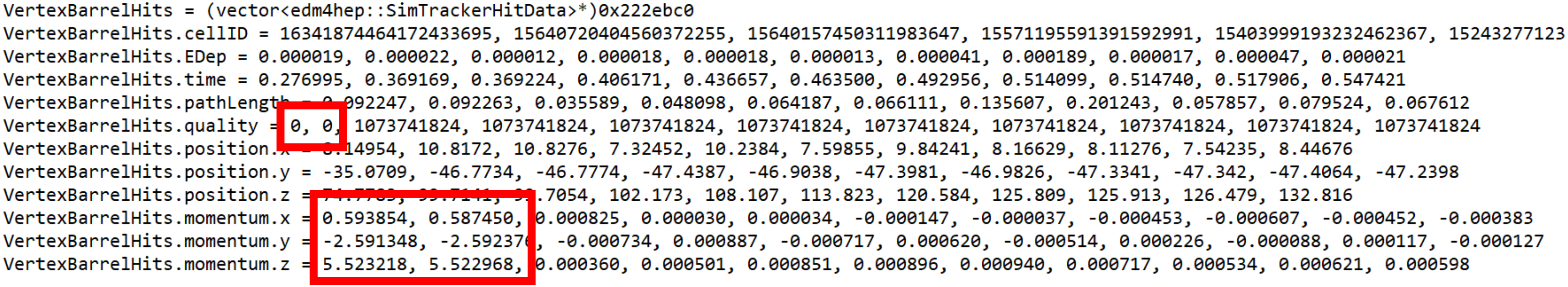
**Geant-level hits in the inner Si vertex layers**

# Track hit and generated particle association



```
MCParticles        (vector<edm4hep::MCParticleData>*)0x29a77d0
MCParticles.PDG  11, 22, 22, 22, 11, 11
MCParticles.generatorStatus = 1, 0, 0, 0, 0, 0
MCParticles.simulatorStatus = 16777216, 1358954496, 1358954496, 1358954496, 1493172224, 1493172224
MCParticles.charge = -1.000000, 0.000000, 0.000000, 0.000000, -1.000000, -1.000000
MCParticles.time = 0.000000, 3.410558, 5.462960, 5.471226, 5.549411, 5.551634
MCParticles.mass = 0.000510999, 0, 0, 0, 0.000510999, 0.000510999
MCParticles.vertex.x = 0, 83.7574, 118.089, 118.202, 119.298, 119.329
MCParticles.vertex.y = 0, -435.419, -701.478, -702.554, -712.742, -713.032
MCParticles.vertex.z = 0, 921.253, 1474.98, 1477.21, 1498.29, 1498.89
MCParticles.endpoint.x = 164.395, 251.477, 217.333, 218.062, 118.757, 120.029
MCParticles.endpoint.y = -1721.74, -1628.97, -1644.07, -1652.48, -713.65, -713.638
MCParticles.endpoint.z = 3442.93, 3449.95, 3427.32, 3443.79, 1498.71, 1499.36
MCParticles.momentum.x = 0.610760, 0.000126, 0.000306, 0.093521, -0.000542, 0.000971
MCParticles.momentum.y = -2.588442, -0.000828, -0.002902, -0.889625, -0.001341, -0.000824
MCParticles.momentum.z = 5.523186, 0.001736, 0.006011, 1.841742, 0.000675, 0.000692
```

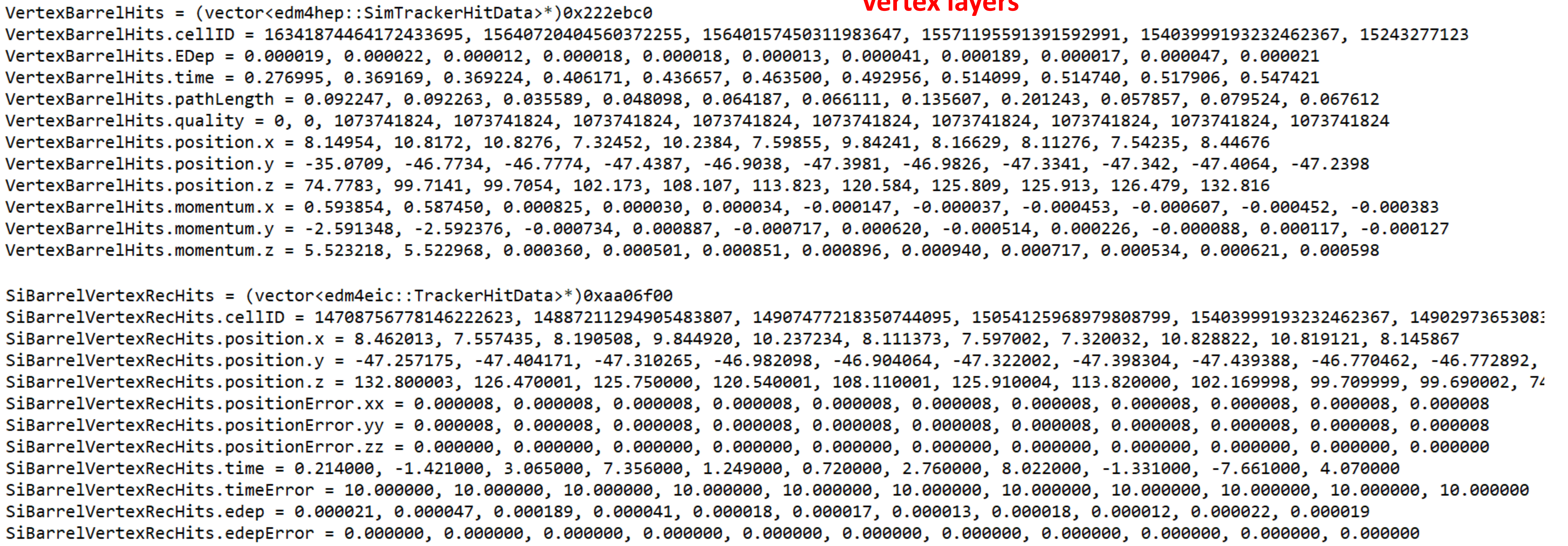
We see that one primary particle has been generated.
By comparing the hit 'quality' or momentum, we can see which hits come from the primary particle. (This way probably wouldn't work with multiple primary particles.)



```
VertexBarrelHits = (vector<edm4hep::SimTrackerHitData>*)0x222ebc0
VertexBarrelHits.cellID = 16341874464172433695, 15640720404560372255, 15640157450311983647, 15571195591391592991, 15403999193232462367, 15243277123
VertexBarrelHits.EDep = 0.000019, 0.000022, 0.000012, 0.000018, 0.000018, 0.000013, 0.000041, 0.000189, 0.000017, 0.000047, 0.000021
VertexBarrelHits.time = 0.276995, 0.369169, 0.369224, 0.406171, 0.436657, 0.463500, 0.492956, 0.514099, 0.514740, 0.517906, 0.547421
VertexBarrelHits.pathLength = 0.092247, 0.092263, 0.035589, 0.048098, 0.064187, 0.066111, 0.135607, 0.201243, 0.057857, 0.079524, 0.067612
VertexBarrelHits.quality = 0, 0, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824
VertexBarrelHits.position.x = 8.14954, 10.8172, 10.8276, 7.32452, 10.2384, 7.59855, 9.84241, 8.16629, 8.11276, 7.54235, 8.44676
VertexBarrelHits.position.y = -35.0709, -46.7734, -46.7774, -47.4387, -46.9038, -47.3981, -46.9826, -47.3341, -47.342, -47.4064, -47.2398
VertexBarrelHits.position.z = 74.7783, 99.7141, 99.7054, 102.173, 108.107, 113.823, 120.584, 125.809, 125.913, 126.479, 132.816
VertexBarrelHits.momentum.x = 0.593854, 0.587450, 0.000825, 0.000030, 0.000034, -0.000147, -0.000037, -0.000453, -0.000607, -0.000452, -0.000383
VertexBarrelHits.momentum.y = -2.591348, -2.592376, -0.000734, 0.000887, -0.000717, 0.000620, -0.000514, 0.000226, -0.000088, 0.000117, -0.000127
VertexBarrelHits.momentum.z = 5.523218, 5.522968, 0.000360, 0.000501, 0.000851, 0.000896, 0.000940, 0.000717, 0.000534, 0.000621, 0.000598
```

# Track hit and generated particle association

**Geant-level hits in the inner Si vertex layers**

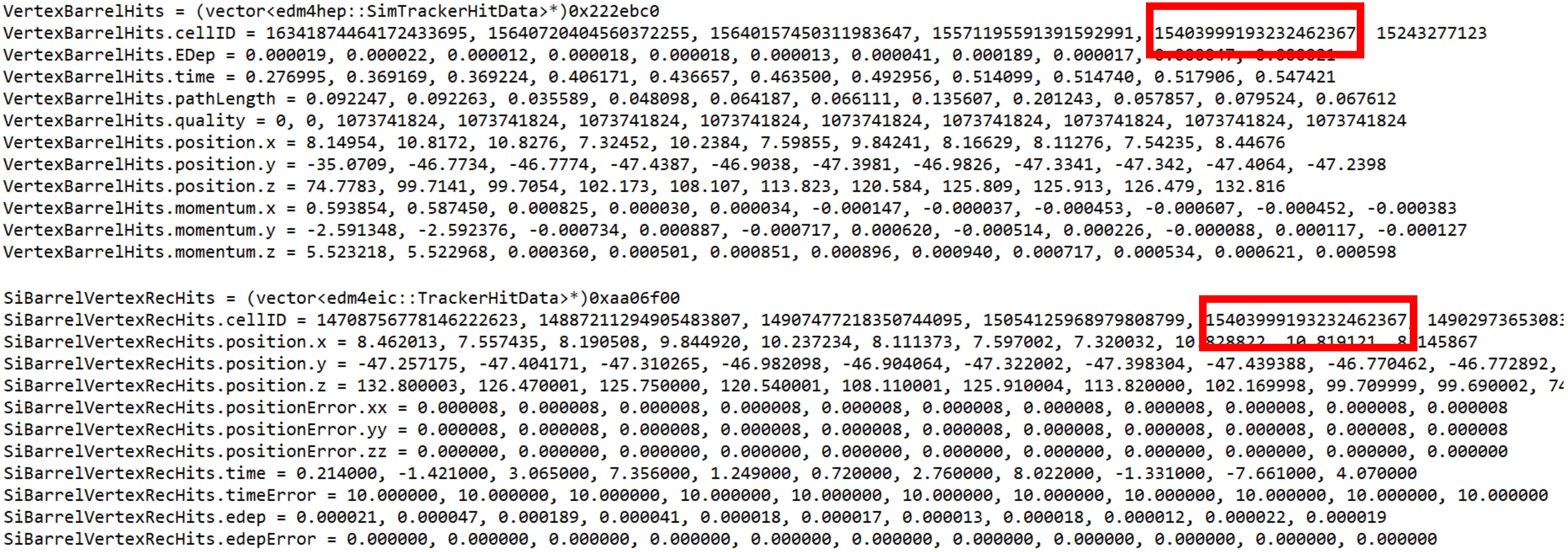


```
VertexBarrelHits = (vector<edm4hep::SimTrackerHitData>*)0x222ebc0
VertexBarrelHits.cellID = 16341874464172433695, 15640720404560372255, 15640157450311983647, 15571195591391592991, 15403999193232462367, 15243277123
VertexBarrelHits.EDep = 0.000019, 0.000022, 0.000012, 0.000018, 0.000018, 0.000013, 0.000041, 0.000189, 0.000017, 0.000047, 0.000021
VertexBarrelHits.time = 0.276995, 0.369169, 0.369224, 0.406171, 0.436657, 0.463500, 0.492956, 0.514099, 0.514740, 0.517906, 0.547421
VertexBarrelHits.pathLength = 0.092247, 0.092263, 0.035589, 0.048098, 0.064187, 0.066111, 0.135607, 0.201243, 0.057857, 0.079524, 0.067612
VertexBarrelHits.quality = 0, 0, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824
VertexBarrelHits.position.x = 8.14954, 10.8172, 10.8276, 7.32452, 10.2384, 7.59855, 9.84241, 8.16629, 8.11276, 7.54235, 8.44676
VertexBarrelHits.position.y = -35.0709, -46.7734, -46.7774, -47.4387, -46.9038, -47.3981, -46.9826, -47.3341, -47.342, -47.4064, -47.2398
VertexBarrelHits.position.z = 74.7783, 99.7141, 99.7054, 102.173, 108.107, 113.823, 120.584, 125.809, 125.913, 126.479, 132.816
VertexBarrelHits.momentum.x = 0.593854, 0.587450, 0.000825, 0.000030, 0.000034, -0.000147, -0.000037, -0.000453, -0.000607, -0.000452, -0.000383
VertexBarrelHits.momentum.y = -2.591348, -2.592376, -0.000734, 0.000887, -0.000717, 0.000620, -0.000514, 0.000226, -0.000088, 0.000117, -0.000127
VertexBarrelHits.momentum.z = 5.523218, 5.522968, 0.000360, 0.000501, 0.000851, 0.000896, 0.000940, 0.000717, 0.000534, 0.000621, 0.000598

SiBarrelVertexRecHits = (vector<edm4eic::TrackerHitData>*)0xaa06f00
SiBarrelVertexRecHits.cellID = 14708756778146222623, 14887211294905483807, 14907477218350744095, 15054125968979808799, 15403999193232462367, 1490297365308
SiBarrelVertexRecHits.position.x = 8.462013, 7.557435, 8.190508, 9.844920, 10.237234, 8.111373, 7.597002, 7.320032, 10.828822, 10.819121, 8.145867
SiBarrelVertexRecHits.position.y = -47.257175, -47.404171, -47.310265, -46.982098, -46.904064, -47.322002, -47.398304, -47.439388, -46.770462, -46.772892,
SiBarrelVertexRecHits.position.z = 132.800003, 126.470001, 125.750000, 120.540001, 108.110001, 125.910004, 113.820000, 102.169998, 99.709999, 99.690002, 74
SiBarrelVertexRecHits.positionError.xx = 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008
SiBarrelVertexRecHits.positionError.yy = 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008
SiBarrelVertexRecHits.positionError.zz = 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000
SiBarrelVertexRecHits.time = 0.214000, -1.421000, 3.065000, 7.356000, 1.249000, 0.720000, 2.760000, 8.022000, -1.331000, -7.661000, 4.070000
SiBarrelVertexRecHits.timeError = 10.000000, 10.000000, 10.000000, 10.000000, 10.000000, 10.000000, 10.000000, 10.000000, 10.000000, 10.000000, 10.000000
SiBarrelVertexRecHits.edep = 0.000021, 0.000047, 0.000189, 0.000041, 0.000018, 0.000017, 0.000013, 0.000018, 0.000012, 0.000022, 0.000019
SiBarrelVertexRecHits.edepError = 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000
```

**Digitized hits in the same detector**

# Track hit and generated particle association

**We can match based on cell ID and then use above method to associate with the primary particle. A simpler way would be better.**



```
VertexBarrelHits = (vector<edm4hep::SimTrackerHitData>*)0x222ebc0
VertexBarrelHits.cellID = 16341874464172433695, 15640720404560372255, 15640157450311983647, 15571195591391592991, 1540399193232462367  15243277123
VertexBarrelHits.EDep = 0.000019, 0.000022, 0.000012, 0.000018, 0.000018, 0.000013, 0.000041, 0.000189, 0.000017, 0.000047, 0.000021
VertexBarrelHits.time = 0.276995, 0.369169, 0.369224, 0.406171, 0.436657, 0.463500, 0.492956, 0.514099, 0.514740, 0.517906, 0.547421
VertexBarrelHits.pathLength = 0.092247, 0.092263, 0.035589, 0.048098, 0.064187, 0.066111, 0.135607, 0.201243, 0.057857, 0.079524, 0.067612
VertexBarrelHits.quality = 0, 0, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824, 1073741824
VertexBarrelHits.position.x = 8.14954, 10.8172, 10.8276, 7.32452, 10.2384, 7.59855, 9.84241, 8.16629, 8.11276, 7.54235, 8.44676
VertexBarrelHits.position.y = -35.0709, -46.7734, -46.7774, -47.4387, -46.9038, -47.3981, -46.9826, -47.3341, -47.342, -47.4064, -47.2398
VertexBarrelHits.position.z = 74.7783, 99.7141, 99.7054, 102.173, 108.107, 113.823, 120.584, 125.809, 125.913, 126.479, 132.816
VertexBarrelHits.momentum.x = 0.593854, 0.587450, 0.000825, 0.000030, 0.000034, -0.000147, -0.000037, -0.000453, -0.000607, -0.000452, -0.000383
VertexBarrelHits.momentum.y = -2.591348, -2.592376, -0.000734, 0.000887, -0.000717, 0.000620, -0.000514, 0.000226, -0.000088, 0.000117, -0.000127
VertexBarrelHits.momentum.z = 5.523218, 5.522968, 0.000360, 0.000501, 0.000851, 0.000896, 0.000940, 0.000717, 0.000534, 0.000621, 0.000598

SiBarrelVertexRecHits = (vector<edm4eic::TrackerHitData>*)0xaa06f00
SiBarrelVertexRecHits.cellID = 14708756778146222623, 14887211294905483807, 14907477218350744095, 15054125968979808799, 1540399193232462367, 1490297365308
SiBarrelVertexRecHits.position.x = 8.462013, 7.557435, 8.190508, 9.844920, 10.237234, 8.111373, 7.597002, 7.320032, 10.828822, 10.819121, 8.145867
SiBarrelVertexRecHits.position.y = -47.257175, -47.404171, -47.310265, -46.982098, -46.904064, -47.322002, -47.398304, -47.439388, -46.770462, -46.772892,
SiBarrelVertexRecHits.position.z = 132.800003, 126.470001, 125.750000, 120.540001, 108.110001, 125.910004, 113.820000, 102.169998, 99.709999, 99.690002, 74
SiBarrelVertexRecHits.positionError.xx = 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008
SiBarrelVertexRecHits.positionError.yy = 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008, 0.000008
SiBarrelVertexRecHits.positionError.zz = 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000
SiBarrelVertexRecHits.time = 0.214000, -1.421000, 3.065000, 7.356000, 1.249000, 0.720000, 2.760000, 8.022000, -1.331000, -7.661000, 4.070000
SiBarrelVertexRecHits.timeError = 10.000000, 10.000000, 10.000000, 10.000000, 10.000000, 10.000000, 10.000000, 10.000000, 10.000000, 10.000000, 10.000000
SiBarrelVertexRecHits.edep = 0.000021, 0.000047, 0.000189, 0.000041, 0.000018, 0.000017, 0.000013, 0.000018, 0.000012, 0.000022, 0.000019
SiBarrelVertexRecHits.edepError = 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000
```

# Summary

- ➢We have developed the machinery to use the real seeds in the track fitting.
- ➢The real-seeded tracks can be accessed in an EICRecon Plugin. A user-controlled flag is being developed to switch between the seeding types – see next talk by Dmitry.
- ➢We are working on optimizing the seed parameters.