Effect of creation vertex on single-particle reconstruction

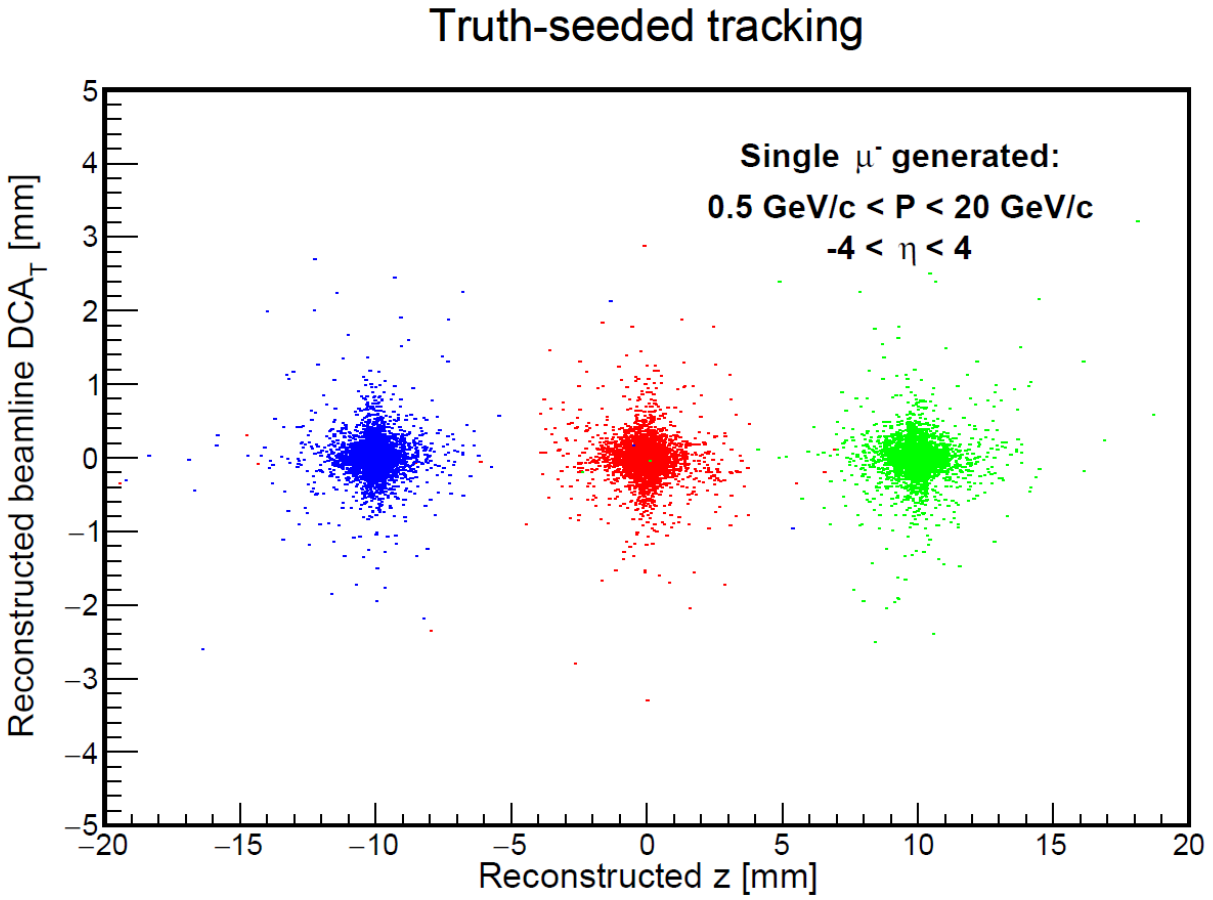
Barak Schmookler, Jeetendra Gupta

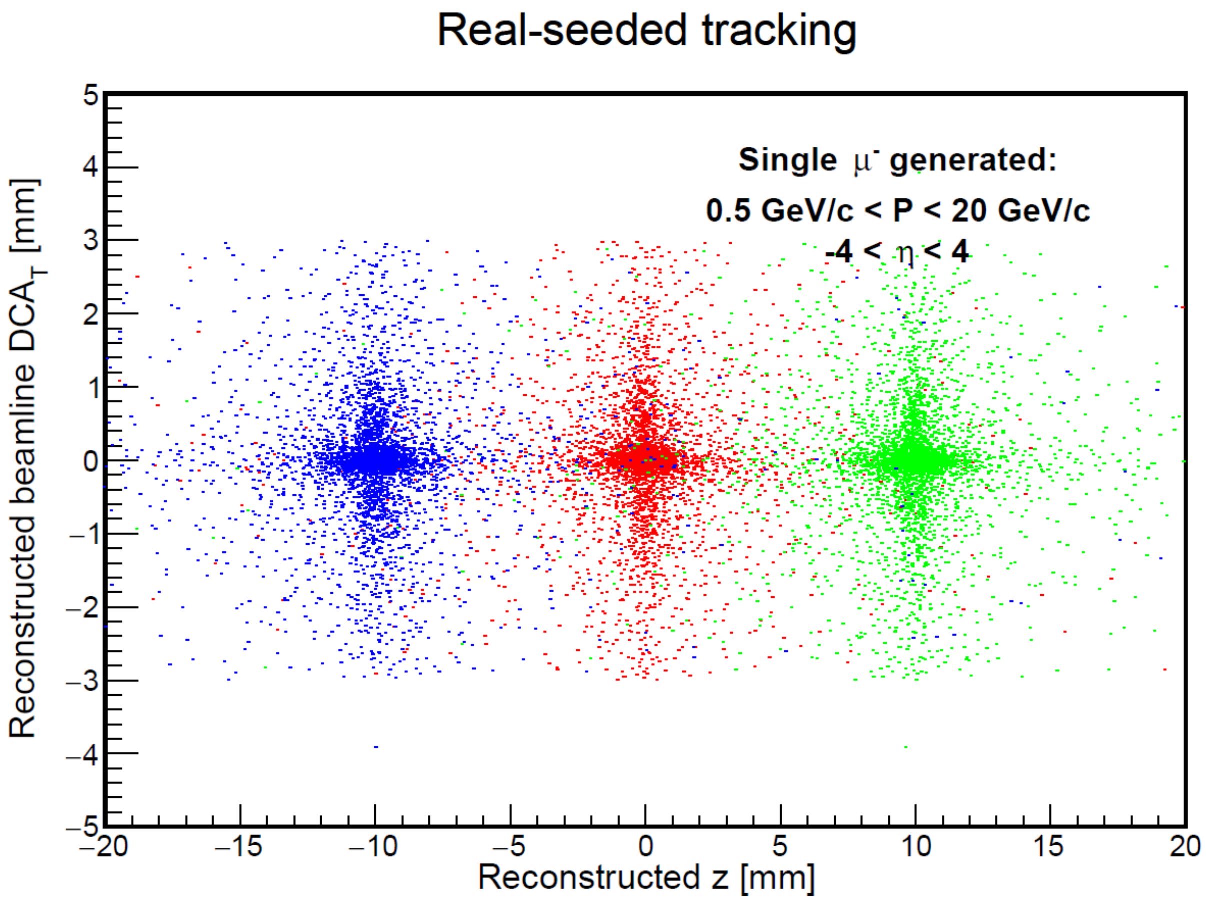
Motivation / Simulation procedure

- We want to systematically study the truth-seeded and real-seeded track reconstruction as a function of generation vertex.
- We do this with a single particle simulation:
  - Single negative muon
  - Uniform eta from [-4,4]; uniform momentum from [0.5,20] GeV/c; uniform phi
  - We first study the reconstruction when the particle is moved along the z axis: (0,0,0) mm; (0,0,+10)mm; (0,0,-10)mm.
  - We then study the effect of moving the particle off the beamline: e.g. (+10,0,0) mm

(vx,vy,vz) = (0,0,0) mm
(vx,vy,vz) = (0,0,+10) mm
(vx,vy,vz) = (0,0,-10) mm

Single-particle reconstruction



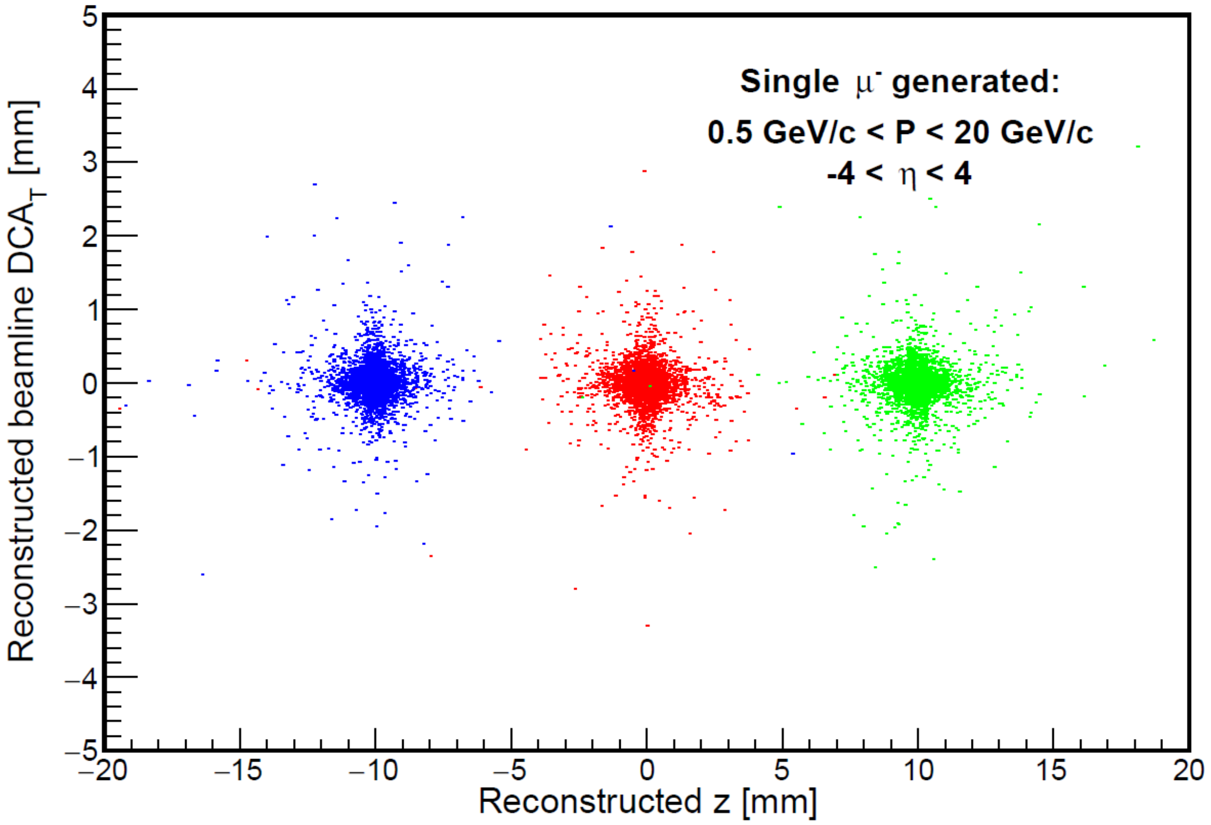


Reconstructed z: longitudinal impact parameter with respect to (0,0,0)
Reconstructed transverse DCA: transverse impact parameter with respect to (0,0,0)

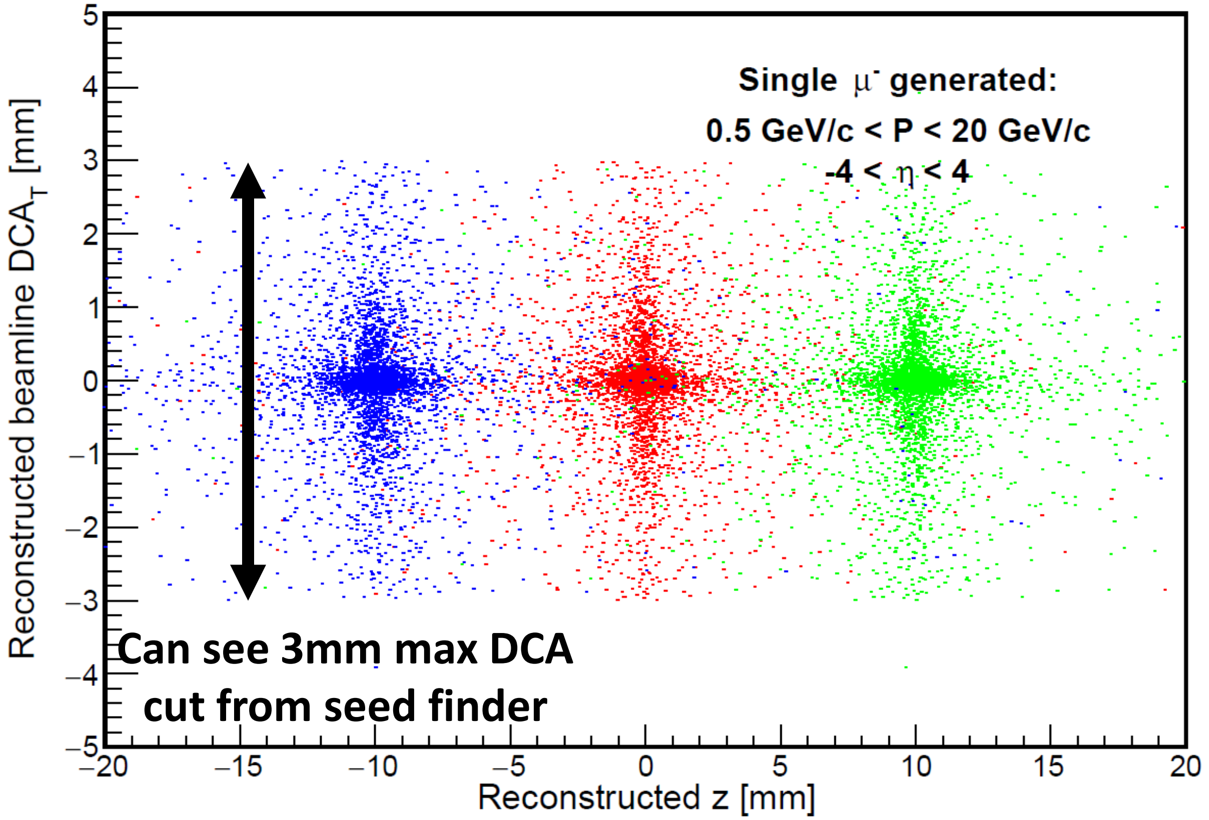
(vx,vy,vz) = (0,0,0) mm
(vx,vy,vz) = (0,0,+10) mm
(vx,vy,vz) = (0,0,-10) mm

Single-particle reconstruction

Truth-seeded tracking



Real-seeded tracking

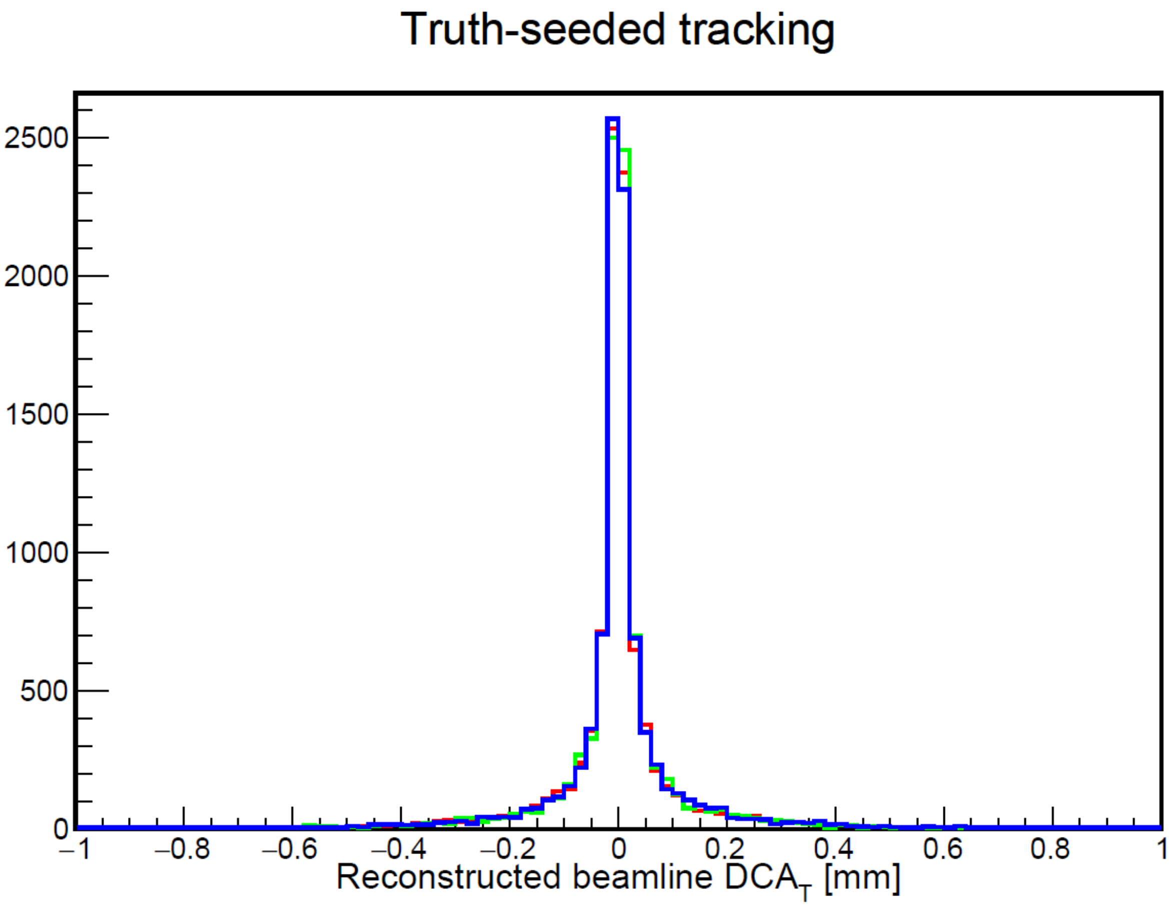


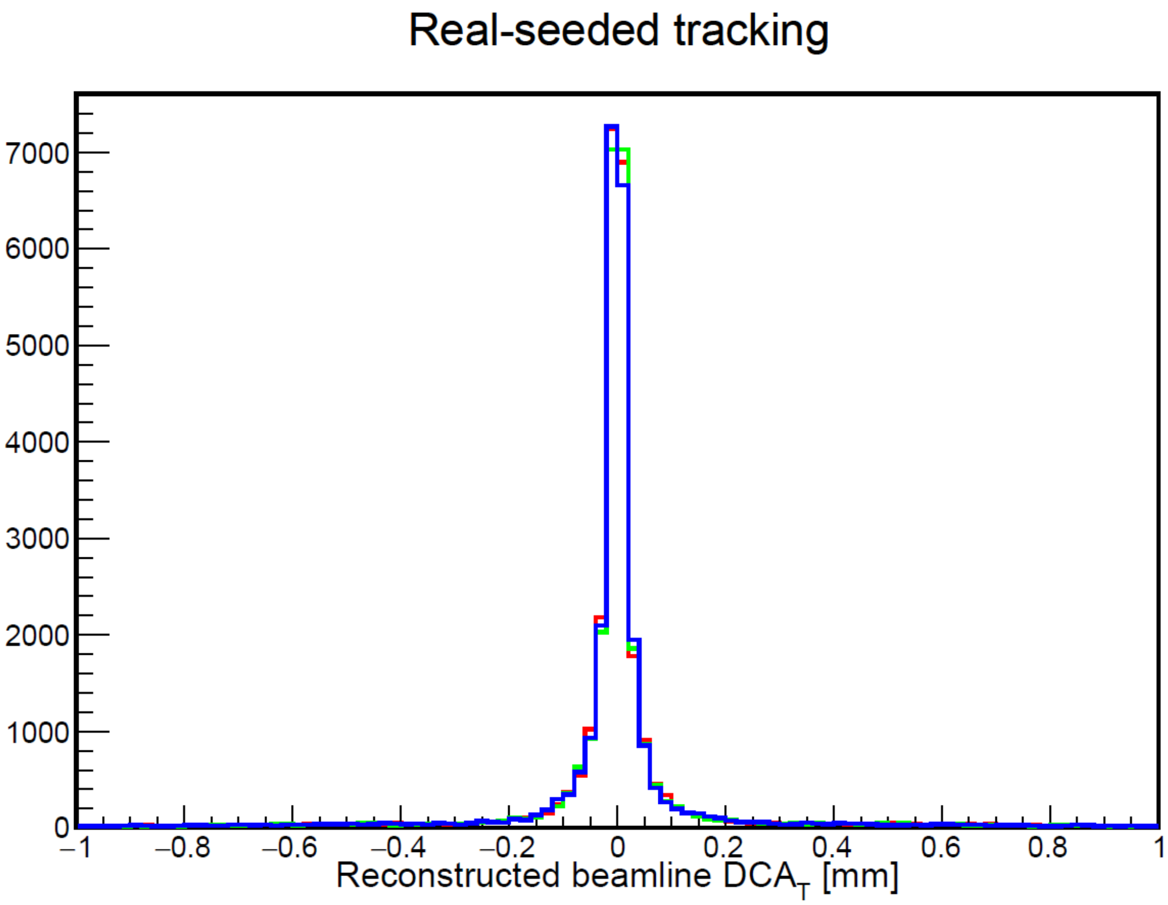
Can see 3mm max DCA cut from seed finder

Reconstructed z: longitudinal impact parameter with respect to (0,0,0)
Reconstructed transverse DCA: transverse impact parameter with respect to (0,0,0)

(vx,vy,vz) = (0,0,0) mm
(vx,vy,vz) = (0,0,+10) mm
(vx,vy,vz) = (0,0,-10) mm

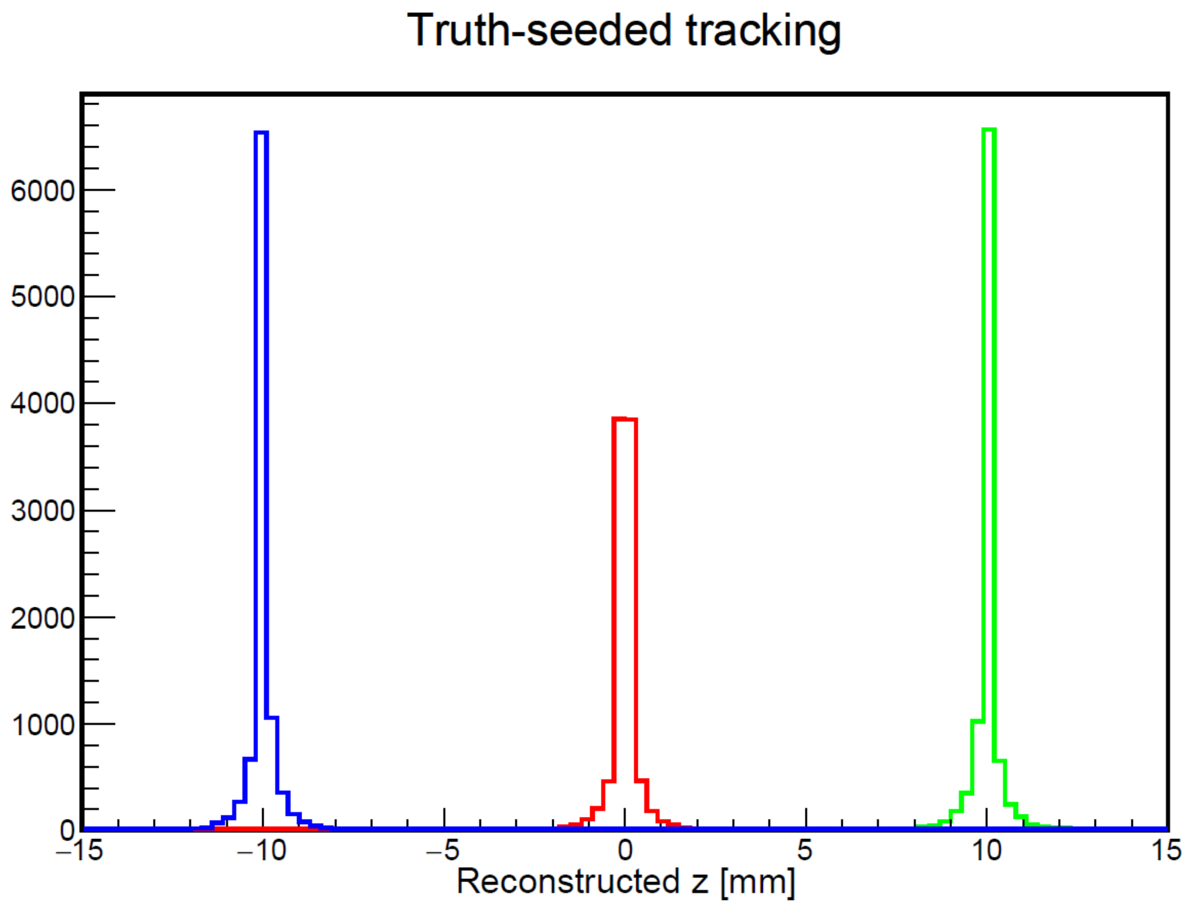
Single-particle reconstruction

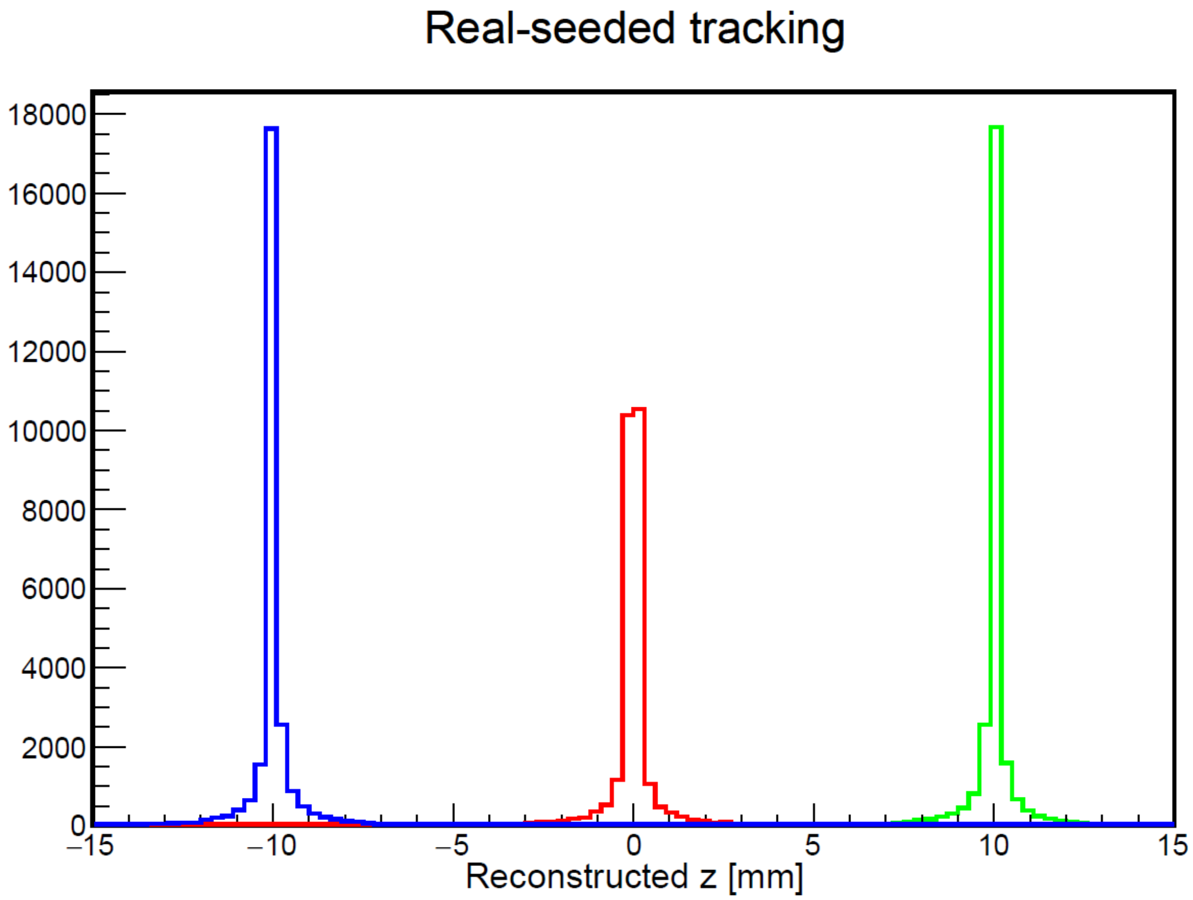




(vx,vy,vz) = (0,0,0) mm
(vx,vy,vz) = (0,0,+10) mm
(vx,vy,vz) = (0,0,-10) mm

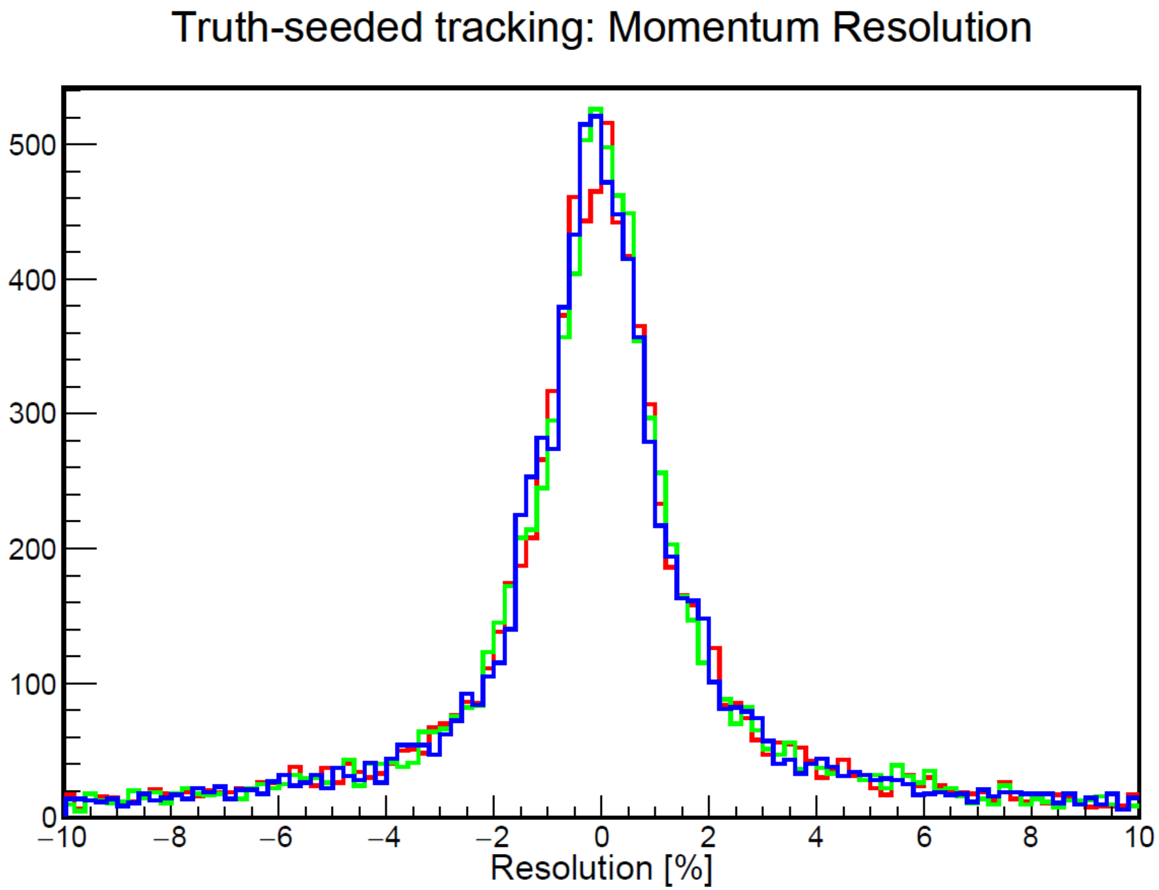
Single-particle reconstruction

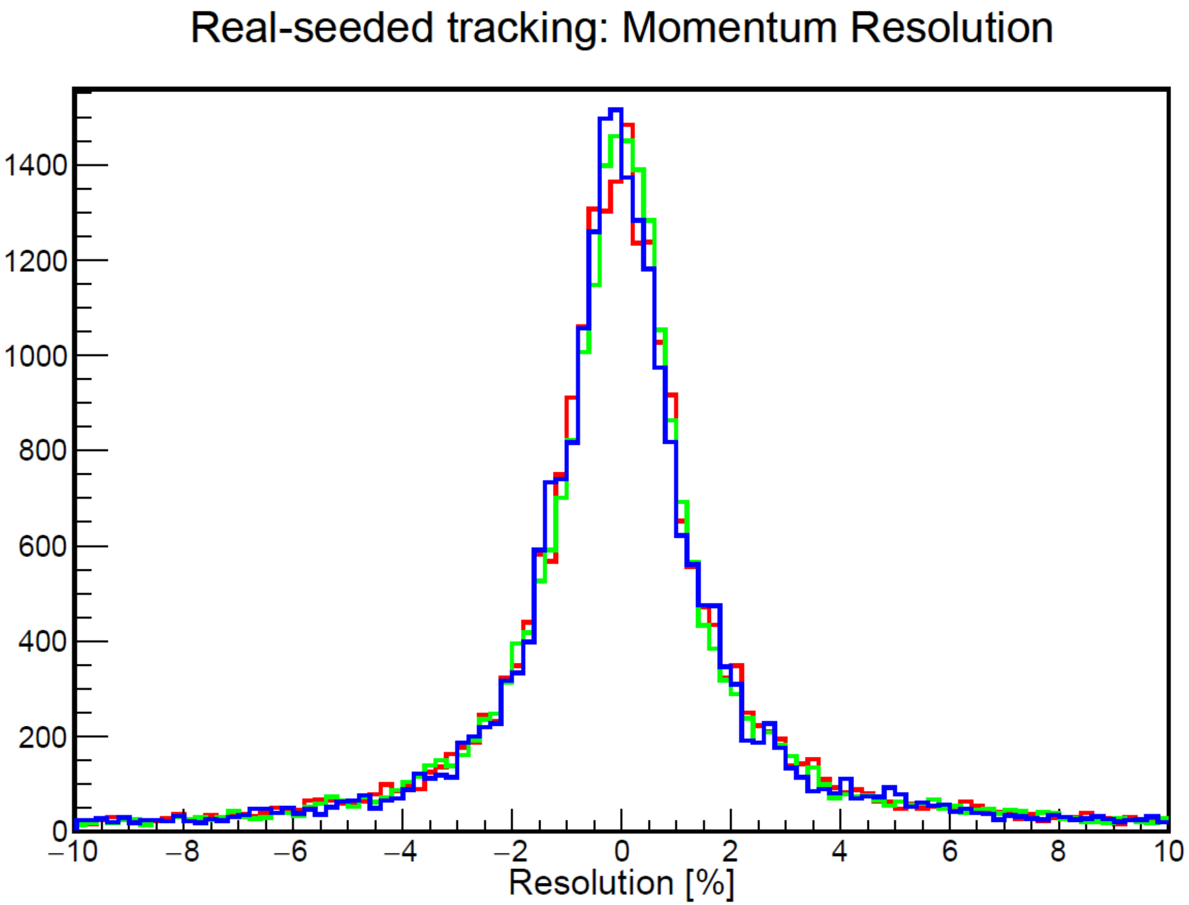




(vx,vy,vz) = (0,0,0) mm
(vx,vy,vz) = (0,0,+10) mm
(vx,vy,vz) = (0,0,-10) mm

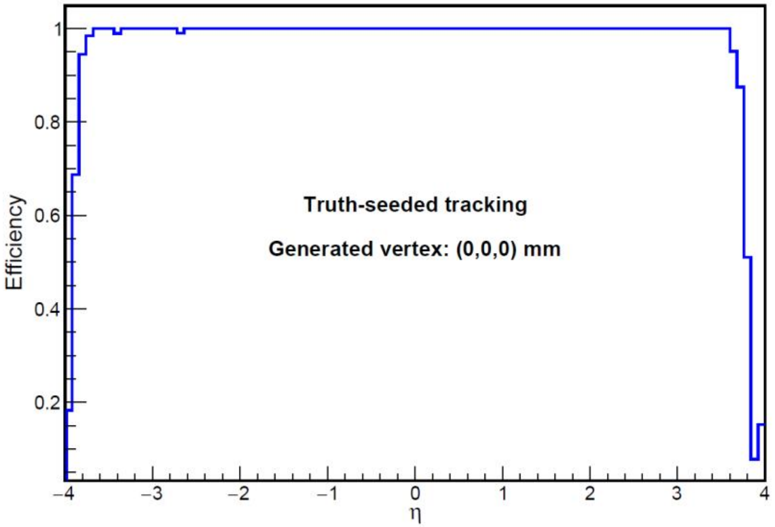
Single-particle reconstruction



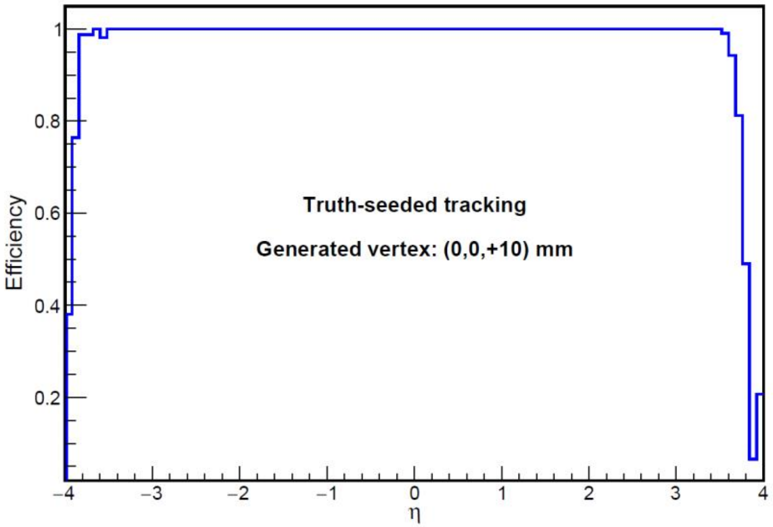


Single-particle reconstruction

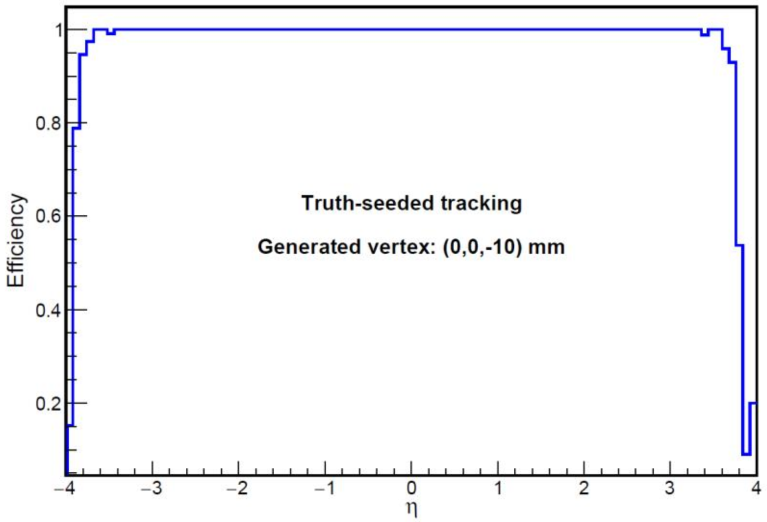
Tracker Efficiency vs. generated particle η



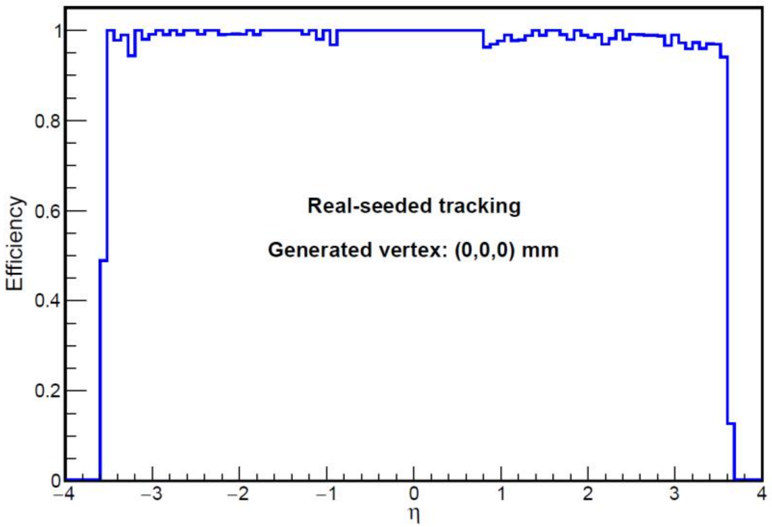
Tracker Efficiency vs. generated particle η



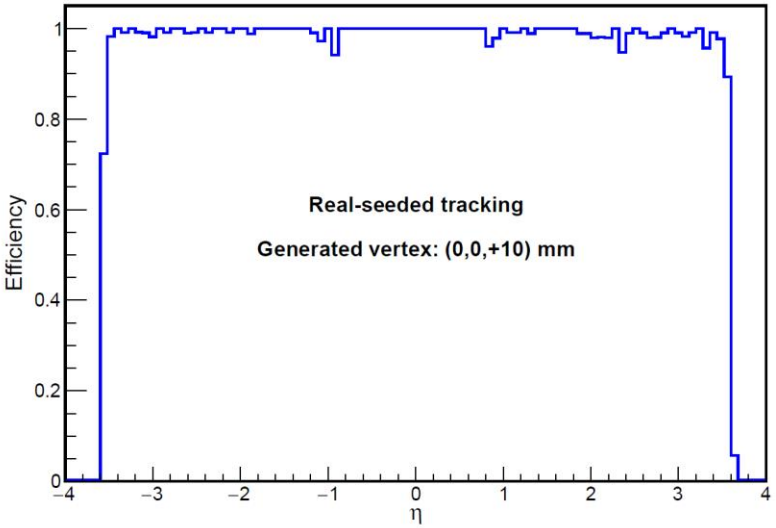
Tracker Efficiency vs. generated particle η



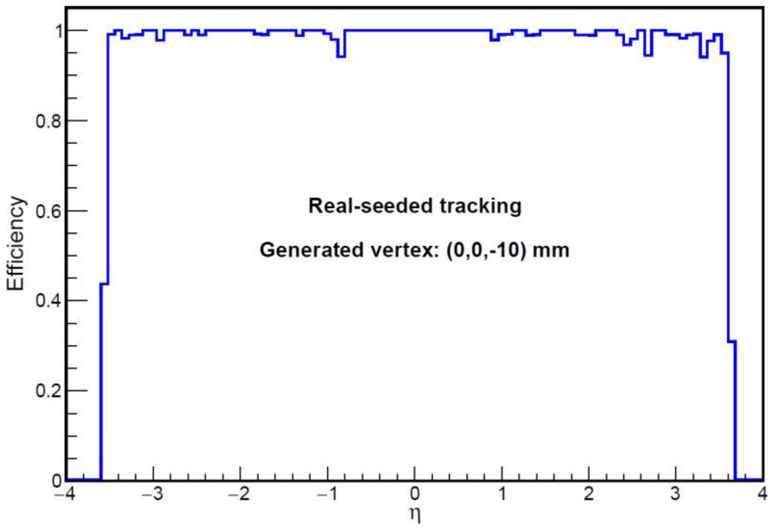
Tracker Efficiency vs. generated particle η



Tracker Efficiency vs. generated particle η

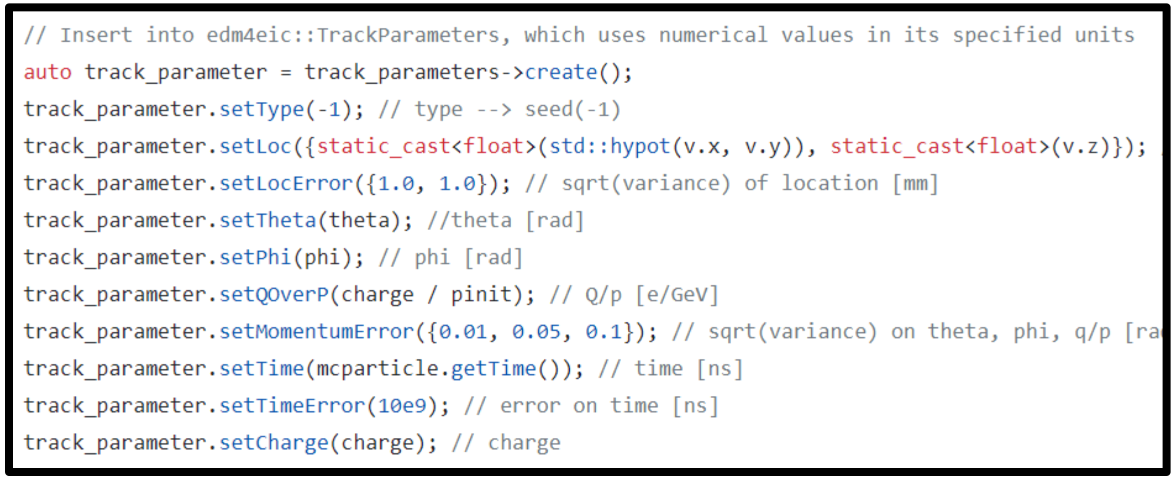


Tracker Efficiency vs. generated particle η

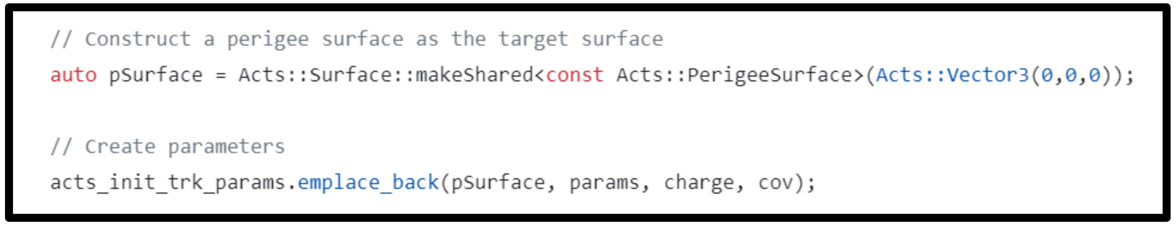


What about particles generated off the beamline?

- Before showing any simulation results, if we start with the truth seeding, there is a reason to think the current implementation is problematic.
- We set the initial track parameters based on the generated particle’s momentum vector, charge, and creation point. This information is then fed into the CKF in addition to a line surface (perigee surface) along the z axis through (0,0,0).



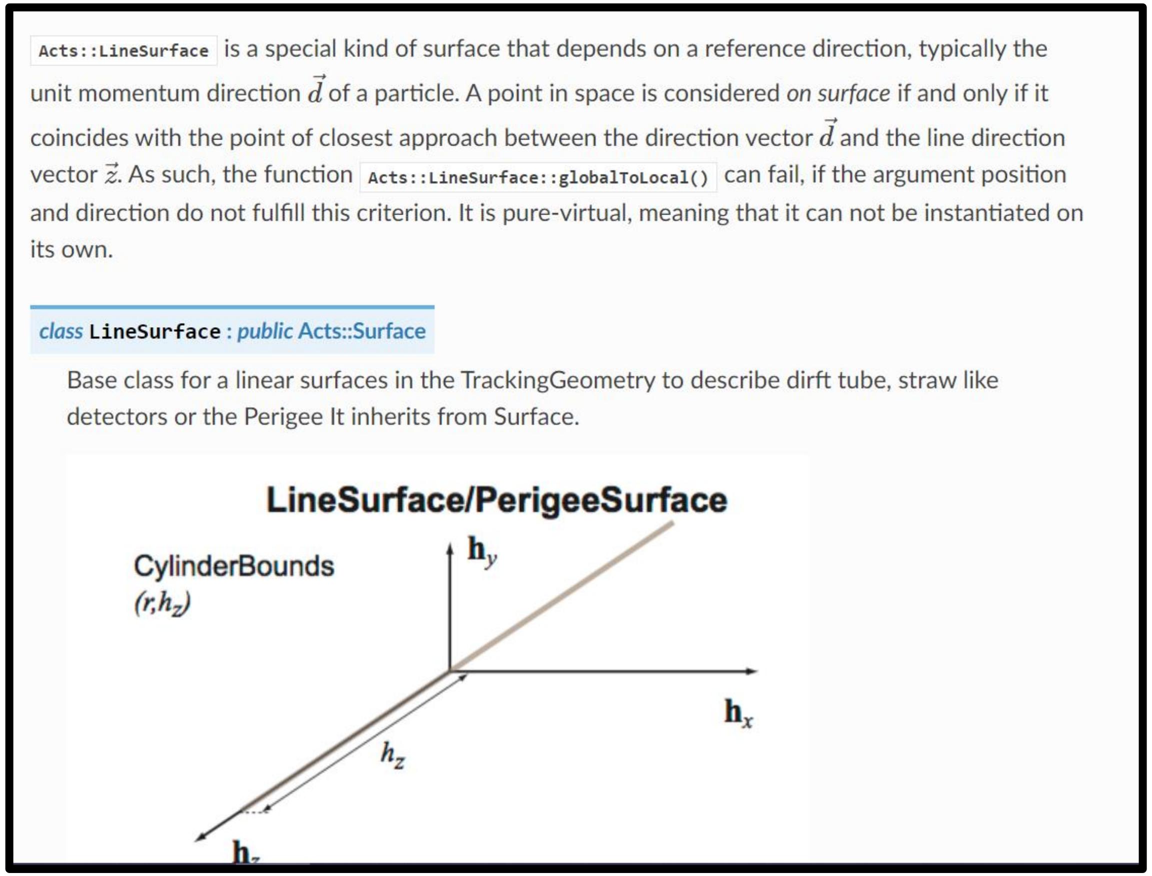
```
// Insert into edm4eic::TrackParameters, which uses numerical values in its specified units
auto track_parameter = track_parameters->create();
track_parameter.setType(-1); // type --> seed(-1)
track_parameter.setLoc({static_cast<float>(std::hypot(v.x, v.y)), static_cast<float>(v.z)});
track_parameter.setLocError({1.0, 1.0}); // sqrt(variance) of location [mm]
track_parameter.setTheta(theta); //theta [rad]
track_parameter.setPhi(phi); // phi [rad]
track_parameter.setQOverP(charge / pinit); // Q/p [e/GeV]
track_parameter.setMomentumError({0.01, 0.05, 0.1}); // sqrt(variance) on theta, phi, q/p [ra
track_parameter.setTime(mcparticle.getTime()); // time [ns]
track_parameter.setTimeError(10e9); // error on time [ns]
track_parameter.setCharge(charge); // charge
```



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

// Create parameters
acts_init_trk_params.emplace_back(pSurface, params, charge, cov);
```

What about particles generated off the beamline?

- Before showing any simulation results, if we start with the truth seeding, there is a reason to think the current implementation is problematic.
- We set the initial track parameters based on the generated particle's momentum vector, charge, and creation point. This information is then fed into the CKF in addition to a line surface (perigee surface) along the z axis through (0,0,0).
- The created particle's momentum vector may not be tangential to the cylinder surrounding the line surface at its creation point.



`Acts::LineSurface` is a special kind of surface that depends on a reference direction, typically the unit momentum direction $\vec{d}$ of a particle. A point in space is considered *on surface* if and only if it coincides with the point of closest approach between the direction vector $\vec{d}$ and the line direction vector $\vec{z}$. As such, the function `Acts::LineSurface::globalToLocal()` can fail, if the argument position and direction do not fulfill this criterion. It is pure-virtual, meaning that it can not be instantiated on its own.

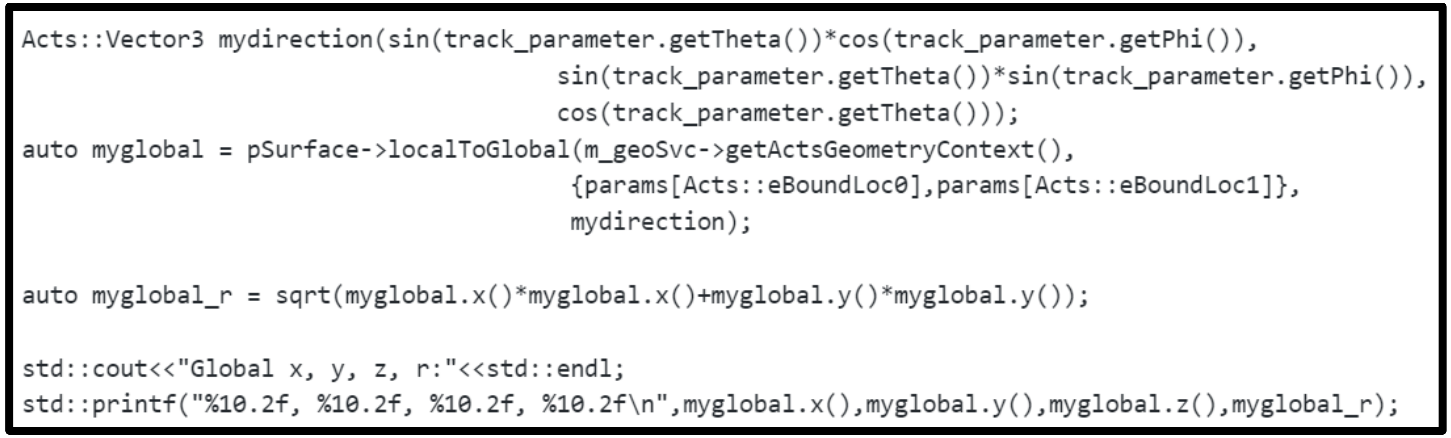
class `LineSurface` : *public* Acts::Surface

Base class for a linear surfaces in the TrackingGeometry to describe dirft tube, straw like detectors or the Perigee It inherits from Surface.

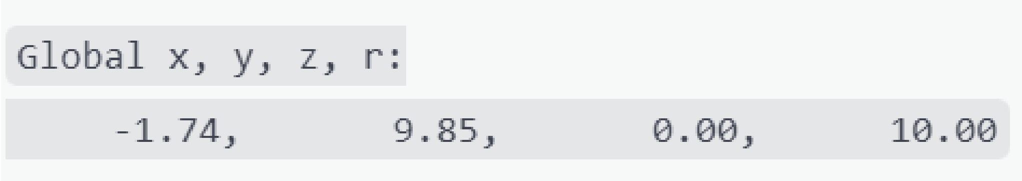
https://acts.readthedocs.io/en/latest/core/geometry/surfaces.html#line-surface

What about particles generated off the beamline?

- To test, generate a single particle from **(x,y,z) = (10,0,0) mm**, with a momentum direction of **(px,py,pz) = {cos(10 degrees), sin(10 degrees), 0}.**
- In the EICRecon CKF class, use the **LocaltoGlobal** function on the initial track parameters. My guess is that the CKF uses this same function internally when doing the particle propagation.
- We see that the CFK will think that the particle’s parameters were given at a different position than the creation point – i.e. the tangential point around the line surface which is at the same radius. But this point is not usually a point on the particle’s trajectory.



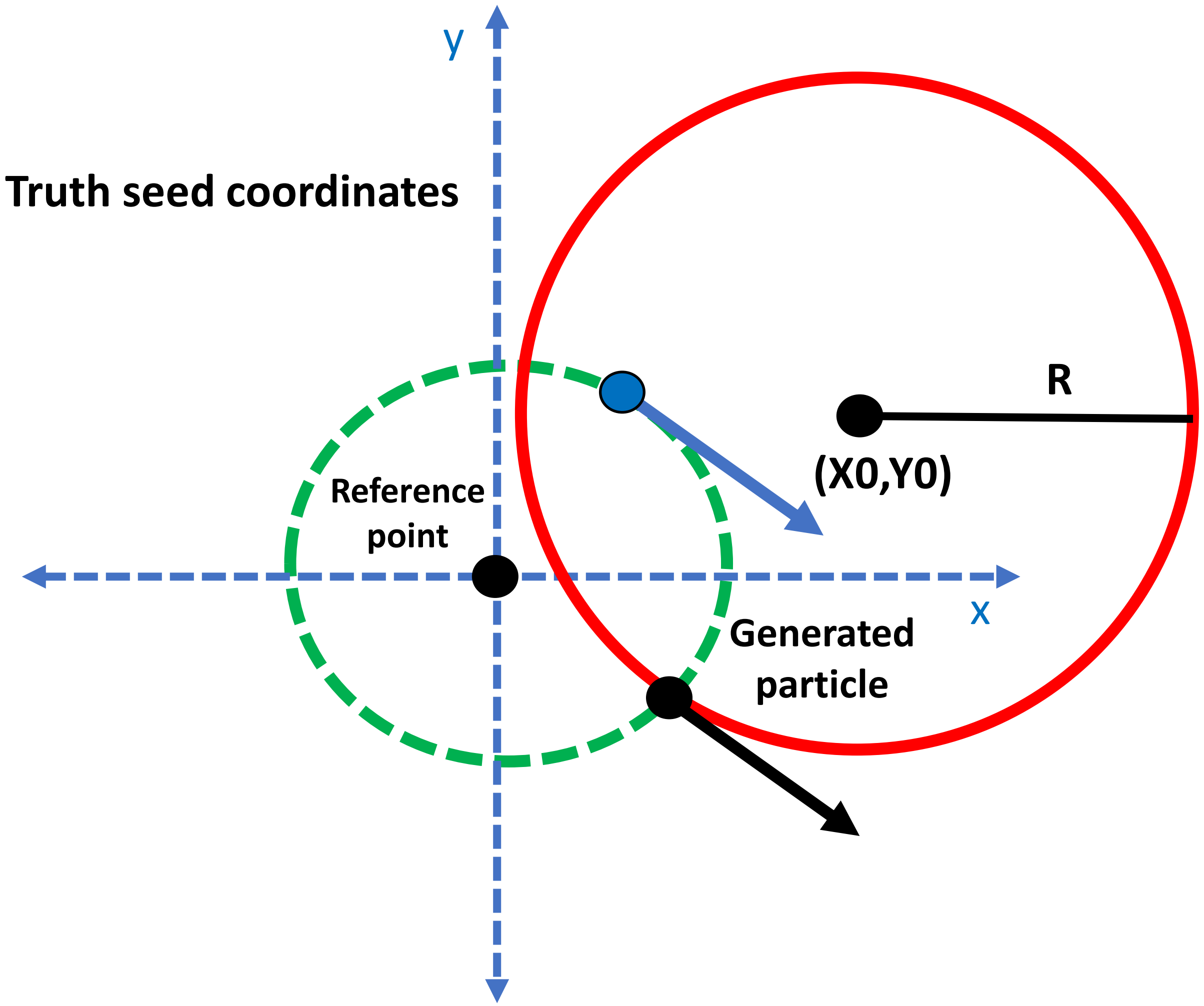
```
Acts::Vector3 mydirection(sin(track_parameter.getTheta())*cos(track_parameter.getPhi()),
                                      sin(track_parameter.getTheta())*sin(track_parameter.getPhi()),
                                      cos(track_parameter.getTheta()));
auto myglobal = pSurface->localToGlobal(m_geoSvc->getActsGeometryContext(),
                                       {params[Acts::eBoundLoc0],params[Acts::eBoundLoc1]},
                                       mydirection);

auto myglobal_r = sqrt(myglobal.x()*myglobal.x()+myglobal.y()*myglobal.y());

std::cout<<"Global x, y, z, r:"<<std::endl;
std::printf("%10.2f, %10.2f, %10.2f, %10.2f\n",myglobal.x(),myglobal.y(),myglobal.z(),myglobal_r);
```



```
Global x, y, z, r:
     -1.74,       9.85,       0.00,      10.00
```

What about particles generated off the beamline?

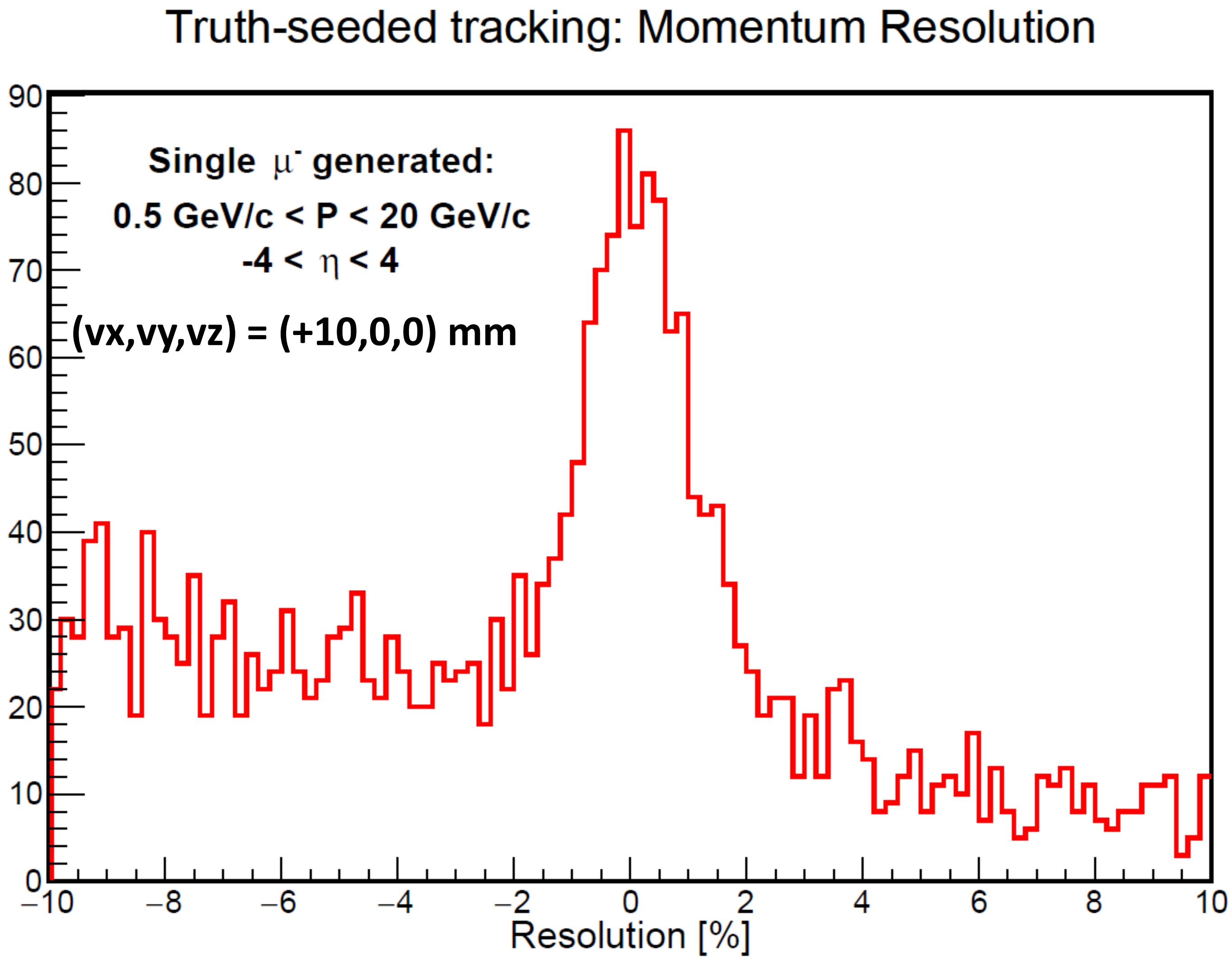
Truth seed coordinates



Black arrow: Generated particle at its creation point

Blue arrow: Where the CKF will think the particle comes from in the current truth seeding implementation.

What about particles generated off the beamline?

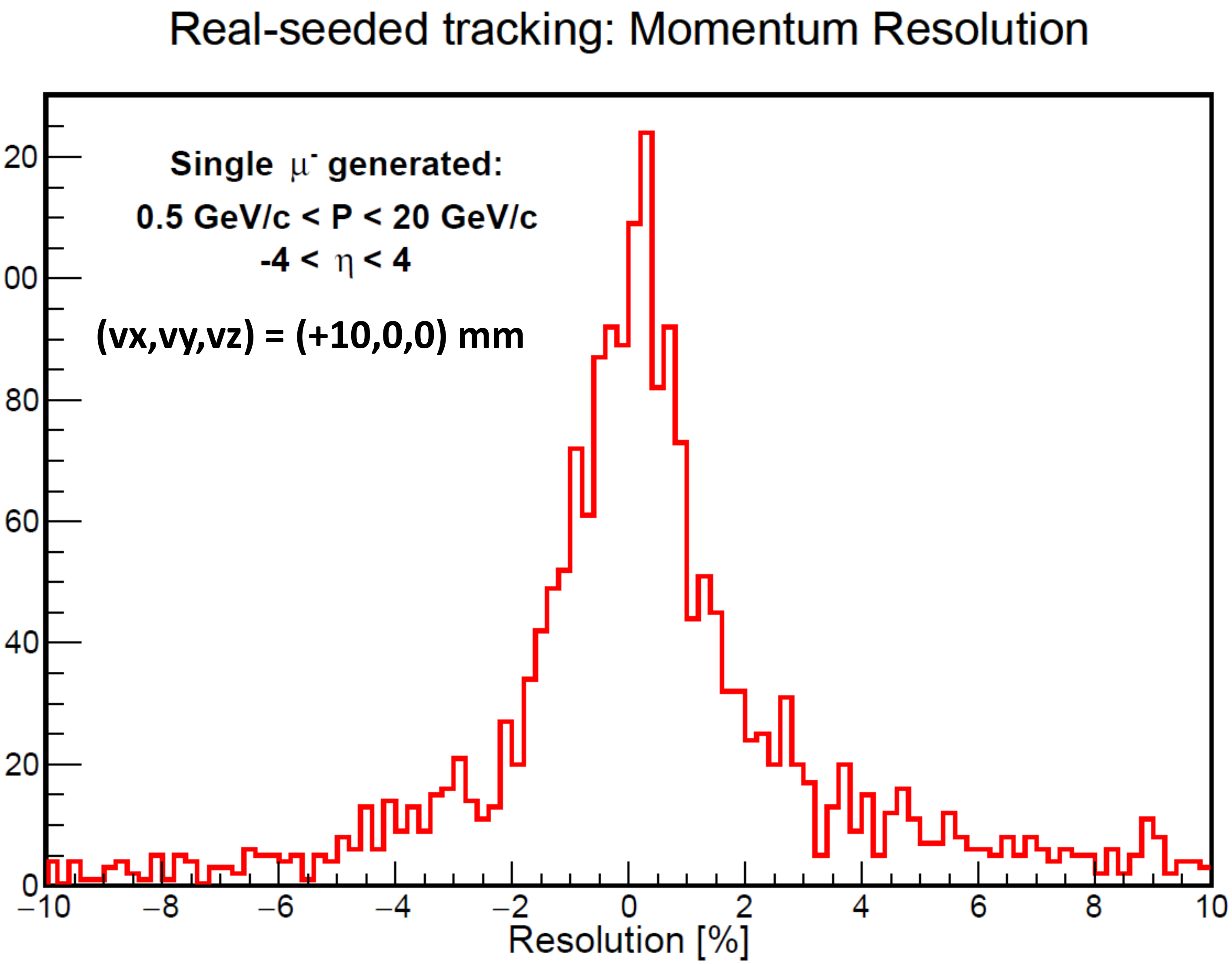


We see poor truth-seeded momentum reconstruction when a particle is generated 10mm off from the beamline.

Possible fixes for truth seeding

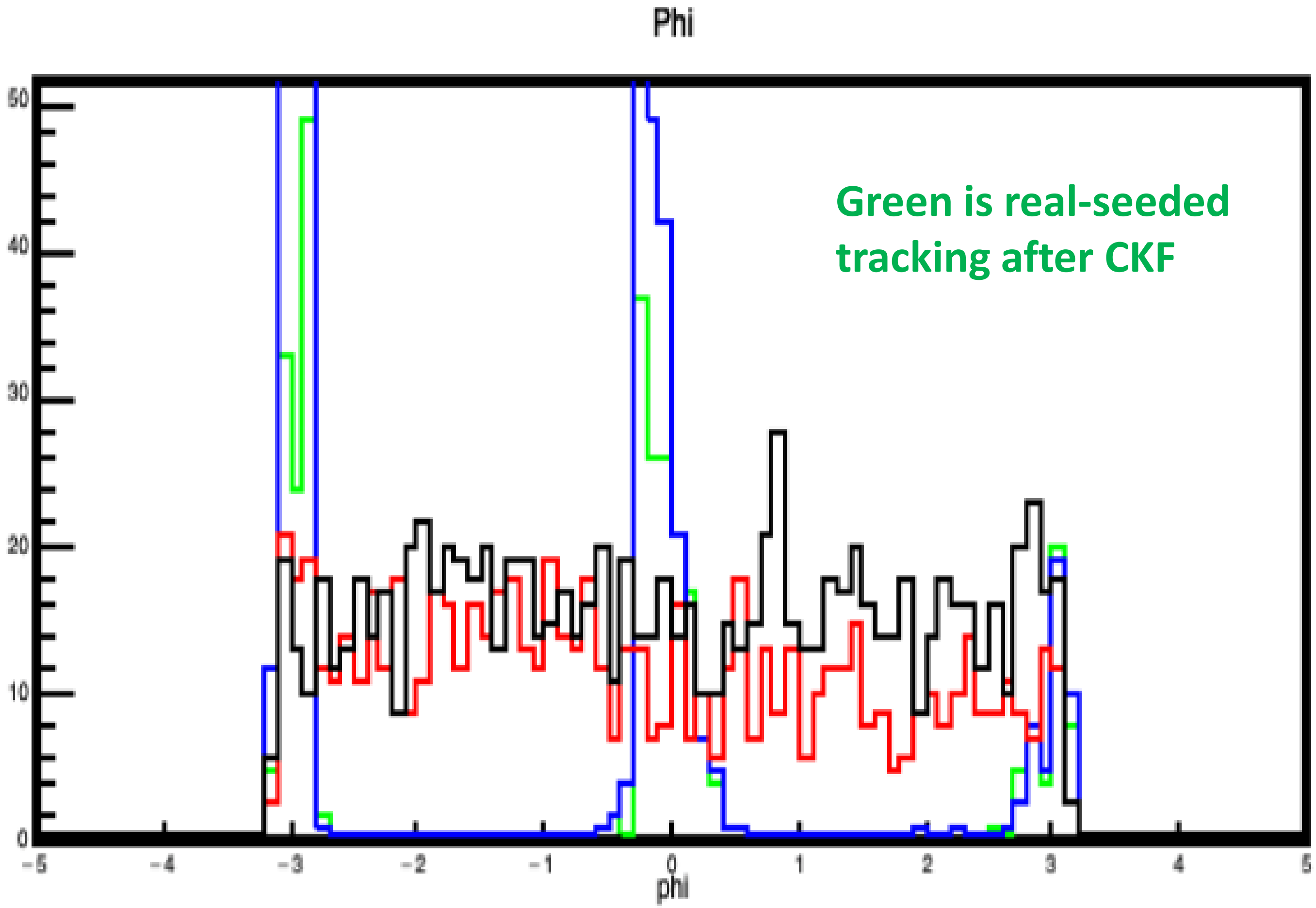
1. For a given generated particle, shift the line surface so that the reference point is the creation point of the particle. (The line surface would still be parallel to the z axis.) This will require some modification to the data model.
2. Using the particle's truth information, track the particle back to the DCA point with respect to the current line surface. This is similar to the approach used for real seeding.

What about real seeding?



We see better real-seeded momentum reconstruction when a particle is generated 10mm off from the beamline.

What about real seeding?



Green is real-seeded tracking after CKF

We see better real-seeded momentum reconstruction when a particle is generated 10mm off from the beamline.

But the efficiency is very poor. Need to first look at the effect of the seed finder max DCA parameter.

Conclusions

- Our current implementation for truth- and real-seeded tracking works well for particles generated on the z axis.
- For truth-seeding, the current implementation is problematic for off-beamline particles. I proposed two potential fixes. Any suggestions?
- For real-seeding, some more studies are needed before drawing conclusions.