Update on low-momentum seeding inefficiencies

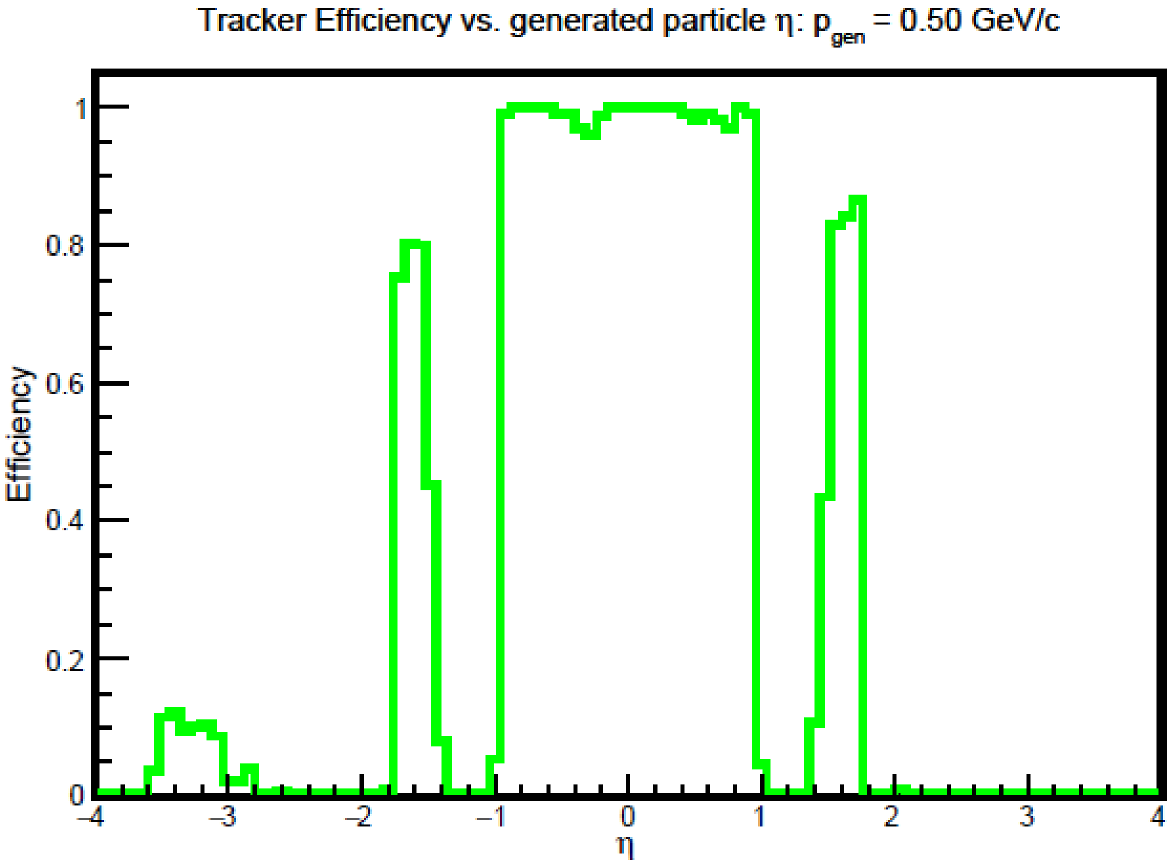
Barak Schmookler

Work by Jeetendra Gupta

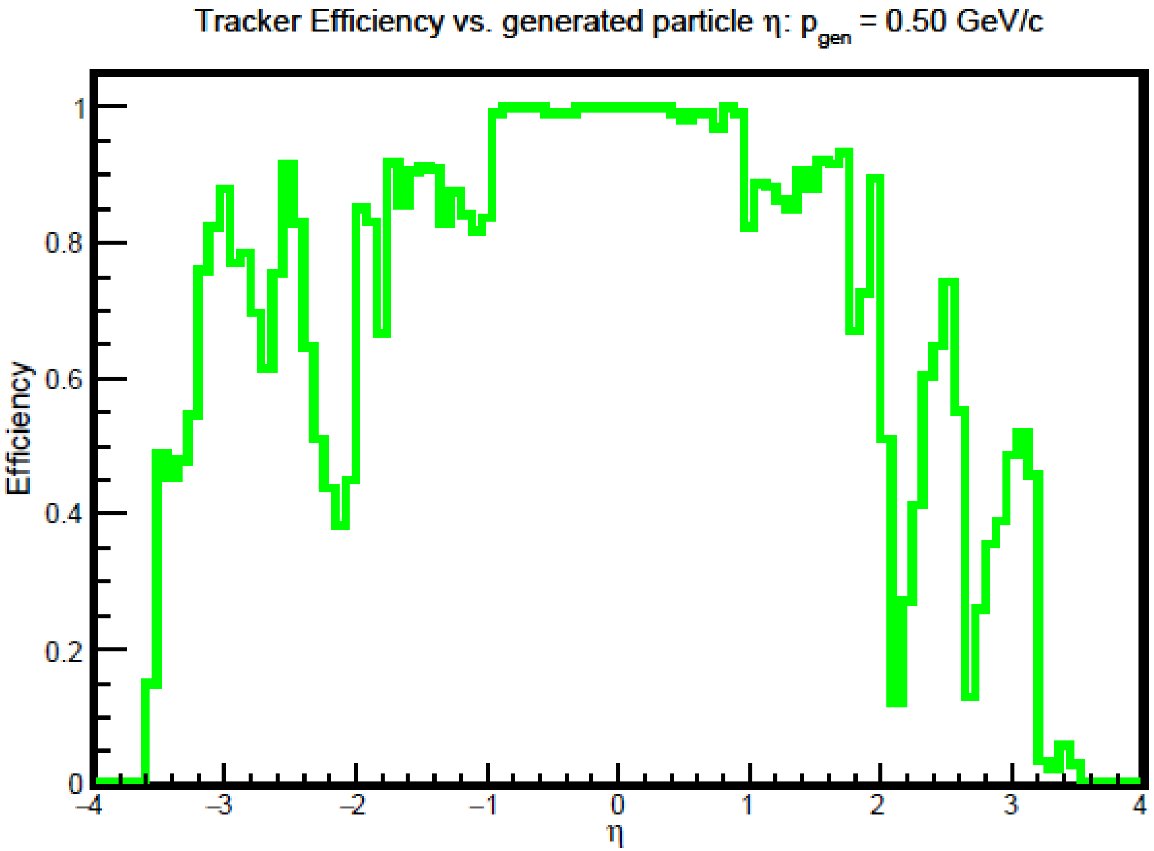
10k single negative muon events
Eta = [-4,4], Phi = [0,2Pi]
P = 500 MeV/c
(vx,vy,vz) = (0,0,0)

Previous studies

Default *deltaPhiMax* parameter = 0.085



Modified *deltaPhiMax* parameter = 0.16

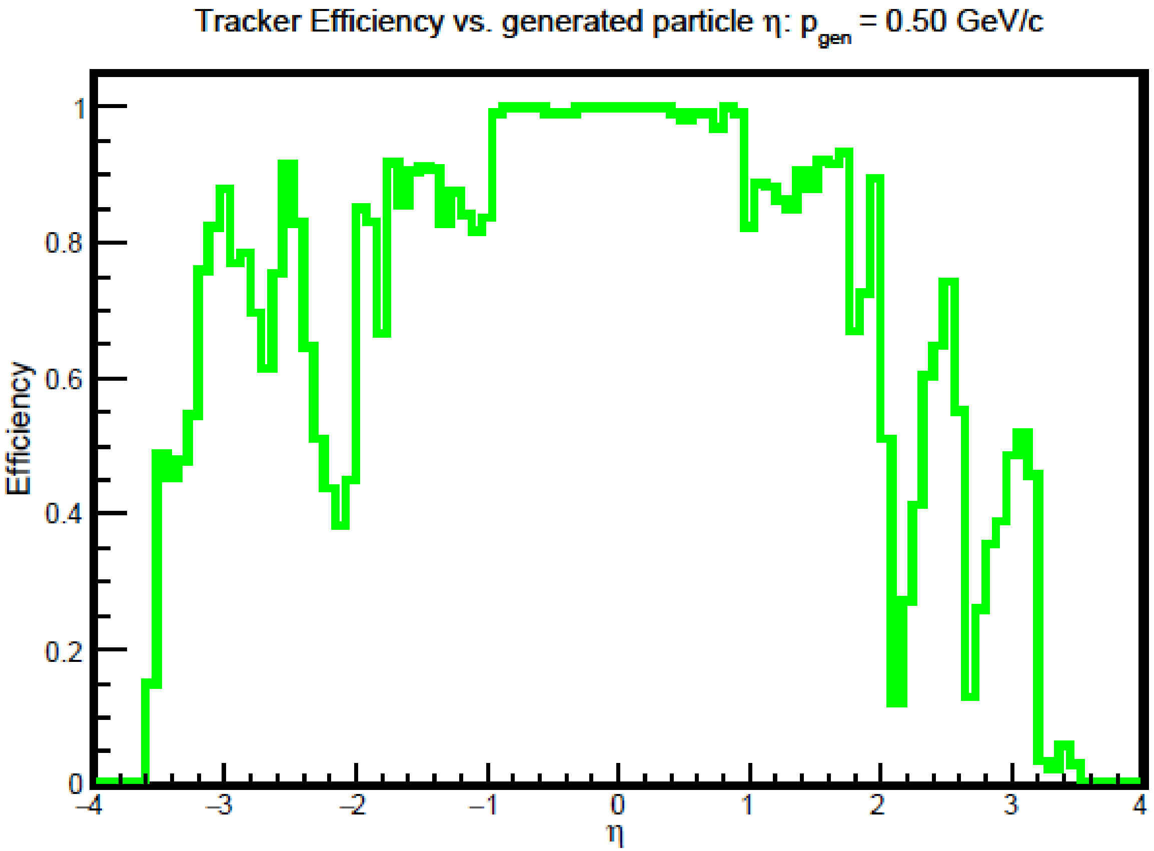


Work by Jeetendra Gupta

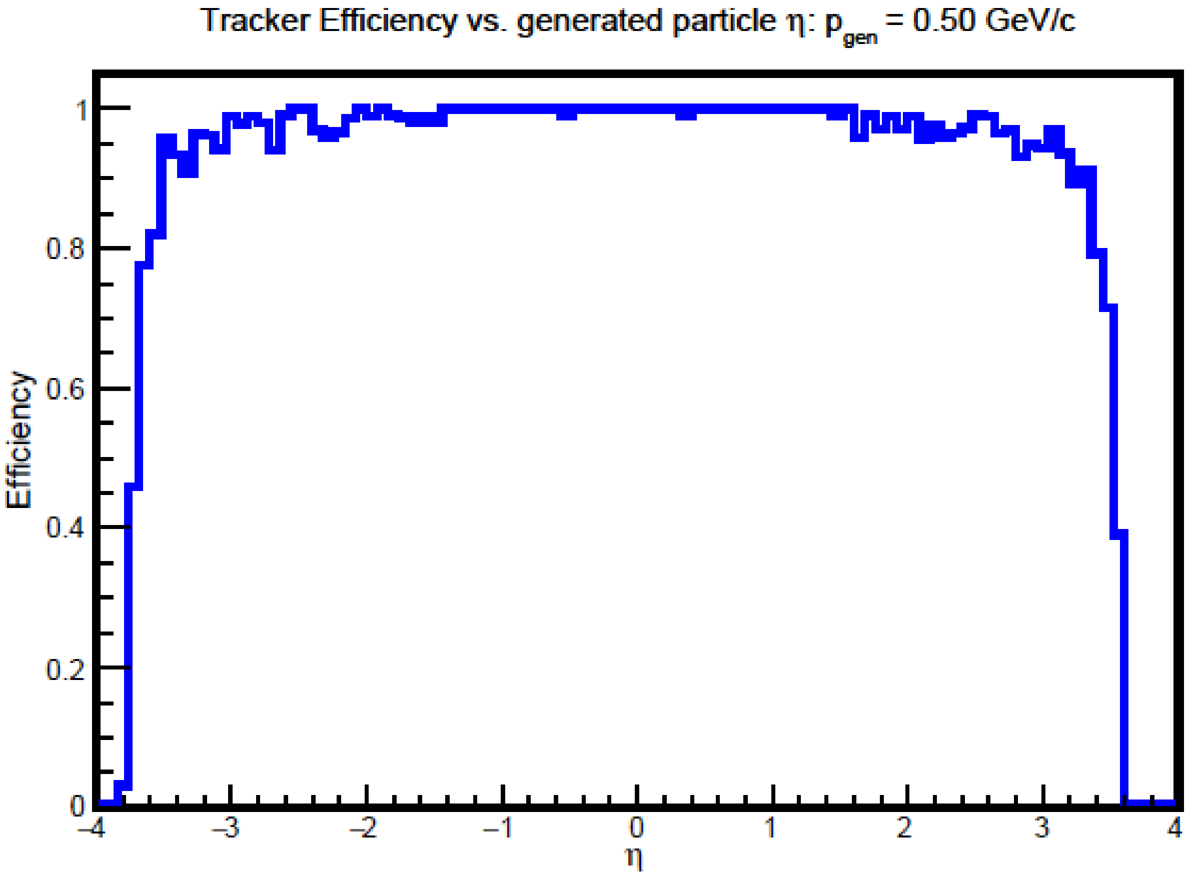
10k single negative muon events
Eta = [-4,4], Phi = [0,2Pi]
P = 500 MeV/c
(vx,vy,vz) = (0,0,0)

Previous studies

Modified *deltaPhiMax* parameter = 0.16



Truth-seeded track reconstruction with at least 3 measurement points used in track fit

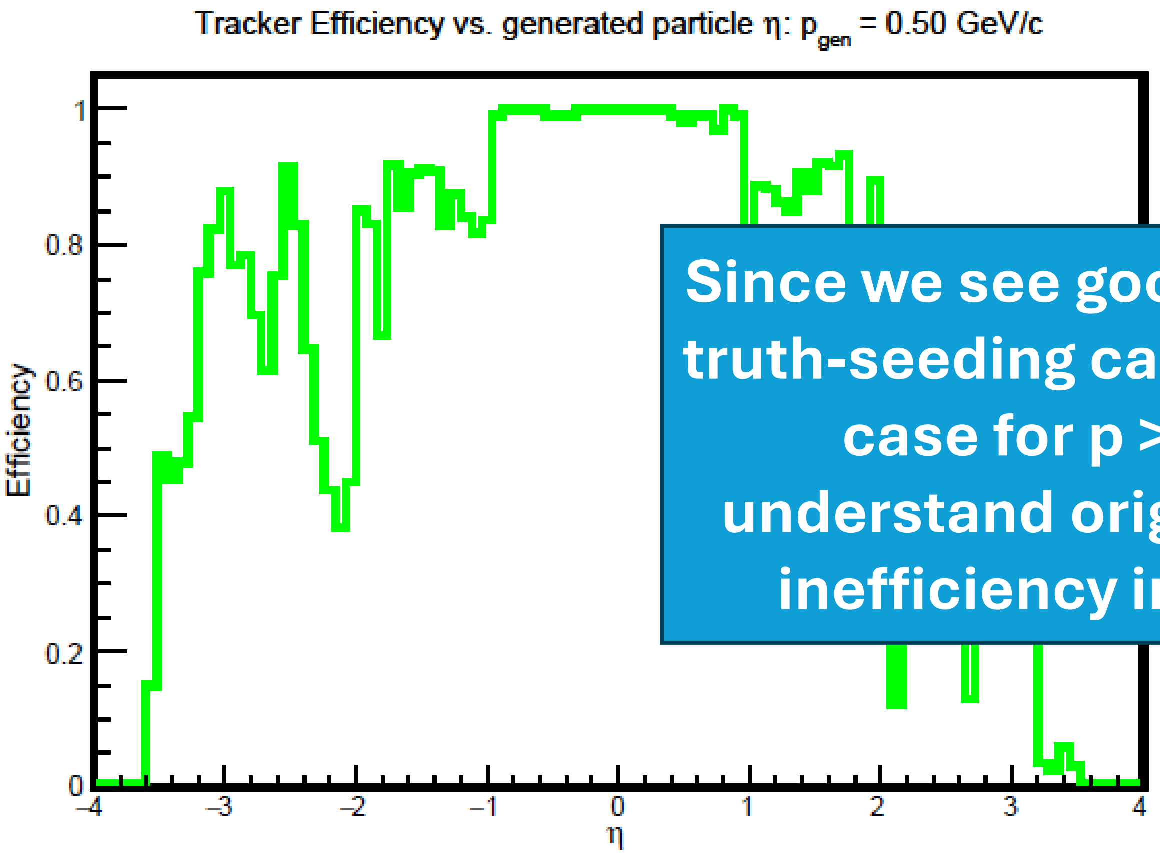


Work by Jeetendra Gupta

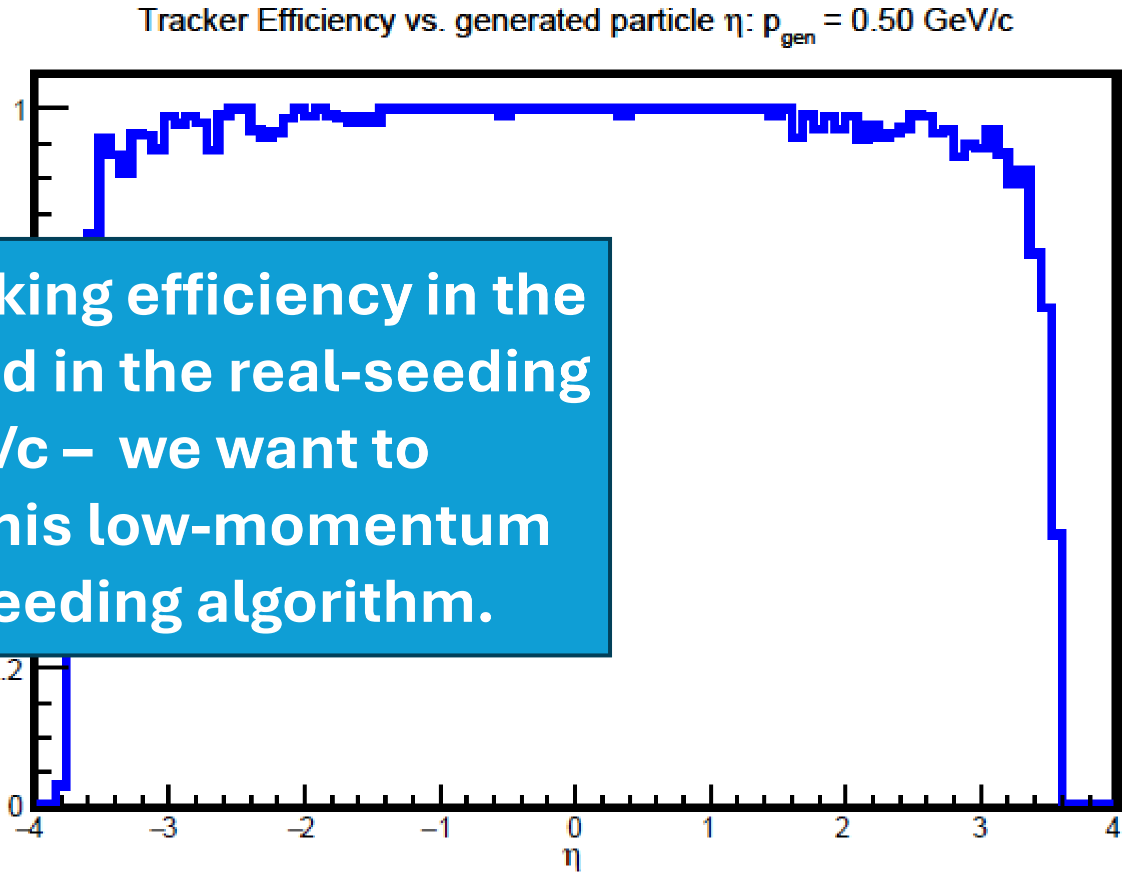
10k single negative muon events
Eta = [-4,4], Phi = [0,2Pi]
P = 500 MeV/c
(vx,vy,vz) = (0,0,0)

Previous studies

Modified *deltaPhiMax* parameter = 0.16



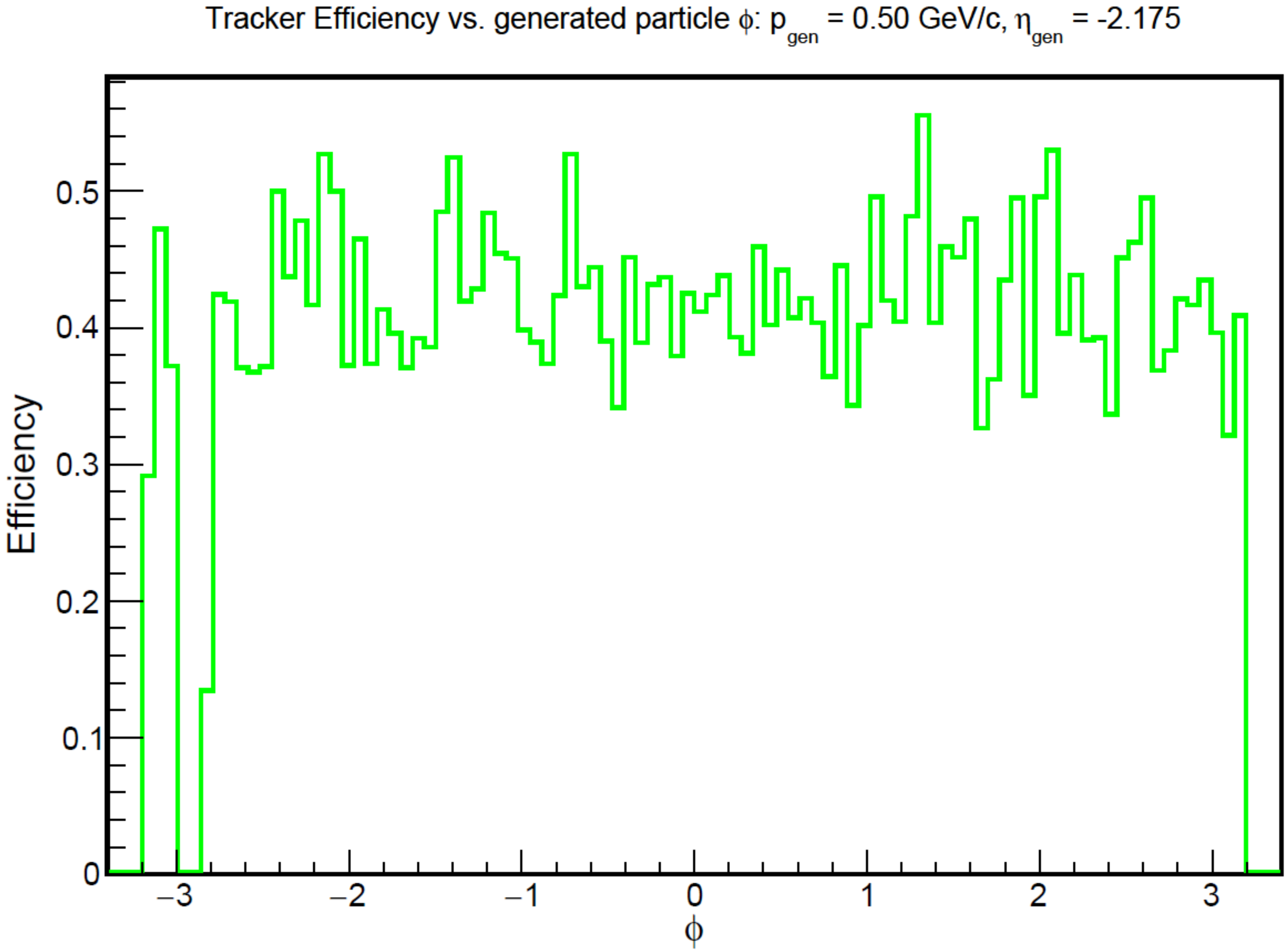
Truth-seeded track reconstruction with at least 3 measurement points used in track fit



Since we see good tracking efficiency in the truth-seeding case – and in the real-seeding case for p > 1 GeV/c – we want to understand origin of this low-momentum inefficiency in our seeding algorithm.

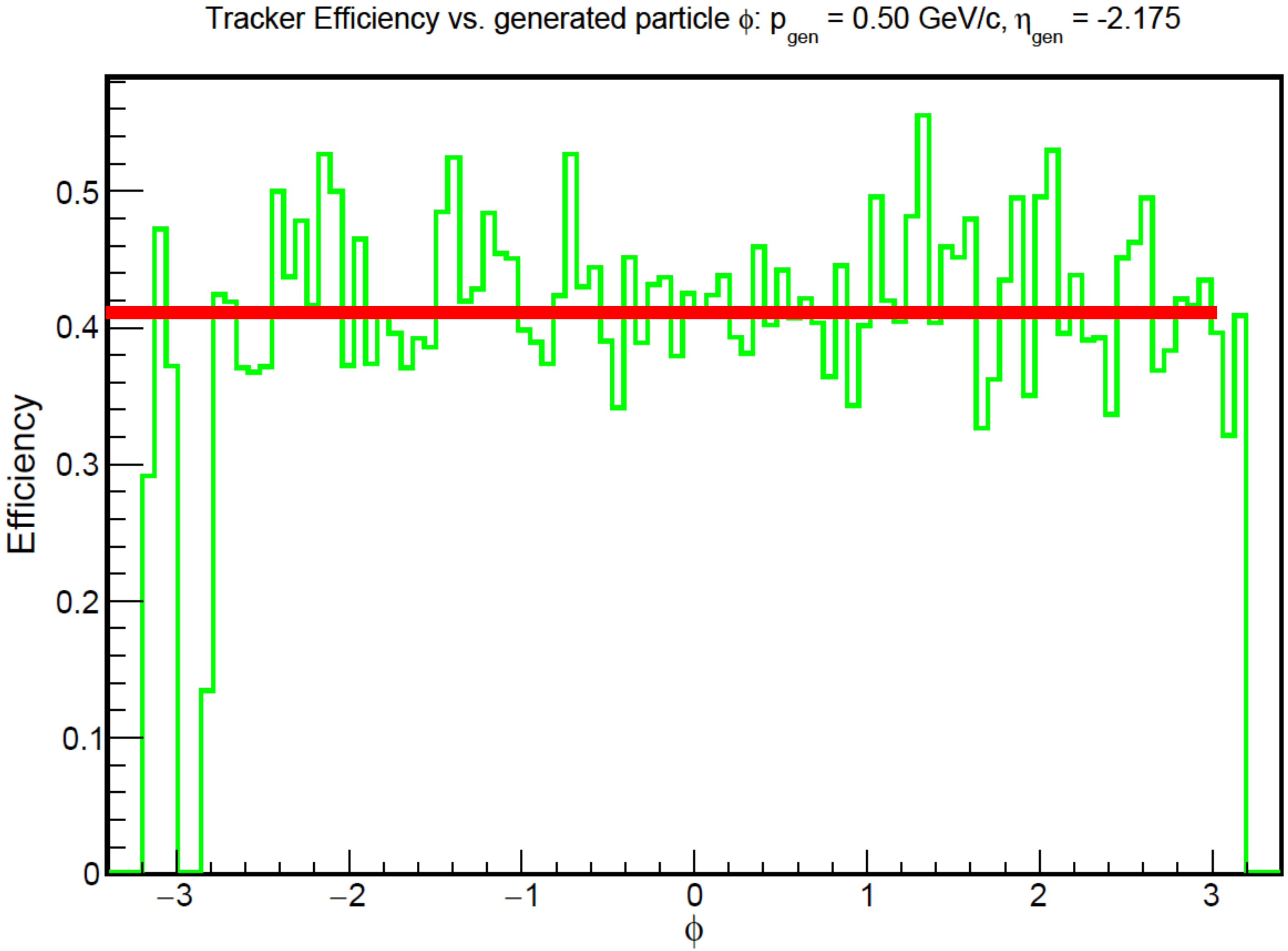
Throw particles at fixed momentum and eta

10k single negative muon events
Eta = -2.175, Phi = [0,2Pi]
P = 500 MeV/c
(vx,vy,vz) = (0,0,0)



Throw particles at fixed momentum and eta

10k single negative muon events
Eta = -2.175, Phi = [0,2Pi]
P = 500 MeV/c
(vx,vy,vz) = (0,0,0)

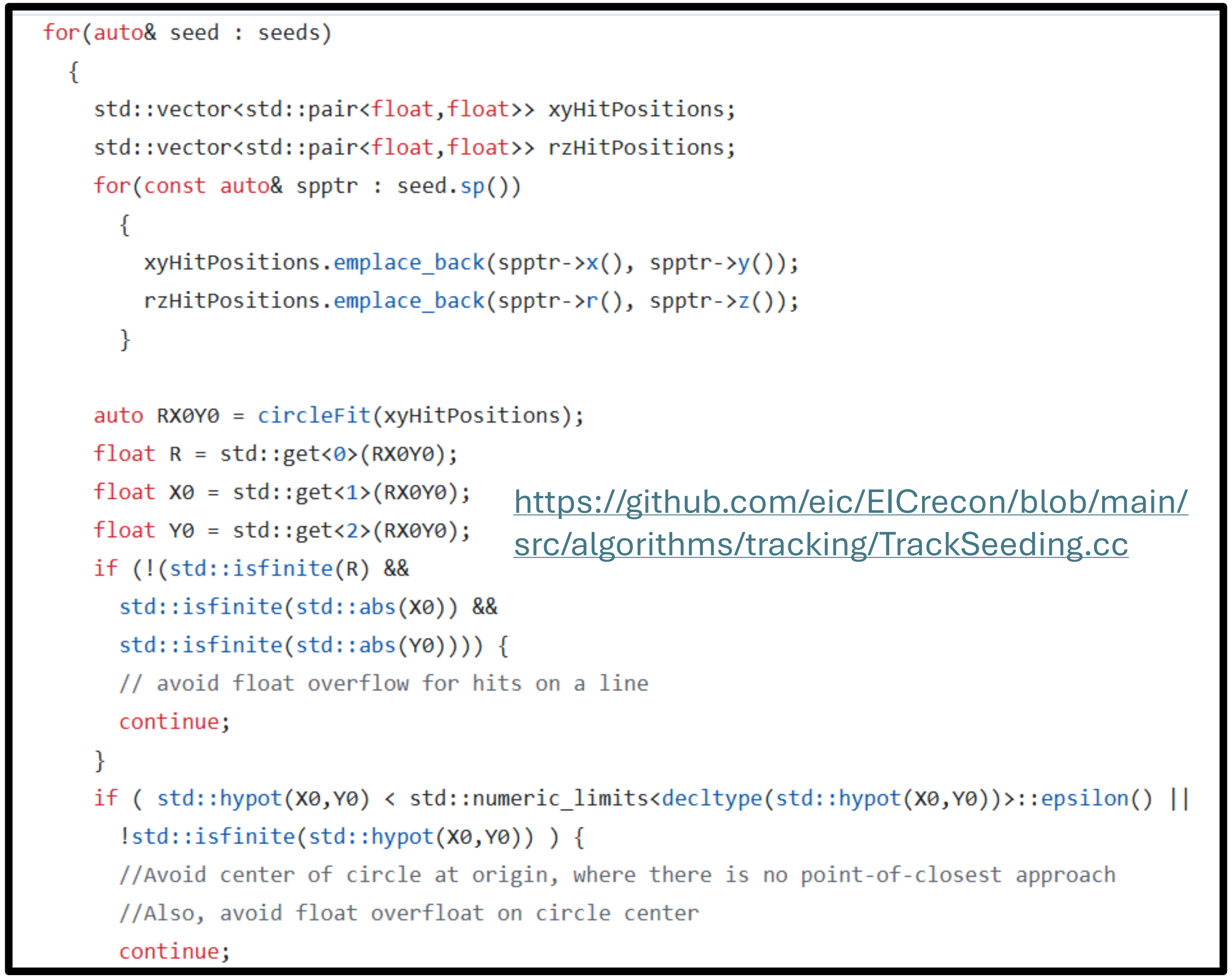


We see inefficiency over whole phi range.

This means that almost identical events can get different seeding results

First check: do we get seeds but fail to fit them

Do we enter the seed fitting loop and then trigger one of the ‘continue’ conditions?



```
for(auto& seed : seeds)
  {
    std::vector<std::pair<float,float>> xyHitPositions;
    std::vector<std::pair<float,float>> rzHitPositions;
    for(const auto& spptr : seed.sp())
      {
        xyHitPositions.emplace_back(spptr->x(), spptr->y());
        rzHitPositions.emplace_back(spptr->r(), spptr->z());
      }

    auto RX0Y0 = circleFit(xyHitPositions);
    float R = std::get<0>(RX0Y0);
    float X0 = std::get<1>(RX0Y0);
    float Y0 = std::get<2>(RX0Y0);
    if (!(std::isfinite(R) &&
      std::isfinite(std::abs(X0)) &&
      std::isfinite(std::abs(Y0)))) {
      // avoid float overflow for hits on a line
      continue;
    }
    if ( std::hypot(X0,Y0) < std::numeric_limits<decltype(std::hypot(X0,Y0))>::epsilon() ||
      !std::isfinite(std::hypot(X0,Y0)) ) {
      //Avoid center of circle at origin, where there is no point-of-closest approach
      //Also, avoid float overfloat on circle center
      continue;
```

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

Answer: no

Adding some debug statements shows that we never enter the seed loop when we don’t get a seed in the above plots.

So, the failure comes for the Acts Orthogonal seeder itself.

Next steps

- Contact Acts experts about this issue and get suggestions