

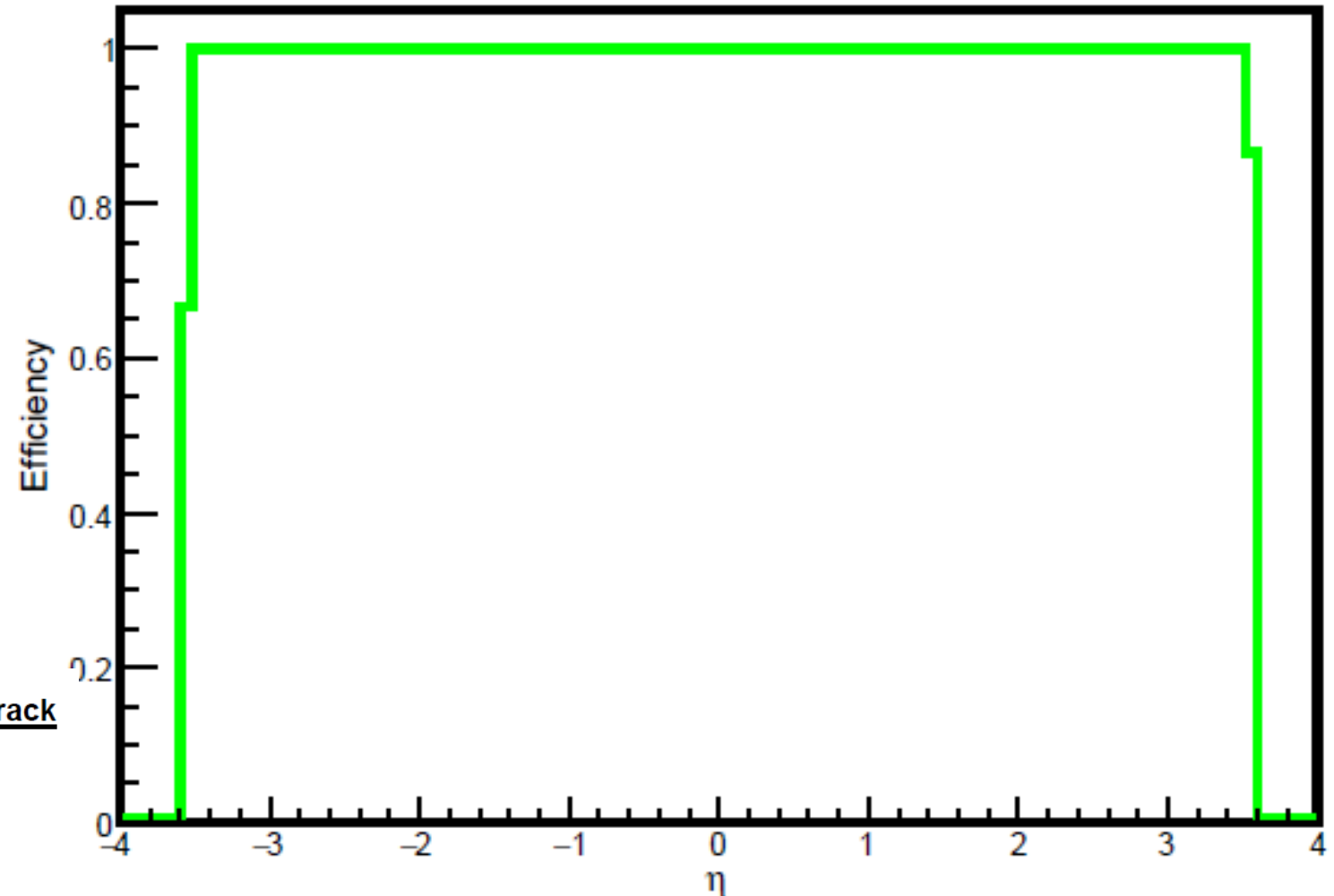
# Origin of seeding inefficiencies at low momentum

Barak Schmookler

# Good seeding and tracking efficiency seen at higher momentum

Tracker Efficiency vs. generated particle  $\eta$ :  $p_{\text{gen}} = 5.00 \text{ GeV}/c$

Generated particle:  
Single negative muon  
Eta = [-4,4]  
Phi = [0,2Pi]  
(Vx,Vy,Vz) = (0,0,0)



Efficiency in each  $\eta_{\text{gen}}$  bin =  $\frac{\text{Number of events w/ at least 1 track}}{\text{Number of events}}$

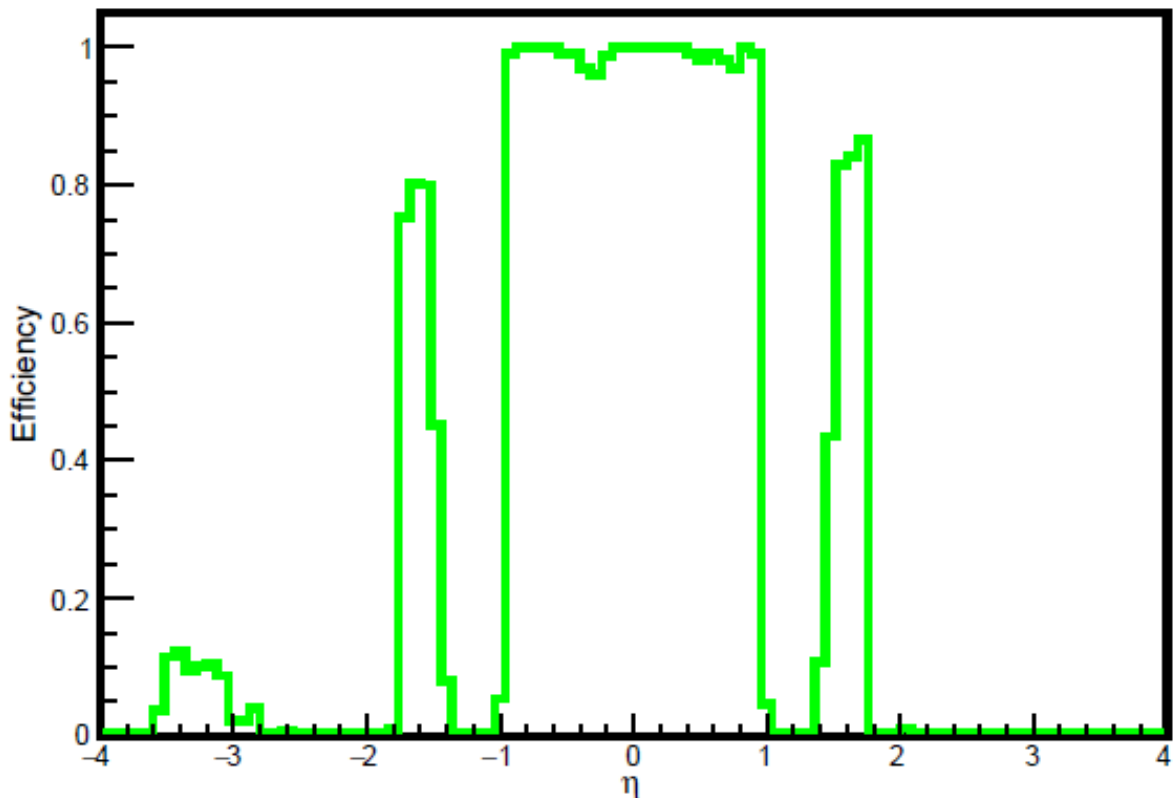
# Efficiency at low momentum

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

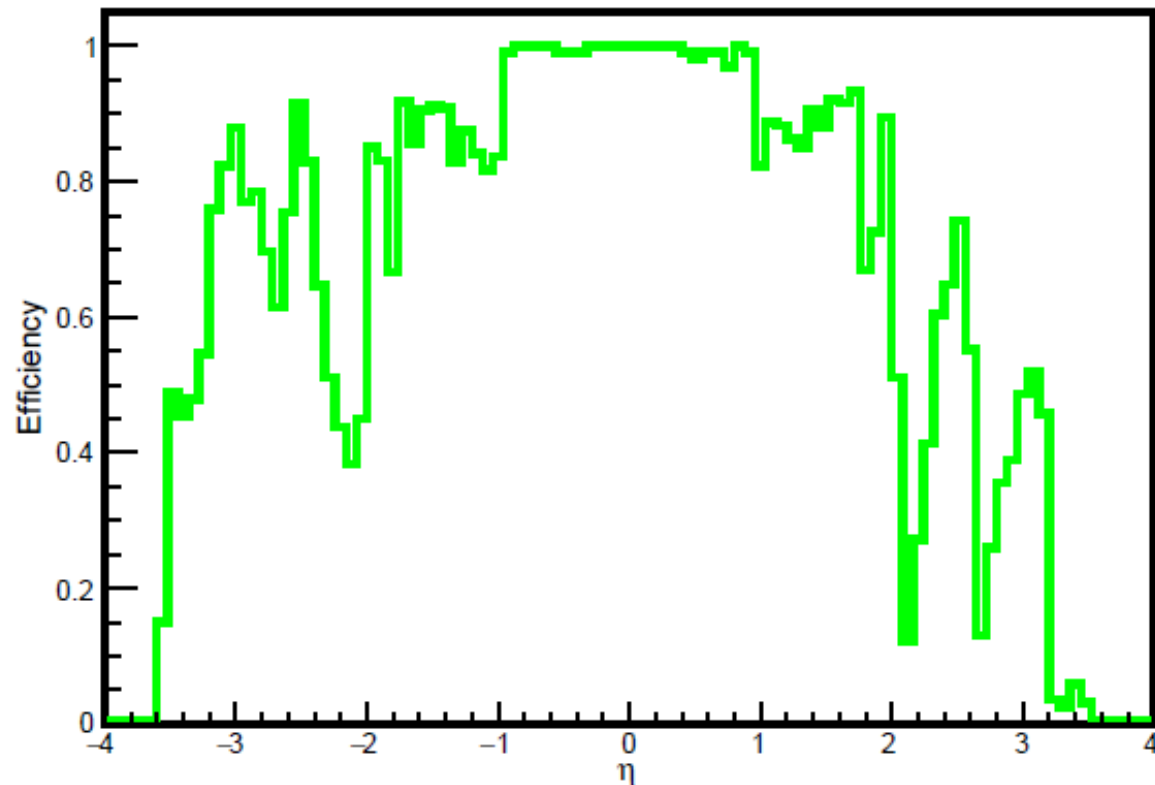
Default *deltaPhiMax* parameter = 0.085

Modified *deltaPhiMax* parameter = 0.16

Tracker Efficiency vs. generated particle  $\eta$ :  $p_{\text{gen}} = 0.50 \text{ GeV}/c$



Tracker Efficiency vs. generated particle  $\eta$ :  $p_{\text{gen}} = 0.50 \text{ GeV}/c$



# Efficiency at low momentum

10k single negative muons

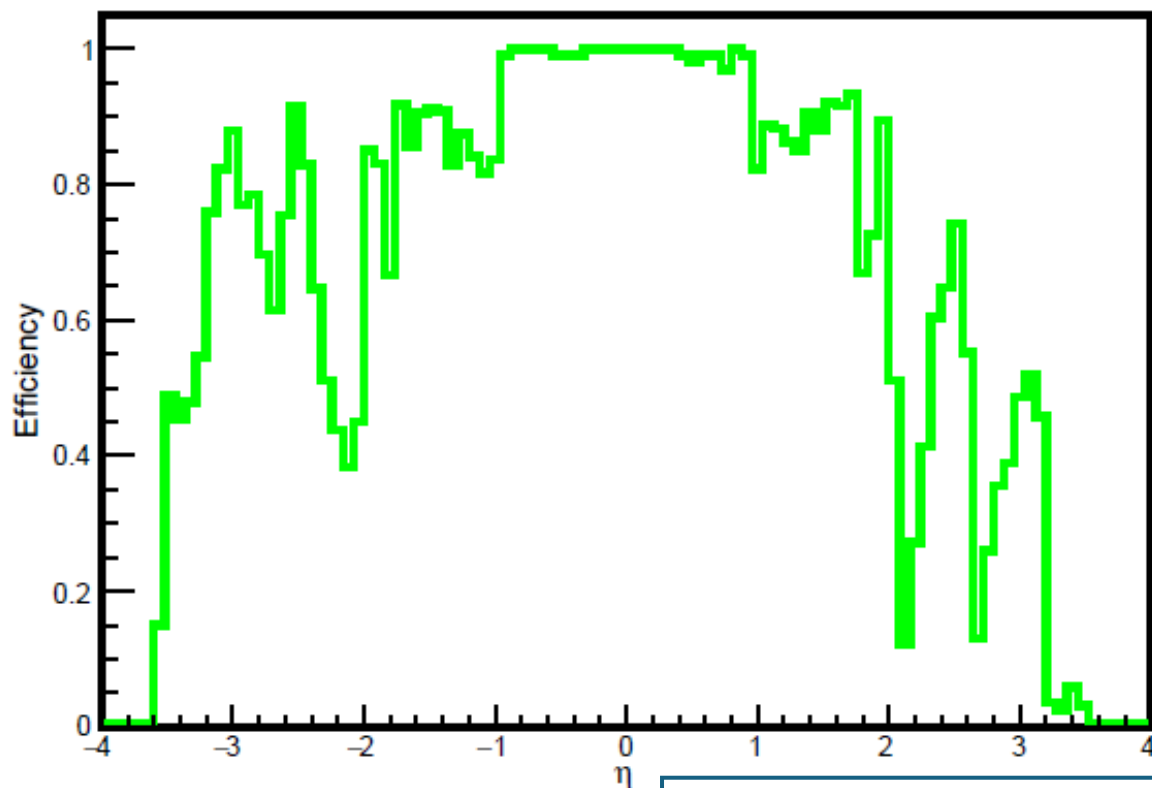
$\eta = [-4,4]$ ,  $\phi = [0,2\pi]$

$P = 500 \text{ MeV}/c$

$(v_x, v_y, v_z) = (0,0,0)$

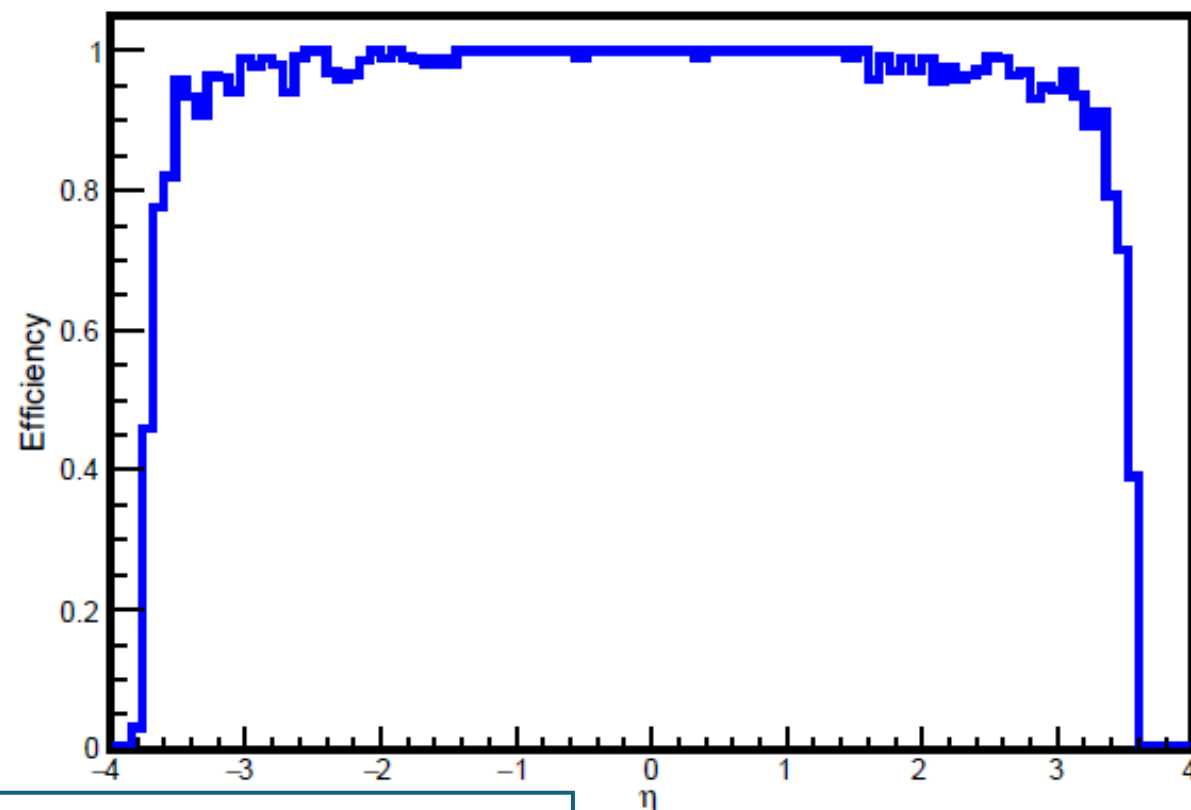
Modified  $\Delta\phi_{\text{Max}}$  parameter = 0.16

Tracker Efficiency vs. generated particle  $\eta$ :  $p_{\text{gen}} = 0.50 \text{ GeV}/c$



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

Tracker Efficiency vs. generated particle  $\eta$ :  $p_{\text{gen}} = 0.50 \text{ GeV}/c$



We should be able to get close to the truth-seeding efficiency result with realistic seed finding

# Test: Throw particles at fixed momentum and eta

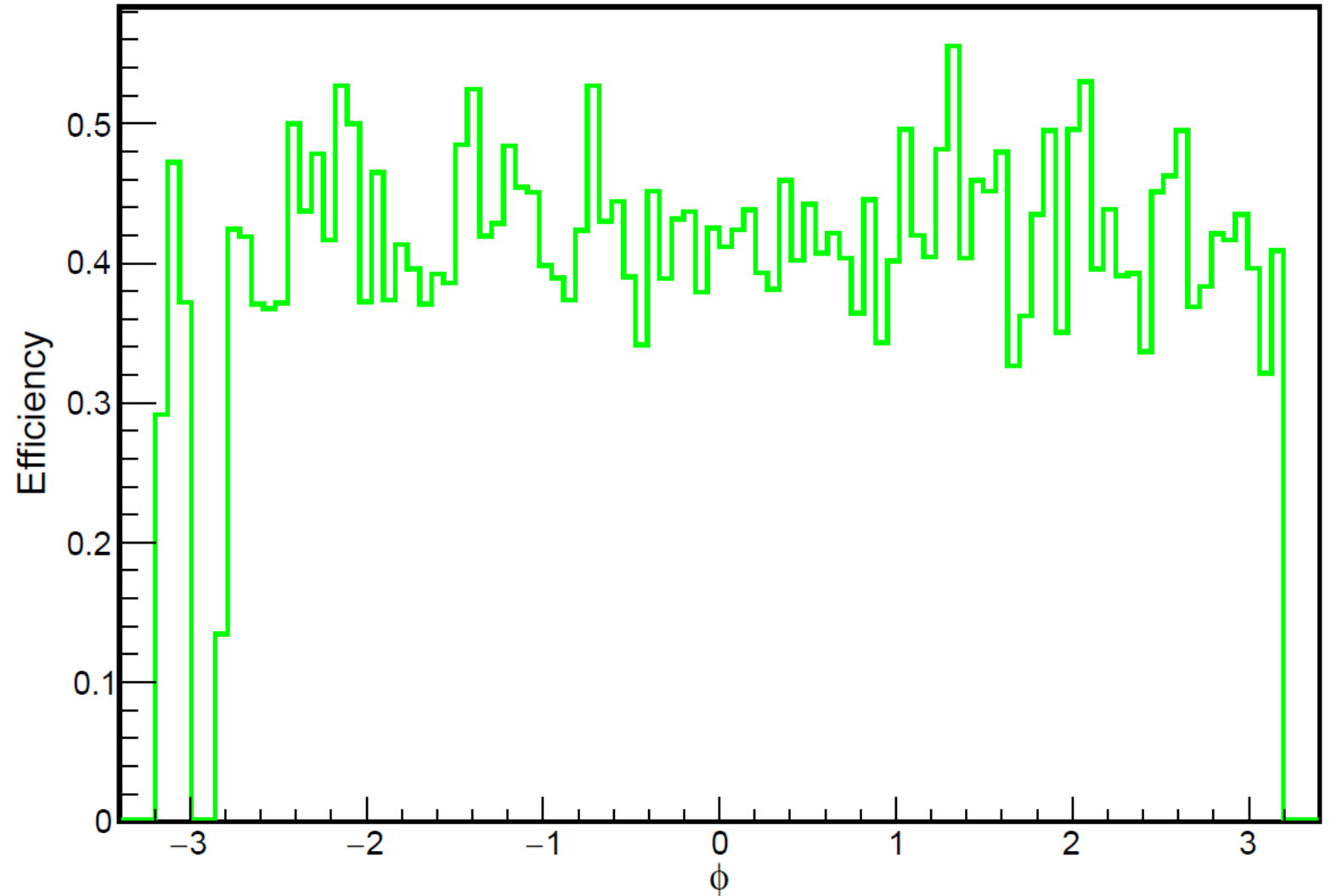
10k single negative muon events

Eta = -2.175, Phi = [0,2Pi]

P = 500 MeV/c

(vx,vy,vz) = (0,0,0)

Tracker Efficiency vs. generated particle  $\phi$ :  $p_{\text{gen}} = 0.50 \text{ GeV}/c$ ,  $\eta_{\text{gen}} = -2.175$



# Test: Throw particles at fixed momentum and eta

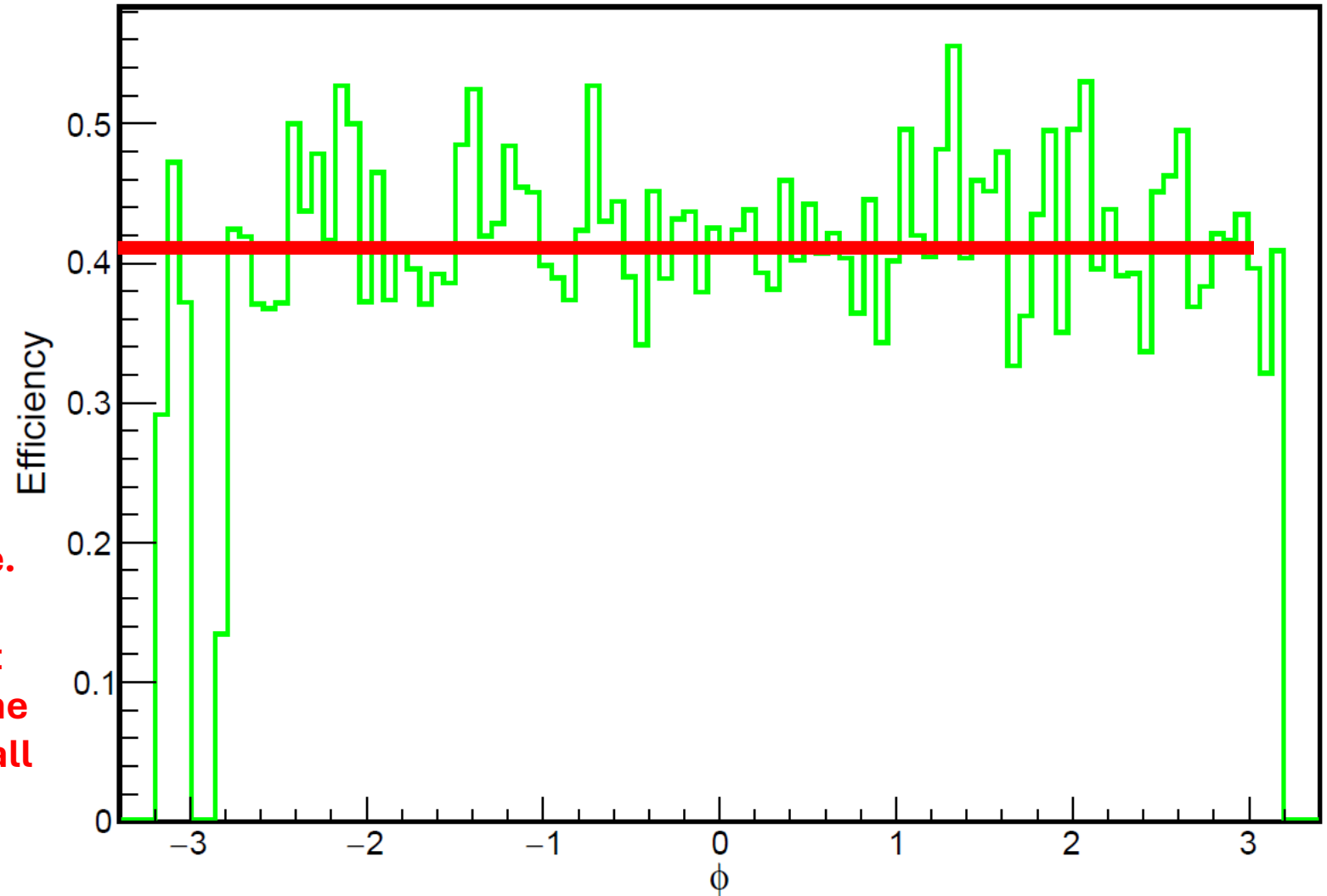
10k single negative muon events

Eta = -2.175, Phi = [0,2Pi]

P = 500 MeV/c

(vx,vy,vz) = (0,0,0)

Tracker Efficiency vs. generated particle  $\phi$ :  $p_{\text{gen}} = 0.50 \text{ GeV}/c$ ,  $\eta_{\text{gen}} = -2.175$

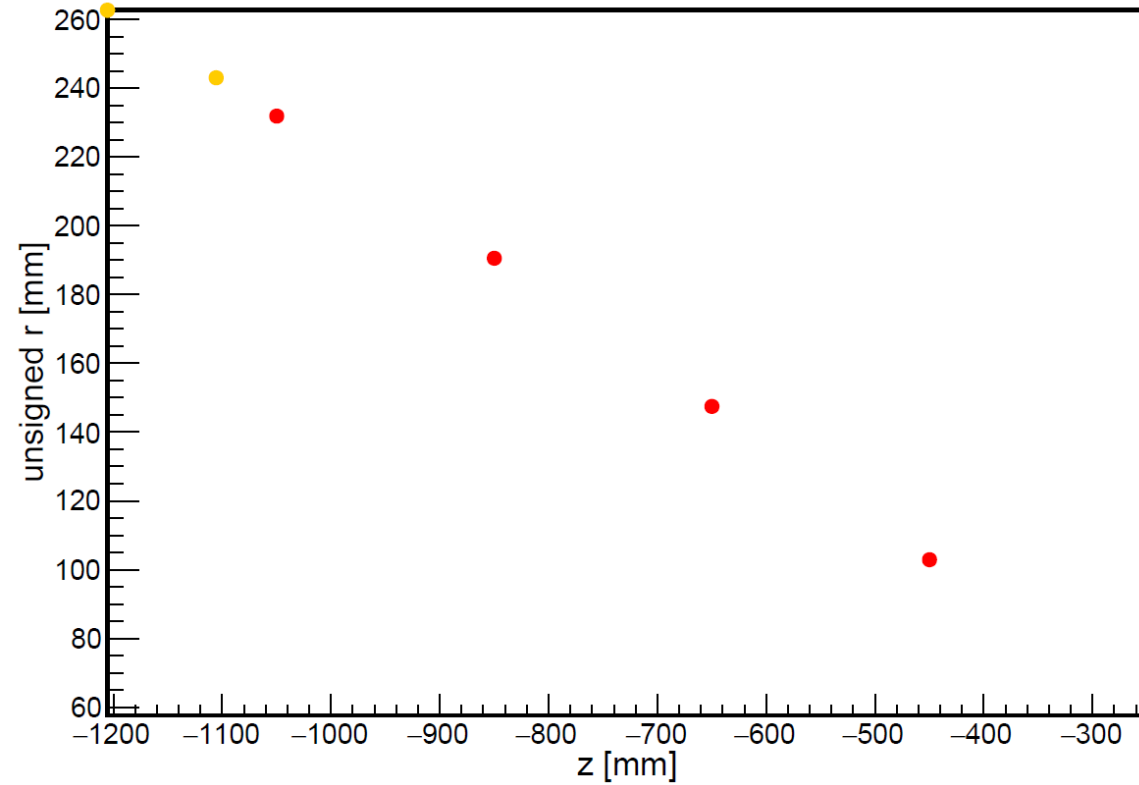


**We see inefficiency over whole phi range.**

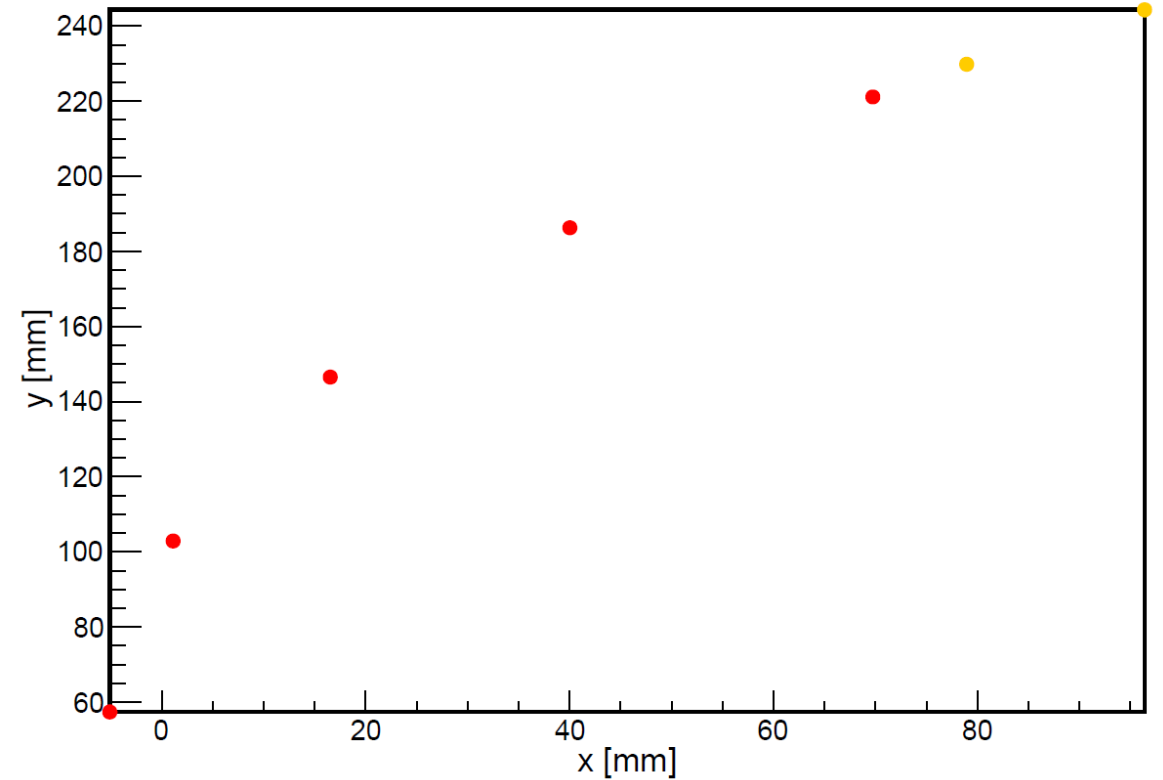
**Almost identical events can get different seeding results. This means there is some cut being applied that is sensitive to small fluctuations in detector hit positions.**

# Example event: Forms one seed

Tracker hits from primary particle for event 8500



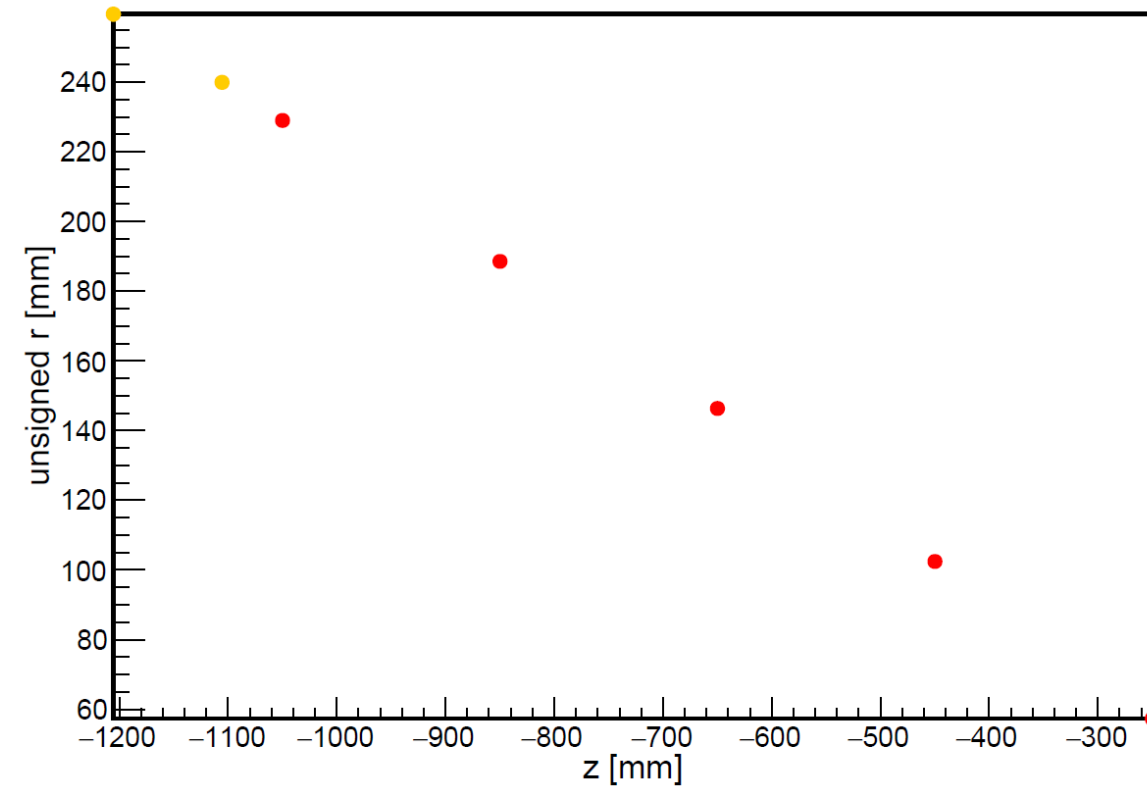
Tracker hits from primary particle for event 8500



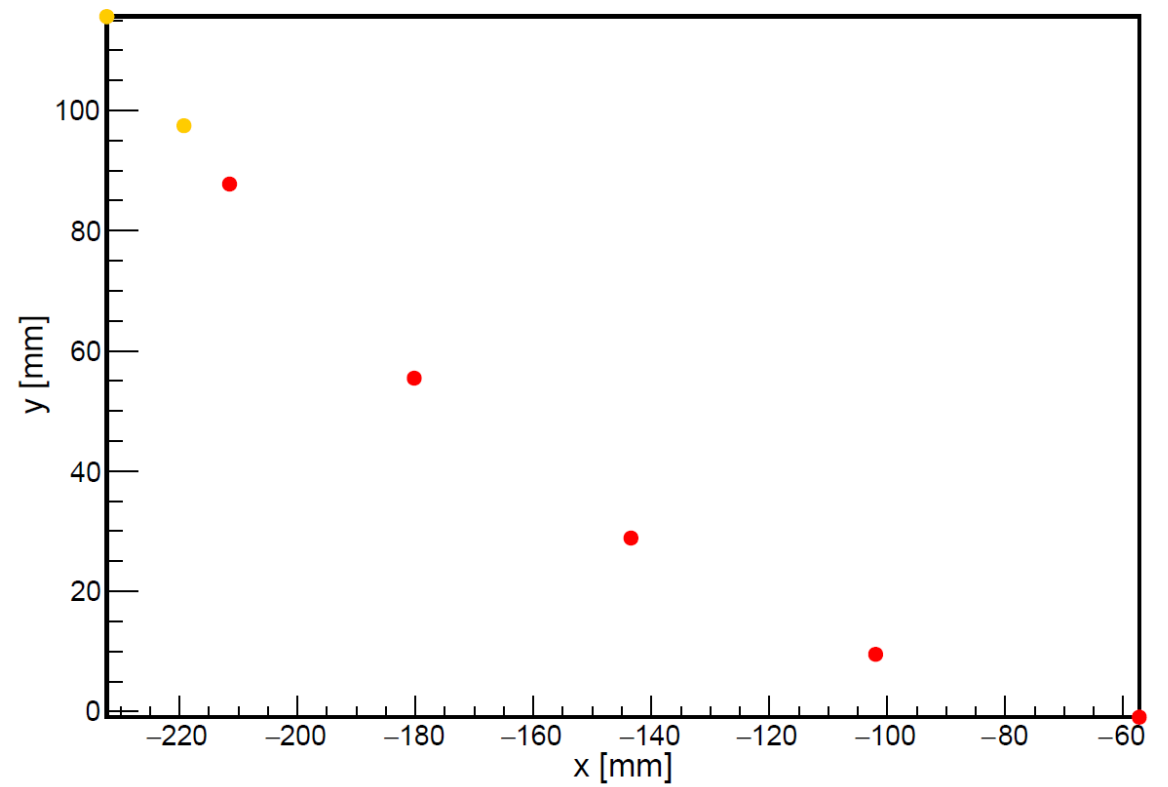
Single negative muon events  
Eta = -2.175, P = 500 MeV/c  
( $v_x, v_y, v_z$ ) = (0, 0, 0)

# Example event: Fails to form any seed

Tracker hits from primary particle for event 7



Tracker hits from primary particle for event 7



Single negative muon events

$\text{Eta} = -2.175$ ,  $P = 500 \text{ MeV}/c$

$(v_x, v_y, v_z) = (0, 0, 0)$

# Hardcoded cut in Acts Orthogonal seed finder

- There is a hardcoded cut in the Acts Orthogonal seed finder that is causing our failure to find seeds at low momentum for some particles.
- What is happening:
  - We have particles where valid bottom and top space point candidates are found around a middle space point (passing all doublet-level cuts including *validTuple*).
  - However, when *filterCandidates* attempts to combine these into triplets, all combinations are rejected by the hardcoded  $|\tan LM - \tan MT| > 0.005$  angular consistency cut.
  - This cut is too tight for our geometry, where the bottom–middle and middle–top slopes can differ by more than 0.005 rad for low-momentum particles.



```
362     }
363     if (m_config.seedConfirmation &&
364         candidates_collector.nHighQualityCandidates()) {
365         minCompatibleTopSPs++;
366     }
367
368     // clear all vectors used in each inner for loop
369     top_valid.clear();
370     curvatures.clear();
371     impactParameters.clear();
372
373     float tanLM = std::atan2(rM - bottom[b]->radius(), zM - bottom[b]->z());
374
375     for (std::size_t index_t = t0; index_t < numTopSP; index_t++) {
376         const std::size_t t = sorted_tops[index_t];
377         auto lt = linCircleTop[t];
378
379         if (std::abs(tanLM - tanMT[t]) > 0.005) {
380             continue;
381         }
```

# How to resolve this

Opened an Acts issue. The current implementation of the seeder we are using is deprecated. The new implementation does not have this hardcoded cut.

Open Hardcoded cut in Orthogonal seed finder #5286



andiwand last month

Contributor ...

It definitely can but note that this implementation is about to be removed in an upcoming major release. The new implementation of the generic triplet seeder is here <https://github.com/acts-project/acts/blob/v46.0.0/Core/include/Acts/Seeding2/TripletSeeder.hpp>

How to build the orthogonal seeder form this is exercised here <https://github.com/acts-project/acts/blob/v46.0.0/Examples/Algorithms/TrackFinding/include/ActsExamples/TrackFinding/OrthogonalTripletSeedingAlgorithm.hpp>

I don't see such a cut in the new generic filter implementation here <https://github.com/acts-project/acts/blob/v46.0.0/Core/src/Seeding2/BroadTripletSeedFilter.cpp> but in case you need different cuts they can be added, if they are generic enough, or you can implement your own filter <https://github.com/acts-project/acts/blob/v46.0.0/Core/include/Acts/Seeding2/TripletSeedFilter.hpp>



<https://github.com/acts-project/acts/issues/5286>

# How to resolve this

Opened an Acts issue. The current implementation of the seeder we are using is deprecated. The new implementation does not have this hardcoded cut.

The next step then is to incorporate the new Acts seeding implementation into EICRecon. So, my plan is to start helping with this

Open Hardcoded cut in Orthogonal seed finder #5286



andiwand last month

Contributor

It definitely can but note that this implementation is about to be removed in an upcoming major release. The new implementation of the generic triplet seeder is here <https://github.com/acts-project/acts/blob/v46.0.0/Core/include/Acts/Seeding2/TripletSeeder.hpp>

How to build the orthogonal seeder from this is exercised here <https://github.com/acts-project/acts/blob/v46.0.0/Examples/Algorithms/TrackFinding/include/ActsExamples/TrackFinding/OrthogonalTripletSeedingAlgorithm.hpp>

I don't see such a cut in the new generic filter implementation here <https://github.com/acts-project/acts/blob/v46.0.0/Core/src/Seeding2/BroadTripletSeedFilter.cpp> but in case you need different cuts they can be added, if they are generic enough, or you can implement your own filter <https://github.com/acts-project/acts/blob/v46.0.0/Core/include/Acts/Seeding2/TripletSeedFilter.hpp>

<https://github.com/acts-project/acts/issues/5286>

Seeding2 algorithm #2524

Draft wdconinc wants to merge 6 commits into main from seeding2-algorithm

Conversation 1 Commits 6 Checks 280 Files changed 7



wdconinc commented on Feb 28

Member

Briefly, what does this PR introduce?

Needs:

- [Adapt seeding algorithm for Acts >= 46 removal of ActsExamples::SpacePointContainer<T>](#) #2517

This PR adds the orthogonal track seeding algorithm in the Seeding2 API of Acts v46+.

What kind of change does this PR introduce?

- Bug fix (issue #\_)
- New feature (issue: upgraded seeding API)
- Documentation update
- Other: \_

Please check if this PR fulfills the following:

<https://github.com/eic/EICrecon/pull/2524>