Single-particle simulations with new seeding framework

Barak Schmookler & Jeetendra Gupta

Motivation

- We recently updated our seeding infrastructure in EICRecon.
  - Previously, we passed our tracker hits into an algorithm called the Acts Orthogonal seed finder.
  - Now, we use the updated, modular seed-finding infrastructure in Acts. (Details on next slide.)
- In this presentation, we look at the seed-finding and track-finding efficiency for single-particle simulations with the old (Orthogonal) and new (Seeding2) approaches.
- We can do direct comparisons between both approaches in our current version of EICRecon.

Seeding2: ACTS Orthogonal Triplet Seed Finder

How triplets are built, and how they're scored

Build KD-tree

index all space points
in (ϕ, r, z)

Middle SP cuts

r / z / ϕ region
of interest

Box search

Δϕ, Δr, Δz windows →
candidate bottoms & tops

Doublet + helix fit

cotTheta, impact,
scattering cuts

Seed filter

weight, rank,
cap per middle SP

Seed weight (computed per top candidate, for a fixed bottom–middle doublet)

$$w = c_1 \cdot N_t - c_2 \cdot d_0 \quad \left(- c_3 \cdot |z_0| \;\; \textit{only if seedConfirmation = true} \right)$$

N_t = number of *other* top candidates (same bottom–middle pair) that are curvature-consistent and ≥ Δr_{min} apart, capped at compatSeedLimit. d_0, z_0 = impact parameter, vertex z of the fitted helix.

maxSeedsPerSpM keeps only the top-(maxSeedsPerSpM + 1) weighted seeds *per middle space point* — the cap is applied independently for every middle SP

Only φ, r, z drive candidate finding — no timing information is used at any stage of this algorithm.

Simulation configuration

```bash
if [ "$run_npsim" -eq 1 ]; then
    npsim --compactFile "$DETECTOR_PATH/epic_craterlake.xml" --enableGun --gun.distribution 'eta' \
        --gun.thetaMax 3.106 --gun.thetaMin 0.036 \
        --gun.momentumMin "${p}*GeV" --gun.momentumMax "${p}*GeV" \
        --numberOfEvents "${nevents}" --outputFile "$simfile"
fi

# tag:seedingMethod pairs -- add more here if another method is needed
local seed_configs=("orth:orthogonal" "seed2:seeding2")
local cfg tag seedmethod dpm
for dpm in "${delta_phi_values[@]}"; do
    for cfg in "${seed_configs[@]}"; do
        tag=${cfg%%:*}
        seedmethod=${cfg##*:}

        eicrecon -Ppodio:output_file="output/eicrecon_${tag}_delPhiMax_${dpm}_${label}.root" \
            -Pjana:nevents="${nevents}" \
            -Pectrk:siendcaptrackernoiserawhits:addnoise=0 \
            -Pbvtx:sibarrelvertexnoiserawhits:addnoise=0 \
            -Pbtrk:sibarrelnoiserawhits:addnoise=0 \
            -Ptracking:CentralTrackSeeds:deltaPhiMax="${dpm}" \
            -PTRACKING:CentralTrackSeeds:seedingMethod="${seedmethod}" \
            -Pdd4hep:xml_files=epic_craterlake.xml "$simfile" \
            | tee "output/log/out_${tag}_delPhiMax_${dpm}_${label}.txt"
    done
```

Simulation configuration

```bash
if [ "$run_npsim" -eq 1 ]; then
    npsim --compactFile "$DETECTOR_PATH/epic_craterlake.xml" --enableGun --gun.distribution 'eta' \
        --gun.thetaMax 3.106 --gun.thetaMin 0.036 \
        --gun.momentumMin "${p}*GeV" --gun.momentumMax "${p}*GeV" \
        --numberOfEvents "${nevents}" --outputFile "$simfile"
fi

# tag:seedingMethod pairs -- add more here if another method is need
local seed_configs=("orth:orthogonal" "seed2:seeding2")
local cfg tag seedmethod dpm
for dpm in "${delta_phi_values[@]}"; do
    for cfg in "${seed_configs[@]}"; do
        tag=${cfg%%:*}
        seedmethod=${cfg##*:}

        eicrecon -Ppodio:output_file="output/eicrecon_${tag}_delPhiMax_${dpm}_${label}.root" \
            -Pjana:nevents="${nevents}" \
            -Pectrk:siendcaptrackernoiserawhits:addnoise=0 \
            -Pbvtx:sibarrelvertexnoiserawhits:addnoise=0 \
            -Pbtrk:sibarrelnoiserawhits:addnoise=0 \
            -Ptracking:CentralTrackSeeds:deltaPhiMax="${dpm}" \
            -PTRACKING:CentralTrackSeeds:seedingMethod="${seedmethod}" \
            -Pdd4hep:xml_files=epic_craterlake.xml "$simfile" \
            | tee "output/log/out_${tag}_delPhiMax_${dpm}_${label}.txt"
    done
```

Single-muon simulations at fixed momentum from eta = [-4,4] and (vx, vy, vz) = (0,0,0): 0.3, 0.4, 0.5, 1.0, 2.0, 5.0, 10.0, 20.0 GeV/c

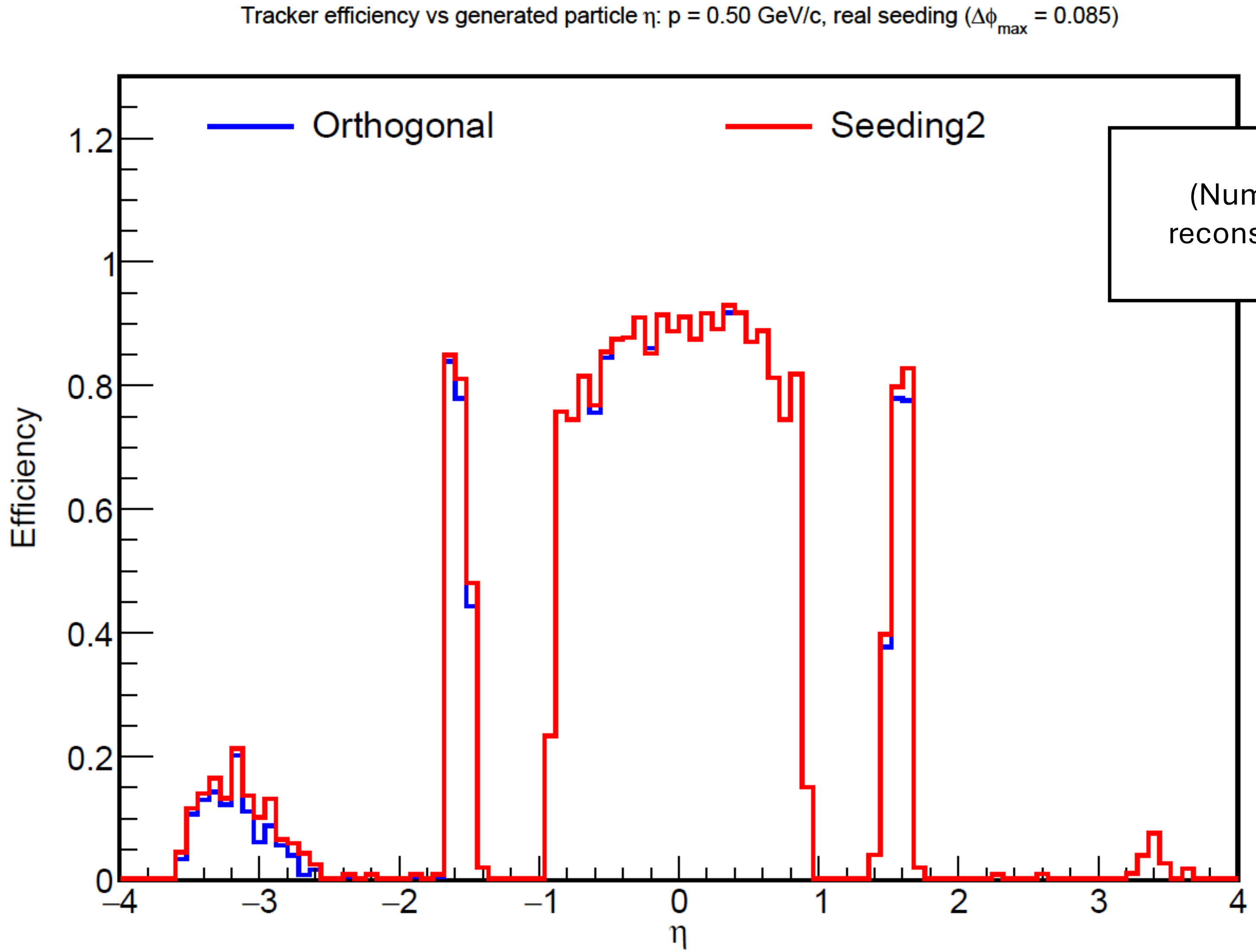
Simulation configuration

```
if [ "$run_npsim" -eq 1 ]; then
    npsim --compactFile "$DETECTOR_PATH/epic_craterlake.xml" -
        --gun.thetaMax 3.106 --gun.thetaMin 0.036 \
        --gun.momentumMin "${p}*GeV" --gun.momentumMax "${p}*G
        --numberOfEvents "${nevents}" --outputFile "$simfile"
fi

# tag:seedingMethod pairs -- add more here if another method is needed
local seed_configs=("orth:orthogonal" "seed2:seeding2")
local cfg tag seedmethod dpm
for dpm in "${delta_phi_values[@]}"; do
    for cfg in "${seed_configs[@]}"; do
        tag=${cfg%%:*}
        seedmethod=${cfg##*:}

        eicrecon -Ppodio:output_file="output/eicrecon_${tag}_delPhiMax_${dpm}_${label}.root" \
            -Pjana:nevents="${nevents}" \
            -Pectrk:siendcaptrackernoiserawhits:addnoise=0 \
            -Pbvtx:sibarrelvertexnoiserawhits:addnoise=0 \
            -Pbtrk:sibarrelnoiserawhits:addnoise=0 \
            -Ptracking:CentralTrackSeeds:deltaPhiMax="${dpm}" \
            -PTRACKING:CentralTrackSeeds:seedingMethod="${seedmethod}" \
            -Pdd4hep:xml_files=epic_craterlake.xml "$simfile" \
            | tee "output/log/out_${tag}_delPhiMax_${dpm}_${label}.txt"
    done
```

For each simulation, run EICRecon with different seeding configuration and different phi search window maximum.

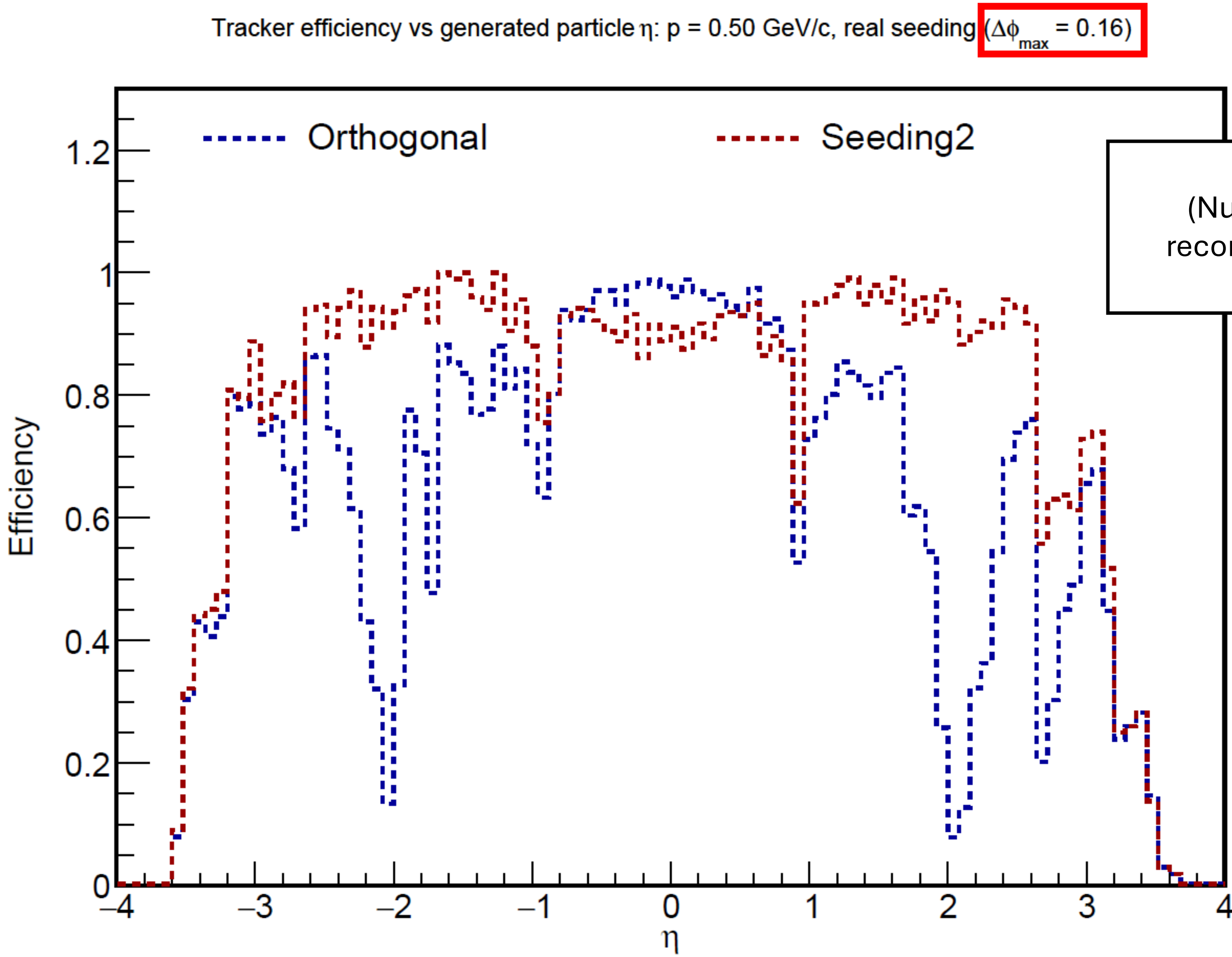
Note how SVT noise is turned off for these simulations.

Results at low momentum: p = 0.5 GeV/c



Efficiency definition:
(Number of events with at least one reconstructed track) / (Total number of events in true eta bin)

Results at low momentum: p = 0.5 GeV/c



Efficiency definition:
(Number of events with at least one reconstructed track) / (Total number of events in true eta bin)

Results at low momentum: p = 0.5 GeV/c

Tracker efficiency vs generated particle η: p = 0.50 GeV/c, real seeding $\boxed{(\Delta\phi_{max} = 0.16)}$

Hardcoded cut in Orthogonal seed finder #5286

Closed

bschmookler opened on Mar 25

We are using the Acts Orthogonal seed finder in ePIC. We observe some seeding inefficiencies in the transition region from the barrel to the endcap for lower momentum particles. I think the issue is that a hardcoded cut is applied here: https://github.com/acts-project/acts/blob/v46.0.0/Core/include/Acts/Seeding/SeedFinderOrthogonal.ipp#L379.

As I understand the code, 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 |tanLM − tanMT| > 0.005 angular consistency cut. This cut is too tight for our endcap geometry, where non-uniform layer spacing in z causes the bottom–middle and middle–top slopes to differ by more than 0.005 rad even for genuine tracks.

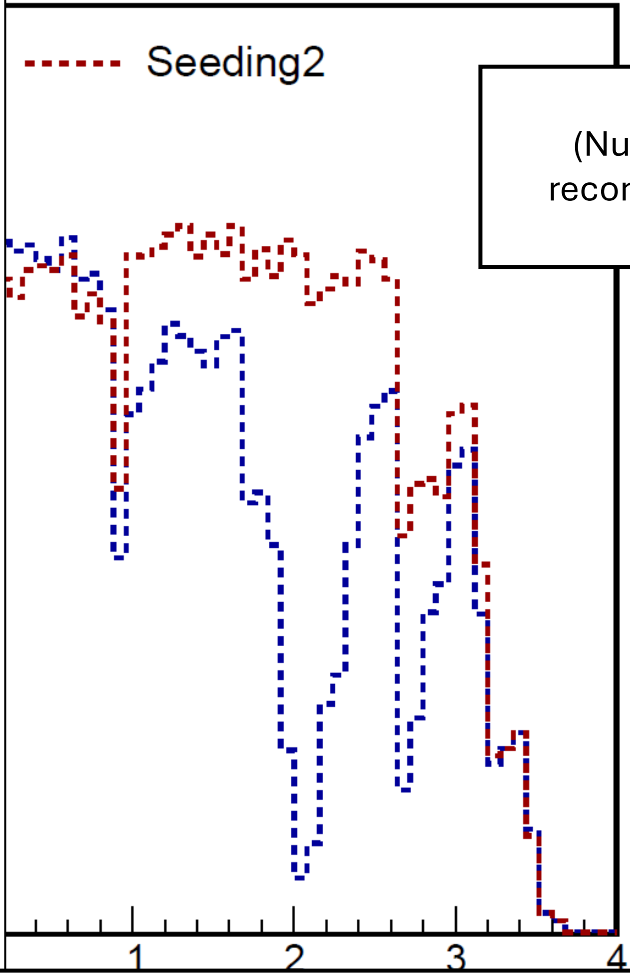
Can this cut be replaced by a configurable parameter? Or perhaps the cut can be removed since a cut on deltaCotTheta2 is applied a few lines after?

andiwand on Mar 25 — 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

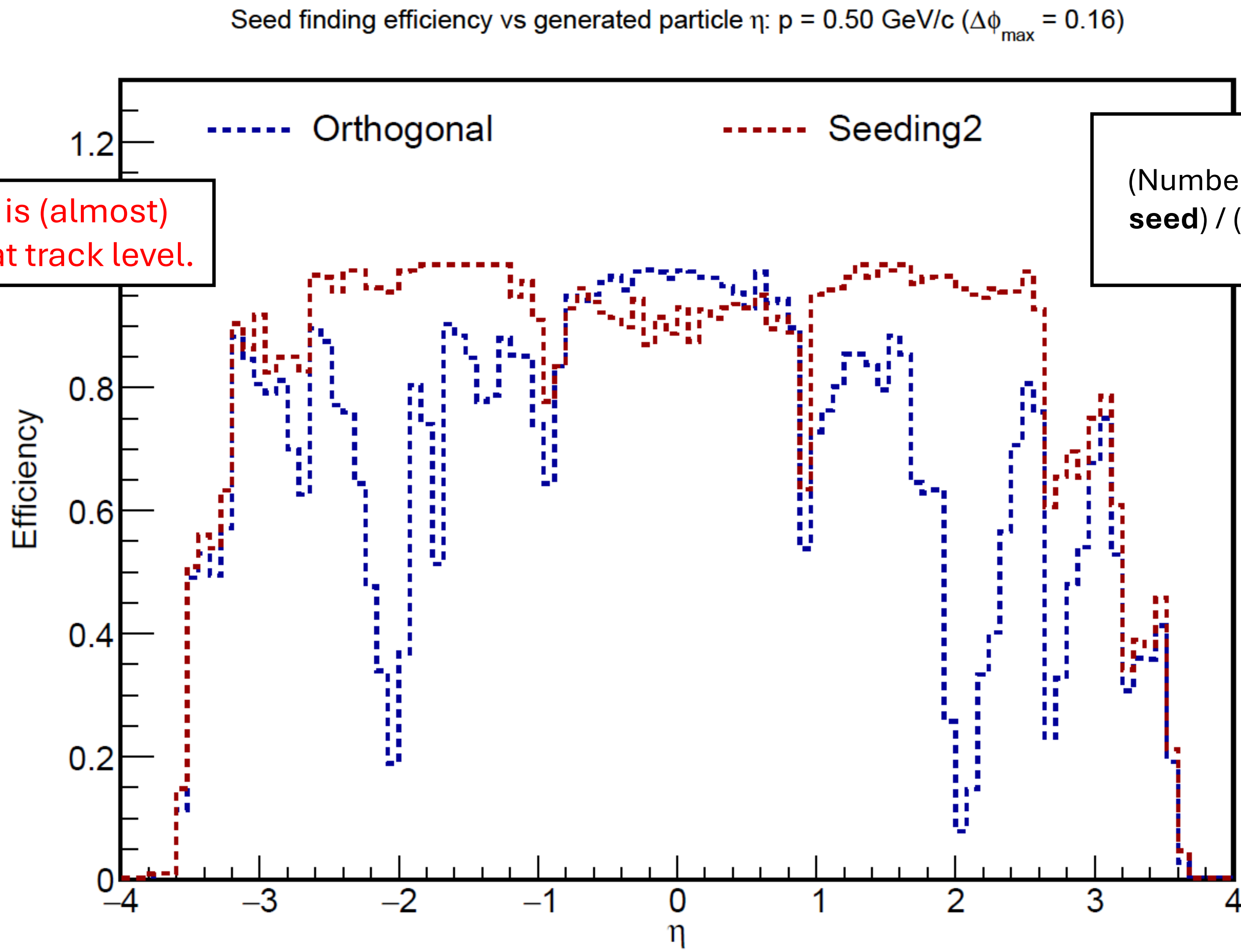
I don't see such a cut in the new generic filter implementation here https://github.com/acts-project/acts/blob/v46.0.0/Core/src/Seeding2/BroadTripletSeedFilter.cpp
but in case you need different cuts they can be added, if they are generic enough, or you can implement your own filter https://github.com/acts-project/acts/blob/v46.0.0/Core/include/Acts/Seeding2/ITr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



Efficiency definition:
(Number of events with at least one reconstructed track) / (Total number of events in true eta bin)

Result makes sense, since previously-identified hardcoded cut is not present in new seeding framework.

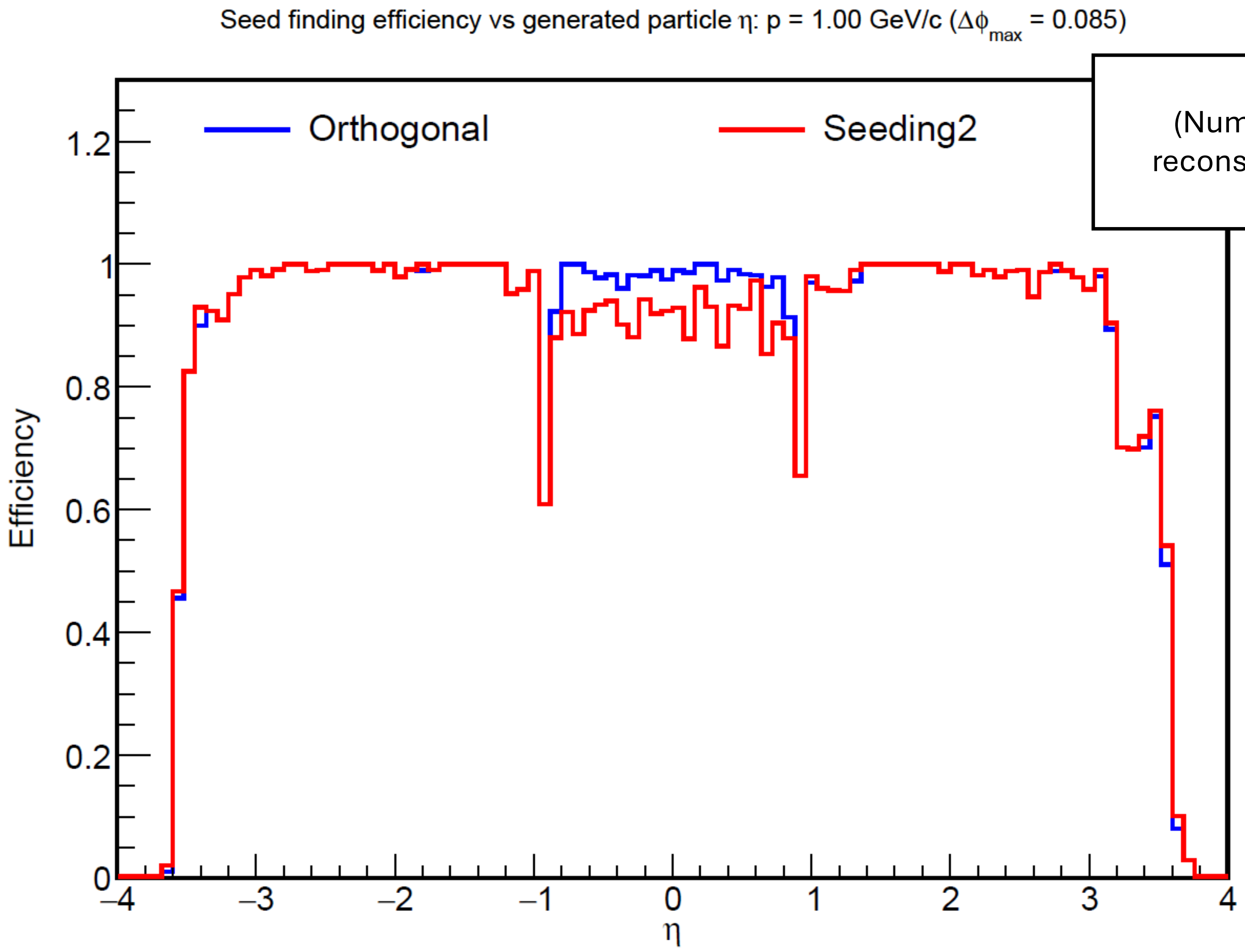
Results at low momentum: p = 0.5 GeV/c; seed-level



Efficiency at seed level is (almost) equivalent to efficiency at track level.

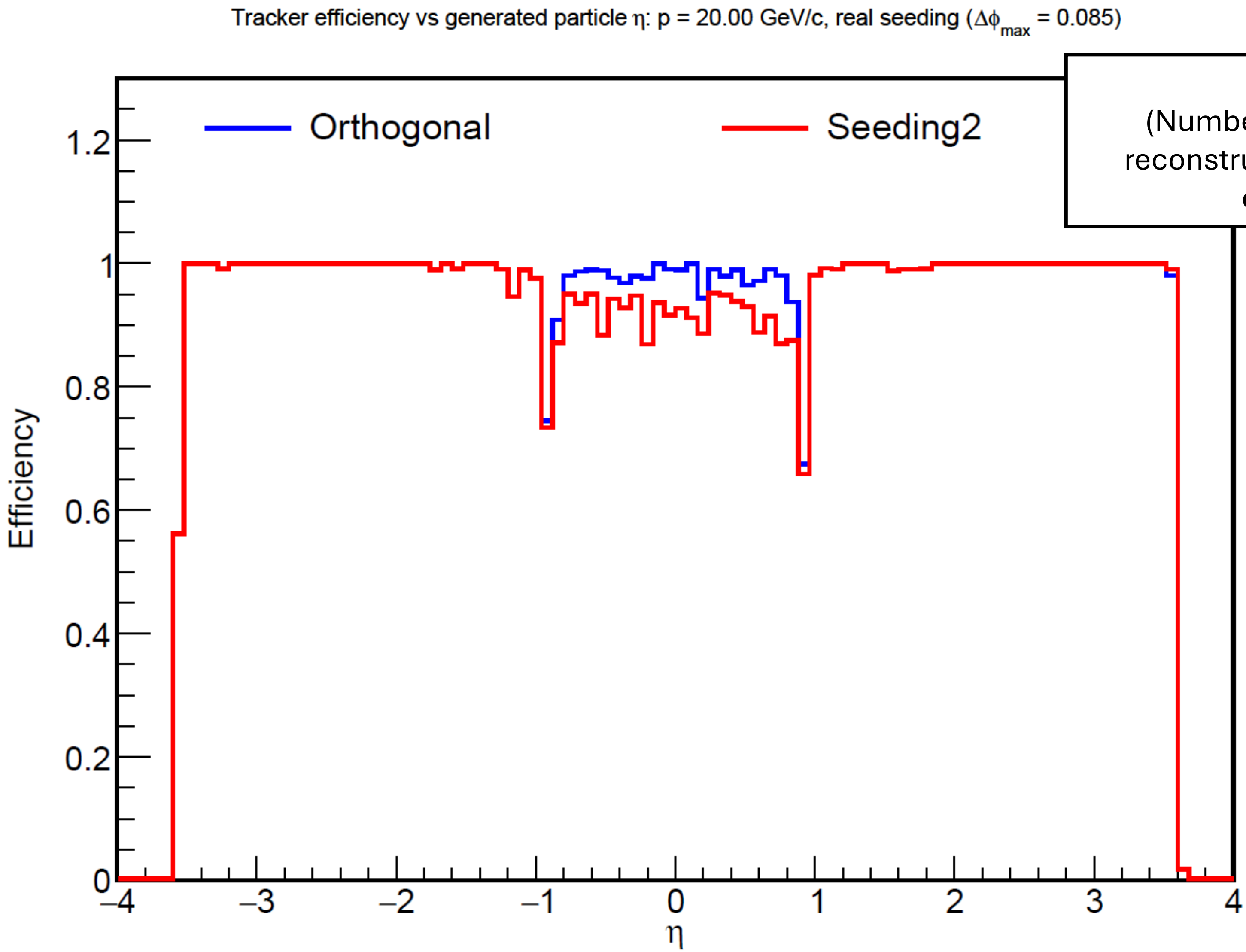
Efficiency definition:
(Number of events with at least one found **seed**) / (Total number of events in true eta bin)

Results at higher momentum: p = 1 GeV/c



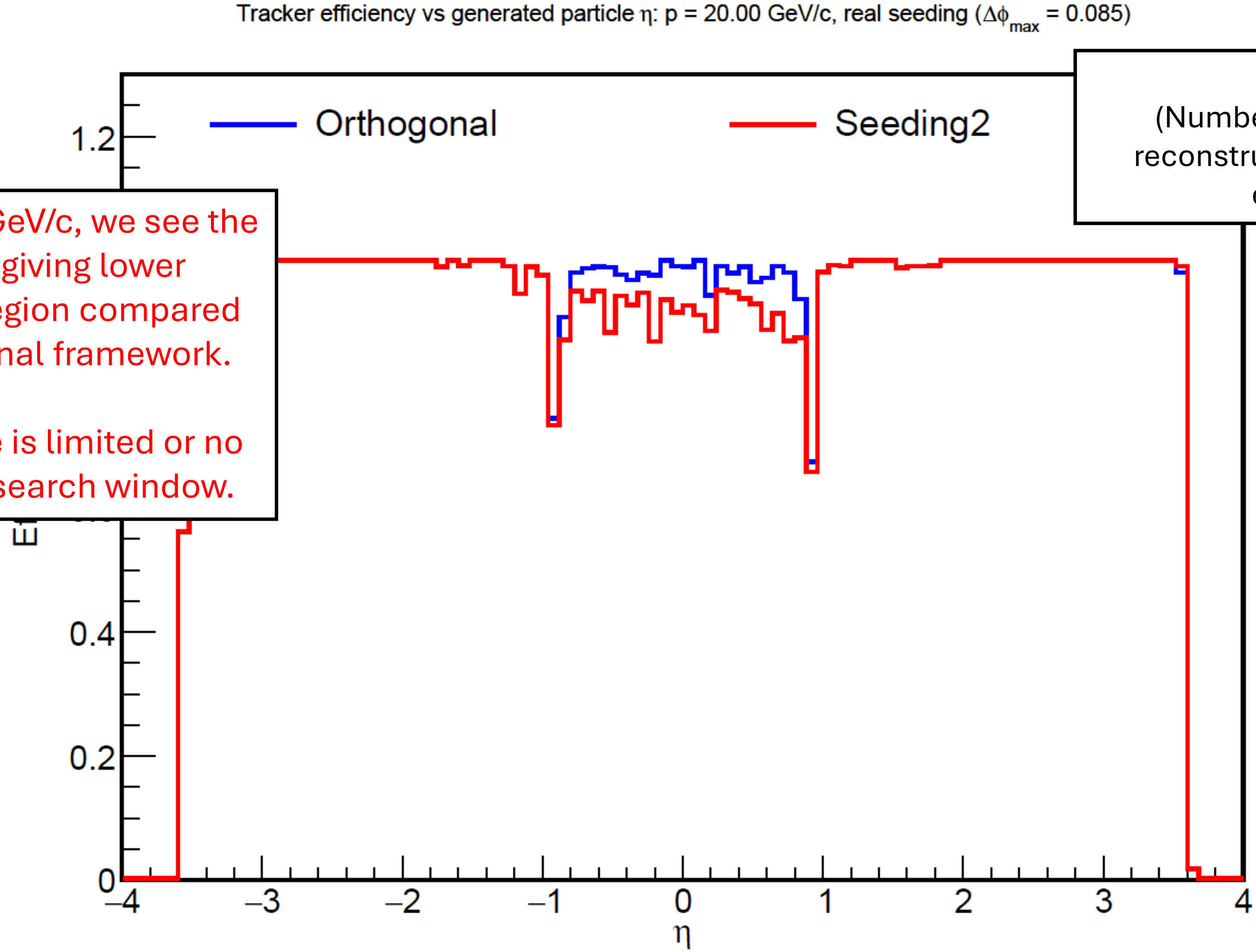
Efficiency definition:
(Number of events with at least one reconstructed track) / (Total number of events in true eta bin)

Results at higher momentum: p = 20 GeV/c



Efficiency definition:
(Number of events with at least one reconstructed track) / (Total number of events in true eta bin)

Results at higher momentum: p = 20 GeV/c

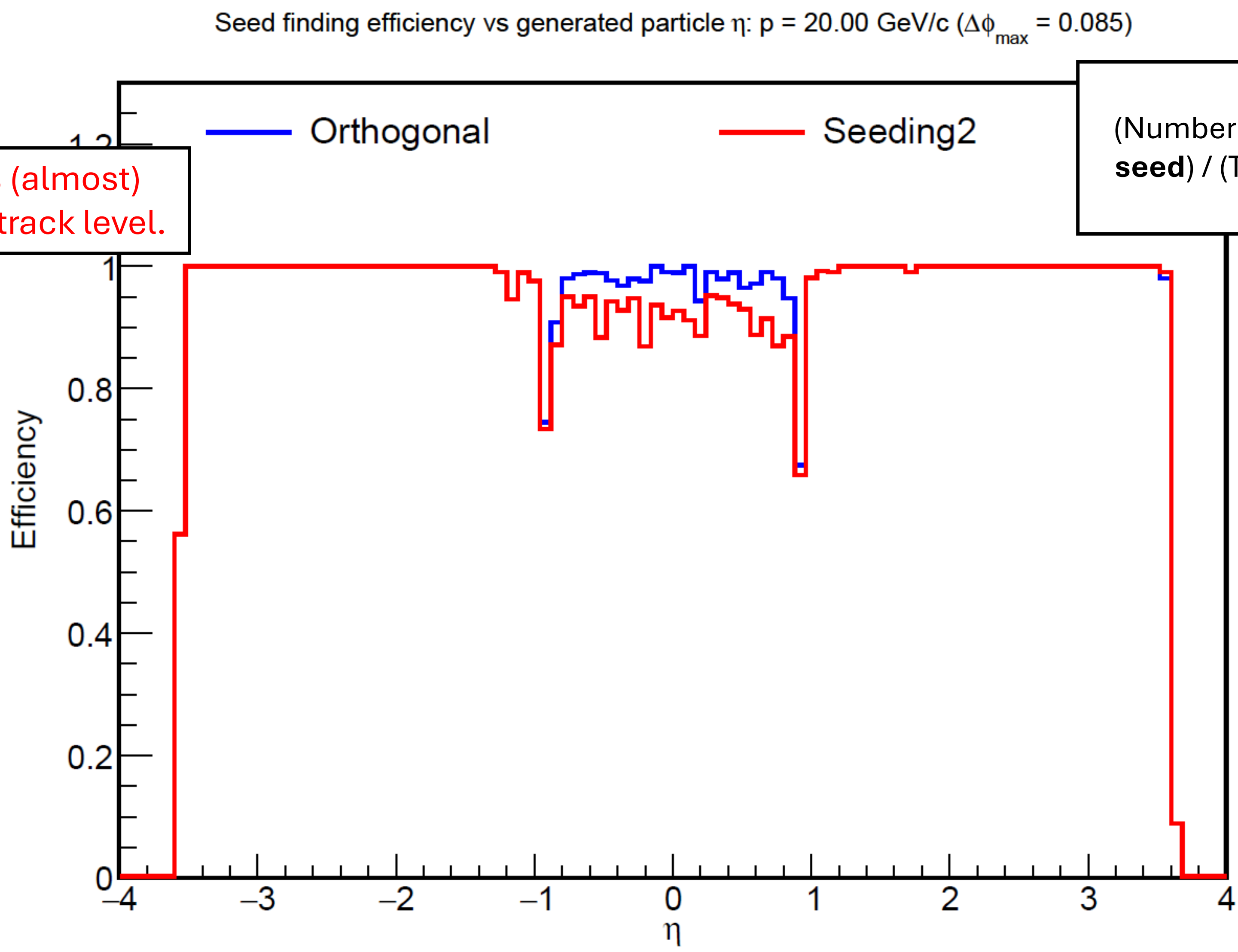


Efficiency definition:
(Number of events with at least one reconstructed track) / (Total number of events in true eta bin)

For simulations above 1 GeV/c, we see the Seeding2 framework giving lower efficiency in the barrel region compared to the previous Orthogonal framework.

At these momenta, there is limited or no impact of changing the search window.

Results at higher momentum: p = 20 GeV/c; seed level



Efficiency at seed level is (almost) equivalent to efficiency at track level.

Efficiency definition:
(Number of events with at least one found **seed**) / (Total number of events in true eta bin)

Conclusions/Next steps

- Conclusions:
  - At low momentum (500 MeV/c and below), we see that the seeding2 framework works better than the older Orthogonal algorithm when we increase the *deltaPhiMax* parameter. This is expected, since the new seeding framework does not contain the hard-coded cut we had before.
  - At higher momentum (1 GeV/c and above), we see that the new seeding 2 framework gives lower efficiency in the barrel region. This inefficiency exists at the seed level (and therefore at the track level).
- Next steps:
  - Single-particle simulations: Identify the reason for the lower efficiency in the barrel region.
  - DIS simulations: perform a similar seed-finding efficiency and tracking efficiency study with DIS events.