Update on eic-smear work for photon-eta-fix

Mark Ddamulira
Department of Physics & Astronomy
Michigan State University

June 25, 2025

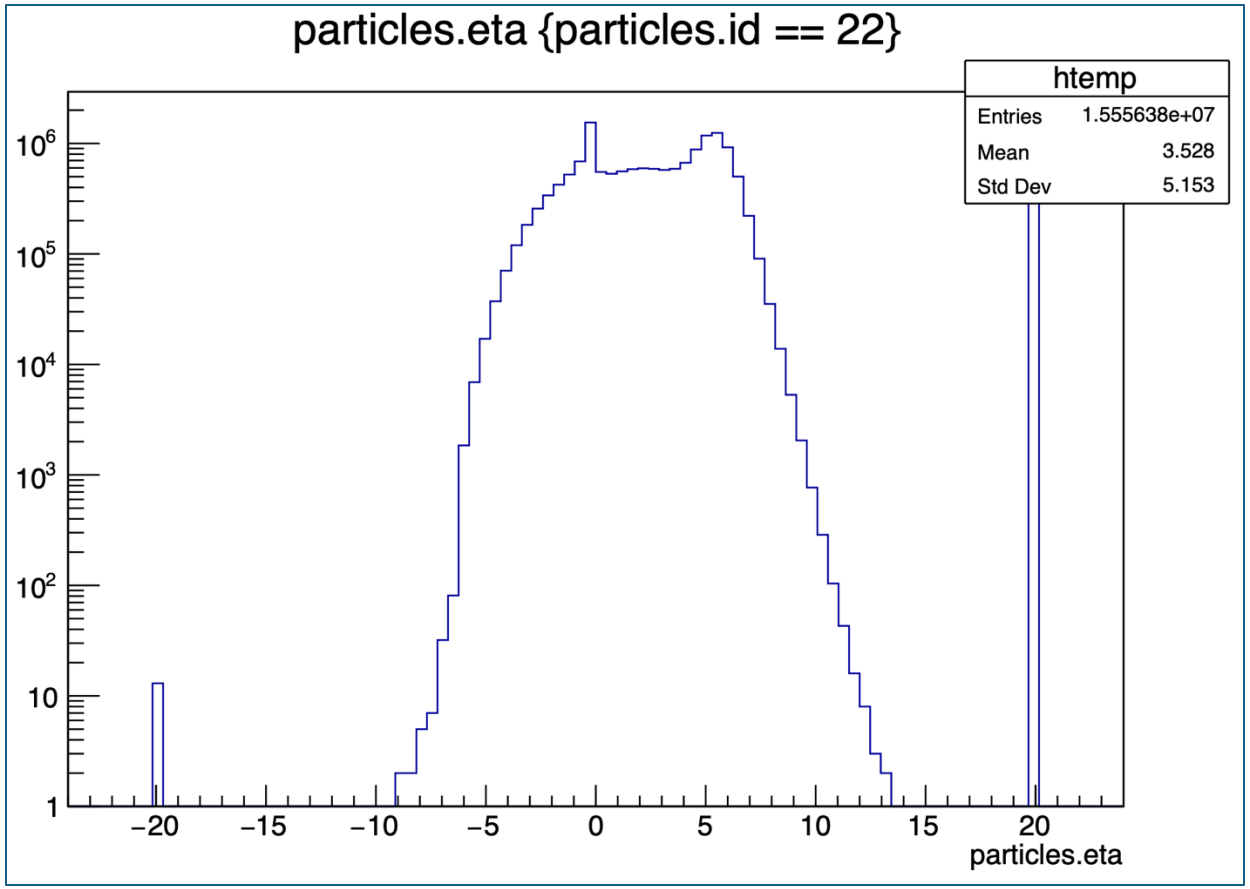
Outline

- Brief recap
- Comparing results from eic-smear versions
- Summary

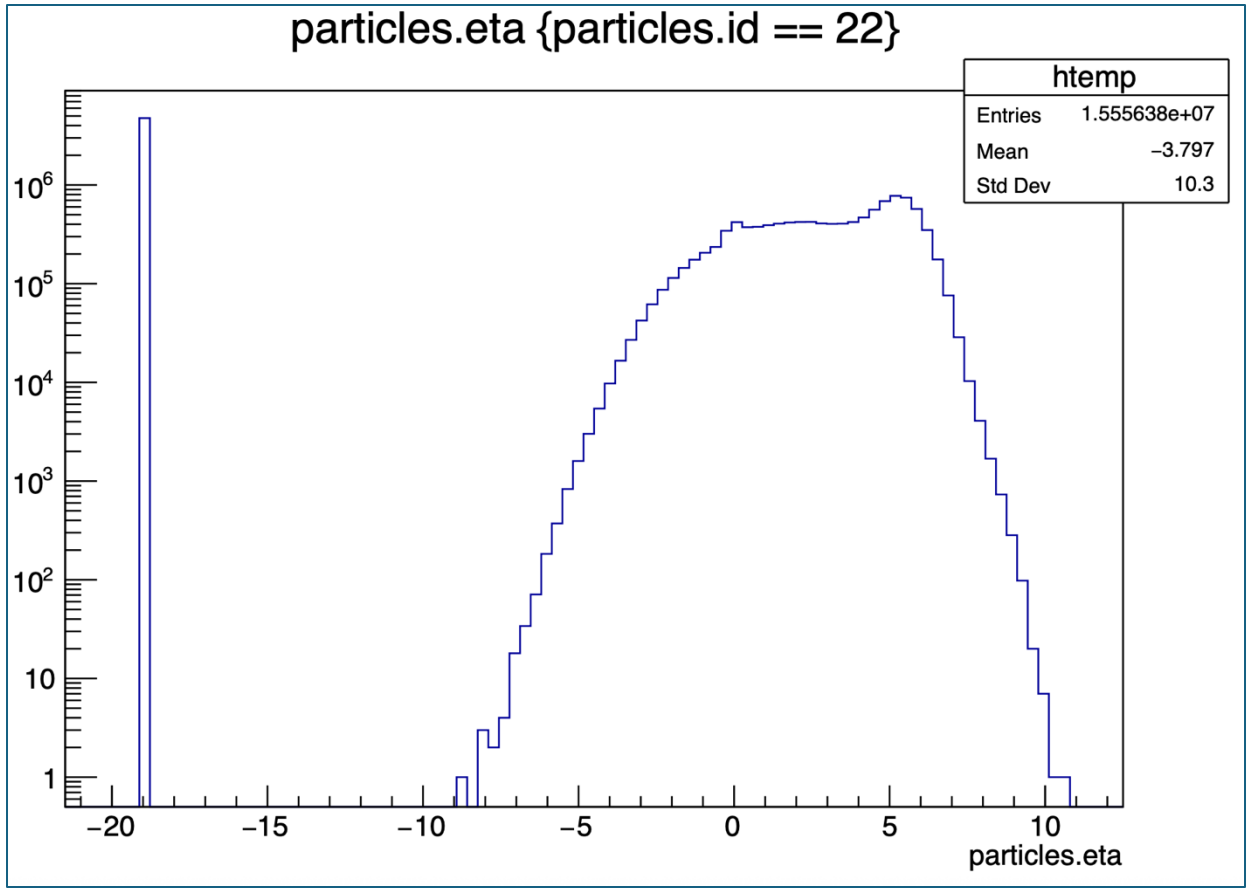
Brief recap

- There were some binning warnings due to infinity in arguments for eta calculation
- Some challenges were encountered when processing higher statistics
  - WriteBuffer errors with ROOT → turns out these happened because of exceeding my quota
- Testing script to properly store ROOT files while avoiding overwriting from the different batch jobs

Total photon yield



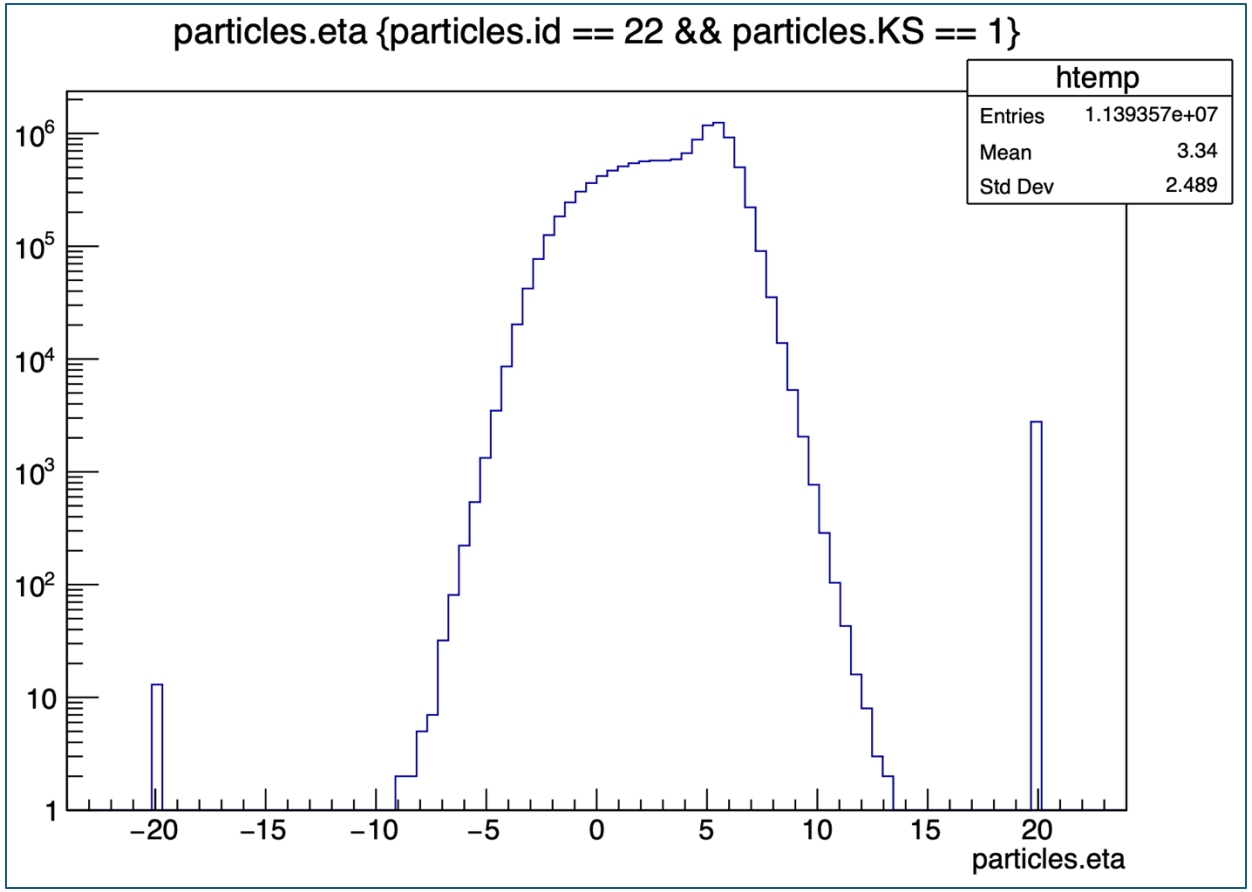
“eta-fixed” eic-smear (v2)



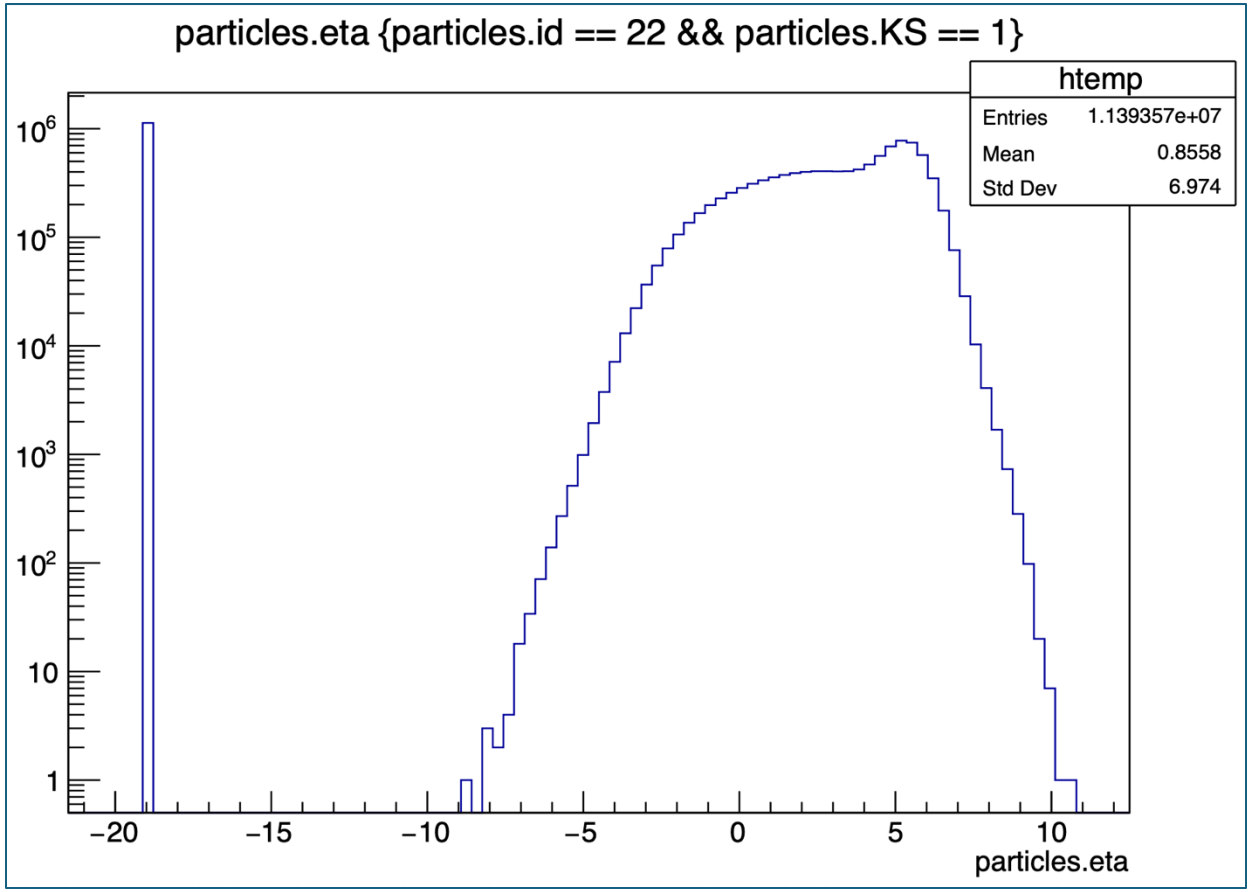
eic-smear (original)

Final state photons

“eta-fixed” eic-smear (v2)

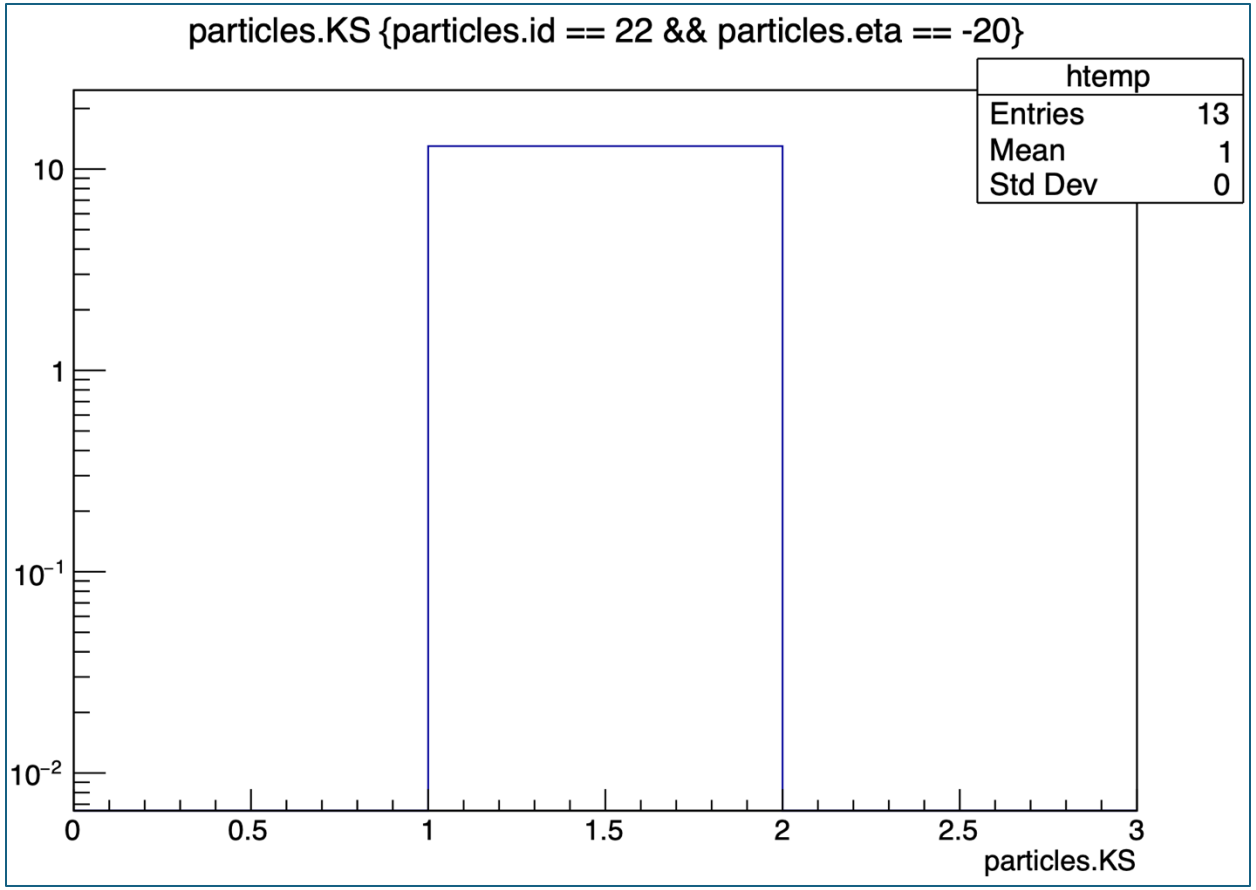


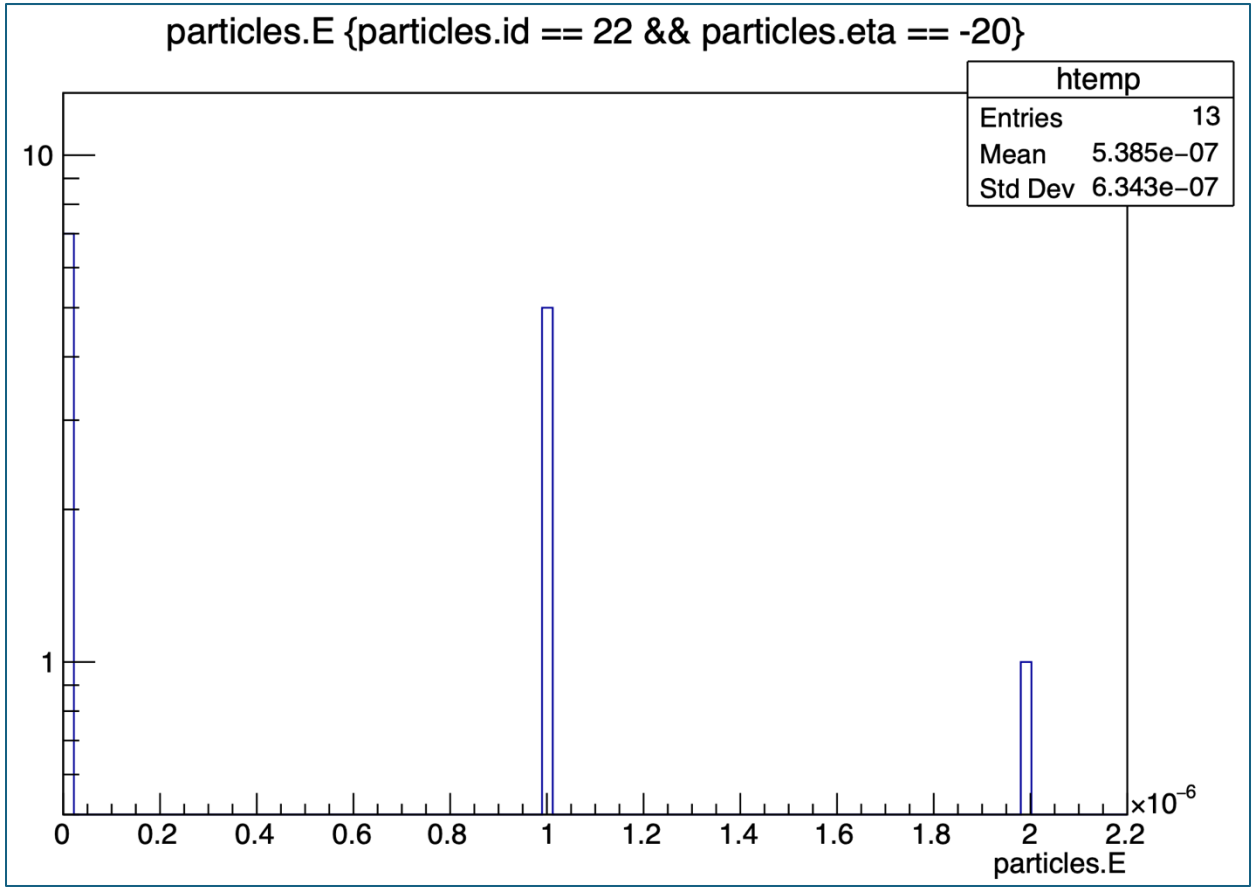
eic-smear (original)



Final state (status = 1) photons are those that detectors would potentially detect

Photons at eta = -20

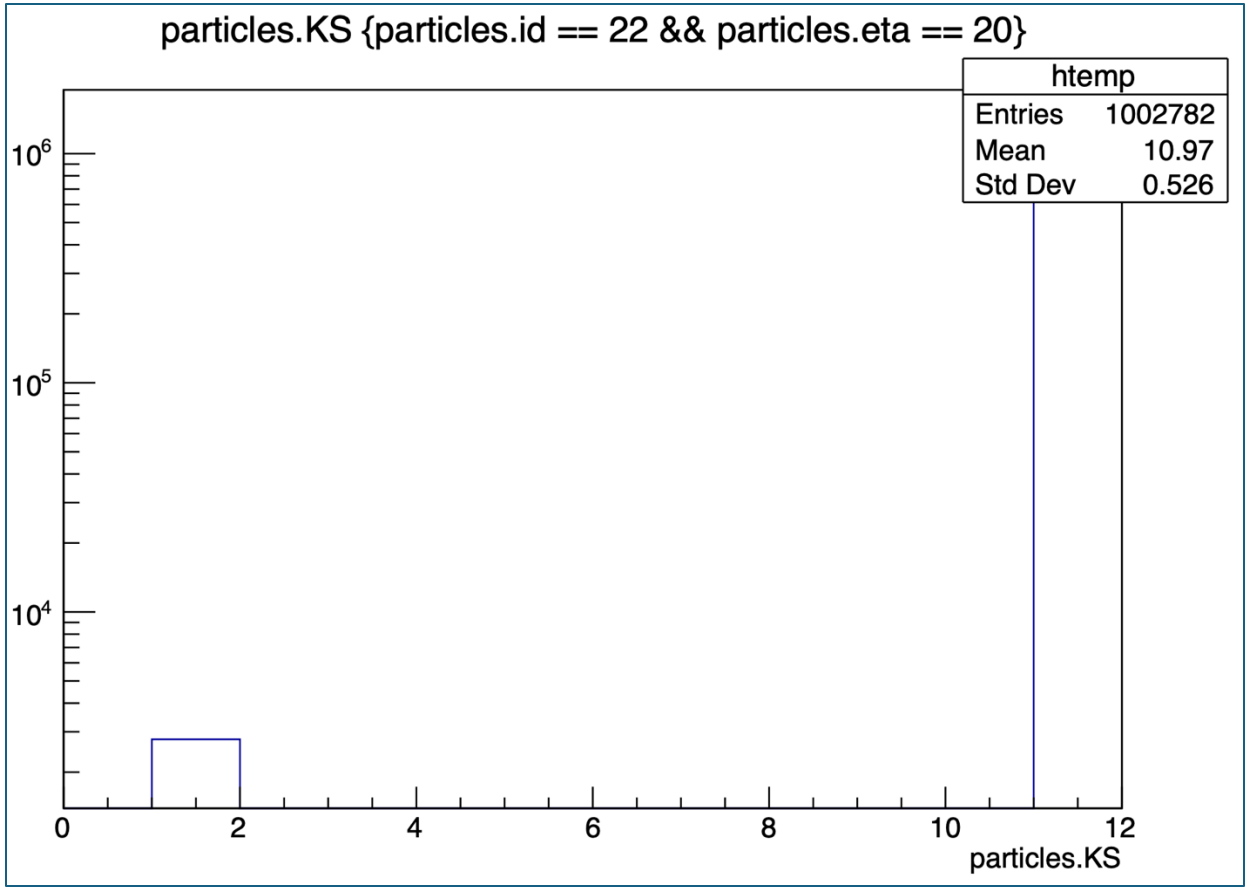




These photons have very small energy (GeV)

Comparing photons at extremes

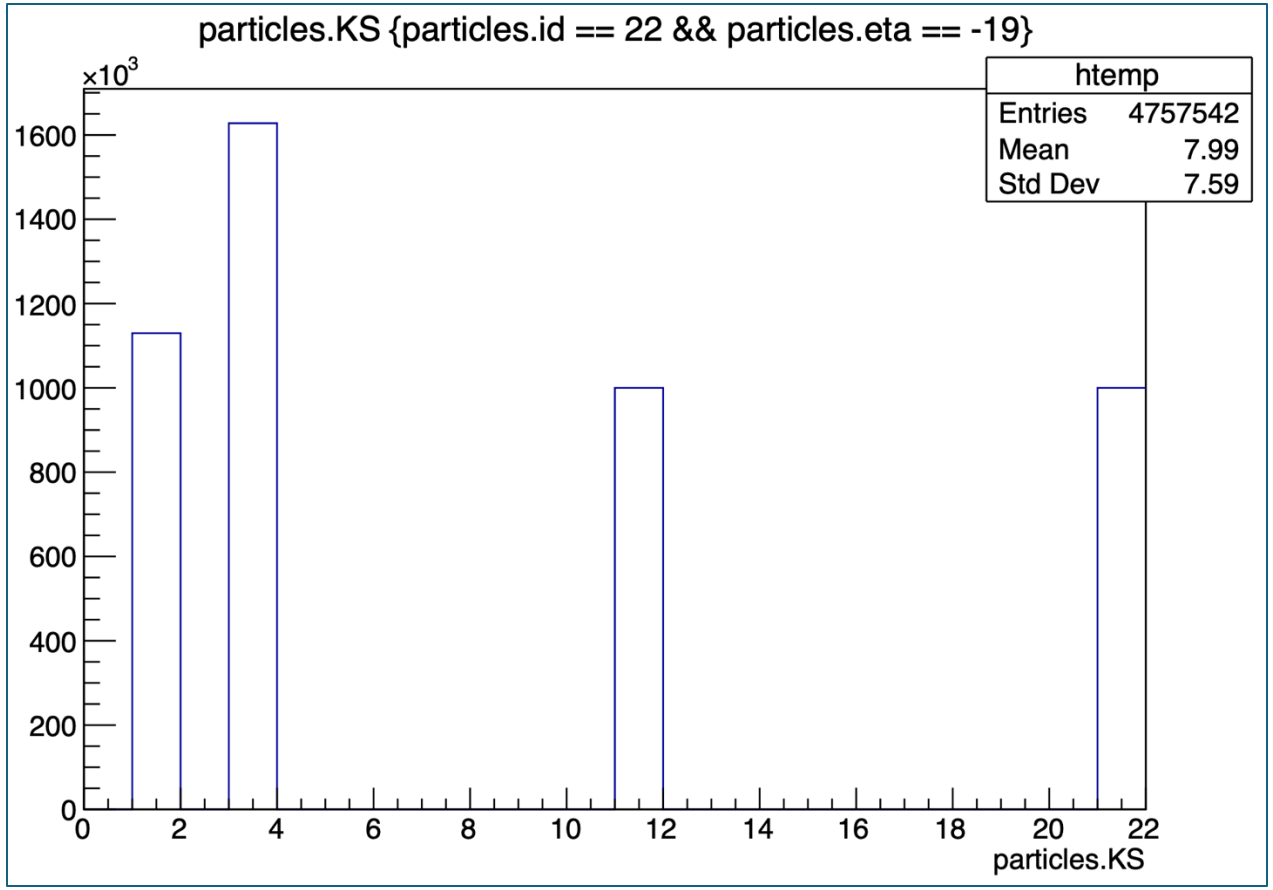
“eta-fixed” eic-smear (v2)



1 : stable final state particle

3 : documentation line describing the collision

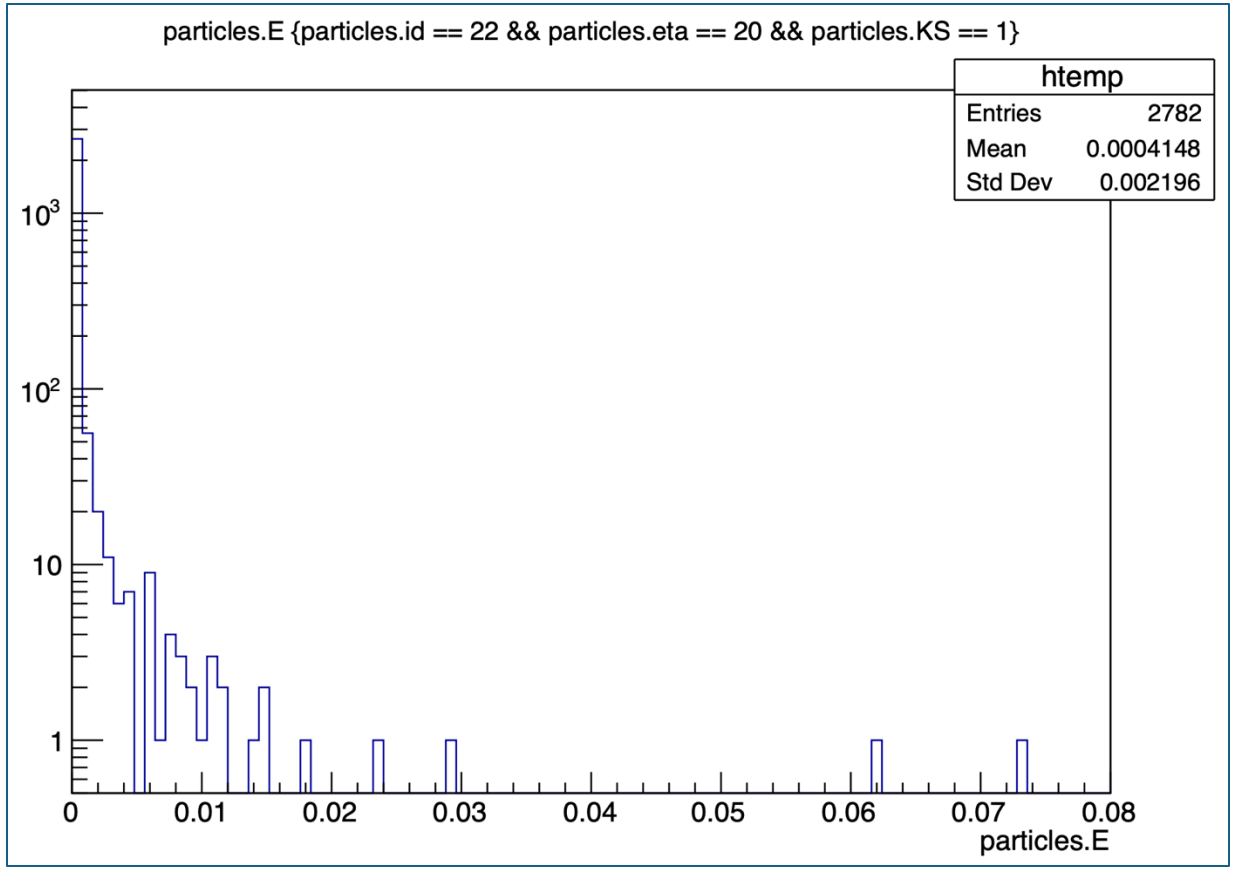
eic-smear (original)



11 : projectile (γ*) documentation line

21 : documentation line describing the collision

The eta = 20 photons



- These photons have much higher energy than η = -20 photons
- These photons account for 0.024% of the status = 1 photons

Summary

- Adjusted pseudorapidity calculation for the eic-smear source code for BeAGLE simulation
  - Reintroduced dummy values

- Changes to eic-smear source code (ParticleMC.cxx file) were added and pushed to github
  - Working on the photon_eta_fix branch

Bonus: Changes that were made to eic-smear code

```
135 void ParticleMCbase::ComputeDerivedQuantities() {
136   // Calculate quantities that depend only on the properties already read.
137   pt = sqrt(pow(px, 2.) + pow(py, 2.));
138   p = sqrt(pow(pt, 2.) + pow(pz, 2.));
139   // Rapidity and pseudorapidity
140   Double_t Epluspz = E + pz;
141   Double_t Eminuspz = E - pz;
142   Double_t Ppluspz = p + pz;
143   Double_t Pminuspz = p - pz;
144
145   // rapidity
146   if (Eminuspz <= 0.0 || Pminuspz == 0.0 ||
147      Ppluspz == 0.0 || Epluspz <= 0.0) {
148     // Dummy values to avoid zero or infinite arguments in calculations
149     // calculating pseudorapidity (eta) with the polarangle
150     // leaving rapidity untouched for now
151     rapidity = -20.;
152
153   } else {
154     rapidity = 0.5 * log(Epluspz / Eminuspz);
155   }  // end if
156   // pseudorapidity
157   theta = atan2(pt, pz);
158   if (theta < std::numeric_limits<float>::epsilon()){
159           // dummy values to avoid infinity or zero in calculations
160           eta = 20.;
161   } else if ((M_PI - theta) < std::numeric_limits<float>::epsilon()){
162           eta = -20.;
163   } else {
164           eta = (-1) * log(tan(theta/2));
165   }
166
167   phi = TVector2::Phi_0_2pi(atan2(py, px));
168 }
```

“eta-fixed” eic-smear (v2)

```
136  void ParticleMCbase::ComputeDerivedQuantities() {
137    // Calculate quantities that depend only on the properties already read.
138    pt = sqrt(pow(px, 2.) + pow(py, 2.));
139    p = sqrt(pow(pt, 2.) + pow(pz, 2.));
140    // Rapidity and pseudorapidity
141    Double_t Epluspz = E + pz;
142    Double_t Eminuspz = E - pz;
143    Double_t Ppluspz = p + pz;
144    Double_t Pminuspz = p - pz;
145    if (Eminuspz <= 0.0 || Pminuspz == 0.0 ||
146       Ppluspz == 0.0 || Epluspz <= 0.0) {
147      // (previous) Dummy values to avoid zero or infinite arguments in calculations
148      // calculating pseudorapidity (eta) with the polarangle
149      // leaving rapidity untouched for now
150      rapidity = -19.;
151
152    } else {
153      rapidity = 0.5 * log(Epluspz / Eminuspz);
154      //eta = 0.5 * log(Ppluspz / Pminuspz);
155
156    }  // if
157    theta = atan2(pt, pz);
158    eta = (-1) * log(tan(theta/2));
159    phi = TVector2::Phi_0_2pi(atan2(py, px));
160  }
```

“eta-fixed” eic-smear (v1)