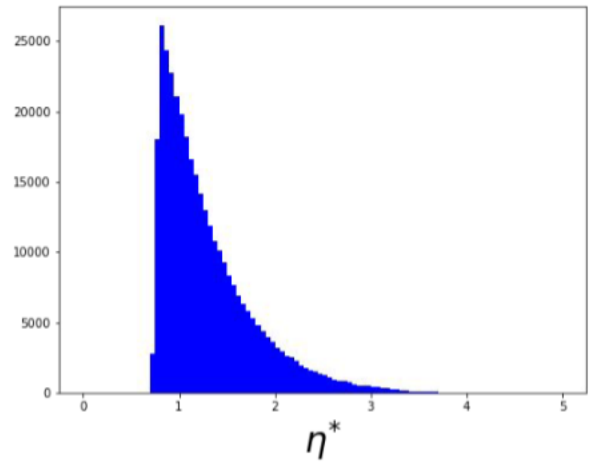
Single particle simulations for Hcal Insert in *BryceCanyon*

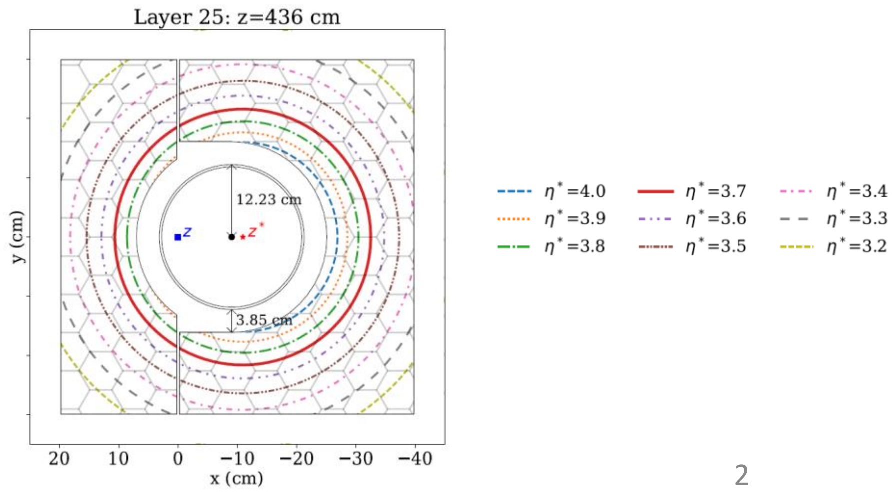
Barak Schmookler

Insert simulations

- ➢Hcal insert and Ecal insert are in *BryceCanyon* configuration.
- ➢Unable to use single particle simulations on S3 because of generated angles are too large.
- ➢In following slides, we'll show Hcal insert results when we generate single particles at correct angles for insert

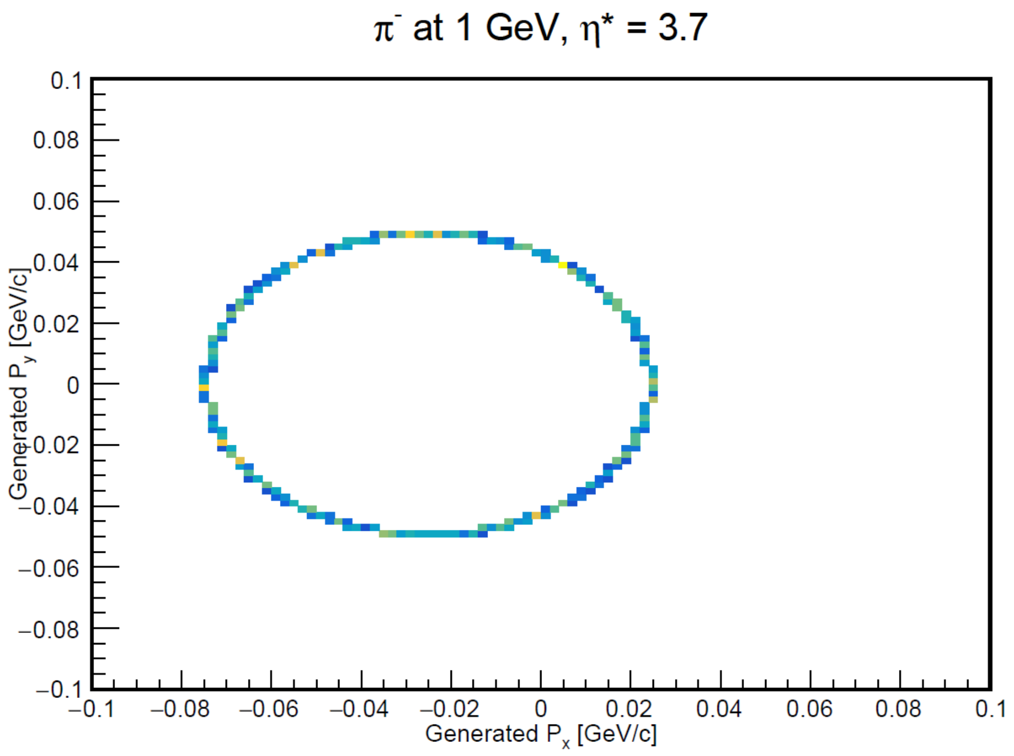
10 GeV π^- Angle 3 - 50 deg EPIC FILES





Particle generation

- We generate single particles at fixed angles w.r.t. the proton beam and put them into a HEPMC3 file.



MCParticles.momentum.y: MCParticles.momentum.x

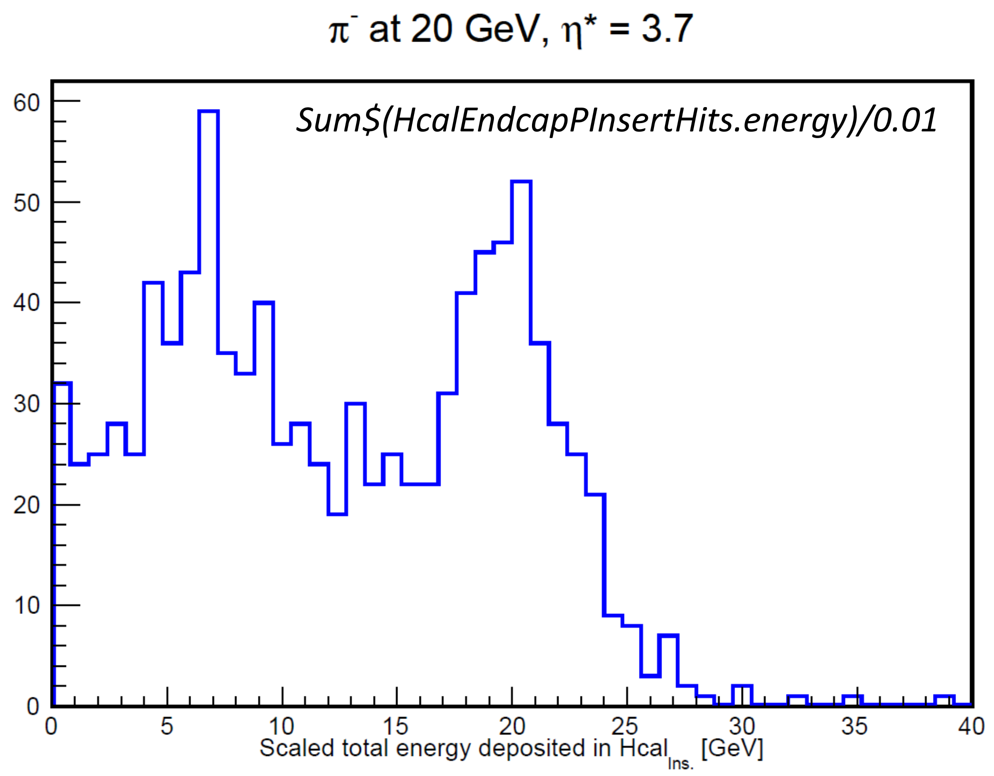
for MCParticles.generatorStatus==1

DD4HEP simulation

- We run the events through the *Brycecanyon* simulation by doing

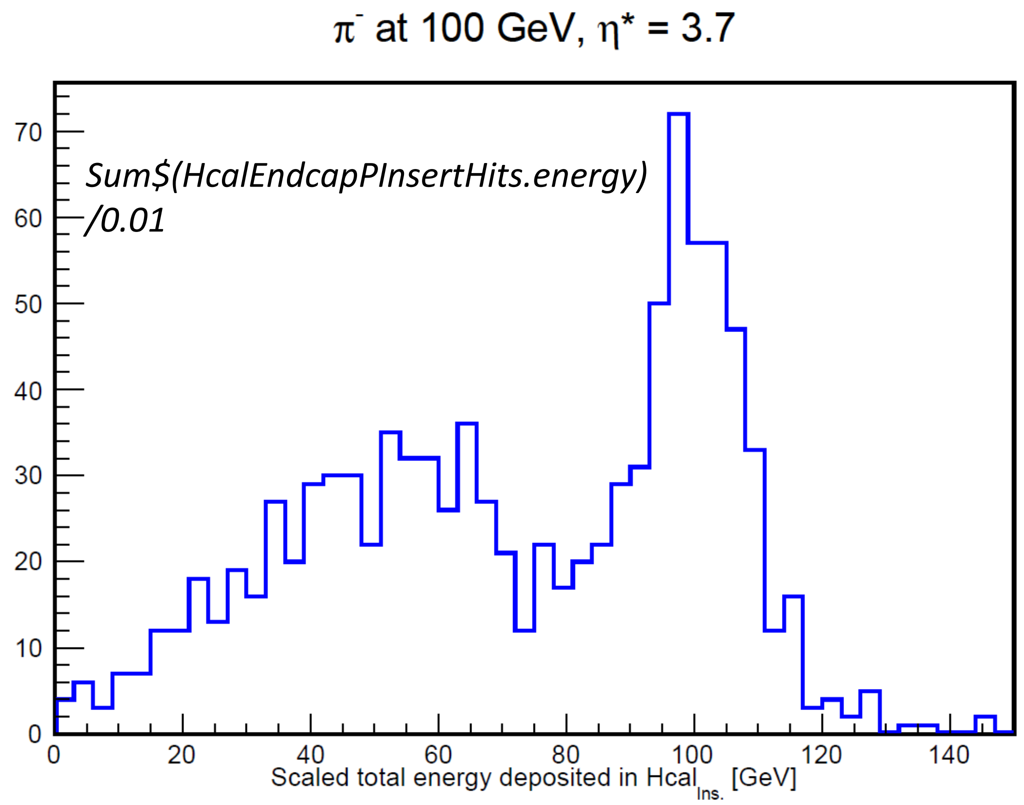
```
ddsim
--compactFile $DETECTOR_PATH/epic_brycecanyon.xml
--numberOfEvents 1000
--inputFiles input/gen_pi-_20GeV_theta_2.83deg.hepmc
--outputFile output.edm4hep.root
```

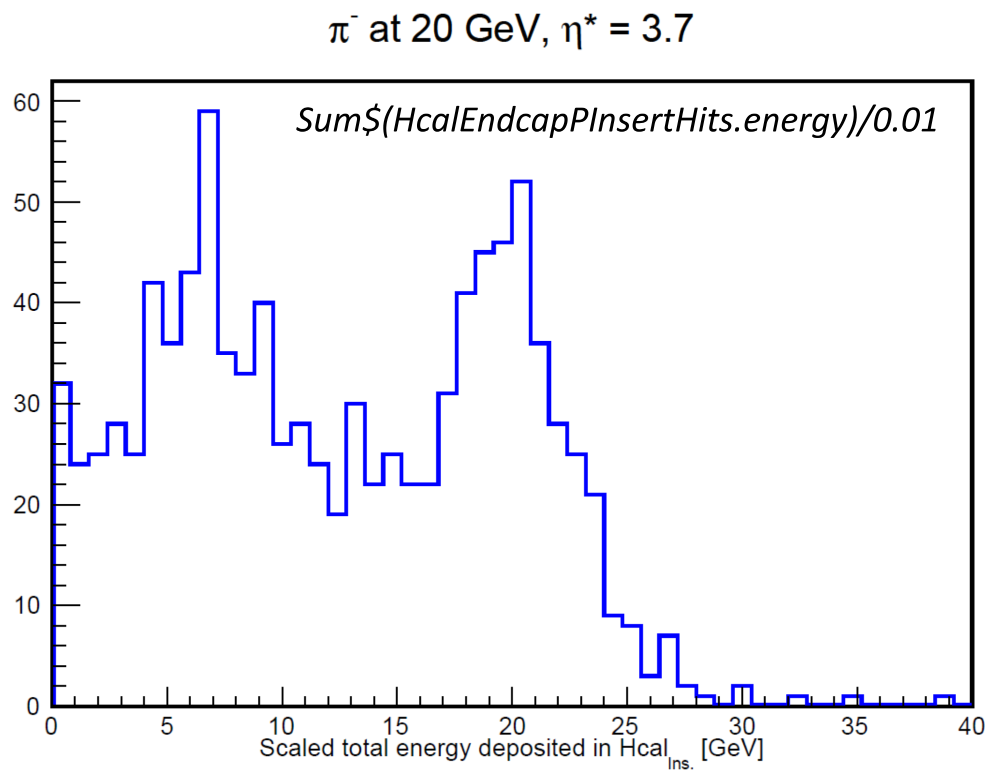
- We can then look at the true (i.e. Geant-level) total energy deposited in the Hcal insert.



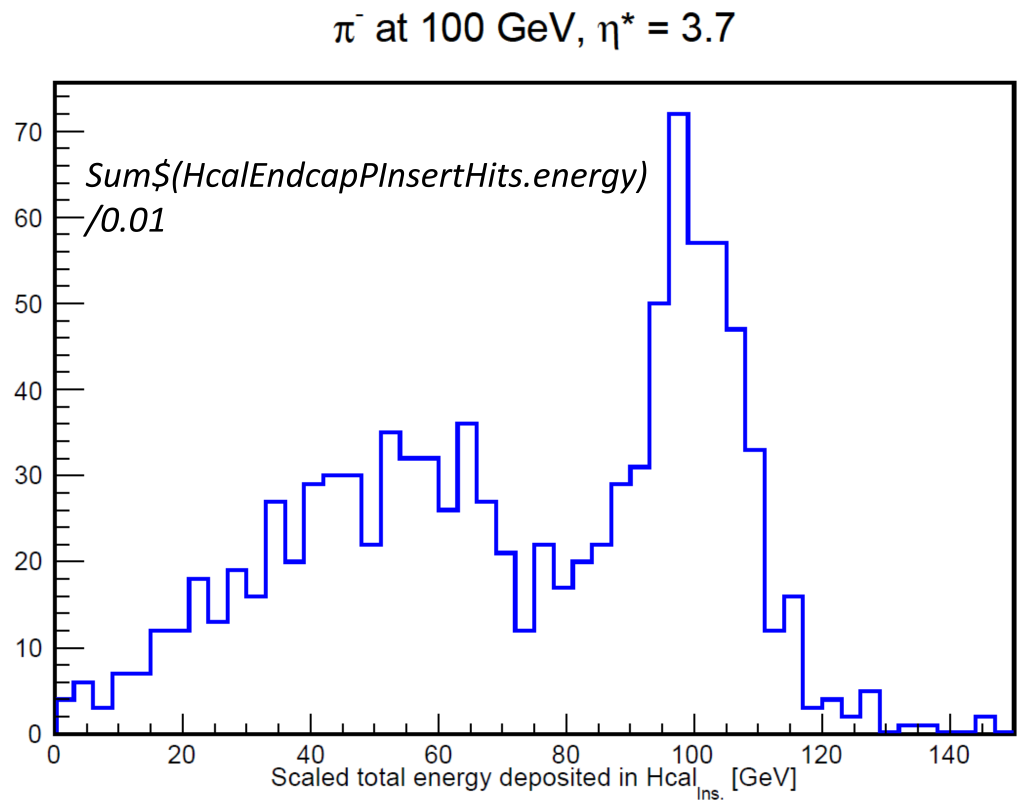
Total energy in Hcal insert scaled up by a factor of 100 – sampling fraction is about 1%.

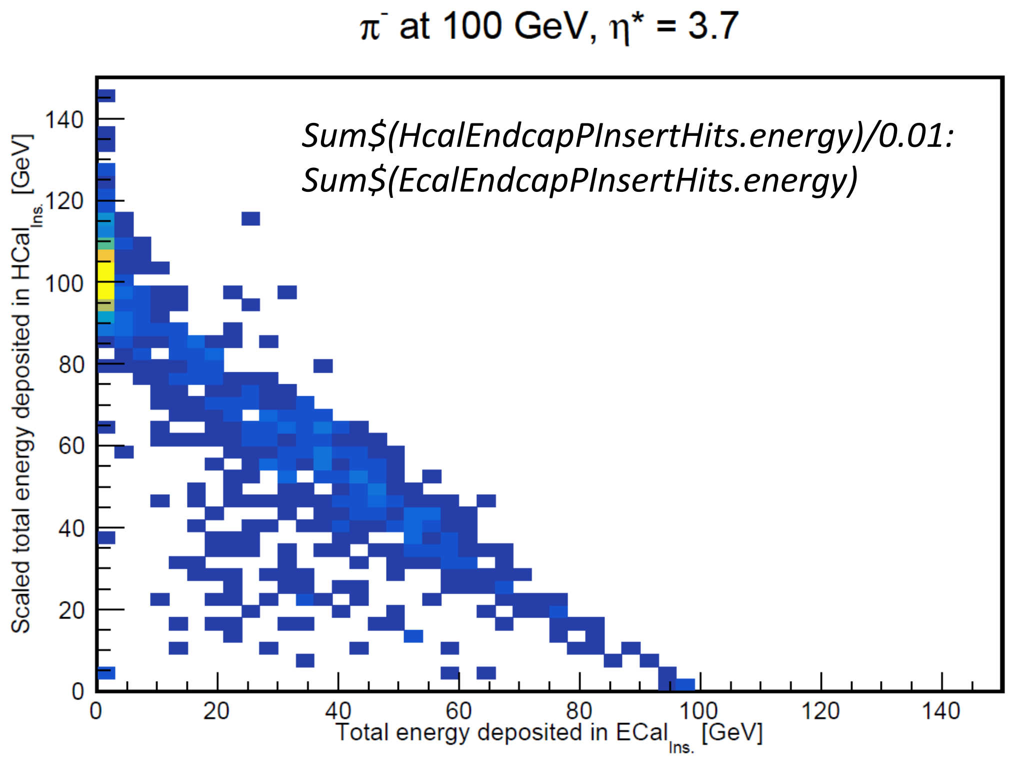
DD4HEP simulation





DD4HEP simulation





Broad peak at lower energy corresponds to showering in the Ecal insert. (Note that the Ecal insert is implemented with a ‘homogeneous’ structure; so, plot shows total true energy prior to applying scale down factor.

EICRecon simulation

- We next put the output from DD4HEP through the EICRecon simulation:

*eicrecon -Ppodio:output_file=eicrecon_out.root *

*-PHCAL:HcalEndcapPInsertRecHits:samplingFraction=0.01 *

*-Pjana:nevents=1000 -Pdd4hep:xml_files=epic_brycecanyon.xml *

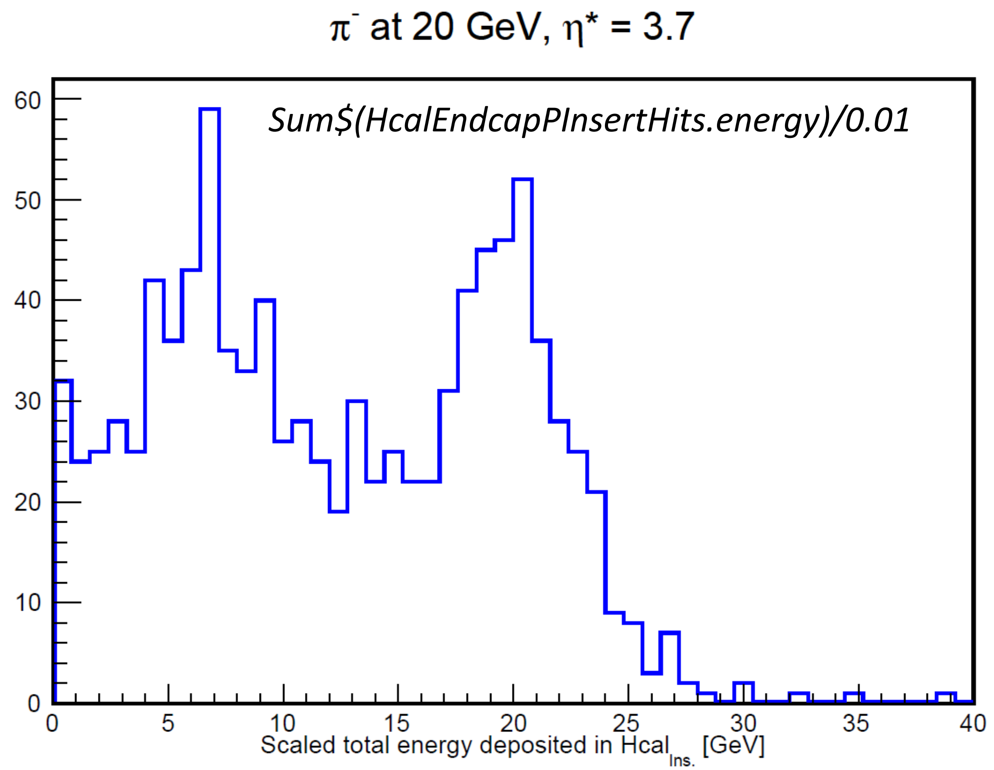
output.edm4hep.root

- In the above command, we tell the reconstruction to assume a 1% sampling fraction.
- For the Hcal insert, the above command gives equivalent results to doing the following (after also setting a 1% sampling fraction in the reco_flags.py file):

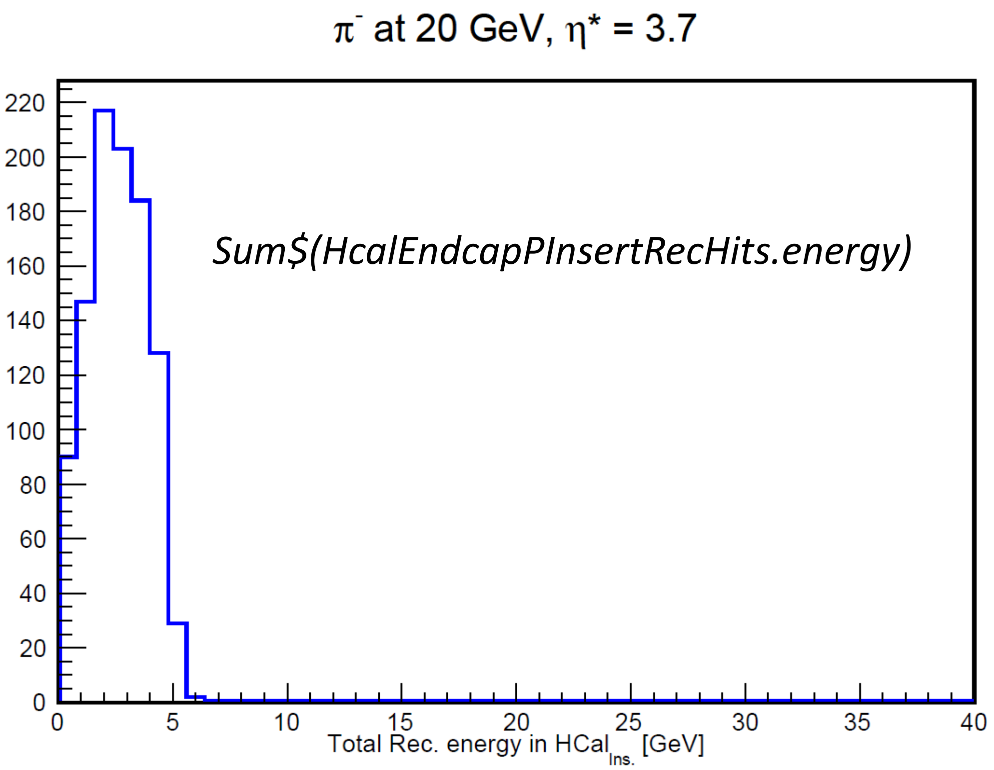
python3 reco_flags.py --nevents 1000 -Pdd4hep:xml_files=epic_brycecanyon.xml output.edm4hep.root eicrecon_out

EICRecon Simulation – total energy reconstruction

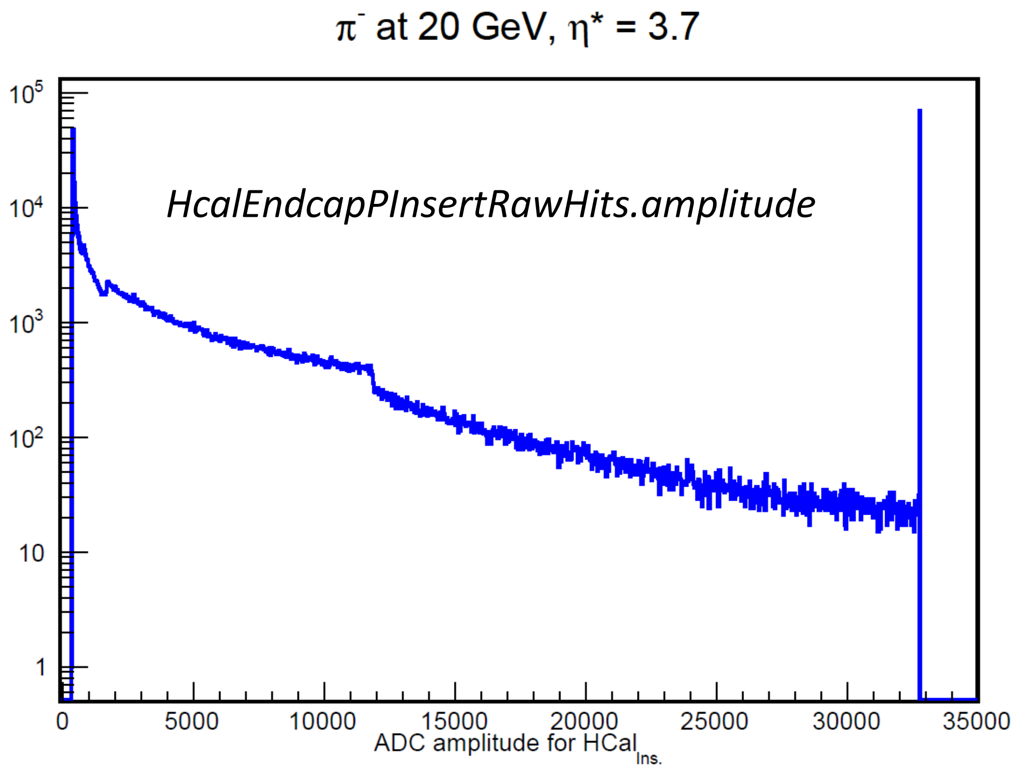
Geant-level scaled output

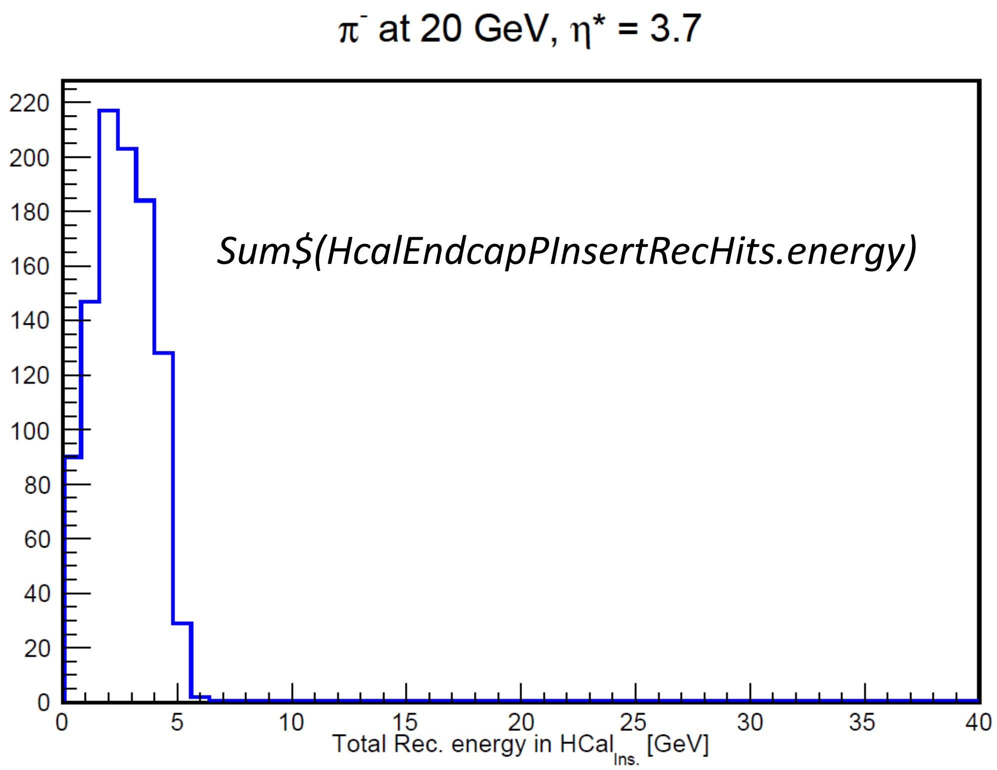


EICRecon output



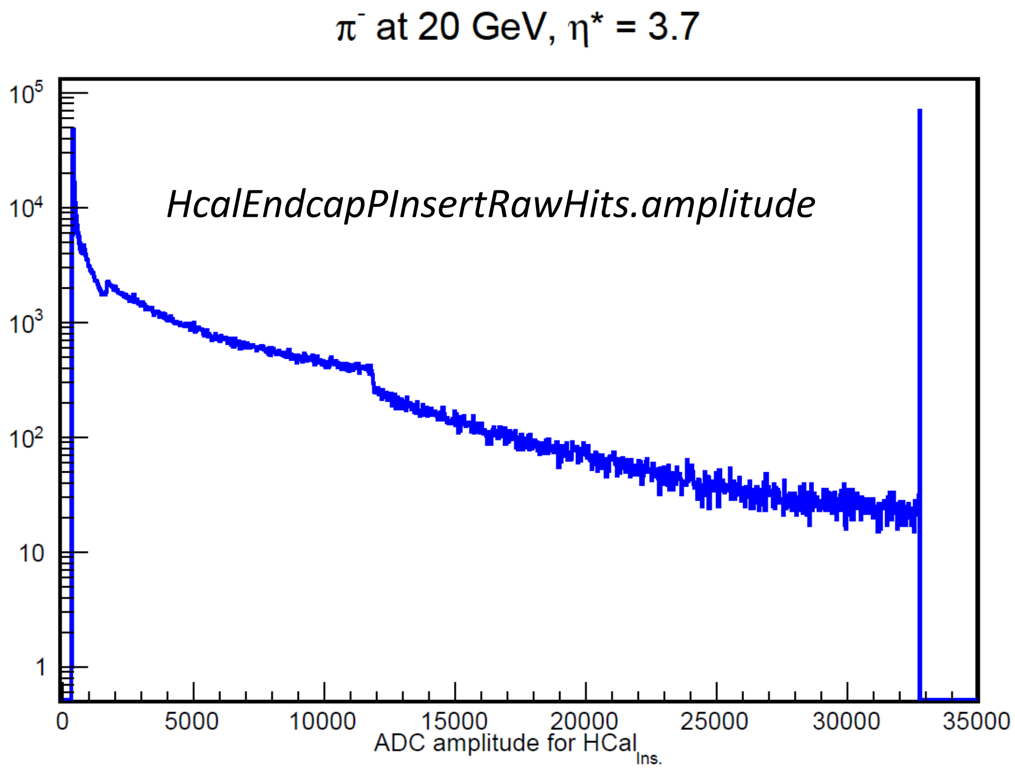
EICRecon Simulation – ADC seems to saturate

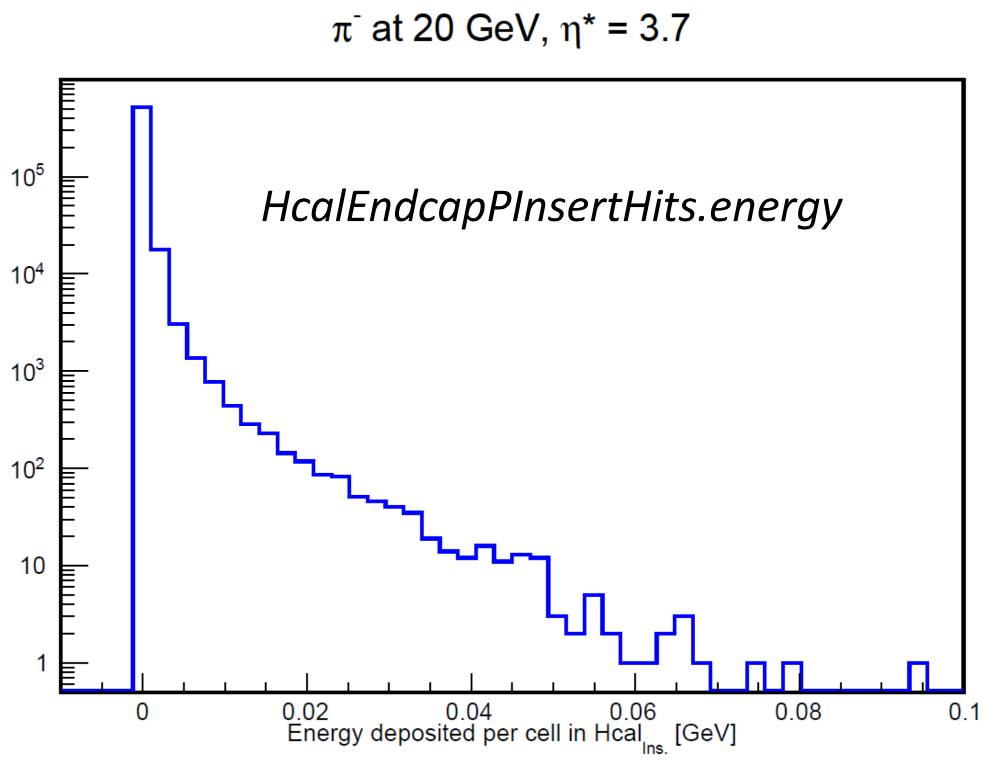




EICRecon Simulation – ADC seems to saturate

ADC capacity (per 3D cell, I think) is set to 200 MeV, so it should not saturate based on Geant-level cell energy deposit





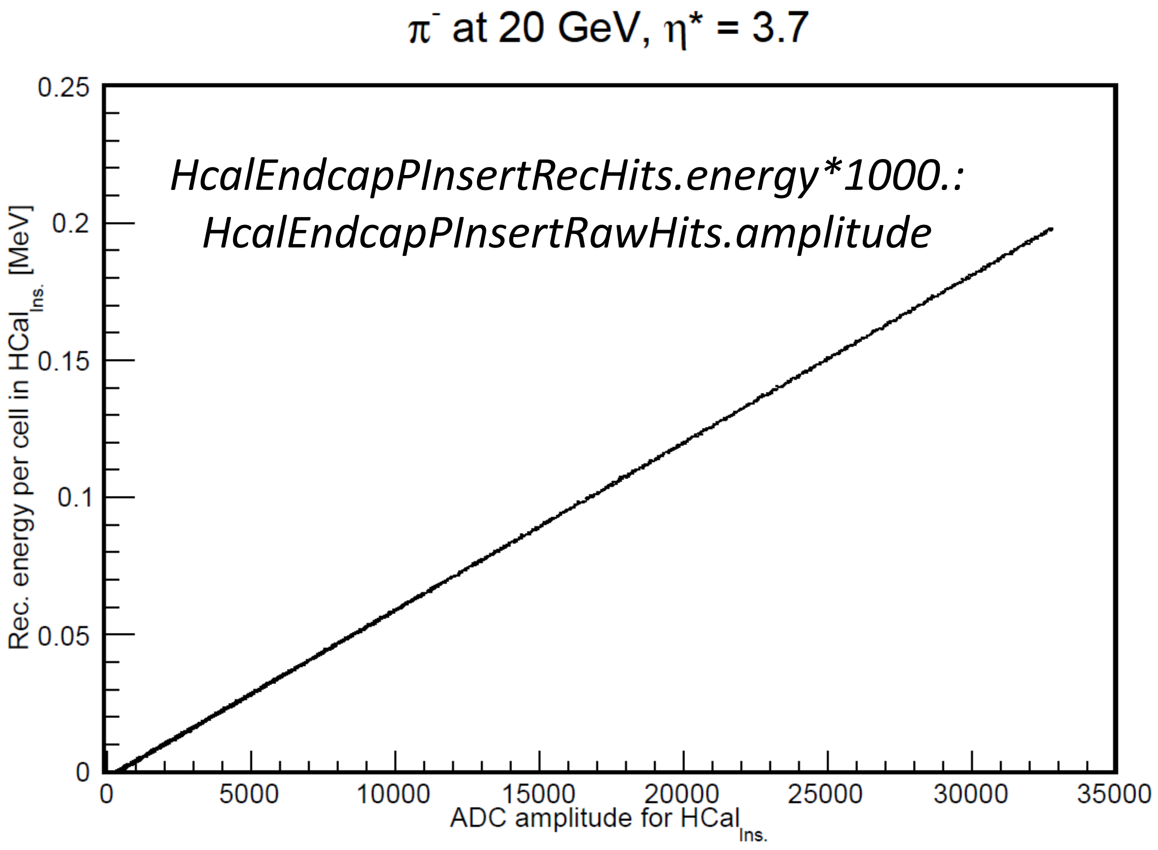
EICRecon Simulation – ADC seems to saturate

- We can test where the saturation occurs by removing setting the sampling fraction to 1 and removing the threshold to reconstruct the energy. This keeps the reconstructed energy and ADC arrays 'in sync'.

-PHCAL:HcalEndcapPInsertRecHits:samplingFraction=1

-PHCAL:HcalEndcapPInsertRecHits:thresholdFactor=0.

-PHCAL:HcalEndcapPInsertRecHits:thresholdValue=-100.



See expected linear dependence. ADC is saturated at 0.2 MeV (not 200 MeV as it should be).

Calorimeter digitization algorithm

Default parameters in
RawCalorimeterHit_factory_HcalEndcapPInsertRawHits.h
(For Hcal insert, same values are in reco_flags.py)

```
// Set default values for all config. parameters in CalorimeterHitDigi algorithm
m_input_tag = "HcalEndcapPInsertHits";
u_eRes = {};
m_tRes = 0.0 * dd4hep::ns;
m_capADC = 32768;
m_dyRangeADC = 200 * dd4hep::MeV;
m_pedMeanADC = 400;
m_pedSigmaADC = 10;
m_resolutionTDC = 10 * dd4hep::picosecond;
m_corrMeanScale = 1.0;
u_fields={};
u_refs={};
m_geoSvcName = "ActsGeometryProvider";
m_readout = "";
m_geoSvc = app->GetService<JDD4hep_service>(); // TODO: implement named geometry service?
```

Can recent changes to digitization algorithm explain this?

fix: also avoid MeV conversion in CalorimeterHitDigi

main (#365)

wdconinc authored and **c-dilks** committed last week   1 parent 353c8dd

Showing **1 changed file** with **3 additions** and **4 deletions**.

7 src/algorithms/calorimetry/CalorimeterHitDigi.cc

```diff
@@ -60,8 +60,7 @@ void CalorimeterHitDigi::AlgorithmInit(std::shared_ptr<spdlog::logger>& logger)
         eRes[i] = u_eRes[i];
     }

-    // using juggler internal units (GeV, dd4hep::mm, dd4hep::radian, dd4hep::ns)
-    dyRangeADC = m_dyRangeADC * dd4hep::MeV; // value of m_dyRangeADC is in dd4hep::MeV
+    // using juggler internal units (GeV, mm, radian, ns)
     tRes       = m_tRes / dd4hep::ns;
     stepTDC    = dd4hep::ns / m_resolutionTDC;

@@ -154,7 +153,7 @@ void CalorimeterHitDigi::single_hits_digi(){
//                              : 0;

        const double ped    = m_pedMeanADC + m_normDist(generator) * m_pedSigmaADC;
-        const long long adc = std::llround(ped + eDep * (m_corrMeanScale + eResRel) / dyRangeADC * m_capADC);
+        const long long adc = std::llround(ped + eDep * (m_corrMeanScale + eResRel) / m_dyRangeADC * m_capADC);
```

Summary

- For the insert detectors, we need to generate events at large eta values. As these angles were not included in the files on S3, we performed the simulation ourselves.
- We studied the Hcal insert using negative pions generated at the correct angle for the insert. The DD4HEP (Geant-level) output looks as expected.
- The digitization in EICRecon seems to saturate the ADC for the Hcal insert, which should not happen based on the set ADC capacity. This needs to be understood before resolution studies can be performed.