Status of ECAL Studies

Vivek Jain, SUNY Albany

June 10 », 2023

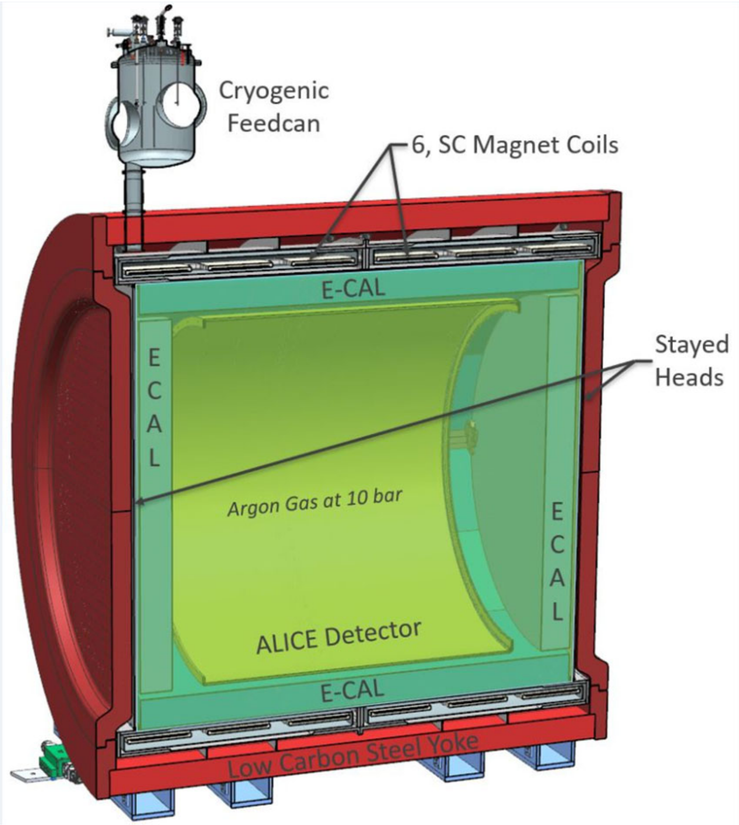
Many thanks to Leo for code fixes, Tom/Leo for software help, Eldwan Brianne for implementing the geometry

Outline

- A status report of ECAL studies:
  - Bugs found and fixed
  - What next?

Latest Geometry

- New ECAL geometry –
    - 42 layers of Pb-Scintillator sandwich (~ 10.5 X_0) – tiles & strips
        - Barrel:
            - Scintillator: 8 tile layers - each 0.5 cm & 34 strip layers - each 1 cm
            - Pb – 8 layers, each 0.7 mm thick, 34 layers, each 1.4 mm thick
        - Endcaps are 6 + 36 layers
    - Barrel has 12-fold symmetry
    - Newly optimized SPY magnet and cryostat as the pressure vessel
        - No extra material between the ECAL and TPC



TPCRadius = 273 cm, TPCLength = 259

TPCFidRadius = 222.5, cm TPCFidLength = 215

ECALInnerRadius = 278, ECALOuterRadius = 334 cm

ECALStartX = 328 , ECALEndX = 375 cm

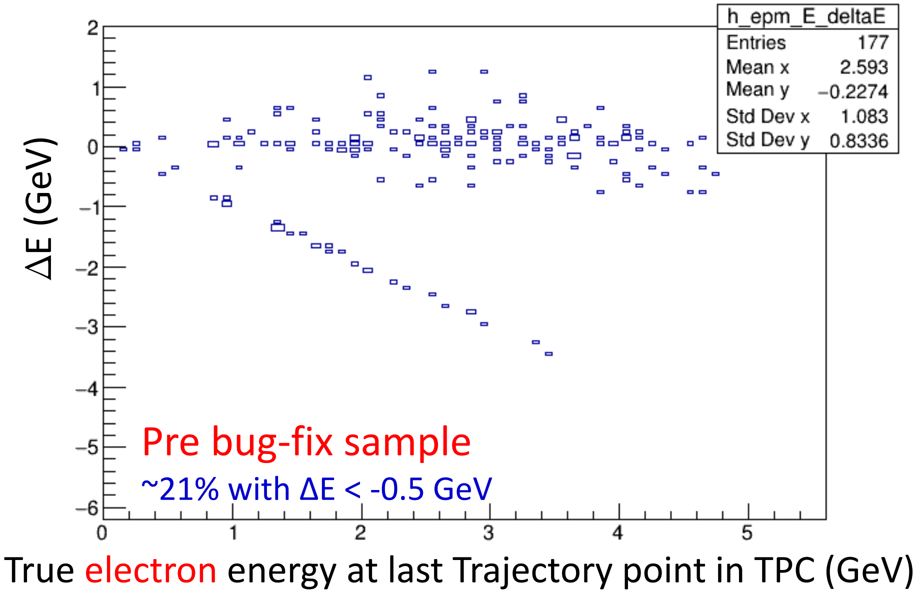
Samples - All use the latest ECAL geometry - *nd_hall_mpd_only_ECal12sides_42l_SPY_v3_wMuID.gdml*

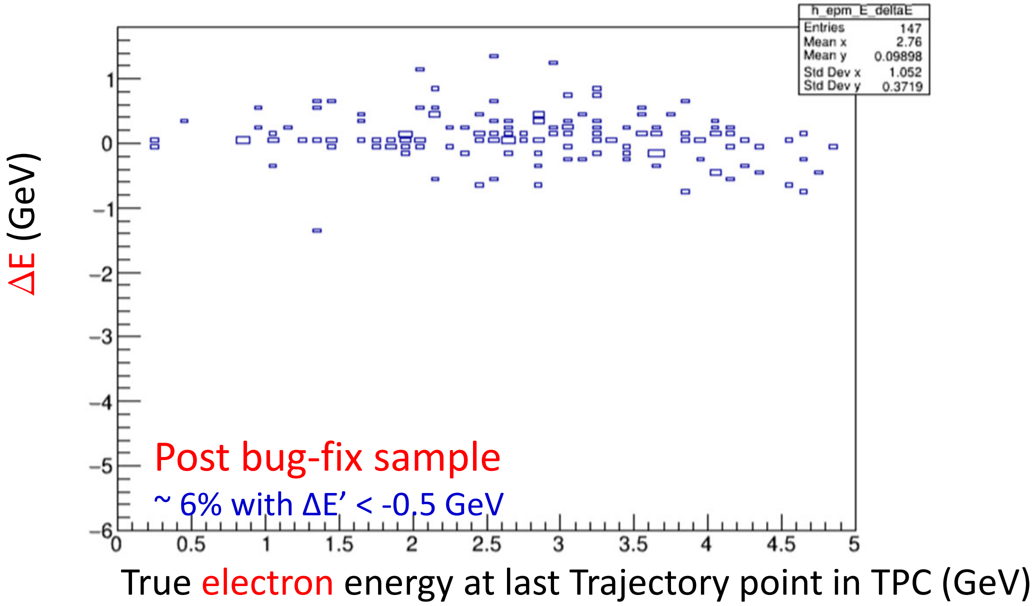
- Reconstruction uses the default SiPM hit finding algorithm – well-understood
  - Neutrino events - locally generated sample (GENIE v3.0.6 and no edep-sim)
    - **Neutrino vertex is in TPC fiducial volume**
  - Single electrons – locally generated (GENIE v3.0.6 and no edep-sim)
    - Start the particles within (± 10 cm) of middle of the TPC, in a restricted kinematic range:
      - P ~ 3 ± 1 GeV, θ_{XZ} ~ $(0 \pm 3)^0$, θ_{YZ} ~ $(-5 \pm 1)^0$ - these values chosen somewhat arbitrarily
      - Also made samples where single electrons are shot at the endcap, and the corner of the endcap & downstream barrel

A recap of bugs (till April’ 23)*

- Working in tandem with Leo
  - ***Neutrino sample:*** I had noticed that ~9% of hits in the endcap were being lost during digitization. Leo found that the speed of light in scintillator was not set correctly in the code, and since we use photon arrival times during reconstruction it had an impact. This bug-fix (Fall 2022) reduced failures to < 1% - *so there is still a residual effect, which needs to be tracked down*
    - *The original 9% bug was due to issues with transformations between World and Local coordinate systems (the speed of light being wrong messed things up).*
  - ***Single Electrons:*** Comparing the sum of energies of clusters back-tracked to the true single electron energy (ΔE), I saw that there were about 20% of cases where ΔE < -0.5 GeV, implying that some real clusters were not being back-tracked correctly
    - If, instead, I summed up the energies of ALL clusters in the event, then ΔE was nicely peaked around 0
    - It turned out to be a “feature” of the reconstruction code

- *More details in ECAL_Status_Feb_April_2023.pptx





In these plots, the Y-axis: Σ(all clusters back-tracked to the electron) – Energy of electron at last point in TPC.

1) The diagonal band has disappeared in the post bug-fix sample implying that the correct clusters are being back-tracked to the electron (so back-tracker appears to be working correctly – thanks, Leo!!)

2) However, the number of entries has dropped from 177* to 147. I think this is OK, since I am requiring that back-tracker finds at least one cluster matching the electron to make this plot. What is happening is that in other cases, clusters are matching brems from the electron (or daughters of brems from electrons), and once we include them, we recover all entities that actually make it to the CALO (*now this seems to be 180?)

More bug finding/fixing (May'23 onward)

- Was looking at information stored in the anatree for clusters, and noticed that one variable, ClusterTimeDiffFirstLast did not seem correct
  - Supposed to be = (earliest time in the first hit layer in a cluster – earliest time in the last hit layer in a cluster) – *not sure what this tells us, but that's a separate issue*
  - In the process I discovered three bugs:
    - One had already been noticed by Leo – The starting X position of the Endcap ECAL was not correct – traced to GeometryCore.cxx – wrongly assumed that the endcap ECAL started just after the TPC ended.
      - For now, ***I have hacked the code,*** and subtract thickness of the endcap ECAL (44.9 cm) from the outer X of the endcap ECAL (which is correct in the code) – this gives me a value for the starting X that agrees with what I see in plots **<u>(fixed 6/6/23) –</u> again a Geometry issue**
    - The calculation of ClusterTimeDiffFirstLast is being done incorrectly - This led to an investigation of what exactly is being calculated and stored (found and fixed two more bugs)
      - I discovered problems with the variable DigiHitTime – for tiles, *½ the actual value was stored in the anatree*
      - Some recoHit times in a cluster were way off compared to others – turns out that sometimes recoHitX gets a value of ± 0.00499, and then the stripLength gets messed up and subsequently the recoHit time – **probably Geometry issue**

What do Sim/Digi/Reco/Cluster times mean?

- Sim: time of energy deposit – Float – I am guessing 0 starts with the interaction since times are generally greater than 10 ns (for particles starting at the center of the TPC) – hit in Layer 1 has the smallest value
  - *In the anatree, SimHitTime is this value*

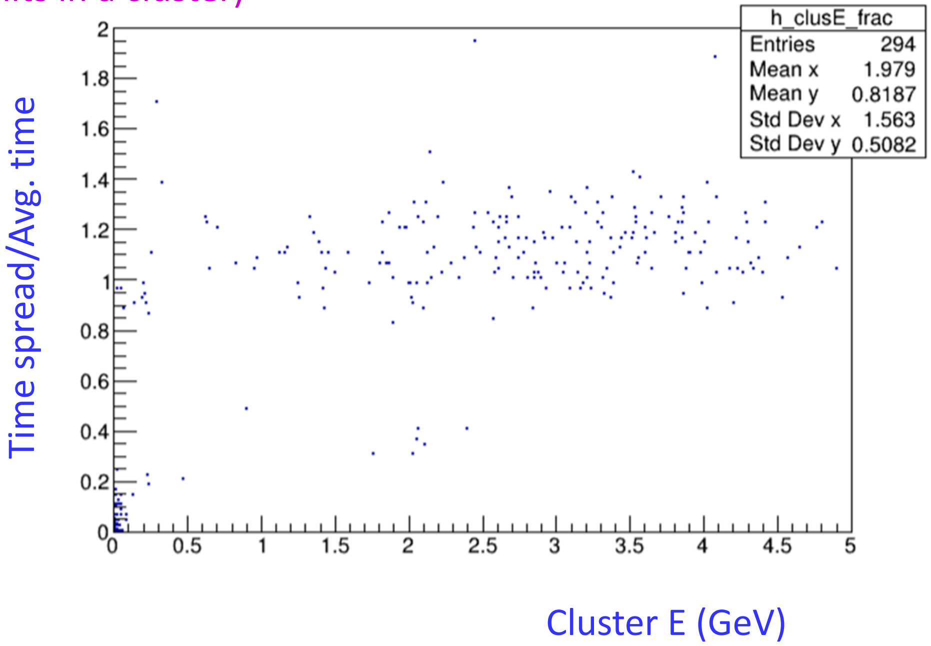
- Digi: Pair of floats - (code is in ECALReadoutSimulation.cxx)
  - For Tile, first is SimHitTime smeared, whereas second is 0
  - For Strips, doLightPropagation returns two values
  - *In anatree*, store (Value_1 + Value_2)/2., which is clearly wrong for Tiles!
    - anatree_module.cc has been fixed on 6/12/23 to store the 1st value in case of tiles

What do Sim/Digi/Reco/Cluster times mean?

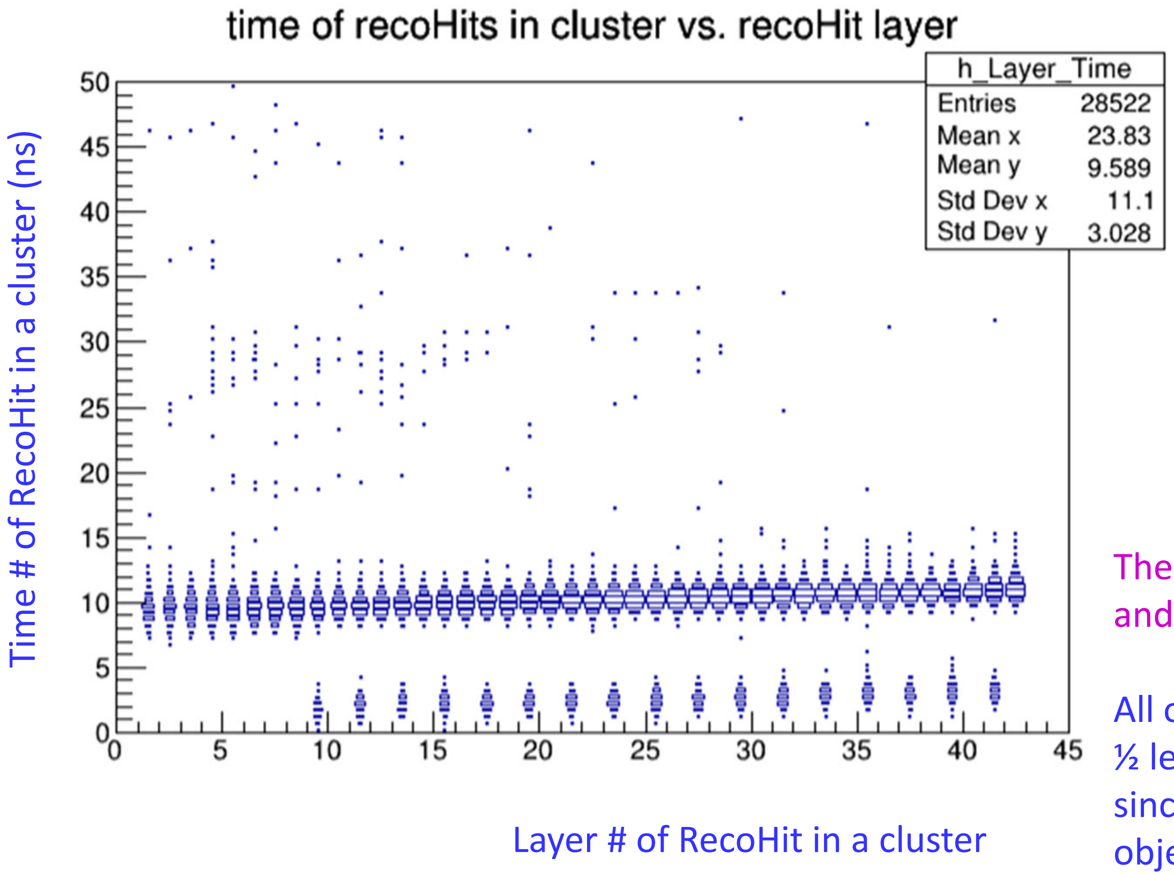
- Reco: Pair of Floats - (code is in SiPMHitFinder_module.cc)
  - Tile: RecoHitTime is the same as DigiTime
  - Strip: Calculation = *(average time at two ends – propagation time along half-strip length)*
  - *In anatree*, we store Time().first, which is correct for both Tile & Strips

- Cluster: Two floats are calculated – (code is in NNCluster.h)
  - Average Cluster time – average time of all reco hits in cluster
  - ClusterTimeDiffFirstLast – logic is convoluted, and generally gives the wrong value
    - Not clear what this can be used for, and so I haven’t fixed it…

Calculate time spread to be difference between
min. and max. times of RecoHits associated with a cluster
(here I am just using the largest and smallest times of
hits in a cluster)

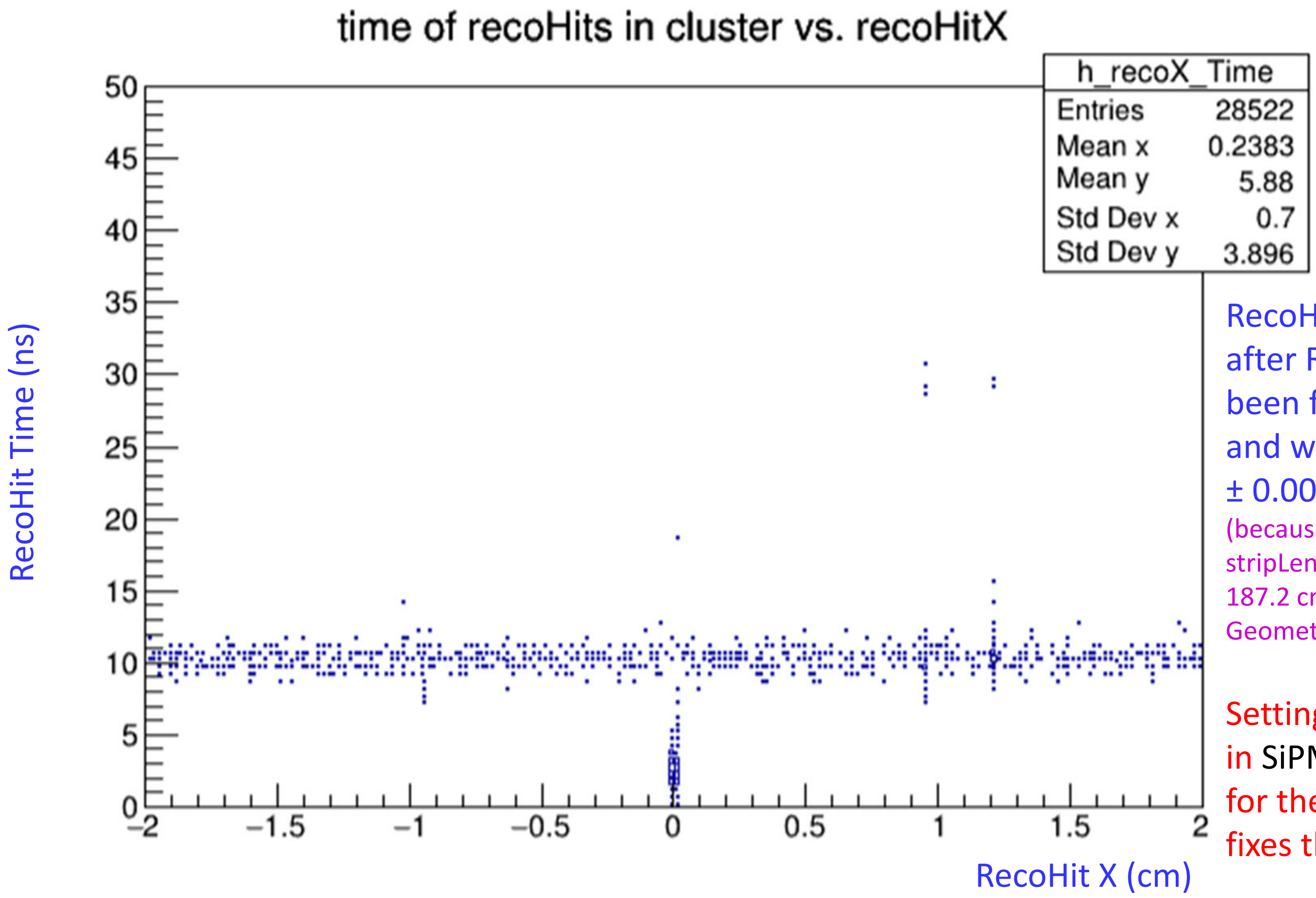


Spread looks very large



The band at ~ 2ns is about 10% of the total, and all of them are in (alternate) strips

All of these values are in strips with ½ length = 333.6 cm? That’s clearly wrong, since a strip along the barrel is not one long object, but is thought of as having two halves, so, in such strips ½ length should be ~ 187 cm.

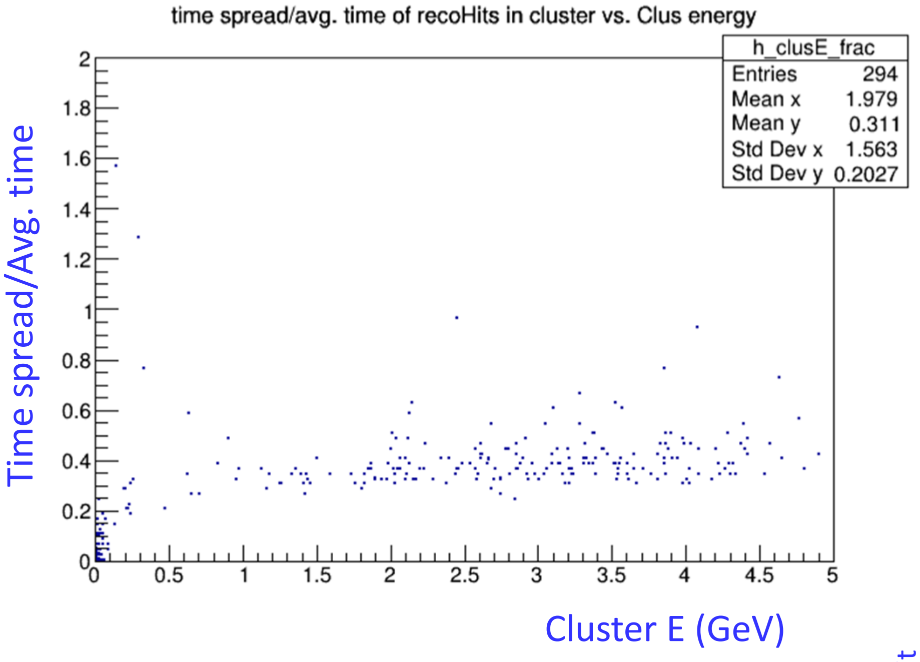


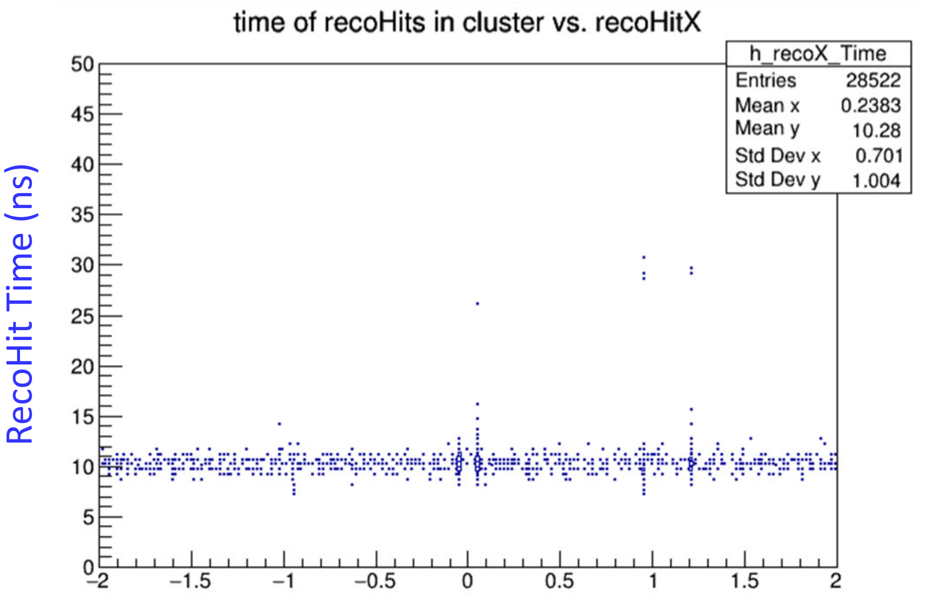
RecoHit Time is calculated after RecoHit position has been found (using DigiHit times), and when RecoHitX comes back as ± 0.004992, times get screwed up (because the code incorrectly gets the stripLength/2 to be 333.6 cm instead of 187.2 cm, from somewhere in the Geometry code)

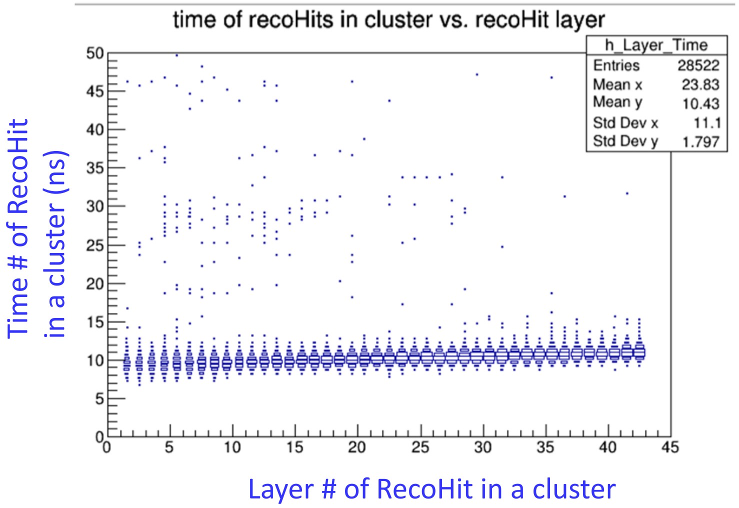
Setting RecoHitX = ± 0.05 in SiPMHitFinder_module.cc for these pathological cases fixes the problem (6/15/23)

I only stumbled upon this because I was shooting single electrons from the center of TPC towards the downstream barrel

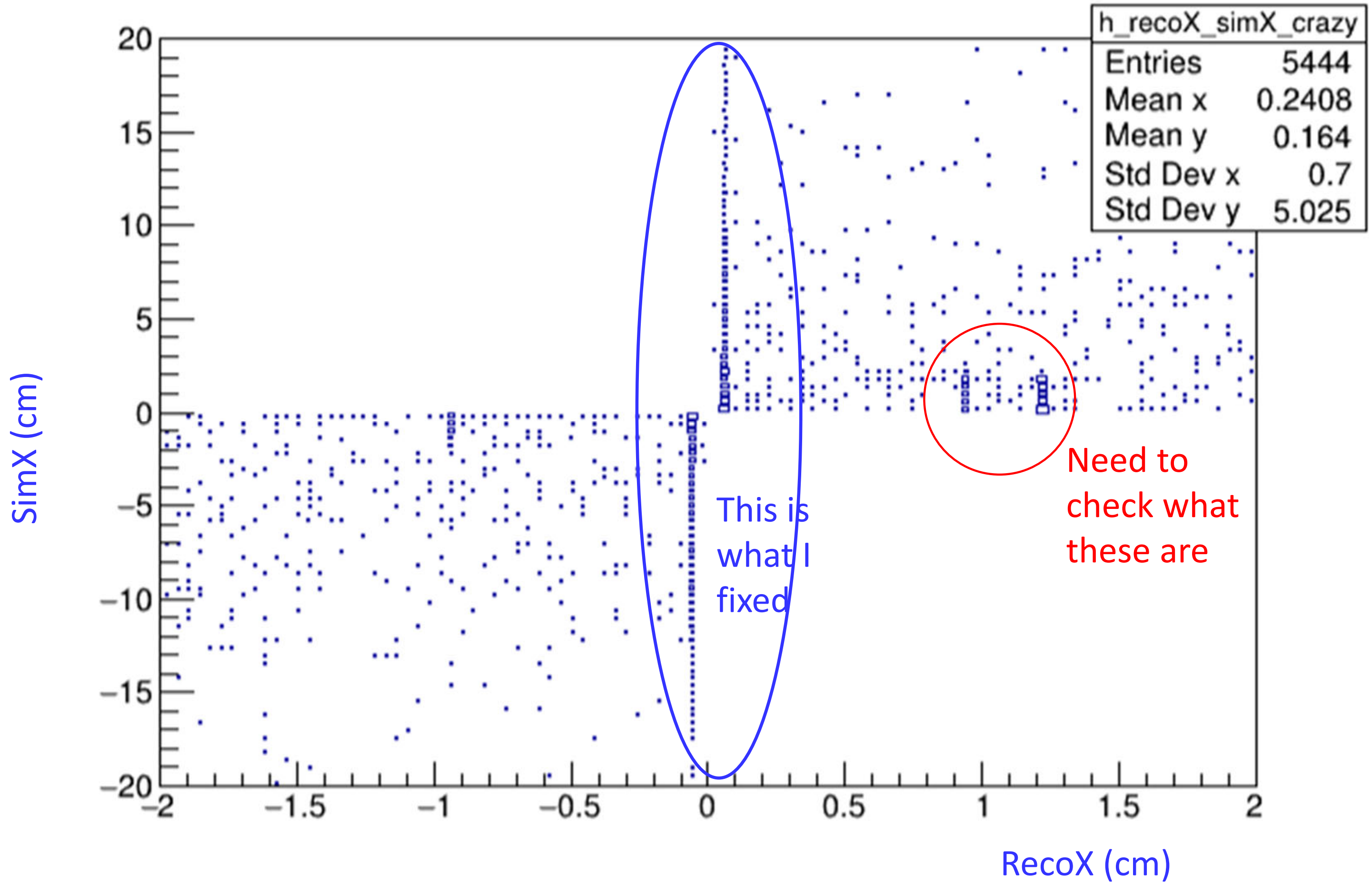
After RecoX bug-fix





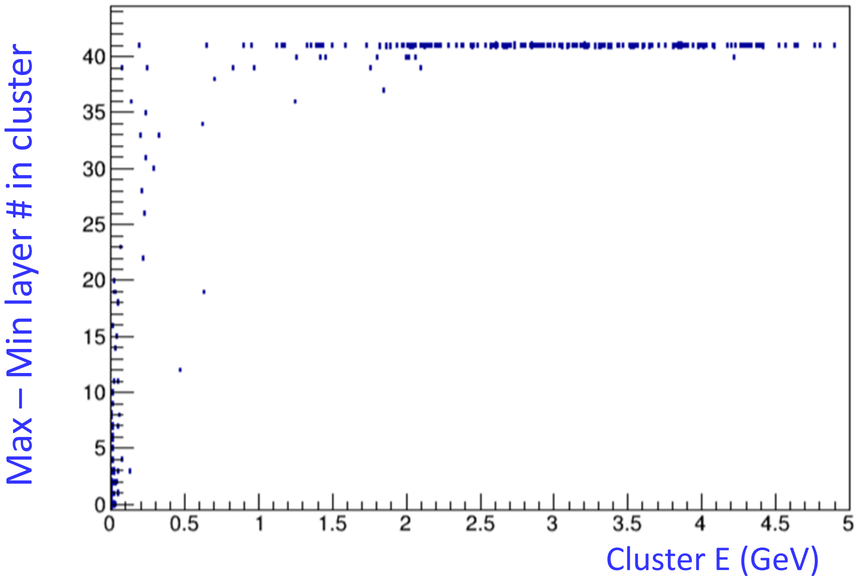


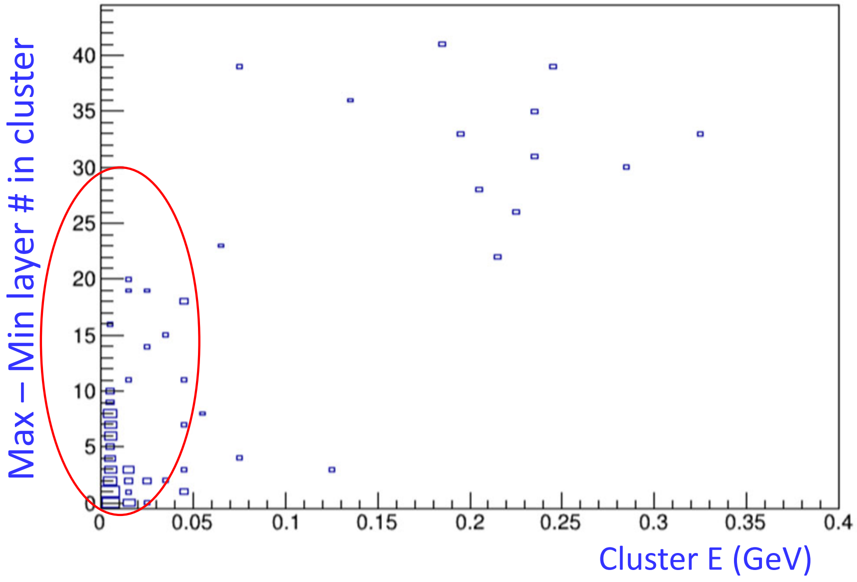
Some residual effects still there?



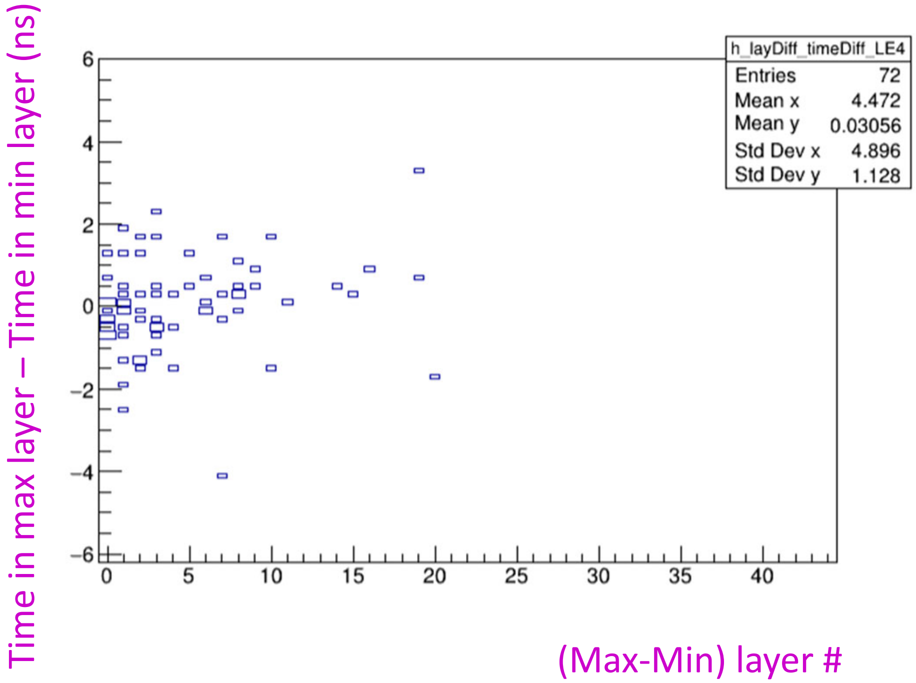
Other quirks

- In low energy clusters, i.e., clusters with very few hits, at times hits are in well- separated layers, which can't be right, e.g.,
  - Cluster with 2 hits, and E = 19.5 MeV, has hits in layers 16 and 27
  - Cluster with 2 hits, and E = 4.2 MeV, has hits in layers 4 and 12
  - Cluster with 8 hits, and E = 48 MeV, has hits in layers 4-6, 10 and 22…
- There is probably something smart we can do, *but for now leave the clustering code as such*
  - Looked at the times of RecoHits in cases with well-separated layers, but no real obvious effects
  - We could make an energy cut on recoHits when they are being used for clustering (currently, we take all)

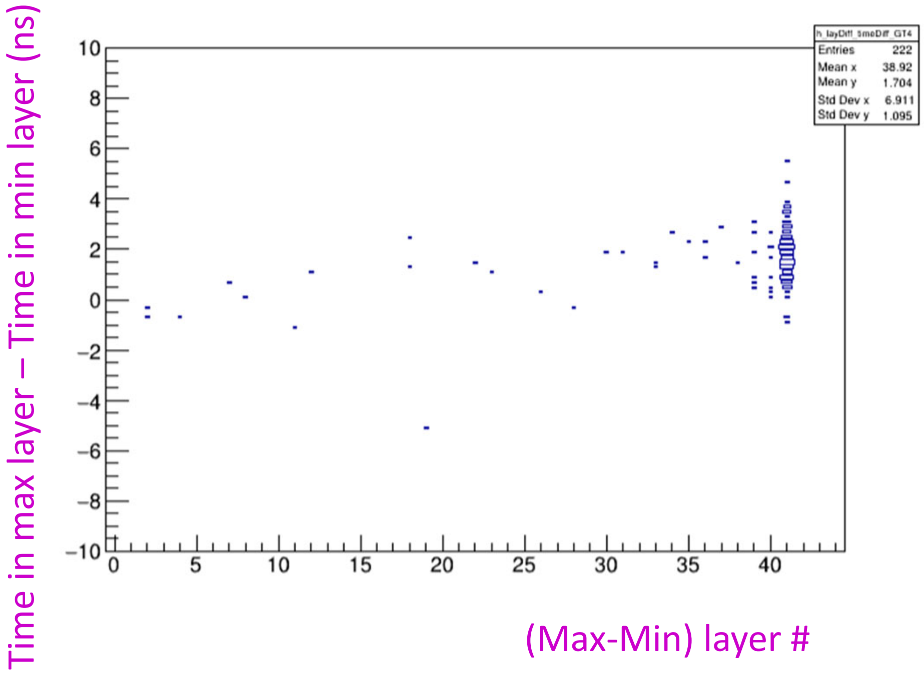




Clusters with ≤ 4 RecoHits



Clusters with > 4 RecoHits



Summary

- Continue looking at single electron, and try some simple things

- Made electron neutrino sample with all fixes – haven’t looked at it yet
    - Use it look at differences between photon and neutron showers – I added many more cluster shape variables in the anatree

- Our NN clustering algorithm is a little too simple, and one could be “smarter”
    - I had tried ordering the list of RecoHit objects according to energy*, but given the way how the clustering works, that had no effect
    - Asked ChatGPT for suggestions for clustering algorithms – will make sense to think about improvements once we have realistic samples, i.e., with overlays, and/or newer geometries →

- Need to understand the Geometry code (and/or the gdml file), since many of the bugs are seemingly coming from there.

*ChatGPT gave me a “sort” command that used something called a Lambda function (had never heard of it – very cool!)

I asked ChatGPT about potential improvements to the nearest neighbor algorithm. Its initial response was to describe jet-finding algorithms at colliders, and then I asked about lower energy experiments. (p.s. on request, it also produced some sample code for the first algorithm – used a library called DBSCAN).

Response: "Yes, there are several ways to improve the nearest-neighbor clustering algorithm beyond its simplest form. Here are a few examples:

1) Density-based clustering: Density-based clustering algorithms, such as DBSCAN, can be used to identify clusters based on local density of hits rather than a fixed distance threshold. This can be useful in cases where the spread of hits is non-uniform or where there are regions with low hit density.

2) Hierarchical clustering: Hierarchical clustering algorithms build a hierarchy of clusters, starting with individual hits and gradually merging them into larger clusters. This can be useful in cases where there are multiple overlapping clusters or where the cluster sizes vary widely.

3) Cluster merging: After the initial nearest-neighbor clustering is performed, it may be possible to improve the cluster definitions by merging nearby clusters that are not well-separated by the initial clustering algorithm. This can be done by defining a new distance metric that takes into account the sizes and shapes of the nearby clusters.

4) Machine learning clustering: Machine learning techniques, such as k-means clustering and spectral clustering, can be used to identify clusters based on patterns in the hit data. These techniques can be more flexible and robust than traditional clustering algorithms, but they require training on labeled data and can be computationally expensive."

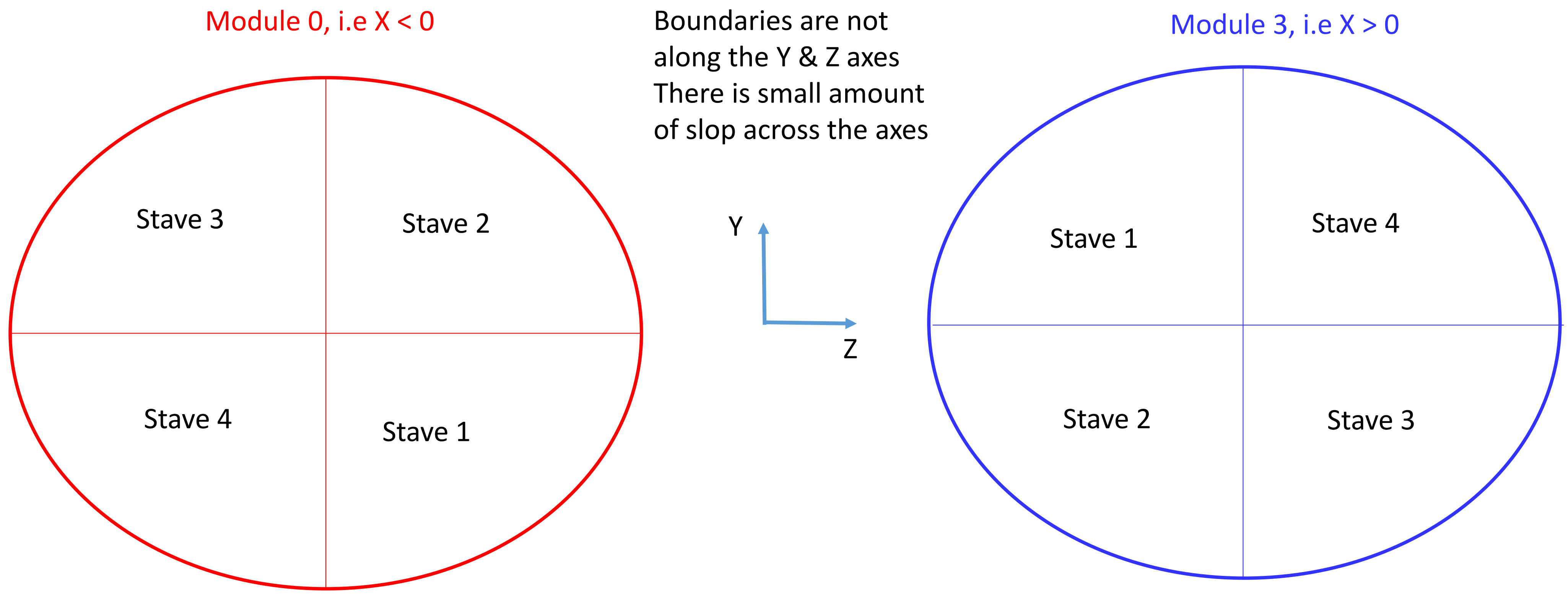
Extra –

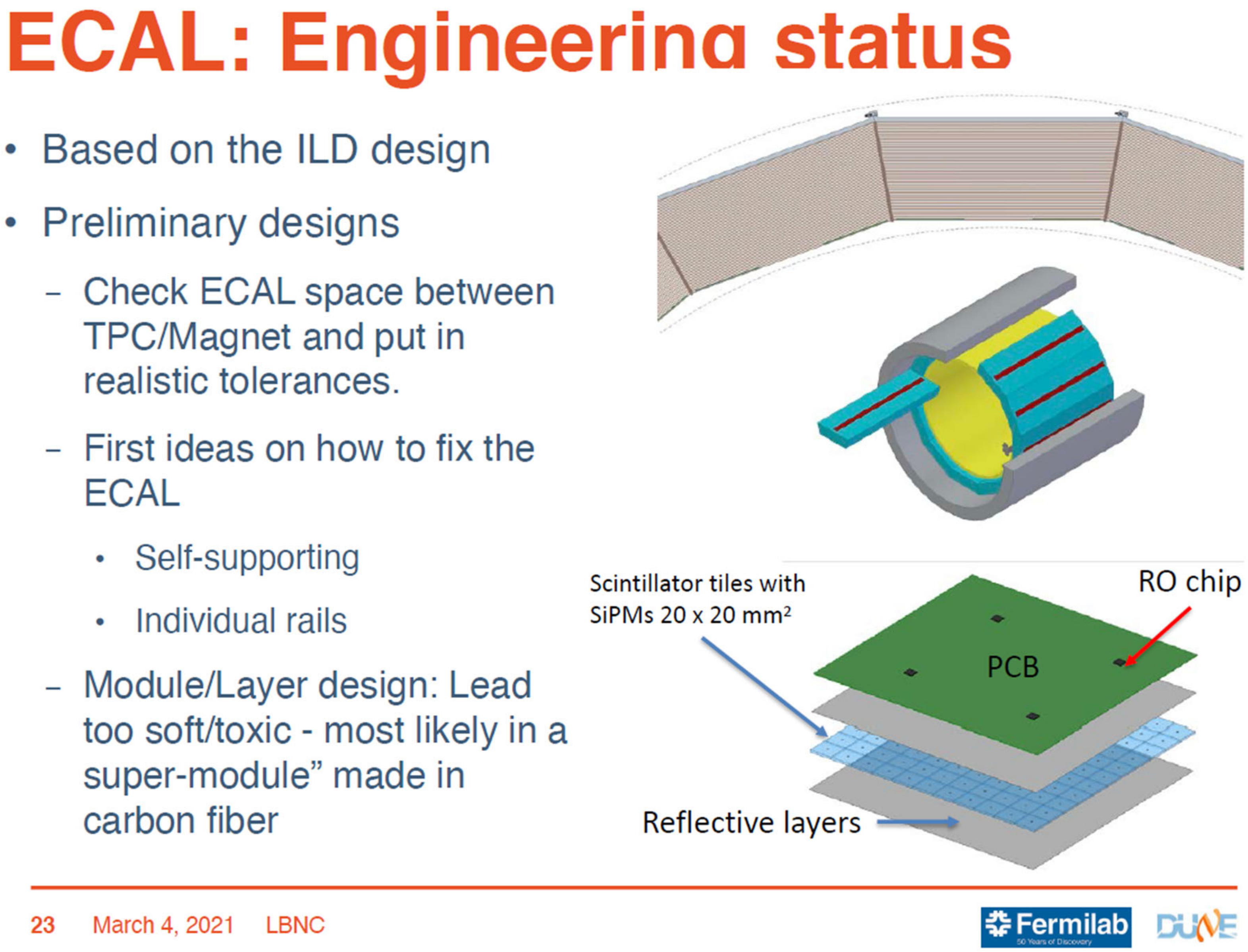
Nomenclature

- ECAL is made up of detector id, staves, modules, slices and layers – these are used to encode the CellID (which is being used during digitization)
  - det_id = 1 (Barrel ECAL), = 2 (Endcap ECAL) [Yoke barrel appears to be 4]
  - Barrel ECAL: has 12 staves (as you go around in phi)
    - Module = 1 for Tile, = 2 for Strips
    - Slice = 1 for absorber, = 2 for Scintillator (for Tiles 3 is for PCB)
    - Layers go from 1 to 42 (first 8 are tiles, remainder are strips)
  - Endcap ECAL: has 4 staves, appear to match the four quadrants
    - Module: = 0 for negative X, = 3 for positive X. So no distinction between strips/tiles?
    - Slice = 1 for absorber, = 2 for Scintillator
    - Layers go from 1 to 42 (first 6 are tiles, remainder are strips)
- Nodenames (in the code) look like
  - BarrelECal_stave10_module02_layer_21_slice2_vol_0
  - EndcapECal_stave02_module03_layer_13_slice2_vol_0

Staves in the Endcap – results based on hand scanning output in log file

Boundaries are not along the Y & Z axes
There is small amount of slop across the axes





ECAL: Engineering status

- Based on the ILD design
- Preliminary designs
  - Check ECAL space between TPC/Magnet and put in realistic tolerances.
  - First ideas on how to fix the ECAL
    - Self-supporting
    - Individual rails
  - Module/Layer design: Lead too soft/toxic - most likely in a super-module” made in carbon fiber

Scintillator tiles with SiPMs 20 x 20 mm²
RO chip
PCB
Reflective layers

From a talk by either Alan or Eldwan

Each strip is 4 cm wide

Structure of AnaTrees

- Although you have arrays like CCNC, Interaction Type, etc., whose length is the number of interactions within a spill, the reconstructed information in CALO, TPC is aggregated over all of them.
  - Current files have 1 interaction/spill, so it’s a moot point

Missing DigiHits problem – how often SimHits are being discarded

	Barrel Tile	Barrel Strip	Endcap Tile	Endcap Strip
% failed before Leo's fixes*	0 out of 9K	0 out of 55K	0 out of 3.9K	~1000 out of 9K
% failed after Leo's fixes*	1 out of 22.7K	0 out of 81K	1 out of 3.6K	89 out of 10.3K

*Caveat – in the old sample, I had not saved Sim/Digi Layer numbers, so deciding what is tile/strip required cuts on X,Y,Z, of the hits

In the new samples, I simply use the layer numbers to decide whether the hit is in the tiles or in the strips

(I do need to use X, Y, Z to decide whether endcap or barrel, so there may be some edge effects)

In new code lose ~0.8% of SimHits in EC Strips (instead of 9%) - From Leo, "…traced the problem to somewhere in the guts of ROOT, *TGeoNavigator::FindNode(Double_t, Double_t, Double_t)* specifically. Looking at the release notes for the last 3 (minor) versions of ROOT, I don't see anything was done to this routine, but the real issue could be elsewhere. " It may be pointing to a problem in how the geometry is being done, either in the gdml file or in Geant – apparently Eldwan kept changing conventions as he tried different geometries

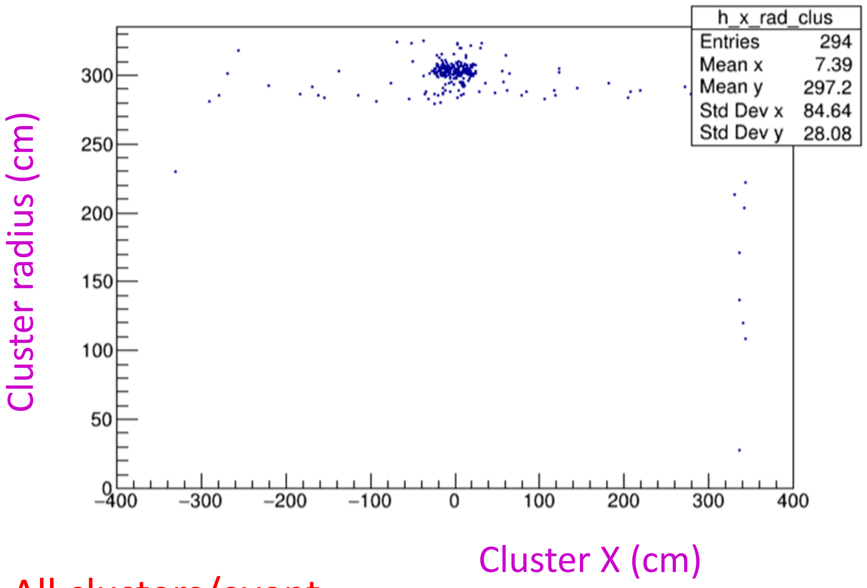
Background: the problem can be seen in CalculatePosition() in garsoft/ReadoutSimulation/ECALReadoutSimStandardAlg.cxx

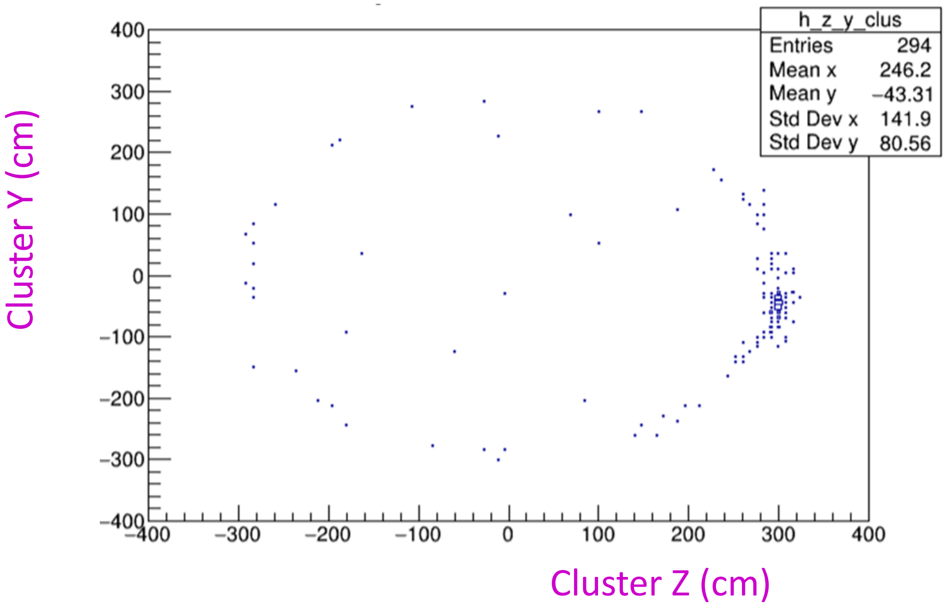
As far as Lorenz and I have understood, the "World" location of the SimHit is first transformed into a "Local" location. This "Local" position is then shifted to where the SiPM is located, and then the "new Local" position is transformed back into a "new World" position.

At this point, the nodenames of the old and new World locations are compared. If they are different, the hit is not digitized

Fix to clustering algorithm (4/14/23) – following two slides refer to this

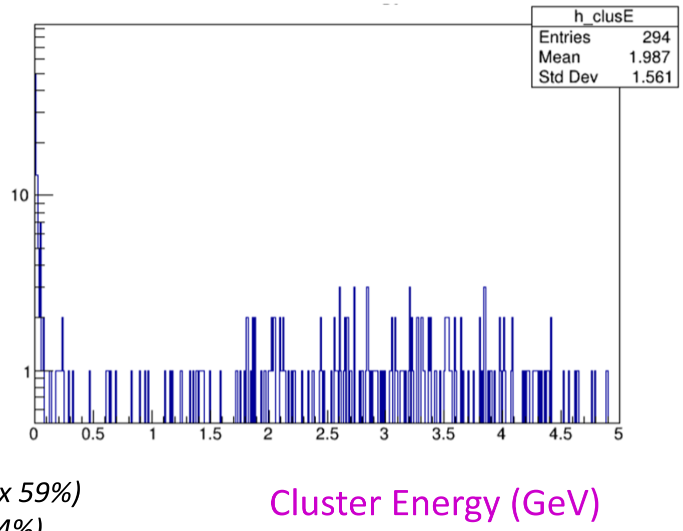
- Found a “feature” in the clustering code – it was designed to speed up clustering but ended up skipping legitimate hit-pairs (RecoAlg/NNClusters.h)
  - The clustering algorithm determines the distance between every pair of hits, and tests to see if it passes the NN distance cut (set via FCL file – currently at 25 cm)
  - However, to speed up clustering the NN distance check is not attempted when the two hits are “too far” apart based on X-coordinate bins, which is hard-coded – care was not taken to ensure that the two tests are compatible, and this test, which is made before the NN distance cut, simply ignores many hit-pairs
- For the time being, I have removed the latter test – since I am looking at single electron samples, speed is not an issue
  - **See an immediate improvement in the metrics that I have been looking at**
  - Once we settle on the NN distance cut, we could, in principle, reinstate the test to increase speed – will have to use events with overlays to study this



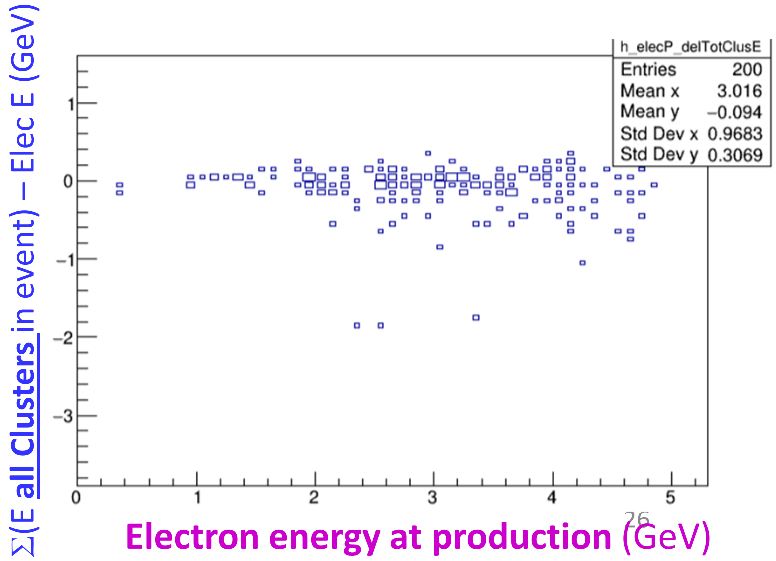


All clusters/event

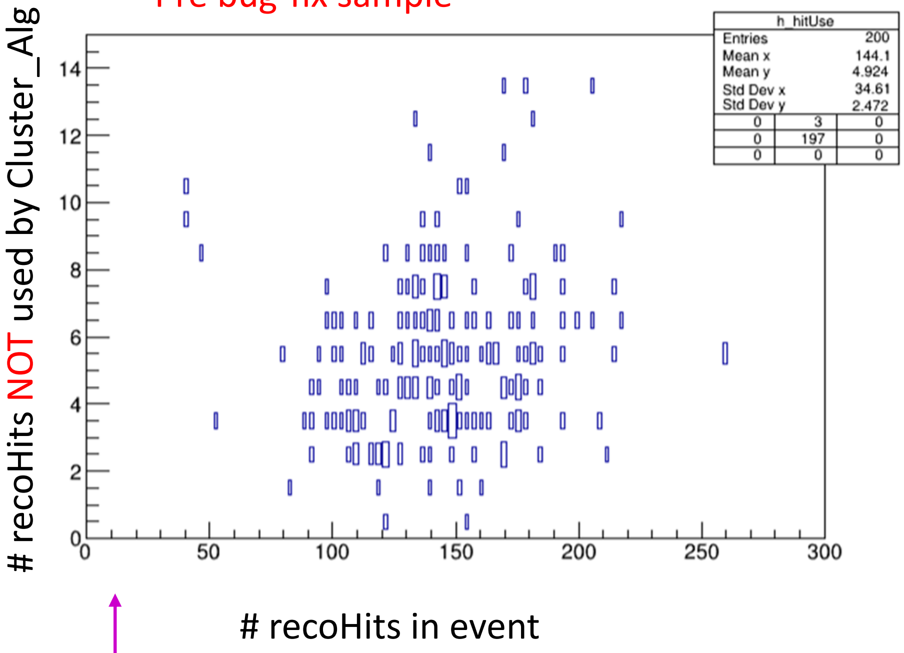
Single Electrons
~ 1.5 clusters/event
(pre bug-fix sample,
2.8 clusters/event)



<Cluster E> is ~ 2x larger than pre bug-fix
clusters with E < 100 MeV (28% - *pre bug-fix 59%)*
clusters with E > 1 GeV (65% - *pre bug-fix 34%)*



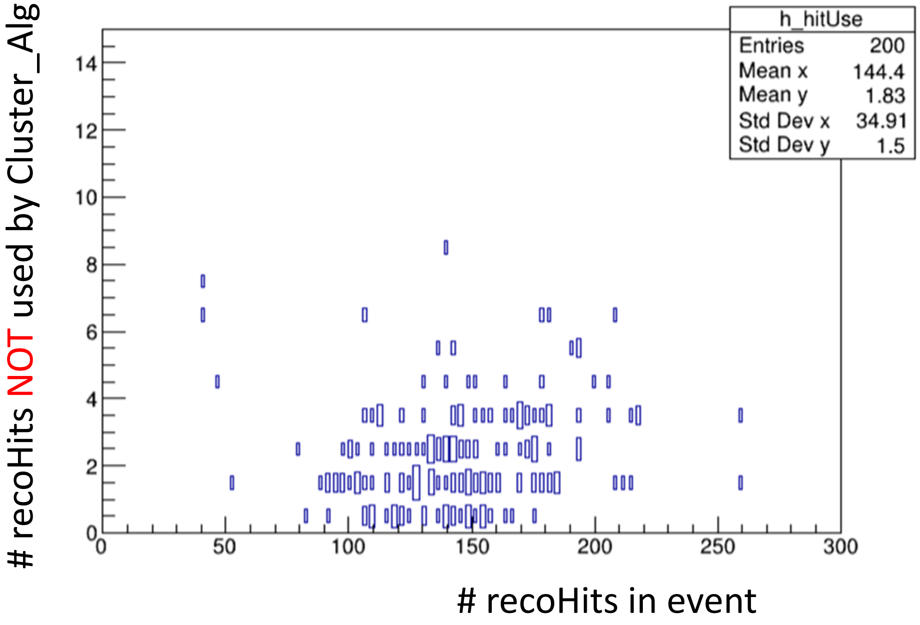
Pre bug-fix sample



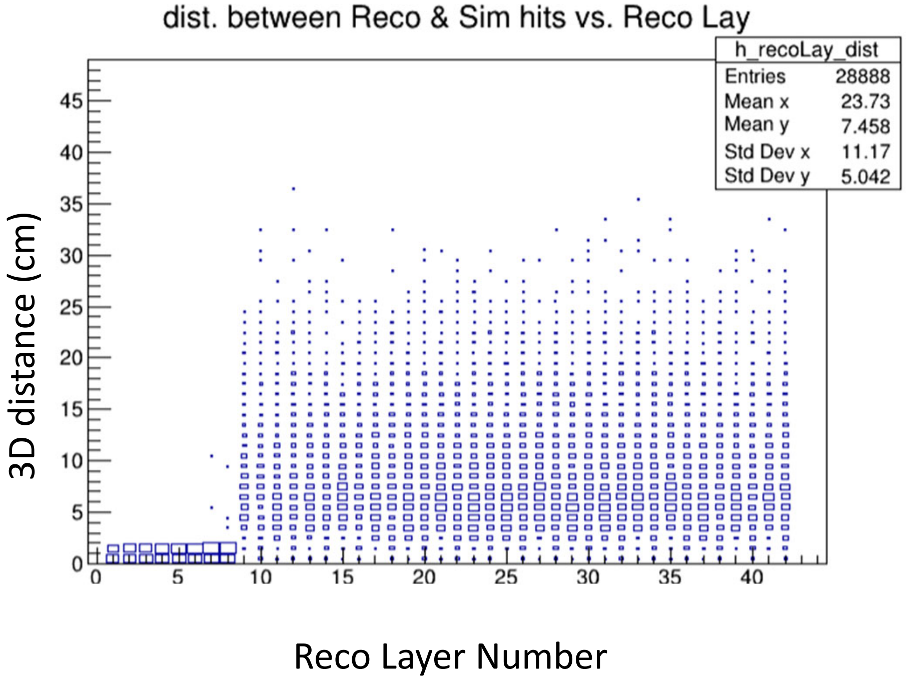
Before bug-fix, ~3% of the events have Y-axis < 2

After bug-fix, ~ 50% of the events have Y-axis < 2

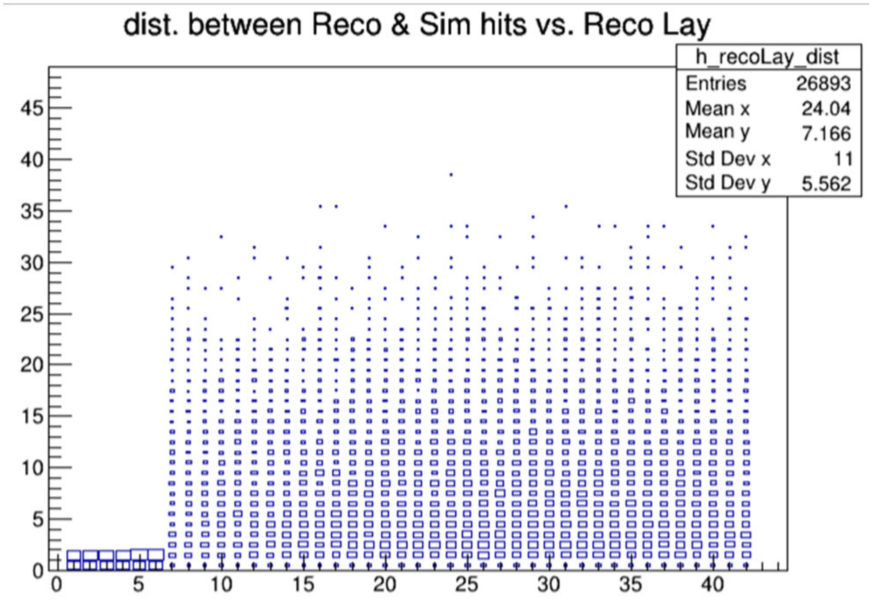
Post bug-fix sample



Downstream Barrel



Endcap at +X



I shoot single electrons from the center of the TPC at the downstream barrel, and one of the endcaps (two separate files)

Made this plot for Francisco