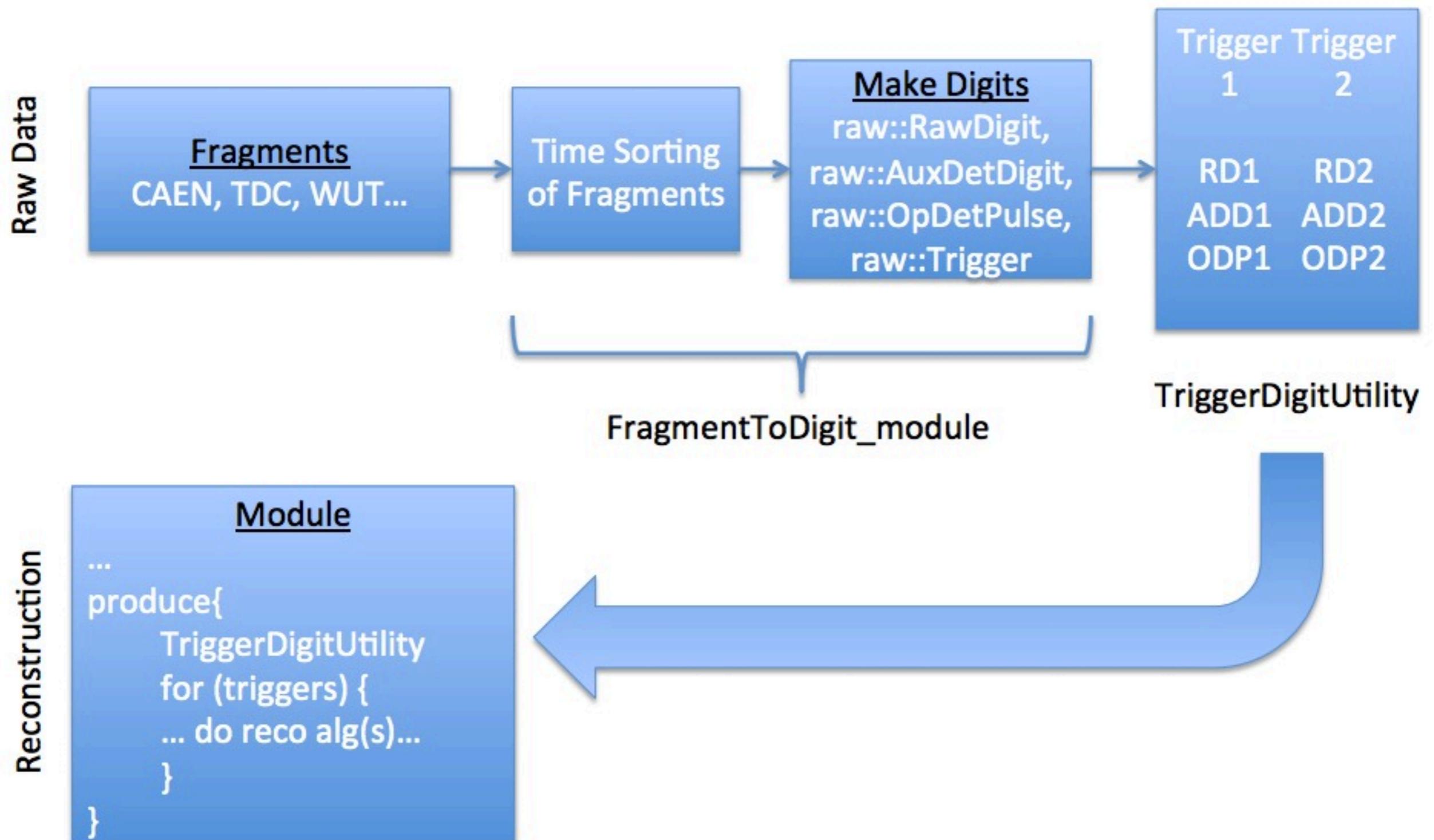


Beam Line Reconstruction

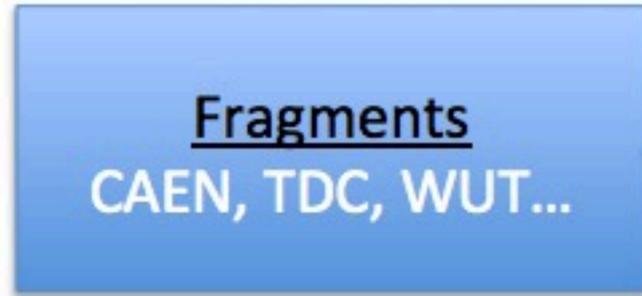
Daniel Smith and Ryan Linehan

Status of reconstruction, as discussed on Friday



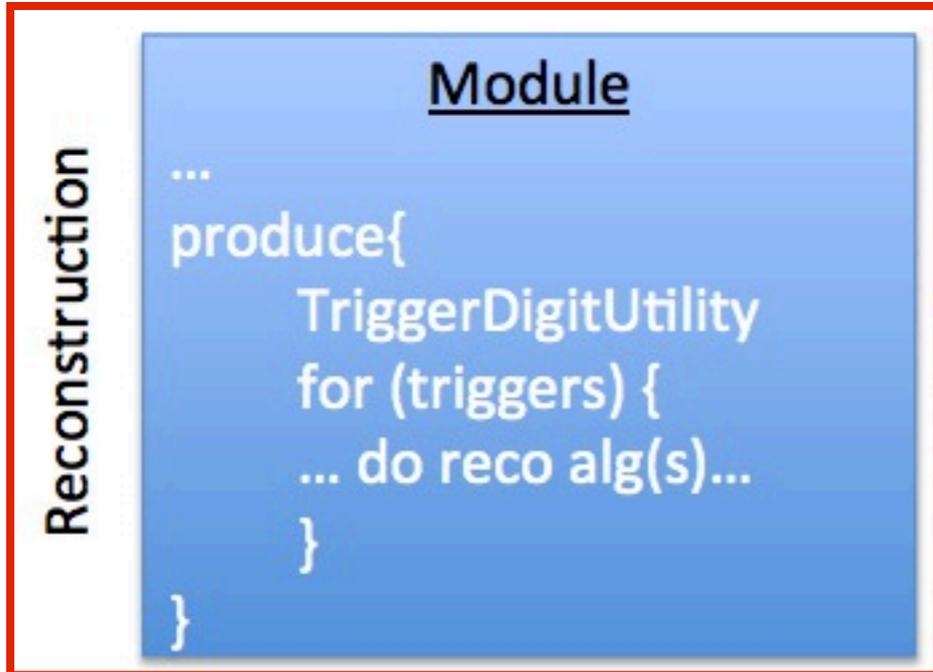
(Figure from Jen)

Raw Data



FragmentToDigit_module

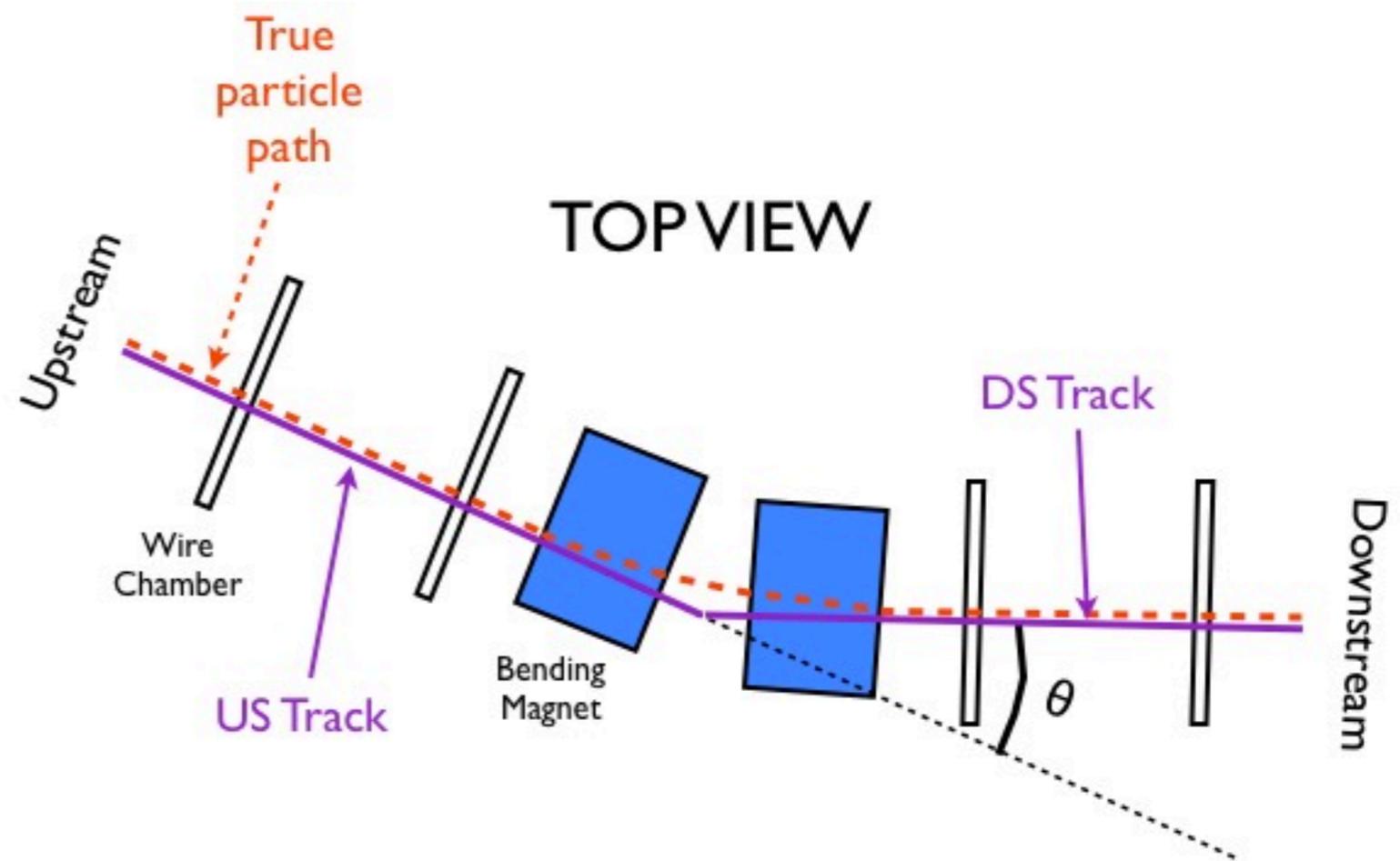
TriggerDigitUtility



Several reconstruction modules for the beamline detectors exist in python*, but now need to be made LArSoft-friendly

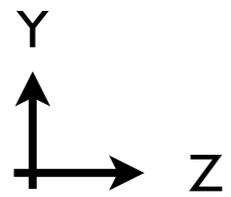
- Momentum reconstruction
- Time of flight

Momentum Reconstruction Using Wire Chambers



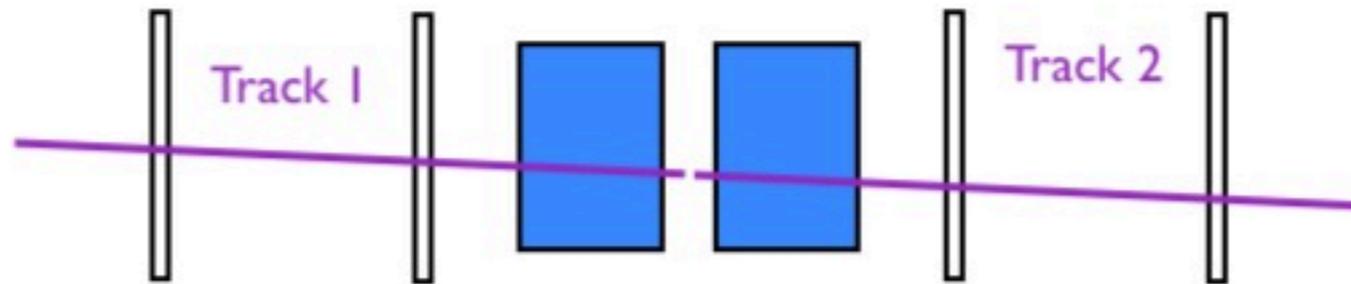
Process:

- 1.) Passing particle creates hits on each WC
- 2.) We build tracks upstream and downstream of the magnets
- 3.) Momentum determined from magnet/B-field settings and reconstructed angle θ between tracks

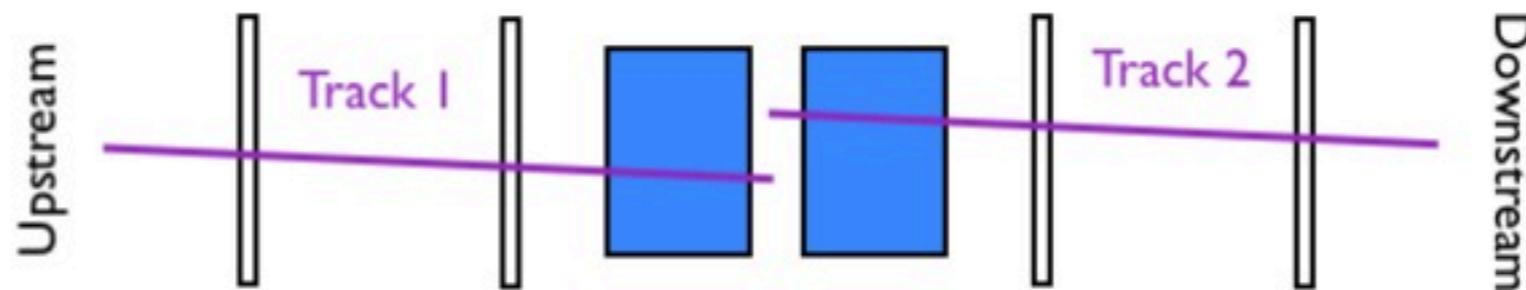


SIDE VIEW

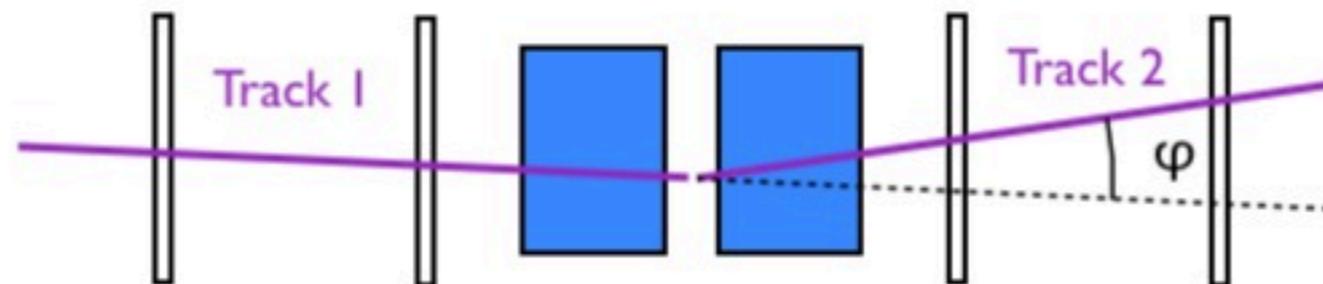
Good Track: Continuous



Bad Track: Discontinuous



Bad Track: Kinked



Also need cuts on track quality

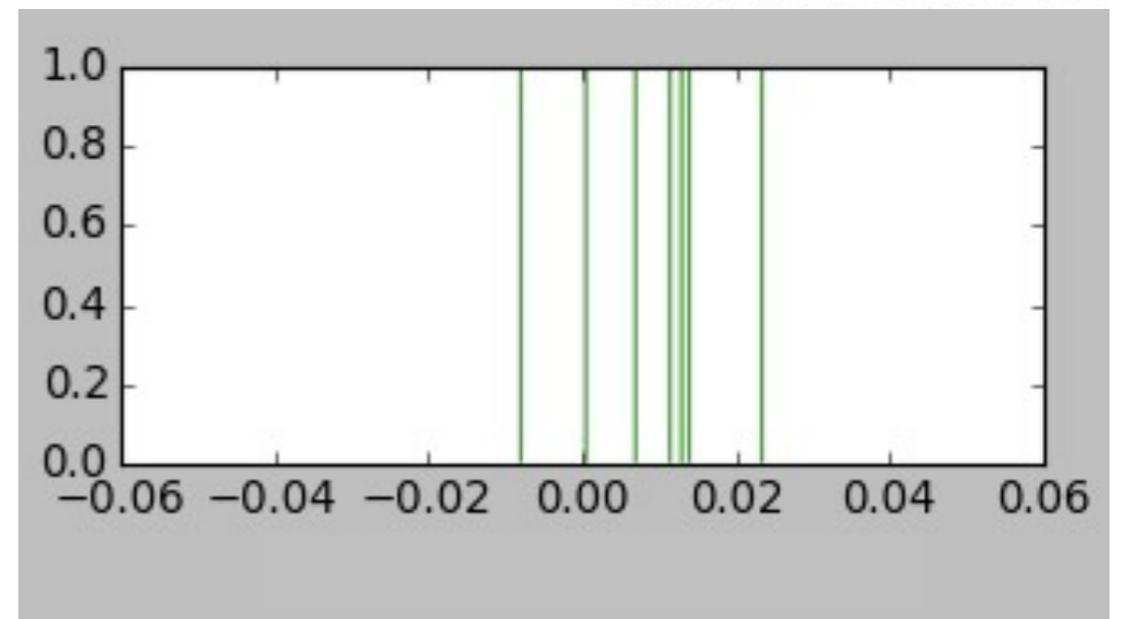
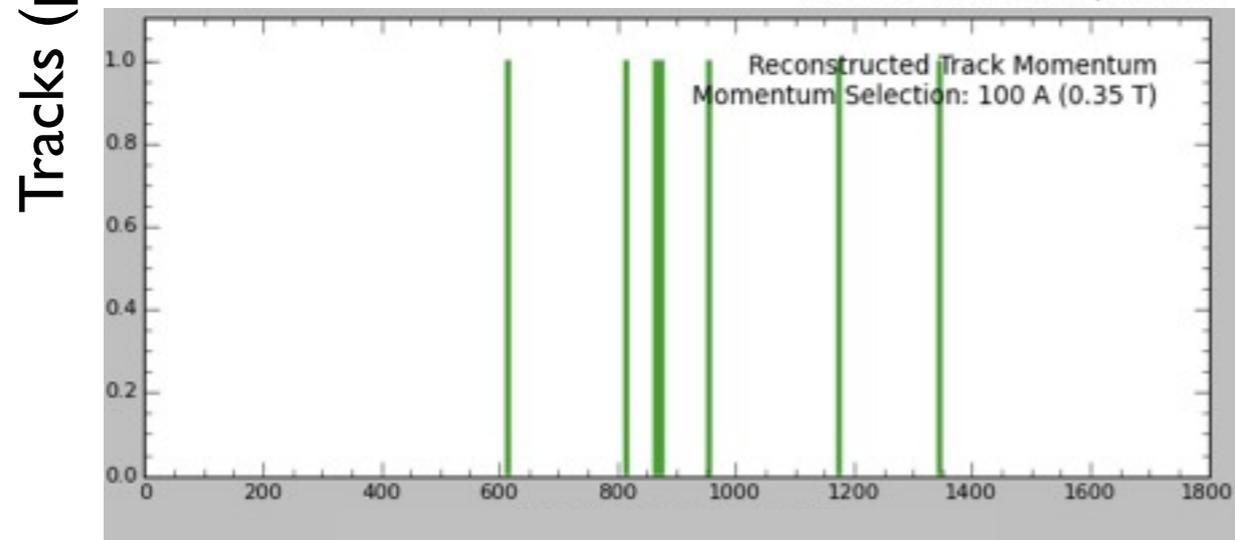
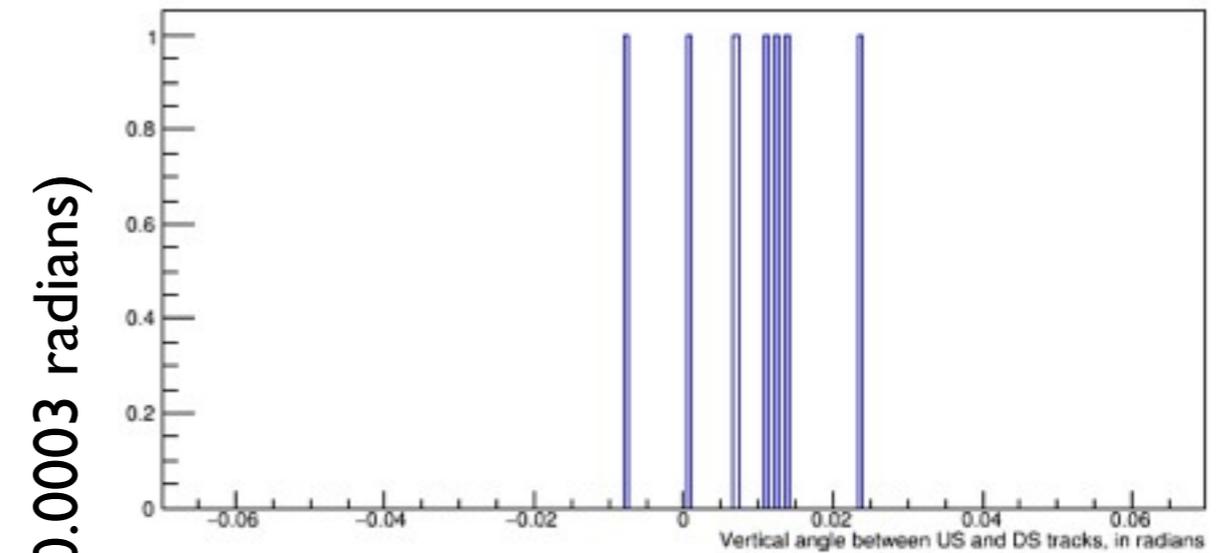
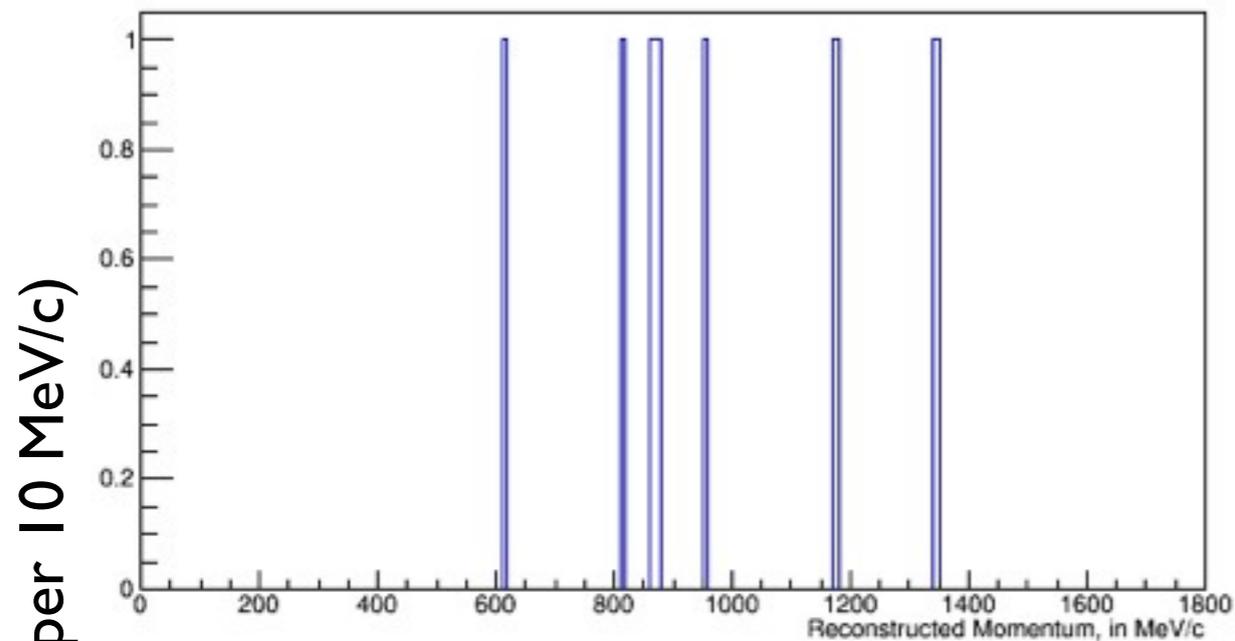
- Need good spatial agreement on where US/DS tracks meet between magnets

- Require that tracks aren't too kinked in the y direction (φ , or y kink is low)

Need to reconstruct all of these: momentum, y kink, and $\Delta X, \Delta Y, \Delta Z$

For triggers with ONLY one hit per wire plane axis

C++ momentum and y kink reconstruction



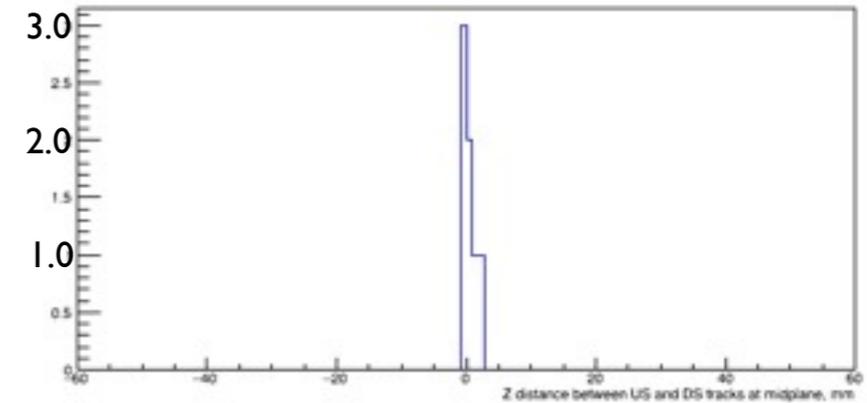
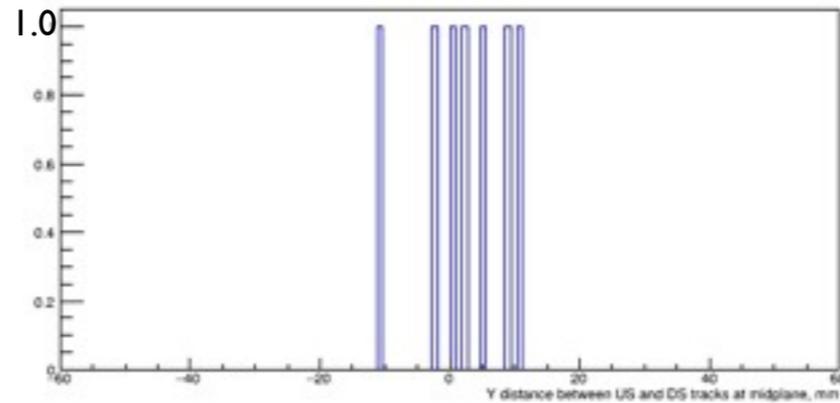
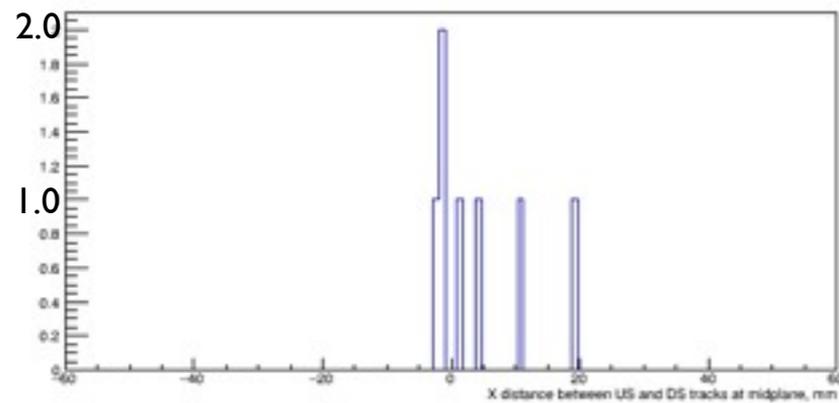
Reconstructed Momentum, MeV/c

y_{kink} (radians)

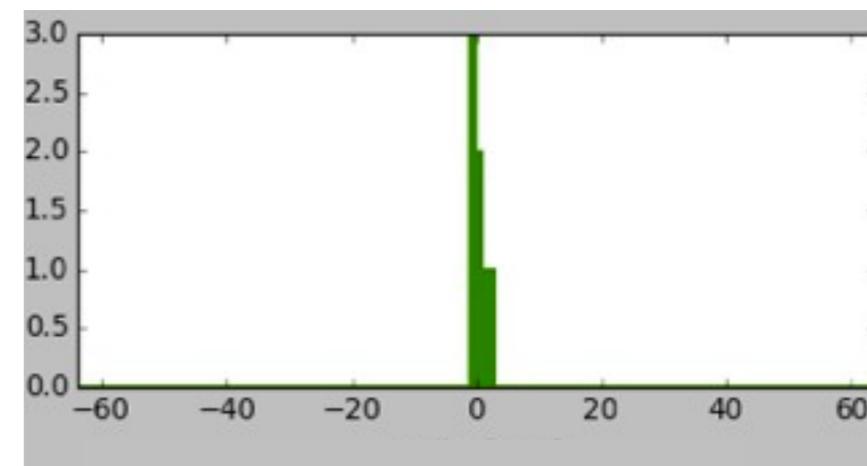
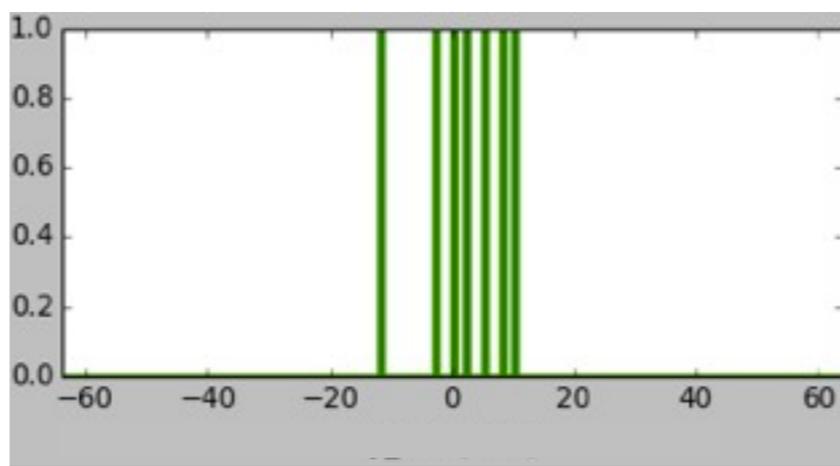
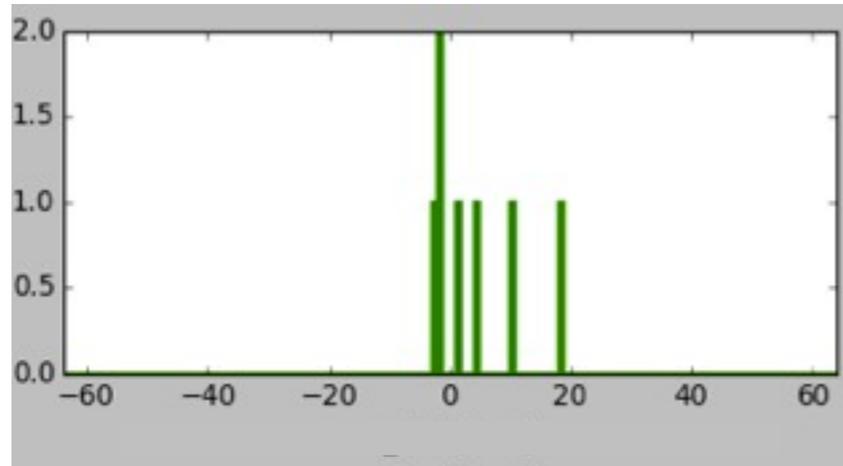
Python momentum and y kink reconstruction

For triggers with ONLY one hit per plane

C++ X,Y,Z distance reconstruction



Tracks (per 0.5 mm)



ΔX of track ends (mm)

ΔY of track ends (mm)

ΔZ of track ends (mm)

Python X,Y,Z distance reconstruction

Where we are:

Right now:

Feature branch: WireChamberTracking

- Code's running inside a LArSoft module (data imported as text file)
 - Still trying to get data from the AuxDetDigits for the 4 WCs

End of today/tomorrow:

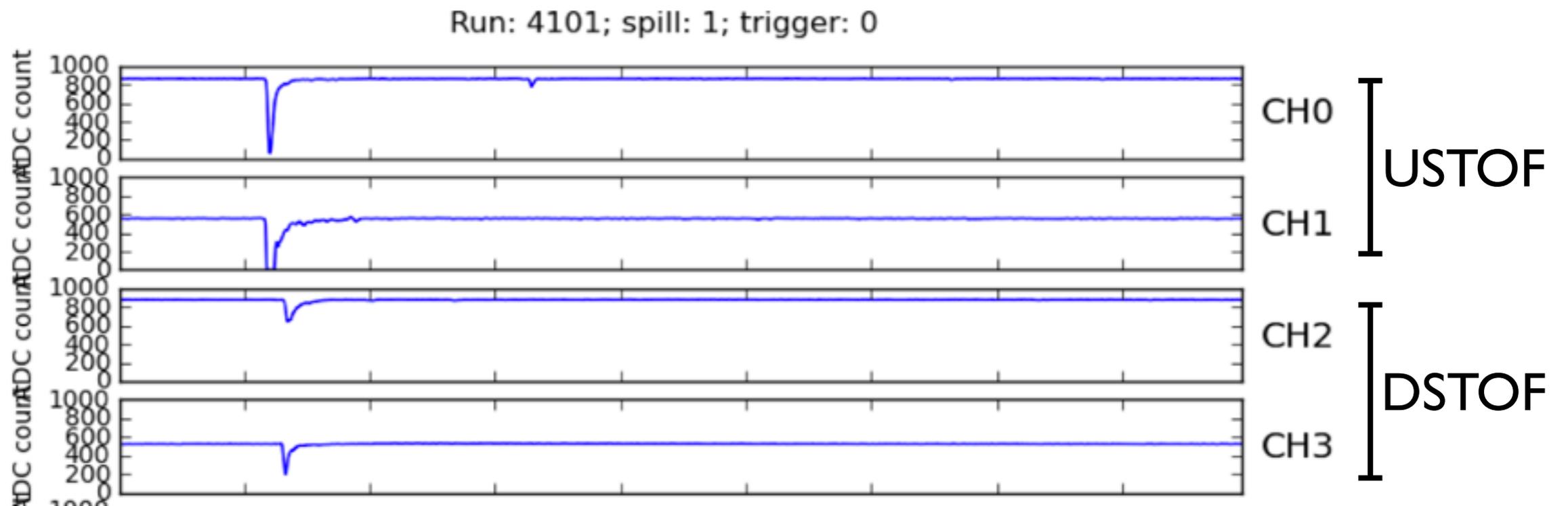
- Hopefully have plots of the location/direction of the particle as it enters the TPC

End of tomorrow/Thursday

- Hopefully finish data product for use with WCTrack

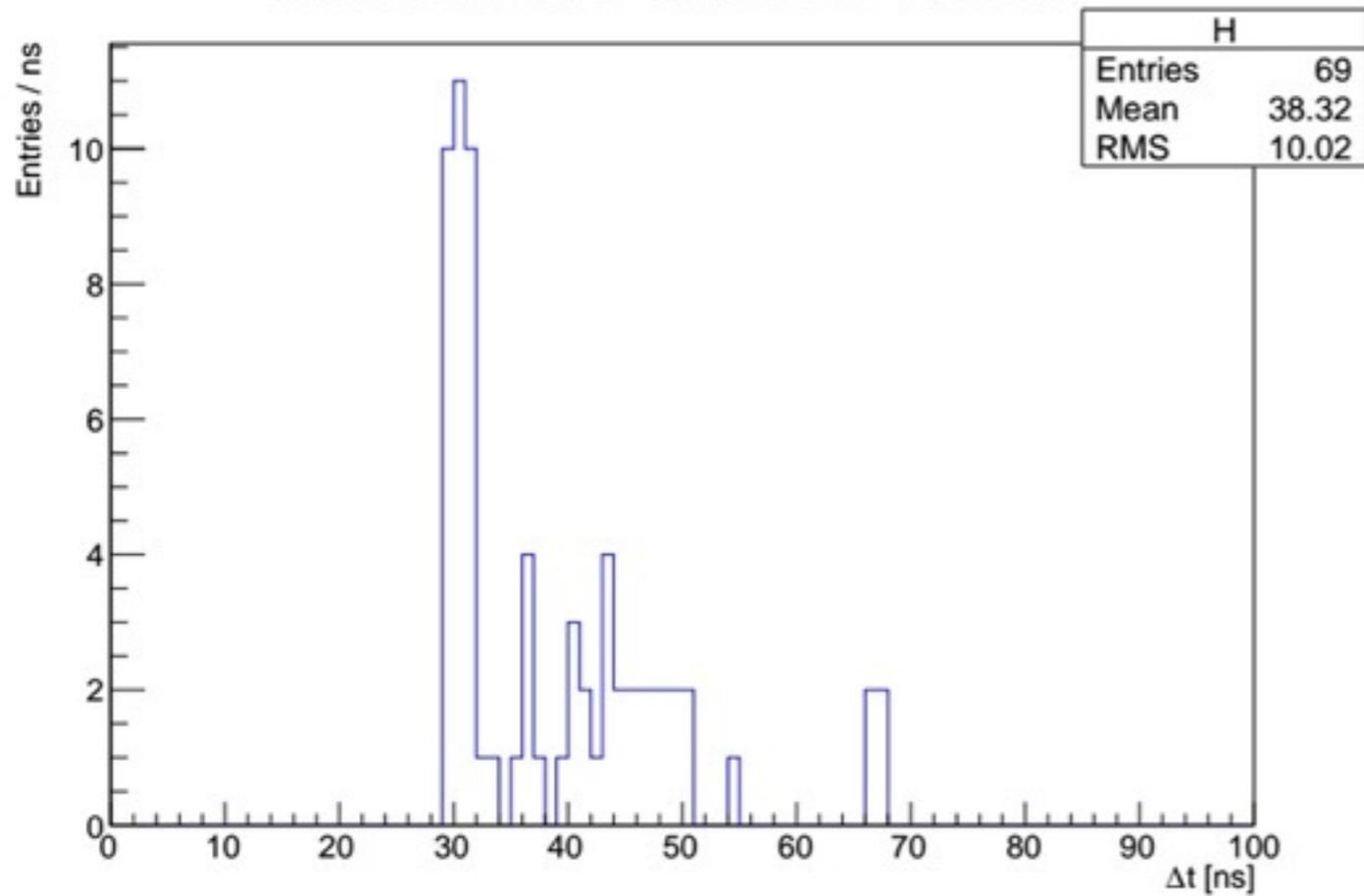
Time of Flight in C++

- Find hits in waveforms of each TOF paddle
- Match hits for upstream and downstream pairs of PMTs
- Find and record the difference between hits on the upstream and downstream paddles



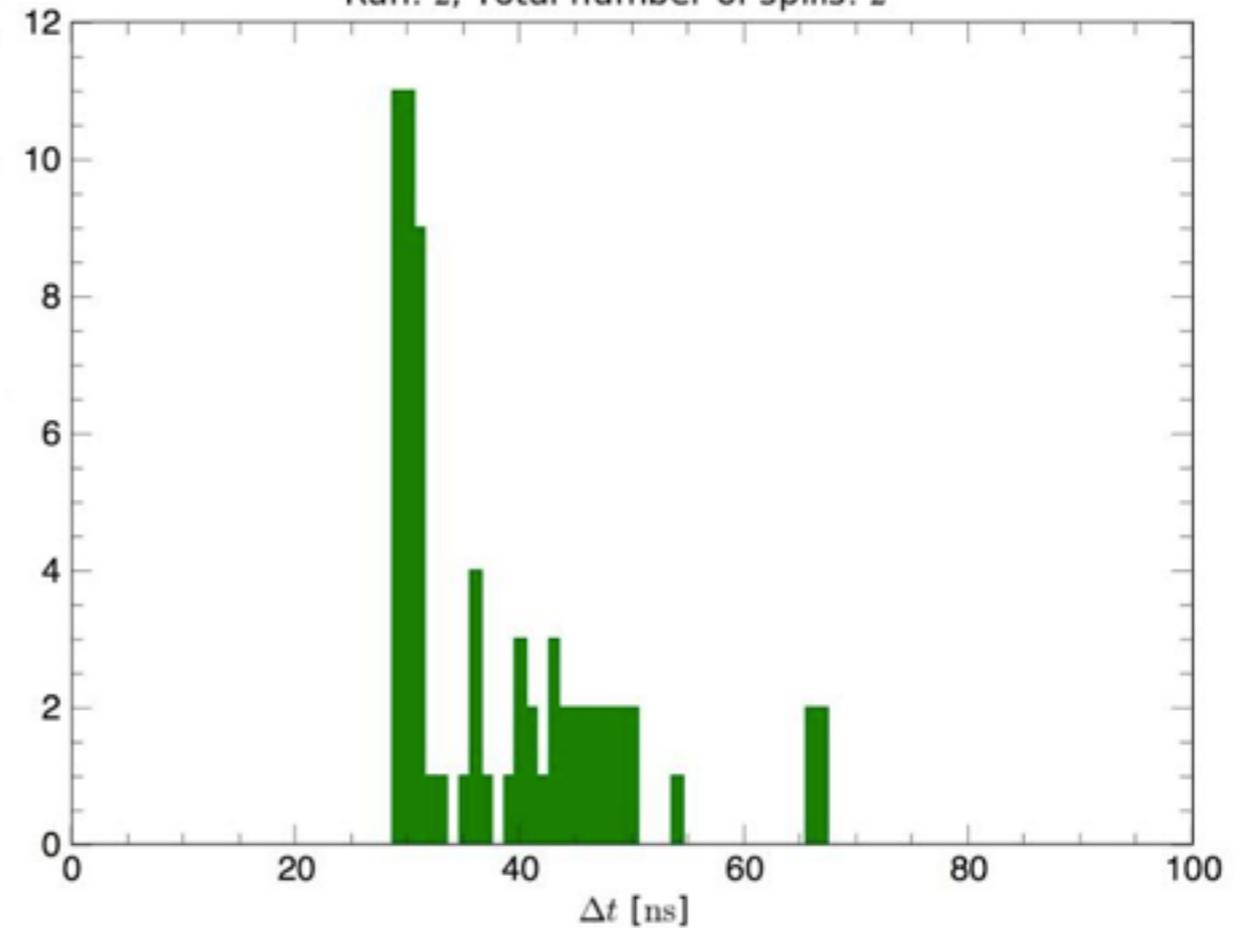
C++ Version

Δt between DSTOF and USTOF V1751 hits



Python Version

Δt between DSTOF and USTOF V1751 hits
Run: 2; Total number of spills: 2



Future plan:

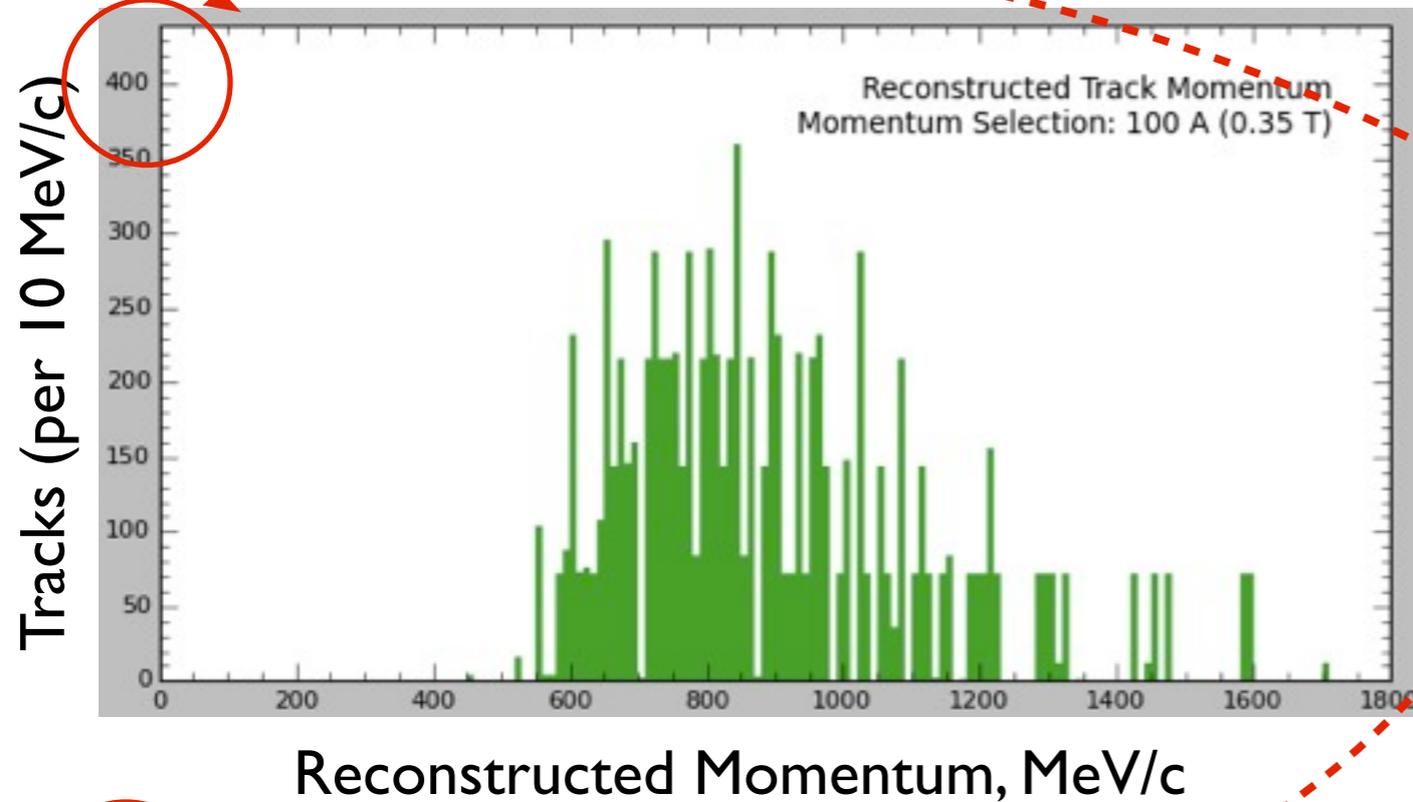
- Create a fit for the plot in ROOT
- Learn LArSoft
- Port TOF into LArSoft

Summary

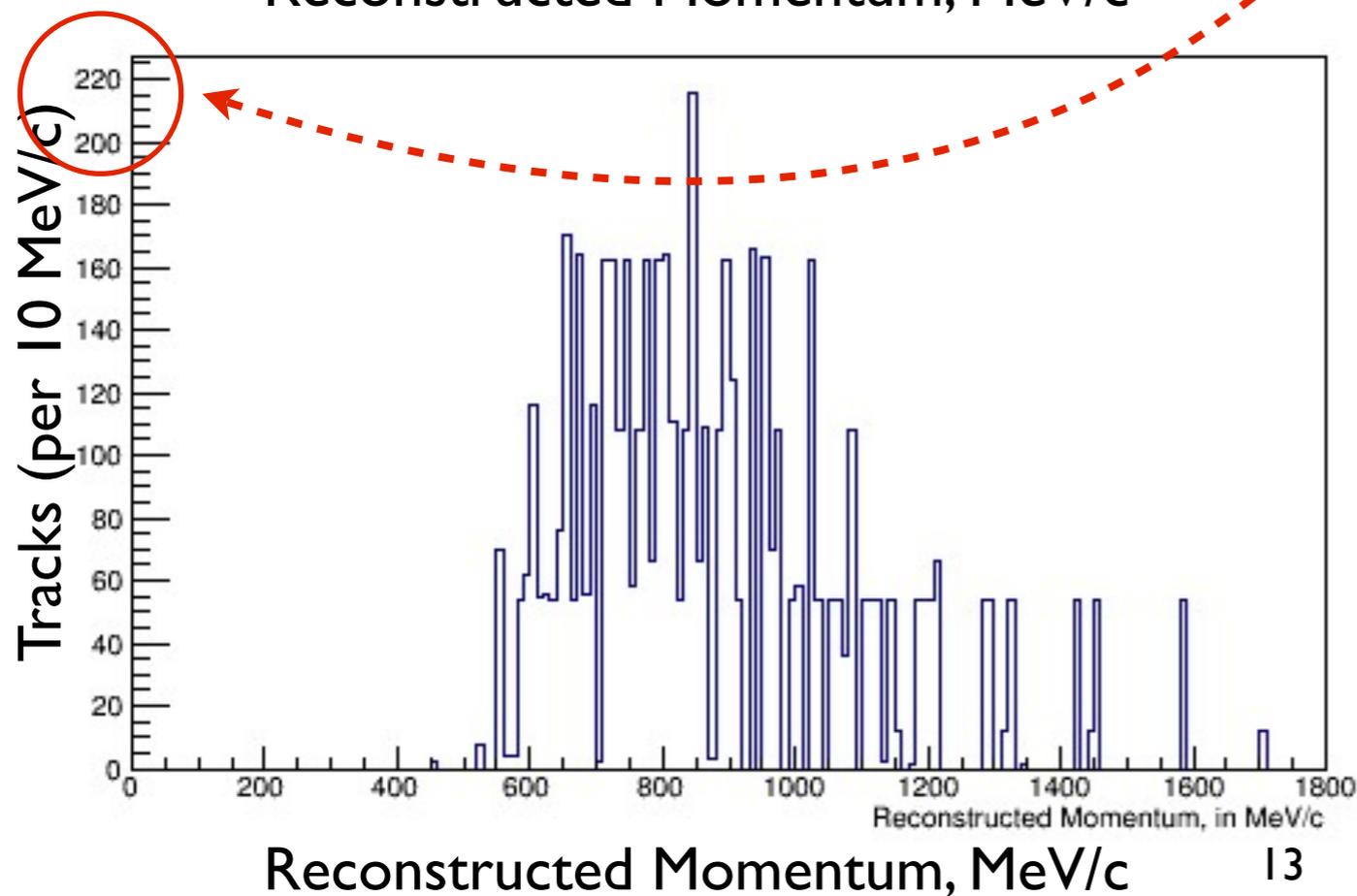
- Momentum reconstruction ported into C++, and almost into LArSoft
- Time of flight reconstruction ported into C++
- Possibly working on porting others in once these are done

Backup

Reco Pz without I-hit/plane restriction (all possible combinations)



Similar but note the larger track counts in the python version



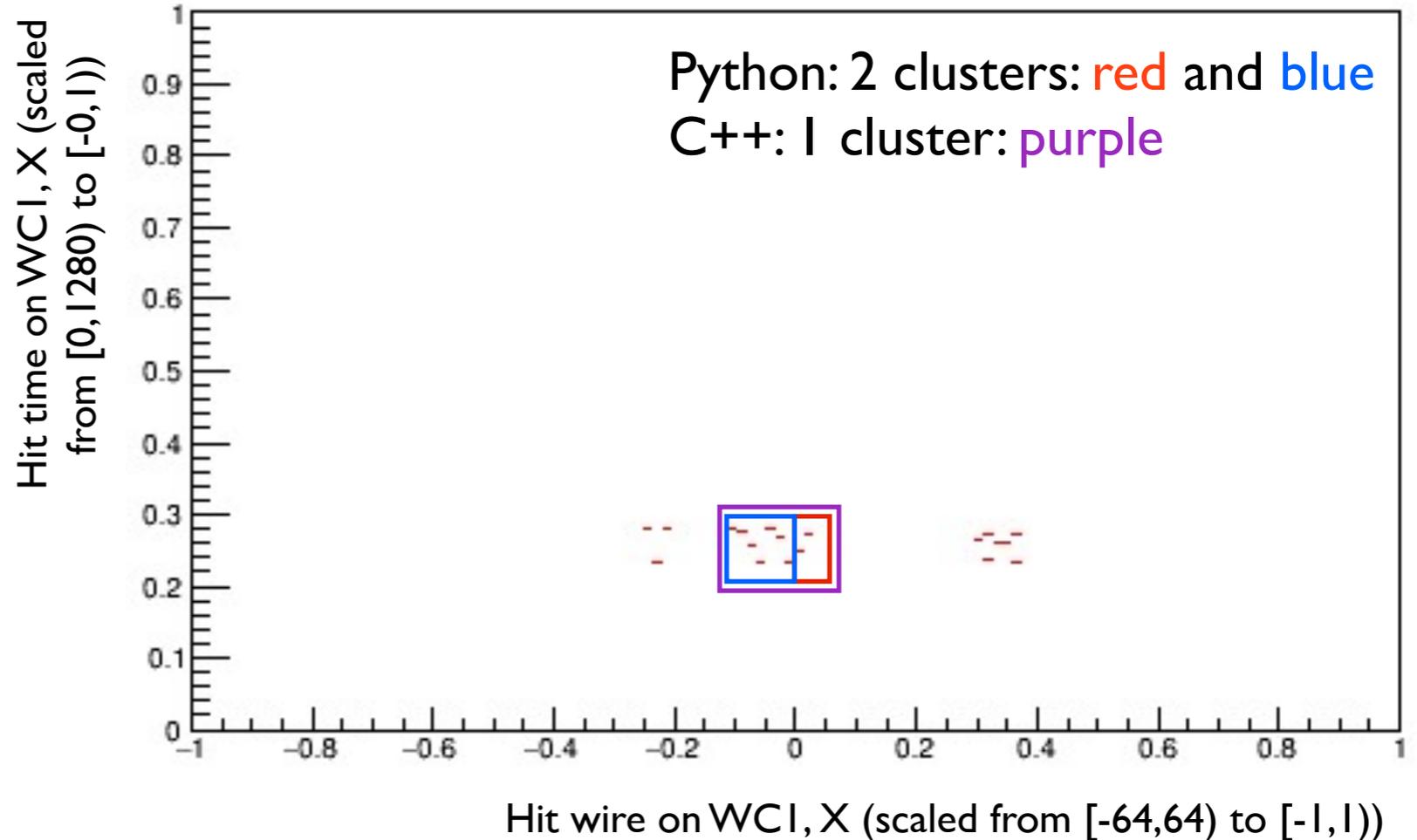
Reason lies in clustering:

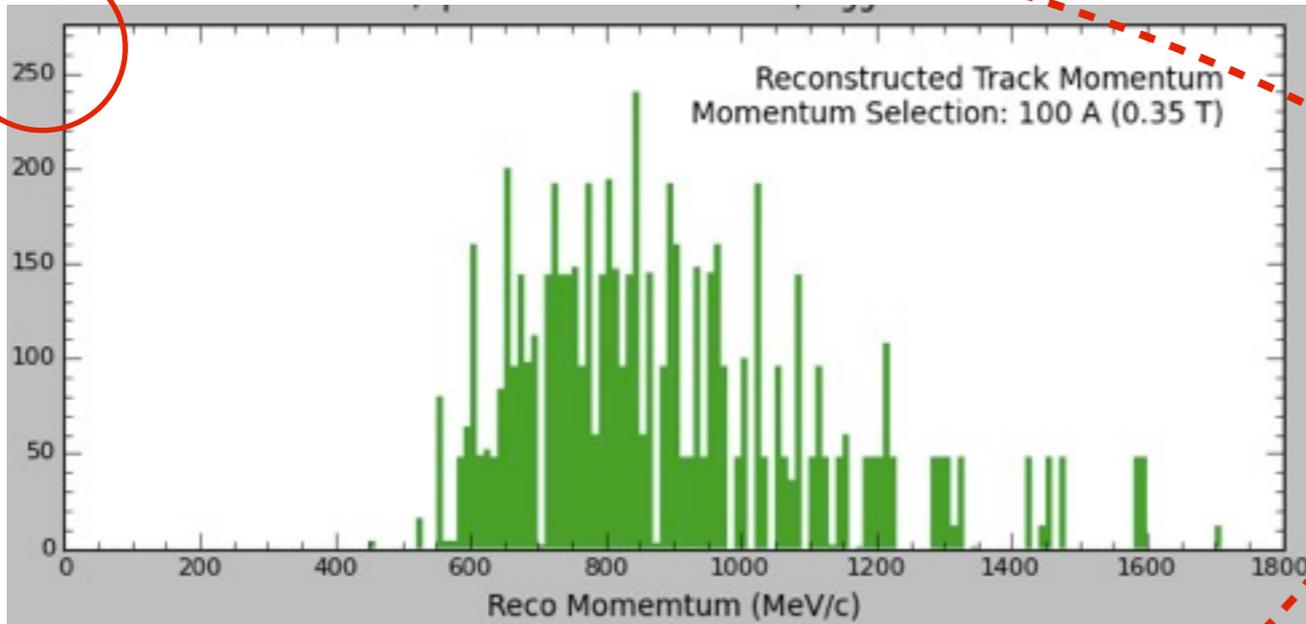
- Python:

- 1.) cluster hits on each TDC
- 2.) average neighboring good clusters on the same wire chamber

+ Single clusters split over 2 TDCs can get broken into 2 clusters, and may not get averaged properly

- C++ version: similar process, but cluster hits over each wire chamber





After tweaking averaging parameters in the python code, we get better agreement

