

# HitFinder Algorithm for MicroBooNE

*(and LarSoft, Argoneut, LBNE, ....)*

**LArSoft Reconstruction  
Meeting**

**Jonathan Asaadi  
Syracuse University**



# Outline

- Motivation for looking into HitFinder
- Overview of FFTHitFinder
- Existing issues with the FFTHitFinder
- Propose a “new” HitFinder Algorithm (GausHitFinder & GausHitFinderAna)
- Preliminary look at GausHitFinder Performance
  - Side-by-Side comparison with FFTHitFinder
- Outstanding Issues and other weirdness...
- Next-steps / Conclusions

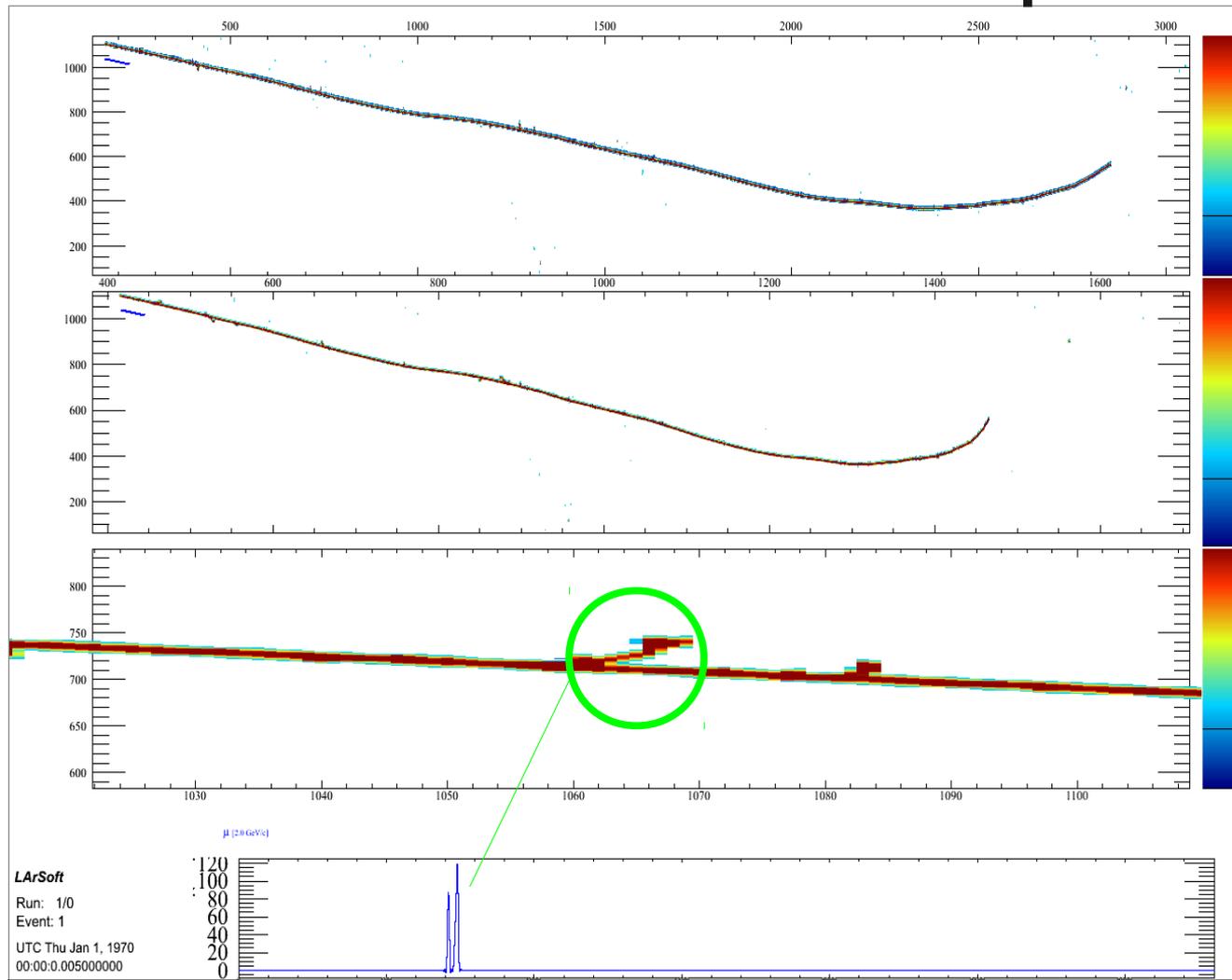
# Motivation for Looking into FFTHitFinder

- **There has been a lot of effort and talk going into our reconstruction code**
  - SpacePoints, Seeds, Showers, Clusters, 3D Tracking, Kalman Filters, etc...
- **All the reconstruction efforts use the hits**
- **Some of the problems faced by the reconstruction algorithms could have their root in poor quality hits**
  - Could cause tracking algorithms to go astray or make 3d projections fail....

Question:  
**How good are the hits being found by  
the FFTHitFinder?**

# Overview of FFTHitFinder *(Simplified)*

FFTHitFinder takes as input signals on wires that have already had an FFT run on them to remove electronic signal shaping and looks for distinct pulses (Gaussians)

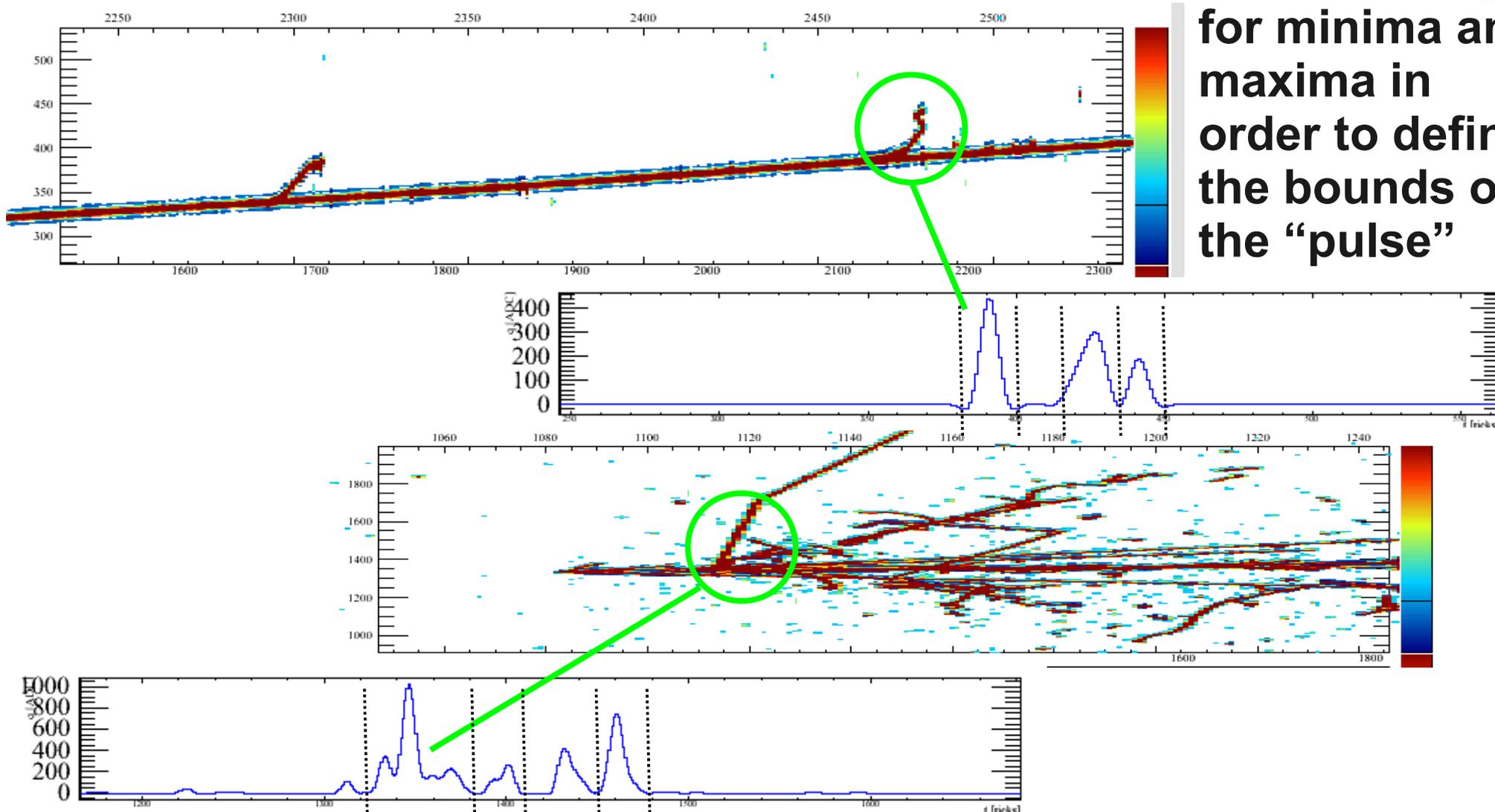


**Fits the pulses with a Gaussian and reports back information on the “hit” found.**

# Overview of FFTHitFinder

*(a little more detail)*

Loop over the wires looking for minima and maxima in order to define the bounds of the “pulse”



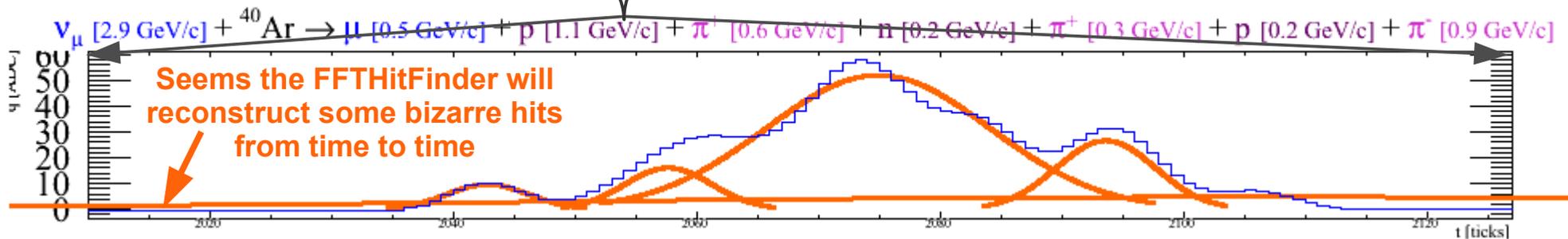
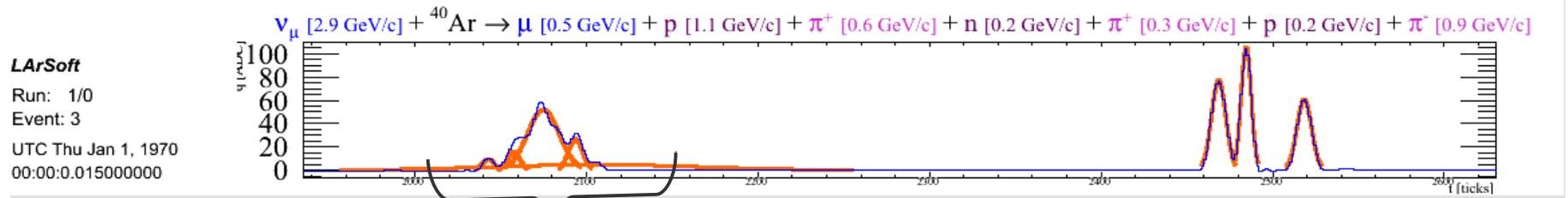
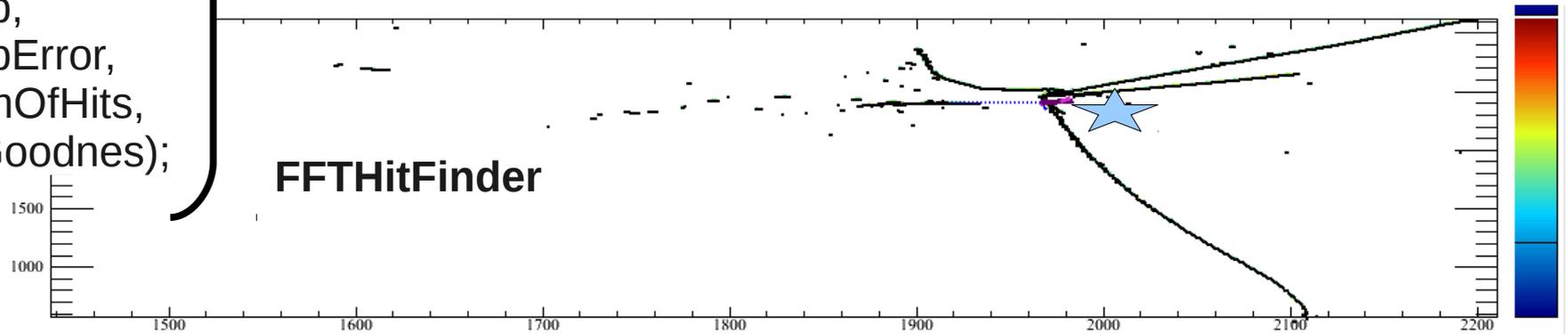
If there are multiple “pulses” near each other it then merges them and attempts to do multiple Gaussian fits for hits over threshold <sup>5</sup>

# Output of FFTHitFinder

```
recob::Hit hit(wire,
  Starttime,
  StartTimeError,
  EndTime,
  EndTimeError,
  MeanPosition,
  MeanPosError,
  Charge,
  ChargeError,
  Amp,
  AmpError,
  NumOfHits,
  FitGoodnes);
```

The output of the FFTHitFinder reports back the Gaussian fits giving the start, end, and mean position of each Gaussian (with associated errors) as well as the charge represented by the Gaussian.

However, the “Number of Hits” and “FitGoodness” both seem to be variables with questionable output



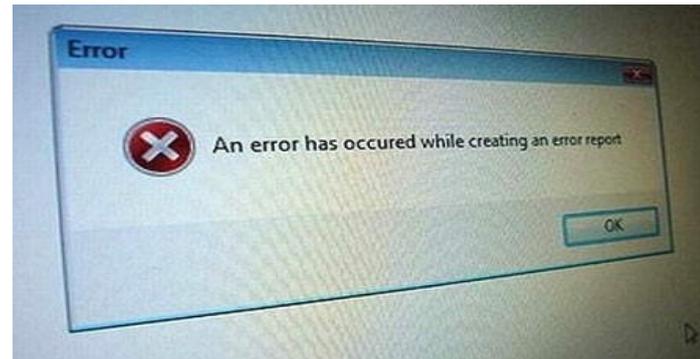
# FFTHitFinder “Problems”

- **Code was difficult to read and figure out what it was doing** (*minor issue, but leads to the next problem*)



- **Errors reported back on the hits didn't make sense** (*see next slide*)

- Made it difficult to evaluate if the algorithm was doing something reasonable



- **The multiplicity of the hit was hard coded to one for all hits**

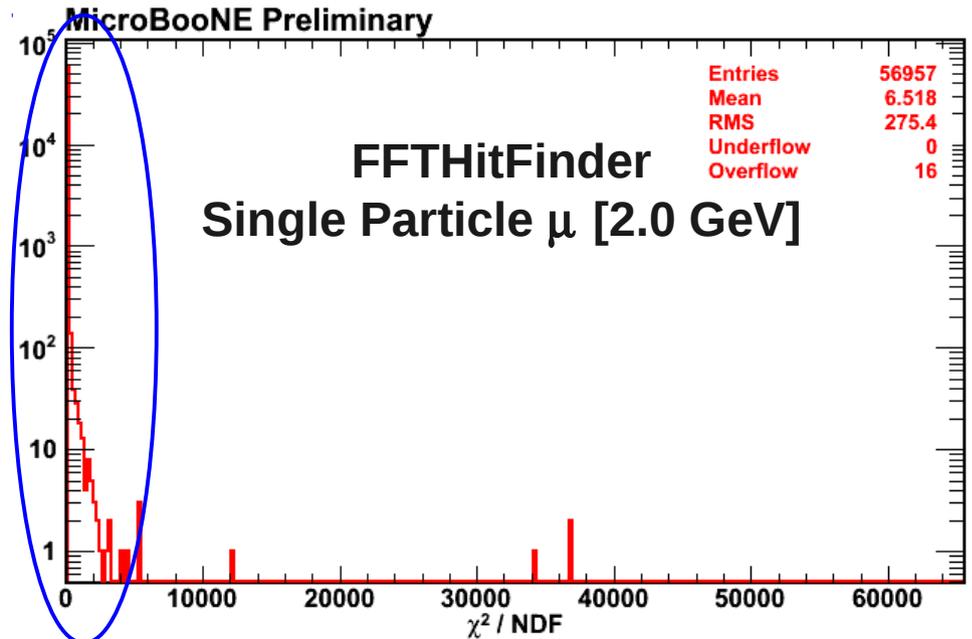
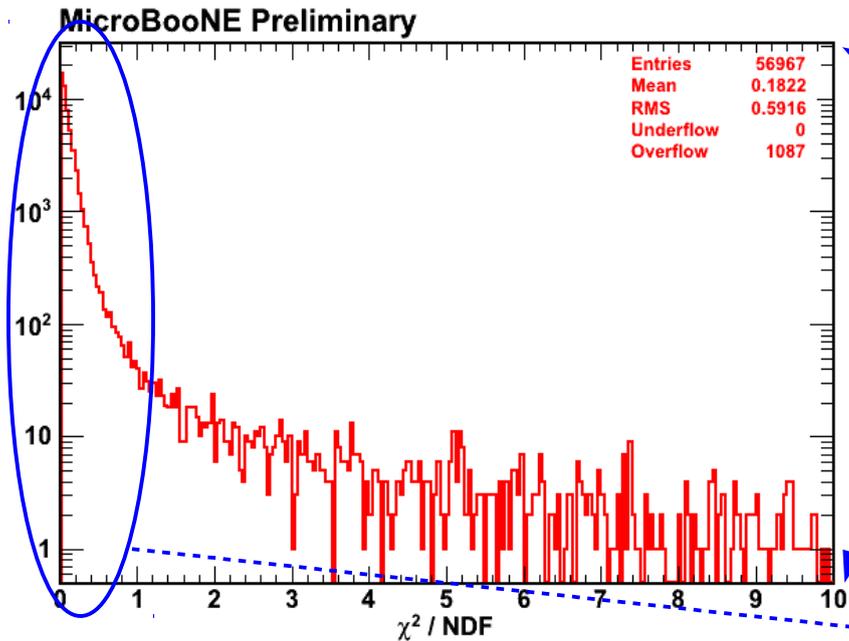
- Not entirely sure why this was since the ability to evaluate this seemed to be present in the code already

04/18/12

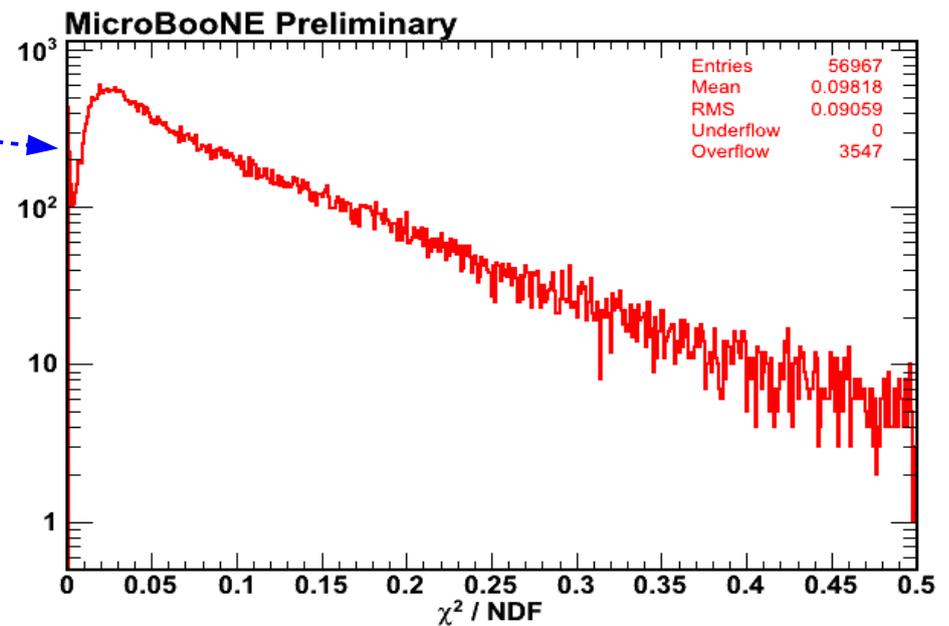


# FFTHitFinder “Problems”

Looking at the  $\chi^2 / \text{NDF}$  for the hits found the values range from 60,000 to much less than one!

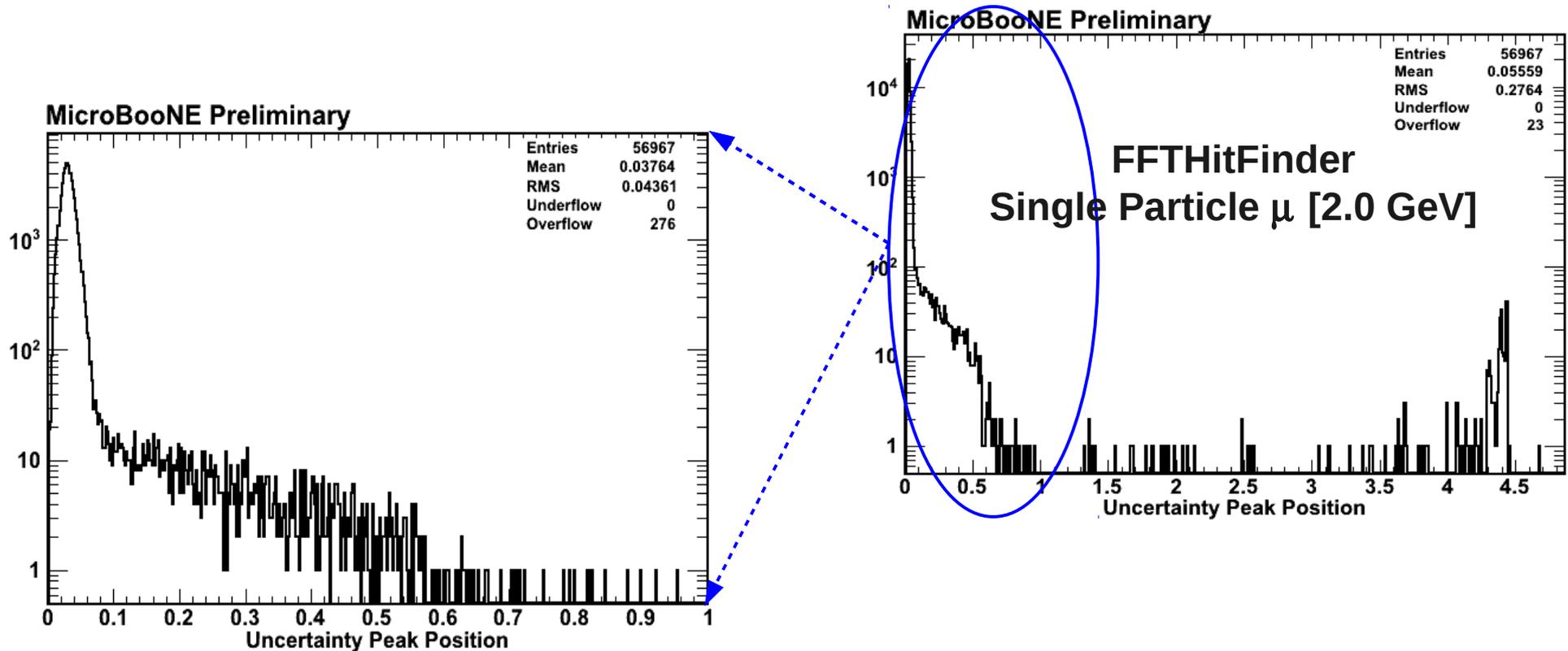


This seems to be an artifact of the way the fitting is done in the FFTHitFinder and makes evaluating these hits very difficult



*Note: FFTHitFinder lacks the ability to separate out single peak pulses and multi-peak pulses*

# FFTHitFinder “Problems”



The reported uncertainty in the peak position is reported to be much less than 0.1 for the overwhelming majority of hits

This seems “too good” for a set of fits that have no quality cuts on them...

**Note:** FFTHitFinder lacks the ability to separate out single peak pulses and multi-peak pulses

# “New” Hit Finding Algorithm (GausHitFinder)

**Keep all the good parts of the FFTHitFinder!**

- Finding local minima and maxima
  - Don't mess with a good thing
- **Keep the same interface and data members with the Hit Reco object**
  - Hits should look and interface exactly the same way as before

**Same output as the  
FFTHitFinder**



```
recob::Hit hit(wire,  
Starttime,  
StartTimeError,  
EndTime,  
EndTimeError,  
MeanPosition,  
MeanPosError,  
Charge,  
ChargeError,  
Amp,  
AmpError,  
NumOfHits,  
FitGoodnes);
```

# “New” Hit Finding Algorithm (GausHitFinder)

- **Change the fitting procedure slightly**  
*(more on this to come)*
  - Allows the fit to tell us if this is a “good” hit or not
- **Rewrite the code to be more “user friendly”**
  - Also make it easier to spot mistakes...
- **Propose to throw out obviously bad hits**
  - More to say on this later

# GausHitFinder Algorithm

LOOP OVER WIRES LOOKING AT PULSES FOR LOCAL MINIMA AND MAXIMA  
(same as in FFTHitFinder)

Split pulses found into “merged” and “unmerged” pulses  
(similar to the FFTHitFinder)

UNMERGED PULSES  
(Fit with a single Gaussian)

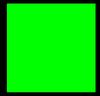
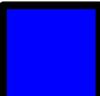
MERGED PULSES  
(Fit by multiple Gaussian)

1) If there is no gap between the end of the previous pulse and the start of the next

2) If the height of the minimum is greater than  $\frac{1}{2}$  the ADC threshold for hits

3) If the pulse is not at the end of the wire

4) If the number of consecutive pulses is less than the # of maximum consecutive hits (i.e. = 3 by default)

 = same as FFTHitFinder  
 = new to GausHitFinder

# GausHitFinder Algorithm

UNMERGED  
PULSES

(Fit with a single  
Gaussian)

MERGED PULSES

(Fit by multiple  
Gaussian)

More to  
say on  
this later...

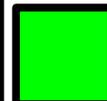
FIND THE "SEED" MEAN POSITION OF THE PULSE

(Fit a Gaussian around the local maxima of the pulse  
allowing the RMS, and normalization to vary  
unconstrained and the mean to be +/- 2 time ticks around  
the previously found maxima)

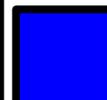
FIT A GAUSSIAN TO THE "HIT"  
FIXING THE MEAN TO THE SEED  
POSITION

(We require the fit normalization to  
be  $> \frac{1}{2}$  the threshold and the fit RMS  
 $>$  minimum width)

WRITE OUT  
RECO:HIT



= same as FFTHitFinder



= new to GausHitFinder

# GausHitFinder Algorithm

UNMERGED  
PULSES

(Fit with a single  
Gaussian)

MERGED PULSES

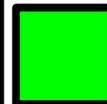
(Fit by multiple  
Gaussian)

DETERMINE THE MULTIPLICITY OF THE PULSE

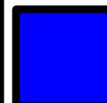
(Based on how many pulses were merged into a single pulse)

FIND THE "SEED" MEAN POSITION OF EACH PEAK IN  
THE PULSE

(Fit a Gaussian around the local maxima of the pulse  
allowing the RMS, and normalization to vary  
unconstrained and the mean to be +/- 2 time ticks around  
the previously found maxima)



= same as FFTHitFinder



= new to GausHitFinder

# GausHitFinder Algorithm

CONTINUED FROM LAST SLIDE...

FIT A GAUSSIAN TO EACH "HIT" FIXING THE MEAN TO THE SEED POSITION OF EACH PEAK

*(We require the fit normalization to be  $> \frac{1}{2}$  the threshold and the fit RMS  $>$  minimum width)*

WRITE OUT RECO:HIT

**Note: We calculate the charge using the amplitude of the Gaussian (not using the area method present in the FFTHitFinder)**

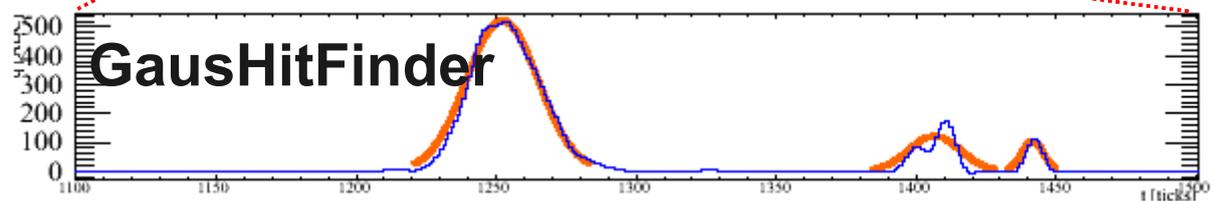
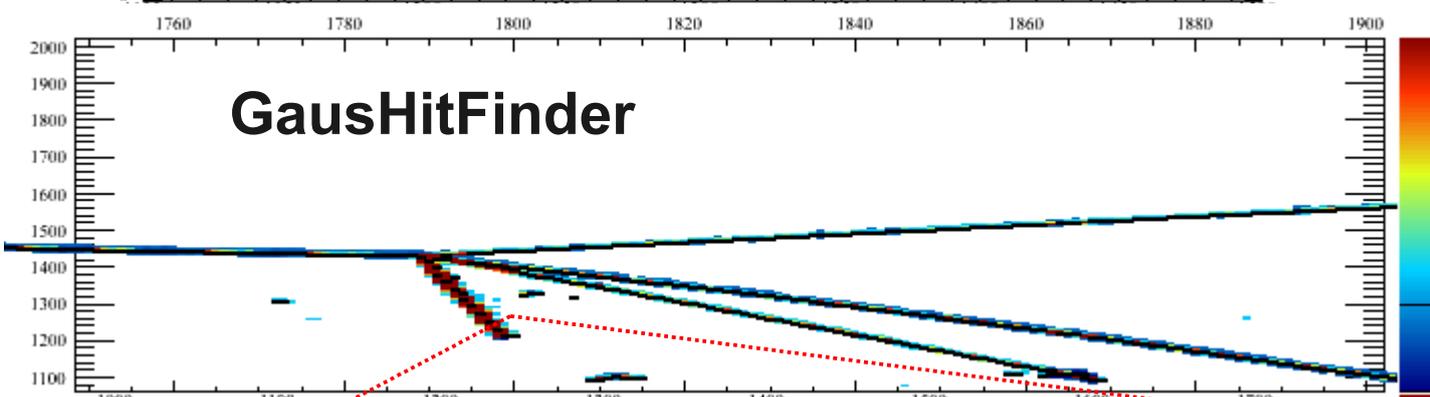
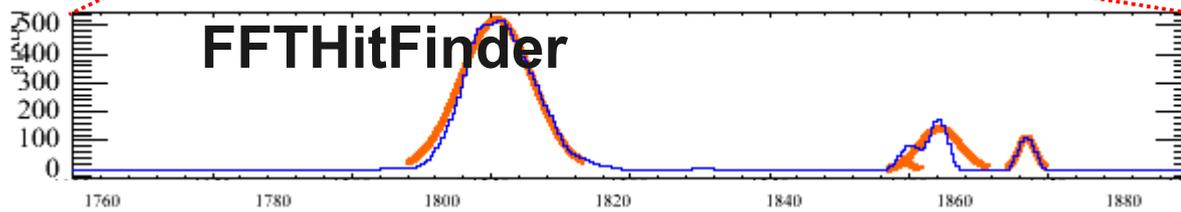
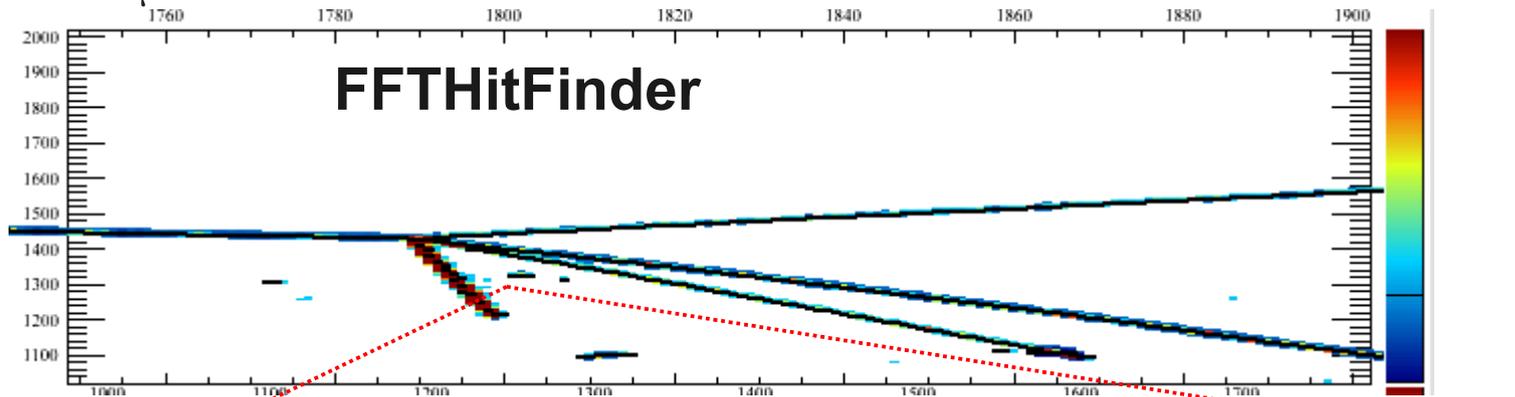
**This could be modified to use a multi-Gaussian fit function for the charge... (still being considered)**

**Note: No rejection of hits based on  $\chi^2/\text{NDF}$  is implemented yet**

**However, I would like to put this up for discussion**

# Results of the GausHitFinder Algorithm

$$\nu_{\mu} [4.4 \text{ GeV}] + Ar \rightarrow \mu [2.6 \text{ GeV}] + 3 \text{ protons} + \pi + n' s$$



For simple events...  
very little  
difference in the  
nature of the  
hits found

# Results of the GausHitFinder Algorithm

Single Particle  $\mu$  [2.0 GeV]

Event #	# of Hits Found (GausHitFinder)	# of Hits Found (FFTHitFinder)
1	5502	5528
2	5786	5824
3	5803	5838
4	5531	5556
5	5476	5550
6	5744	5771
7	5622	5717
8	5711	5746
9	5643	5672
10	5782	5805
<b>Totals</b>	<b>56600</b>	<b>57007</b>



**GausHitFinder finds 99.3 % of the same hits as FFTHitFinder**

→ *This 0.7 % difference comes entirely to how we handle multi-peaked pulses  
(still looking into this further)*

# Results of the GausHitFinder Algorithm

Single Particle  $\mu$  [2.0 GeV]

What is the time performance of the two algorithms?



## FFTHitFinder

*(Running over 10 single muon events)*

Avg. Time  
~ 14 seconds per event

## GausHitFinder

*(Running over 10 single muon events)*

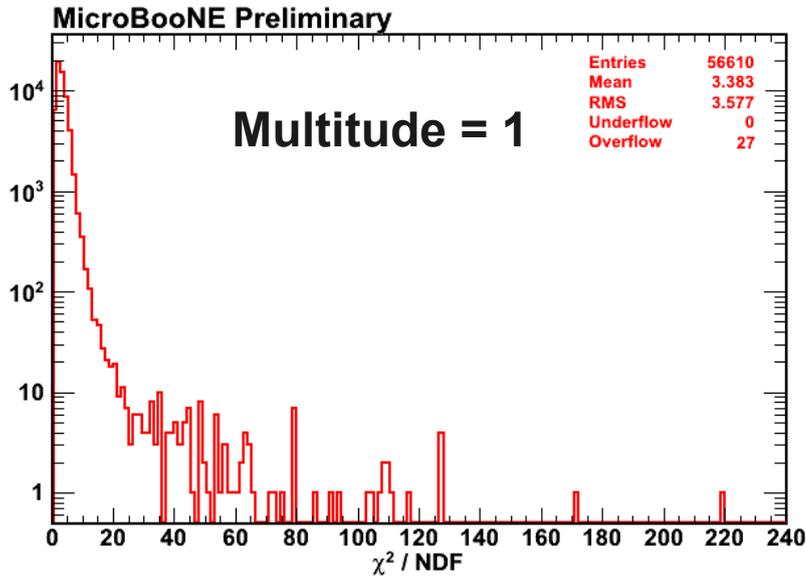
Avg. Time  
~ 32 seconds per event



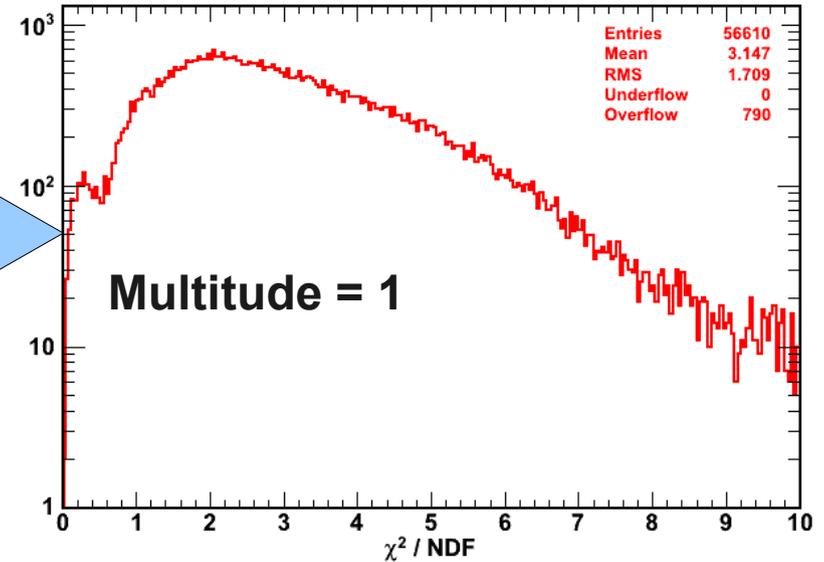
*A difference in performance time...but could still be improved (work in progress)*

# Results of the GausHitFinder Algorithm

Single Particle  $\mu$  [2.0 GeV]



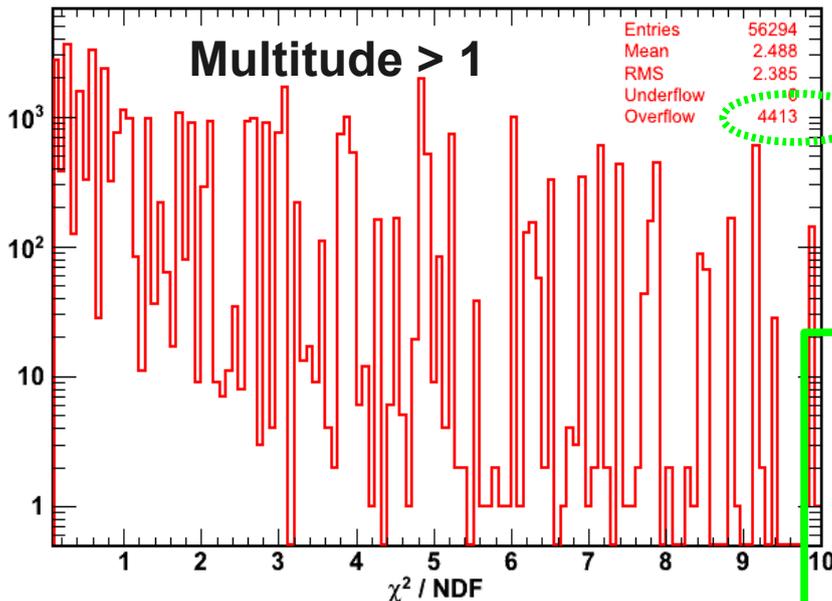
zoom in



$\chi^2 / \text{NDF}$  values make much more sense for the fits performed

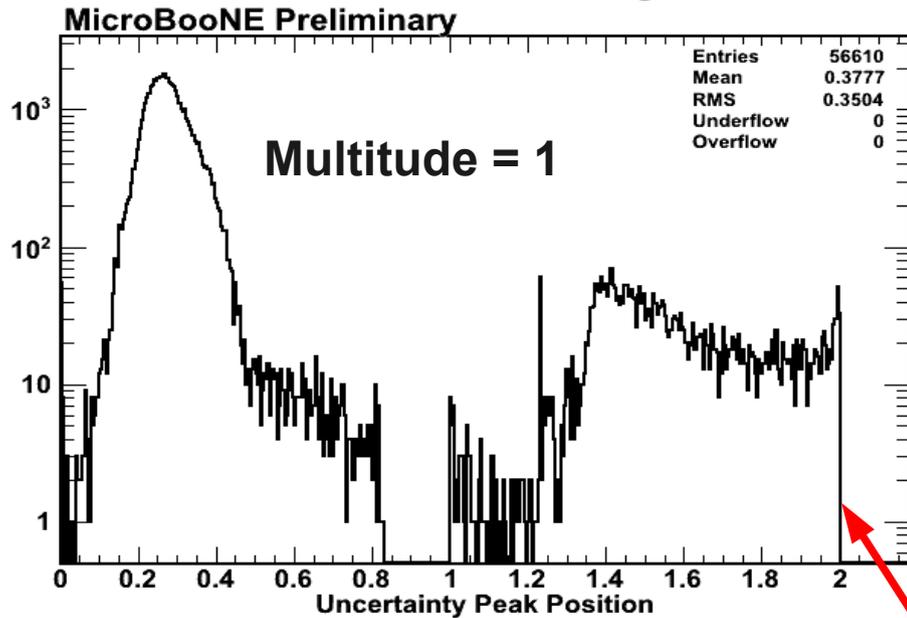
Possibly suggests having a loose cut to throw out large  $\chi^2 / \text{NDF}$  (Require  $\chi^2 / \text{NDF} < 10$  or  $20$ ?)

Investigating why this has an overflow...something goofy with my GausHitFinderAna

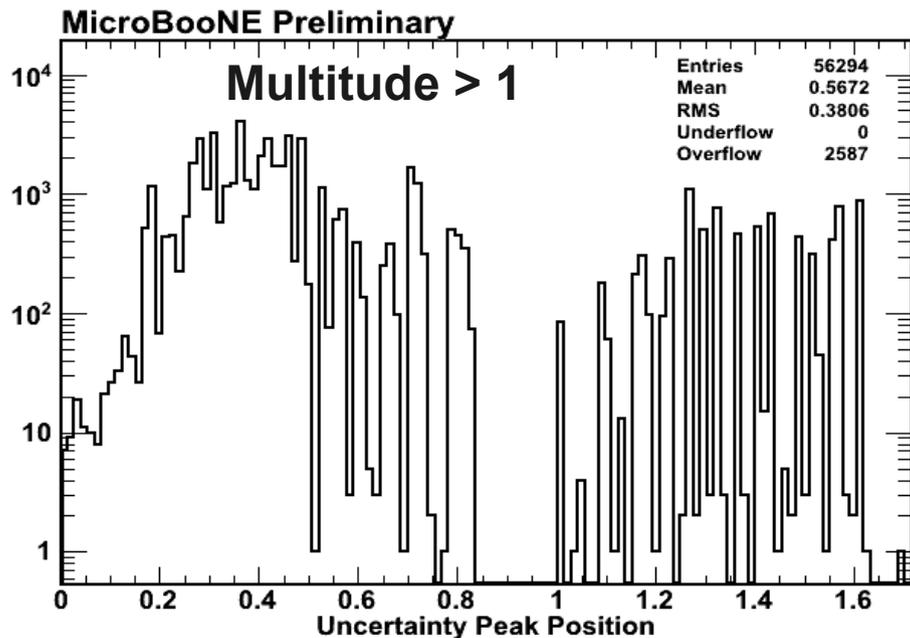


# Results of the GausHitFinder Algorithm

Single Particle  $\mu$  [2.0 GeV]



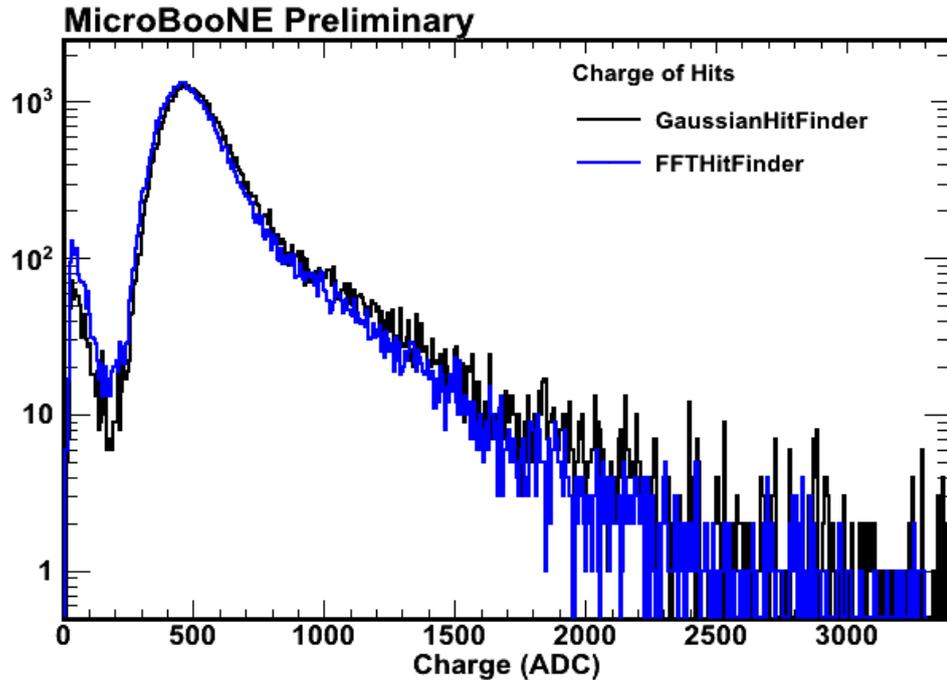
The reported uncertainty in the peak position of the Gaussian remains small but now has a value that seems more reasonable and is itself Gaussian (more like what I expected)



The "hard edge" at 2 is a "feature" of how I find the peak of the hit (mean of the Gaussian) to begin with and can be relaxed to allow a broader seed range of positions.

# Results of the GausHitFinder Algorithm

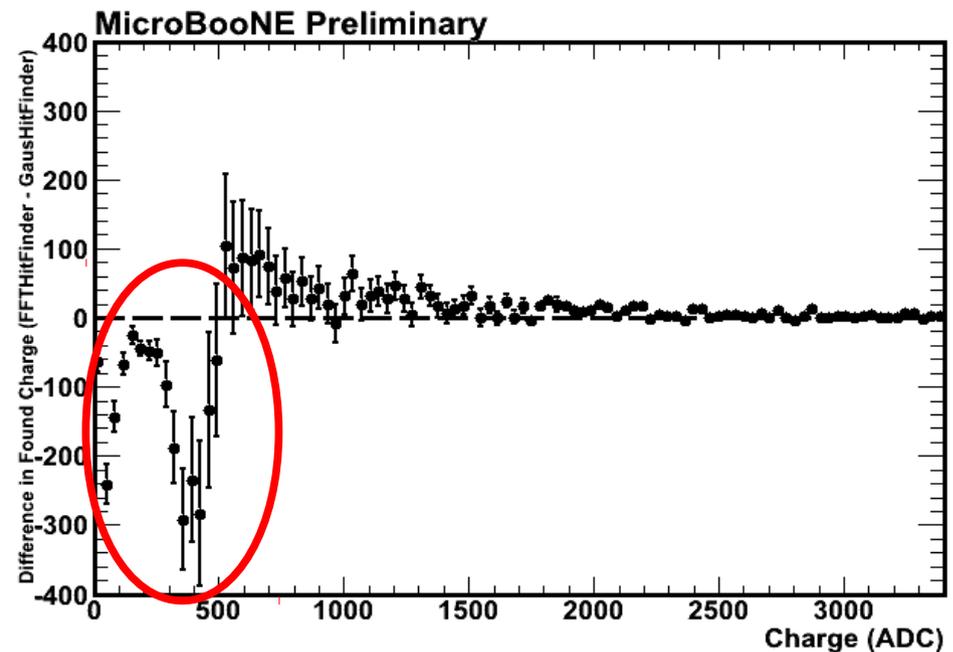
Single Particle  $\mu$  [2.0 GeV]



Looking at the charge found by the two algorithms

Mimics the results of the FFTHitfinder for most hits

Need to understand if it is a good or bad thing that the GausHitFinder finds more charge than FFTHitFinder



# Genie Events

Also looked at a series of Genie events to compare the GausHitFinder to the FFTHitFinder

- Performance is a little “worse” (finds fewer of the same hits)
- This seems to have to do with the multi-peak hits in Genie events
- There is also some weird “undershoot” that seems to be present in some Genie events (more to say in the coming slides)



**See Back-up Slides for  
further details**

# Weirdness / Things I don't understand yet....



# Overview of FFTHitFinder

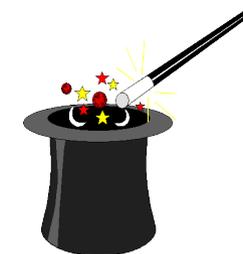
*(a little more detail)*

```
if(numHits > 1) {
    TArrayD data(numHits*numHits);
    TVectorD amps(numHits);
    for(int i = 0; i < numHits; ++i) {
        amps[i] = signal[maxTimes[hitIndex+i]];
        for(int j = 0; j < numHits;j++)
            data[i+numHits*j] = TMath::Gaus(maxTimes[hitIndex+j],
                                             maxTimes[hitIndex+i],
                                             fitWidth);
    }//end loop over hits

    //This section uses a linear approximation in order to get
    //initial value of the individual hit amplitudes
    try{
        TMatrixD h(numHits,numHits);
        h.Use(numHits,numHits,data.GetArray());
        TDecompSVD a(h);
        a.Solve(amps);
    }
    catch(...){
        mf::LogInfo("FFTHitFinder")<<"TDcompSVD failed";
        hitIndex += numHits;
        continue;
    }
}
```

There is some other “magic” going on in this code in the case of multiple “pulses” found within a single “hit” ...

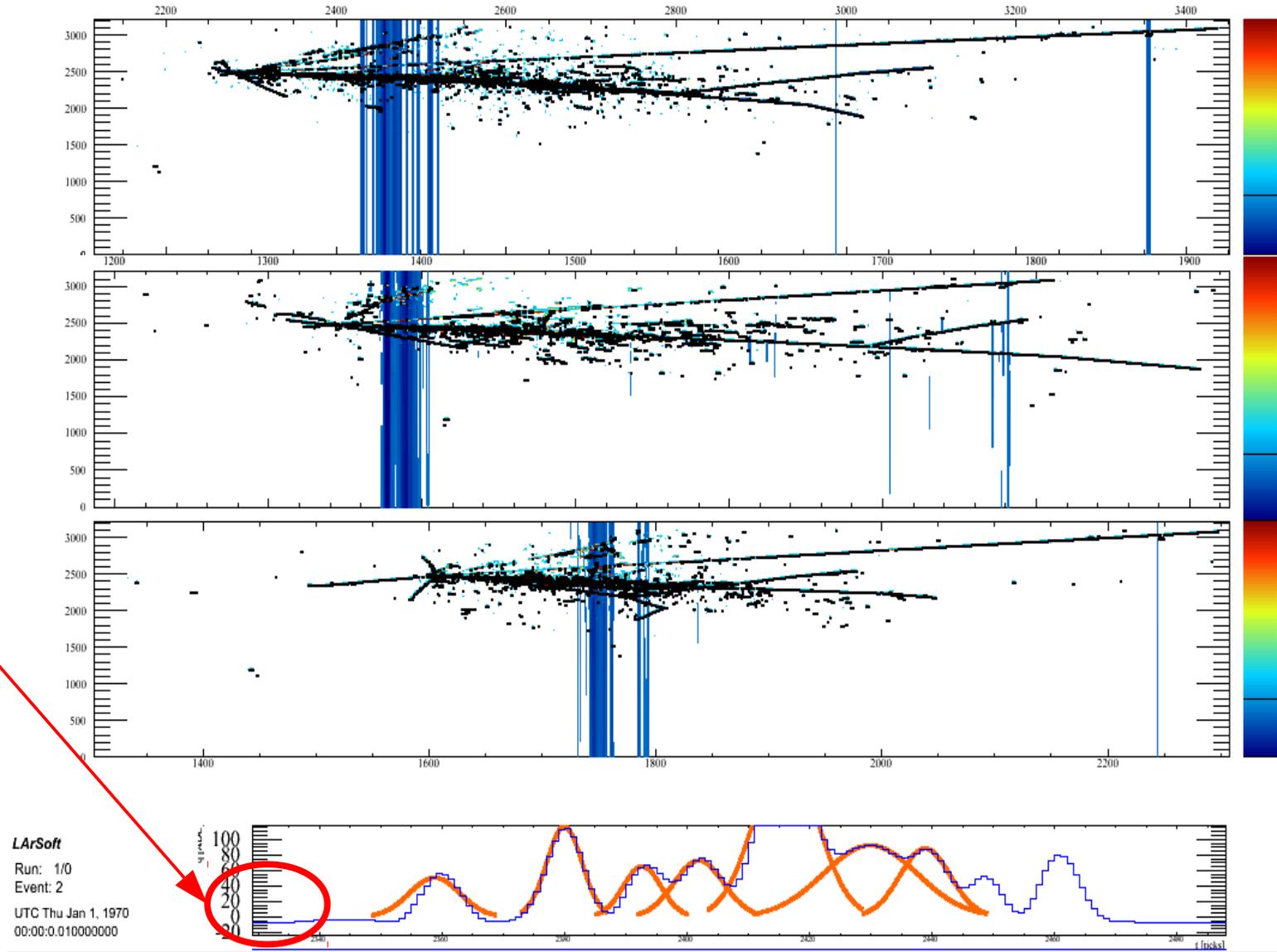
I didn't understand how this helped in finding the amplitudes



In very “busy” events generated in Genie after reconstruction there are wires where the ADC baseline is less than 0

This looks like a long and drawn out undershoot

Causes some events to fall “below threshold”



$$\nu_{\mu} [26.2 \text{ GeV}] + Ar \rightarrow \mu [8.8 \text{ GeV}] + \pi^0 [4.4 \text{ GeV}] + \pi [2.5 \text{ GeV}] + \dots$$

# Conclusions

Have introduced a new HitFinding algorithm (GausHitFinder)

→ Based on the FFTHitFinder

Attempts to “fix” some of the problems found with the FFTHitFinder

→ Allow us to evaluate the quality of the hits for use in later reconstruction algorithms

GausHitFinder shown to reproduce the majority of the FFTHitFinder on first simple checks

# Next Steps

- Check in GausHitFinder for others to use
- Finish exploring discrepancies between FFTHitFinder and GausHitFinder and determine necessary “fixes”
  - Difference in # of hits
  - Difference in charge found
- Need to determine the maximum hit multiplicity we can reasonably assume we should be able to distinguish (how many peaks in a single pulse) and allow GausHitFinder to reflect this
- Add the multi-Gaussian fit to calculate the charge from the area of the fit as well as the amplitude
  - Not currently implemented
- Add a loose  $\chi^2$  / NDF to reject poorly reconstructed hits



# Backup Slides

# Results of the GausHitFinder Algorithm

## Genie Events

Event #	# of Hits Found (GausHitFinder)	# of Hits Found (FFTHitFinder)
1	3773	3869
2	11320	14904
3	3522	3845
4	0	0
5	7873	8295
6	2741	2767
7	2676	2777
8	0	0
9	7038	8640
10	4390	4479
<b>Totals</b>	<b>43233</b>	<b>49576</b>

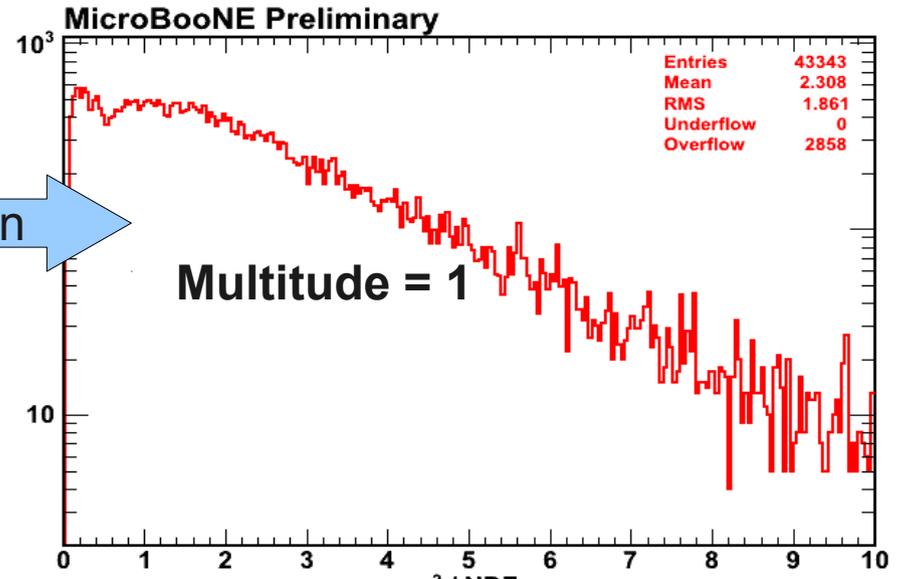
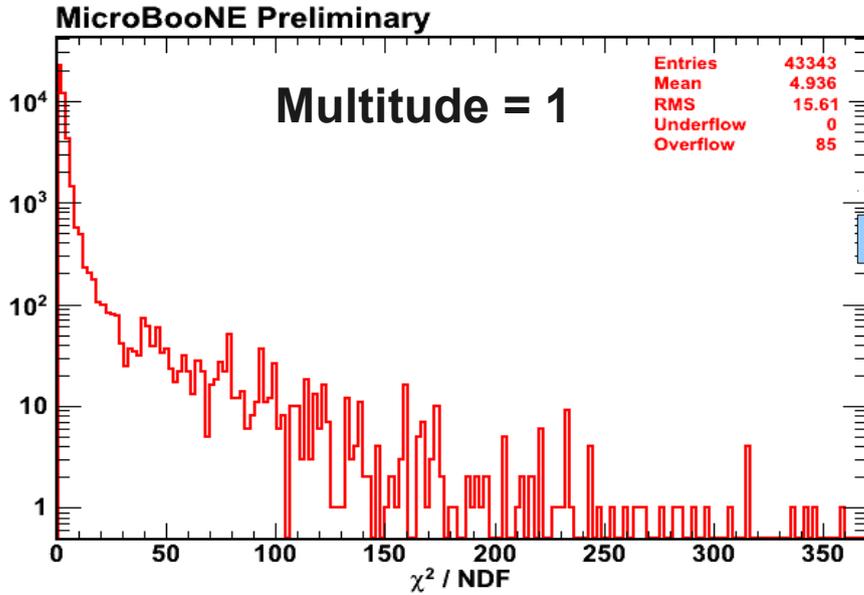


**GausHitFinder finds 87.4 % of the same hits as FFTHitFinder**

→ *This 12.6 % difference comes from a purponderance of different effect that I am still exploring*

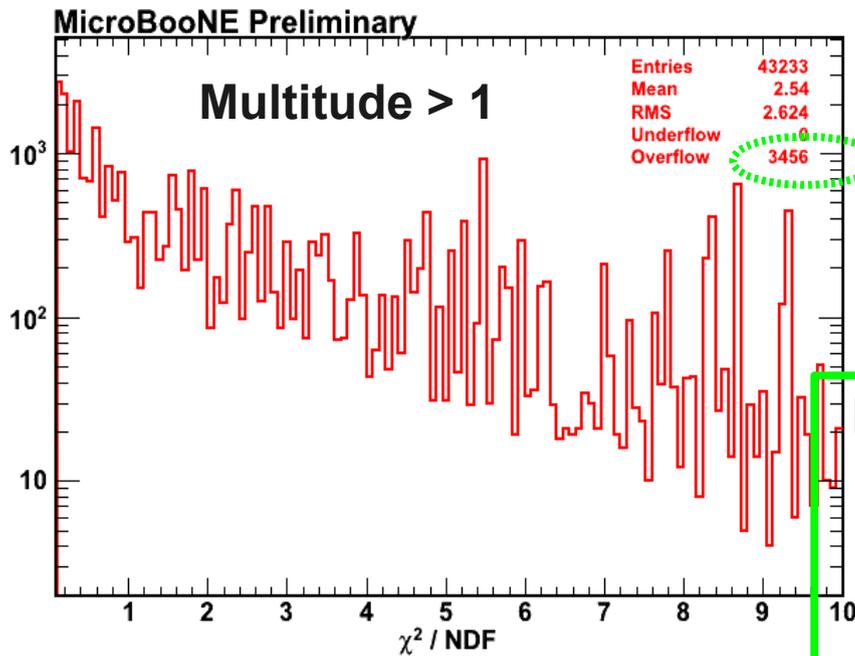
# Results of the GausHitFinder Algorithm

## Genie Events



$\chi^2 / \text{NDF}$  values make much more sense for the fits performed

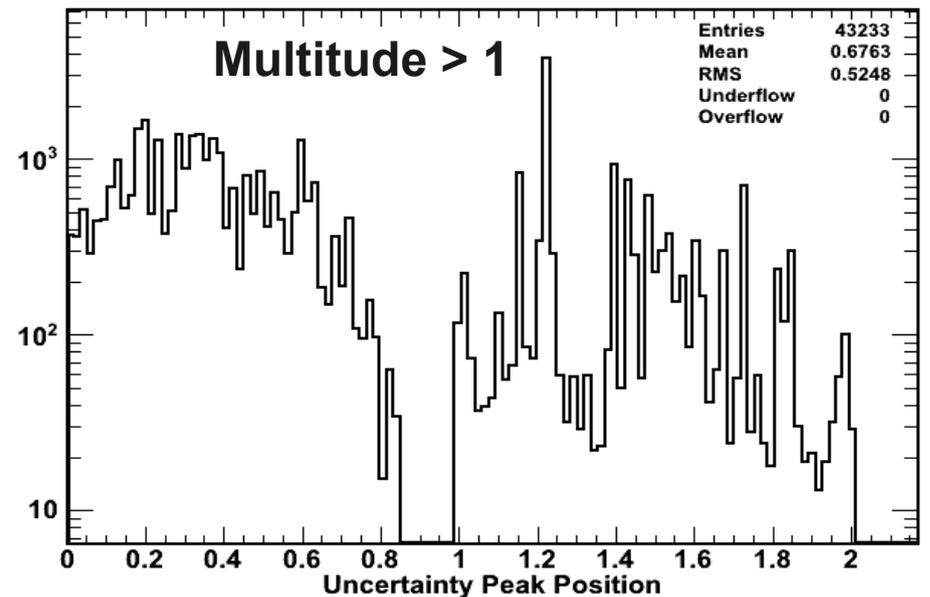
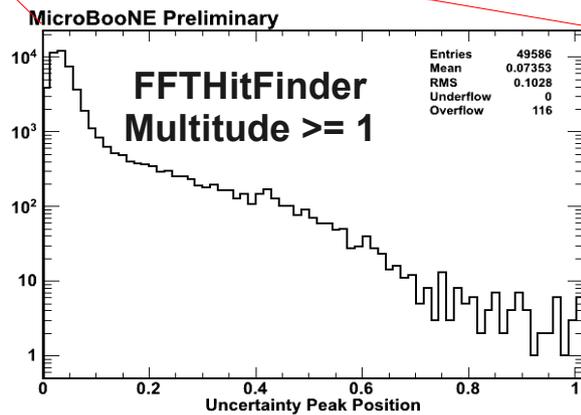
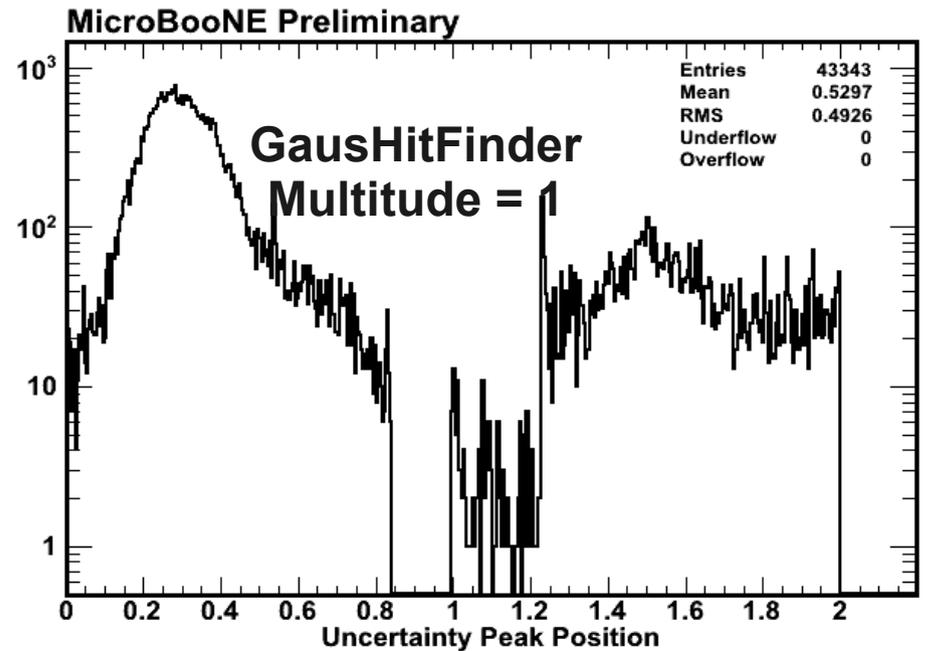
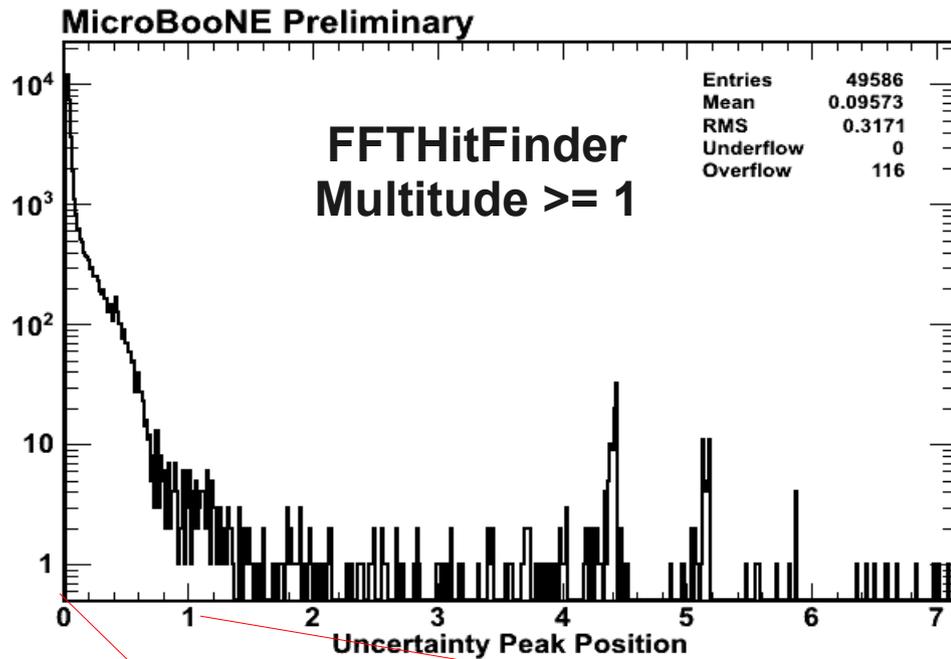
Possibly suggests having a loose cut to throw out large  $\chi^2 / \text{NDF}$  (Require  $\chi^2 / \text{NDF} < 10$  or  $20$ ?)



Investigating why this has an overflow...something goofy with my GausHitFinderAna

# Results of the GausHitFinder Algorithm

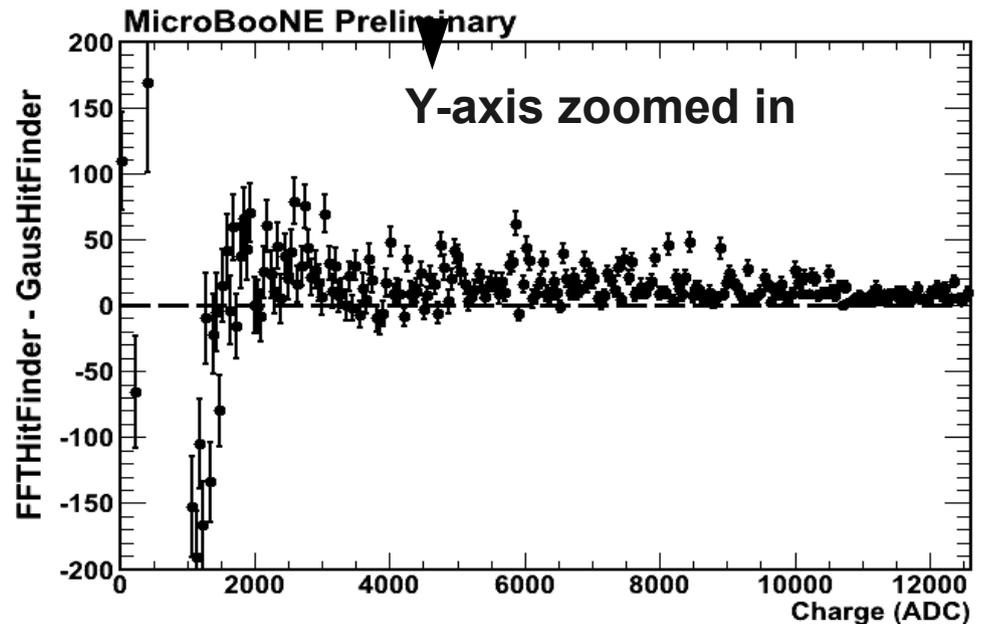
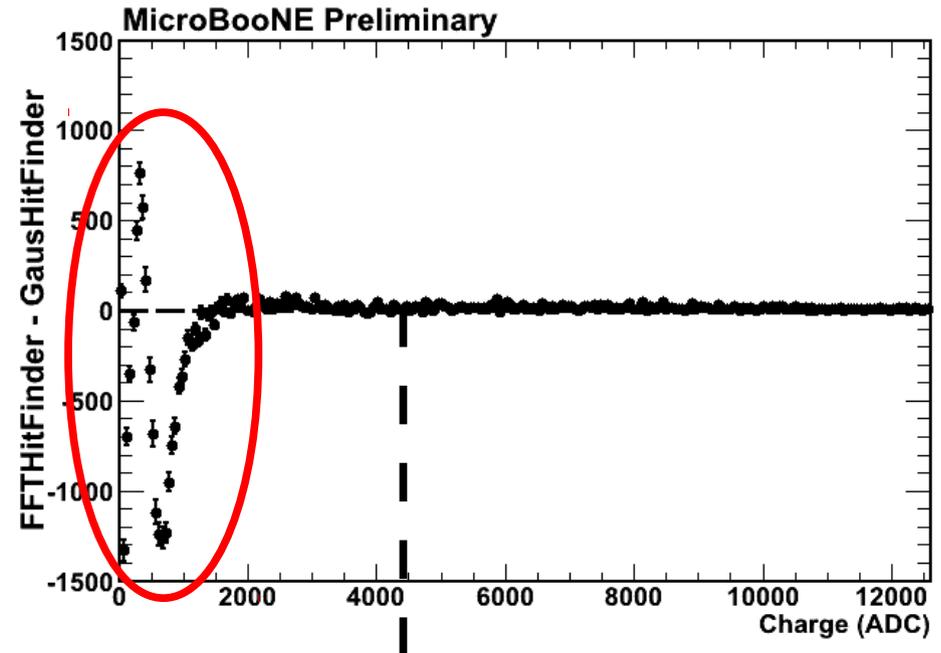
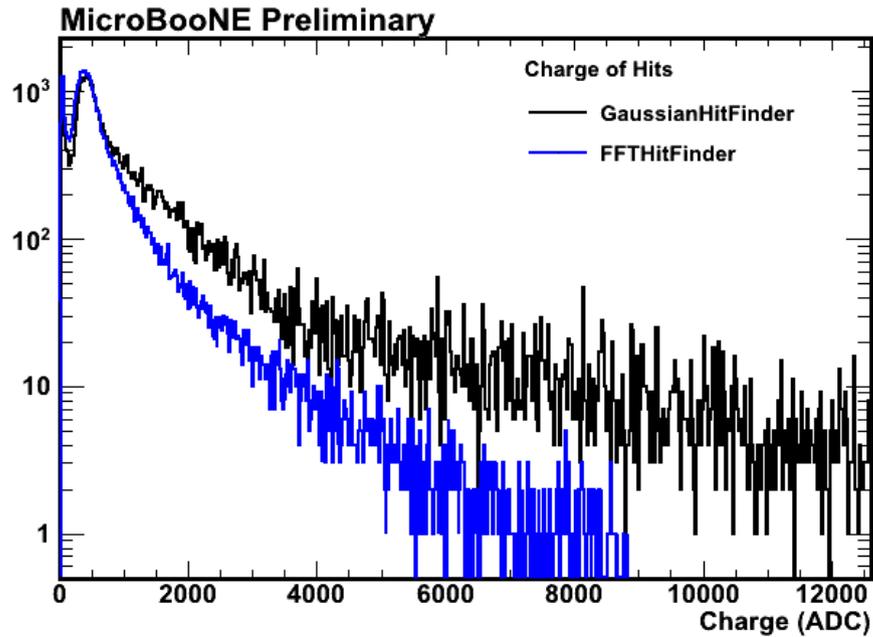
## Genie Events



**Note:** FFTHitFinder lacks the ability to separate out single peak pulses and multi-peak pulses

# Results of the GausHitFinder Algorithm

## Genie Events



Need to understand if it is a good or bad thing that the GausHitFinder finds more charge than FFTHitFinder