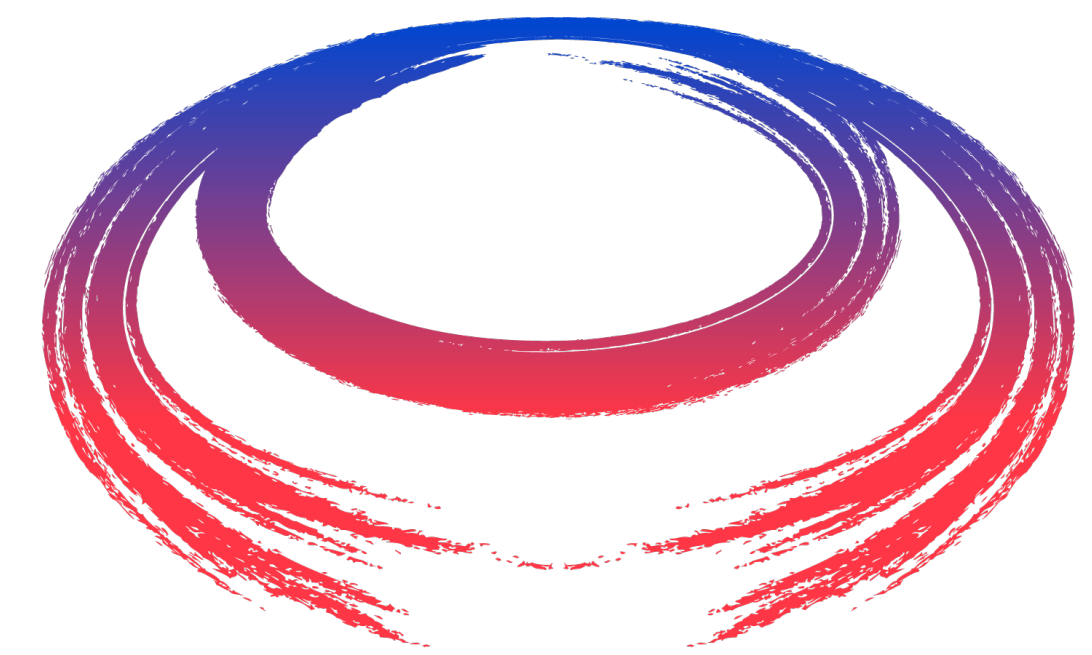


# Muon Collider Multiple Scattering & Realistic Timing Resolution Implementation



Juliet Wright, Angira Rastogi, Simone Pagan Griso, and Laura Jeanty



# Topics

- Sensor Thickness Studies Motivation
- Realistic Timing Resolution Implementation
  - Motivation
  - Application
  - Results
- Multiple Scattering Implementation
  - Motivation
  - Strategy
  - Results

# Motivation

- We want to improve signal pixel cluster selection efficiency by enlarging the signal to beam induced background (BIB) separation.
- The signal to BIB separation in energy deposition can possibly be improved with an increased sensor thickness, but timing resolution worsens.
- Signal can be improved because larger sensors will have better signal-to-noise ratios which can help improve separation in the energy deposition.
- We test this by modifying the MAIA detector geometry in the vertex barrel (VXB)
- Preliminary results from Angira's presentation [here](#)
- This study looks at the signal vs BIB separation for five different VXB sensor thicknesses:
  1.  $50\ \mu\text{m}$  (nominal sensor thickness)
  2.  $75\ \mu\text{m}$
  3.  $100\ \mu\text{m}$
  4.  $200\ \mu\text{m}$
  5.  $400\ \mu\text{m}$

# Simulation Technical Details

- A [particle gun](#) is used to generate 100k 10GeV muon events. No other constraints are specified.
- The generated muons are simulated with the [steer\\_baseline.py](#) code. [This line](#) is edited for the particular detector geometry used.
- Only 1k events are digitized for each VXB sensor thickness using the [digi\\_steer\\_MAI\\_Avo.py](#) code.
- We use the [realistic digitization code](#), so the pixel hits/cluster information is available
- BIB simulation for different sensor thicknesses is produced from FLUKA output converted to LCIO files for muplus and muminus beams using this [script](#)
- 1 BIB event is digitized for all sensor thickness studies
- BIB from FLUKA output was simulated five times for each sensor thickness in [geometry file](#).
- Finally, we run realistic digitization overlaying the same signal process to produce 1 event.

# Realistic Timing Resolution Motivation

- The MAIA detector geometry uses planar sensors, so with increasing sensor thickness we would expect the timing resolution to worsen
- The current timing calculation for hits is based on true time smeared according to gaussian template with  $50\mu\text{m}$  sensor timing resolution
- As sensor thickness increases, the timing resolution will improve which goes against our expectations for planar sensors
- This motivates adding in new realistic timing resolution

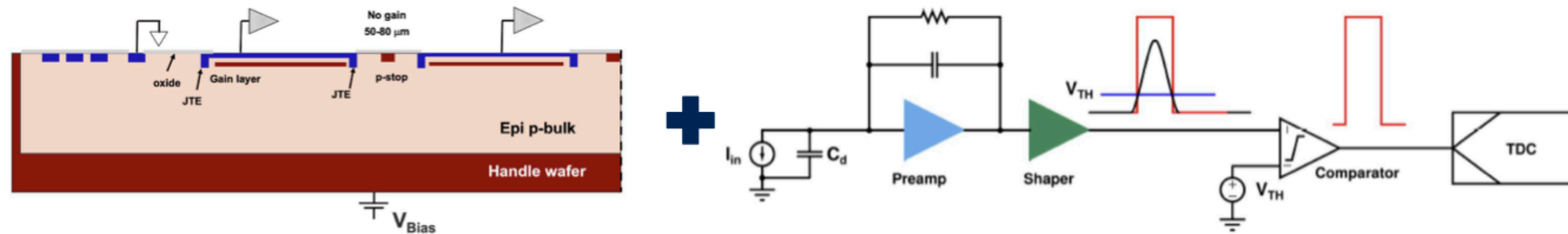
```
---- ..
1041 void MuonCVXDDigitiser::TimeSmearer(SimTrackerHitImplVec &simTrkVec)|
1042 {
1043     streamlog_out (DEBUG6) << "Adding resolution effect to timing measurements" << std::endl;
1044     for (int i = 0; i < (int)simTrkVec.size(); ++i)
1045     {
1046         float delta = RandGauss::shoot(0., _timeSmearingSigma);
1047         SimTrackerHitImpl *hit = simTrkVec[i];
1048         hit->setTime(hit->getTime() + delta);
1049         streamlog_out (DEBUG4) << i << ": x=" << hit->getPosition()[0] << ", y=" << hit->getPosition()[1] << ", z=" << hit->getPosition()[2]
1050             << ", time = " << hit->getTime() << "(delta = " << delta << ")" << std::endl;
1051     }
1052 }
```

# Implementation of realistic timing resolution

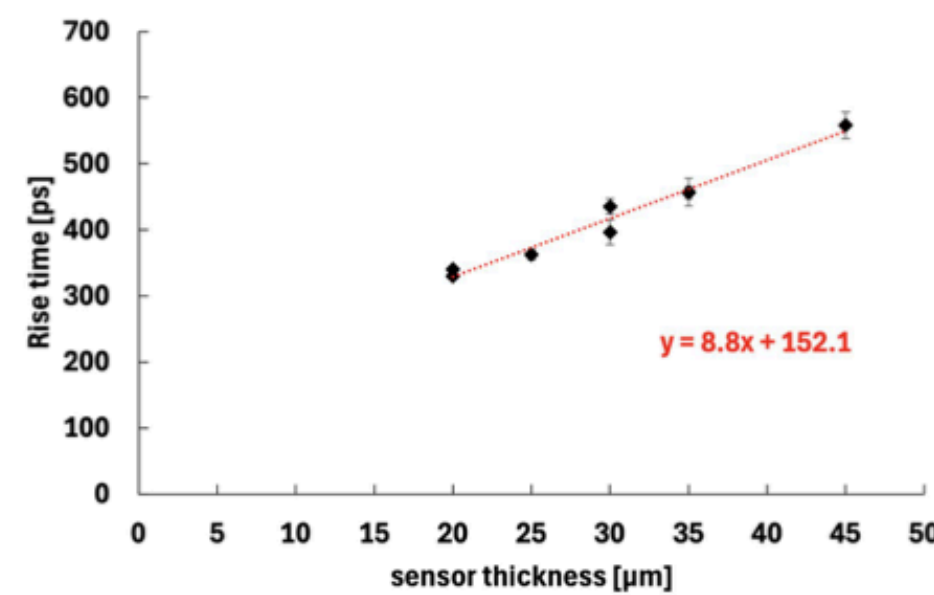
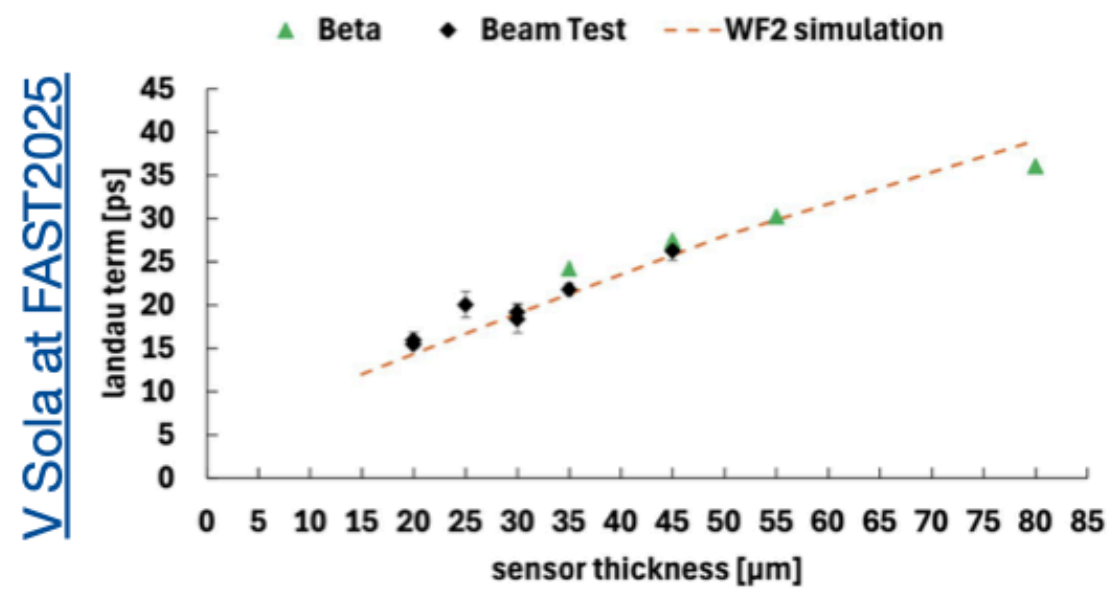
Using information from [this presentation](#)

## Fundamentals of a Timing Detector

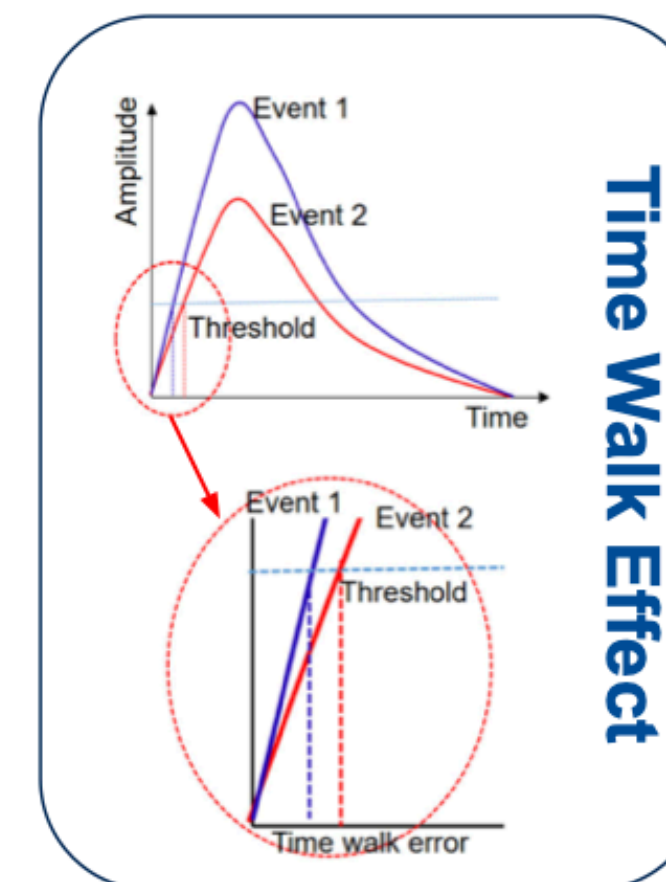
Standard "Hybrid" Pixel Model - Basics of Time Resolution



$$\sigma_t^2 = \sigma_{Landau}^2 + \sigma_{timewalk}^2 + \sigma_{jitter}^2 + \sigma_{TDC}^2 + \sigma_{clock}^2$$



$$\frac{t_{rise}}{S/N}$$



# Implement Realistic Timing Resolution

- For realistic timing resolution, we can apply this formula:

$$\sigma_{\text{total}} = \sqrt{\sigma_{\text{Landau}}^2 + \sigma_{\text{timewalk}}^2 + \sigma_{\text{jitter}}^2 + \sigma_{\text{TDC}}^2 + \sigma_{\text{clock}}^2}$$

- $\sigma_{\text{Landau}} = 30[\text{ps}] \times \text{sensor thickness}/50[\mu\text{m}]$
- $\sigma_{\text{timewalk}} = 0.1 \times t_{\text{rise}}$
- $t_{\text{rise}} = (8.8 \times \text{sensor thickness} + 152.1)[\text{ns}]$  ( $t_{\text{rise}}$  is the time it takes for the pulse to go from **10% to 90% of its maximum amplitude**)
- $\sigma_{\text{jitter}} = \text{electronic noise} \times t_{\text{rise}}/(\text{signal amplitude}) = 80[e] \times t_{\text{rise}}[\text{ns}]/(80000[e/\text{mm}] \times \text{sensor thickness}[\text{mm}])$
- $\sigma_{\text{TDC}} = \Delta t/\sqrt{12}$  (time to digital converter where  $\Delta t = 25[\text{ps}]$ )
- $\sigma_{\text{clock}} = 5[\text{ps}]$

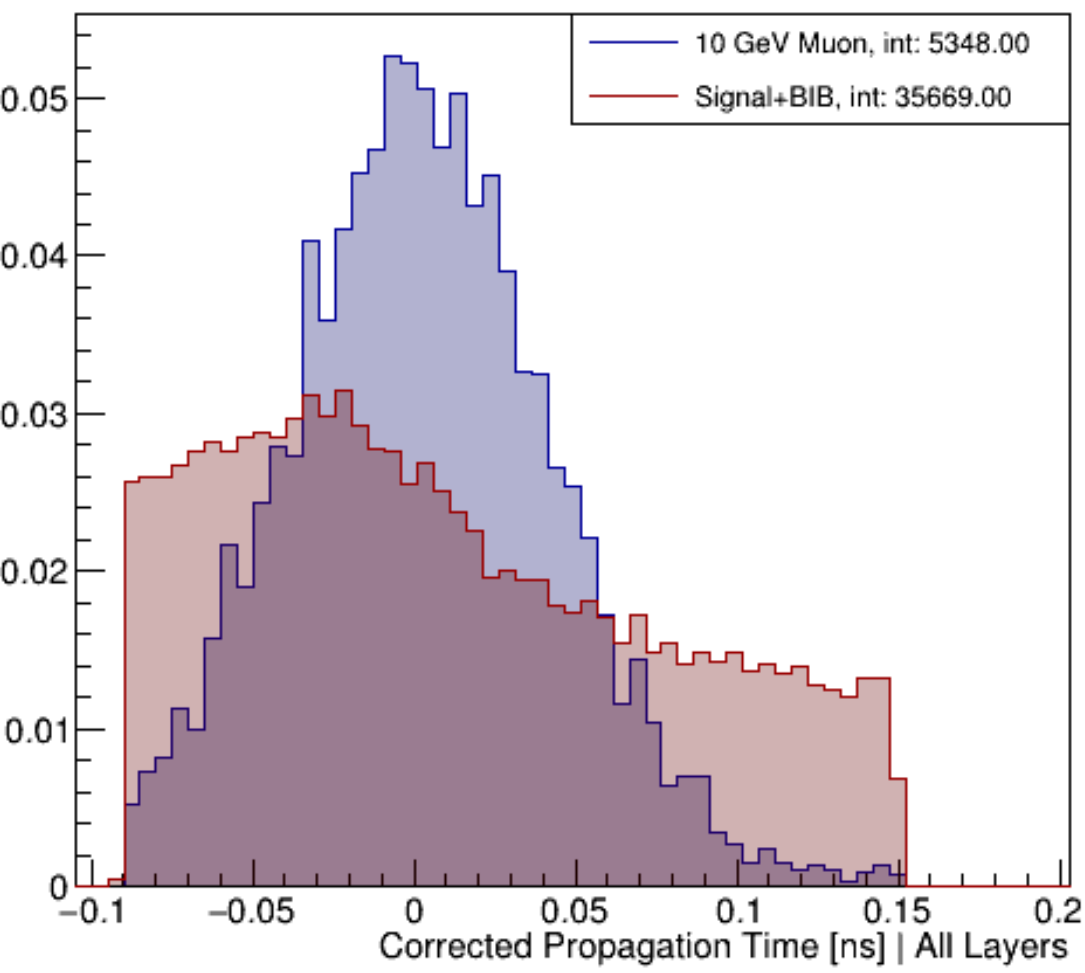
# Timing Resolution Updated Code

- Applying this equation to the digitization code in [Forked branch, mcUpdate](#)

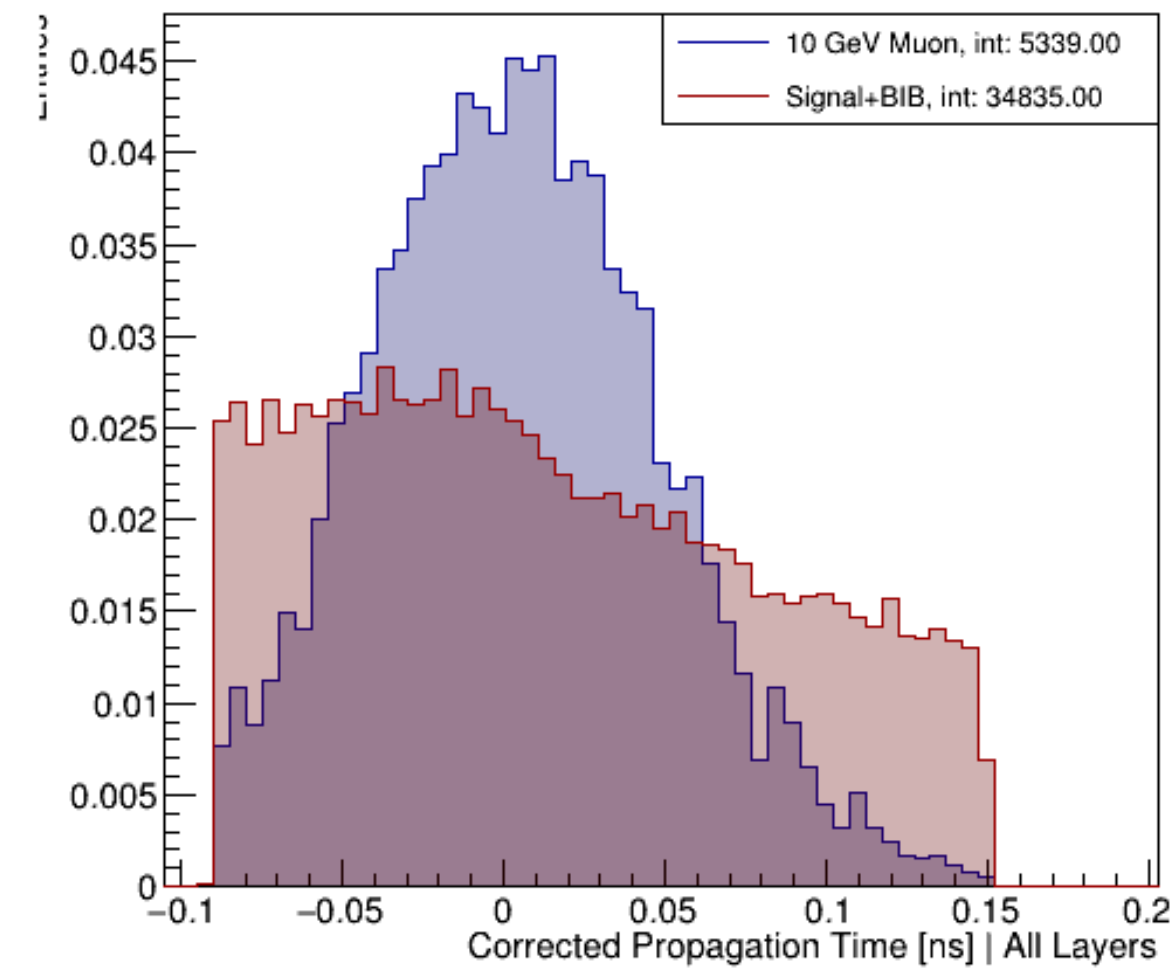
```
1142  void MuonCVXDDigitiser::TimeSmearer(SimTrackerHitImplVec &simTrkVec)
1143  {
1144      streamlog_out (DEBUG6) << "Adding resolution effect to timing measurements" << std::endl;
1145      for (int i = 0; i < (int)simTrkVec.size(); ++i)
1146      {
1147          //make timing more realistic:
1148          SimTrackerHitImpl *hit = simTrkVec[i];
1149          float t_rise = (8.8*_layerThickness[_currentLayer]*1e3 + 152.1)*1e-3; //[ns]
1150          float sigma_landau = 0.03 * _layerThickness[_currentLayer]/0.05; // [ns]from sensor thickness & charge deposition fluctuations - 30ps/50microns
1151          float sigma_timewalk = 0.1 * t_rise; //[ns] t_rise * 0.1
1152          float sigma_jitter = (_electronicNoise*t_rise)/(80000 * _layerThickness[_currentLayer]); // [ns] Q_noise/slope in charge over time 80e/micron = 80000e/mm
1153          float sigma_TDC = 0.025/std::sqrt(12); // time to digital converter using 25 ns for deltaT
1154          float sigma_clock = 0.005; // fixed by clock quality 5ps
1155          float sigma_total = std::sqrt(sigma_landau*sigma_landau + sigma_timewalk*sigma_timewalk + sigma_jitter*sigma_jitter + sigma_TDC*sigma_TDC + sigma_clock*sigma_clock);
1156          float delta = RandGauss::shoot(0., sigma_total);//(0., _timeSmearingSigma);
1157          //std::cout << "sigma_total: " << sigma_total << std::endl;
1158          hit->setTime(hit->getTime() + delta);
1159          streamlog_out (DEBUG4) << i << ": x=" << hit->getPosition()[0] << ", y=" << hit->getPosition()[1] << ", z=" << hit->getPosition()[2]
1160              << ", time = " << hit->getTime() << "(delta = " << delta << ")" << std::endl;
1161      }
1162  }
1163  /**
```

# Results

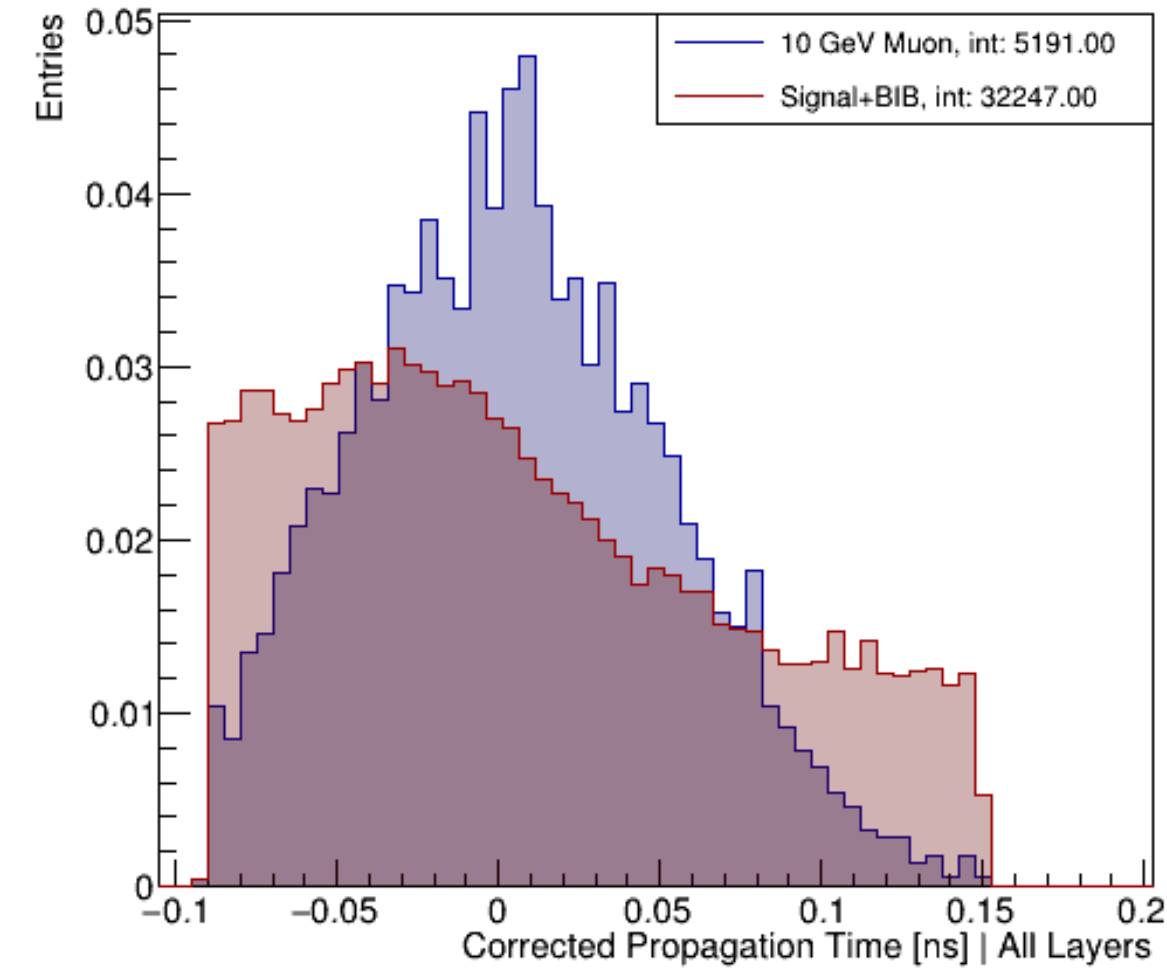
50  $\mu\text{m}$



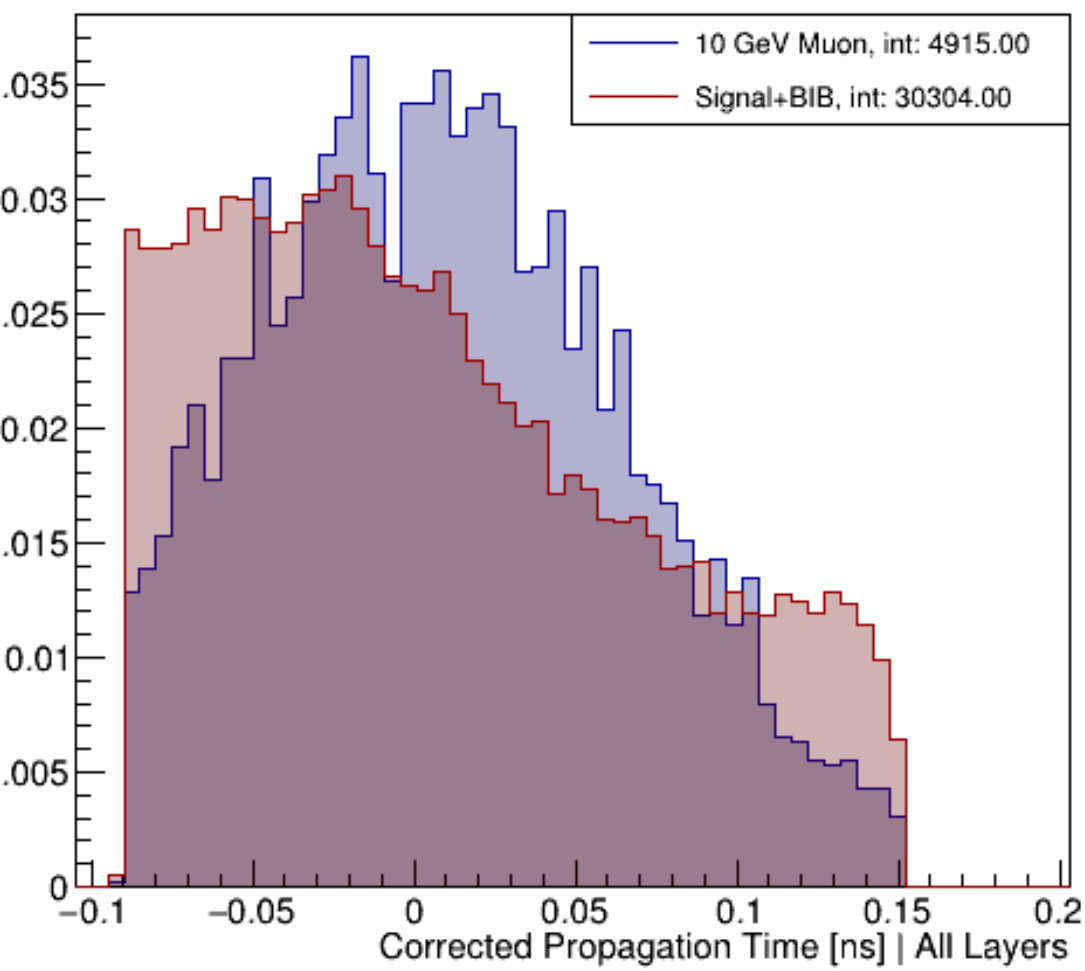
75  $\mu\text{m}$



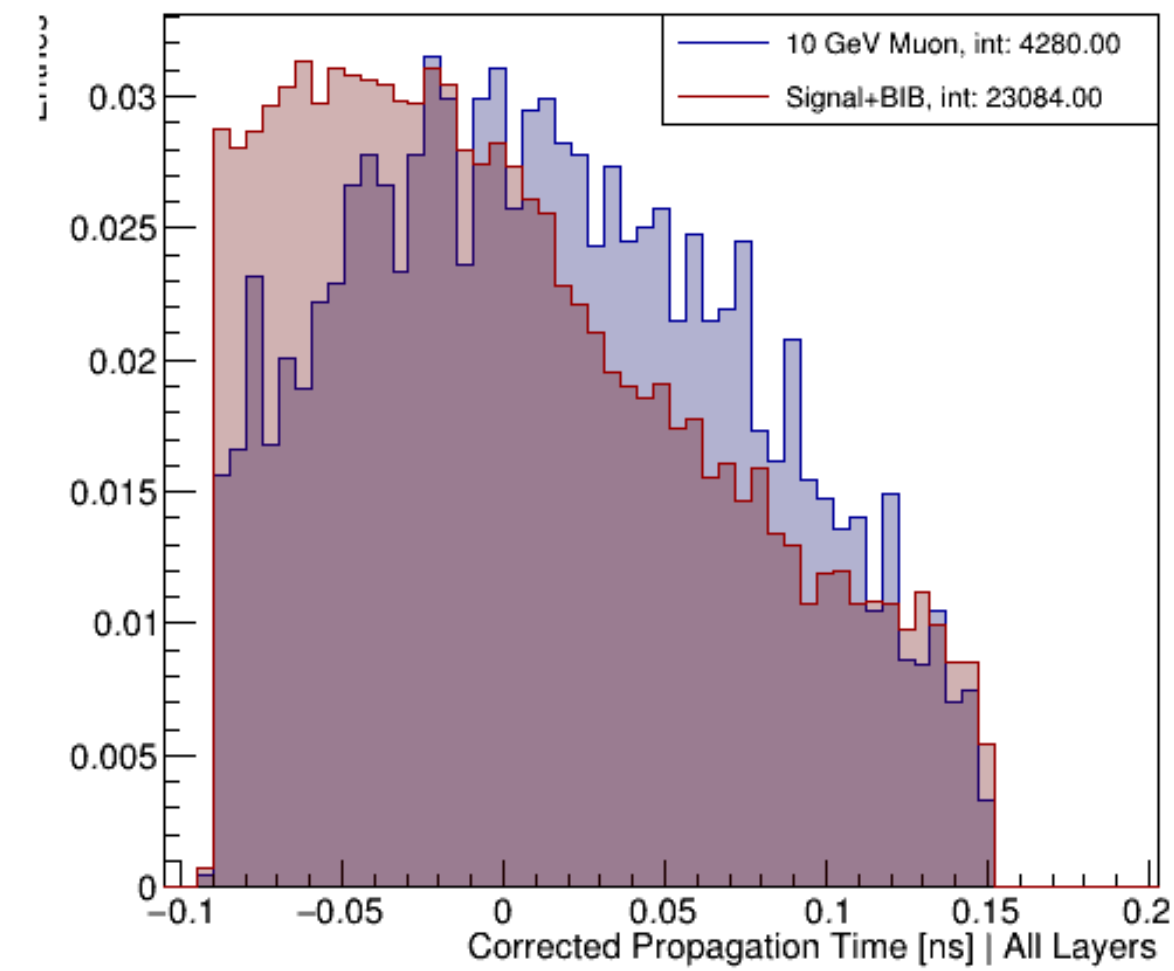
100  $\mu\text{m}$



200  $\mu\text{m}$



400  $\mu\text{m}$



Timing resolution now worsens with increased sensor thickness as expected

# Timing Cut Re-optimization needed

- For VXB, the TOA timing cut range is  $[-0.09, 0.15]$ ns
- Entries lost with realistic timing cut applied:

Sensor Thickness	Signal Entries Before Cut	Signal Entries After Cut	Percentage Loss	BIB Entries Before Cut	BIB Entries After Cut	Percentage Loss
50	5724	5348	6.6%	242542	35669	85.3%
75	5587	5339	4.4%	229579	34835	84.8%
100	5534	5191	6.2%	207262	32247	84.4%
200	5815	4915	15.4%	169882	30304	82.2%
400	5560	4280	23.0%	117259	23084	80.3%

# Timing Cut with $\in [-3\sigma_{\text{Landau}}, 5\sigma_{\text{Landau}}]$

$$\sigma_{\text{Landau}} = 30[\text{ps}] \times \text{sensor thickness}/50[\mu\text{m}]$$

Sensor Thickness [microns]	Time Window [ns]	Signal Entries Before Cut	Signal Entries After Cut	Percentage Loss	BIB Entries Before Cut	BIB Entries After Cut	Percentage Loss
50	[-0.09,0.15]	5724	5360	6.4%	242542	35669	85.3%
75	[-0.14, 0.23]	5588	5511	1.4%	221043	50234	77.3%
100	[-0.18, 0.30]	5534	5456	1.4%	209495	89518	57.2%
200	[-0.36, 0.60]	5815	5640	3.0%	180662	91240	49.5%
400	[-0.72, 1.20]	5560	5484	1.3%	118215	97487	17.5%



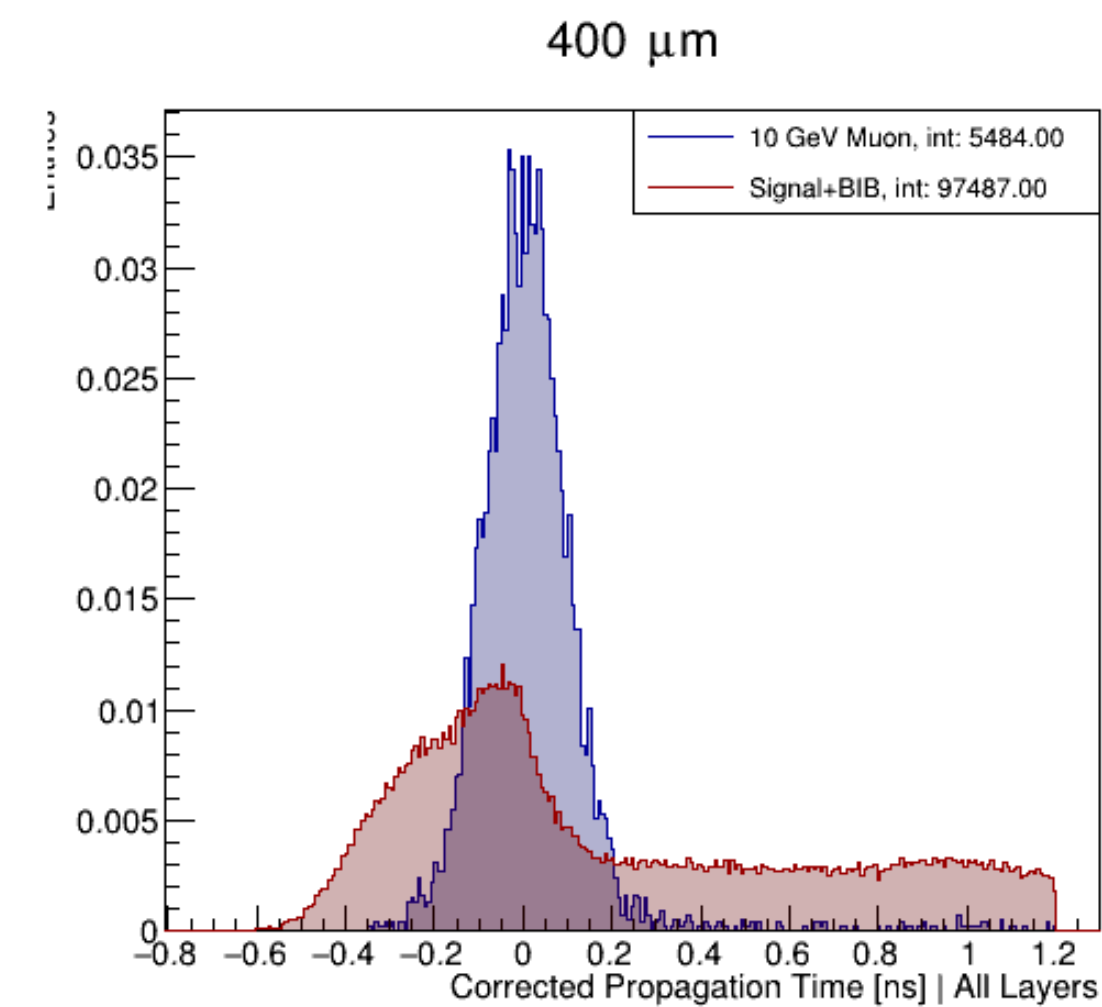
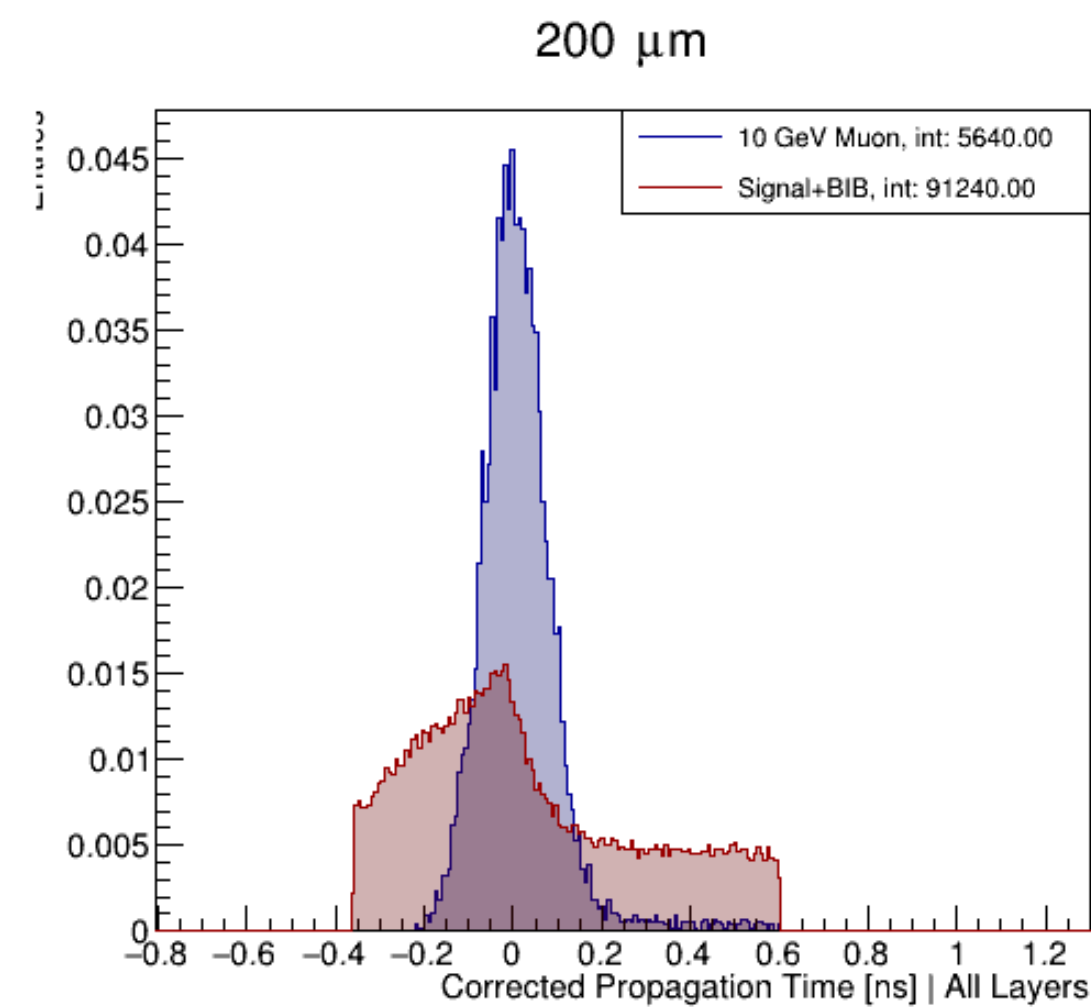
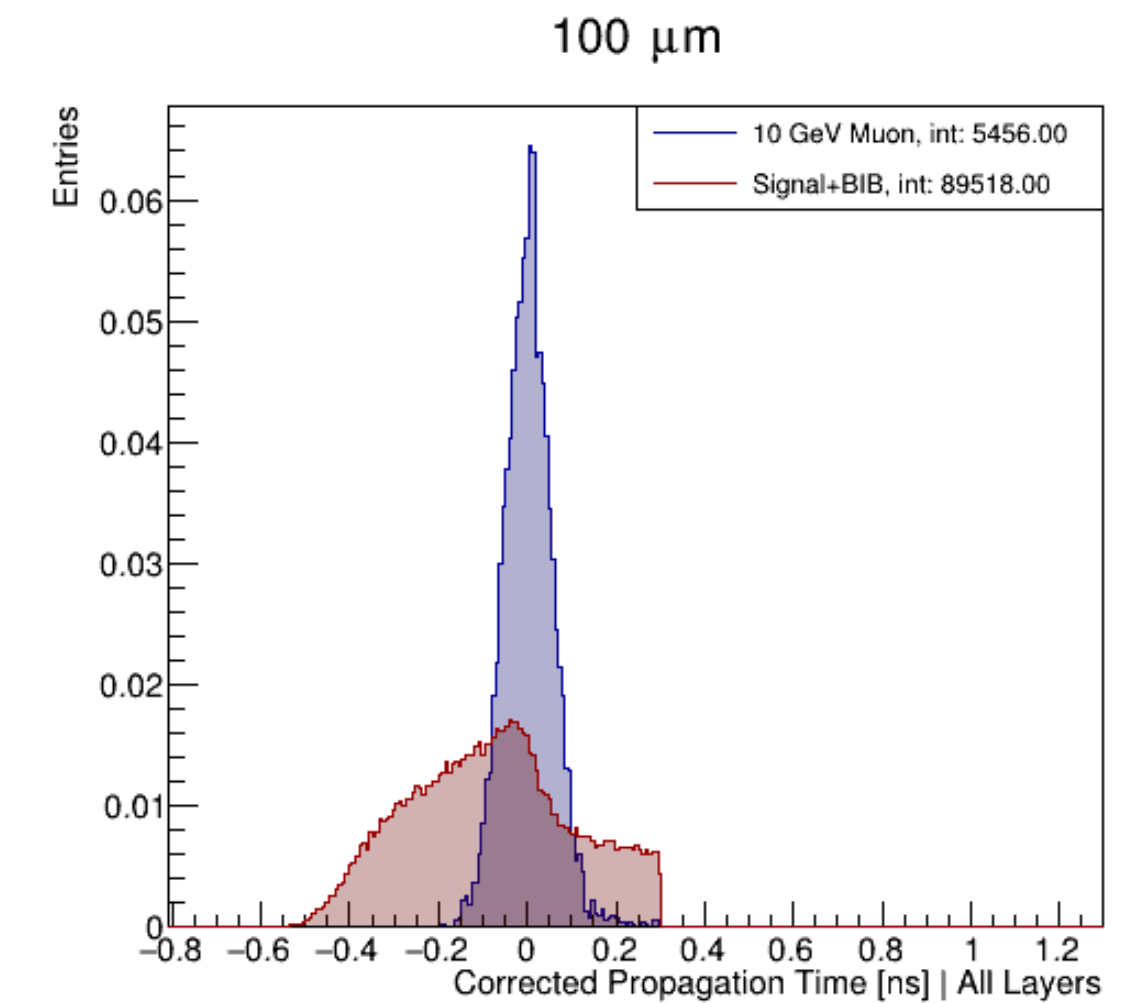
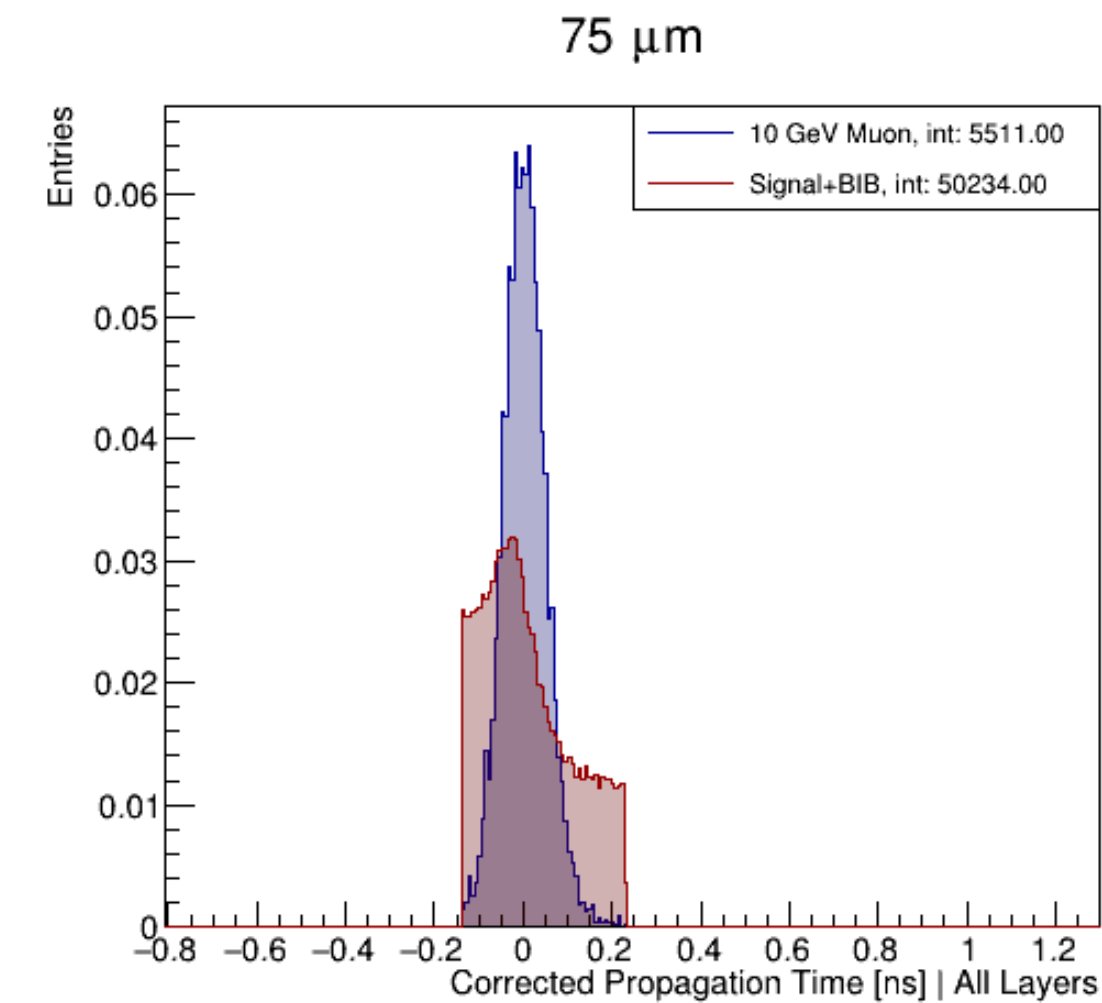
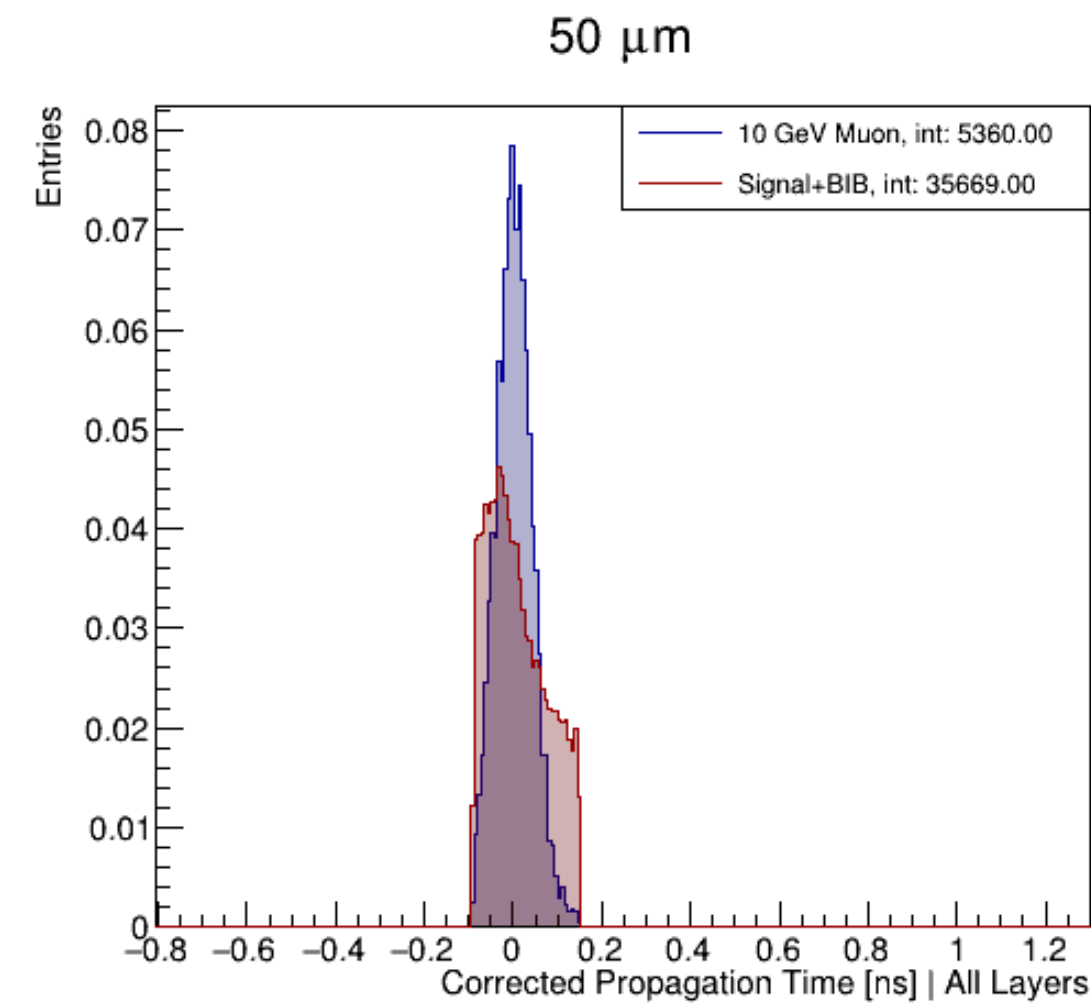
# Timing Cut with $\in [-3\sigma_{\text{Landau}}, 5\sigma_{\text{Landau}}]$

$$\sigma_{\text{Landau}} = 30[\text{ps}] \times \text{sensor thickness}/50[\mu\text{m}]$$

Sensor Thickness [microns]	Time Window [ns]	Signal Entries Loss	BIB Entries Loss	Sig/BIB Entry Loss Percentage
50	[-0.09,0.15]	364	206873	0.18%
75	[-0.14, 0.23]	77	170809	0.045%
100	[-0.18, 0.30]	78	119977	0.065%
200	[-0.36, 0.60]	175	89422	0.20%
400	[-0.72, 1.20]	76	20728	0.37%

# Preliminary Timing Results

- It looks like  $75\text{-}100\ \mu\text{m}$  sensor thickness for the VXB would be an ideal fit based on our results.



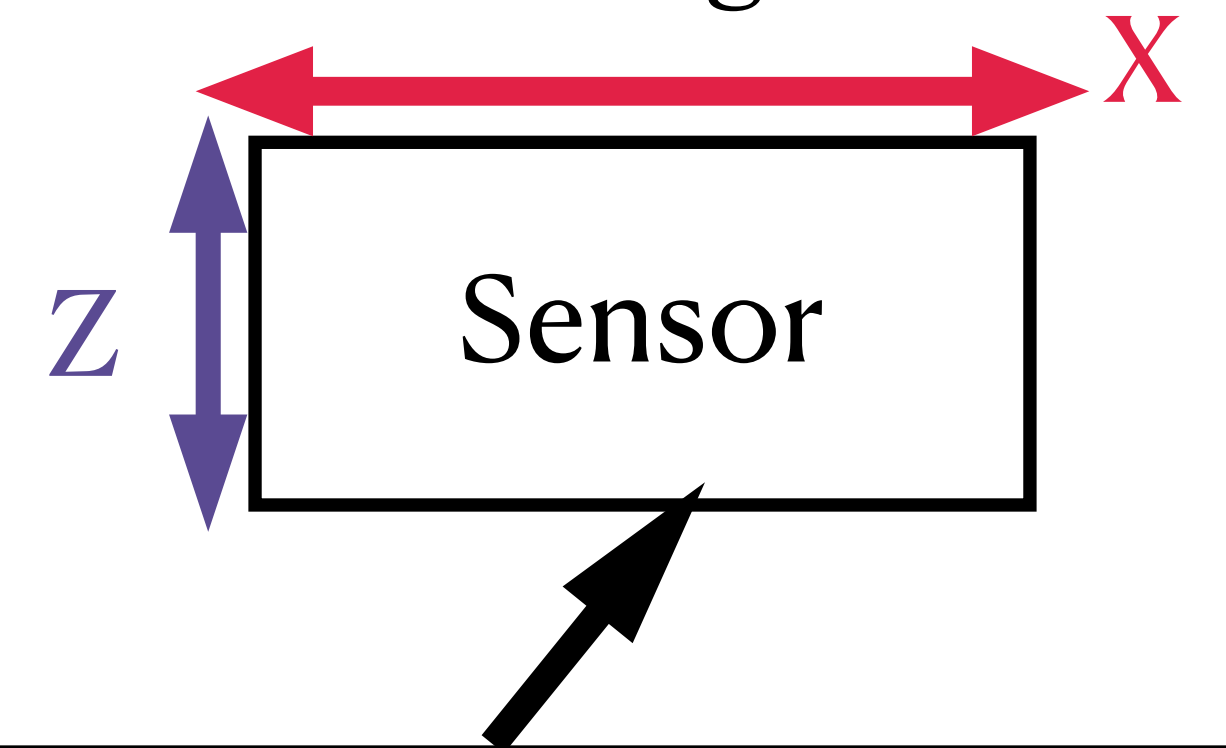
# Conclusion

- Timing resolution has been fixed to realistically respond as planar sensors
- Timing windows have been updated per sensor thickness values as to avoid losing important information
- Comments/suggestions are welcome :)

# Multiple Scattering Motivation

- Currently, the [MuonCVXDDigitiser.cc](http://MuonCVXDDigitiser.cc) code estimates particle exit points via a straight line estimation:

```
661     entry[2] = -_layerHalfThickness[_currentLayer];
662     exit[2] = _layerHalfThickness[_currentLayer];
663     // entry points: hit position is in middle of layer. ex: entry_x = x - (z distance to bottom of layer) * px/pz
664     for (int i = 0; i < 2; ++i) {
665         entry[i] = pos[i] + dir[i] * (entry[2] - pos[2]) / dir[2];
666         exit[i] = pos[i] + dir[i] * (exit[2] - pos[2]) / dir[2];
667     }
668
```



- Where the exit x and y positions are based off of when the particle exits the sensor at the z-plane
- This approach is fine for thin silicon sensors, but if we decide to increase the sensor thickness, and misrepresent the hits/cluster (cluster size)
- this approach can under/overestimate the true exit point for shallow angle particles (especially important for BIB)

# Multiple Scattering (MS) Motivation

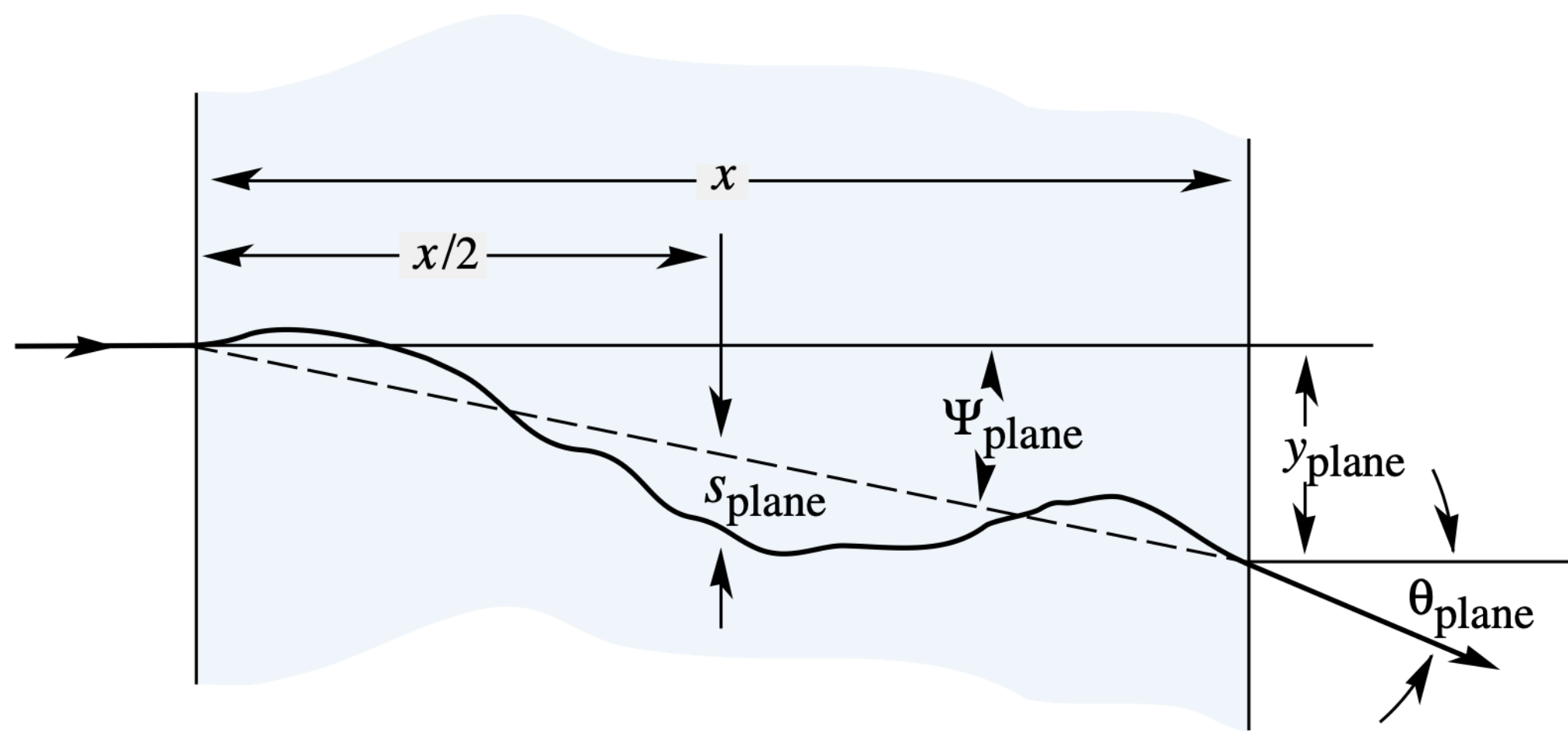


Figure 34.10: Quantities used to describe multiple Coulomb scattering. The particle is incident in the plane of the figure.

[PDG: Passage of particles through matter](#)

$$\theta_0 = \frac{13.6 \text{ MeV}}{\beta c p} z \sqrt{\frac{x}{X_0}} \left[ 1 + 0.088 \log_{10} \left( \frac{x z^2}{X_0 \beta^2} \right) \right]$$

$$= \frac{13.6 \text{ MeV}}{\beta c p} z \sqrt{\frac{x}{X_0}} \left[ 1 + 0.038 \ln \left( \frac{x z^2}{X_0 \beta^2} \right) \right]$$

**We can implement multiple scattering in the muon digitization code to more realistically estimate the hit exit points**

- $(z_1, z_2)$  are gaussian random variables with mean 0 and variance 1.
- $z$  is particle charge

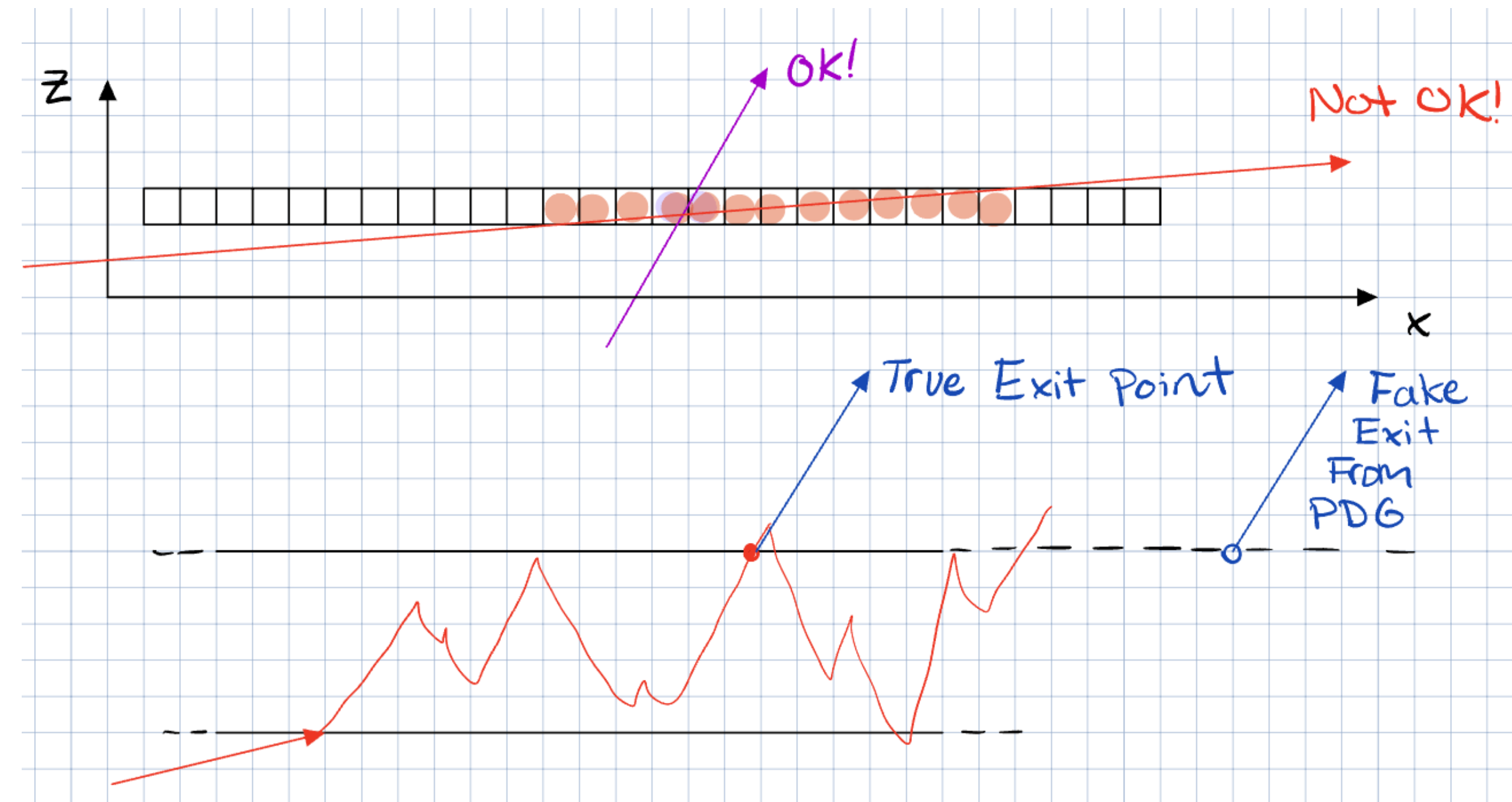
$$y_{\text{plane}} = z_1 x \theta_0 (1 - \rho_{y\theta}^2)^{1/2} / \sqrt{3} + z_2 \rho_{y\theta} x \theta_0 / \sqrt{3}$$

$$= z_1 x \theta_0 / \sqrt{12} + z_2 x \theta_0 / 2;$$

$$\theta_{\text{plane}} = z_2 \theta_0.$$

# Small Caveat

- PDG cannot accurately model shallow particle track multiple scattering
- Shallow particle tracks likely exit the sensor earlier than the PDG estimation for shallow angled particles.
- Once a particle track exits the sensor, it will stop scattering and propagate in a straight line.



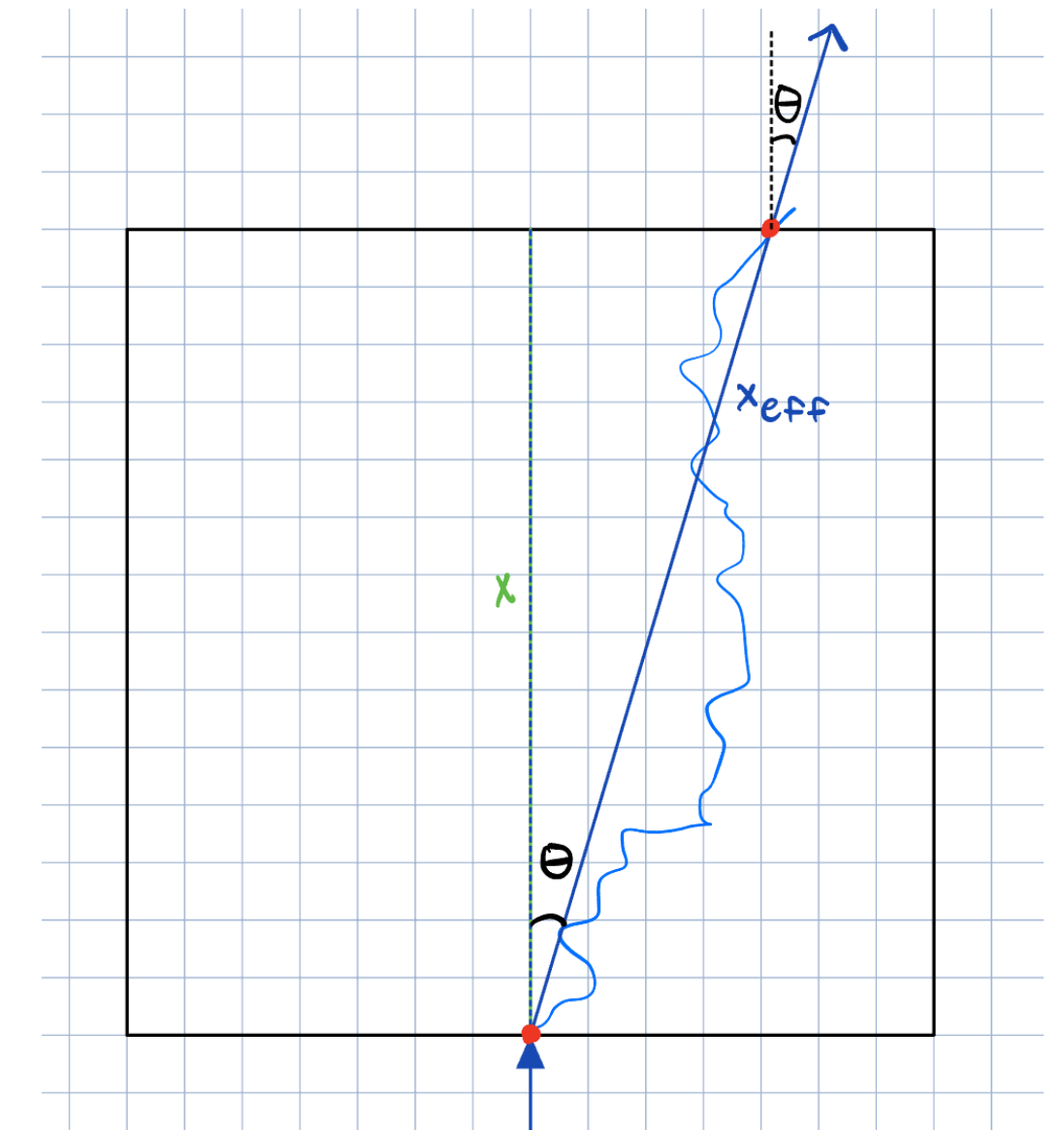
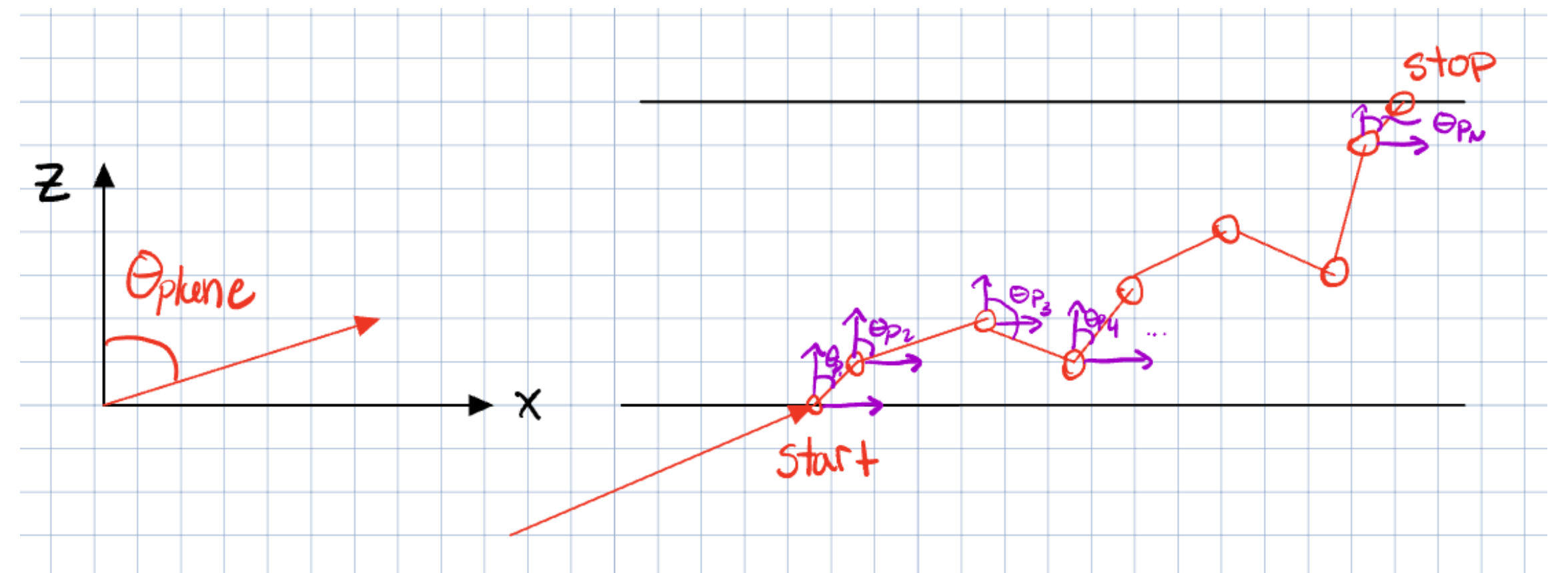
# MS Strategy

## Propagate Particle Track in Increments

- If  $\theta_{T,\text{plane}} > 10^\circ$ , do randomized scattering
- Demonstration linked in indico page
- Else if  $\theta_{T,\text{plane}} < 10^\circ$ , do PDG scattering which is close to normal incidence

$$\theta_{T,\text{plane}} = \sqrt{\left(\theta_{\text{plane},x}\right)^2 + \left(\theta_{\text{plane},y}\right)^2}$$

**Note:**  $\theta_{T,\text{plane}} = \theta_{T,\text{start}}$



# Randomized Scattering Code

1. Define

$$\theta_{T,start} = \sqrt{\tan^{-1} \left( \frac{dir_0}{dir_2} \right)^2 + \tan^{-1} \left( \frac{dir_1}{dir_2} \right)^2}$$

2. Define  $\theta_{plane,x}$  and  $\theta_{plane,y}$

3. If  $\theta_{T,start} \leq 10^\circ$ , do PDG MS  $\rightarrow$  BREAK Loop

4. Else if  $\theta_{T,start} > 10^\circ$ :

1. Define output angles

2. Update direction vector

3. Propagate particle track some fraction of a distance from the entry point (updating the exit)

4. Update position of particle track (if still in sensor)

5. Repeat until particle exits the sensor

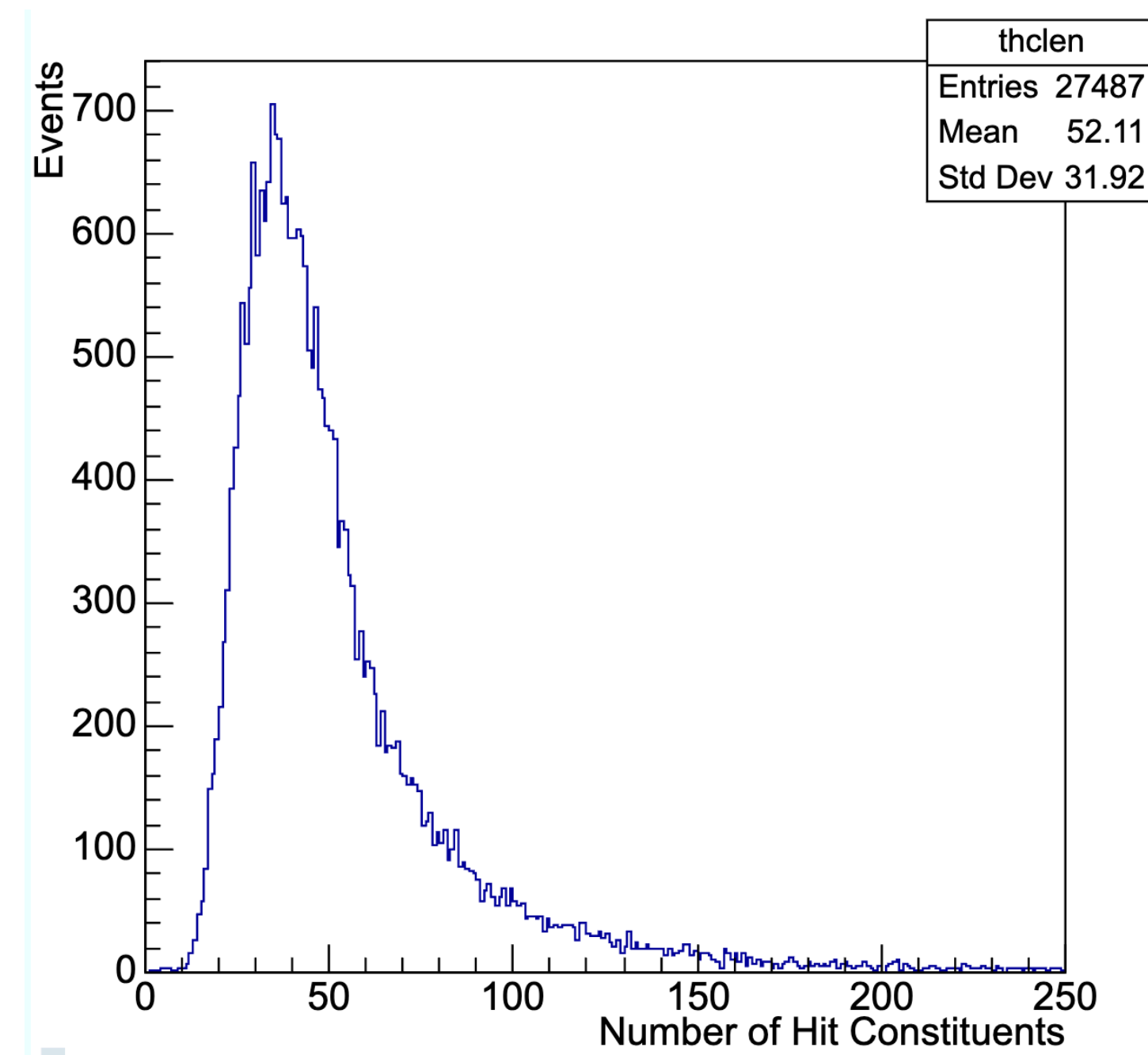
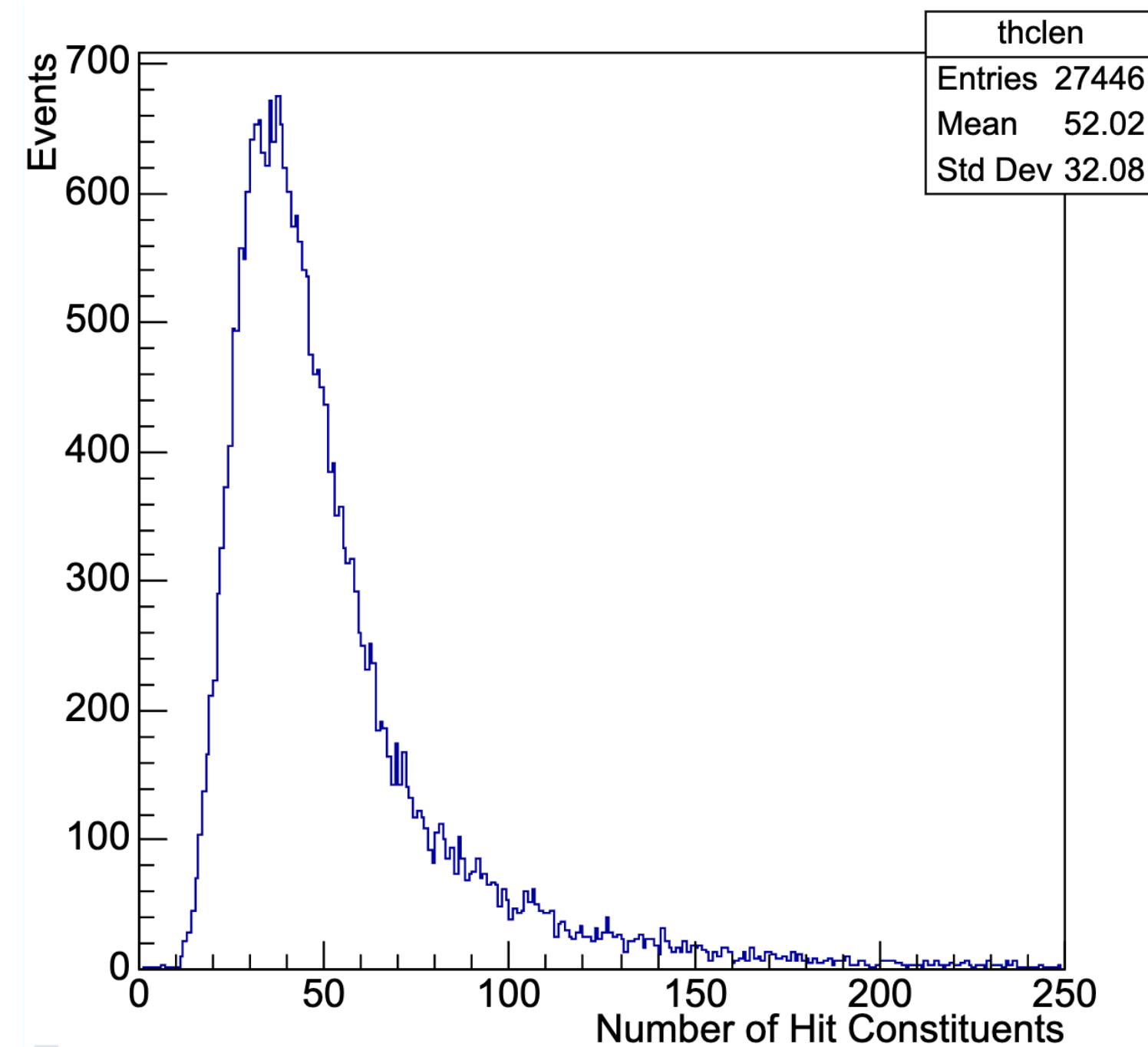
```
709     double thetaT_start = std::sqrt(std::pow(std::atan(dir[0]/dir[2]),2) + std::pow(std::atan(dir[1]/dir[2]),2)); //starting angle in radians
710     //std::cout << "thetaT start: " << thetaT_start << std::endl;
711     for (int i = 0; i < 2; ++i) {
712         entry[i] = pos[i] + dir[i] * (entry[2] - pos[2]) / dir[2];
713     }
714
715     double theta_plane_x, theta_plane_y, theta_out_x, theta_out_y; //, theta_tangent;
716     double r_plane[2], theta_plane[2], z1[2], z2[2];
717     int counter = 0;
718     bool smallAngle = false;
719     exit[2] = 0;
720     double previous_exit[3] = {0,0,0};
721     while (exit[2] < _layerThickness[_currentLayer]){
722         theta_plane_x = gauss(rng)*theta_0;
723         theta_plane_y = gauss(rng)*theta_0;
724         //theta_tangent = std::sqrt(std::pow(theta_plane_x,2) + std::pow(theta_plane_y,2));
725         //do small angle approximation:
726         if(std::abs(thetaT_start) <= 10*M_PI/180 || std::abs(thetaT_start) >= 170*M_PI/180) {
727             for (int i = 0; i < 2; ++i) {
728                 z1[i] = gauss(rng);
729                 z2[i] = gauss(rng);
730                 theta_plane[i] = z1[i] * theta_0;
731                 r_plane[i] = sensorT * theta_0 * (z1[i]/std::sqrt(12) + z2[i]/2);
732             }
733             smallAngle = true;
734             break; //to escape the while loop
735         }
736         else{
737             theta_out_x = theta_plane_x + std::atan2(dir[0],dir[2]);
738             theta_out_y = theta_plane_y + std::atan2(dir[1],dir[2]);
739
740             //update dir vector:
741             dir[0] = std::tan(theta_out_x)*dir[2];
742             dir[1] = std::tan(theta_out_y)*dir[2];
743
744             //prop pttle some fraction of a distance of the exit point:
745             static thread_local std::normal_distribution<> gauss2(_layerThickness[_currentLayer] - exit[2], _layerThickness[_currentLayer]);
746             for (int i = 0; i < 3; i++) previous_exit[i] = exit[i];
747             exit[2] = std::abs(gauss2(rng));
748             exit[0] = pos[0] + dir[0] * (exit[2] - pos[2]) / dir[2];
749             exit[1] = pos[1] + dir[1] * (exit[2] - pos[2]) / dir[2];
750
751             //update position if we are still inside the sensor:
752             if(exit[2] < _layerThickness[_currentLayer]){
753                 pos[0] += (exit[0] - previous_exit[0]);
754                 pos[1] += (exit[1] - previous_exit[1]);
755                 pos[2] += (exit[2] - previous_exit[2]);
756             }
757             //update thetaT_start:
758             thetaT_start = std::sqrt(std::pow(std::atan(dir[0]/dir[2]),2) + std::pow(std::atan(dir[1]/dir[2]),2)); //updated starting angle in radians
759         }
760     }
```

# 10 MeV Muons $\theta \in [80^\circ, 100^\circ]$

Generated with particle gun (10k Muons), Note this angle is wrt the primary vertex

and not  $\theta_{\text{plane}}$  With MS

Without MS

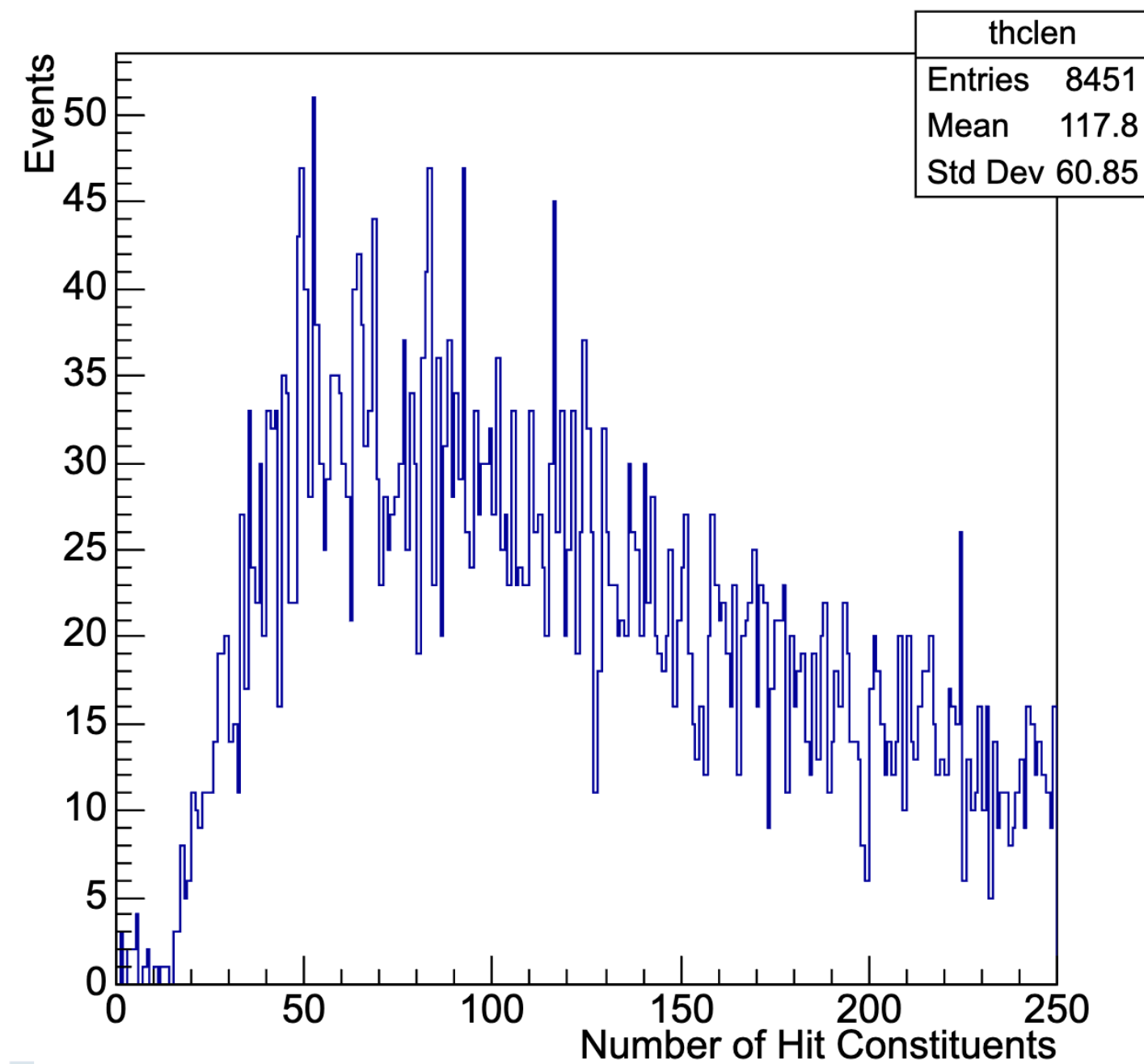


The mean does not change as expected because particles with near-to-normal incidences will not scatter far off from the straight line estimate

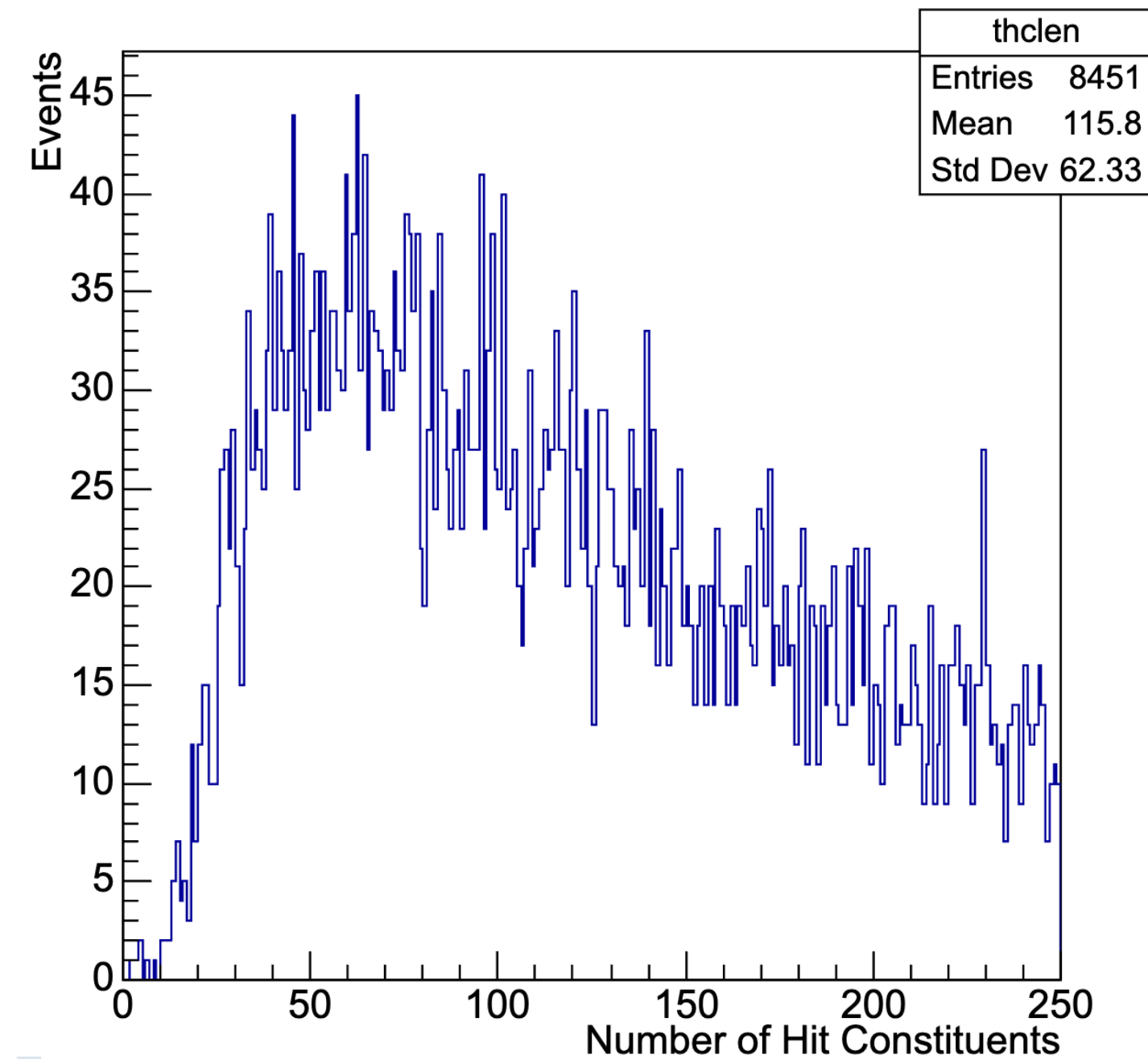
# 10 MeV Muons $\theta \in [0^\circ, 20^\circ]$

Generated with particle gun (10k Muons)

Without MS



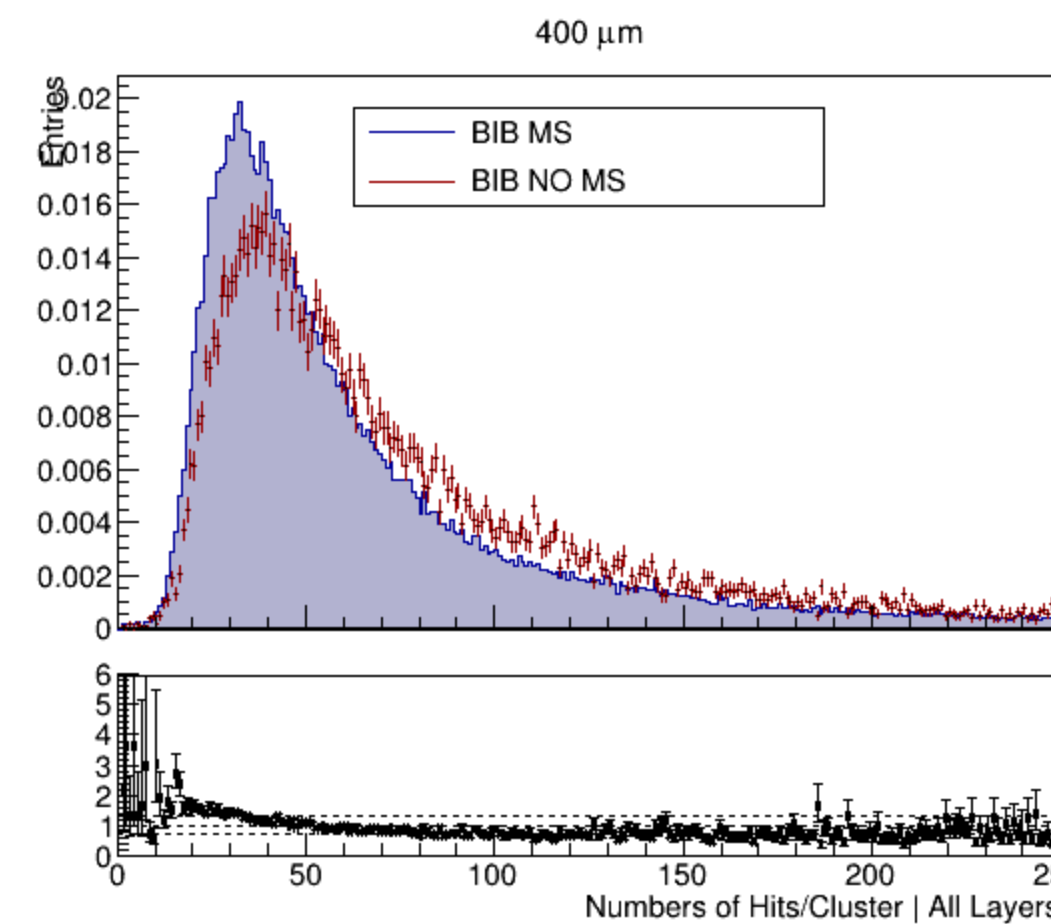
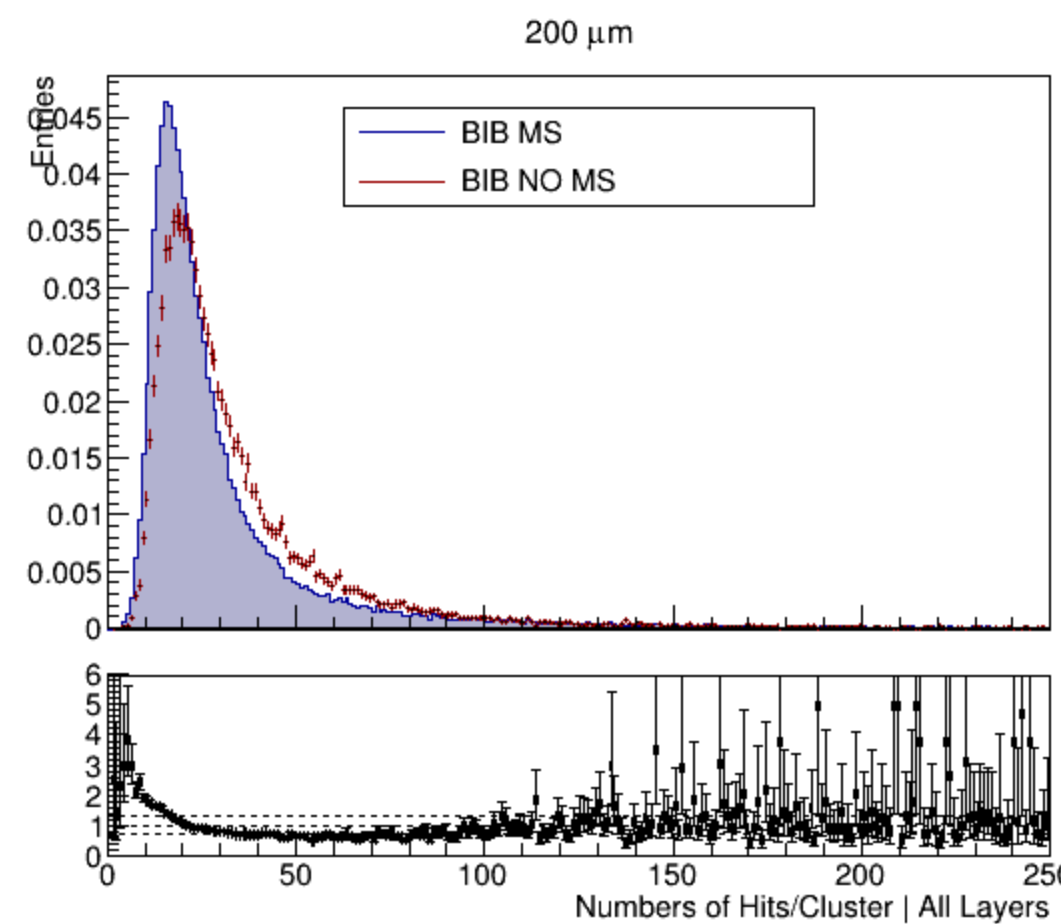
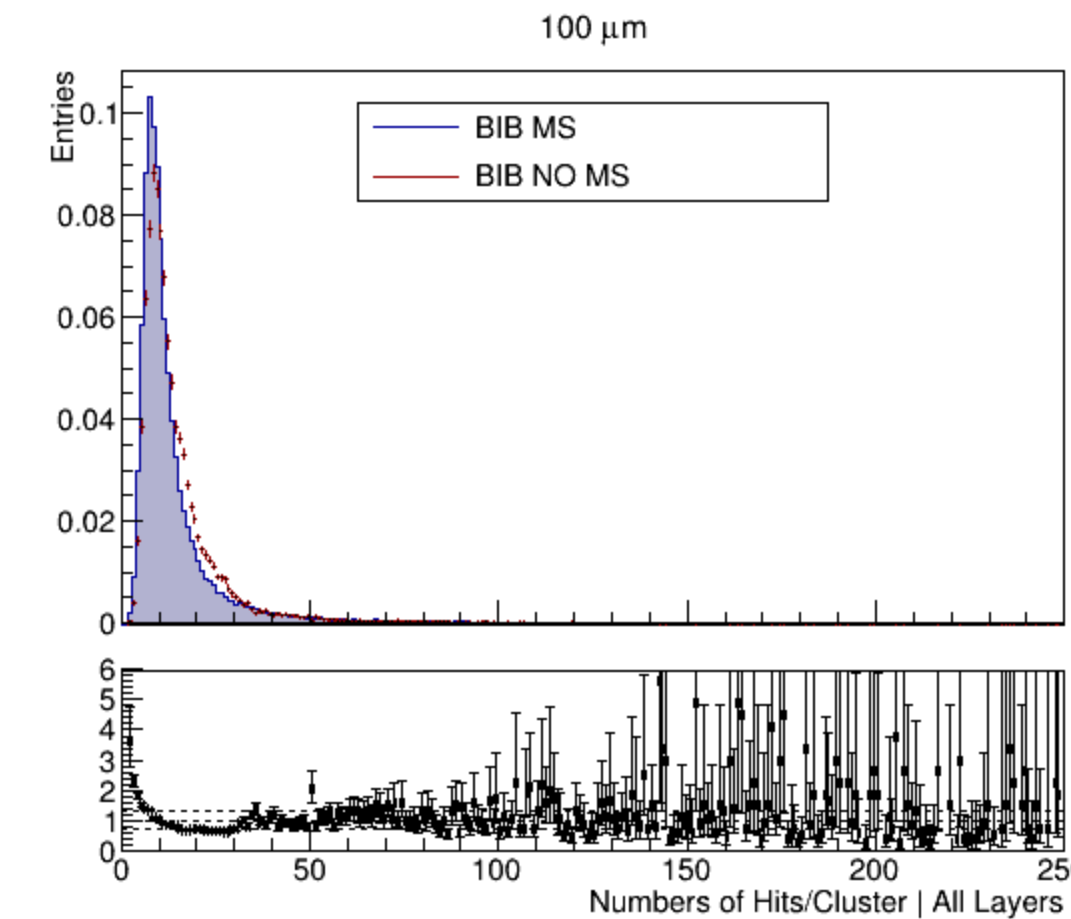
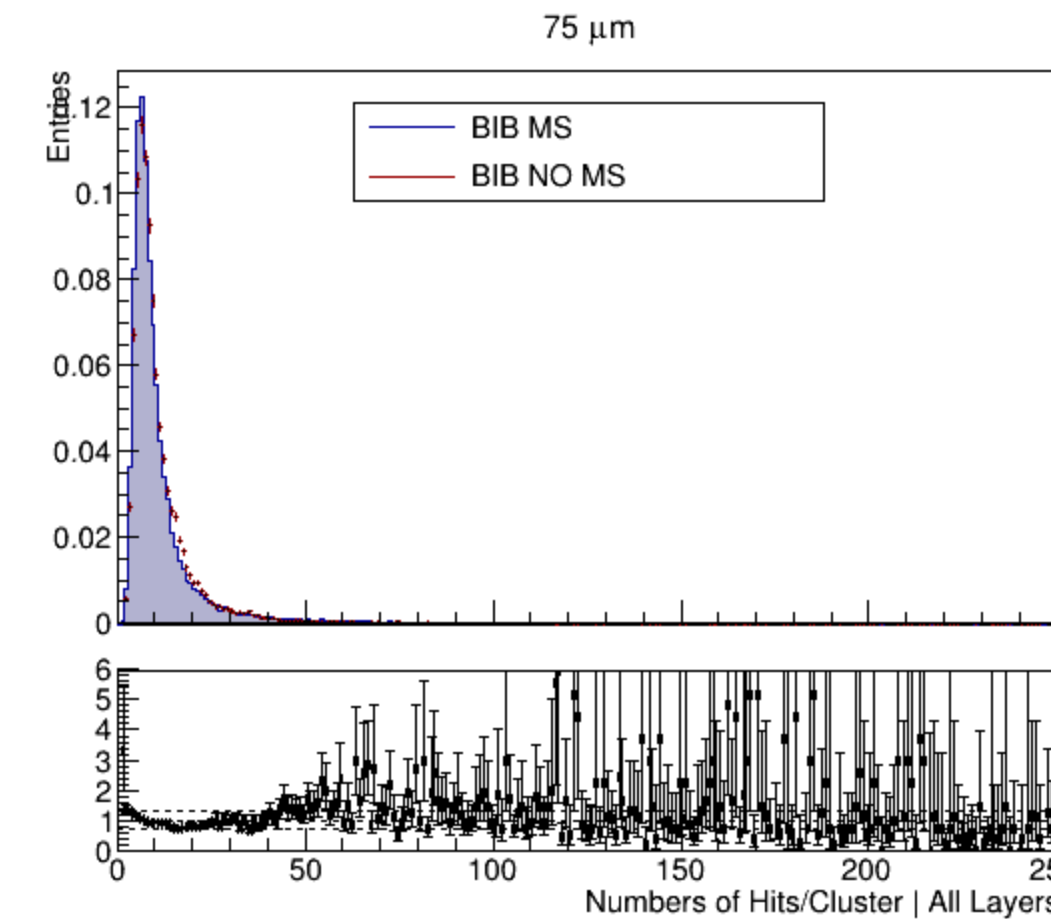
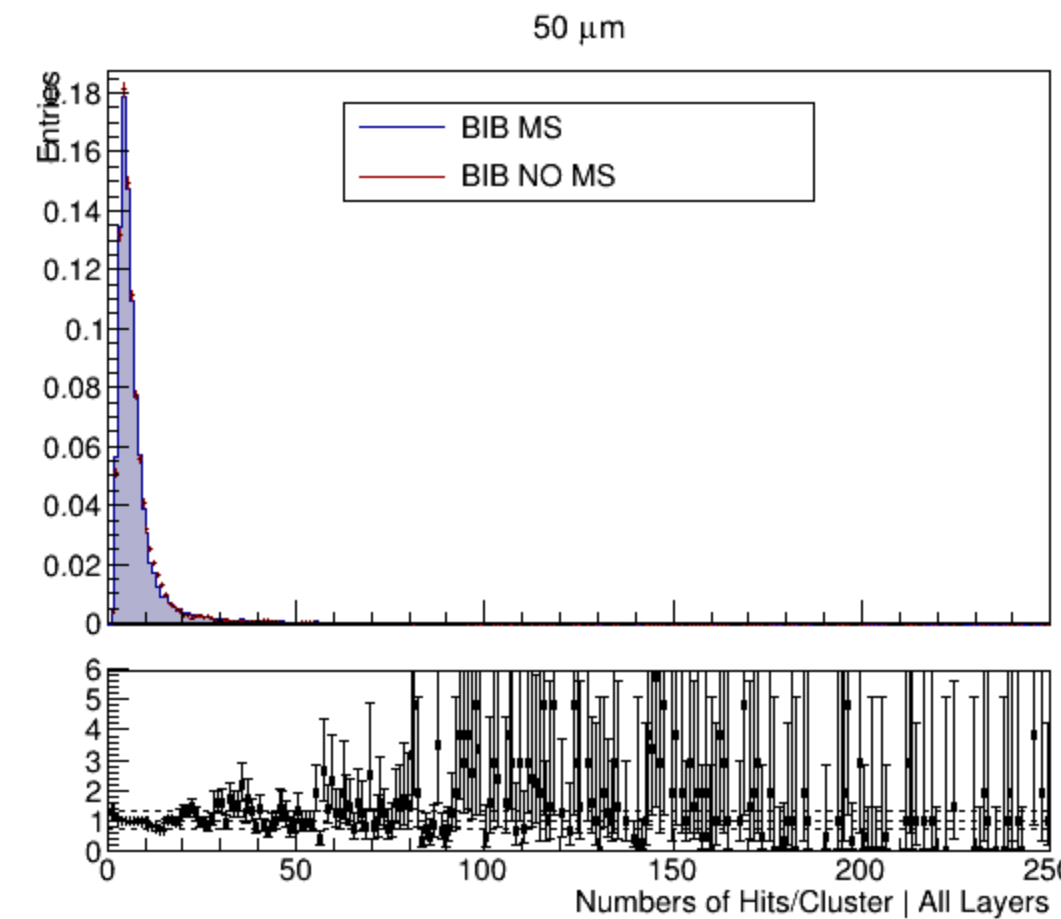
With MS



The mean decreases with MS applied as expected because low-angled particles are expected to typically exit sooner than the straight line estimate

# Preliminary Results: BIB Overlap for MS vs No MS

- These preliminary results seem to show the hits/cluster mean shift further to the right as the thickness increases which is promising
- We plan to plot the  $\theta_{\text{incidence}} \in [0^\circ, 20^\circ]$  and  $[80^\circ, 100^\circ]$  for the BIB as a sanity check



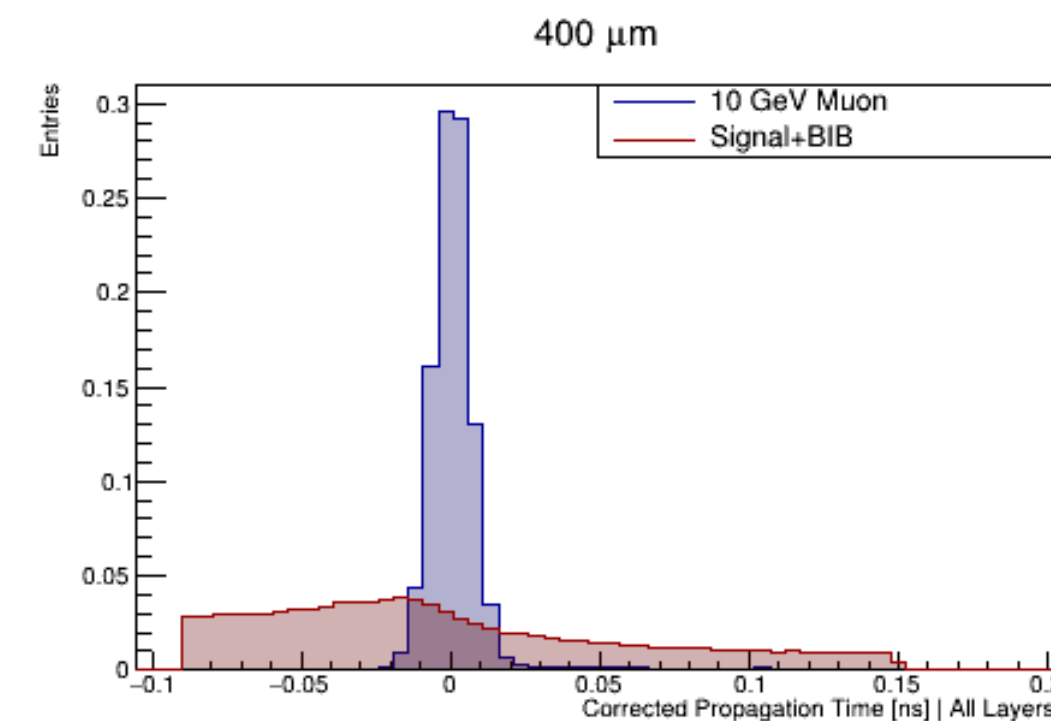
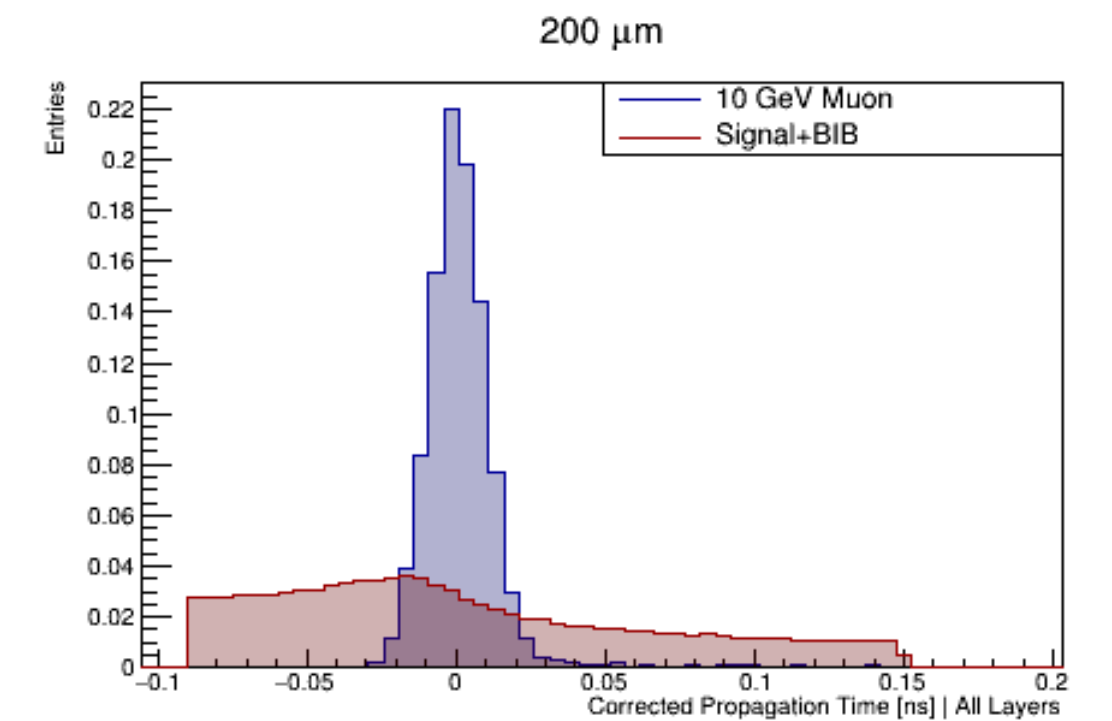
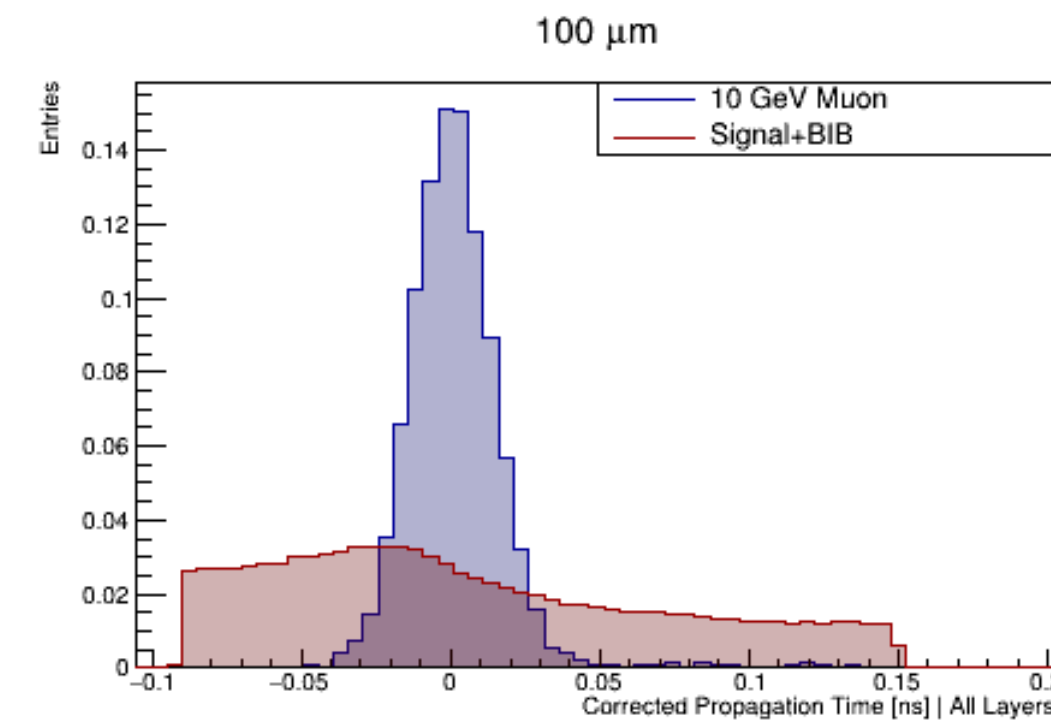
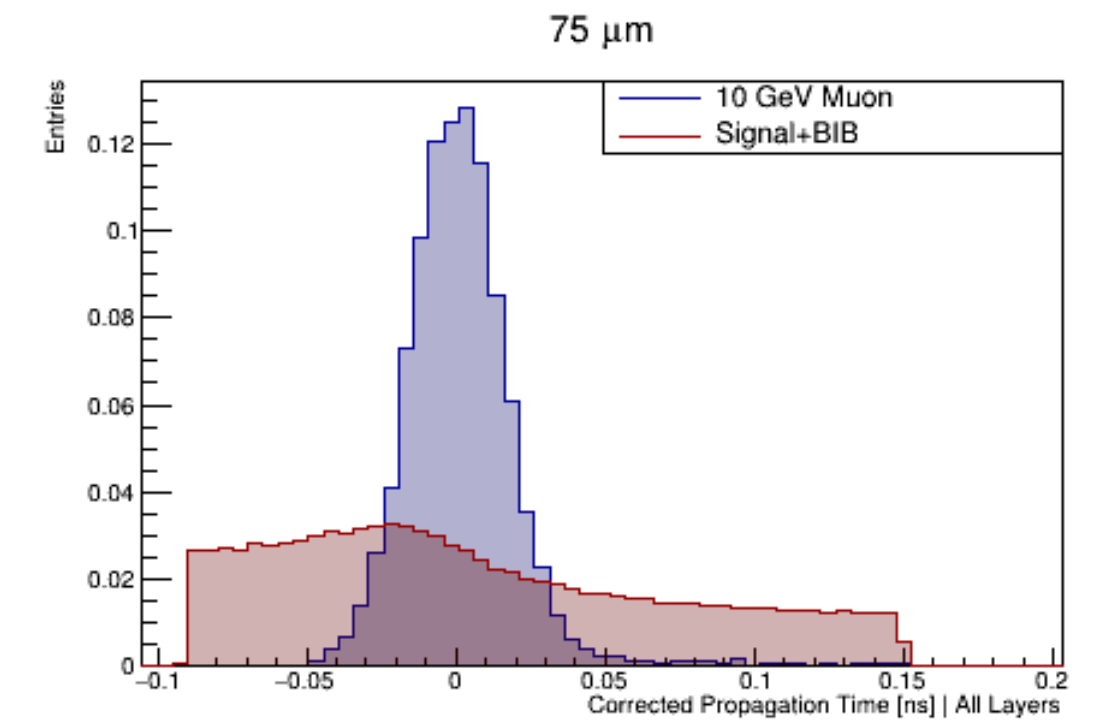
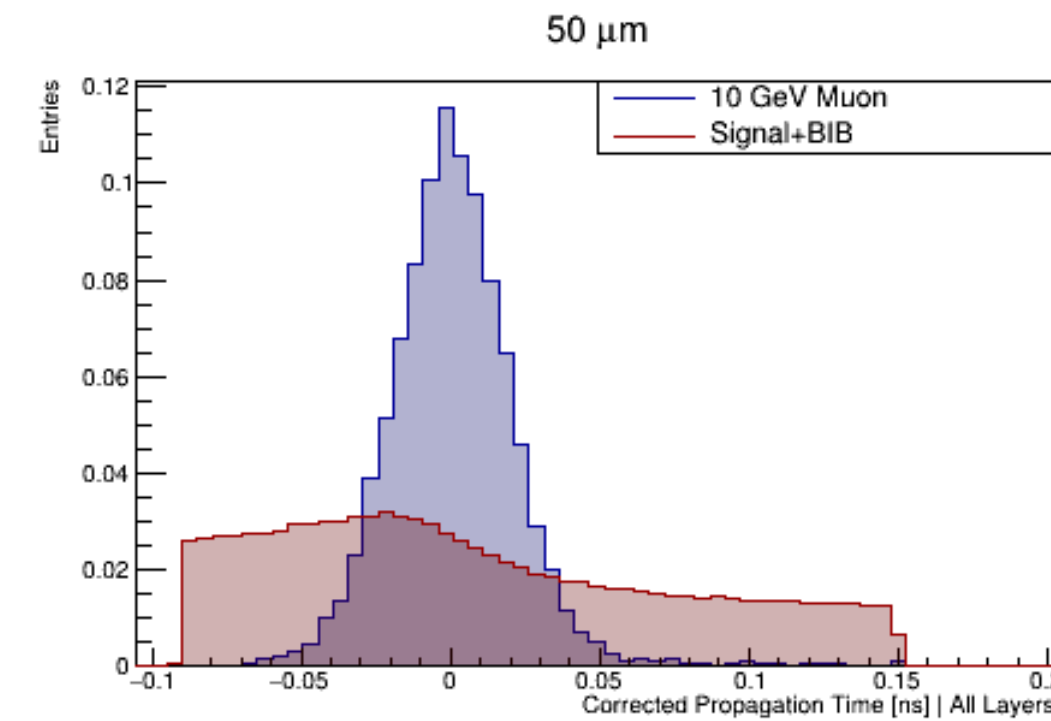
# Conclusion

- The results are not particularly overwhelming
- The MS code is decreasing the averaged pixel hits/clusters, but not by a large margin
- Further validation of individual events is ongoing
- Feedback, comments, thoughts are encouraged :)

# Back Up

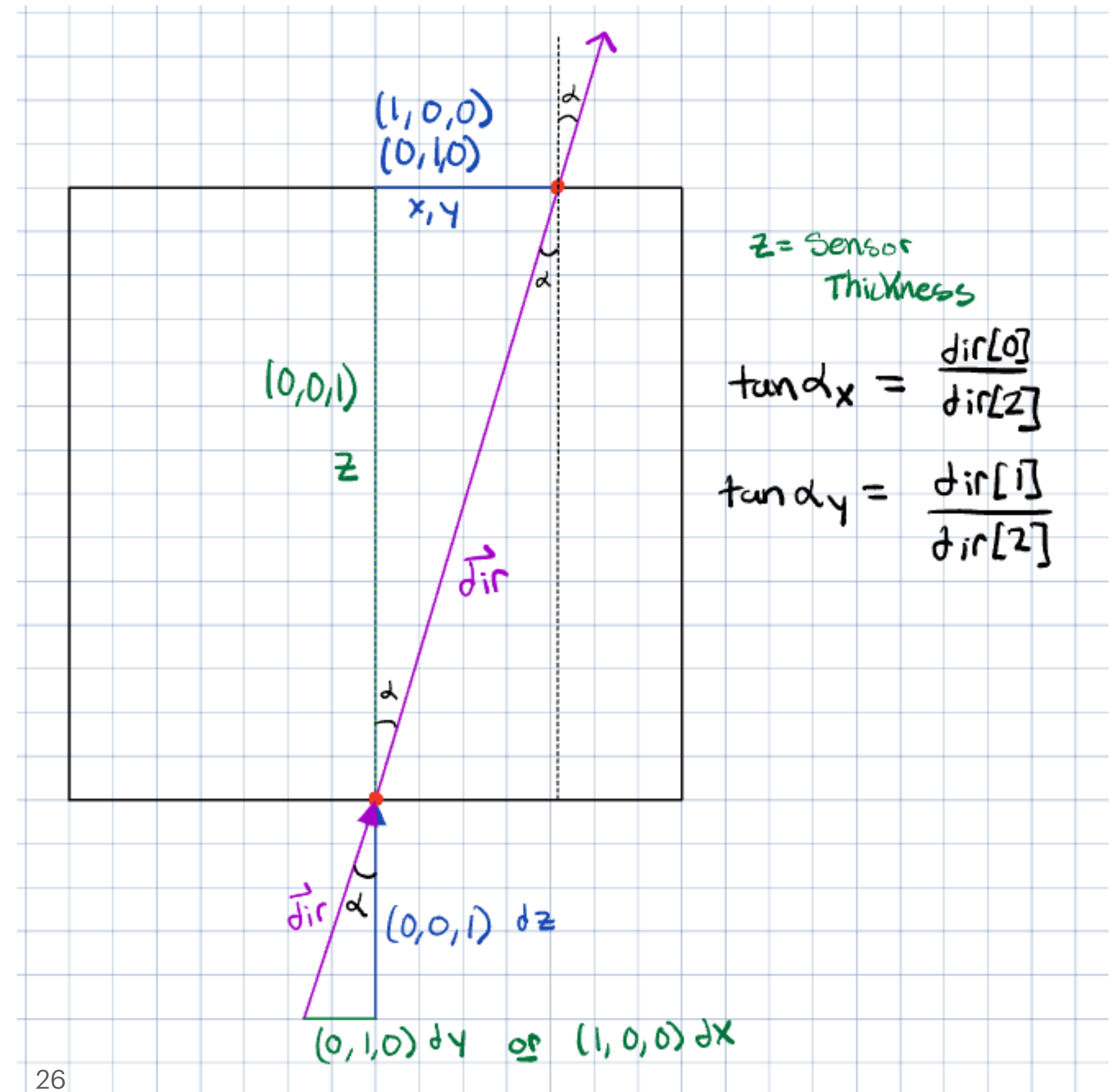
# Propagation time for sensor thickness $\in [50, 400] \mu m$

- We expect to see timing resolution worsen with increasing thickness
- We see the opposite effect of what was originally expected
- Clearly, the current digitization does not accurately model planar sensor timing resolution



# Particles Hitting Sensor at an Angle $\alpha$

- $Z$  = sensor thickness
- This is considering particles that come in at different angles without multiple scattering



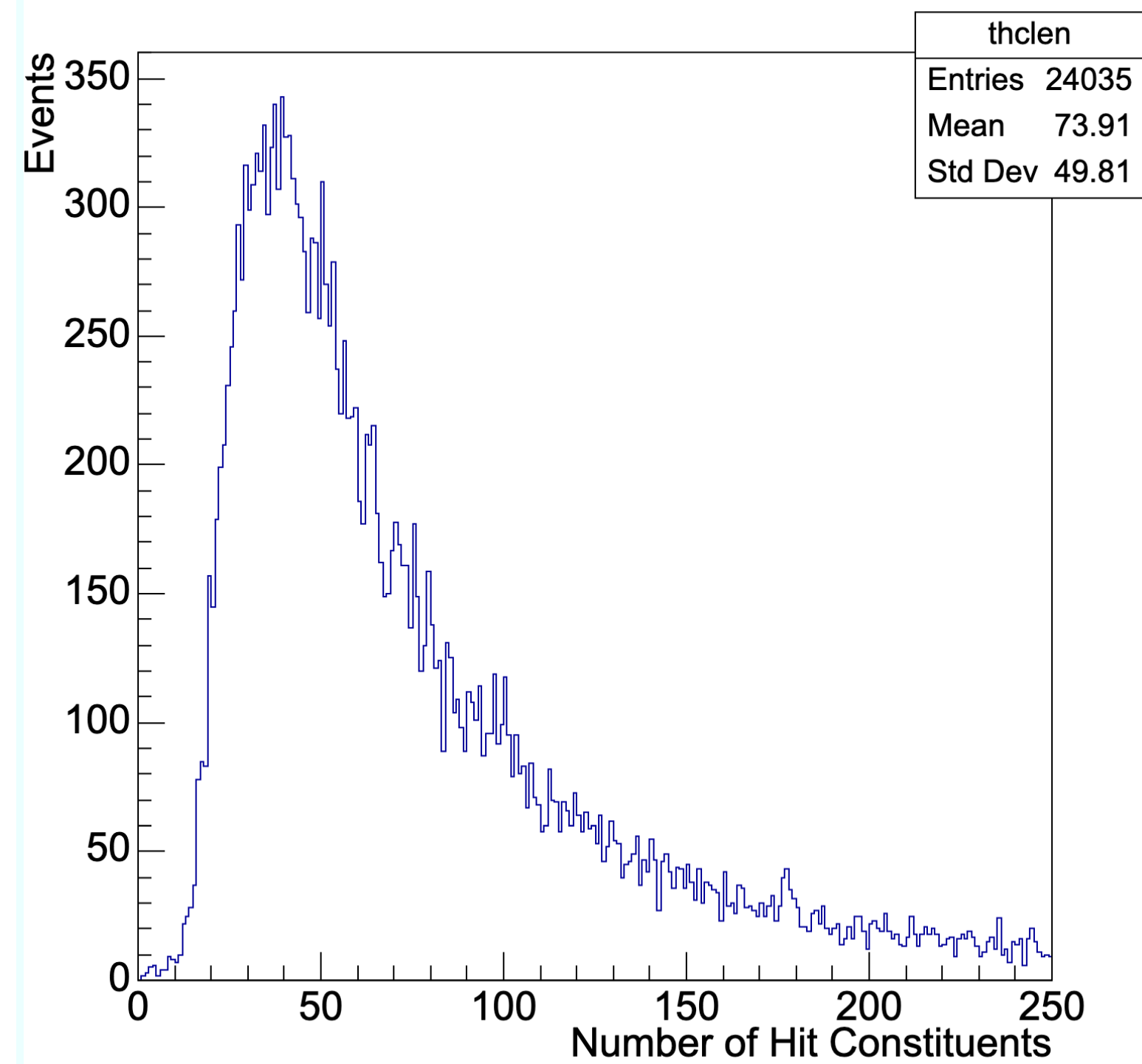
# Timing Cut Problem

- For VXB, the TOA timing cut range is [-0.09,0.15]ns

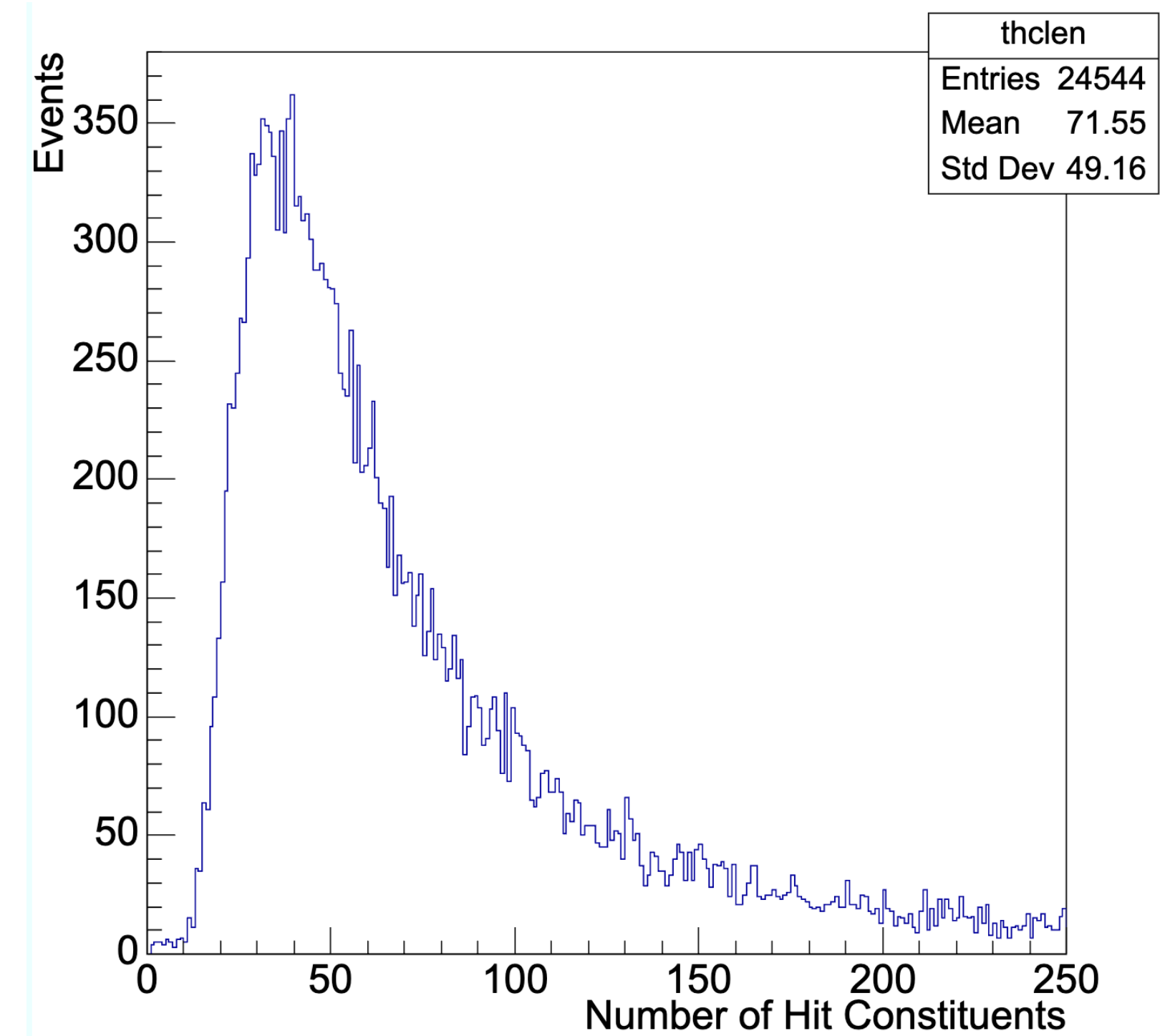
- | Sensor Thickness [microns] | Signal Entries Loss | BIB Entries Loss | Sig/BIB Entry Loss Percentage |
|----------------------------|---------------------|------------------|-------------------------------|
| 50                         | 364                 | 206873           | 0.17%                         |
| 75                         | 248                 | 194744           | 0.13%                         |
| 100                        | 343                 | 175015           | 0.20%                         |
| 200                        | 900                 | 139578           | 0.64%                         |
| 400                        | 1280                | 94175            | 1.4%                          |

# Results for $400\mu m$ BIB in VXB

## Without MS



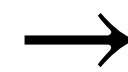
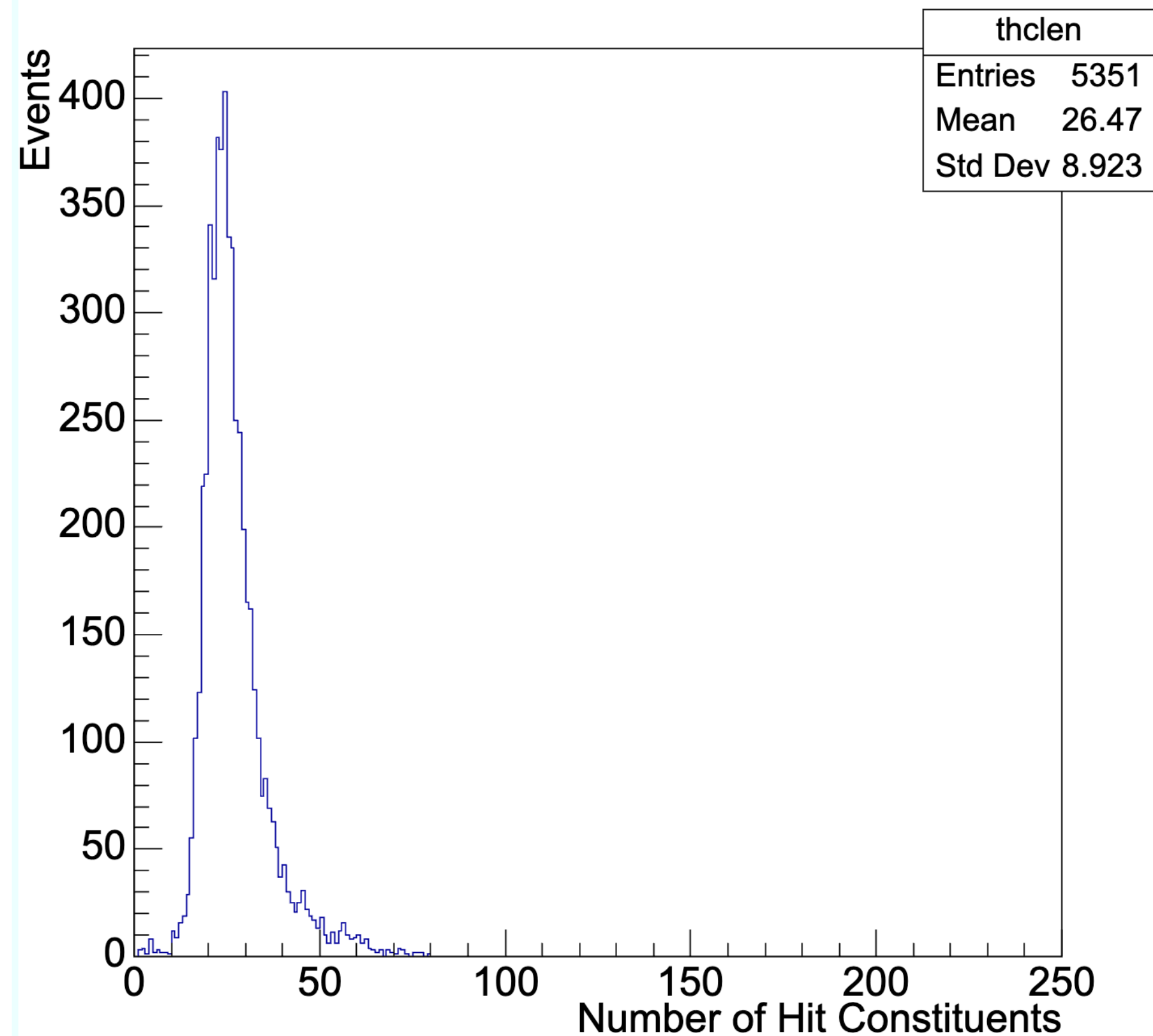
## With MS



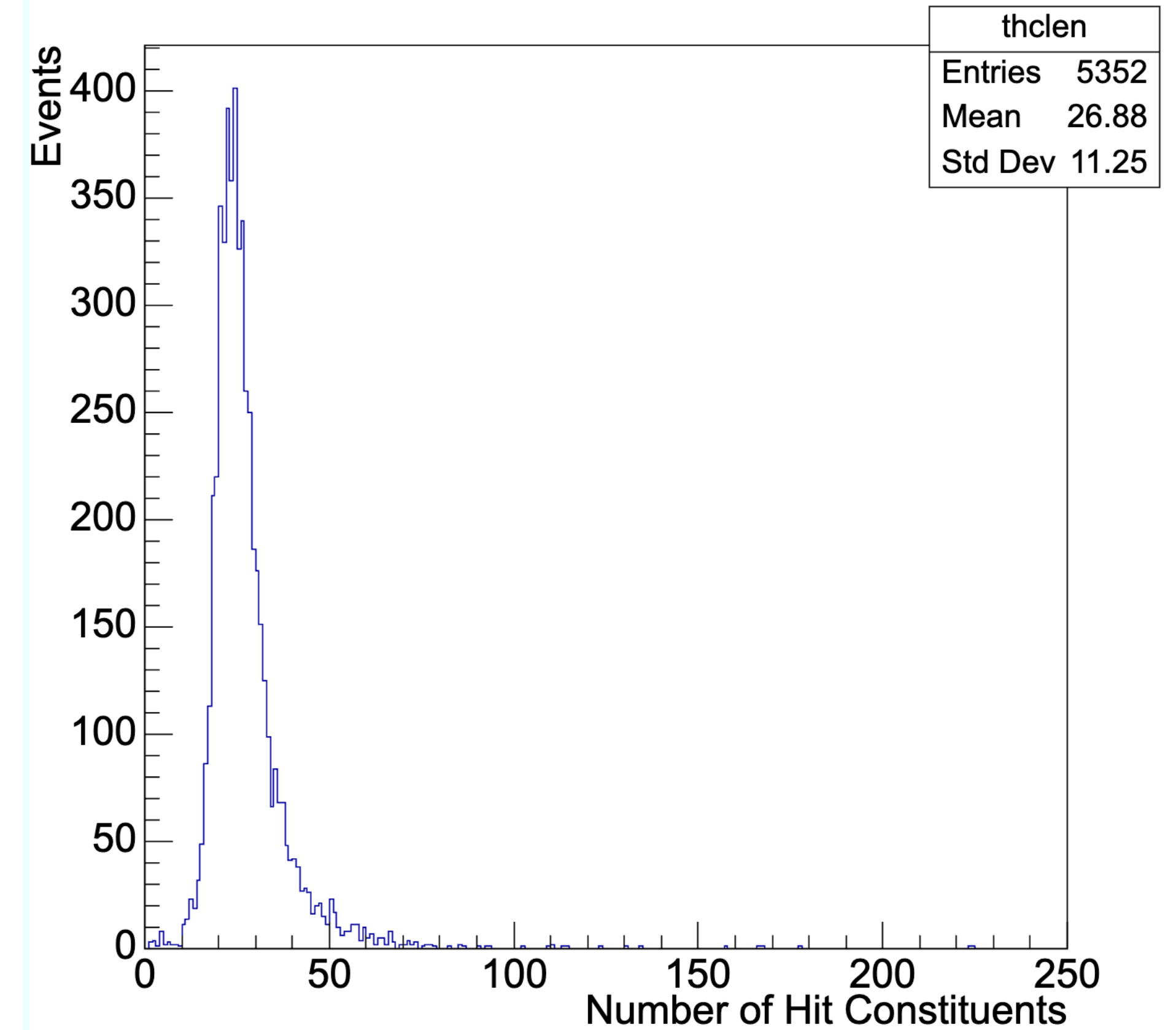
# Results for $400\mu m$ Sig in VXB

Before  $\rightarrow$  After: Hits/Cluster

Without MS



With MS



# Cluster Size Results w/ MS

