Documentation for SoLID Tracking

Weizhi Xiong (weizhi.xiong@duke.edu)

Duke University

Abstract

In this technical note, I will summarize my studies on the SoLID tracking reconstruction since 2015. I will start by introducing the two major tracking related software programs, namely libsolgem for SoLID GEM digitization and SoLIDTracking for SoLID tracking reconstruction. Lastly, I will summarize my recent results, unfinished tasks and also possible improvment for this development.

Contents

1 Introduction **2**

2 SoLID GEM digitization program **2**

3 SoLID tracking reconstruction program **9**
3.1 Structure of the program . . . 9
3.2 GEM clustering . . . 10
3.2.1 GEM signal processing and noise rejection . . . 10
3.2.2 GEM cluster reconstruction . . . 14
3.2.3 Rejection of false combinations . . . 14
3.3 Track finding and fitting . . . 16
3.3.1 Initializing the tracking finder . . . 18
3.3.2 Track follow . . . 19
3.3.3 Final selection on reconstructed tracks . . . 20
3.4 Example of replay script . . . 22

4 Results **24**
4.1 GEM occupancy . . . 24
4.2 Track finding . . . 25
4.3 Vertex resolution . . . 33

5 Summary **34**
5.1 Unfinished tasks and some thoughts for the future development . . . 38
5.1.1 Future development for libsolgem . . . 38
5.1.2 Future development for the tracking reconstruction . . . 38
5.1.3 Suggested piority list . . . 40

6 Appendix **42**
6.1 A-1 Database for libsolgem . . . 42
6.2 A-2 Basic Class introduction for libsolgem . . . 45
6.3 A-3 Validation for the GEM shaping functions . . . 47
6.4 B-1 Basic Class introduction for SoLIDTracking . . . 48
6.5 B-2 Database for SoLIDTracking . . . 52
6.6 B-3 Parameters for the track finders . . . 53
6.7 C-1 Other supporting files and a working directory on ifarm . . . 54

1 Introduction

SoLID uses the Gas Electron Multipliers(GEM) as tracking detectors in a high magnetic solenoidal field. For any tracking reconstruction program, there should be at least two major steps. The first step is the track finding (in many tracking reconstruction literatures, it is called the pattern recognition), which searches for potential hits that belong to the same track from the tracking detectors. The second part is the track fitting, which fit the hits selected from the first step in order to obtain optimal estimations for the vertex variables and other track related qualities.

The major focus prior to the year 2016 was the track fitting part, for which we have developed a Kalman Filter (KF) track fitting program and obtained preliminary results for the three major SoLID configurations (SIDIS, J/ψ and PVDIS). These preliminary results satisfy requirements in original proposals. At present, the focus is shifted toward the track finding part, which is considered more challenging given the high luminosity and high pile-up situation of SoLID. On one hand, a highly efficient and accurate track finding algorithm is needed in order to distinguish the correct track in a high background environment, on the other hand, a more realistic GEM digitization program is needed to fully simulate the GEM detector responses and to test the behavior of the track finding algorithm.

For the GEM digitization part, we have modified and tuned the already-existed SoLID GEM digitization in order to have a better match with real data. Further developments related to segmentation of readout strips have been carried out by Dr. Richard Holmes from Syracuse University and he has already summarized his study in his technical note. Thus, I will not go over this part in this note.

For the track finding part, we have extended the KF track fitting algorithm so that it can be used as a track finder as well, achieving a concurrent track finding and fitting. Currently, track finding programs based on the modified KF is being developed and tested under realistic background environments for the three major SoLID configurations. For the SIDIS 3He configuration, using only one time-sample from the APV25 and GEM digitization, both the efficiency of the track finder and the probability of identifying all correct hits of a track typically exceeds 90% for electron tracks in both the forward angle and the large angle regions. For the PVDIS and J/ψ configurations, using three time-samples from the APV25 and GEM digitization, the efficiency is about 85% for electron tracks. We are considering introducing some machine learning techniques and optimizations for the detector geometry to further improve the results.

2 SoLID GEM digitization program

The GEM digitization for SoLID simulation is carried out by a c++ package called libsolgem. It relies on the SoLID GEMC simulation as input data, and then produce ADC signals on the readout strips. The output data need to be analyzed by some GEM clustering algorithms before they can be actually used in the tracking reconstruction. The software distribution of the program, and also replay scripts and database files for the three SoLID detector configurations can be found on github (https://github.com/xweizhi/libsolgem).

There are three database files needed to initialize the program, *db_gemc.dat* is responsible for setting up the geometry of all the sectors of each GEM tracker, *db_ratedig.dat* contains all the parameters needed for the digitization and *db_generalinfo.dat* contains parameters needed in the replay script and also collects many hard-coded parameters previously in the program. The explanations for all the parameters in these database files can be found in appendix A-1.

The digitization package at the moment supports ROOT input file format only (through TSolROOTFile). Even though there is also an evio input file reader (TSolEVIOFile), it is not updated for a long time. All necessary information needed for the digitization and MC bookkeeping will

be copied to an data carrier object (TSolGEMData). It will bring the copied information to the digitizer (TSolSimGEMDigitization) where the actual digitization calculations are performed.

The digitizer uses essentially some simple models based on various input parameters to simulate the ionization and avalanche process inside the GEM detector, which may be more easily visualized in Figure 1.

Firstly, the energy deposition information from the GEMC simulation and a Poisson distribution are used to calculate the number of ion pairs produced by the particle. These ion pairs are assumed to be evenly distributed on the path that the particle passes through the drift layer. For each ion pair, the program will estimate the total amount of produced charge based on either a Gaussian or Furry statistical model, and also calculate the avalanche area and the time that the signal reaches the readout board by assuming constant drift and diffusion velocities. Previously in the program, the avalanche area was determined based on a Heaviside model, which gave an uniform charge distribution if within a certain range while 0 if outside. This model gives a reasonable description for clusters that have relatively smaller amplitudes but underestimates the size for those that have larger amplitudes (clusters due to low energy photons for instance). To resolve this issue, a Cauchy-Lorentz model (Figure 2) was developed and used, which should give a better description for both small and large amplitude clusters. Figure 3 shows the correlation between the number of fired strips of a cluster and the total ADC of the cluster (ADC sum from all strips of this cluster). With the Heaviside model, there is no clear correlation. The cluster size seems rather constant regardless how large the total ADC is. On the other hand, it is clear that the larger the total ADC, the larger the cluster size for the Cauchy-Lorentz model.

Secondly, the charge deposit from all the ion pairs are summed by a 2D numerical integral and converted to ADC values based on a GEM shaping function, and signals from different particles are superimposed to produce the final signal on a strip. For the GEM shaping function, previously the one we used was a double exponential form from a COMPASS parameterization, which tends to produce a longer tail compared the cosmic data we took from Hall C. Thus, we decided to return to the single exponential form with a shaping time of 56 ns. However, from the data we saw a correlation between this shaping time and the amplitude of the signal (the larger the amplitude the larger the shaping time). This study has not yet been completed and the single exponential form we are using right now may underestimate the occupancy a bit. But it should be more realistic than the double exponential parameterization from COMPASS. More information about this subject is collected in the appendix A3.

Lastly, signal on all the strips are smeared by the pedestal noise and induced cross talk signals are produced based on the main signal. Currently, the pedestal noise model can simulate a Gaussian random noise on top of a sinusoidal noise with some given periods and amplitudes. This is based on some observations from the data, which is shown in Figure 5. It will introduce some correlations for the noises in adjacent APV samples. However, for all the SoLID detector configuration, since at the moment we assume that there will be at most 3 samples (for PVDIS and J/ψ), the effect of this correlation may not be very obvious. For the cross talk signals, they are induced signals by the main signals from adjacent channels inside the APV25 chips. Adjacent channels inside the chip does not in general correspond to adjacent strips on the readout board. As a result, the induced clusters are usually certain strips away from the main cluster on the readout board, with smaller amplitudes. Currently, the program can make a smaller copy of the main signal, and place it certain strips away from the main signal. An example of the simulated cross talk cluster and its main cluster is shown in Figure 6.

The output of the digitization program is in the format of ROOT. For each event, most of the MC information recorded from the input files and also the digitized ADC signals are saved into a TSolSimEvent object, which will be written directly into the output ROOT file. This means one needs to load the libsolgem shared library before they can analyze the data in the output files. One

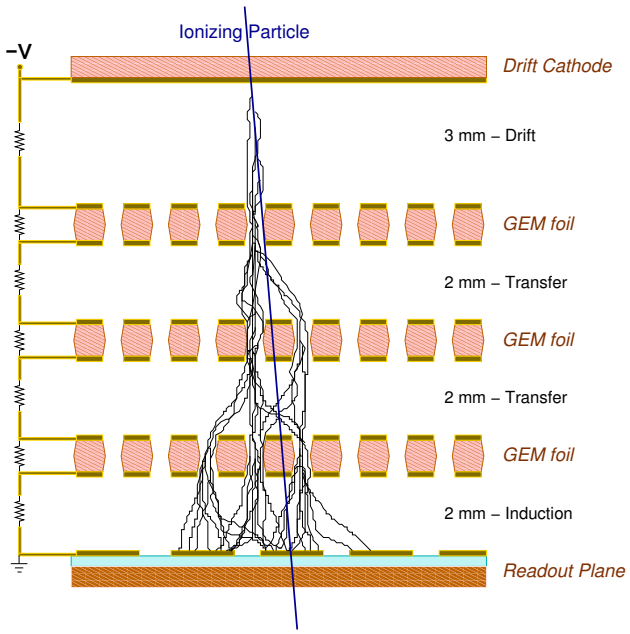


Figure 1: An ionizing particle passes through the GEM detector. Primary ion pairs produced in the 3mm drift layer are amplified by the 3 layers of GEM foils below, by producing large amount of avalanche electrons.

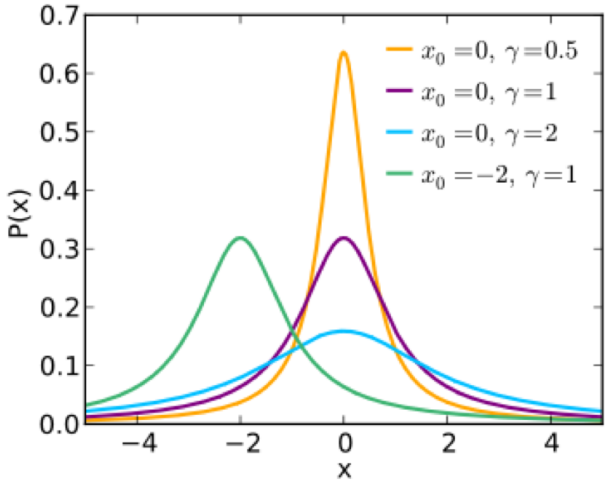


Figure 2: Cauchy-Lorentz distribution with different parameters. Picture taken from `https://en.wikipedia.org/wiki/Cauchy_distribution`

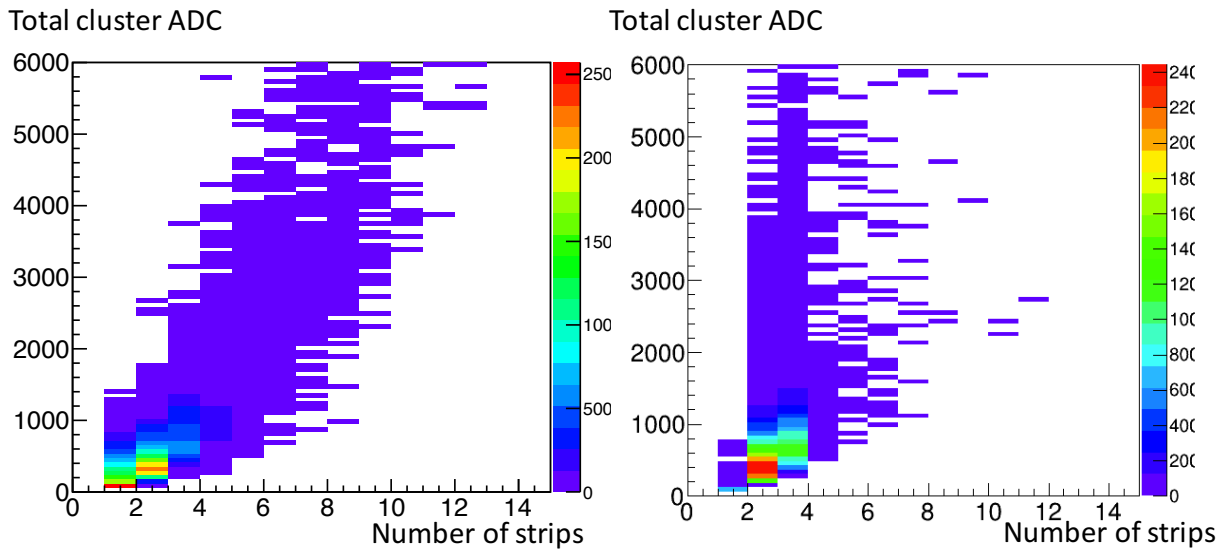


Figure 3: Total cluster ADC as a function of the cluster size. Left plot obtained with the Cauchy-Lorentz model. Right plot obtained with the Heaviside model

simple way to start some analysis scripts is to use the TTree::MakeClass() function from ROOT. Another important thing is that the digitized ADC signal cannot be used directly for the tracking reconstruction. They will need to go through some GEM clustering algorithms for example the SoLIDGEMReadOut::Decode() function in the SoLIDTracking program (see chapter 3).

Figure 4 summarizes the basic structure and how the data flow for the libsolgem program. Lastly, I will discuss some details related to mixing background events with the signal event as this is particularly important for the SoLID tracking reconstruction. When doing the digitization, first of all, a signal event (a DIS electron or multiple particles for instance from the SIDIS and J/ψ generators) is digitized. And then, background tracks will be mixed together with the signal track, but with their timing uniformly distributed within a user-defined time window around the signal track. Currently, these background events are generated by directly shooting electron beam onto the target (so called the beam-on-target events). Based on the current database for PVDIS for example, the background events will be randomly distributed in a window that starts from 200 ns before the trigger start time and 75 ns after (the 200 ns is tunable parameter in the database but the 75 ns is hard-coded), so in total 275 ns. In the replay script, the user needs to manually (and carefully) calculate the total number of background events to be mixed with the signal event, based on the beam current and the size of the time window. For instance, for the PVDIS configuration, the beam current is 50 μA, which means in a 275 ns time window, the beam-on-target events we need are:

$$N = 50\mu A/e \times 275nA \approx 8.6 \times 10^7 \tag{1}$$

where the e is the charge of an electron. And this is defined at the very beginning of the replay script for the PVDIS configuration ($Digitize\_pvdis\_5gem\_qgsp.C$):

Listing 1: define number of background events

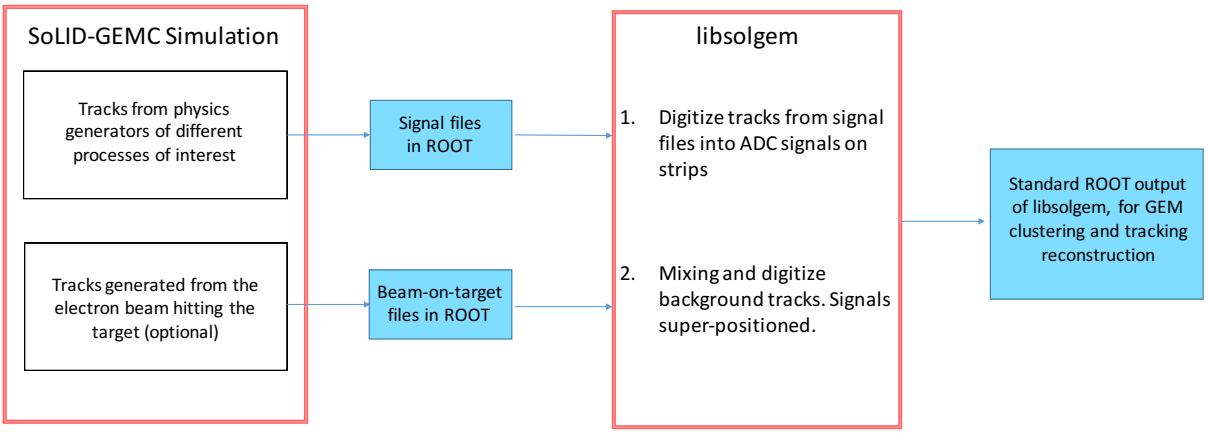


Figure 4: Diagram shows the data flow for libsolgem. Beam-on-target files is only used when one wants to mix background with the signal tracks.

```
1 const double e_charge = 1.60217662e-19; //couloumbs
2 const double beam_current = 50.0e-6; // 50 uA
3 const double time_window = 275.0e-9; //275 ns
4 int nbacktoadd = (5.22438e7/1.e9)* beam_current /
5                  e_charge * time_window * bgratio;
```

There is an additional factor of $5.22438e7/1.e9$ shows up in the total number of background events. This is because when shooting electrons onto the target, most of the time, the incident electrons will just pass through the target without producing any secondary particles. Saving all these empty entries will make the final simulation output file very large and time-consuming to load when doing the digitization. This additional factor is the ratio between the number of non-empty entries over the total number of electron thrown. If we are loading events from a file that keeps only non-empty entries (so called the skimmed file), we will need this additional factor in order to achieve the correct result.

It is very straightforward to obtain this additional factor. From the name of the beam on target file [1], one can get the total number of electron thrown onto the target. For instance, the file I used for the above example is named, "background_solid_PVDIS_LD2_BeamOnTarget_1e9_skim.root", and the number $1.e9$ is obtained from here. Also notice that this type of file has the key word "skim" at the end. For the number of electrons being saved, one can obtain this number from the total entries of the "generated" branch of this file.

Another important thing related to background events is the option of mapping sectors. This option is designed for the PVDIS configuration only and should not be enabled for other configurations. By the design of the PVDIS configuration, there are 30 independent detector systems, each covers a 12° azimuthal angle. Each detector system consists of one GEM sector from each GEM tracker and tracking reconstruction focuses on tracks within the same sector only. Due to this feature, we can map background events from all the sectors into one particular sector so that

[1] At least for the simulations files that Zhiwen Zhao produces, he keeps the total number of electron thrown in he name of the file.

when the map sector option is enabled, one beam on target event is approximately equivalent to 30 that has this option turned off. This drastically reduces the computation time and the size of the output data files.

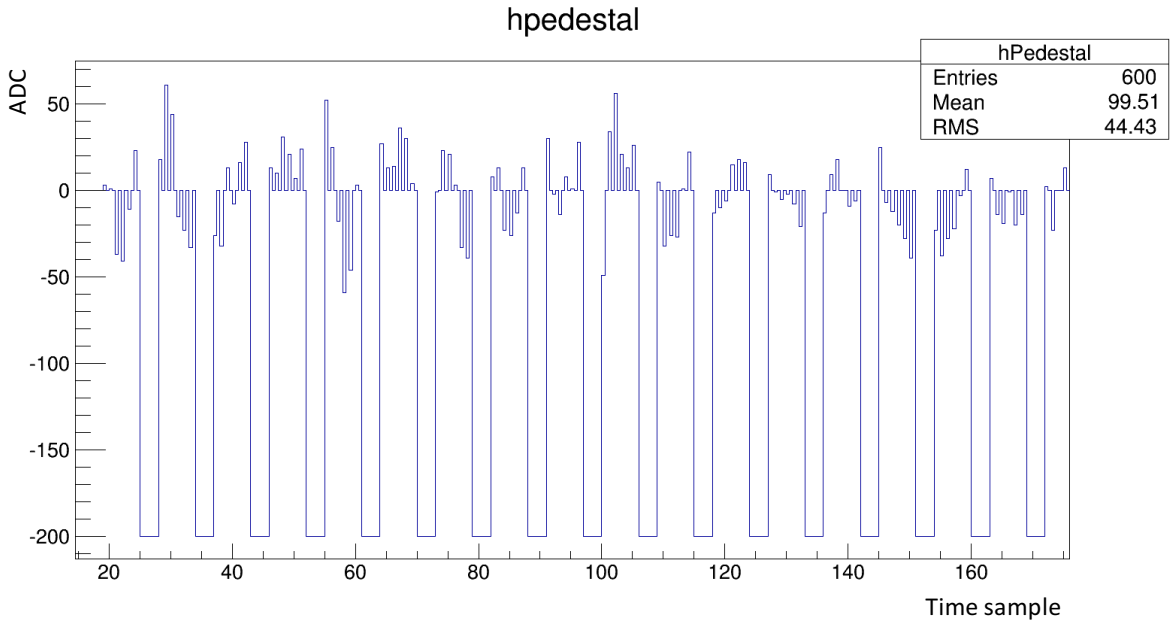


Figure 5: GEM pedestal data measured by Danning Di, Prof. Nilanga Liyanage's graduate student. Each event has 6 time samples, separated by large gaps in the histogram.

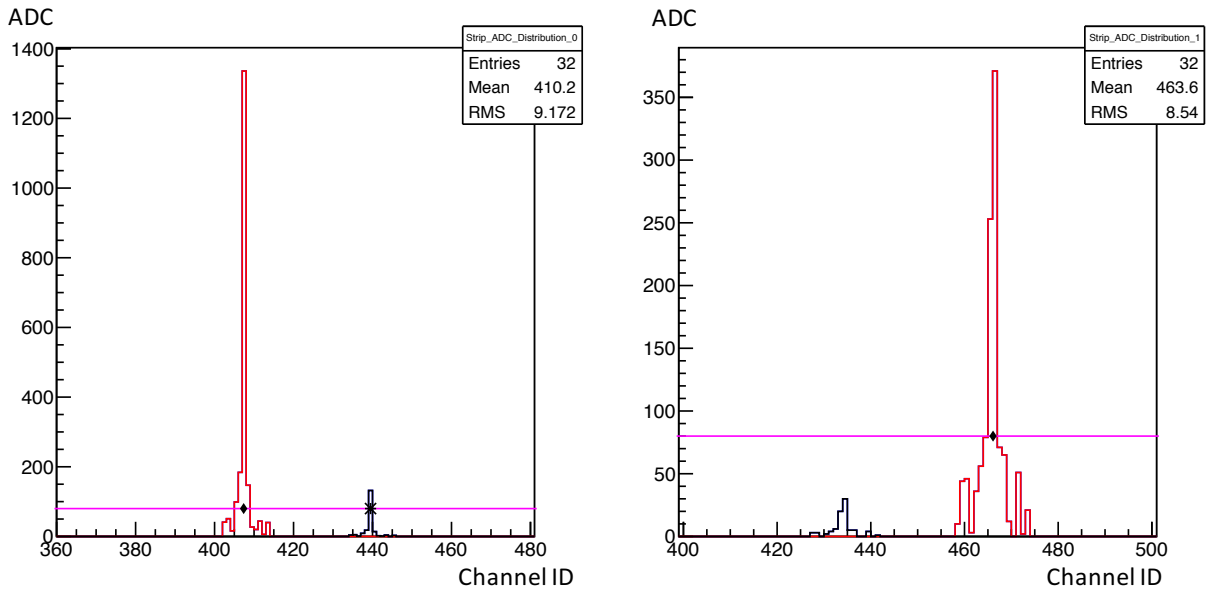


Figure 6: An example of the cross talk cluster (shown in black) and its main cluster (shown in red). The left plot represent the signals on the v strips, while the right one is for u strips. The induced cluster is set to have around 10% amplitude of the main signal, and it is placed 32 strips away. The magenta lines show the threshold cut. Black diamonds shows the reconstructed position for the signal and the black star shows the reconstructed position for the cross talk signal. For the right plot, since the cross talk signal is blow threshold, it is not reconstructed.

3 SoLID tracking reconstruction program

SoLIDTracking is a c++ program that is responsible for the tracking reconstruction for the SoLID detector, using output files from libsolgem. A large potion of this program, particularly the GEM clustering and geometries, is inherited from Dr. Ole Hansen's tracking reconstruction program using the Tree Search algorithm (previously used for the PVDIS tracking reconstruction study with the magnetic field off). And also a significant part related to the tracking reconstruction is based on two other reconstruction programs, namely KalTest [4] and the reconstruction program for the HADES experiment [5]. As shown by Figure 7, there are three major tasks for the program. Firstly, it will load the ADC values for each readout strip from the input data file and perform a GEM cluster reconstruction algorithm. Secondly, the reconstructed hits are passed to the track finder in order to find the right hit combination for the track. And lastly, hits from the same track are fitted in order to obtain estimation for the vertex variables. In this chapter, I will discuss details related to the basic structure of the program and also the major tasks mentioned above.

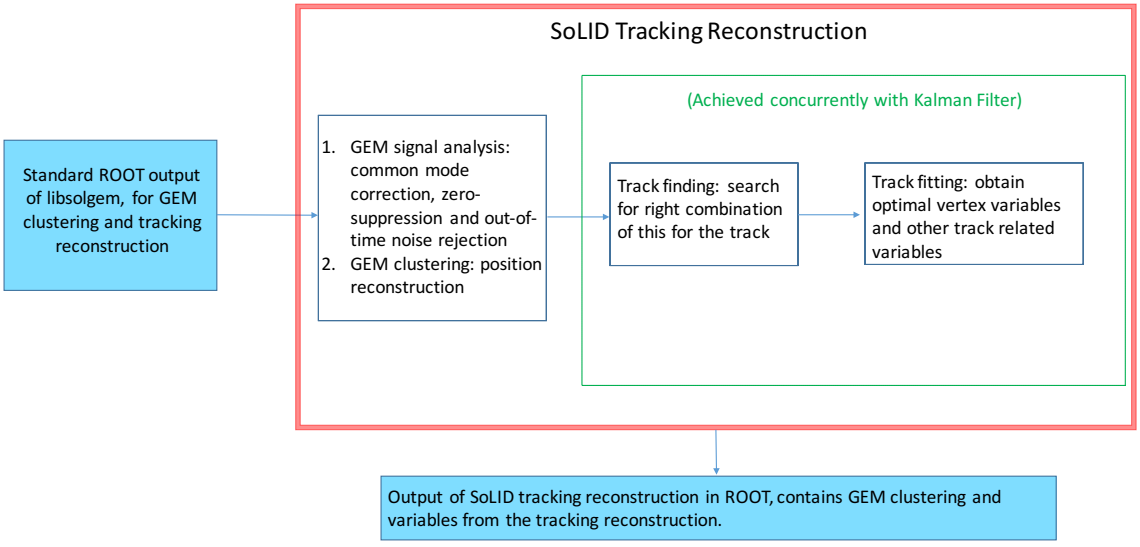


Figure 7: Data flow for the SoLID tracking reconstruction program

3.1 Structure of the program

Similar to libsogem, the SoLIDTracking program also relies on the Hall A Analyzer. The interface between SoLIDTracking and Hall A Analyzer is the SoLIDSpectrometer class, which in some sense correspond to the SoLID detector as a whole. The SoLIDSpectrometer class contains one or multiple SoLIDTrackerSystem objects depending on the specific SoLID detector configuration and each SoLIDTrackerSystem has its own tracking reconstruction. For SIDIS and J/ψ, there will be just one SoLIDTrackerSystem. For the PVDIS configuration, there will be 30 as there are 30 independent detector systems and the reconstruction is done independently inside each detector system. Each SoLIDTrackerSystem contains multiple SoLIDGEMTracker objects, depending on the detector configuration. For PVDIS, there will be 5 GEM trackers while for the SIDIS and J/ψ, there will be 6. Each SoLIDGEMTracker contains a number of SoLIDGEMChamber objects, again depending on the detector configuration. For PVDIS, there is only one chamber following the

above logic while for the SIDIS and J/ψ there will be 30 [2]. Each SoLIDGEMChamber contains two SoLIDGEMReadout objects, which represent the two sets of readout strips for each chamber. This is, in many places of the program, hard-coded and required as this 2D readout system is common for all SoLID detector configurations and many algorithms used in the program were developed by assuming a 2D readout system. This type of structure may not be the optimal option for the execution speed, but it is rather easy to switch from one detector configuration to the other, which simply requires using the corresponding database files and re-compile the program with the detector configuration flag turned on/off.

For each event, the data buffer will first flow into the SoLIDGEMReadout class. The SoLIDGEMReadOut::Decode() method is responsible to load the ADC values from the right crate-slot-channel, analyze the signal and then perform GEM cluster reconstruction. The reconstructed hits are stored inside a TClonesArray object, which is immediately accessible for the track finding algorithms that start with 1D hit, for instance the Tree Search algorithm. For the current track finding algorithm in the program, which is the Kalman Filter algorithm, it requires 2D hits. This means that the two 1D hit arrays from the two SoLIDGEMReadOut classes must be combined before they can be used in the track finding. This is done in the SoLIDGEMChamber::ProcessRawHits() function. The SoLIDGEMChamber can also transform the hits into the lab frame (when hits are reconstructed in the readout, they are usually in some internal reference frame). In the end, all hit arrays from all the GEM detectors are accessible at the SoLIDTrackerSystem level, where the functions related to the track finding step is called in the SoLIDTrackerSystem::CoarseTrack() function. More details related the track finding and track fitting parts will be discussed in their corresponding sub-sections.

3.2 GEM clustering

As mentioned in the previous sub-section, the clustering of GEM detector is perform first in the SoLIDGEMReadOut::Decode() function. In reality, when analyzing ADC data from APV25s, a number of analysis steps must be performed to the raw data before they can be used in the cluster reconstruction. Thus, in this section, I will first discuss details about these analysis steps before I move on to the GEM cluster reconstruction. Lastly, I will discuss about false combinations from our GEM detectors, which appears to be one of the greatest challenges for the tracking reconstruction.

3.2.1 GEM signal processing and noise rejection

As mentioned in the previous sub-section, the clustering of GEM detector is perform first in the SoLIDGEMReadOut::Decode() function. In reality, when analyzing ADC data from APV25s, people usually need to perform the common mode correction to correct the baselines of the data samples. As shown in Figure 8, the baselines of each time sample from an APV25 are randomly fluctuating, and to determine their values, one typically need to take an average of signals from the seemingly unfired strips. A failure to do so can result in an inaccurate threshold cut for the real GEM signals. It will be more difficult to do when the GEM occupancy is high as there will be less seemingly unfired strips. This is extremely crucial for GEMs in the PVDIS configuration as the background rate is not only high but also highly non-uniform. However, in the current digitization, there is no smearing for the baseline and this correction is also skipped in the reconstruction program at the moment. This should be one of the top piorities for the future tracking studies.

After the common mode correction, all signals from all the channels should be at the same baseline. And then, a zero suppression can be applied in order to remove signals below certain

[2] this number is not finalized yet at this stage and so as the final design of the GEM tracker in the SIDIS and J/ψ configurations. Based on pCDR, there should be 20 with some overlapping area. But for simplicity of my study, I assume 30 chambers with no dead area in between. This needs to be updated in the future.

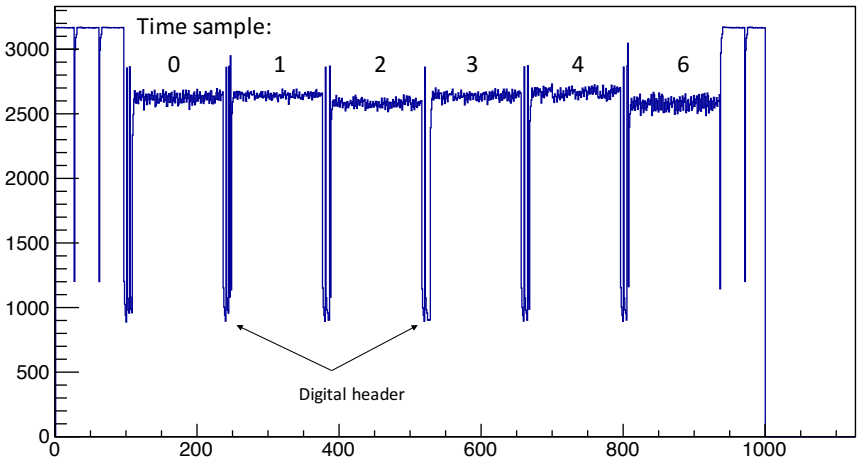


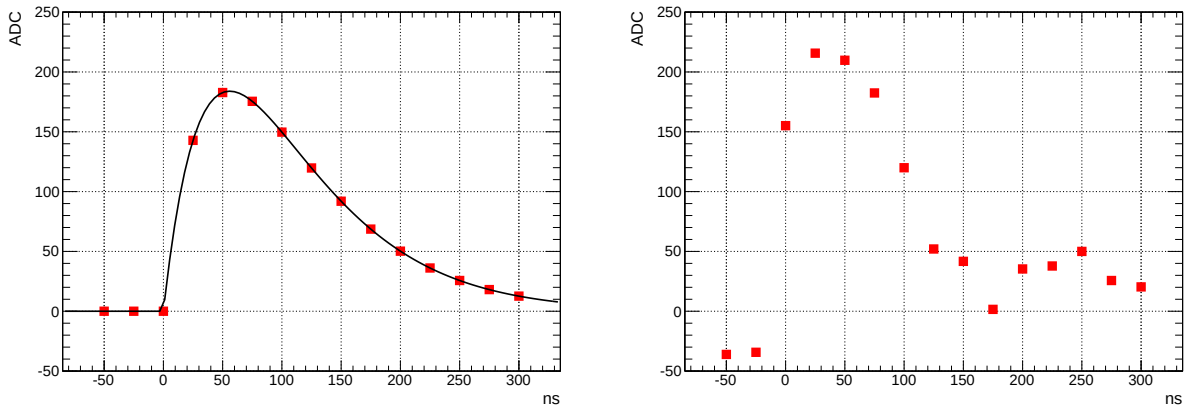
Figure 8: The plot shows raw data from an APV25 for a pedestal measurement using the SRS system. There are 128 channels connected to this APV25 and from each channel, six time samples are recorded. Difference time-samples from these 128 channels are separated by the digital header. One can see that baselines of each time-sample is randomly fluctuating. One can do the common mode correction for each time sample separately to correct this fluctuation, by using the average of signals from all the seemingly unfired channels. For this event, all channels are not fired as it is a pedestal event.

thresholds. Currently, in order to keep the simulation file size small, only signals on strips that are hit by avalanche electrons are smeared by the pedestal noise and saved in the digitization output. So the zero suppression part is also not fully studies, even though the effect is most likely small (assuming the common mode correction can be done accurately enough) as we usually apply a threshold cut that is larger than 4 or 5 σ of the pedestal noise. In the case where there is only one time sample like the SIDIS, the zero suppression is rather trivial. We suppress channels that have ADC data less than the threshold and perform clustering with the remaining signals. When there are multiple samples like PVDIS and J/ψ (with the current design, 3 samples will be recorded), there are different ways to analyze the samples and may also reject significant amount of noises that are not in coincidence with the trigger time (out-of-time noises). For instance, it is a general practice to sum over all the samples in order to average out certain part of the noises as the signal is always positive while some noises may be randomly fluctuating around 0. However, if there is a strong correlation in the pedestal noises such as what we are assuming right now in the pedestal model, the effect of this cancellation may be diminished to some extend [3].

There are other benefits from taking multiple samples. But first of all, one needs to tune the trigger latency and decide which time samples to record. Figure. 9a shows the GEM shaping function and the data points we can record from APV25 (no smearing from pedestal noises and time jitters). It is certain that one would want to record a point at the maxima, which has the best signal to pedestal noise ratio. If one wish to use multiple samples to reject out-of-time noises, then one should also include the leading edge of the signal, which contains most of the timing information of the signal. A balanced configuration is to start recording immediately after the signal arrives [4], so that the first one is close to a pedestal noise, second one is on the leading edge,

[3] In our pedestal model, we have a Gaussian random noise (with mean = 0) sitting on top of a sinusoidal noise whose period (200 ns) is much longer than sampling period (75 ns for 3 samples)

[4] There are other possibilities, such as start recording 25 ns before or after the signal arrives



(a) 1a (b) 1b

Figure 9: The black curve shows the GEM shaping function we are using in the digitization. Red points are the APV sampled shaped signals, separated by 25 ns. No smearing from pedestal noises and time jitters for the right plot while both effects are turned on for the right one. The signals are arriving at $t = 0$.

and the third one is close to the maxima.

There are ways to use these samples to reject out-of-time noises. A commonly used technique is the deconvolution method [1], which has a superb ability to retrieve the original impulse signal from the sampled shaped signal and the ability to separate two very close signals in time,

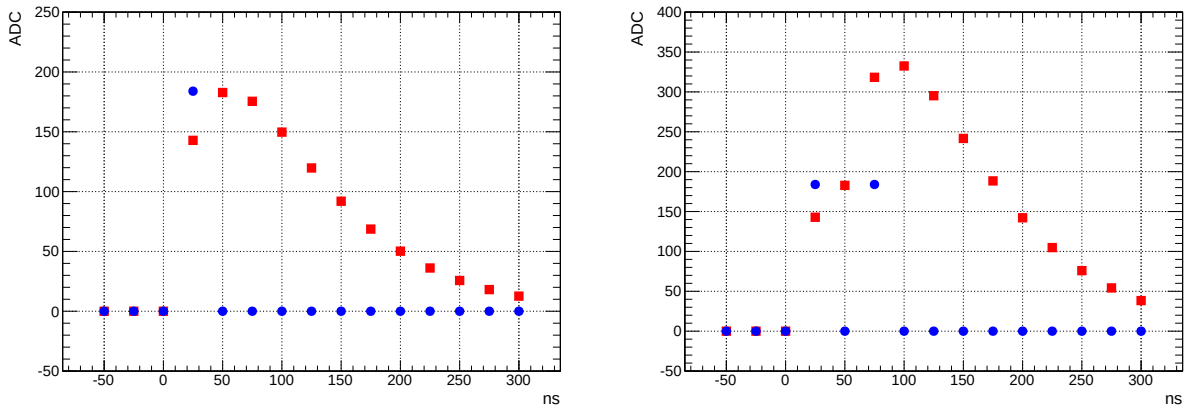
$$s_k = w_1 v_k + w_2 v_{k-1} + w_3 v_{k-2} \tag{2}$$

with

$$w_1 = e^{x-1}/x, w_2 = -2e^{-1}/x, w_3 = e^{-x-1}/x, x = \Delta t/\tau, \tag{3}$$

where τ is the shaping time of the signal, Δt is the sampling period, s_k is the original impulse signal and v_k is the k-th sampled shaped signal. Figure 10 shows a comparison between the deconvoluted signal and the sampled shaped signal, and also its ability to separate two very close signals. In principle, this technique is very ideal for SoLID, which has a high pile-up environment. However, it seems rather vulnerable to the pedestal noise and time jitter. It is expected that for the SoLID GEM detectors, we will have a relatively high pedestal noise due to long readout strips. And the APV25 chips always have a $\pm 12.5ns$ time jitter due to that fact that their internal clocks cannot be synchronized with random triggers. Together with some other time jitters due to other electronics, the total time jitter we assumed in the current simulation can be as large as ± 25 ns for many events. Consider we only have 3 time samples at most (sampled every 25 ns), this time jitter is rather significant. In the ideal case, such as in Figure 10a, one can simply use the amplitude of the deconvoluted sample at $t = 25$ ns to reject out-of-time noises, as signals arriving at $t < -25$ ns or $t > 25$ ns will have no contribution to the amplitude of the deconvoluted sample at that point. However, when there is a time jitter, adjacent time bins start to share the amplitude of the deconvoluted sample, and if the jitter is larger than ± 25 ns, the time bin is completely shifted, as shown in Figure 11. This means that we can not focus on just one deconvoluted sample. Even though one may argue that it would be a rare case that the deconvoluted amplitude is completely shifted to adjacent time bin, as long as the amplitude is shared, the signal to pedestal noise ratio is lower for the time bin we focus on, which may lead to lower GEM efficiency [5]. Eventually we may

[5] I also did not finalize a way to deal with pedestal noise when doing the deconvolution. It is possible that setting all signals below threshold to zero will lead to better result in deconvolution. But this has not been fully tested.



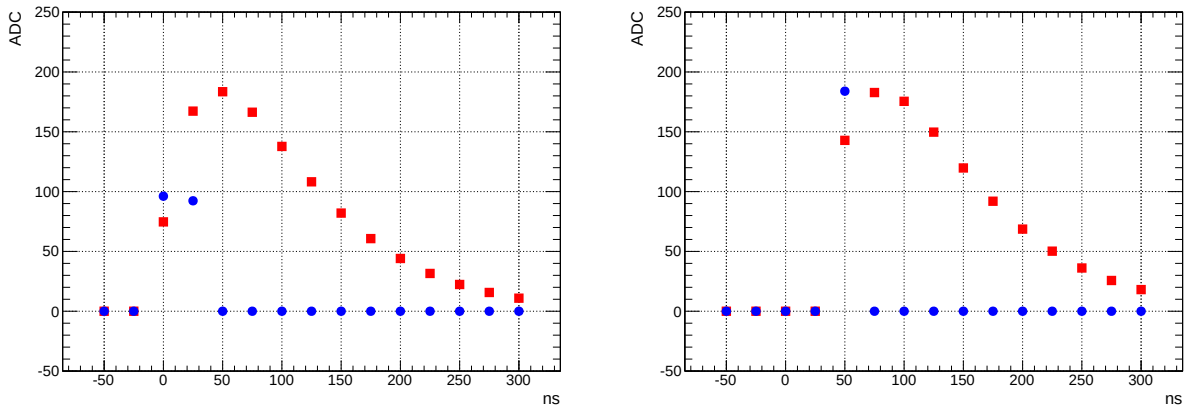
(a) 1a (b) 1b

Figure 10: Left plot shows the sampled shaped signal as the red data points and the deconvoluted signal as the blue data points). A spike-like signal retrieved after the deconvolution. Red data points in the right plots are produced by superpositioning two signals with the same amplitudes (same as the one in the left plot) but one arrives at $t = 0$ ns the other at $t = 50$ ns. After deconvolution, two separated spike-like signals are obtained with similar amplitudes as the deconvoluted signal in the left plot.

have to use some other methods like summing over two or three adjacent deconvoluted samples. The second difficulty in our case is that we only have three sampled shaped signals at most, while equation 2 suggests that to get the correct deconvoluted sample at k, we need the current and two previous sampled shaped signals. This means that we only have enough information to deconvolute the last sample correctly while the first and second have to rely on some assumptions about the shaped signals before them (which are not recorded), for instance, the shaped signals are zero before the first sampled one. This is indeed a good assumption if the signal arrives at $t \geq 0$ ns and without considering the situation of pile-ups. But it can lead to some unexpected deconvoluted results for signals arrives at $t < 0$ ns, as the assumption is no longer valid. One way to overcome this is to start recording samples 25 ns before the signal arrives so that the first two samples are pedestal noises and the third one is on the leading edge (assuming no pile-ups). In this case, the third deconvoluted sample will have the full amplitude if the signal arrives on time and it can always be correctly deconvoluted. However, a major drawback of this approach is that the maxima of the shaped signal is not among the recorded samples, and due to time jitter, we may completely miss the signal. In a word, the major challenge for the deconvolution method, is the time jitter problem, which is the main reason why I have not figured out a way to fully utilize it yet [6].

Currently the out-of-time noise rejection in the program mainly relies on another method, which simply compares the relative amplitude between the samples. This technique is described in page 51 - 53 in Frank Simon's thesis [2]. It is as simple as placing some 2D straight line cuts on a plot with y axis as the ratio between sample 0 and sample 2 and x axis as the ratio between sample 1 and sample 2. It is rather easy to tune and use. In the ideal case such as the one shown in Figure 9a, the three samples are in increasing order. When there is a pedestal noise and time jitter, this relation may be slightly violated as shown in Figure 9b. One can use a set of simulation signals and study their relative ratios in order to find the suitable cuts. However, this method may not be very efficient in some circumstances like to separate two very close signals in time. For instance, it will be unlikely to separate the two signals in Figure 10b. If we decide to reject this type of

[6]But it should still be kept as a possibility. It is a very powerful tool and maybe if we change the sampling setup a bit or someone figures out a better way to handle the jitter and noises so that it is usable again



(a) 1a (b) 1b

Figure 11: Red squared show the sampled shaped time samples, and blue dots are their deconvoluted samples. The left plot shows a signal arrives at $t = -10$ ns relative to the trigger start time and the right plot is for a signal arrives at $t = 25$ ns. In the left plot the deconvoluted amplitudes are shared by the first and second samples after $t = 0$ ns and in the right one, the amplitude is completely shifted to the third one.

signals, we will have lower GEM efficiency for a high occupancy area as it will be a rather common case when an on-time signal is overlapping on some previous signals. The subject of rejecting out-of-time noise is very crucial for SoLID. A lot of difficulties in the tracking reconstruction can be solved if we can improve the results of this rejection. So this subject should be studied further more.

3.2.2 GEM cluster reconstruction

Once the signals are analyzed by the above processes, the remaining one will be clustered and given estimation for the hit position. The clustering algorithm is written by Dr. Ole Hansen. It is a simple peak-and-valley splitting algorithm. It will first look for peaks among all the strips, if there are two close peaks with a significant valley in between, the two peaks are split with half of the ADC in the valley distributed into each of the peak. After this step, a center of gravity method is applied to calculate the hit position of each cluster:

$$x = \frac{\sum_{i=1}^{N} q_i x_i}{\sum_{n=1}^{N} q_i} \tag{4}$$

where x is the estimated position of the cluster, N is the total number of strips the cluster has, q_i is the ADC value from strip i, and x_i is the central position of strip i. However, this algorithm will not work if part of the strips are segmented. There will be at least some modifications needed especially in the transition region between the segmented strips and unsegmented ones.

3.2.3 Rejection of false combinations

The above clustering algorithm is applied to each set of readout strips separately, so the output are two arrays of 1D hits that have no correlation in between. At some point of the reconstruction steps, these two sets of hits must be combined to form 2D hits [7]. Some track finding algorithm

[7] Or 3D hits. Our GEM does not measure directly the z coordinate of a hit, but usually we use the center of the drift layer as the z of a hit, and it is common for all the hits on a GEM tracker.

does not require 2D hits like the Tree Search algorithm, so the combine of hits can actually be done after track finding. However, for the current track finding algorithm in the program, the 1D hits need to be combined in advance. So in the last part of this sub-section, I will discuss a bit more details related to combining 1D hits.

It is easy to imagine that there will be a lot of false combinations (as shown in Figure 12) if we simply exhaust all the possible combinations between the two sets of 1D hits [8]. This is in fact,

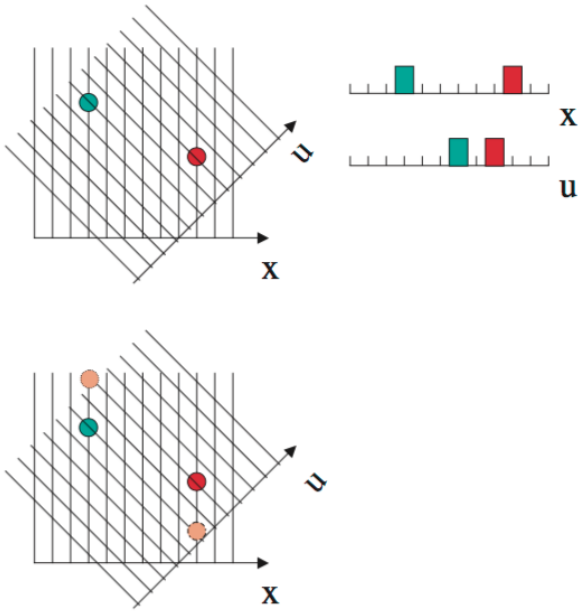


Figure 12: Two particles (shown as green and red) are hitting the chamber resulting in two reconstructed hit on each set of readout strips, which leads to four 2D hits after combining. However, the top-left one can be rejected because there is no intersection between the two sets of strips at that position. Picture taken from [3]

a common problem for any 2D readout detectors. There are ways to reject false combinations. A trivial one is to check whether the combined hit is coming from strips that have physics intersection on the readout board. This may not be very efficient especially if the hit density is high in a small region. But it is certainly correct and safe. The second thing one can check is the charge asymmetry:

$$A = \frac{q_u - q_v}{q_u + q_v} \tag{5}$$

where $q_u(q_v)$ is the total charge of a cluster on the $u(v)$ strips. If the readout board of a GEM chamber is designed such that the u and v strips cover equal area, then in principle, we would expect the asymmetry A to be approximately 0 as the avalanche electrons have equal possibility to hit the two sets of strips. However, if the two 1D hits are coming from different particle, there is no such correlation and we may cut on this variable in some ways to reduce to number of false combinations. However, the effect of this type of cut is much reduced if the occupancy is high. In this case, there is a high possibility that a hit is overlapping on the other, which result in a smearing on the total charge of a cluster (see Figure 13 and Figure 14). Thus, this cut needs to be

[8] For example, if there are N particles hitting the chamber, result in N reconstructed hit on u and N on v strips, there will be $N \times N$ different combinations in total. Much more than what it supposed to be if N is large, which happens to be the case for SoLID, due to high background rate.

applied carefully. Currently, instead of applying the cut at the combination level, it is applied only after tracking reconstruction, by requiring that a reconstructed track must have certain number of hits that have good charge asymmetries (say 2 out of 5 from a PVDIS track).

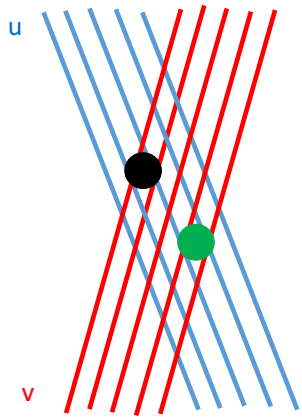


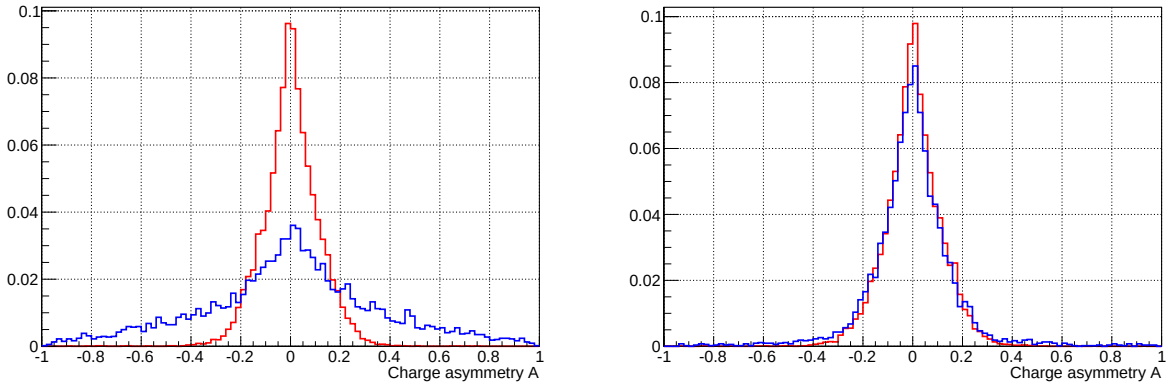
Figure 13: Two particles hit the chamber, results in two very close peaks on the u strips, but two separated peaks on the v strips. There is a high possibility that the reconstructed total charge on the u strips is not as accurate as that on the v strips, due to the cluster overlapping and spliting in the reconstruction. Thus, their charge asymmetries are smeared.

The ultimate way to deal with the issue of false combination is certainly on the hardware level, for instance using shorter strips so that there are less intersections between strips. And false combinations certainly cannot be form between strips that have no intersections.

3.3 Track finding and fitting

The track finding algorithm that we have in this program is based on Kalman Filter (KF). A very detailed introduction and derivations for all the formulae of the theory are summarized in the reference manual of the KalTest program [4]. Thus, most of the theoretical related subjects of KF will not be covered in this note. KF is also a χ^2 minimizing method. Compared to the normal global χ^2 fitting methods that give one set of optimal parameters in the end, KF allows the track parameters (or state vector) to evolve along the trajectory by collecting local information such as the non-uniformity of local magnetic field, variances due to multiple scatterings when particles pass though materials and so on. Thus, it is possible to achieve optimal estimates for track parameters at every measurement site. To use KF, firstly one need to calculate a set of initial estimate for the track parameters. For example, if the track is approximately helical, then one can use three hits from three different tracking detectors to get the initial estimation. And then, the track parameters and the corresponding covariance matrix need to be propagated toward the next detector. The propagation can be as simple as some geometrical calculation especially if the track follows some simple geometrical models like a straight line or helix. For SoLID, due to non-uniformity of its magnetic field [9], the field map and a field swim method is used in order to achieve a better accuracy in the propagation. During the propagation, another covariance matrix associated with multiple scatterings is calculated based on the material that the particle passed through. Once the parameters and the covariance matrices are propagated to the next detector,

[9] Even though the magnetic field is reasonably uniform in the central area of the solenoid, in all detector configurations we considered, the tracks are always going to pass through some spaces with very non-uniform field. For example, for SIDIS, the target is placed outside of the solenoid and for PVDIS, most of the tracking detectors are in the fringing field.



(a) 1a (b) 1b

Figure 14: Charge asymmetries on two GEM trackers in the SIDIS configuration. Clusters showing in the histogram must contain contribution from the signal particle. The left plot is for the second GEM tracker, which has the highest background rate in the configuration. The right plot is for the sixth GEM tracker, which has the lowest background rate. Red histograms show the charge asymmetries when there is no background events. Blue histograms obtained by assuming full background from the target. The red distributions are rather similar as there is no background contamination and the reconstruction is performed similarly on these two GEMs. There is a very drastic difference between the blue distributions due to cluster overlapping and splitting under different background environments.

the next measurement can be used to obtain an optimal estimation for the track parameters and the covariance matrix at the new location. This procedure can be applied iteratively until the track reaches the last tracking detector.

KF as described above can be used for track fitting. But with some small modifications to the algorithm, it can also be used to do track finding, thus achieve a concurrent track finding and track fitting. Once the propagation is finished so that the track just reaches the next tracking detector, a predicted hit position can be calculated. We can judge whether hits on the next detector agree with the prediction based on the detector resolution and also the uncertainty in the prediction. In addition, the χ^2 increment after adding the new hit can be used as another hit selection method.

The above summarized the basic methodology of the tracking finding and fitting algorithm. Next, I will describe more details about each steps that were mentioned. Firstly, it is very important to decide what track parameters we should use to describe a track in the SoLID detector. Again, having in mind that a SoLID track will typically pass through both uniform and fringing fields, it may be a better idea to pick a track parameterization that is as generic as possible. Thus, we decided to use the following track parameterization:

$$\overrightarrow{a} = (x, y, t_x, t_y, q/p)^T \tag{6}$$

where x and y are the two Cartesian coordinates of the track at a given z position (z is along the beam), t_x and t_y are the slopes in the $x-z$ and $y-z$ planes respectively at that given z position, and q/p is the charge of the particle over its momentum magnitude. The propagation of the track parameterization and its covariance matrix can be done by using the 4-th order Runge-Kutta method, which is a very stable differential equation solver. For more details related to this track parameterization and the corresponding Runge-Kutta method, please refer to the master thesis of Erik Krebs [5], in which he gives very detailed introduction and derivations for all the related formulae.

3.3.1 Initializing the tracking finder

To initialize this track parameterization, we need to search for a short track segment (or called the seed) and use its information to make initial estimation for the five track parameters. It is advised to use hits on downstream detectors as the hit density there is much lower than the upstream ones. For this reason, it is rather simple to initialize a track in the PVDIS configuration as the tracks are approximately straight in the downstream tracker region. One way is to use two hits from two downstream detectors and assume that the track is straight in order to get the first four parameters in the track parameterization. And since we are interested in only electrons for PVDIS, we can use the energy measurement from the calorimeter to get the last parameter. In addition, when searching for seeds, one needs to keep in mind that the GEM efficiency is around 95%, which means if we only look for seeds coming from the last two GEMs, we will limit the tracking efficiency to around 90%. Thus, one may need to widen the seed search region to something like two out of three from the last three GEMs. This initialization is slightly more complicated for the SIDIS and J/ψ configurations. On one hand, the GEM trackers are all in the strong and uniform magnetic field region, which means the track is approximately helix and we will need three hits to calculate the initial track parameters if we are using the helix equation. On the other hand, if we consider the limitation of GEM efficiency, we may have to choose three out of four down stream detectors. This will drastically increase the possible combination of seeds and also the computation time needed to analyze these seeds. However, it is found that with some polynomials obtained with the help from the simulation, we can only use two hits to initialize a track for SIDIS and J/ψ as well. For more details about this part, and also some parameters obtained from the simulation, please refer to the appendix B-3.

Before we extend the seeds and try to connect hits from all the upstream detectors, it may be a good idea to check whether some seeds are unlikely coming from the target or lead to hits or other type of detectors such as the MRPC and the calorimeters. We can kill significant amount of fake seeds before putting them into the KF loop, which always involve time-consuming matrices multiplications and inversions. Some fast Runge-Kutta propagations (or something simpler), with large step size, can be use to quickly propagate the track segment back to the target and to the downstream detectors to check the consistency. Figure 15 shows some kinematic variables and predicted hit positions using only the seeds in the SIDIS forward angle region. Some wide cuts can be applied on these variables to suppress the amount of fake seed.

Another thing worth mentioning about the initialization is that the final result of KF may depend on the initial estimation for the track parameters. For example, in the extreme case, if we are doing backward tracking from the downstream GEM to the upstream one but set the initial p_z of the track to be positive so that it starts to propagate in the downstream direction, KF will fail as it is almost impossible to find the prediction on the next GEM. In some cases, one may need to refit the track in order to get rid of this dependency [10]. In addition to the track parameters themselves, the initial values we set in the covariance matrix may also make a difference. For instance, if we set them to be infinitely small, the final fit result will be very close to the initial track parameters we set and all the measurements will have very little effect on the track parameters (unless errors we set for the measurement are also similarly small). Typically, one will only set the diagonal elements of the initial covariance matrix, with somewhat larger errors than the errors we actually expect. In this case, the fit is less restricted by the initial guess. The only exception I found in my studies is the initial momentum in the PVDIS configuration. It is found that assigning an error that is similar to the actual expected value gives slightly better result (I am using $0.1/\sqrt{E}$ for the calorimeter resolution). One possible explanation is that the magnetic field is very weak in

[10] take the fitted result from the first fit and initialize the second one for instance, may not be the best way, but works.

the end-cap region of SoLID and there is no much improvement for the track momentum before using the field between the target and the upstream GEM. In this case, the error in the calorimeter measurement maybe actually better than the error in the momentum we can get from the hits on the last few GEMs.

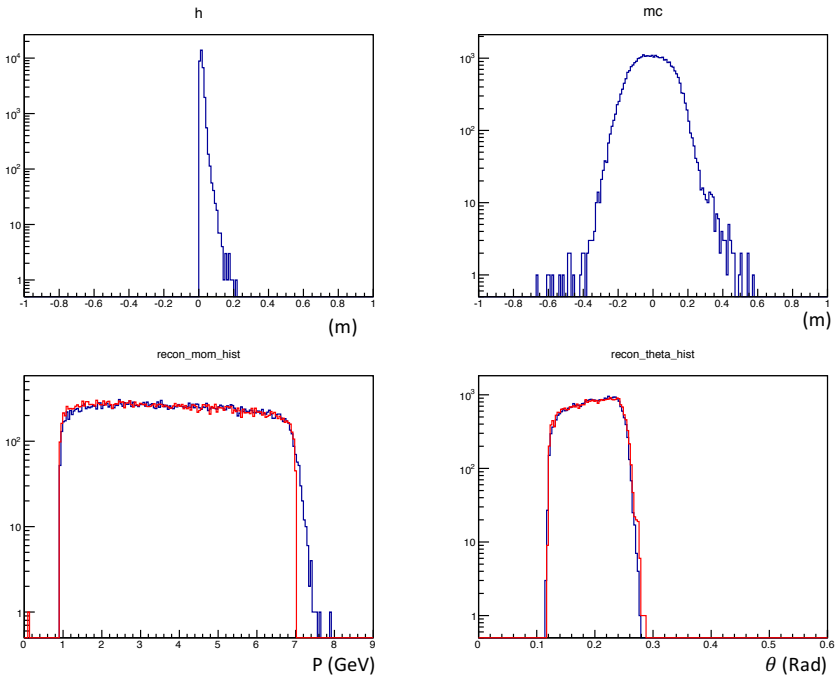


Figure 15: The top-left plot shows the r coordinates difference between the hit measured by the forward angle calorimeter and the predicted hit position using seeds. The top-right plot shows difference between the target center and the reconstructed vertex z using seeds. The bottom left plot shows the comparison between MC momenta (red) and reconstructed momenta (blue). The bottom right plot shows the comparison between MC scattering angle (red) and reconstructed scattering angle. In both case, using seeds only. High precision is not required at this stage and cuts on these variables can be fairly wide. The purpose of this simple reconstruction is to suppress the amount of fake seeds in order to speed up the program.

3.3.2 Track follow

The rest is simply ask the KF to look for hits along its way of propagation. The method is exactly the same as described above. If a hit is reasonably close to the prediction of KF and the χ^2 increment is reasonably small after adding the hit, the hit is accepted and KF will continue look for the next hit. At first, KF may not be very stable and the cuts for the prediction and the measured position may require some manually set values [11]. At a latter stage, it is found that the diagonal elements for x and y in the predicted covariance matrices can be used as a reasonable criterion (see Figure 16 and Figure 17).

At some point, if the track has missed enough hits, it should be terminated as it is becoming more and more unlikely that we can reconstruct a good track. Currently, the program allows only one missing hits in total. In other word, we are requiring four out of five GEM trackers have a hit found for the PVDIS configuration, three out of four for the large angle region of SIDIS and J/ψ and four out of five for the forward angle region of SIDIS and J/ψ. In the end, all tracks that

[11] KF is under-determined before the third hit as there are five parameters and each hit adds 2 degree of freedom.

satisfy certain selection conditions are propagated towards to target, where a special hit from the beam position monitor (BPM) is added. We assume the BPM can provide the x and y position of the beam spot with a resolution of around 300 μm. The z position of this hit is determined by the reconstructed vertex z. Thus, the last point has very little improvment for the polar angle and vertex z but can improve the azimuthal angle and momentum resolution, particularly for the PVDIS configuration [12].

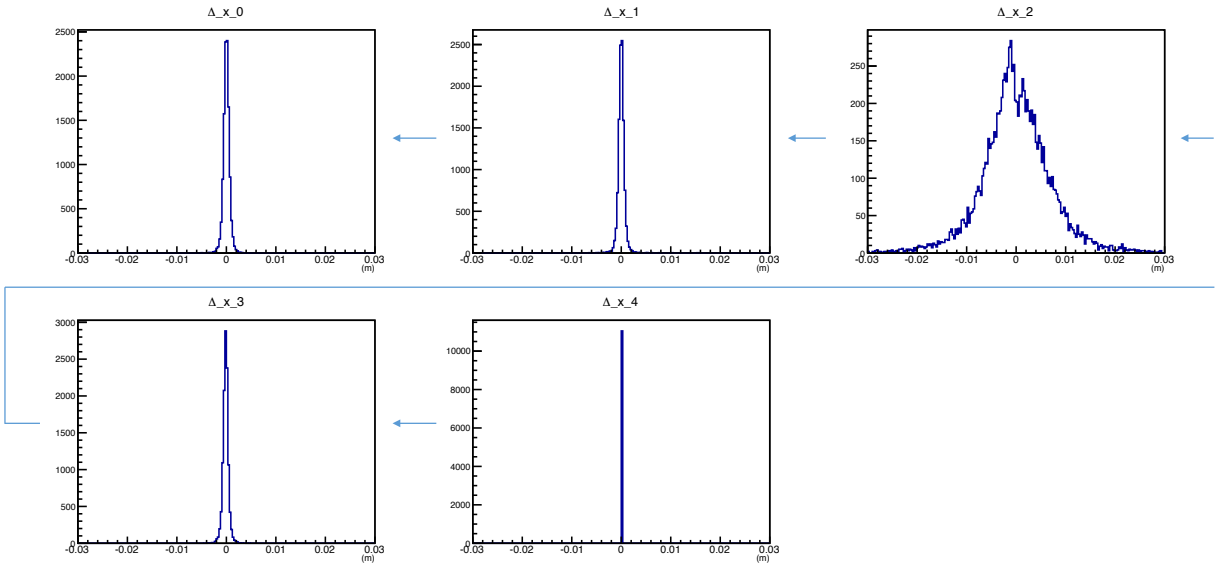


Figure 16: Plots show the x coordinate difference between the predicted hit positions and the actual measured hit positions for tracks in the PVDIS configuration. The blue arrows indicate the direction of track finding, which is from the most downstream one to the upstream direction. The measured hit on the most downstream GEM is used to estimate the initial position of the track, which is the reason for the delta function in the last plot. The width of the distribution depend on several factors, such as materials and distances in between the trackers.

3.3.3 Final selection on reconstructed tracks

At the very last step, all tracks are propagated back to the interaction vertex. A set of final selection rules can be applied to further suppress possible wrong tracks. For instance, we can propagate the track to the calorimeter and MRPC again to check the agreement. The previous check was done at the end of seeding, at which stage our knowledge about the track was much worse. It can be shown that even in the SIDIS forward angle region, where the calorimeter is over 1 m downstream compared to the last GEM in the configuration, the difference between the calorimeter reconstructed hit and the predicted hit from current stage is completely dominated by the calorimeter position resolution, as shown in Figure 18. If without smearing on the calorimeter hit, the width will be just a few mm. So improving the position resolution of the calorimeter can also help the selection in the tracking reconstruction. The other thing one might compare with the

[12] there could be better ways to utilize the measurement from the BPM, but this is what I come up with at the moment.

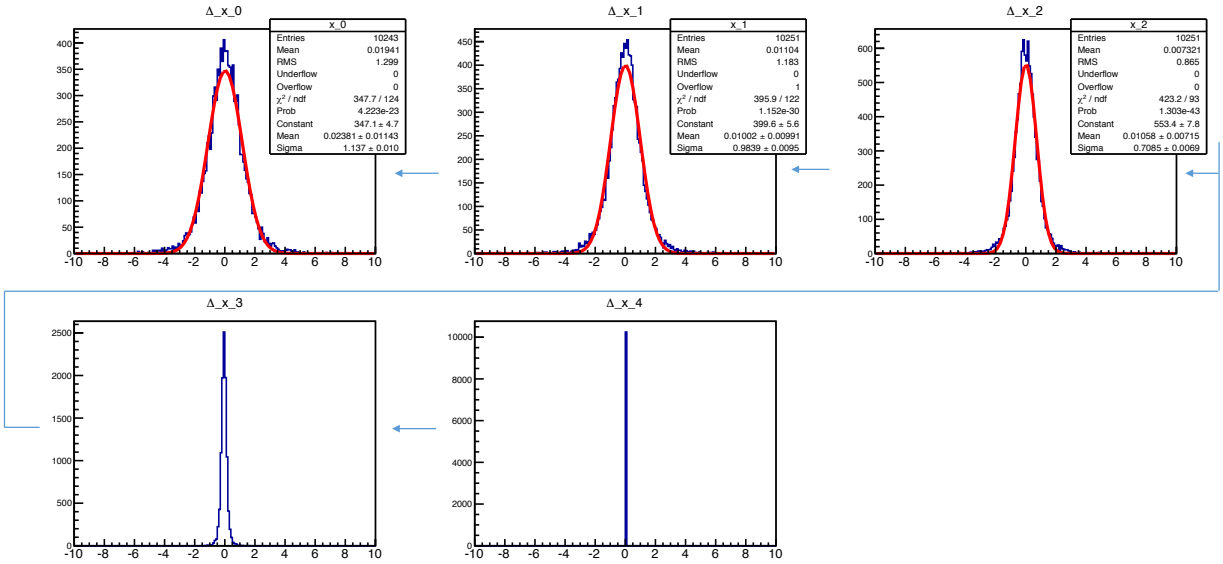


Figure 17: Plots show exactly the same set of events as in Figure 16, but the difference is divided by the square-root of the diagonal matrix element in the predicted covariance matrix for the predicted track parameter x. They show that after adding a few hits, the square-root of this matrix element becomes very close to the actual width of the distributions shown in Figure 16. And we can use this value to judge whether a hit on the next GEM can be accepted.

calorimeter is, for electron tracks, one may be able to compared the reconstructed momentum from the tracking with the energy measurement from the calorimeter. But one need to keep in mind that, without a GEM tracker and a sufficient field integral, there will be no improvement for the local reconstructed momentum. For instance, during the propagation from the last SIDIS forward angle GEM to the calorimeter, our knowledge about the momentum of the track basically stays the same as the one on the last GEM. So even if the electron has a significant radiative energy loss on the way, we may not be able to notice it. And there could be a significant radiative tail if we compare the reconstructed momentum and the measured one by the calorimeter. Similar argument may be applied to the PVDIS configuration as well, as there is essentially no field integral after the third GEM. Another final selection rule I applied at this stage is that for reconstructed tracks that share a common hit, only the track that has the largest number of hits are kept. If there are more than one track have the same number of hits, then the one with the smallest χ^2 will be kept [13].

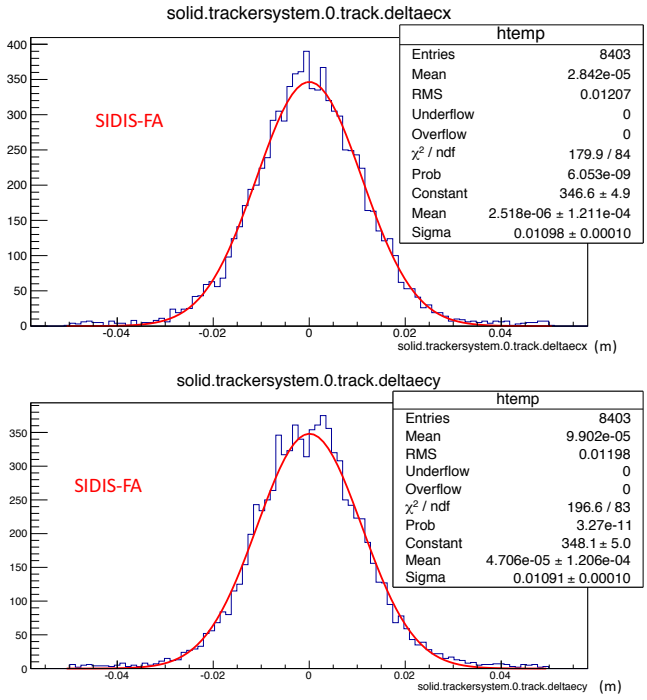


Figure 18: Plots show the differences between the calorimeter measured x and y coordinates and the predicted ones from KF, for tracks in SIDIS forward angle region. The position resolution of the calorimeter is assumed to be 1 cm.

3.4 Example of replay script

It is rahter straight forward to run the program with a very simple replay script. Here is an example of replay script for the SIDIS configuration. Scripts for other configurations can be exactly the same as most of the information are stored in the databased files. There is only one line needs to be modified for other configurations. Line 13 tells the program to set up a system with 1 SoLIDTrackerSystem, which has 6 SoLIDGEMTrackers. Each SoLIDGEMTracker has 30

[13] This selection method is inherited from Xin Qian. In fact, I have no much idea why it works, but it seems work quite well most of the time. Maybe we should do some studies about this part and see if there are something we can improve.

SoLIDGEMChambers and each SOLIDGEMChamber has 2 SoLIDGEMReadOut. In the case of PVDIS when the map sector option is turned on, the last four parameters should be 1, 5, 1, 2.

Listing 2: Replay script example for SIDIS

```
using namespace std;
void test_script()
{
  gSystem->Load("libSoLIDTracking.so");
  gSystem->Load("libsolgem.so");

  TSolDBManager* manager = TSolDBManager::GetInstance();
  manager->LoadGeneralInfo("db_generalinfo.dat");
  manager->LoadGeoInfo("db_gemc.dat");

  THaInterface::SetDecoder( TSolSimDecoder::Class() );
  THaApparatus* SOL =
  new SoLIDSpectrometer( "solid", "SoLID spectrometer", 1 , 6, 30, 2);
  gHaApps->Add( SOL );
  TString db_prefix = SOL->GetName();
  db_prefix += ".trackersystem";
  gHaTextvars->Add( "DET", db_prefix.Data() );
  gHaTextvars->Add( "APP", SOL->GetName() );

  THaAnalyzer* analyzer = new THaAnalyzer;
  analyzer->EnableBenchmarks();
  analyzer->SetOdefFile("solsim.odef");
  analyzer->SetOutFile("output_file_name_replayed.root");

  analyzer->SetCrateMapFileName("db_solsim_cratemap.dat");
  analyzer->SetVerbosity(3);
  THaRunBase* run[5];
  for (Int_t i=0; i<5; i++){
   run[i] = new TSolSimFile(Form("input_file_name_digitized_%d.root", i),
                            Form("test %d", i));
  }

  analyzer->(Init(run));
  for (Int_t i=0; i<5; i++){
   if( i>0 )
     run[i]->SetDate(run[0]->GetDate());
  analyzer->Process(run[i]);
  }
  analyzer->Close();

}
```

4 Results

4.1 GEM occupancy

In this part, I will summarize my latest results on the GEM occupancies for PVDIS, SIDIS-^{3}He and J/ψ. In all cases, occupancies are obtained at the proposed beam currents, which means 15 μA for SIDIS-^{3}He, 3 μA for J/ψ and 50 μA for PVDIS. We use Zhiwen Zhao's beam-on-target simulation files as background events, which in all cases have their timings randomized in a time window that is 200 ns before the trigger start time and 75 ns after, so in total 275 ns.

For SIDIS, since at the moment we assume we will take only one time-sample from APV25s, we can only obtain the raw occupancy, which means the number of strips that has signals above certain thresholds (fired strips), over the total number of strips on a GEM tracker. For PVDIS and J/ψ, some out-of-time noises are rejected by the noise rejection algorithm discussed in the GEM clustering section. So in addition to the raw occupancy, we can also obtain the noise rejected occupancies, which means the number of fired and un-rejected strips over the total number of strips on a GEM tracker. Another thing worth mentioning is that in this study, each GEM tracker in SIDIS and J/ψ configurations has 30 sectors and with no dead area in between, which maybe fairly different from the final design. As a result, the occupancy number may change significantly.

The following table summarizes the raw occupancies for all GEM trackers in the SIDIS-^{3}He case, using 4σ threshold cut (σ = 20.7 ADC, which is the width of the pedestal noise assumed at the moment):

Tracker	total number of strips (u + v) per sector	raw occupancy (%)
1	906	2.37
2	1020	7.98
3	1166	3.40
4	1404	2.24
5	1040	2.03
6	1280	1.52

The following table summarizes the raw and noise rejected occupancies for all GEM trackers in the J/ψ case, threshold cuts are the same as for the SIDIS-^{3}He:

Tracker	total number of strips (u + v) per sector	raw occupancy (%)	noise rejected occupancy
1	906	7.68	4.65
2	1020	14.4	9.28
3	1166	8.82	5.49
4	1404	7.00	4.30
5	1040	5.92	3.78
6	1280	4.58	2.95

The raw and noise rejected occupancies for all GEM trackers in the PVDIS configuration are summarized in the following table, threshold cuts are the same as for the SIDIS-^{3}He:

Tracker	total number of strips (u + v) per sector	raw occupancy (%)	noise rejected occupancy
1	1156	21.17	9.97
2	1374	10.35	5.11
3	1374	8.81	4.42
4	2287	3.07	1.64
5	2350	2.79	1.50

There are a few more things worth mentioning for the occupancies in the PVDIS configuration. First of all, due to the baffle, hit distributions on the GEM trackers are highly non-uniform for both DIS electrons and backgrounds, as shown in Figure 19 and Figure 20. This means that one may turn off certain high voltage sectors inside the GEM in order to reduce its occupancy. This will help significantly for the last two GEMs as the background distribution is almost separated from the hit position from DIS electrons. In the current program, we can turn off certain high voltage sectors, like what Figure 21 shows [14]. Also, notice that the inner radii of the first two GEMs seem too small as there is almost no particle of interest in those region, and they only receive background events. So in the study, those region are turned off as well. The occupancies by strips for each GEM tracker after setting off certain high voltage sectors are presented in Figure 22.

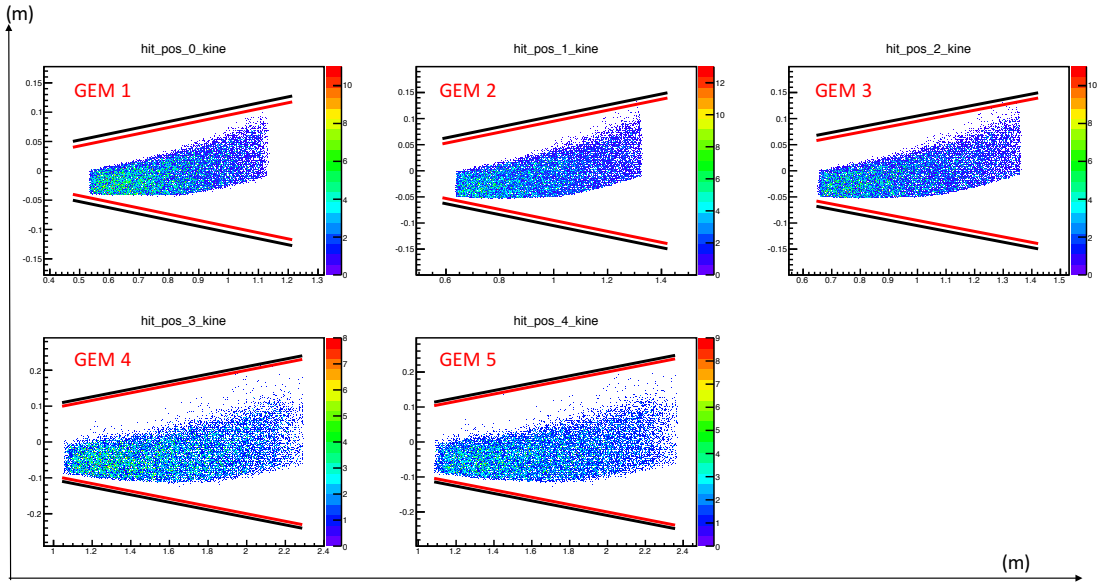


Figure 19: Distributions of electrons from the DIS process on the five GEM trackers in the PVDIS configuration. Since different sectors share azimuthal symmetry, all events are mapped to one sector. Black and red lines mark the frame of each GEM sector. In addition to the trigger condition, all events shown satisfy the kinematic cuts: Q^2 >6 GeV^2, $x_{Bjorken}$ >0.55 and W >2 GeV.

4.2 Track finding

Currently, we have studied the single electron track efficiencies and accuracies for the three configurations mentioned in the previous section. In all cases, we assume 100% beam-on-target background and always use the calorimeter to start track finding. At the moment, we assume that there is only one high energy hit on the calorimeter, which is coming from the signal track. The output of track finding can be zero-track, which means the track finder fails to find any track, single-track, which means one track is found and multi-track, which means multiple tracks are found. Since at the moment, there is only one signal track for each event, we use the single-track rate as the track finding efficiency. Some additional cuts may be applied to further suppress multi-track events. In general, tracks that have more hits found and smaller χ^2/ndf have higher chance to be the right tracks. One can see the effect of such cuts from Figure 23 and Figure 24, which show resolutions of

[14] Currently the high voltage sector must be rectangular. Dr. Richard Holmes has extended this to arbitary polygons, and has more advanced design for the their shapes.

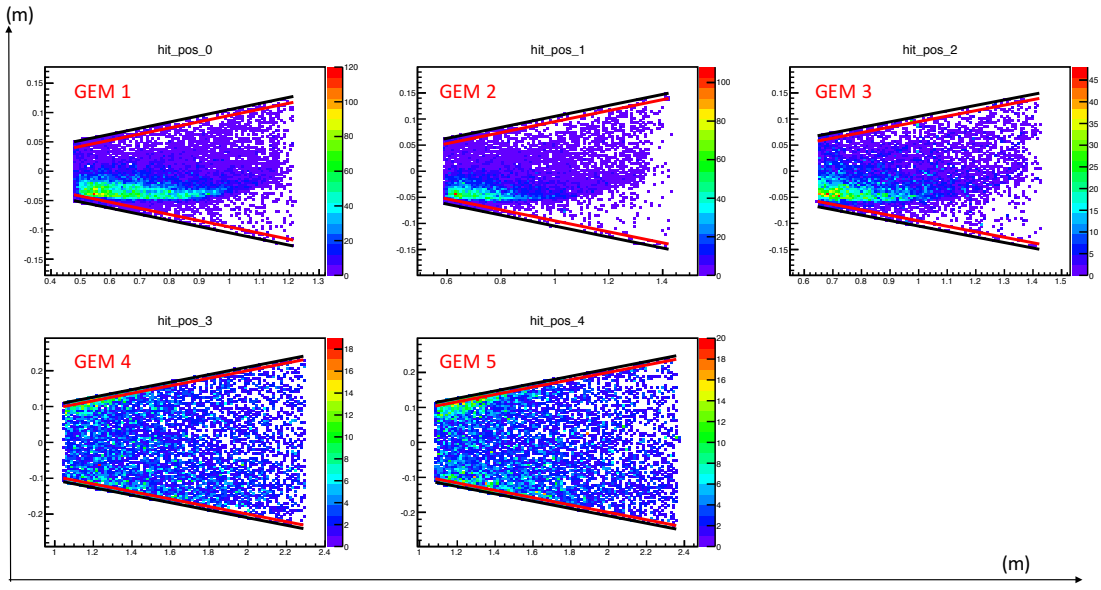


Figure 20: Distributions of beam on target events on the five GEM trackers in the PVDIS configuration. Since different sectors share azimuthal symmetry, all events are mapped to one sector. Black and red lines mark the frame of each GEM sector. Only hits with non-zero energy deposition in the GEM drift layer are shown.

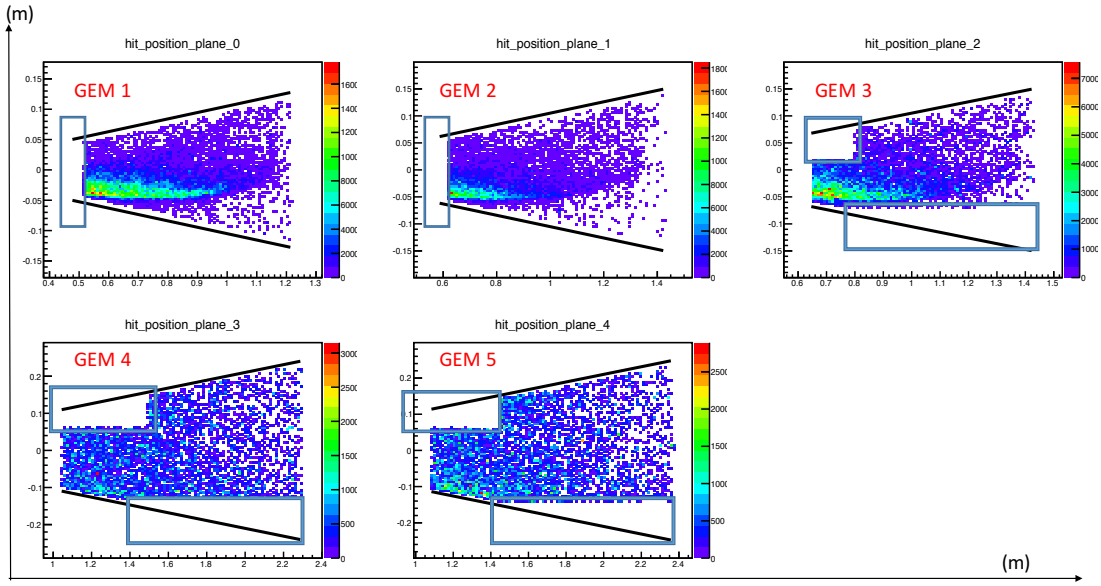
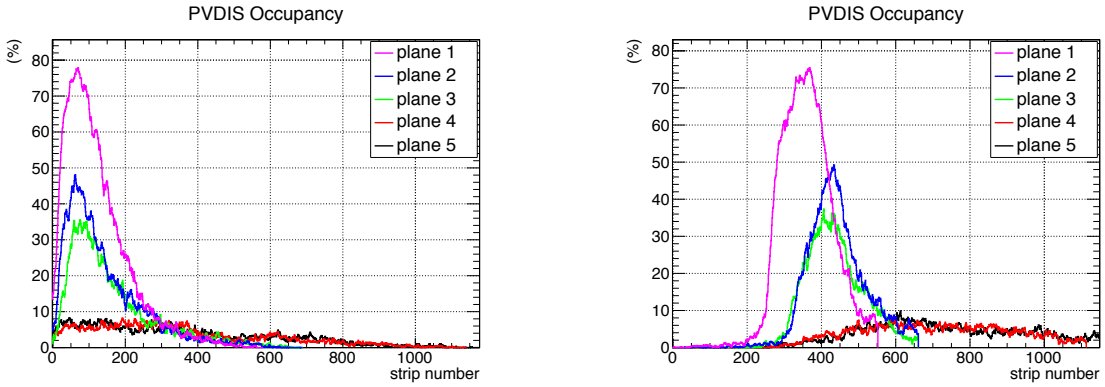


Figure 21: Distributions of beam on target events on the five GEM trackers in the PVDIS configuration. Certain high voltage sectors are turn off, as marked by the blue boxes.



(a) 1a (b) 1b

Figure 22: Occupancies by strips in the PVDIS configuration. If one look at Figure 20, u strips are parallel to the lower edge of a chamber, and v strips are parallel to the upper edge of the chamber. The Strip index starts counting from down to up. The exact distributions of occupancies by strips depends also on the azimuthal angle offset of each chamber, as the hit distributions are not azimuthal symmetric. The offset angles for this study are: 3.2°, 2.2°, 2.2°, -0.8° and -0.8° relative to the nominal azimuthal angles of chambers on each tracker.

some kinematic variables on the front GEM tracker for some PVDIS multi-track events. Without these cuts, there are clearly flat backgrounds for these variables. They are most likely coming from the wrong tracks, whose reconstructed variables are not strongly correlated with the true MC variables. After the cuts, the remaining tracks show much better agreement with the true MC values. In addition to the efficiency, one should also consider the accuracy of the track finding, as even for a single track event, the track found is not necessary the right track. In this study, the accurate track (or right track) must find all the closest reconstructed hit to the MC hit on each GEM tracker.

However, this type of cut may not be very sufficent as the χ^2/ndf for right tracks and wrong tracks may not be very well separated (χ^2 distribution may has a long tail for instance). It is quite likely we can introduce some machine learning techniques to enchance this part.

The track finding averaged efficiencies and accuracies for single electron events for SIDIS-^{3}He forword angle region, SIDIS-^{3}He large angle region, J/ψ forward angle region, J/ψ large angle region, and PVDIS are shown in Figure 25 to Figure 29. In all cases, track multiplicity 2 actually means more than one track found, not necessary means exactly two tracks found, even though the chance of finding more tracks drops very fast with larger track multiplicities. All results are weighted by cross sections of corresponding processes. In all cases, it is required that signal tracks must hit all the GEM trackers in MC and the corresponding calorimeter[15]. For SIDIS-^{3}He, signal tracks are selected by requiring electrons must have enough energy deposition in order to trigger the calorimeters. For J/ψ, signal tracks much have energy greater than 0.6 GeV when they reach the calorimeters. For PVDIS, in addition to the trigger condition on the calorimeter, it is also required that signal tracks must satisfy: Q^2 >6 GeV^2, $x_{Bjorken}$ >0.55 and W >2 GeV. In addition, more detailed track finding efficiencies and accuracies are presented from Figure 30 to Figure 34, in which the results are binned in difference kinematic variables, $y = (E - E')/E$ and $x_{Bjorken}$ for PVDIS, and momentum p and polar angle θ for SIDIS and J/ψ.

[15]For SIDIS and J/ψ, it seems like when requiring a high energy track hit the calorimeter, it is almost guaranteed that the track will hit all GEMs in the MC. For PVDIS however, there seems to be a small amount of track will miss the last two GEMs. It is possible that either the first few GEMs are two large, all the last few GEMs are too small.

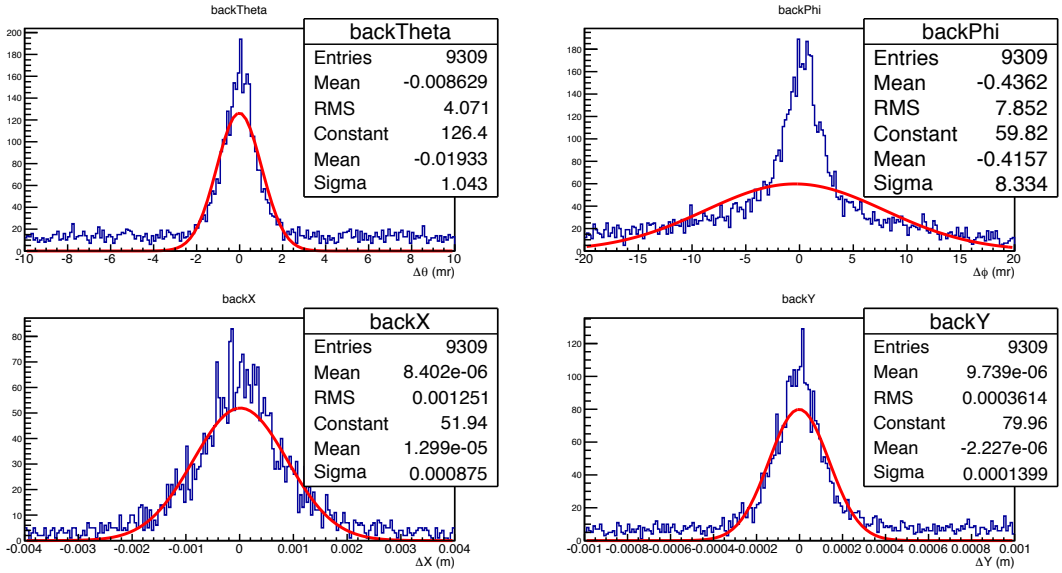


Figure 23: Plots show the difference between reconstructed variables from all the multi-track events and the MC true variable on the first GEM tracker in the PVDIS configuration. Top left plot is for the polar angle of the track as it crosses the first GEM tracker, top right plot if for the azimuthal angle. Bottom left plot is for the x coordinate and bottom right plot is for the y coordinate. Most of the wrong tracks are shown as the flat part of the distribution as their reconstructed variables have no strong correlation with the MC true values.

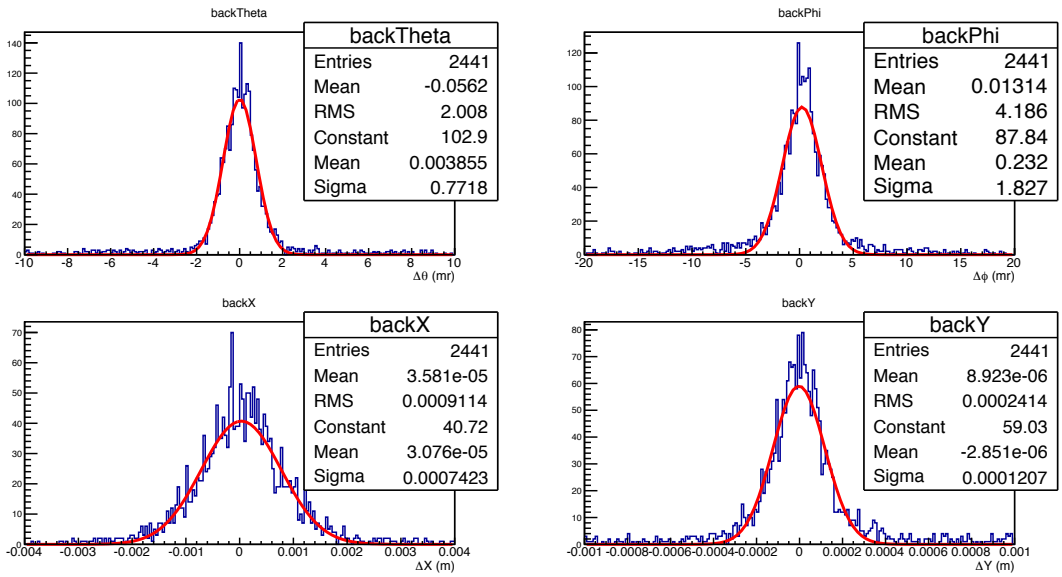
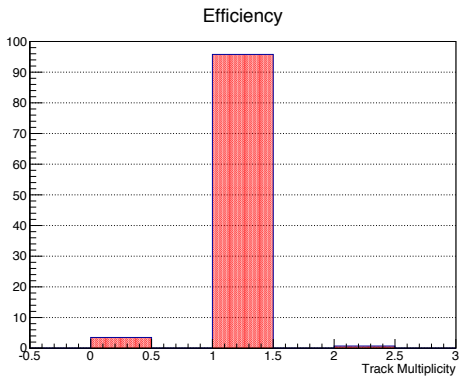
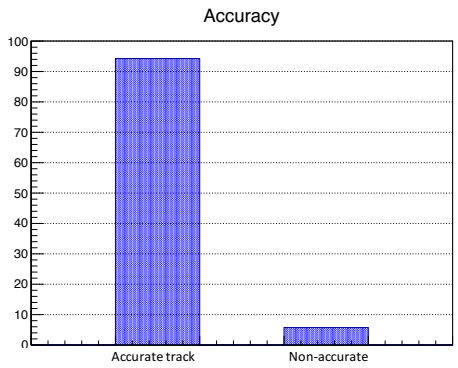


Figure 24: Same data set as shown in Figure 23, but obtained by requiring reconstructed tracks must have hit on every GEM tracker their χ^2/ndf must be greater than 8.

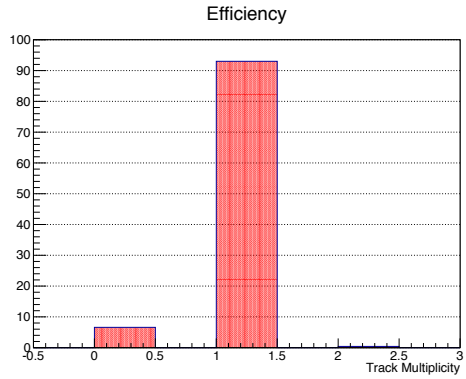


(a) 1a

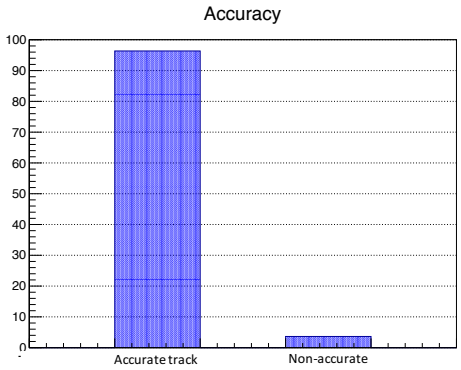


(b) 1b

Figure 25: Averaged single electron track efficiency and accuracy for tracks in SIDIS-^{3}He forward angle region.

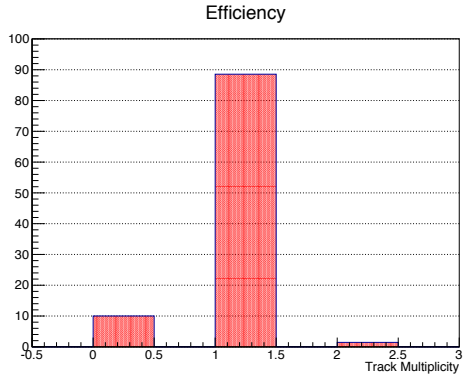


(a) 1a

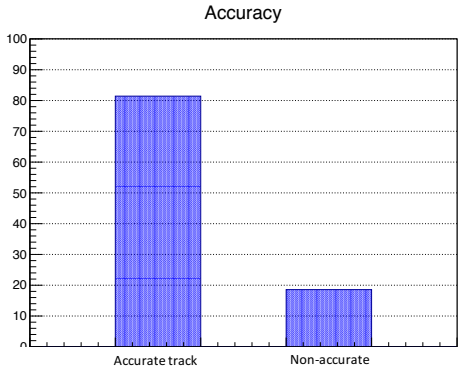


(b) 1b

Figure 26: Averaged single electron track efficiency and accuracy for tracks in SIDIS-^{3}He large angle region.

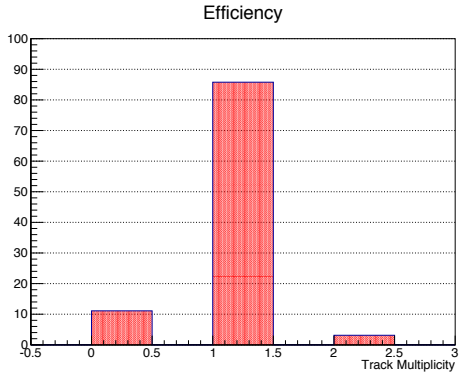


(a) 1a

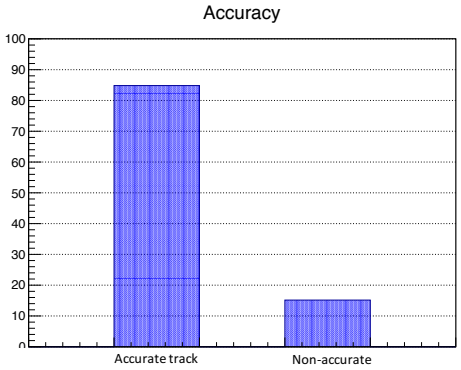


(b) 1b

Figure 27: Averaged single electron track efficiency and accuracy for tracks in J/ψ forward angle region.

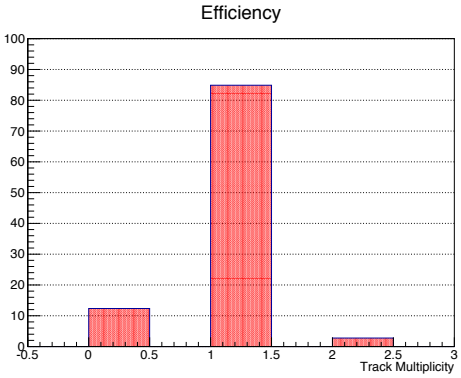


(a) 1a

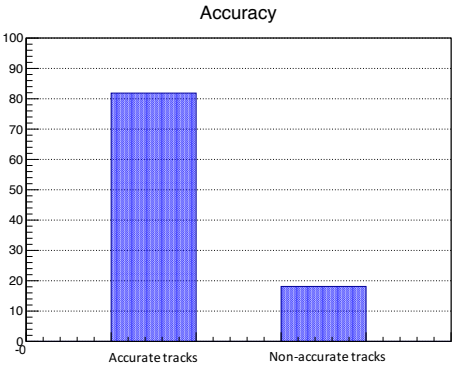


(b) 1b

Figure 28: Averaged single electron track efficiency and accuracy for tracks in J/ψ large angle region.



(a) 1a



(b) 1b

Figure 29: Averaged single electron track efficiency and accuracy for tracks in the PVDIS configuration.

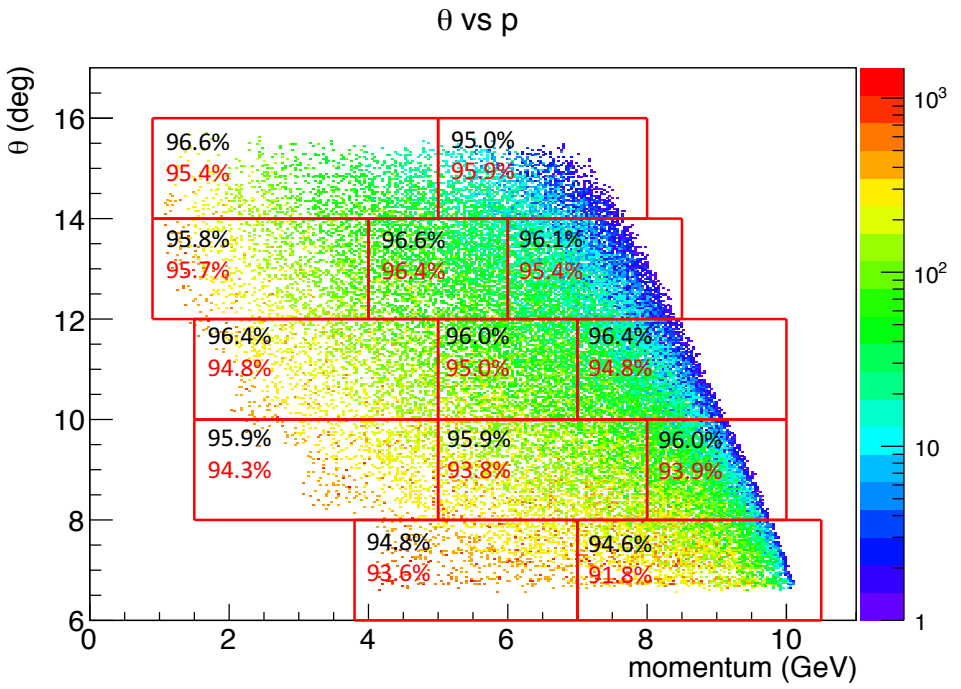


Figure 30: Track finding efficiencies (black numbers) and accuracies (red numbers) in difference polar angle θ and momentum p bins in SIDIS-^{3}He forward angle region. The efficiencies refer to single-track efficiencies.

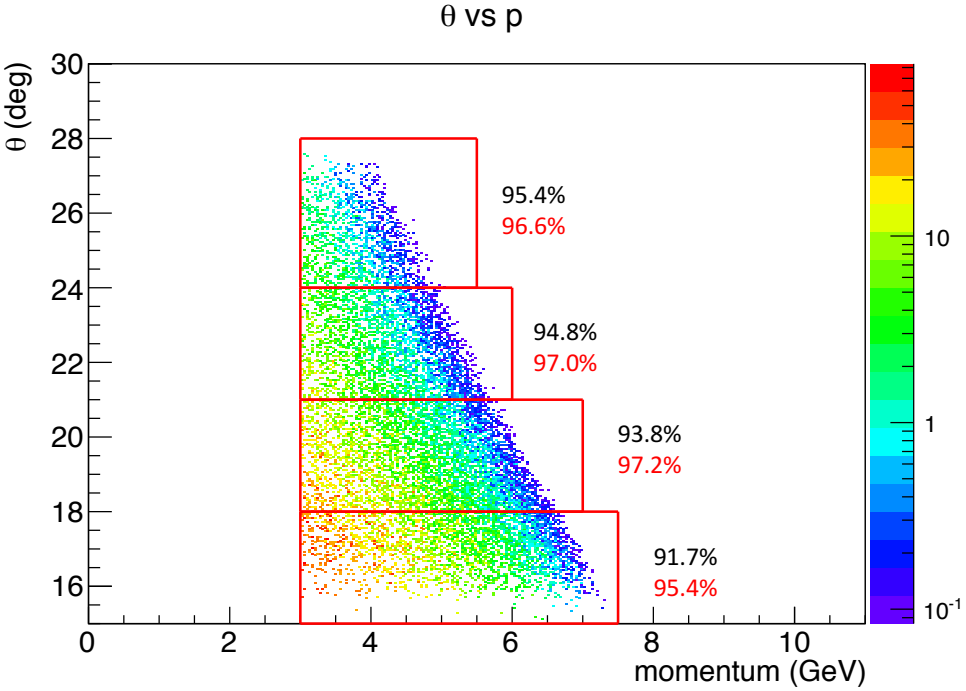


Figure 31: Track finding efficiencies (black numbers) and accuracies (red numbers) in difference polar angle θ and momentum p bins in SIDIS-^{3}He large angle region. The efficiencies refer to single-track efficiencies.

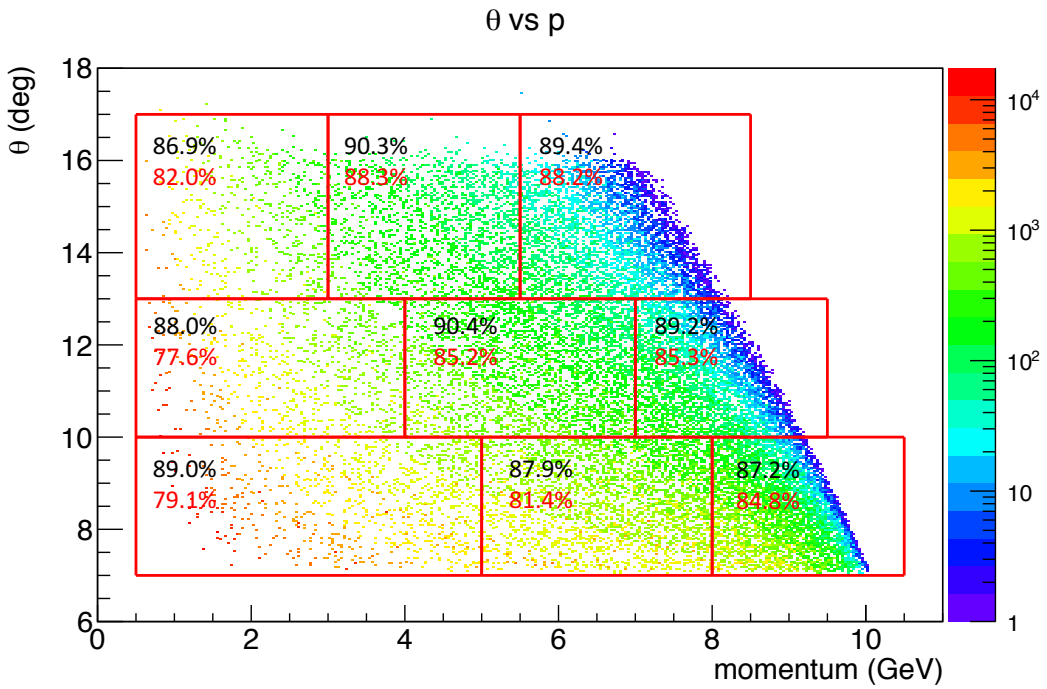


Figure 32: Track finding efficiencies (black numbers) and accuracies (red numbers) in difference polar angle θ and momentum p bins in J/ψ forward angle region. The efficiencies refer to single-track efficiencies.

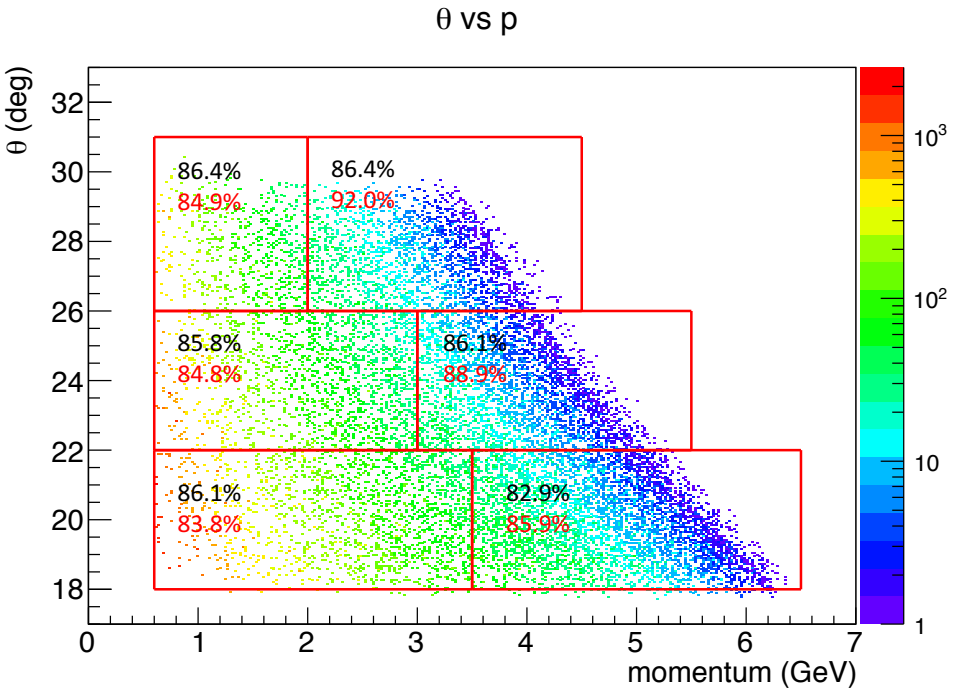


Figure 33: Track finding efficiencies (black numbers) and accuracies (red numbers) in difference polar angle θ and momentum p bins in J/ψ large angle region. The efficiencies refer to single-track efficiencies.

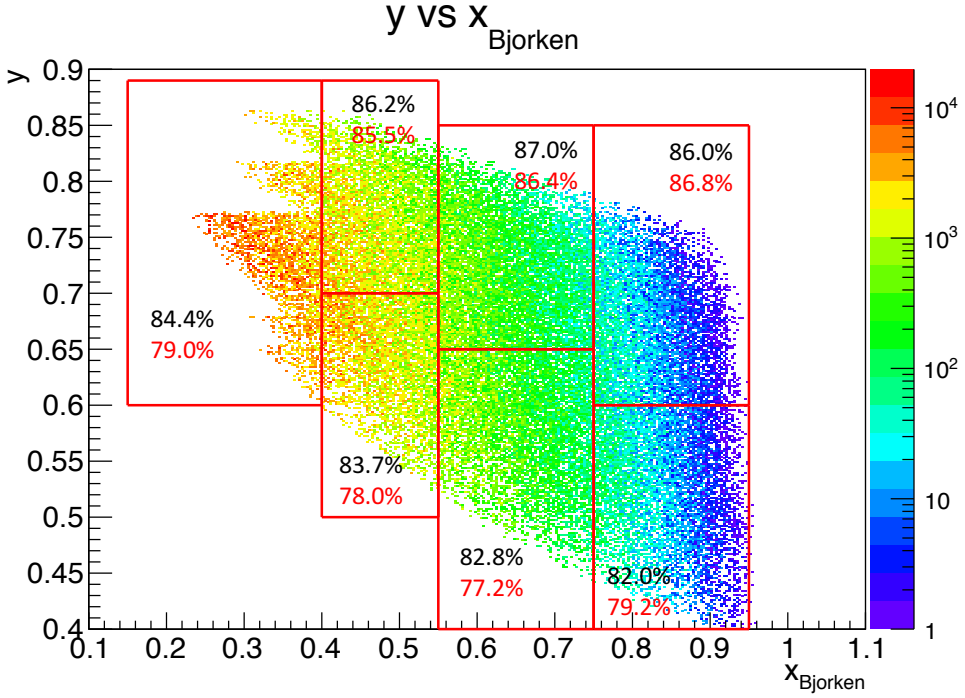


Figure 34: Track finding efficiencies (black numbers) and accuracies (red numbers) in difference $y = (E - E')/E$ and $x_{Bjorken}$ bins for the PVDIS configuration. The efficiencies refer to single-track efficiencies.

4.3 Vertex resolution

Most of the vertex resolution studies are done before the year 2016, using another program, whose essential part is almost exactly the same as the track finding and fitting part of the current tracking reconstruction program. Those studies were done with non-digitized hits on GEMs and no contamination from beam-on-target events. The position resolution of GEM hits was smeared by a Gaussian distribution with $\sigma = 90\mu$m. The results are summarized on SoLID wiki page `https://jlabsvn.jlab.org/svnroot/solid/subsystem/gem/resolution/v1/`. In this study, I calculated the resolutions of polar angle, azimuthal angle, momentum and vertex z for electrons and pions for SIDIS-^{3}He, electrons and protons for J/ψ and electrons for PVDIS. The results were binned in the polar angle and momentum bins. An example is shown in Figure 35, which is for the momentum resolution of electron tracks in SIDIS-^{3}He configuration.

Unfortunately, such detailed studies have not been carried out with the new program yet. But in principle, this should be rather easy. There is however, one missing piece in the current program, which is a re-fit function. Based on previous experience, track fittings for SIDIS and J/ψ basically converge within the first fit while for PVDIS, the momentum resolution gets a significant improvement after the first re-fit, which basically requires one to take the result from the first fit and initiate a second one [16]. Once we have this component in the new program, we will be able to check other things such as the effect of background and wrong tracks on the physics results. An alternative, and perhaps better approach is to use the method that developed by Dr. Richard

[16] This probably has something to do with the characteristics of the magnetic field. For SIDIS and J/ψ, all GEMs are rather evenly distributed inside a strong and uniform field, this gives KF enough room and information to get rid of the dependency on the initial estimate. For PVDIS, however, the field integral between the GEMs is very small compared to that between the vertex and the first GEM, so before adding the BPM position, the reconstructed momentum may still has a strong dependency on the initial estimate.

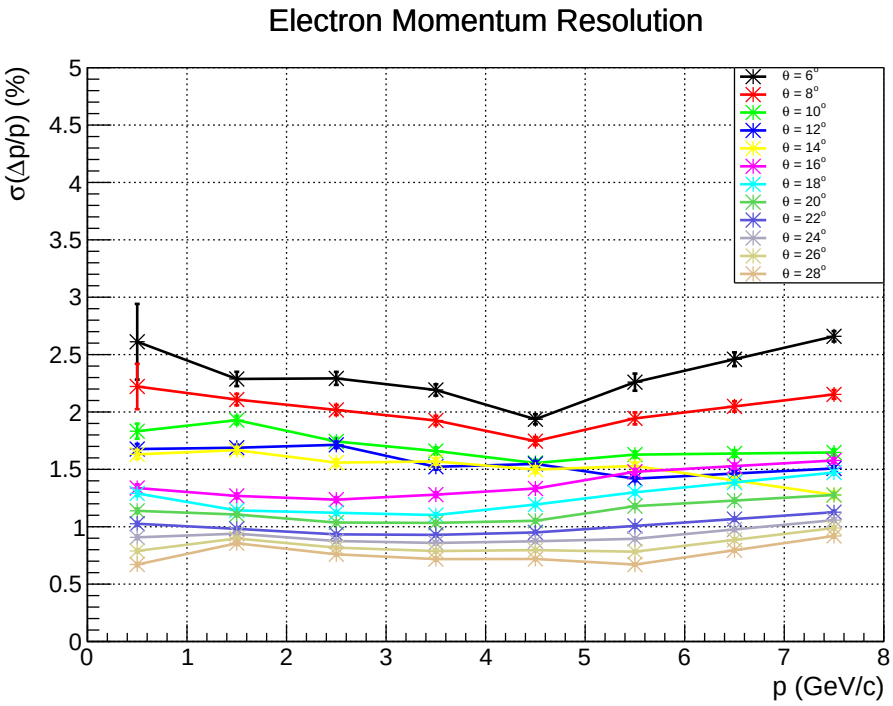


Figure 35: Momentum resolution for electrons in SIDIS-^{3}He configuration.

Holmes, using which one can calculate the vertex variables using high order polynomials that are extracted from the simulation. Based on prevoius comparison, this method is able to give fairly similar resolution, and it is certainly much faster.

At the moment, only averaged resolutions are studied using the new program. They are shown in Figure 36 to Figure 40. Again, each event is weighted by the cross section for the corresponding processes, and all results obtained under 100% beam-on-target background. Only resolutions for single track events and multi-tracks that pass the number of hits and χ^2/ndf cuts are shown in the histograms.

5 Summary

In this technical note, I summarize most of my studies related to SoLID tracking reconstruction since 2015. At the moment, the GEM digitization and the tracking reconstruction program are both available on Github. These two programs can be used to digitize the SoLID-GEMC simulation files and perform track finding and fitting tasks. The vertex resolution we obtained should satisfy requirements in original proposals for all detector configuration mentioned in this note. The tracking efficiency for SIDIS-^{3}He is above 90% at the moment, which is reasonably good. The efficiency for PVDIS and J/ψ on the other hand, is only around 85%, which is only marginal. There are certainly a lot of unfinished tasks related to both the GEM digitization and tracking reconstruction. In the following sections, I will discuss some unfinished tasks I have in mind and may also give some thoughts about how to proceed with them.

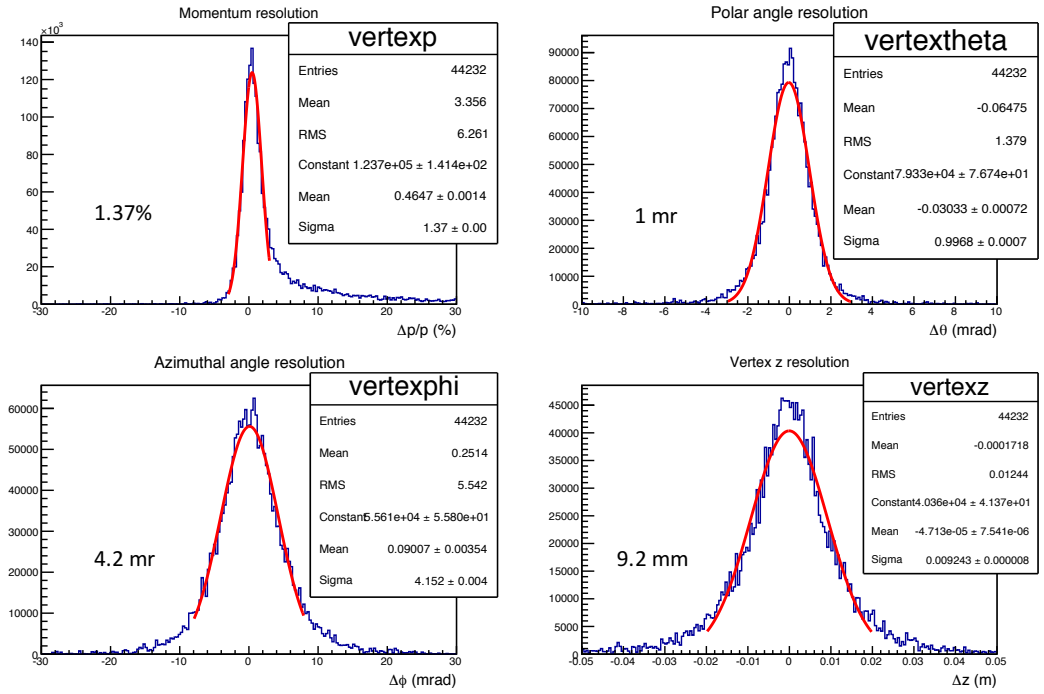


Figure 36: Resolutions for the vertex variables in SIDIS-^{3}He forward angle region.

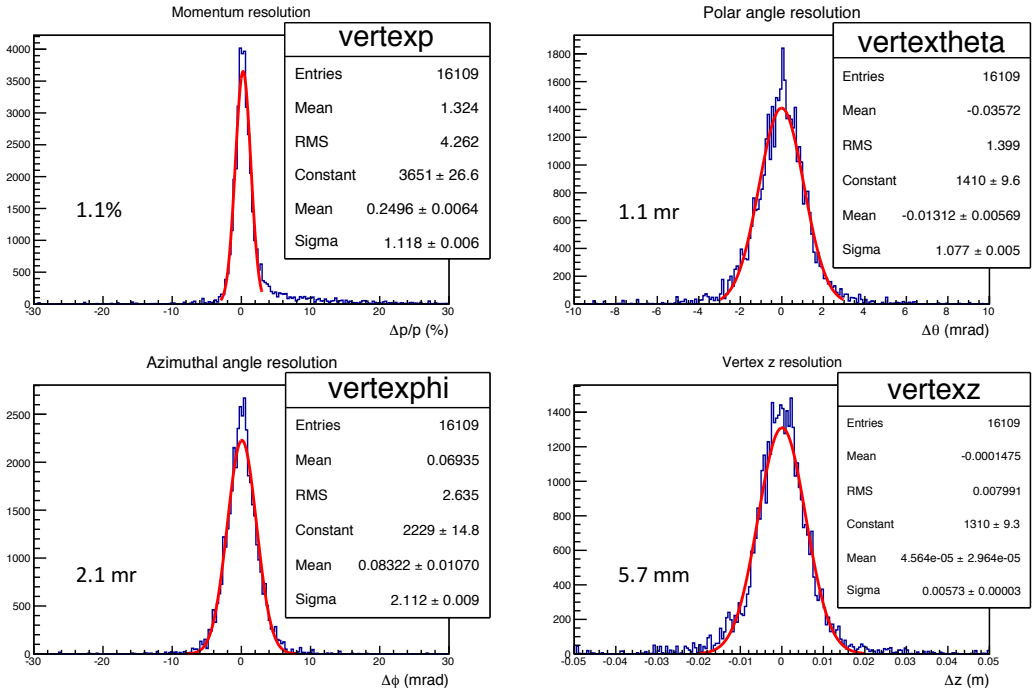


Figure 37: Resolutions for the vertex variables in SIDIS-^{3}He large angle region.

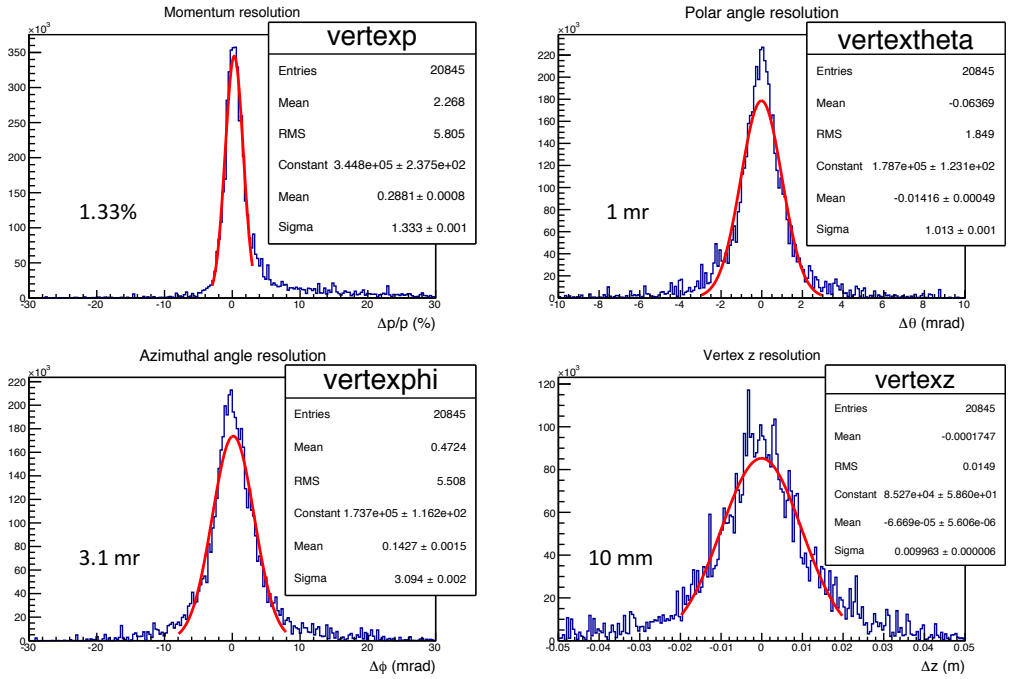


Figure 38: Resolutions for the vertex variables in J/ψ forward angle region.

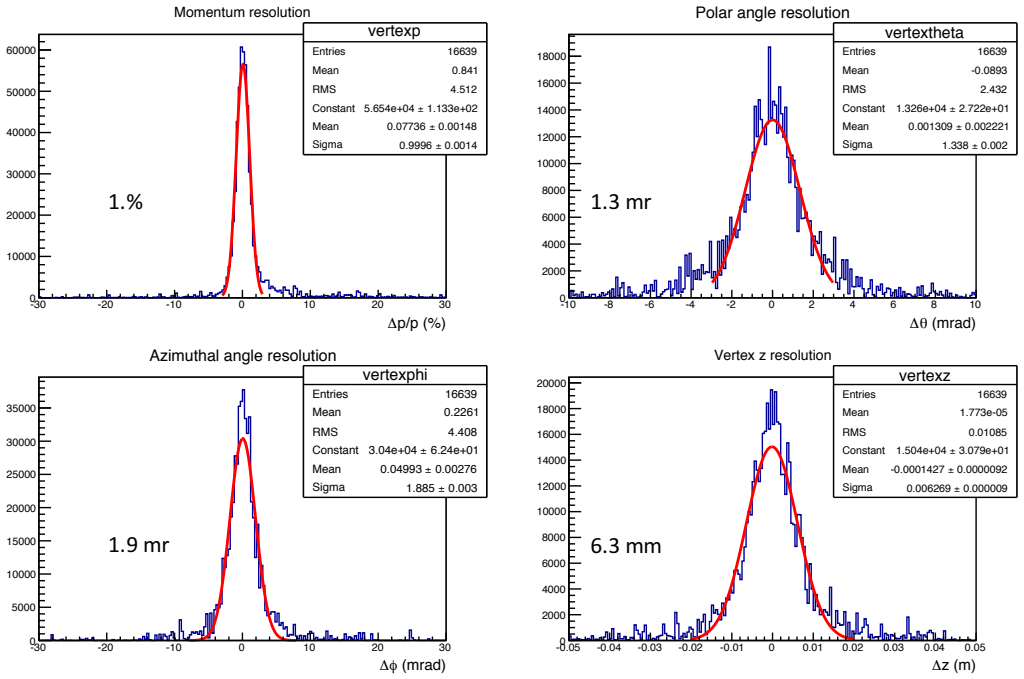


Figure 39: Resolutions for the vertex variables in J/ψ large angle region.

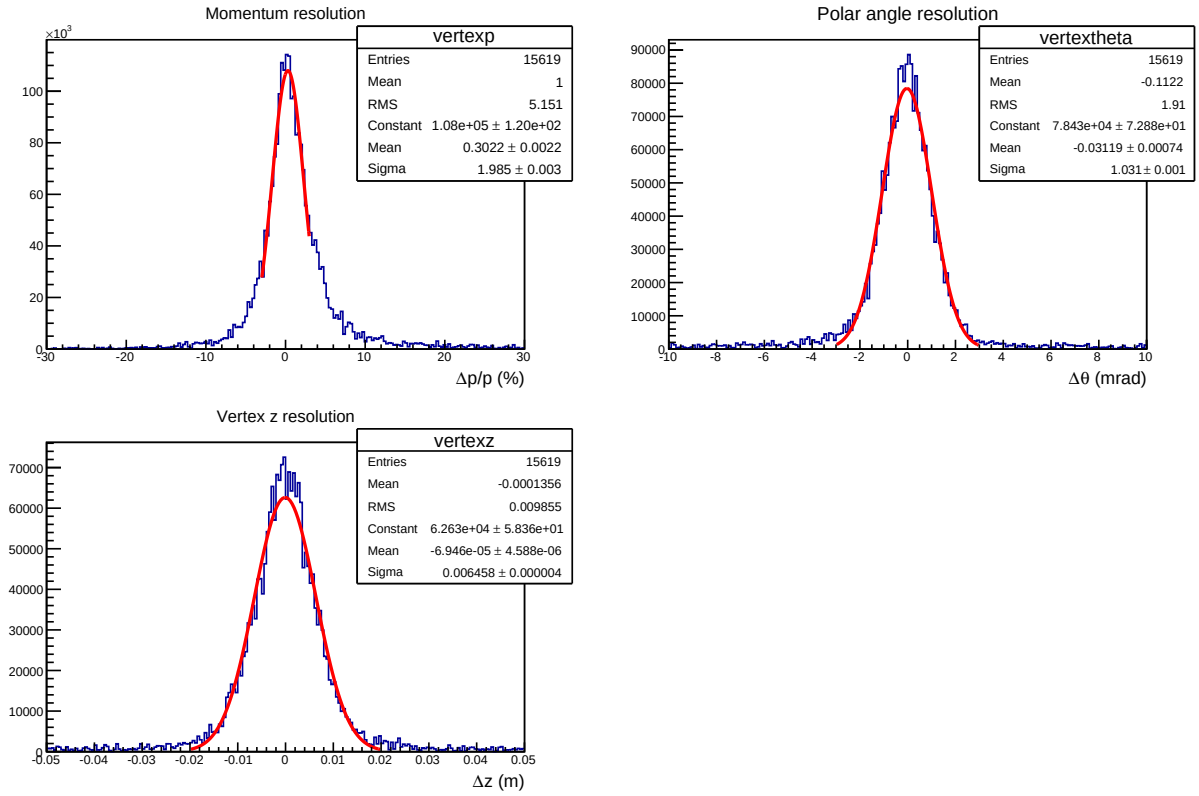


Figure 40: Resolutions for the vertex variables for the PVDIS configuration.

5.1 Unfinished tasks and some thoughts for the future development

5.1.1 Future development for libsolgem

At this stage, there is no class corresponds to APV25 in the program. All strips are assumed to be identical, which means they have the same pedestal noise level, and same ADC offset and so on. In reality, strips are connected to different APV25s, which have different pedestal noise level. And different time samples from a APV25 have different ADC offsets. In fact, one will need to do the common mode correction to correct the baseline of the time samples for each of the APV25s. And this correction is done by utilizing the ADC signals on the channels that are not fired. Clearly, this is certainly not tested with the current digitization and it may be urgent to study this effect as the SoLID GEMs need to perform in the high rate environment, which will make the common mode correction harder as there will be less unfired strips. Adding an APV25 structure to the program will also make the simulation of cross talk signals more realistic. Right now the cross talk signals are randomly placed certain strips apart from and either on left of right of the main signal. In reality, cross talk signals only appear for channels that are connected to the same APV25. The correct way to do the cross talk simulation is by grouping all the strips into APV25 and use an realistic mapping between channels inside the APV and strips on the readout board.

Secondly, even though we have spent quite some works on tuning the program so that the simulation matches more with the data, this part of the work is not completely finished. For instance, the resolution from the digitization still seems to be too good compared to the data (typically around 70 μm for 400 μm readout GEMs, while we are having something around 60 μm). And also, many important properties of a GEM detector depend on the direction of the incident particle. For instance, when the particle is shooting vertically into the GEM detector, we tend to have a more concentrated cluster that is smaller in size but may have larger maximum for its ADCs. The resolution and detection efficiency for these kind of particles might be better than the other case when the particles have larger incident angle. We certainly need to do more studies to compare the GEM digitization and the data, and continue improve the program.

Thirdly, the computation time needed for the libsolgem is very long, and it certainly needs to be improved. This problem may be less obvious for PVDIS because of the option of map sector, which may accelerate the program by a factor of 30. For SIDIS and J/ψ however, it may take a few seconds to digitize one event, which is quite unacceptable. But in any case, consider in the future we may need to digitize some number of events that have similar statistics compared to the experimental data, we need to modify this program in some fundamental ways.

The most time consuming part is to digitize background events, during which we actually re-use the background events by randomize their timing relative to the signal time. This means that a lot of computation power is wasted by digitizing the same background events. A possible solution (other than ask for more CPUs) is that we digitize all the signals and backgrounds only once, and obtain files that contain their ADC data separately. And then we mix the ADCs by randomize the timing and amplitudes and so on. There may be quite some works needed for this modification, and also validate the results. But this may improve the execution speed of the program by a few orders of magnitude.

5.1.2 Future development for the tracking reconstruction

At the moment, the reconstruction program is less as advanced as libsolgem, which was already a very well developed program when I received it. There are a lot of works we can do to make it cleaner and more efficient. But in this section, I would like to focus on something more important.

The most urgent task in my opinion is to finalize the GEM design for the SIDIS and J/ψ configuration. It is almost certain that there will be some overlap regions for the GEMs in these

two configurations, as shown in page 80 of the SoLID updated preliminary conceptual design report [6]. However, currently for these two configurations, the GEMs have 30 non-overlaping sectors and with no dead area in between. The tracking reconstruction also have no particular treatment to deal with overlap regions. So this part certainly needs to be updated and as soon as possible. More importantly, because of this simplification of the GEM design, there is also no GEM frame in the simulation. The GEM frame is typically made of G10 fiberglass, which has a relative radiation length of around 10% each. It is quite likely that they will be the dominating source of external radiative effects and multiple scatterings for SIDIS and J/ψ [17]. And because these frames may change the tracks significantly when the particles are propagating in between the GEMs, they may make the reconstruction much more difficult [18]. Thus, the current results we get may be too optimistic. It has been a long argument what the final design of SIDIS and J/ψ GEMs should be. If this cannot be decided soon. I would suggest we just put some arbitary overlapping GEMs in the simulation to start with. They don't have to have 12° azimuthal angle coverage, or 20 chambers on each GEM tracker. As long as, there are frames and overlapping regions, we can start modify and test the reconstruction program, and get more realistic estimations for the tracking efficiency and accuracy. One should also try to introduce less hard-coding for the geometry part, so that when the final design is fixed, we will only need to change a few numbers in the database files.

Another equally urgent task for the tracking reconstruction is the GEM clustering part. First of all, the current GEM clustering algorithm will not work at the transition region between segmented strips and unsegmented ones. This part of the GEM digitization is finished by Dr. Richard Holmes, but we cannot test how it can improve the tracking reconstruction without having the GEM clustering part updated accordingly (one will also need to change the crate-slot-channel mapping in libsolgem and the reconstruction program). Other than this, there are a number of important but unfinished tasks such as simulating and testing the common mode correction, the zero-suppression, improving the out-of-time noise rejection algorithm and so on.

The third thing is the hadron tracking for the SIDIS case. Previously I did some studies about this and also coincidence tracking (finding an electron track and hadron track at the same time). However, this part of the study was terminated when noticing that the GEM efficiency for hadrons is significantly lower than that for the electrons, which is somewhat unexpected. I believe the major reason is that from the GEMC simulation, hadrons deposit much less energy compared to electrons in the drift layer of the GEM detector, as shown in Figure 41. As a result, the GEM signals from pions have much smaller amplitudes than that from electrons, so that they have higher chances to be suppressed by GEM threshold cuts [19]. We will need to understand if this reflects the reality.

For the long term development, we should certainly introduce some machine learning algorithms into the tracking reconstruction. This has been achieved by many large scale high energy physics detectors like CMS and ATLAS. We may or may not be able to completely rely on the machine learning for the reconstruction, but certainly a combination of KF and machine learning is a achievable thing. An easy spot to introduce the machine learning algorithm is the final selection part of the tracking reconstruction. One can use some neural network algorithms to help identify the correct track in the multi-track events. The other place one may consider is the seeding part, which currently relies on a lot of parameters from the simulation and some simple geometric models. It should be replaced by some machine learning algorithm as well. In addition to the track finding, there are also more advanced algorithm for the track fitting. One of them is the Gaussian sum filter, which is an enhancement of Kalman Filter. It is known that KF cannot deal with non-Gaussian

[17] For PVDIS, they are typically behind the baffel so the situation is more or less fine

[18] If the track gets a significant direction change before the first GEM, it is still more or less fine becasue from all the GEMs we can find a smooth track.

[19] GEM efficiency for electrons is tuned to be around 95%, with the same gain, the efficiency for pions is slightly less than 90%.

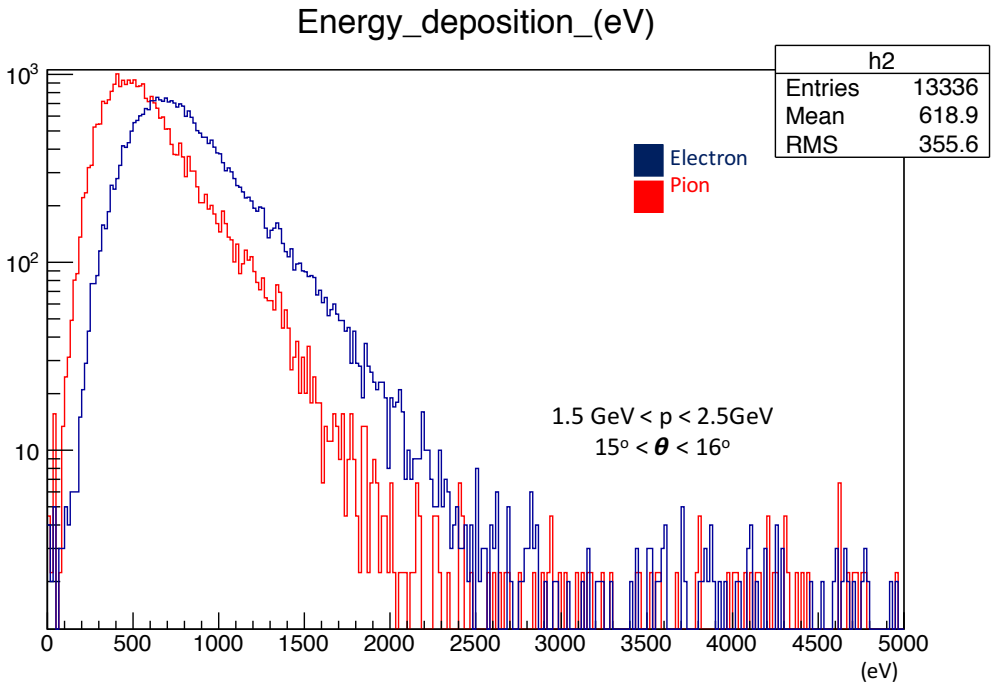


Figure 41: Energy deposition of electrons and pions in the GEM drift layer. In both case, incident particles have momentum from 1.5 to 2.5 GeV, and incident angles from 15° to 16°.

noise properly but in tracking reconstruction, there are many non-Gaussian noises. An important one is the radiative energy loss for electrons. The Gaussian sum filter is particularly designed to deal with this problem. So this is something we can consider as well.

5.1.3 Suggested piority list

Based on my experience and understanding, here I summarize the unfinished tasks mentioned above and rank them in decreasing priority order in the following priority list. Some of the points are not mentioned in this section, but mentioned elsewhere in this note. These items focus on the development of libsolgem and the tracking reconstruction themselves. Things like using the reconstruction program to study how wrong tracks may affect the final physics result will not be included in this list.

(1)Finalize SIDIS and J/ψ GEM configuration, putting GEM frames and overlap regions in the simulation, modify the digitization and tracking reconstruction accordingly.

(2)Modify the GEM clustering algorithm for segmented strips. Test the effect for the PVDIS configuration.

(3)Add APV objects in the digitization, add common mode noise and try to do common mode correction in the GEM clustering.

(4)Understand the low efficiency for hadrons. Continue hardon tracking and coincidence tracking.

(5)Add refit function to the tracking reconstruction program or use polynominals that extracted from simulation.

(6)Try various machine learning algorithms, see how we can use them to improve the reconstruction.

(7)Optimize the out-of-time noise rejection algorithm (and simulation), may use some machine learning algorithm here as well.

(8)Fine-tunning the digitization, including finalizing the GEM shaping function and obtaining better match with experimental data.

(9)Speed up the digitization.

(10)Try other reconstruction and fitting algorithms.

6 Appendix

6.1 A-1 Database for libsolgem

Parameters in *db_gemc.dat* are introduced in the following table. Noticed that due to recent development by Dr. Richard Holmes, some of the parameters summarized here are different from his new version. Only parameters for the first GEM sector in the PVDIS configuration is introduced here as all the other sectors are simply just very similar copies.

parameter	explanation
gemc.gem1.r0 (m)	inner radius of a GEM sector
gemc.gem1.r1 (m)	outer radius of a GEM sector
gemc.gem1.phi0 (deg)	starting azimuthal angle of a GEM sector
gemc.gem1.dphi (deg)	azimuthal angle coverage of a GEM sector
gemc.gem1.z0 (m)	z position of the GEM sector, defined in the reference frame of GEMC
gemc.gem1.depth (m)	thickness of a GEM sector (no really used in the program)
gemc.gem1.frame_width (m)	width of the GEM frame, used in estimation of dead area
gemc.gem1.n_HV_sector_off	number of high voltage sectors that is going to be turned off
gemc.gem1.gem1x.stripangle (deg)	strip direction in the sector frame (x-axis parallel to the symmetric axis of the GEM sector) for the first set of strips
gemc.gem1.gem1x.pitch (m)	pitch between adjacent strips for the first set of strips
gemc.gem1.gem1y.stripangle (deg)	strip direction in the sector frame (x-axis parallel to the symmetric axis of the GEM sector) for the second set of strips
gemc.gem1.gem1y.pitch (m)	pitch between adjacent strips for the second set of strips
gemc.gem1.HV1.bound (m)	bounding box for the first HV sector that is going to be turned off parameters correspond to a rectangle defined in the sector frame
gemc.gem1.HV2.bound (m)	bounding box for the second HV sector that is going to be turned off parameters correspond to a rectangle defined in the sector frame
gemc.gem1.HV3.bound (m)	bounding box for the third HV sector that is going to be turned off parameters correspond to a rectangle defined in the sector frame

Parameters in *db_ratedig.dat* are introduced in the following table. These parameters are responsible for the calculations in the GEM digitization steps.

parameter	explanation
ratedig.gasionwidth (eV)	energy needed to produce one ion pair on average
ratedig.gasdiffusion (mm^2/s)	electron diffusion speed in the lateral direction
ratedig.gasdriftvelocity (mm/s)	electron drift speed in the vertical direction
ratedig.avalanchefiducialband	size of boundary area used for the integration on total charge on the readout
ratedig.avalanchechargestatistics	statistical model for the number of avalanche electron produced
ratedig.ava_model	probability density function for the avalanche electron in space, 0 : Heaviside, 1 : Gaussian, 2: Cauchy-Lorentz
ratedig.ava_gain	used in conversion of charge to ADC
ratedig.gainmean	used in estimation for the total charge each ion pair can produce
ratedig.gain0	used in estimation for the total charge each ion pair can produce
ratedig.avalateraluncertainty	not really used
ratedig.avalanche_range	not really used
ratedig.max_ion	maximum number of ion pairs allowed
ratedig.triggeroffset (ns)	trigger offset for each GEM tracker (array with length equals to total number of trackers)
ratedig.triggerjitter (ns)	width of the trigger jitter from general electronics, assume Gaussian smearing (except for APV internal time jitter)
ratedig.apv_time_jitter (ns)	APV internal time jitter, assuming flat distribution
ratedig.elesamplingpoints	total number of APV samples from each strip
ratedig.elesamplingperiod (ns)	time between adjacent APV samples
ratedig.adcoffset	ADC offset
ratedig.adcgain	used in conversion of charge to ADC
ratedig.adcbits	maximum allowed bits for an APV sample
ratedig.gatewidth (ns)	time window used for background event simulation, background event time randomized in this time window before trigger starts
ratedig.pulsenoisesigma	parameter for pedestal simulation in readout
ratedig.pulsenoiseperiod	parameter for pedestal simulation in readout
ratedig.pulsenoiseampconst	parameter for pedestal simulation in readout
ratedig.pulsenoiseampsigma	parameter for pedestal simulation in readout
ratedig.pulseshapetau0	parameter used for the GEM signal shaping function
ratedig.pulseshapetau1	parameter used for the GEM signal shaping function
ratedig.zrout	thickness of the GEM detector
ratedig.use_tracker_frame	1: recorded MC information will be in the tracker frame, 0: recorded MC information in the same frame as GEMC
ratedig.entrance_ref	relative z distance between the top of the drift layer and the center of the GEM detector
ratedig.y_integral_step_per_pitch	total number of integration step in y direction (along the strip)
ratedig.x_integral_step_per_pitch	total number of integration step in x direction (vertical to the strip)
ratedig.do_crosstalk	1 will produce a induced cross talk signal on either left or right side of the main signal
ratedig.crosstalk_mean	amplitude of the induced cross talk signal relative to the main signal
ratedig.crosstalk_sigma	uncertainty for the amplitude of the induced cross talk signal
ratedig.crosstalk_strip_apart (strip)	distance between the induced cross talk signal and the main signal

Lastly, parameters in the *db_generalinfo.dat* are summarized in the following table.

parameter	explanation
generalinfo.do_map_sector	1 will map all the hits into a particular sector, should be used only for the PVDIS configuration
generalinfo.sector_mapped	which sector the hits are going to be mapped to
generalinfo.ntracker	total number of GEM trackers
generalinfo.nsector	total number of sectors for each tracker
generalinfo.nreadout	total number of readout sets for each tracker
generalinfo.gem_drift_id	ID of the GEM drift layer, defined as in the GEMC
generalinfo.gem_copper_front_id	ID of the copper layer above the drift layer, defined as in the GEMC
generalinfo.gem_copper_back_id	ID of the copper layer below the drift layer, defined as in the GEMC
generalinfo.gem_strip_id	ID of the readout board, defined as in the GEMC
generalinfo.faec_id	ID of the forward angle EC virtual plane, defined as in the GEMC
generalinfo.laec_id	ID of the large angle EC virtual plane, defined as in the GEMC
generalinfo.chan_per_slot	used in calculating crate slot channel map
generalinfo.modules_per_readout	used in calculating crate slot channel map
generalinfo.self_defined_sector	1 will use the azimuthal angle defined in the db_gemc.dat to re-define sector ID, otherwise use the ID defined in the GEMC simulation
generalinfo.threshold_total_adc	ADC threshold for each strip, strip that has maximum signal below the threshold will not be recorded in the output
generalinfo.nsignal	number of signal track in the simulation
generalinfo.signal1.pid	PID of the first signal track, as given in the GEMC
generalinfo.signal1.tid	TID of the first signal track, as given in the GEMC
generalinfo.signal2.pid	PID of the second signal track (if there is any), as given in the GEMC
generalinfo.signal2.tid	TID of the second signal track (if there is any), as given in the GEMC

6.2 A-2 Basic Class introduction for libsolgem

In this section some simple introductions are given for each class defined in the libsolgem library. They are based on my personal understanding on the program, if there is any incorrect statement, please contact me, and I apologize in advance.

TSolROOTFile: The purpose of this class is to deal with the input data file, which needs to be in ROOT format. It copies all the necessary information needed in the GEM digitization and MC bookkeeping into a data carrier object (TSolGEMData), which then will carry that information to the digitizer (TSolSimGEMDigitization). It requires the input file has the following four branches, the "generated" branch, the "solid_gem" branch, the "flux" branch and the "header" branch. And the branches need to follow the definition currently used in the SoLID GEMC simulation.

TSolEVIOFile: This class should in principle serve the same purpose as the TSolROOTFile class. However, many functions in this class are not updated. So it is not available at this moment. Some simple modification is needed if there is ever a need to use the evio input file format again.

TSolGEMData: an object responsible for saving the input information from the GEMC simulation file and carry it to the TSolSimGEMDigitization class to perform all the digitization calculation.

TSolDBManager: a singleton class that stored some general parameters used in the libsolgem. It needs to be initialized at the very beginning of the replay script.

TSolSpec: Not really performing any real work, it is more like a holder for all the GEM sectors. A requirement from the c++ Analyzer.

TSolGEMChamber: This class represent a GEM sector/chamber. It must be added to the TSolSpec class through the TSolSpec::AddGEM function. It includes two TSolGEMPlane objects, which represents to the two sets of readout boards, and a TSolWedge object to store some of its geometric information. In addition, dead areas on GEM are defined in this class. The function TSolGEMChamber::IsInDeadArea can be used to judge whether a hit is landed in one of its dead area.

TSolGEMPlane: This class represent a readout board of a GEM sector/chamber. It contains some transformation functions from the lab frame to the strip frame and also some strip index lookup functions.

TSolWedge: Most of the geometric information related to the GEM sector/chamber is stored inside this class, such as the radii, the starting azimuthal angle and the azimuthal angle coverage.

TSolSimEvent: A data object that is going to be filled and saved into the output ROOT file, which means if one wants to use some root script to analyze the output file, one needs to load the definition from this class. It contains the MC information that has been saved from the input files and most importantly, the ADC samples from the readout strips.

TSolSimAux: This simple class contains mainly a few helper functions. Tow of them are used rather frequently, TSolSimAux::ADCConvert converts the charges in to ADC values and TSolSi-

mAux::PulseShape is the GEM shaping follow from where we sample the data point.

TSolGEMVStrip Working variable used mostly by the TSolSimGEMDigitization::AvaModel function. Each time this function will digitize all the ion pairs from the incident particle and the resulted ADC samples from all the strips are saved temporally into a TSolGEMVStrip object. In the end, data in all the TSolGEMVStrip objects from all incident particles of an events are summed in a superimposed way.

TSolSimGEMDigitization This is the main class of the program and it is responsible for digitizing the input data (carried by a TSolGEMData object), convert them into ADC samples for all strips, and save the data input the output TSolSimEvent class.

TSolSimDecoder This class is not really needed for the digitization purpose. It is used by the tracking reconstruction program for dealing with the input data file (output of the digitization program). It loads the event buffer from the data file, and then populate them into some imaginary crate-slot-channel data structure. In the reconstruction program, there needs to be also a similar structure with the correct mapping in order to get the right data from the right channel.

TSolSimFile is an interface class for the output of the libsolgem. It can be used to open the output data file and load entries to a TSolSimEvent object.

6.3 A-3 Validation for the GEM shaping functions

In early 2017, Prof. Nilanga Liyanage's graduate student Xinzhan Bai helped us to get some GEM cosmic data taken from JLab Hall-C. For each event, 15 time samples from APV25 were recorded, which served perfectly for the study of parameterization of the GEM shaping functions. In this section, some preliminary studies on this subject will be presented and discussed. This study has not been completely finalized yet. Please contact me (weizhi.xiong@duke.edu) if you wish to have a copy of the data and continue this study.

For the GEM response function, previous we use the parameterization function and parameters from COMPASS [2] (from now on referred to as the COMPASS form). The functional form is given by the following equation:

$$P = A(1 - e^{-\frac{t-t_0}{\tau_1}})e^{-\frac{t-t_0}{\tau_2}} \tag{7}$$

where t is the variable, A is a normalization constant, t_0 is the signal start time, τ_1 and τ_2 are two shaping times for the signal, for which previously, we used 38ns and 129ns respectively in libsolgem.

On the other hand, there is a more standard from for the response function (from now on referred to as the standard form), which is given as:

$$P = A(\frac{t-t_0}{\tau})e^{-\frac{t-t_0}{\tau}} \tag{8}$$

where there is only one parameter for the shaping time τ.

In this study, I tried to use these two forms to fit the data and see which one can give a better fit. In addition, I also tried using the COMPASS form with parameters τ_1 and τ_2 fixed as 38 ns and 129 ns respectively, and the standard form with $\tau = 56$ns see if they describe the data well enough.

Figure. 42 shows the fitting results for one event, using these four type of fitting functions. It is easy to notice that the COMPASS form with fixed τ_1 and τ_2 doesn't describe the data very well as the tail tends to be larger than the data. In fact, the χ^2 of the fittings in this case tend to be much larger than the expected χ^2 distribution with $ndf = 13$ (15 data points, 2 free parameters), as shown in Figure. 43. The other thing one may notice is that if τ_1 and τ_2 are released as free parameters, the fitting function describes the data points much better. However, the parameter τ_1 tends to go to infinity. If this happens, one can take a Taylor expansion for the first exponential term in the COMPASS form and it reduces to exactly the standard form. In fact, based on the parameters from top-right and top-left plots of Figure. 42, the two fitting functions are almost the same numerically. This situation happens rather often for fittings using the COMPASS form. Thus, for numerical stability for the fitted parameters, I decided to use the standard form, as it gives much stabler values for the shaping time τ while having similar χ^2 distribution as the COMPASS form without fixing the parameters for the shaping times. The results of fittings using the standard form are shown in Figure. 44. The estimated value for the shaping time τ is around 56 ns. The χ^2 distribution does not agree perfectly with the expect curve, which may have something to do with the pedestal noise. There is some indications that the GEM pedestal noise is not exactly Gaussian and may have some sinusoidal dependency. This is one of the reasons that this subject may need further studies. Another more important thing is that there seems to be a dependency in between the shaping time τ and the amplitude of the signal, which can be seen in Figure 45. In this case, maximum ADC is the largest ADC value among the 15 time samples for each event. The shaping time starts from around 56ns for signals with smaller amplitude but grows up to around 65 ns for larger signals. There is a strange shape when maximum ADC is above 1200, which may due to saturations of APV25. In the current digitization, we use the standard form with shaping time as 56 ns, which may underestimate the tail of the signal due to this dependency but certainly

has much better description for the data compared to the original COMPASS form. However, the dependency certainly needs to be addressed better in the future.

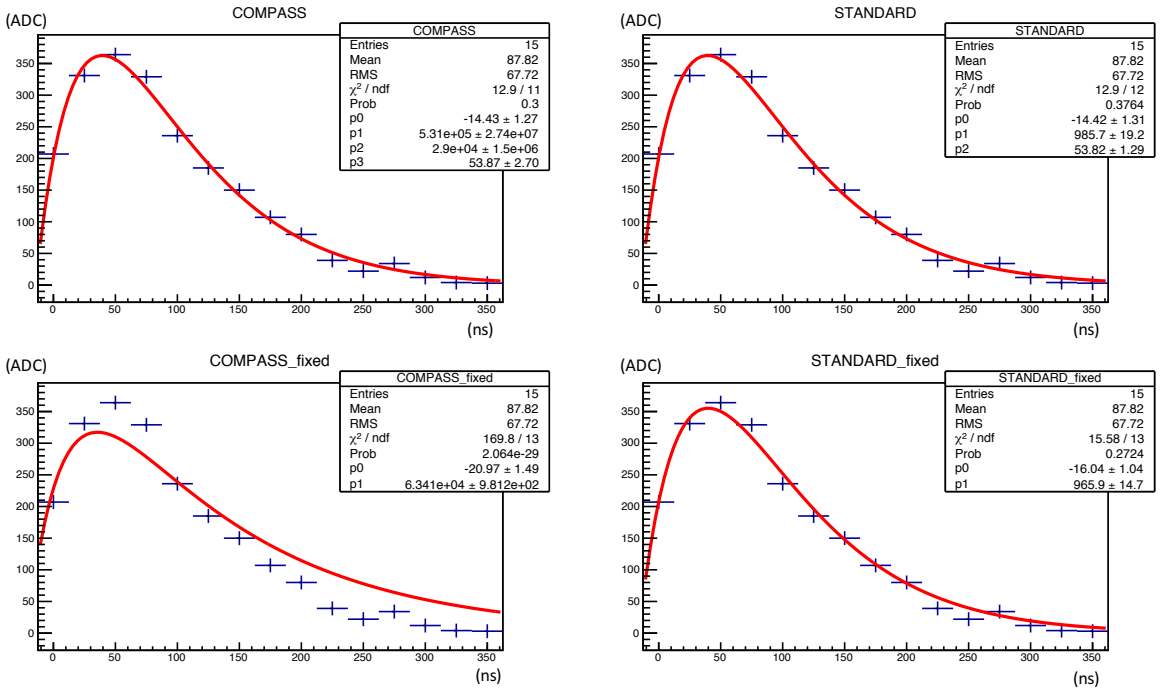


Figure 42: Top-left plot shows the data and fitting result using the COMPASS form. Top-right plot shows the same data but fitted using the standard form. Bottom-left plot uses the COMPASS form but with fixed τ_1 and $\tau 2$. Bottom-right plot uses the standard form with fixed τ.. For the COMPASS form, p_0 is the starting time t_0, p_1 is the normalization constant A, p_2 and p_3 are the shaping times τ_1 and τ_2 respectively. For the standard form, p_0 is the starting time t_0, p_1 is A, and p_2 is τ.

6.4 B-1 Basic Class introduction for SoLIDTracking

SoLIDSpectrometer is a top level class of this program. It must be initialized in the replay script by the user. Most of the jobs are performed by the objects that it owns rather than by itself. It provides an interface to the Hall A Analyzer, whose event loop will automatically call the functions of this class in order to perform all the analysis steps.

SoLIDTrackerSystem contains many tracking detectors, and the amount depends on the detector configuration. The tracking reconstruction functions are called inside this class and the reconstruction is done in all the tracking detectors that it owns. For the SIDIS configuration, this class should own all the GEM trackers and chambers. For PVDIS, each of this class should own only one chamber of each GEM tracker.

SoLIDGEMTracker represents one of the GEM trackers for SoLID. It may contains a number of GEM chambers depending on the configuration. At the moment, this class is not really perform any real calculations other than storing some geometric information for the tracker and serving as a container for the GEM chambers.

SoLIDGEMChamber represents one of the GEM chambers for a GEM tracker. Each of this

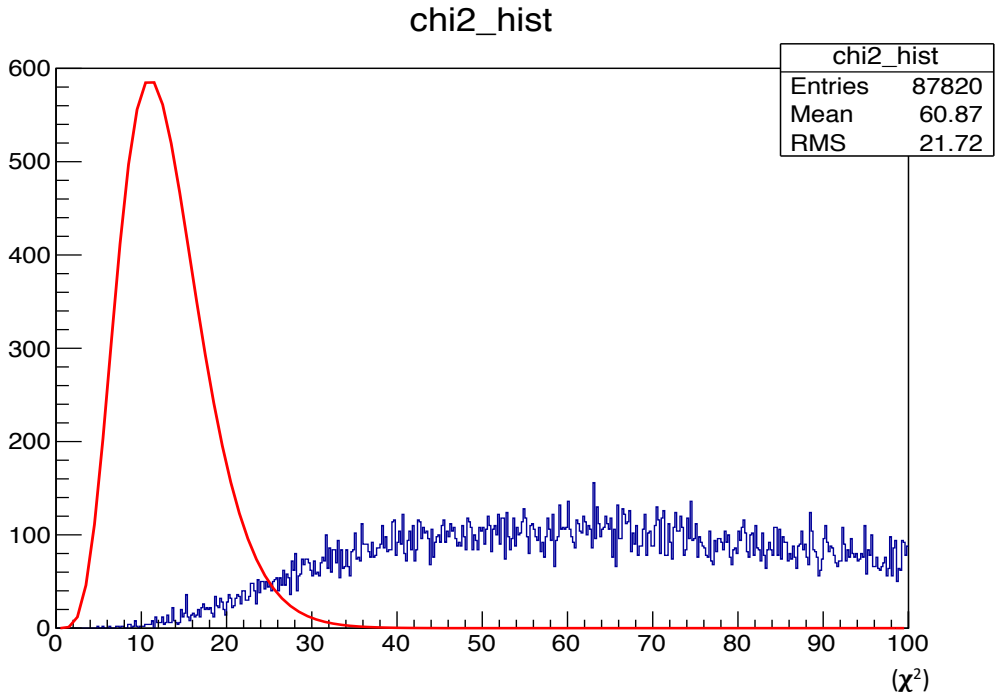
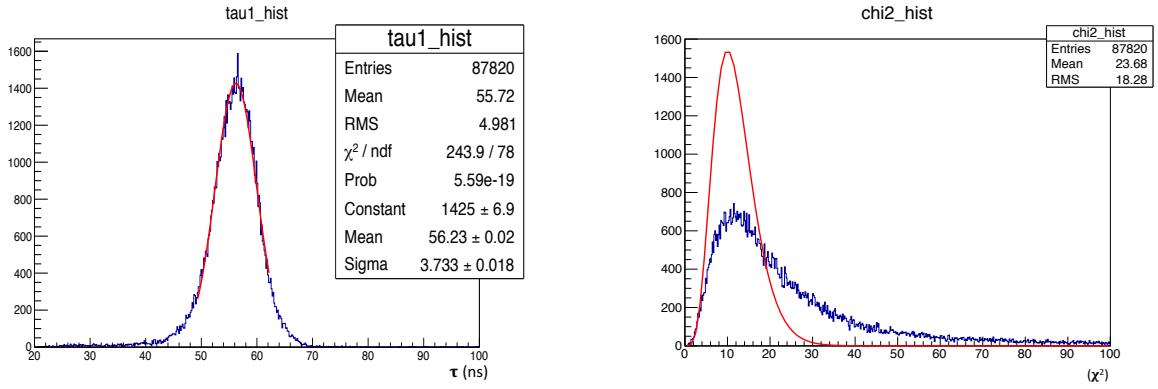


Figure 43: χ^2 distribution for fittings using the COMPASS form with fixed τ_1 and τ_2, red curve shows the expected χ^2 distribution for $ndf = 13$.



(a) 1a (b) 1b

Figure 44: Left plot: distribution of the shaping time τ using the standard form fitting function, red curve is a Gaussian fit from ROOT. Right plot: χ^2 distribution from fits using the standard form, red curve is the expected $\chi 2$ distribution with $ndf = 12$.

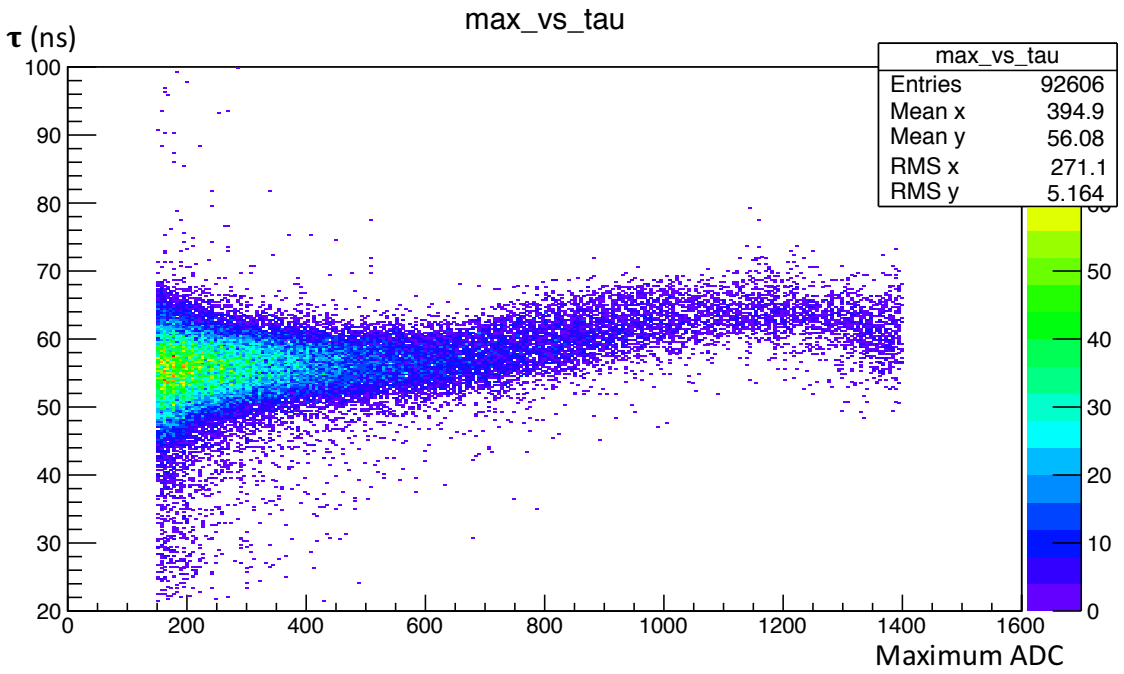


Figure 45: The shaping time τ from the fittings using the standard form as a function of the maximum ADC value of the signal, there is a clear dependency that larger signals tend to have slightly larger τ. Fittings for signal with maximum ADC > 1200 may not be reliable as many of them have the saturation issue.

class contains two sets of readout strips, as designed for SoLID. Other than storing some geometric information related to the chambers, this class can also combine the two 1D hit array from the strips into 2D hits.

SoLIDGEMReadOut presents one of the two sets of readout strips for each GEM chamber. It is responsible for doing the zero-suppresion for the GEM signals and performing the GEM cluster reconstruction. The output of this class is an array containing the reconstructed hit position.

SoLIDECal stores the information from the calorimeter (hit position and energy measurement), which we can use to start finding seed for the tracking reconstruction. At the moment, we are simply just using the MC recorded hit position and energy instead of those after some calorimeter digitizations and reconstruction. So this class does not contain any calorimeter reconstruction algorithms. The hit position an energy are simply just smeared by some Gaussian distributions with some pre-assumed resolutions. This class should be replaced once the calorimeter digitization and reconstruction is ready.

SoLIDGEMHit contains definitions for the 1D and 2D hits used for the GEM cluster reconstruction.

SoLIDTrack contains definition for the reconstructed track.

SoLIDFieldMap is a singleton class can is used to load the field map. It also provide a function to calculate the three components of the field given the Cartesian coordinate at a point, using the linear interpolation. Currently, it can only load the 2D field map file, which has a 1cm grid and is

azimuthal symmetric (only gives field at different r and z). For other type of field maps, this class may needs to be modified, but it should be a simple task.

SoLKalTrackFinder is a virtual class for the track finders in the program. Currently there are three track finders, each corresponds to a different detector configuration. They are SDISITrackFinder, PVDISTrackFinder and JPsiTrackFinder (maybe they can be actually combined into one, which can auto-configure based on different database files). They perform the track finding in a rather similar way. First, doublet seeds are searched from downstream trackers in their FindDoubletSeed() functions. And then some doublet seeds are merged in MergeSeed() functions because they may actually come from the same tracks. Next, the seeds will be extended toward the upstream in the TrackFollow() functions, in which new hits will be added to the track if they are close enough to the prediction of the track. Tracks that have multiple missing hits will be terminated. All the remaining tracks that are able to reach the upstream tracker will be propagated toward the target, where a special hit from the BPM will be added to the track in the FindandAddVertex() function. In the end, after some final selections in the FinalSelection() function, the reconstructed tracks will be copied to the output array.

SoLKalMatrix contains the definition of a matrix used in the KF. It is simply a wrapper around the TMatrixD in ROOT.

SoLKalTrackSite represents a measurement site in the literature of KF. Since KF allows the track parameters to be different at different sites, this class contains not only the hits from the GEMs but also many different state vectors or sets of track parameters and their covariance matrices, such as the predicted, the filtered, the smoothed and the inverse filtered. In addition, the class can calculate the predicted hit position based on the predicted state vector and also obtain the optimal state vector (the filtered) based on the measurement, the preidcted state vector and the covariance matrices.

SoLKalTrackState represents the state vector or a set of track parameters used in KF. Usually owned by a SoLKalTrackSite object.

SoLKalTrackSystem represents a reconstructed track in KF. It contains a number of SoLKalTrackSite objects, which in turn, contain a number of SoLKalTrackState objects. In addition, the reconstructed vertex variables such as the polar angle, azimuthal angle and momentum of the track is also stored in this class.

SoLKalFieldStepper is a singleton class that can take a KF state vector from one measurement site to another, using the information from field map and the 4th order Runge-Kutta method.

6.5 B-2 Database for SoLIDTracking

In this subsection I will introduce the database files needed in the SoLIDTracking reconstruction program. The first one is $db\_generalinfo.dat$, which is exactly the same as the one used in the libsolgem. The second one is $db\_solsim\_cratemap.dat$, which contains the total numbers of crates, slots and channels that are needed in the decoder. The third one is the field map $solenoid\_CLEOv8.dat$. It contains the magnetic field in the z and r directions at some given z and r position. In other words, it assumes that the field is azimuthal symmetric.

The most complicated one is the $db\_solid.trackersystem.dat$, which contains all the geometric information, parameters needed in the GEM cluster reconstruction and various flags that can be turned on/off during the analysis steps. First of all, all parameters need to start with the prefix "solid.trackersystem.". There are a set of numbers follow the prefix, separated by a period ".". The first number that follows is the tracker system ID, the second one is the tracker ID that the tracker system owns, the third one is the chamber ID that the tracker owns and the fourth one is the readout set ID that the chamber owns. An example for the prefix of a parameter needed for the first readout set, of the third chamber, of the second GEM tracker, or the first tracker system is given as:

$$\text{solid.trackersystem.0.1.2.0} \tag{9}$$

However, simply follow this pattern can easily make the database file unreasonably large and there will be many unnecessary repetitions. A number of phrases are introduced in order to ease this problem. For instance, if a parameter is common to all tracker system, all trackers, all chambers, and all readout. One can simply use the prefix:

$$\text{solid.trackersystem.\$\{allsystems\}.\$\{alltrackers\}.\$\{allchambers\}.\$\{allreadouts\}} \tag{10}$$

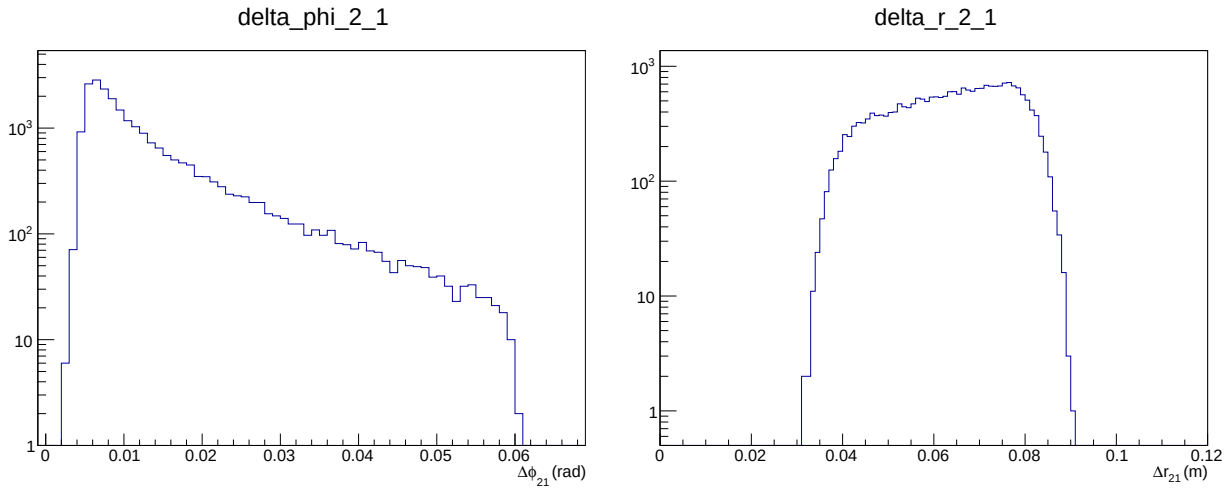
for this parameter. In addition, not all parameters need four IDs. For instance, if the parameter is used only at the tracker level, and it is common for all systems and all trackers, the prefix will simply be

$$\text{solid.trackersystem.\$\{allsystems\}.\$\{alltrackers\}.} \tag{11}$$

so the last two numbers are dropped.

Lastly, there is a database file for the track finders. Depending on the detector configuration, there are $db\_SIDISTrackFinder.dat$, $db\_PVDISTrackFinder.dat$ and $db\_JPsiTrackFinder.dat$. Most of them are related to the seed finding and are obtained using the simulation [20]. Most details related to how to obtain those parameters will be discussed in the next subsection of this appendix.

[20] Some of the hard coded parameters are still left in the track finders, they can be moved to these database files



(a) 1a

(b) 1b

Figure 46: Left plot: $\Delta\phi$ between hits on the second and the third GEM for tracks in the SIDIS forward angle region. Right plot: Δr between hits on the second and the third GEM for tracks in the SIDIS forward angle region. Index of a GEM starts from 0. Data sample requires that momenta $p > 0.9$ GeV and must hit the forward angle calorimeter.

6.6 B-3 Parameters for the track finders

For each of the track finders, there are a lot of parameters obtained from the simulation in order to assist the seed finding. Some of the ideas here are inherited from Xin Qian, and there are certainly many other ways one can do in order to achieve this. One possibility is to use some machine learning techniques to develop some models to find seeds. Such techniques may also significantly speed up the program as the seed finding algorithm we have right now is a $\mathcal{O}(N^n)$ algorithm, where N is the averaged number of hits on a GEM tracker, and n is the number of trackers we use to find a seed. Here, I will just describe the things I did in order to obtain some of the parameters. The idea involved here is rather simple; we select a set of simulated signal tracks (electrons from eDIS process for instance) with some kinematic cuts (satisfy the trigger condition for instance), and we study their characteristics in terms of some measurable or reconstructable variables, such as measured position or reconstructed angles based on the measured positions. For example, since our interested particles usually have high energy, their tracks are fairly straight compared to many background tracks and their coordinates on adjacent GEMs are usually highly correlated. One can plot the Δr and $\Delta\phi$ coordinates between two hits of a track on two adjacent GEM detectors, and set up cuts based on the selected data sample. For example, Figure 46 shows the Δr and $\Delta\phi$ between hits on the second and the third GEMs, for all the selected tracks in the SIDIS forward angle region. The requirement for the data sample is that momenta $p > 0.9$ GeV and must hit the forward angle calorimeter. In addition, one can also estimate the local momentum and angles based on the measured r and ϕ coordinates from two GEM trackers. The purpose of this type of parameters is to initialize Kalman Filter, which prefers local information to start with. We will need MC to provide the 3-momentum vector of a track as it crosses a GEM tracker, on which we want to initiate the track. First of all, the momentum of the track has a strong correlation with the $\Delta\phi$ between two hits as low energy tracks usually have a larger banding angle. One can find this correlation by plotting the local momentum from MC on the y-axis and $\Delta\phi$ on the x-axis, as shown in Figure 47. One can fit the distribution to get a function that describes the behavior, for which I used a hyperbola. For the angles, we can obtain a first order estimation by assuming the

track is straight in between two GEM trackers, in which case, the two angles are given by

$$\theta = tan^{-1}(\frac{\Delta r}{\Delta z}), \Phi = tan^{-1}(\frac{\Delta y}{\Delta x}) \tag{12}$$

where Δr, Δx and Δy are coordinate differences between hits on two different GEM trackers, and Δz is their distance in z. Please notice that Φ is the angle of the track, while ϕ as mentioned previously was the coordinate of hit. In the simulation, of course, tracks will have a certain deviation from the straight line case, especially for low energy tracks that have large $\Delta\phi$. One way to get a correction is to plot the difference between the MC local values and the first order reconstructed values on the y-axis and then $\Delta\phi$ on the x-axis, for instance the plots in Figure 48. One can then fit the distribution with some functions (I am using second order polynomials), and then add the fitted function to the first order reconstructed values to get the second order estimation. There are certainly other ways and better ways to do this. But as I only use these values to initialize KF, these second order reconstructed values seem good enough for now.

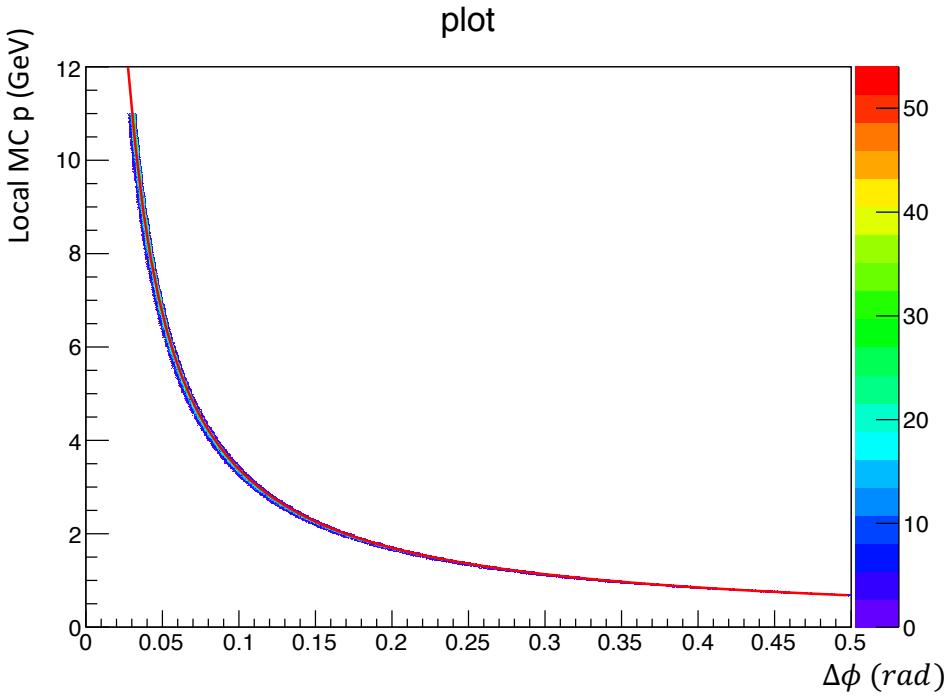


Figure 47: y-axis is the local momentum from MC on a GEM tracker and the x-axis shows the $\Delta\phi$ between a hit on the current GEM and the previous GEM. Red curve is a hyperbola fit to the distribution.

6.7 C-1 Other supporting files and a working directory on ifarm

A few supporting programs are worth mentioning here. And hopefully they will simplify certain tasks for others. We have a small program that can generate the *db_gemc.dat* files for libsolgem and *db_solid.trackersystem.dat* files for the tracking reconstruction program. It is located on GitHub `https://github.com/xweizhi/db_SoLIDTracking`. It can be used to generate files for the PVDIS and SIDIS configuration (J/ψ is almost the same as SIDIS).

A simple ROOT script in the "example" folder of libsolgem, named "split_file.cxx", can be used to split the large size background files into smaller ones. It is needed when adding beam-on-target

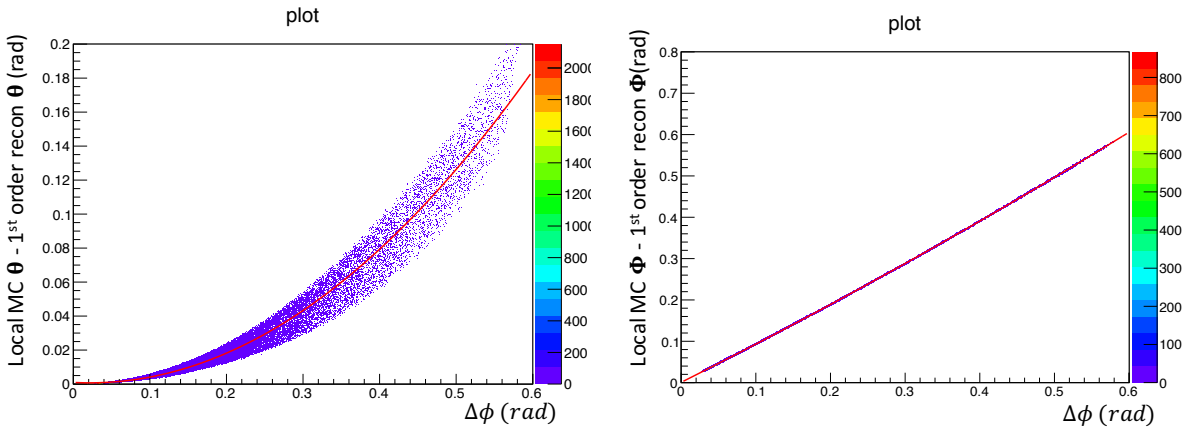


Figure 48: y-axis is the local momentum from MC on a GEM tracker and the x-axis shows the $\Delta\phi$ between a hit on the current GEM and the previous GEM. Red curve is a hyperbola fit to the distribution.

events in the GEM digitization. It is easier to use smaller files compared to the large simulation files, which is usually near 100 Gb.

References

[1] N. Bingefors *et al.*, Nucl. Instrum. Meth. A **326**, 112 (1993).

[2] Frank Simon, Diploma Thesis. https://vanderby.web.cern.ch/vanderby/dem/products/gem/franks_diploma.pdf

[3] R. Mankel, Rept. Prog. Phys. **67**, 553 (2004)

[4] Keisuke Fujii, Reference Manual for KalTest. http://www-jlc.kek.jp/subg/offl/kaltest/doc/ReferenceManual.pdf

[5] Erik Krebs, Application of a Kalman filter and a Deterministic Annealing filter for track reconstruction in the HADES experiment. https://hades.gsi.de/sites/default/files/web/media/documents/thesis/Master/Application_of_a_Kalman_filter_and_a_Deterministic_Annealing_filter_for_track_reconstruction_in_the_HADES_experiment_E._Krebs_2013-Aug.pdf

[6] SoLID Updated Preliminary Conceptual Design Report. http://hallaweb.jlab.org/12GeV/SoLID/download/doc/solid_precdr_2017.pdf

[7] W. Adam, R. Fruhwirth, A. Strandlie and T. Todorov, eConf C **0303241**, TULT009 (2003) [J. Phys. G **31**, N9 (2005)]