SBS Software Status

Andrew Puckett
University of Connecticut
SBS Collaboration Meeting
July 15, 2020

UCONN

Outline

- SBS Software Overview
- SBS software working group and weekly meeting
- Progress since last collaboration meeting
  - GEM reconstruction software development and analysis highlights
  - Recent simulation studies and geometry updates
- Issues/problems/delays
- Current status and timelines

Existing SBS Software

- SBS online/offline analysis software will be based on Podd, the standard C++/ROOT-based Hall A analysis framework, and use the ROOT-based "OnlineGUI" for online monitoring plots for shift workers, etc.
- Existing repositories:
  - **SBS-offline**: (principal authors: S. Riordan, A. Puckett, E. Fuchey, O. Hansen, J. C. Cornejo) https://github.com/JeffersonLab/SBS-Offline Main software repository of SBS-specific libraries, source codes, database files and replay scripts. Includes raw data decoders for all currently planned SBS DAQ modules and basic skeleton classes for all major SBS subsystems, within Podd/analyzer framework. Not yet in widespread use for detector commissioning. **Already in use in HCAL commissioning (and BigBite timing hodoscope analysis).**
  - **Libsbsdig**: (principal author Eric Fuchey) https://github.com/JeffersonLab/libsbsdig Main library for digitization of simulation output; translates *g4sbs* output (hit time, position, energy deposit) into simulated raw detector signals ("pseudo-data"), populates "hit" data structures used by reconstruction (ADC, TDC, crate, slot, channel, etc); purpose is to test and develop reconstruction algorithms on simulated events using identical algorithms to those used for real data.
  - **Libsolgem/libsbsgem:** (principal author Eric Fuchey; adapted from SOLID gem digitization) https://github.com/JeffersonLab/libsolgem State of the art GEM digitization, "tuned" to real data. Converts simulated GEM hits (position, time, energy deposition) to pseudo-raw data (strip ADC samples, including effects of pedestal noise, common-mode noise, and crosstalk)
  - **G4sbs:** (principal authors Andrew Puckett, Seamus Riordan, Eric Fuchey, many contributors) https://github.com/JeffersonLab/g4sbs GEANT4-based simulation of all of the SBS experiments. Documentation at https://hallaweb.jlab.org/wiki/index.php/Documentation_of_g4sbs
  - **SBSGEM_standalone:** (principal author A. Puckett) https://github.com/ajpuckett/SBSGEM_standalone standalone GEM reconstruction code, takes decoded raw data (after common-mode/pedestal subtraction and zero suppression), does clustering, tracking, and alignment. Already used extensively in GEM commissioning.

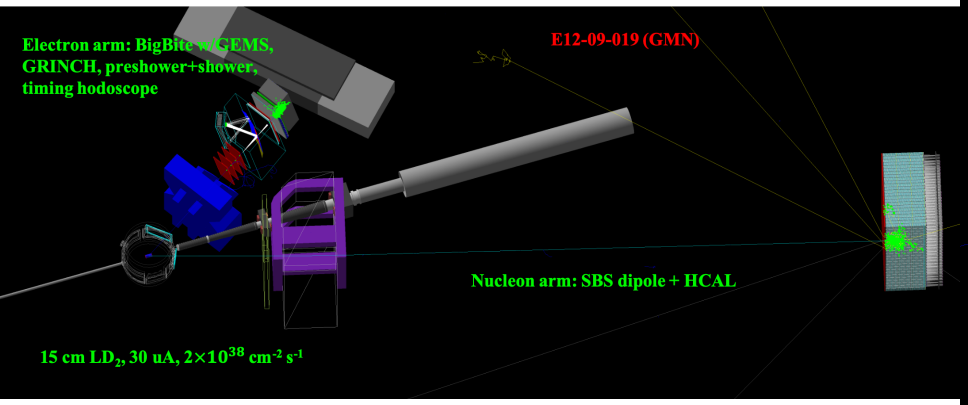
SBS software working group

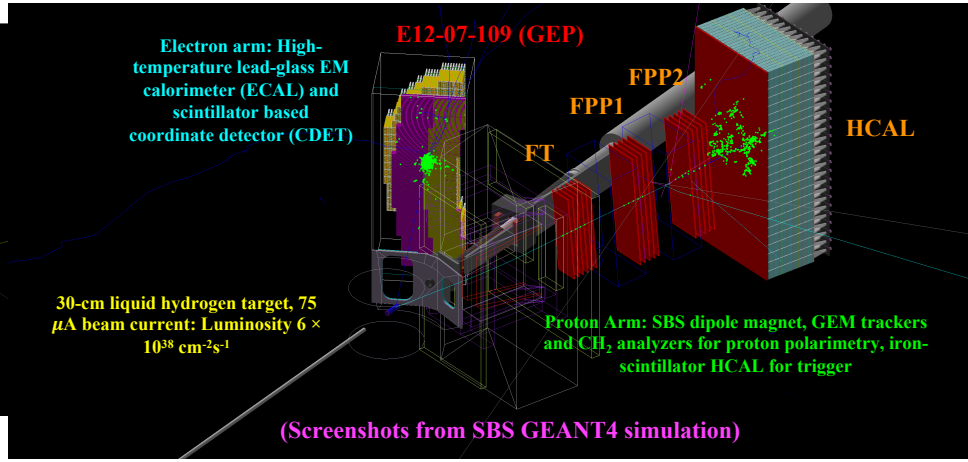
- Mailing list: https://mailman.jlab.org/mailman/listinfo/Sbs_software
- Standing weekly meeting; currently Fridays at 1:00 PM
- SBS Software Coordinator: Andrew Puckett
- Core software working group members/participants:
  - JLab: J.-O. Hansen, A. Camsonne, M. Jones, S. Barcus, D. Flay, H. S.-Vance, B. Wojtsekhowski
  - CMU: J.-C. Cornejo, B. Quinn
  - Glasgow: R. Montgomery, D. Hamilton, R. Marinaro
  - Syracuse: W. Xiong
  - UConn: A. Puckett, E. Fuchey, P. Datta, S. Seeds
  - Hampton U: M. Kohl, T. Gautam *et al.*
  - Northern Michigan U: W. Tireman
  - Significant contributions from UVA, W&M, others

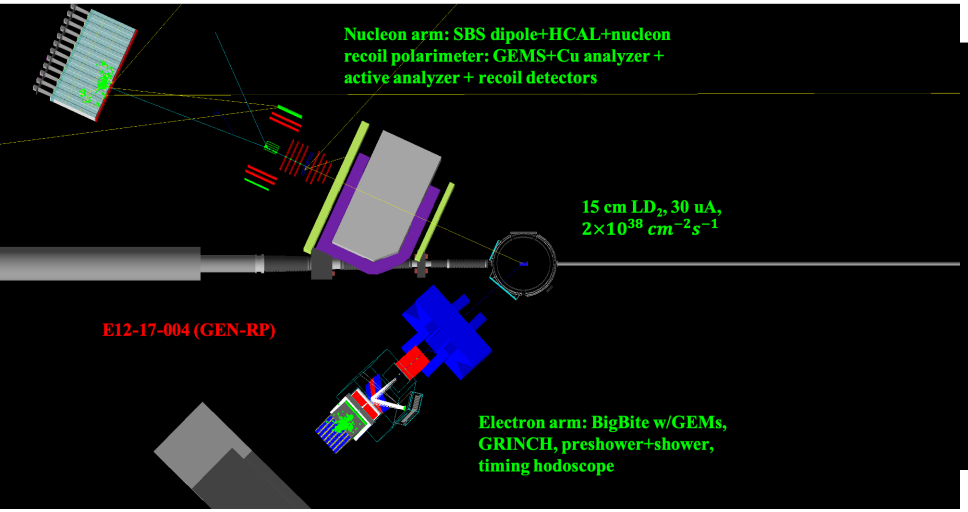
G4sbs: SBS Monte Carlo Simulation

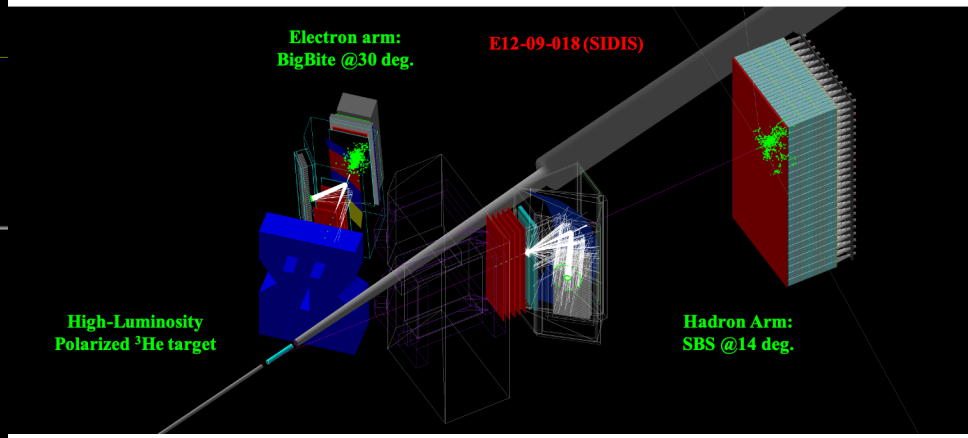
- ***G4sbs* is a success story within the overall SBS software effort:**
  - Many users and contributors from inside and outside the core collaboration; **successful use in new proposal development and approval, assisting in detector design, and even in reanalysis of Hall A GEN data (E02-013).**
  - Self-contained GEANT4 application with self-documenting ROOT output including version control
  - Github version control and code maintenance
  - CMake-based build system works "out of the box" on most Linux and Mac OS systems
  - Minimal external dependencies (only ROOT, GEANT4, and python required), and planning to keep it that way!
  - Thorough documentation, maintained by the UConn group
  - Flexible geometry configuration/event generation machinery
  - Straightforward addition of new detectors/geometries with standardized ROOT outputs using pre-defined "sensitive detector" types that can handle many use cases without modification.
  - **STL vectorization of array-valued output ROOT Tree branches**—completely dynamically sized arrays, easier to read back with far less "memory waste" during analysis.
  - Built-in event generators for main processes/targets of interest for SBS program
  - Flexible interface to generic external event generators—separate event generation from detector simulation, re-use same events in many detector configurations.
  - Anyone with reasonable C++/GEANT4/ROOT proficiency can contribute (and many have!)
- **Plan is to replicate successful *g4sbs* design approach for SBS-offline, within Podd framework**
- **See more details from Eric Fuchey's simulation/digitization talks**

SBS Experiment Layouts in Monte Carlo









- SBS Monte Carlo simulation: https://hallaweb.jlab.org/wiki/index.php/Documentation_of_g4sbs

SBS GEM standalone reconstruction code

- Standalone GEM reconstruction (and alignment) code, written by yours truly, stored and maintained in github repository: https://github.com/ajpuckett/SBSGEM_standalone
- This is already being used to analyze decoded data from a wide variety of test setups:
  - INFN cosmic data: 4 layers of 3 GEM modules each, 40x50 cm^2 per module area
  - UVA GEMs, beam test in Hall A, 2016, 5 layers one module each, 50x60 cm^2
  - UVA EEL clean room data, 2019-2020, 4 and 5-layers data with four modules each, 16-20 modules total
  - GEMs in HRSs during PREX/CREX
  - Simulated GEP data (see Weizhi's talk)

README.md

SBSGEM_standalone

Standalone ROOT macros for analysis of GEM data.

Requirements: Working ROOT installation; decoded GEM data in standard ROOT Tree format.

GEM_reconstruct.C: main clustering/hit reconstruction/tracking code; developed for analysis of GEM data from INFN cosmic ray test stand. Takes decoded GEM data as input (strips fired by module and six ADC samples from APV25), reconstructs 2D hits, finds straight-line tracks, produces diagnostic histograms and ROOT Tree.

GEM_reconstruct_HallAtest.C: same as GEM_reconstruct.C, but specialized for analysis of Hall A GEM test data from 2016. Uses additional information from the calorimeter that was part of the test; computes simple sum of calorimeter ADC values.

GEM_reconstruct_standalone_consolidated.C: consolidated version with capability to analyze three different data formats: Hall A GEM test 2016, INFN cosmic stand, UVA EEL cosmic stand. THIS IS THE VERSION THAT WILL BE DEVELOPED AND SUPPORTED GOING FORWARD!

GEM_align.C: Alignment code; takes the output of GEM_reconstruct.C as input, determines best set of translational (x0, y0, z0) and rotational (alpha_x, alpha_y, alpha_z) "yaw, pitch, roll" offests to minimize chi-squared of straight line tracks.

Usage example:

root [0] .L GEM_reconstruct.C+ root [1] GEM_reconstruct("../HitData/GEMfixNov18 /Hit_run3805_*.root","configINFN.txt","temp.root");

General usage: GEM_reconstruct(infilename, configfilename, outfilename);

Here infilename gives the absolute or relative path name from the working directory to the decoded GEM data file in ROOT Tree format.

configfilename gives the name of the text configuration file, which defines the geometry and all relevant parameters for the clustering, hit reconstruction, and tracking.

outfilename is the desired name of the output ROOT file.

Similarly, for the alignment code:

root [0] .L GEM_align.C+ root [1] GEM_align(inputfilename, configfilename, outputfilename);

where inputfilename is the name of a root file that is the output of GEM_reconstruct containing the relevant reconstructed hit and track information, configfilename is the text configuration file, and outputfilename is the name of a text output file containing the new alignment parameters for each module. The output of GEM_align can be directly copied and pasted into the configuration file for GEM_reconstruct to repeat the tracking with the new alignment parameters.

Example configuration files are:

configINFN.txt: Example configuration file for INFN GEM four layer cosmic test data from Nov. 2018 configHallAtest.txt: Example configuration file for five layer UVA GEM Hall A beam test data from 2016. configalign.txt: Example alignment configuration file for INFN four-layer cosmic test data. configalignHallAtest.txt: Example alignment configuration file for five-layer UVA GEM Hall A beam test data 2016.

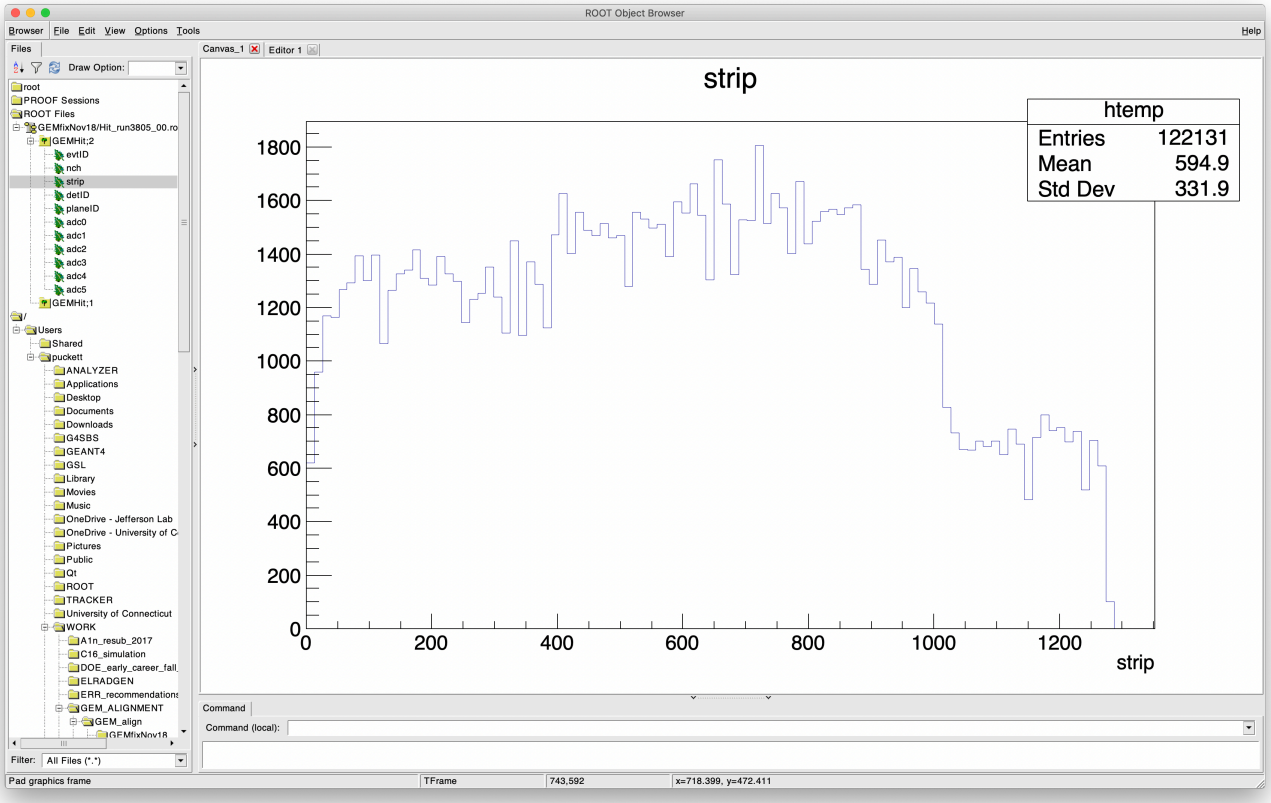
I have also added two utility macros:

plot_GEMrun_summary.C plot_GEMrun_summary_HallA.C

which can be used to generate a PDF file with a nice "standard" group of summary plots of the analysis.

Detailed documentation of configuration parameters is forthcoming. In the mean time, read the source code.

Decoded Data Format—"Hit" ROOT Files



- SBS GEM readout is based on APV25 front-end; six ADC samples at 25-ns intervals
- All analysis to be presented here is based on decoded data prepared by UVA/INFN GEM teams AFTER common-mode/baseline/pedestal subtraction and zero suppression

- "evtID" = event ID number
- "nch" = number of strips fired, AFTER common-mode subtraction, zero suppression
- "detID" = module number
- "planeID" (0 or 1): 0 (1) = Y (X) strips (measuring horizontal (vertical) coordinate)
- "strip" = strip fired
- adc0,1,2,3,4,5: ADC samples

Standalone/production code: General features

- Flexible enough to handle arbitrary layout/geometry.
- Main assumption is that a GEM module has two non-parallel orientations of strips.
- Code is written to handle arbitrary strip orientations (in anticipation of new modules with U/V strip geometry being built by UVA).

Hit Reconstruction: Clustering algorithm

1. Filter strips by requiring the max. ADC sample on a strip and the sum of all samples on a strip to be above some user-adjustable thresholds.
   a) No timing cuts or fancy deconvolution algorithms or splitting overlapping clusters yet, but all these are being evaluated with real and simulated data with realistic background superimposed
2. From remaining strips, select candidate "pixels" (combinations of one X and Y strip) to seed cluster formation based on local maxima of $\sqrt{ADCX * ADCY}$, where the ADC values refer to the sum of all six samples on a strip, or, alternatively, a subset of the samples (e.g., 2,3,4) where the signal/background ratio is largest.
3. Allow any strip to be reused in as many clusters as desired; consider all local maxima of ADCX*ADCY (Note that the number of "fake" hits according to this criterion grows like the square of the number of "real" hits) either within the entire module, or within some allowed search region defined by constraints on the locations of "good" tracks from external detectors.
4. For each local maximum of ADCX*ADCY, add all fired contiguous nearest neighbor strips in X and Y up to some user defined maximum limits (currently +/- 4 strips from the maximum in X, +/- 3 strips from the maximum in Y)
5. Once we are done adding strips, compute the Pearson correlation coefficient of the cluster-summed individual ADC samples. This is a measure of the degree of correlation between the X/Y ADC values and the timing of the signals.
6. Optionally filter clusters based on ADC asymmetry and X-Y time difference.
7. Send all candidate clusters passing filtering criteria to tracking algorithm

Tracking

- Track-finding and reconstruction in standalone code is done by "educated brute force", "Kalman Filter-like" approach:
  - Consider all possible reconstructed 2D hits in first layer with available hits
  - For each hit in first layer, consider all possible hits in next layer with available hits compatible with a straight-line track with $\left|\frac{dx}{dz}\right| \leq 1$, $\left|\frac{dy}{dz}\right| \leq 1$ (or other user-adjustable maximum slope along x and y).
  - Third and subsequent layers: for each potentially valid hit combination from the first N-1 layers, consider all hits in layer N falling within some reasonable distance from straight-line projection based on the first N-1 layers; update track parameters "on the fly" with each new hit added (i.e., "evolve the state")
- Prefer N-plane tracks initially, if no N-plane track is found with acceptable χ^2, reduce number of required hits by 1 and try again, as long as the number of required hits is at least 3.
  - When the number of layers required on the current tracking iteration is less than the total number of layers with available hits, we test all possible combinations of $k<N$ layers chosen from among N layers so that we don't bias the search in favor of some tracking layers over others.
- **Analysis rate on INFN/UVA cosmic data and 2016 beam test data at ~1.5% occupancy is of order ~1-2 kHz, depending on the thresholds/cuts applied at the clustering stage**
- This approach should scale to high-rate tracking (within reason), assuming the implementation of constraints from other detectors (calorimeters, CDET, GRINCH, hodoscope, etc) to limit the search region for clusters/tracks *before* 2D clustering. Such constraints are natural to implement within this approach
- **Preliminary results from simulated GEP tracking indicate that this track-finding algorithm works with acceptable efficiency, low false-positive rate, and acceptable speed, up to about ~20% of full GEP occupancy, and ~100% of worst-case GMN background.**
- **For higher-rate tracking, as in GEP, we will need a faster track-finding algorithm, *e.g.*, a "3D TreeSearch" (under development, see Weizhi's talk)**

Alignment, I

- Linearized equations for local/global coordinate transformation:
- Assume rotational offsets of modules are small enough that we can treat all cosines as 1, sines as the angles themselves

$$x_g = x_0 + x_l - \alpha_z y_l$$
$$y_g = y_0 + y_l + \alpha_z x_l$$
$$z_g = z_0 + \alpha_x y_l - \alpha_y x_l$$

$$\begin{aligned}
(x_g, y_g, z_g) &= \text{Global hit coordinates} \\
(x_l, y_l, z_l = 0) &= \text{Local hit coordinates (internal to module)} \\
(x_0, y_0, z_0) &= \text{Global coordinates of module center} \\
(\alpha_x, \alpha_y, \alpha_z) &= \text{Rotational offsets of module } \textit{wrt} \text{ global coordinate system}
\end{aligned}$$

$$\chi^2 = \sum_{i=1}^{N_{tr}} \sum_{j=1}^{N_{hit}} \left(x_g^{(ij)} - \left[x_{tr}^{(i)} + x'^{(i)}_{tr} z_g^{(ij)} \right] \right)^2 + \left(y_g^{(ij)} - \left[y_{tr}^{(i)} + y'^{(i)}_{tr} z_g^{(ij)} \right] \right)^2 ,$$

- Determine offsets and rotations for each module by solving a (linear) system of equations for the offsets that minimize χ^2 defined in terms of track residuals
- Set up and invert a matrix, easy-peasy.
- Iterate the process until the changes on subsequent iterations are smaller than the stat. uncertainties.
- Re-run reconstruction with new offsets, tighter χ^2 cut for tracking, lather, rinse, repeat.

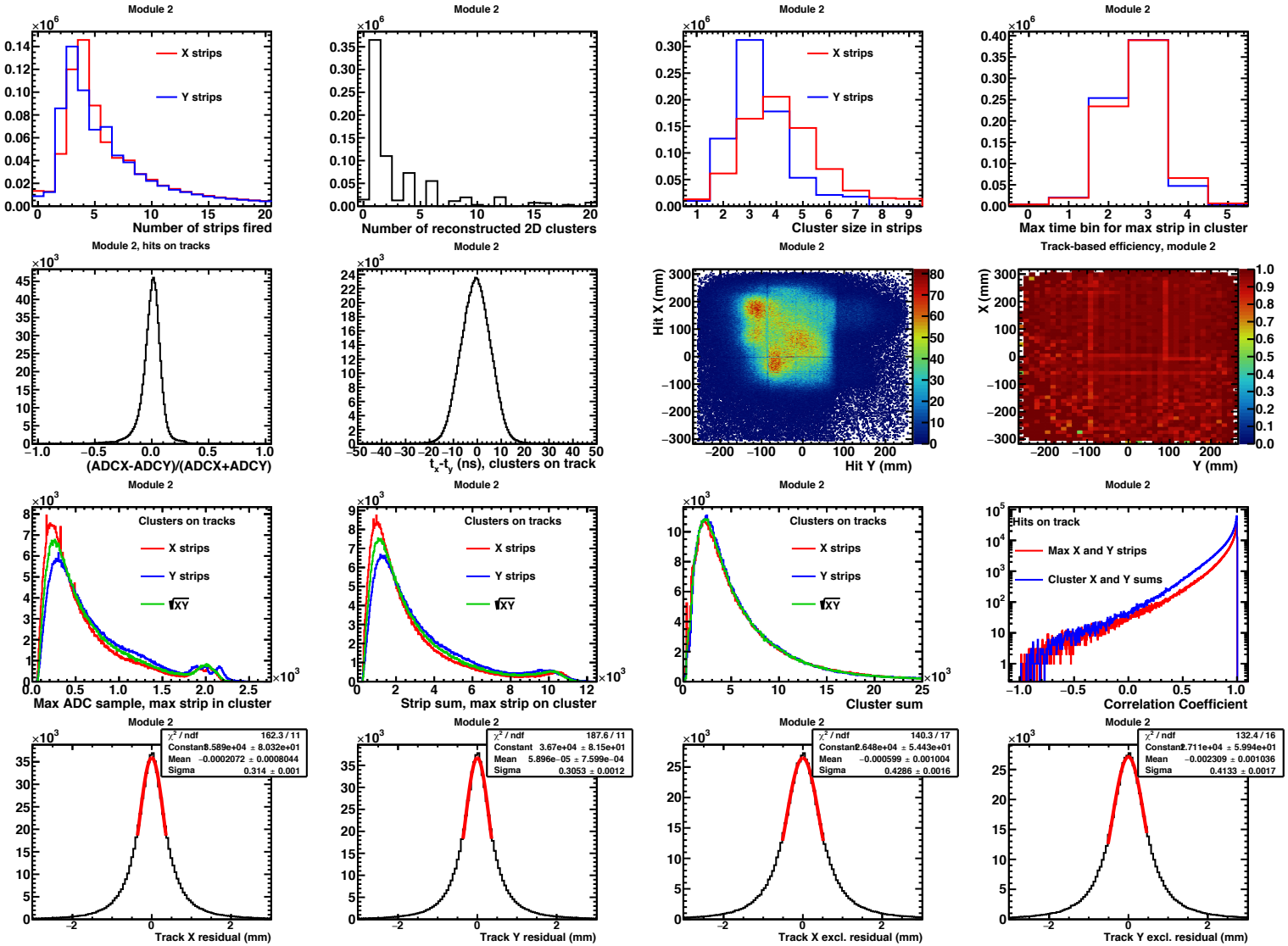
Alignment, II

- See GEM_align.C in the standalone github repo: https://github.com/ajpuckett/SBSGEM_standalone
- The code is quite configurable, allows one to ”fix” all parameters of one or more modules, and fit positions of all other modules relative to that one, perform different numbers of iterations where the track parameters are updated on subsequent iterations based on the results of the previous iterations, etc. etc.
- Convergence and quality of solutions depends on the quality of the initial guesses for module positions and orientations.
- At the end of the linearized procedure, the tracks are fixed; the linearized approximation is removed, and a final fit of the translational and rotational offsets relative to the outcome of the linearized procedure is done numerically using MINUIT.
- **Typical alignment workflow:**
  - Run reconstruction with “educated guesses” for initial module positions, from surveys or crude measurements or whatever. Use loose chi^2 cuts for tracking.
  - Make “coarse” adjustments by hand based on initial tracking results, offsets of residuals from zero by module, run reconstruction again.
  - Once a decent “coarse” alignment is achieved, run alignment code, get parameters
  - Run reconstruction again with tighter track chi^2 cuts.
  - Run alignment again with tighter track chi^2 cuts.
  - Repeat until spatial resolution stops improving.

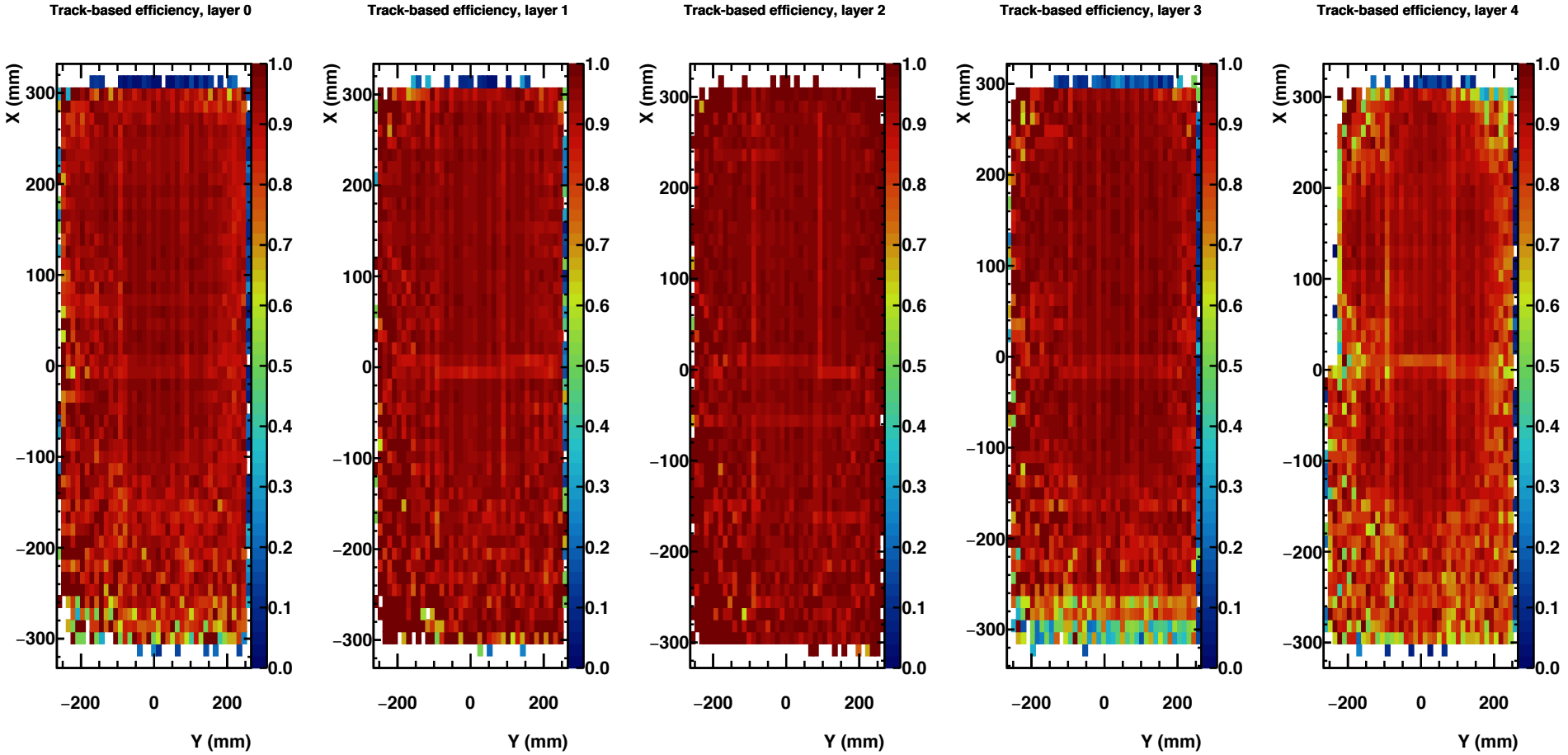
Alignment, III

- General comments: For this alignment procedure to work well, the initial guesses have to be reasonably good
- For rotational offsets and z offsets to converge, tracks used for alignment must populate large fraction of module active area in position.
- Cosmic data are excellent for alignment, due to wide angular distribution that is largely uncorrelated with position; good independent sensitivity to rotations, and z positions
- If tracks used for alignment are concentrated in a small area of the GEM and are all at close to normal incidence, it is more difficult to get rotation offsets and z position offsets to independently converge.
- In a production environment, good initial survey, possible low-current "zero field" runs, inelastic electron scattering from multi-foil targets populating large fraction of acceptance with wide angular distribution will be helpful for alignment

Per-Module Diagnostic Plots, example (Hall A test, 2016):

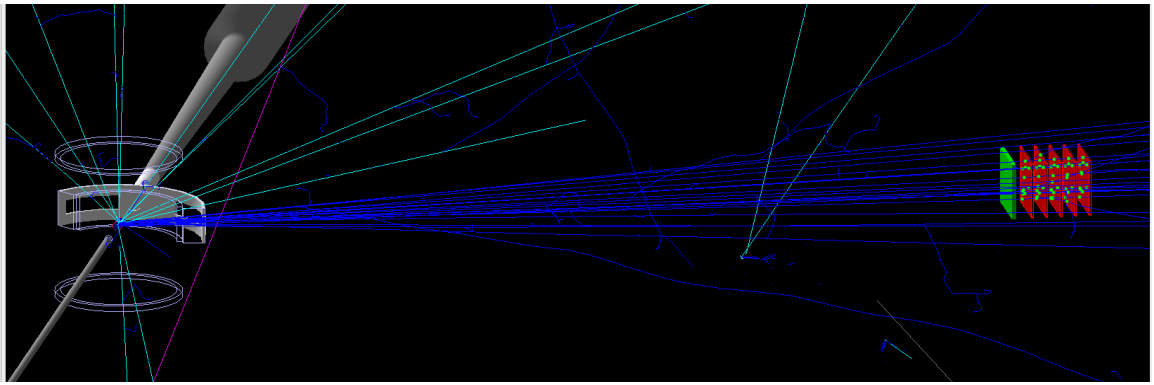


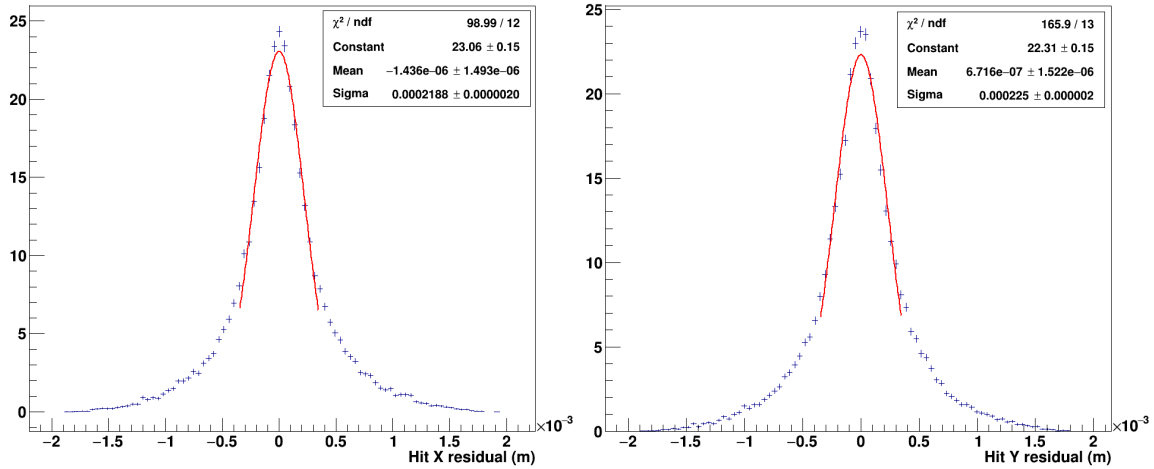
Hall A Test, 2016: Local, Track-Based Efficiency

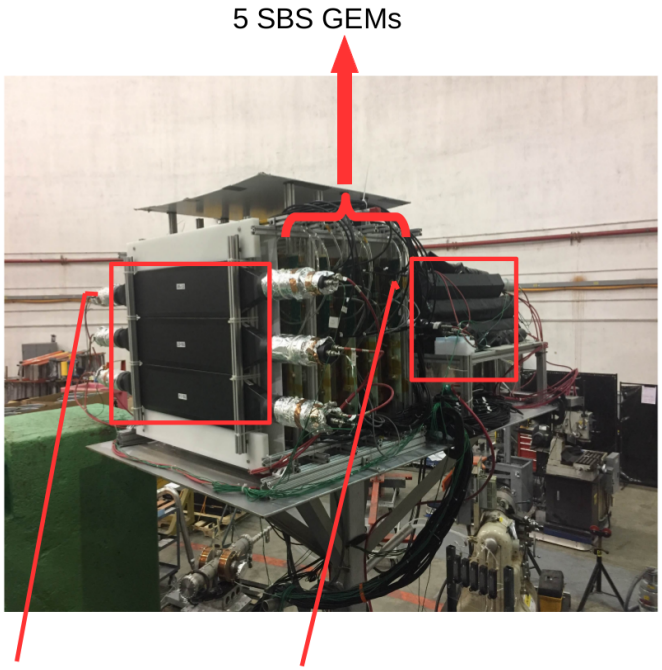


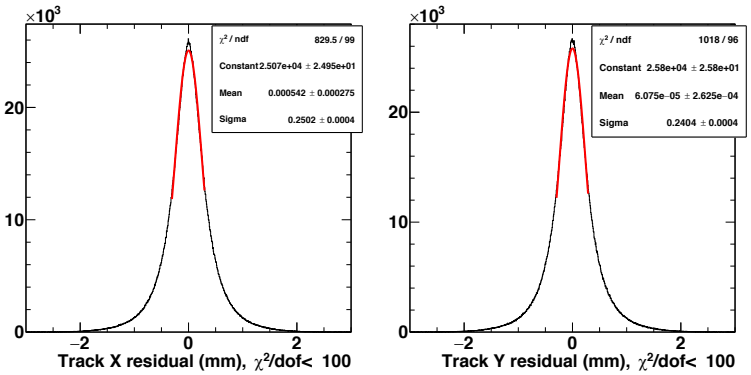
Track-based efficiency is defined as the probability that a hit occurred on a module within some maximum distance from the projection of a track fitted to all the other modules

Hall A Test, 2016: Spatial Resolution



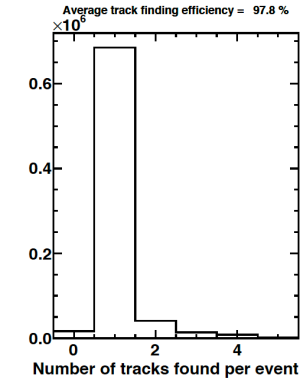


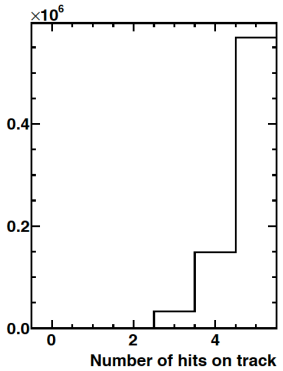


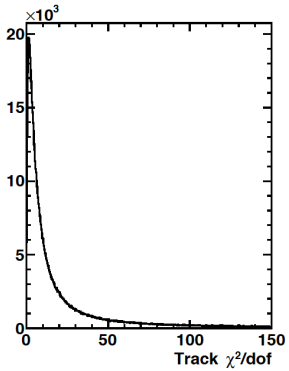


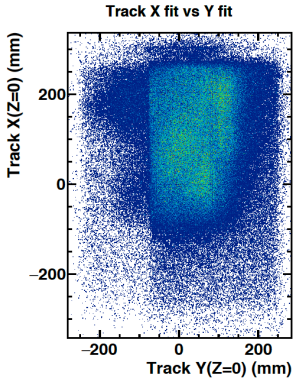
- Comparing real (bottom right) and simulated (above) tracking residuals using equivalent definitions shows that the spatial resolution during this test was dominated by multiple scattering.
- Most of the tracks through the GEMs were electrons with energies ~150-1100 MeV.

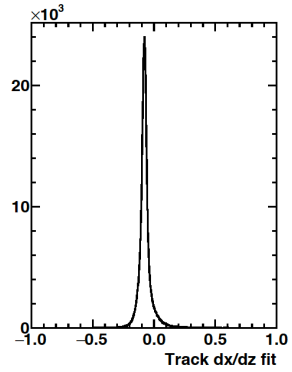
Tracking results: 2016 Hall A test

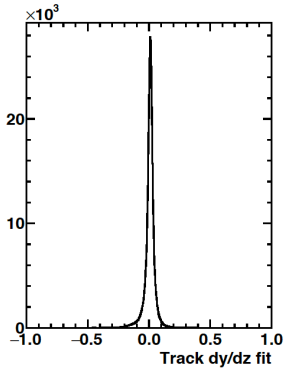


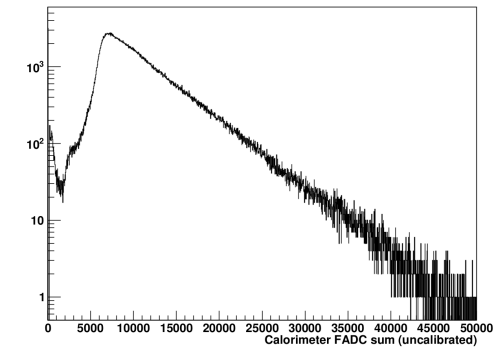


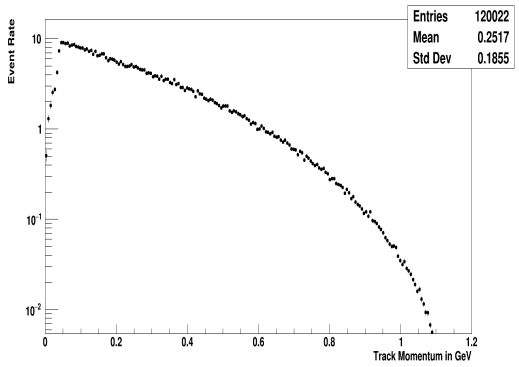






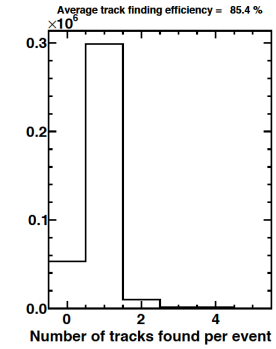


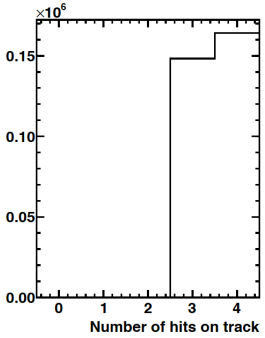


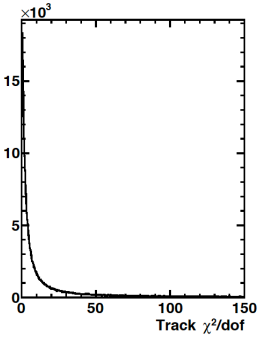


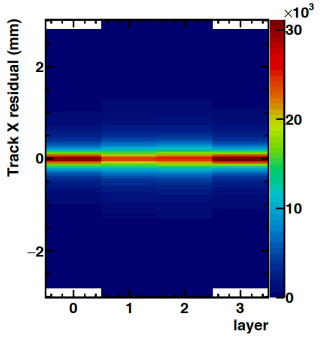
- 5-layer tracking efficiency ≥ 95% based on events with two good scintillator TDC hits and calorimeter FADC sum > 5,000 ADC channels
- Tracking-finding efficiency for events with at least one 2D cluster in at least 3/5 modules is 97.8%
- Efficiency results based on track reduced $\chi^2 \leq 300$, which corresponds to maximum tracking residual of ~5 mm for tracks with all 5 layers fired.
- Bottom left: Uncalibrated ADC sum from calorimeter and simulated track momentum distribution of inelastically scattered electrons for the kinematics of the test

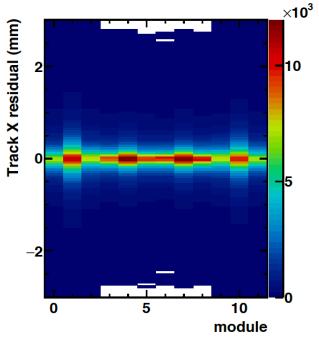
INFN GEM performance (Cosmic Run 3805, 2018)

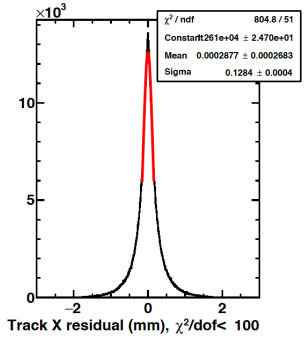


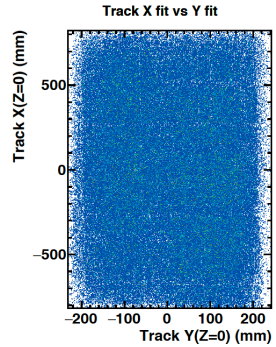


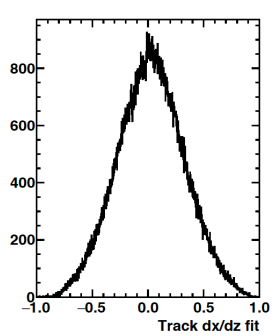


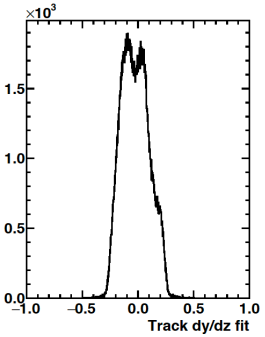


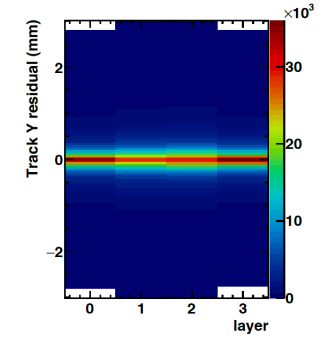


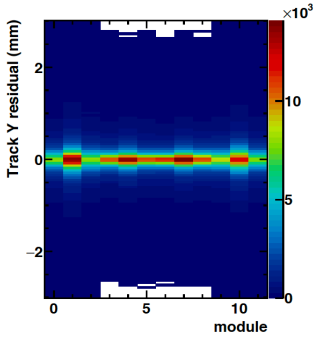


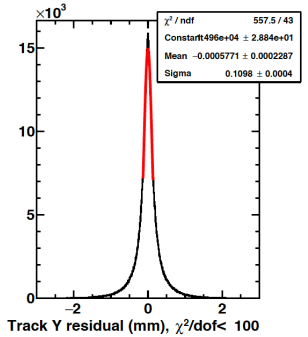




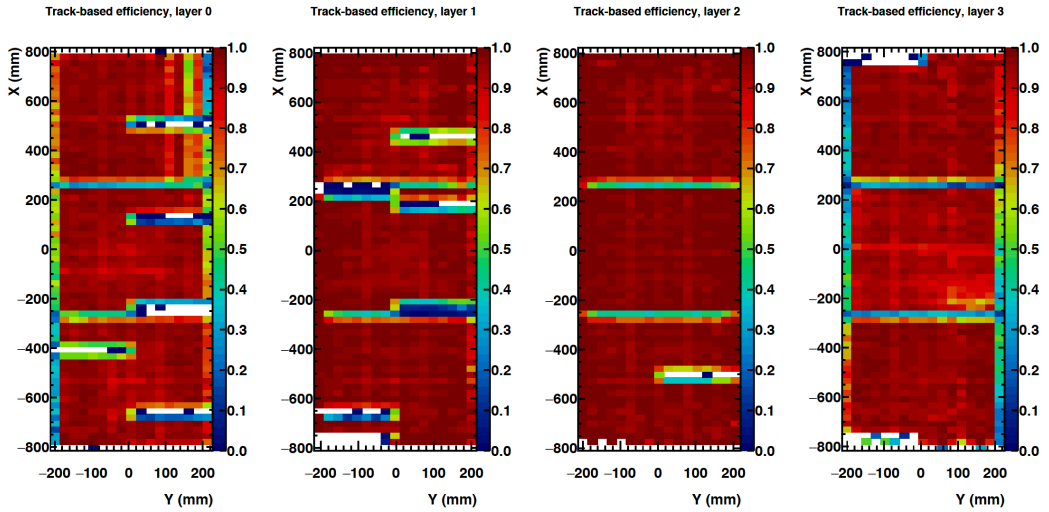




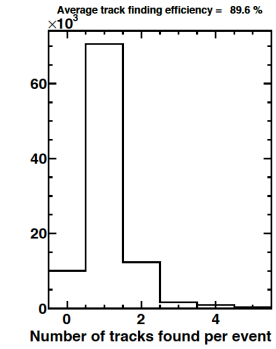


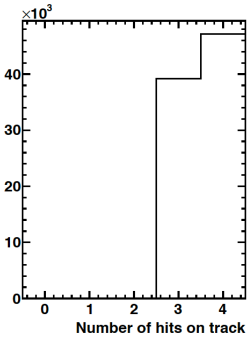


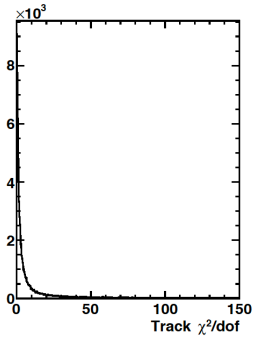
- Local efficiencies in “good” regions of the GEM typically above 95%
- Spatial resolution as measured by tracking residuals ~110-130 micron
- Overall 4-layer track-finding efficiency of 85% (not corrected for effects of dead areas)
- Optimal arrangement of layers and selection of best modules will increase efficiency (see E. Cisbani talk)

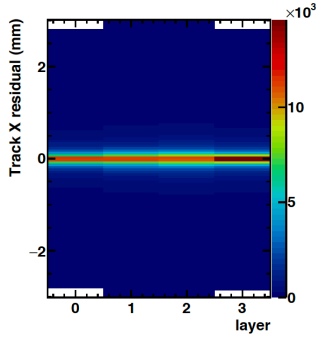


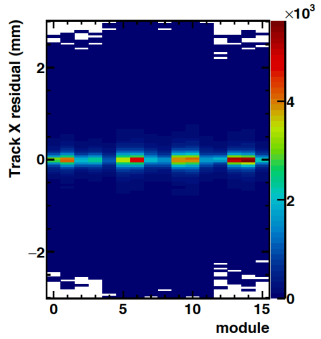
UVA GEM 4-layer performance w/cosmic rays (2020)

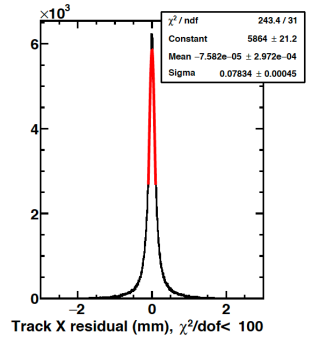


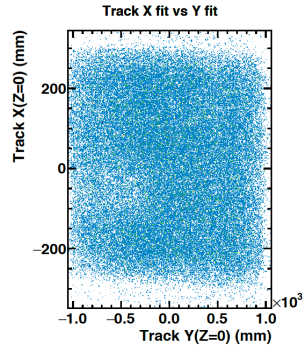


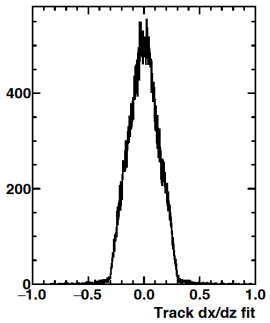


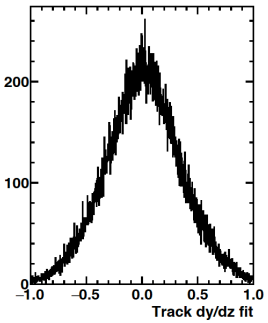


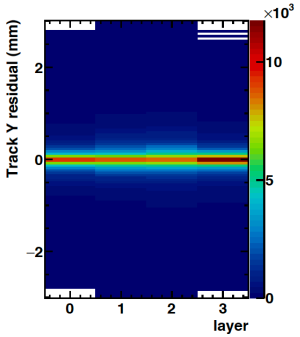


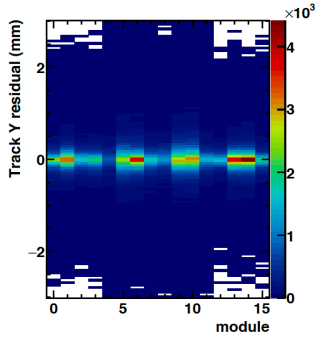


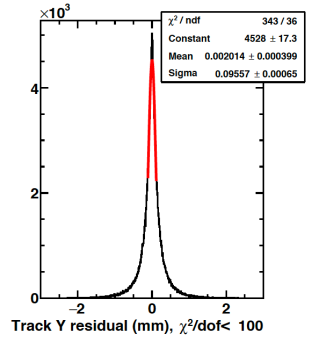




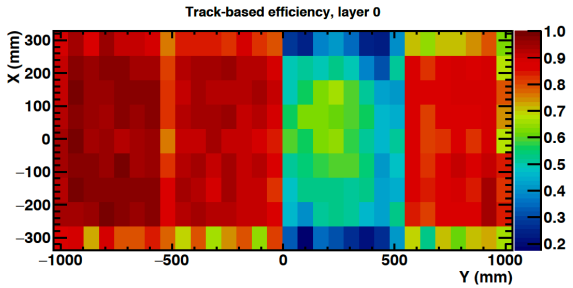


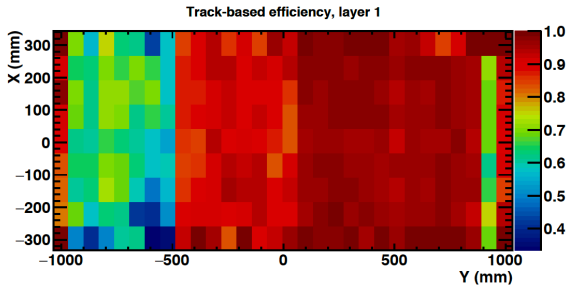


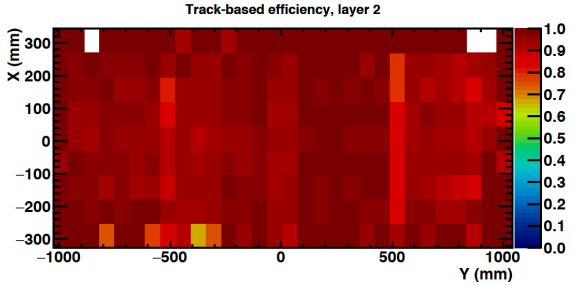


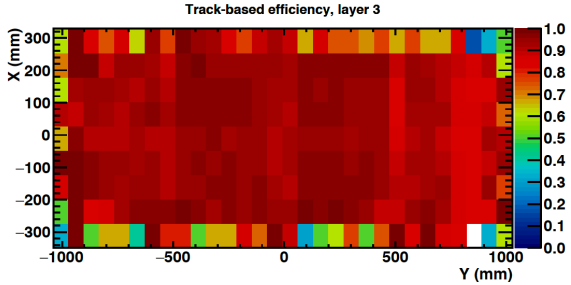


- Except for two modules with known lower gain at operating HV, all modules 95% efficient
- ~90% overall 4-layer track-finding efficiency
- Spatial resolution ~78 (96) microns in X (Y)
- Results obtained before optimization of trigger latency, and correcting noise/grounding issues



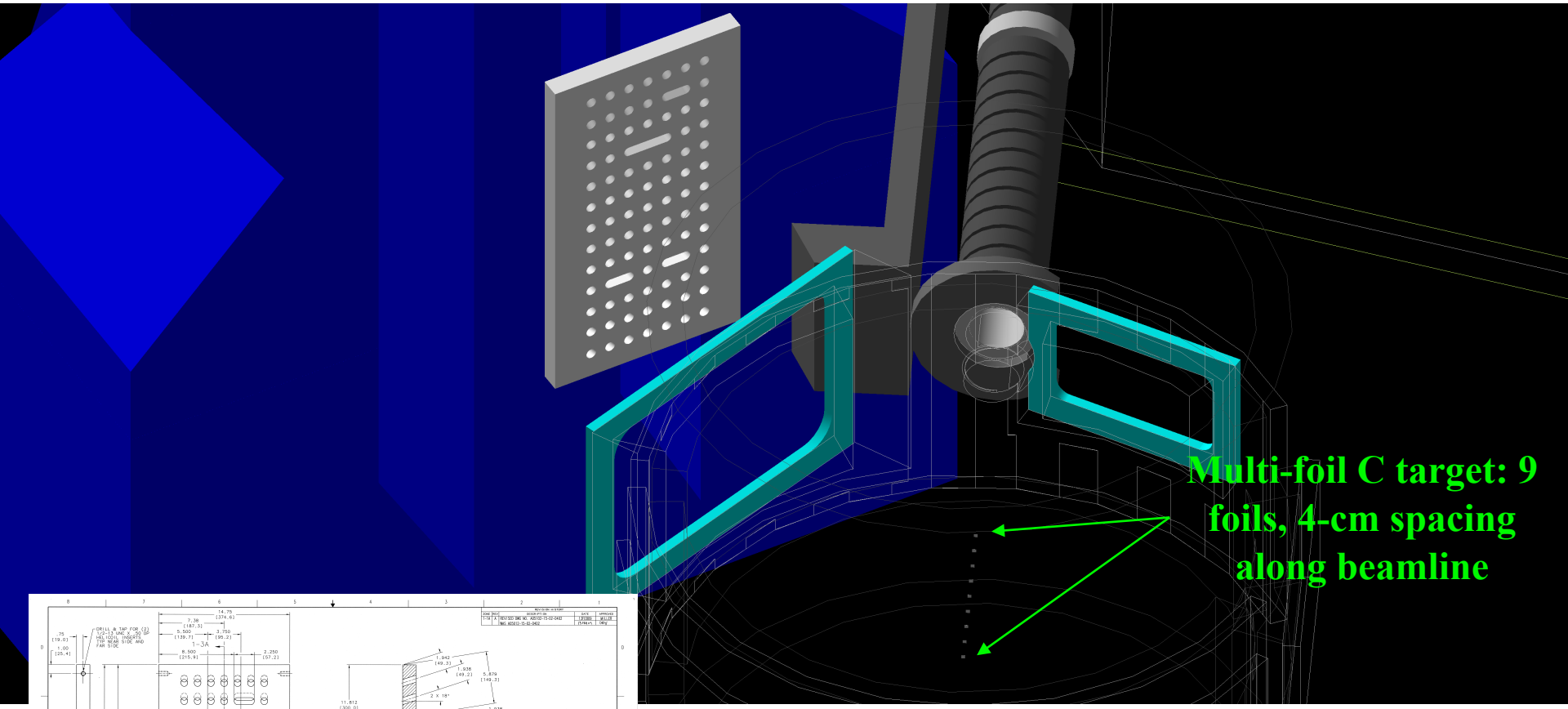


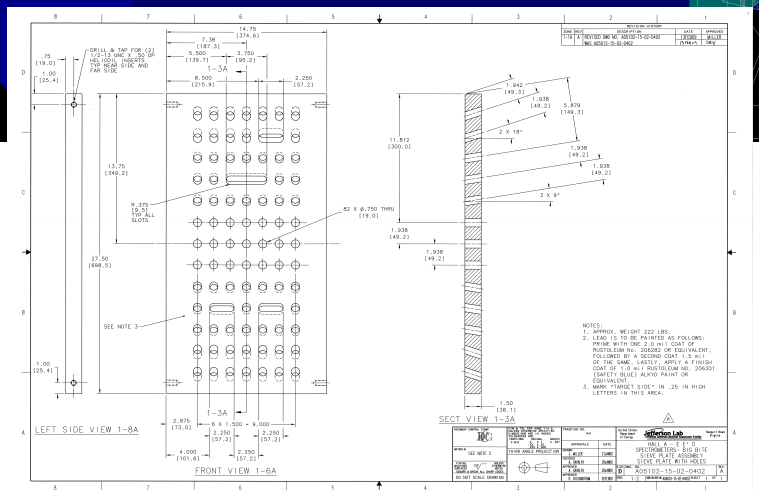




Recent simulation-based studies and geometry updates

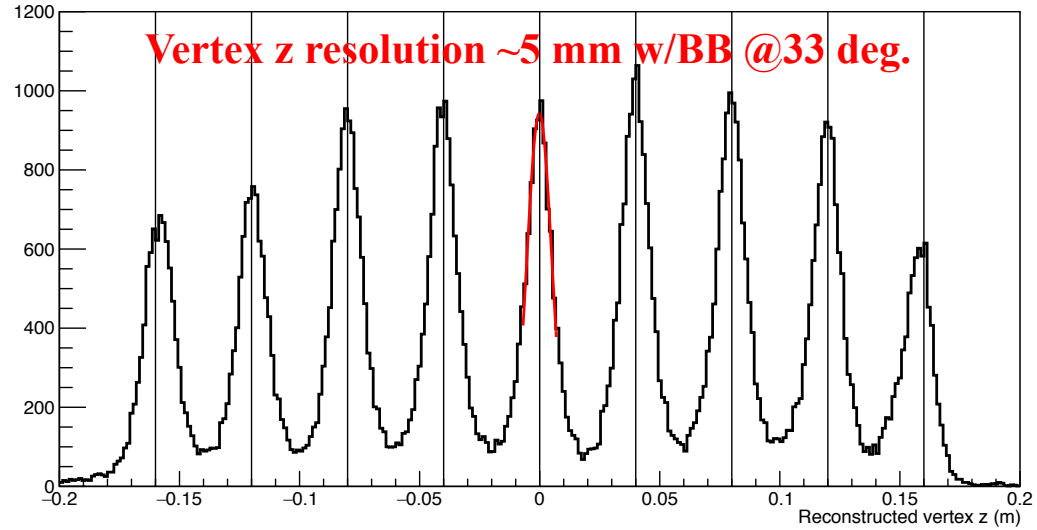
BigBite Optics Planning: Sieve Slit/multi-foil simulation



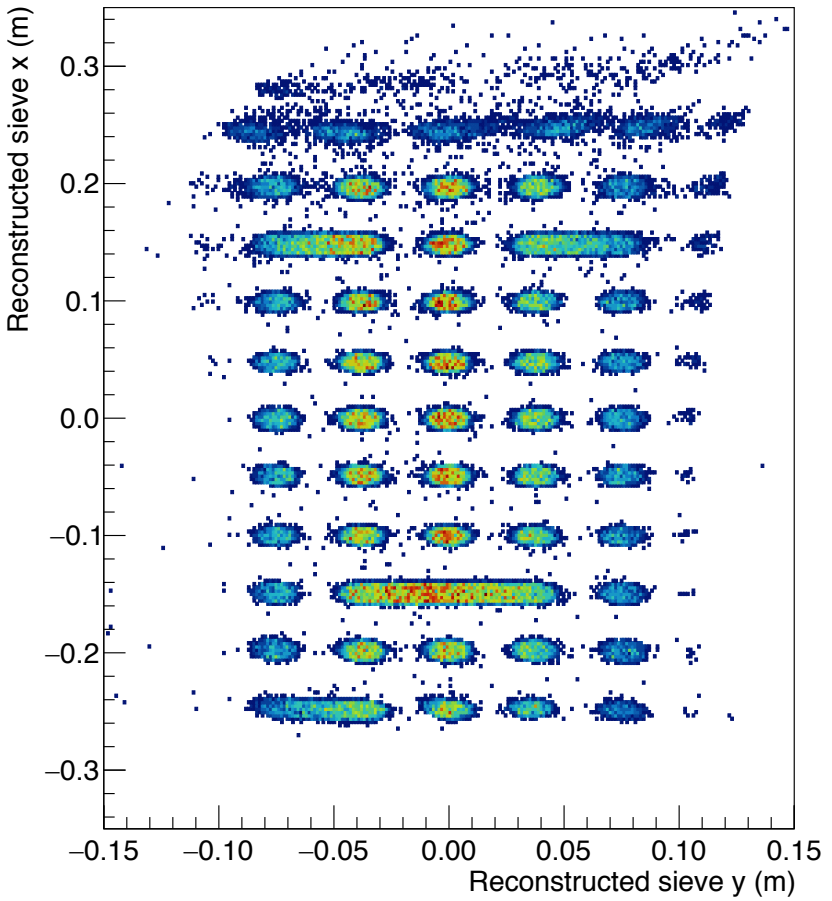


- BigBite Sieve plate design from C. Soova implemented in g4sbs.
- Multi-foil Carbon target for optics calibration added according to GMN run plan
- Test adequacy of sieve plate material, thickness, hole pattern/size/spacing/etc.

BigBite Optics Planning: Vertex and sieve pattern reconstruction



- Current GMN plan calls for optics target with 9 C foils, spaced at 4-cm intervals along z
- **With sieve plate at its nominal position, the two outermost columns and the top and bottom rows of holes seem to be mostly absent from the acceptance.**
- **The performance of the starting (4th-order) optics model is poor near the vertical extremes of the acceptance, particularly at large +X (the bottom)**
- **Basic cuts on track quality, and the preshower+shower E/p ratio after track momentum reconstruction are sufficient to suppress "punch-through" tracks; the reconstructed sieve hole pattern is relatively "clean"**
- ***Next step: implement and test realistic optics calibration, where MC truth info on vertex/track angles is replaced by rays computed from foil and sieve hole positions.***



- Note: both vertical and horizontal axes of TRANSPORT coordinate system are inverted relative to the image on previous slide: +X = vertically down, +Y = left (increasing scattering angle)

BigBite Sieve Plate—Preliminary conclusions

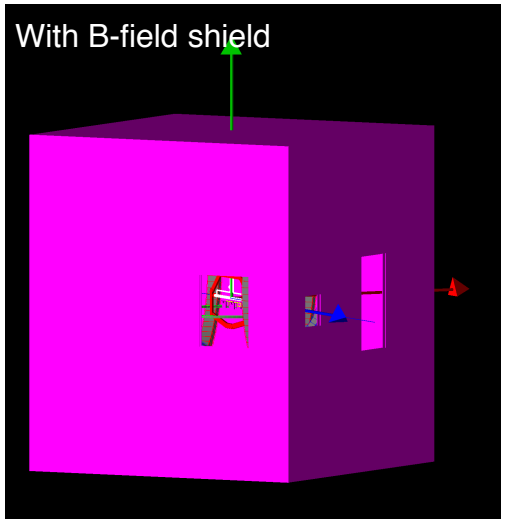
- Given that only 5 of the 7 columns of holes are comfortably within the BigBite horizontal angular acceptance, is this sufficient to calibrate in-plane angle reconstruction?
- At the nominal target-magnet distance for GMN @highest Q^2, each BigBite sieve hole diameter subtends an angle of about 16 (8) mrad max (avg) in both vertical and horizontal directions—this compares to the nominal angular resolutions of about 1 mrad (vertical) and 2 mrad (horizontal)—naively this would seem to make the holes too big—more typical design for Hall A/C spectrometers is hole size ~1-2X angular resolution, and hole spacing ~5-10X angular resolution.
- Hole spacing is ~32 mrad (horizontal) and ~42 mrad (vertical), or about 16-40X angular resolution.
- Solid-angle acceptance is ~0.2 msr per hole, ~19 msr total (not counting extra acceptance of "slots")
- Do we really need the "slots"—what purpose do they serve other than as a sanity check? Could we accomplish the same thing by plugging one of the holes and/or using a smaller diameter on one or several holes?
- Is 4-cm spacing between optics target foils sufficient given the vertex resolution of BB? →Probably, but 5 cm might be safer
- Tentative conclusion, pending test of realistic calibration procedure, is that **existing sieve plate is adequate, but not necessarily optimal**—better would be holes roughly ~1.5-2X smaller, with a 1.5-2X denser grid in both vertical and (especially) horizontal directions
- Thickness (1.5-inch) and material (lead) are probably fine

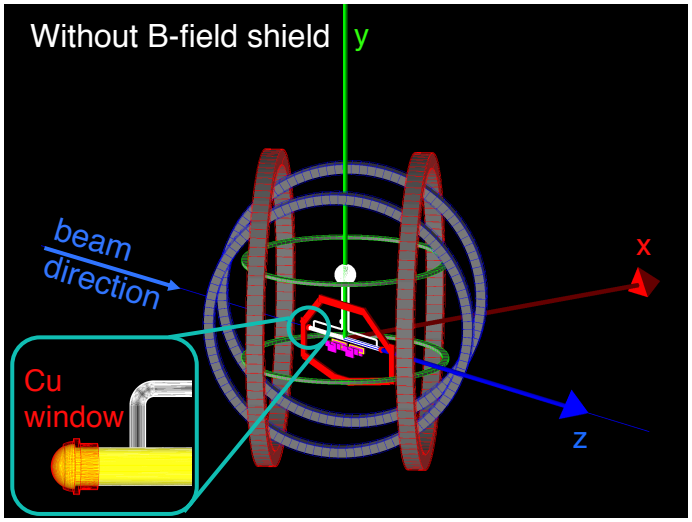
^{3}He target geometry in GEANT4: Slide Credit: David Flay (JLab)

Major push underway to fill in missing geometry for GEN/SIDIS experiments in *g4sbs*, anticipating fall 2020 ERR for GEN

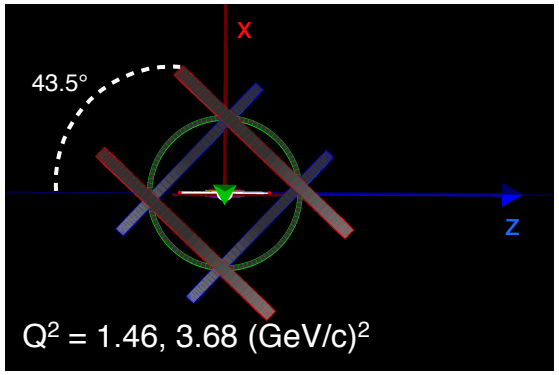
GEn ^{3}He Target in g4sbs

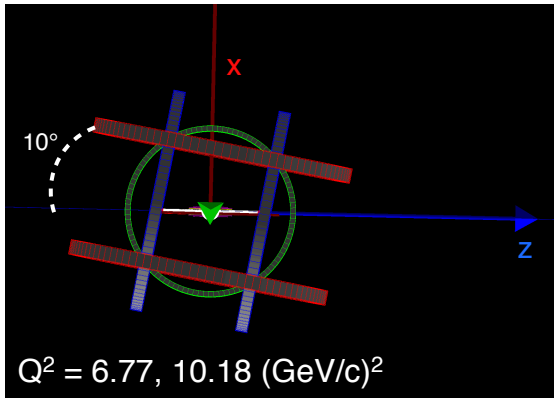
- Major components of ^{3}He target geometry implemented in g4sbs
- Starting to implement hit and sensitive detector capabilities for energy-loss tracking



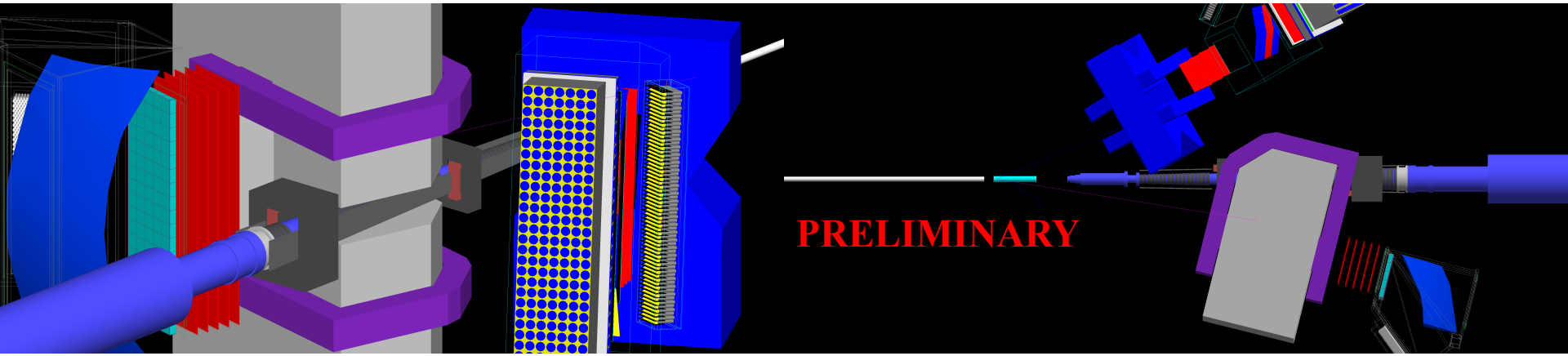


Helmholtz Coil Orientations

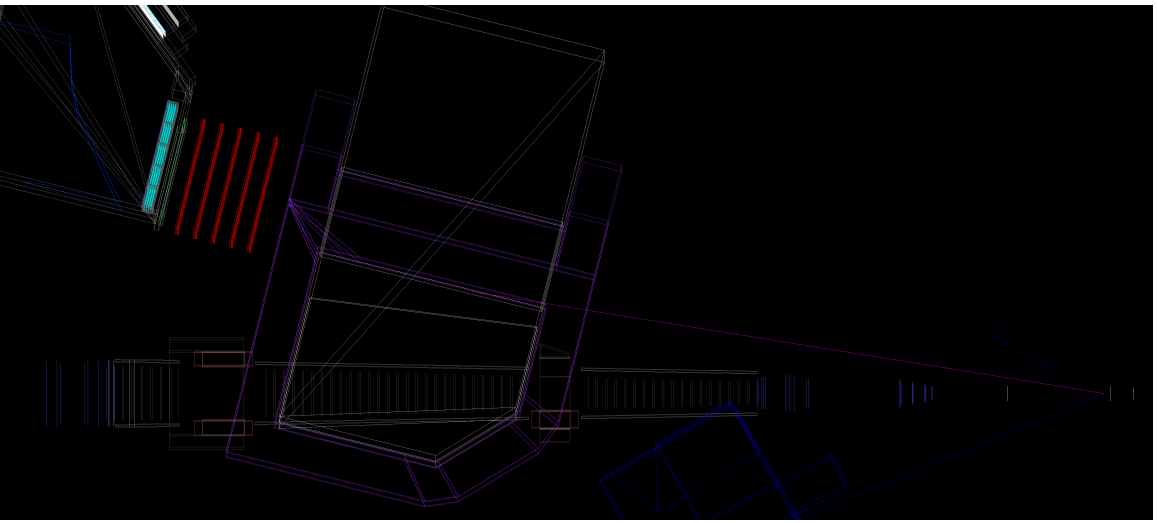




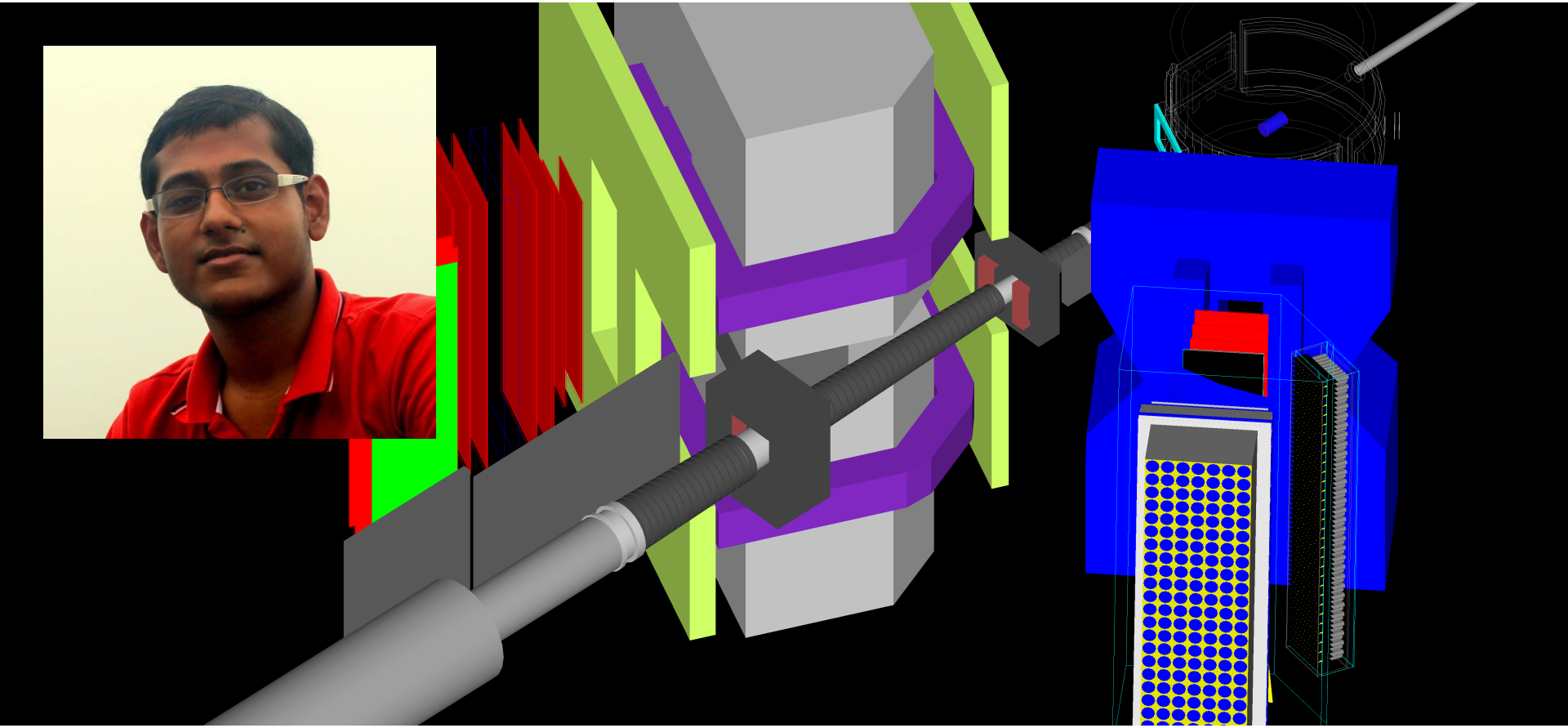
GEN/SIDIS beamline and shielding geometry



- Design by Alan Gavalya, **implemented in *g4sbs* by UConn Ph.D. student Sebastian Seeds**
- David Flay's Helium-3 target geometry still to be merged in
- Still checking geometry for overlaps/interferences, checking all dimensions with STEP file from Alan

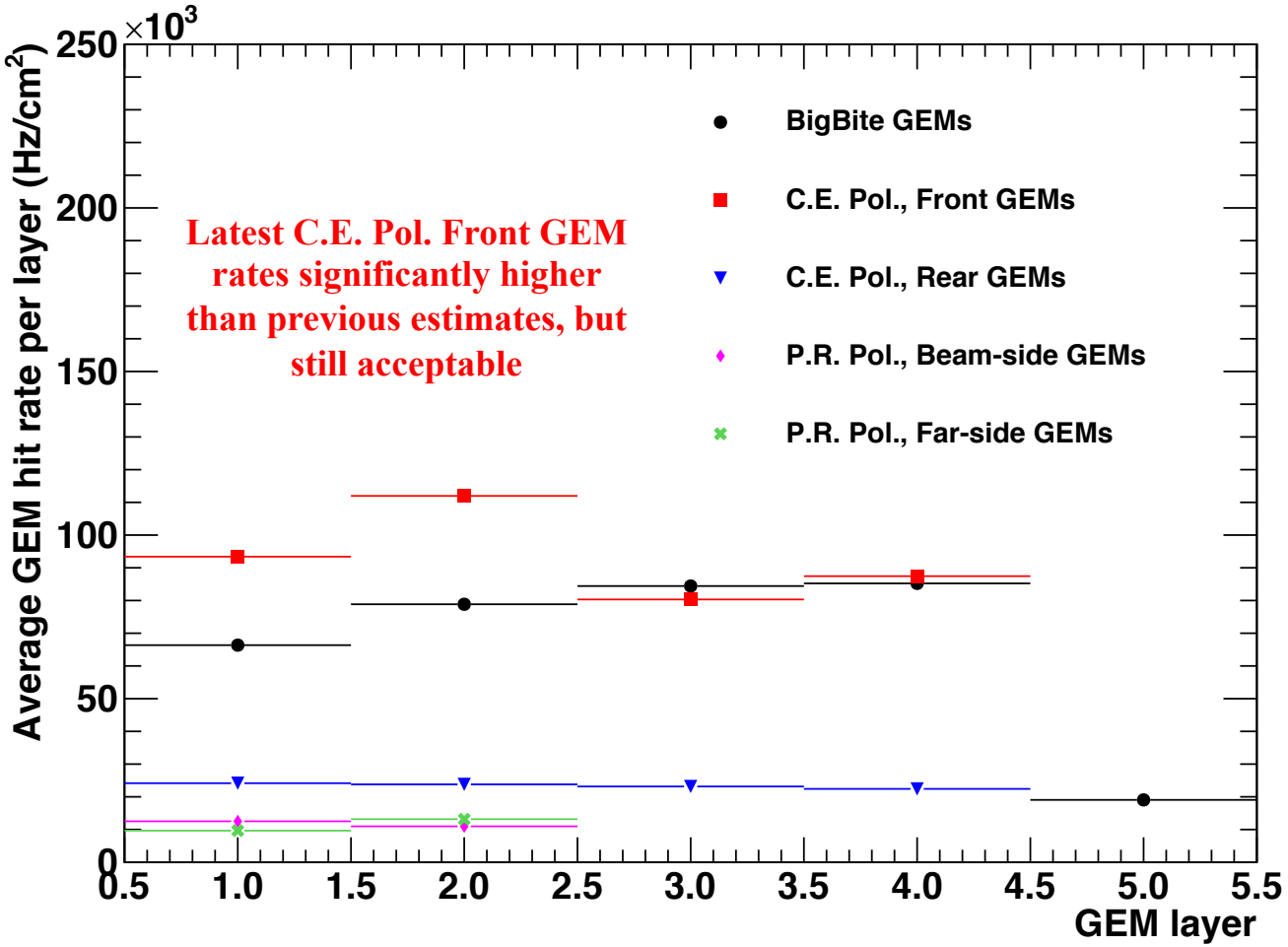


New GEN-RP simulations (Credit: Provakar Datta)



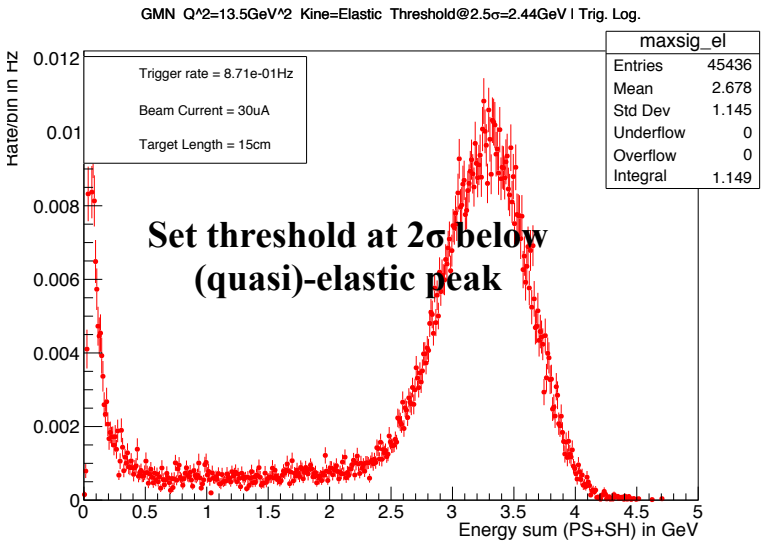
- g4sbs GEN-RP geometry now harmonized with actual plans:
  - GMN scattering chamber and downstream beamline, lead shielding on BB **and SBS side**; BB field map and uniform SBS field 1.4 T (for BdL = 1.7 T*m)
  - GEN-RP polarimeter detectors: front and rear C.E. GEMs, and P.R. GEMs on the sides. Active analyzer and scintillators.
  - **Steel analyzer; 3.5” (8.9 cm) thick**

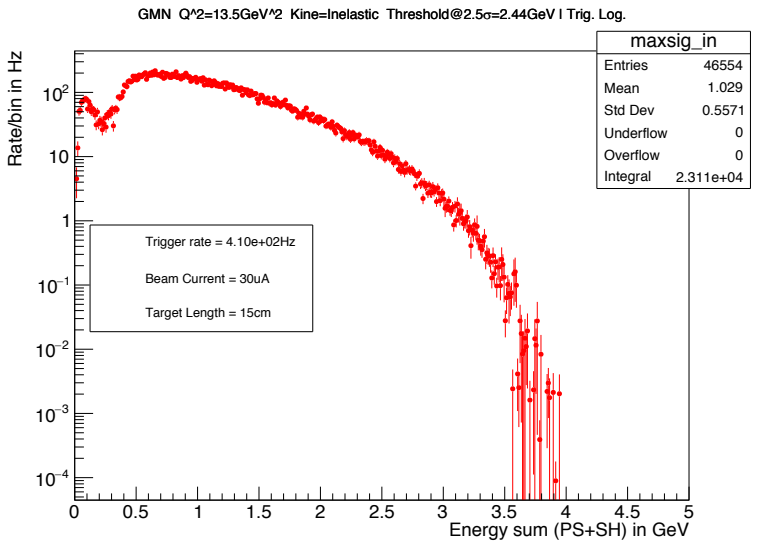
GEN-RP GEM hit rates per unit area by layer:

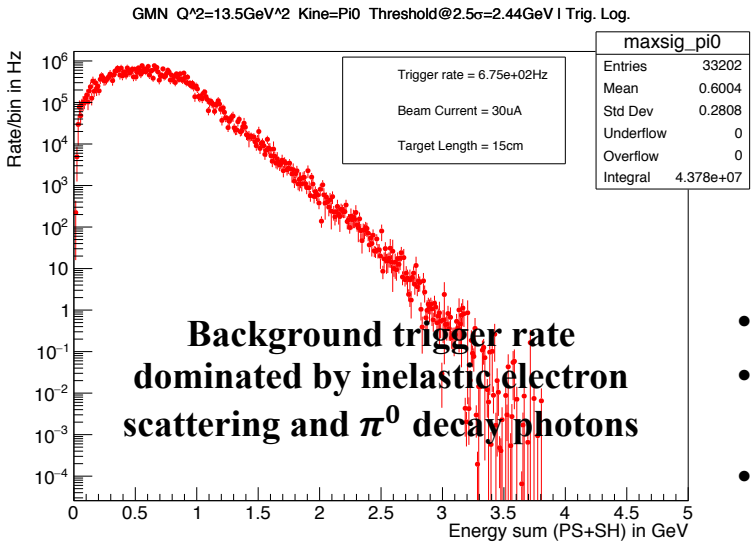


- Caveats: the reduction in average GEM rate in layers 3 and 4 of the front GEMs is an artifact of dividing by the larger total active area of these GEMs; because the field clamp cuts off part of the acceptance, the local rates are actually higher and comparable to those in layers 1 and 2

GMN/GEN-RP Trigger Rate Estimates (Provakar Datta)







Q^2 (GeV^2)	E_e (GeV)	θ_{BB}/d_{BB} (deg)/(m)	E'_e (GeV)	Thrsh. (GeV)	Eff. %	Trig. Rate. (kHz) El.	Inel.	π^+	π^0	π^-	Total (kHz)
3.5	4.4	32.5/1.80	2.18	1.81	96.9	0.8	5.3	0.04	3.5	0.1	9.7
4.5	4.4	41.9/1.55	1.70	1.35	94.0	0.2	2.3	0.2	7.3	0.2	10.2
5.7	4.4	58.4/1.55	1.14	0.89	94.8	0.02	0.6	0.4	12.9	0.6	14.5
6.1	6.6	30.3/1.55	2.90	2.38	95.9	0.08	1.7	0.006	1.0	0.02	2.8
8.1	6.6	43.0/1.55	1.94	1.56	95.5	0.007	0.4	0.06	1.6	0.03	2.1
10.2	8.8	34.0/1.75	2.92	2.37	96.8	0.003	0.3	0.003	0.3	0.002	0.6
12	8.8	44.2/1.55	2.05	1.62	95.4	0.0008	0.1	0.005	1.2	0.02	1.3
13.5	11	33.0/1.55	3.30	2.60	94.0	0.0008	0.2	0.003	0.2	0.002	0.4

- Credit: UConn Ph.D. student Provakar Datta
- BigBite rate estimates include inelastic electrons, inclusive single pion production
- Report: https://puckett.physics.uconn.edu/wp-content/uploads/sites/1958/2020/07/trig_rate_gmn.pdf

Issues/problems/delays/status/timelines

- With a few exceptions (BB timing hodo/HCAL), we have not received input on beam calibration/commissioning plans/procedures from detector groups (see my slides requesting such documents from August 2019 collaboration meeting). I am guilty of this as well
- Simulation digitization currently suffers from issues including:
  - Speed bottlenecks
  - Poor usability/lack of user documentation
  - Unnecessary complexity
- These issues have delayed development and testing of SBS-offline, integration of various standalone analysis codes into Podd
- We believe we are still on track, with prompt corrective action, to demonstrate GMN software readiness with simulation by Sept. 2020
- We are closely coordinating with GEM group to integrate decoding of raw data and clustering/tracking code into SBS-offline—Estimated completion: Sept. 2020
- We are getting increasing involvement from new and existing collaborators, and weekly software meetings are productive.
- **Current software working group weekly meeting time: Fridays, 1 PM**
- **SBS Software Mailing List: https://mailman.jlab.org/mailman/listinfo/Sbs_software**