Data-based simulation of cosmic muons (not only) for calibration

Technical Note

Robert Kralik

July 4, 2023

Abstract

In NOvA we employ samples of stopping and through-going cosmic muons to calibrate our detectors. ITo validate the efficacy of this calibration procedure for both the simulated and real detectors, as well as to determine systematic uncertainties arising from calibration, we require a sample of simulated cosmic muons.

Originally we used (or still use) the CRY cosmic-ray generator to create a large Monte-Carlo (MC) sample. This sample was passed throught the Geant4-based simulation of the detector and underwent the same cosmic muon selection process as the actual data. However, the CRY simulation proved to be highly inefficient, with only a small fraction of the simulated cosmic-ray activity resulting in selected calibration hits, and the majority of particles failing to even hit the detector. This inefficiency consumes significant CPU resources, disk space, and file usage. Moreover, the momentum and angle distributions in CRY are not well suited to the NOvA sites, potentially impacting the calibration accuracy.

To overcome these challenges, we have implemented a data-based simulation method that eliminates the need for the CRY Monte Carlo generator. Instead, the detector activity artdaq-stage data files are passed through the beam removal filter, reconstruction chain, selection of high-quality cosmic muons. Subsequently, charge assignment and smearing are applied before feeding the data into the `"Text File Generator"` simulation.

This approach exhibits near-perfect efficiency, ensuring that almost every simulated muon contributes to the final calibration sample, thus saving processing time, file size, and storage. Additionally, the simulated muon distributions are inherently consistent with the data. Given that the calibration chain itself is a time and CPU intensive process itself, he reduction in simulation files and their sizes significantly benefits downstream of the file generation.

Contents

1 Introduction

The data-based simulation of cosmic muons was initially developed by Teresa Lackey in 2021 for Test Beam detector calibration. Teresa based the reconstruction and selection of data events on the `CerenkovSelection` module from light level tuning and created a simple Python script for event smearing and muon charge assignment. However, when tested by Robert Kralik in 2021 and 2022, the simulation exhibited notable discrepancies when compared to the Test Beam data from period 2.

To address these disparities, Robert made improvements to the simulation throughout 2022, ultimately completing it in 2023. The enhancements involved modifications to event selection, charge assignment and energy correction of the through-going muons. The revised simulation was then employed and tested during the calibration of the Test Beam detector in 2023.

This technical note provides an explanation of the process to create simulation samples for calibration, with a primary focus on its application in Test Beam. However, the approach and underlying code have been designed to be easily adaptable for use with the Near and Far detectors, as well as to generate simulated samples of cosmic muons for purposes other than calibration.".

2 How does it work?

All the code required to generate data-based simulations of cosmic muons is located within the novasoft `CosmicStudies` package.

The process of generating a new data-based simulation begins with an artdaq-stage data sample containing information on Raw Digits hits. These events undergo a series of filtering, reconstruction, and selection steps to obtain the vertex positions and 4-momenta of cosmic muons. The `cosmicgenanajob` ART job generates a ROOT file with a TTree containing all the necessary information, as outlined in section 2.1 of this document.

Subsequently, the reconstructed information for each event is processed by a Python script `GenerateHEPEVTFromROOT.py`. This script corrects the 4-momenta of the through-going muons to account for the missing energy that was not deposited in the detector. Furthemore, it assigns a charge to each cosmic muon based on a statistical distribution, smears the kinematic information to reduce bias from the input data and prints the `HEPEVT`-styled description of each event into a text file. The details of this process are elaborated in section 2.2.

The text file is then passed to the `Text File Generator`, which employs the data information as seeds for the Geant4-based detector simulation. By incorporating additional reconstruction steps, an artdaq-stage simulation sample of cosmic muons is generated. To create calibration samples for stopping and through-going cosmic muons, we apply the same reconstruction and selection criteria used for any data sample. Further details are provided in section 2.3.

A comprehensive outline of all the necessary steps is presented in the box below and can also be found on the DataDriven Cosmics Generation redmine wiki page.

1. Run `cosmicgenanajob.fcl` on detector activity artdaq-stage files;
2. Hadd the outputs and run `GenerateHEPEVTFromROOT.py haddedOutput.root HEPEVTFileName.txt`;
3. Split the resulting txt file into files each containing a subset of lines (events);
4. Generate FHiCL files each sourcing a different text file
   `CreateFclsForSimulation.sh TextFileGenjob_template.fcl inDir outDir`;
5. Make a SAM definition from the FHiCL files with `sam_add_dataset`;
6. Include all the text files into `TextFileGen.cfg` with `-inputfile`;
7. Submit `TextFileGen.cfg`;
8. Move the newly created simulation artdaq files to a persistent area, declare them to SAM and create a definition
9. Generate the calibration files (pclist/pcliststop);
10. Move the results into a persistent location, declare to SAM and make definitions.

2.1 Reconstruction and selection of cosmic muon events from data

Our goal is to examine the response of the simulated detector to realistic cosmic muons found in real-life data. If the selection of reconstructed data does not accurately correspond to reality, either due to misreconstruction or incorrect selection criteria, it can introduce bias into our simulation. Additionally, we want the resulting simulation to primarily consist of events that will be included in the final simulation calibration sample and can be effectively used in the calibration process. Therefore, it is crucial to employ a similar reconstruction approach to that used for the data and establish selection criteria that generate well-understood distributions resembling those of the data calibration samples.

The initial step is to choose an artdaq-stage data sample that represents the detector in a fault-free state. Pre-staging this data can be a time-consuming aspect of the data-based simulation process, so it is advisable to select a data sample that is either already pre-staged or can be easily pre-staged. Section 2.2.4 provides insights into estimating the required data volume.

For Test Beam, we have opted for the full period 4 data sample, as other periods had issues such as faulty FEBs, underfilled cells, or similar complications. In the initial version of the data-based simulation, Teresa utilized the period 2 Test Beam data, as it was the only complete Test Beam data sample available. However, this data included the aforementioned effects, which could have a non-trivial impact on the simulation.

Once the data sample has been selected, we execute a single ART job located at `CosmicStudies/cosmicgenanajob.fcl` to filter, reconstruct, and select the desired events, which are then written to a ROOT TTree.

2.1.1 Remove Beam Spills

The first step is to remove beam spill events using the `RemoveBeamSpills.fcl` job for the Near and Far Detectors, or `RemoveTBSpills.fcl` for the Test Beam detector, which should leave us with mostly cosmic events.

2.1.2 Reconstruction

The `Text File Generator` requires the vertex position and the initial 4-momentum for each event. For that we use:

1. `CalHit` to create Calibrated Cell Hits from Raw Digits,
2. `Slicer` to group hits into slices,
3. `Window CosmicTrack` - a tracking algorithm for cosmic particles,
4. `CosmicRayVertex` to identify vertices for cosmic particles,
5. `FuzzyKVertex` to cluster hits into prongs and
6. `BreakPointFitter (BPF)` to identify muons (or muon-like tracks) and get their initial 4-momenta by fitting to the FuzzyK prongs and CosmicRayVertex vertices.

The first three steps are identical to the full reconstruction applied to get the calibration samples. Since we do not need a 4-momentum information for calibration, we do not need to use the Break Point Fitter.

2.1.3 Selection

After the reconstruction process, we proceed to select events based on their slice's and **BreakPointFitter (BPF) tracks'** properties, saving them in a Root TTree. This step is done using the `CosmicStudies/CosmicGenAna` module.

To select an event, the following conditions must be met. Table 1 provides an overview of all the cuts and their corresponding values.

1. We only use successfully reconstructed 3D tracks with the muon assumption from the BreakPointFitter.

2. As we aim to select cosmic events originating outside the detector, we apply a cut based on the distance of each track's start from the Top/Front/Back/Sides of the detector. This cut has a negligible impact on the BPF tracks, as indicated by the minimal difference between the red and azure lines in Figure 1;

3. We remove all events whose tracks are parallel to the beam direction. Figure 1 demonstrates the presence of events peaked at track lengths of approximately 410cm and 200 cm, which correspond to the total and half length (length of one module) of the detector, respectively. These events are strictly parallel to the beam direction ($\text{Cos}_Z \sim 1$) and are likely remnants of beam events. Applying a cut on the CosZ of the initial track direction effectively removes these events without affecting the rest of the data.

4. To ensure that only events contributing to the final calibration sample are simulated, we employ a selection based on cuts from the Calibration/PCHitsList module. The full "calibration sample" selection is applied to both data and simulation artdaq-stage samples to generate the `pclist` and `pcliststop` calibration samples. However, there are two caveats we need to consider.

   First, for the calibration sample generation process, we utilize tracks from the Window cosmic track algorithm instead of the Break Point Fitter algorithm, which yield different distributions as depicted in Figure 2. Applying the full calibration sample selection on BPF tracks could mistakenly remove events that would pass the same selection when applied to the window cosmic tracks.

   Second, each reconstruction algorithm has intrinsic deficiencies that can lead to misreconstructions. Applying the full calibration sample selection may remove misreconstructed events that should have been included in the simulation, introducing a selection efficiency bias.

   To address these concerns, we have loosened the original calibration sample selection cuts to create a "buffer" around the selected events, allowing for fluctuations of the reconstruction

algorithms while maintaining track quality. The differences between the full calibration sample selection and the employed loosened selection are listed in table 1 and shown on Figure 2. Additionally, we present a comparison to the data calibration sample, which was created by applying the full PCHitsList selection on window cosmic tracks from the same artdaq data sample.

	Cut	Full selection	Loose selection
	Muon assumption and 3D track from BPF		
	Max. track start distance from edge	50 cm	
	Max. Cos_Z	0.98	
Calibration sample selection	Max. number of hits in X or Y	2	
	Min. difference between $Stop_Z$ and $Start_Z$	70 cm	50 cm
	Min. Cos_Z	0.2	0.15
	Min. frac. of slice hits in track in each view	0.8	
	Max. number of cells per plane in each view	6	15
	Max. difference in X-Y for first (last) plane	3	5
	Max. plane asymmetry	0.1	0.2
	Max. step size to median step size ratio	3	5
	Max. vertex distance from edge	10 cm	
	Max. track end distance from edge	10 cm	

Table 1: Event selection for the data-based simulation (in green under Loose selection) and comparison to the Full calibration sample selection cuts. The last two rows are not used for Test Beam, but are employed for the Near and Far detectors and should be studied before creating a data-based simulation for them.

During the selection process, we determine whether the muon is stopping inside the detector or passing through, based on the reconstructed track's end position[1]. This information assists in correcting the energy of through-going muons to account for "lost energy," as outlined in section 2.2.

In the initial iteration of the simulation, Teresa implemented a cut requiring the total track length to be at least 300 cm. This was motivated by the Cerenkov selection for light level tuning but contributed significantly to the discrepancy between data and simulation in the calibration samples. To address this issue, Robert removed this cut and, to ensure the quality of selected events remained high, applied the loosened calibration sample selection cuts along with the Cos_Z cut.

[1]For Test Beam we say it is a stopping muon if its track ends at least 20cm from any edge of the detector. For the far and near detector this is 50cm. This value was chosen by Kevin Mulder [1] as 50cm removed too many cosmic events from the Test Beam detector.

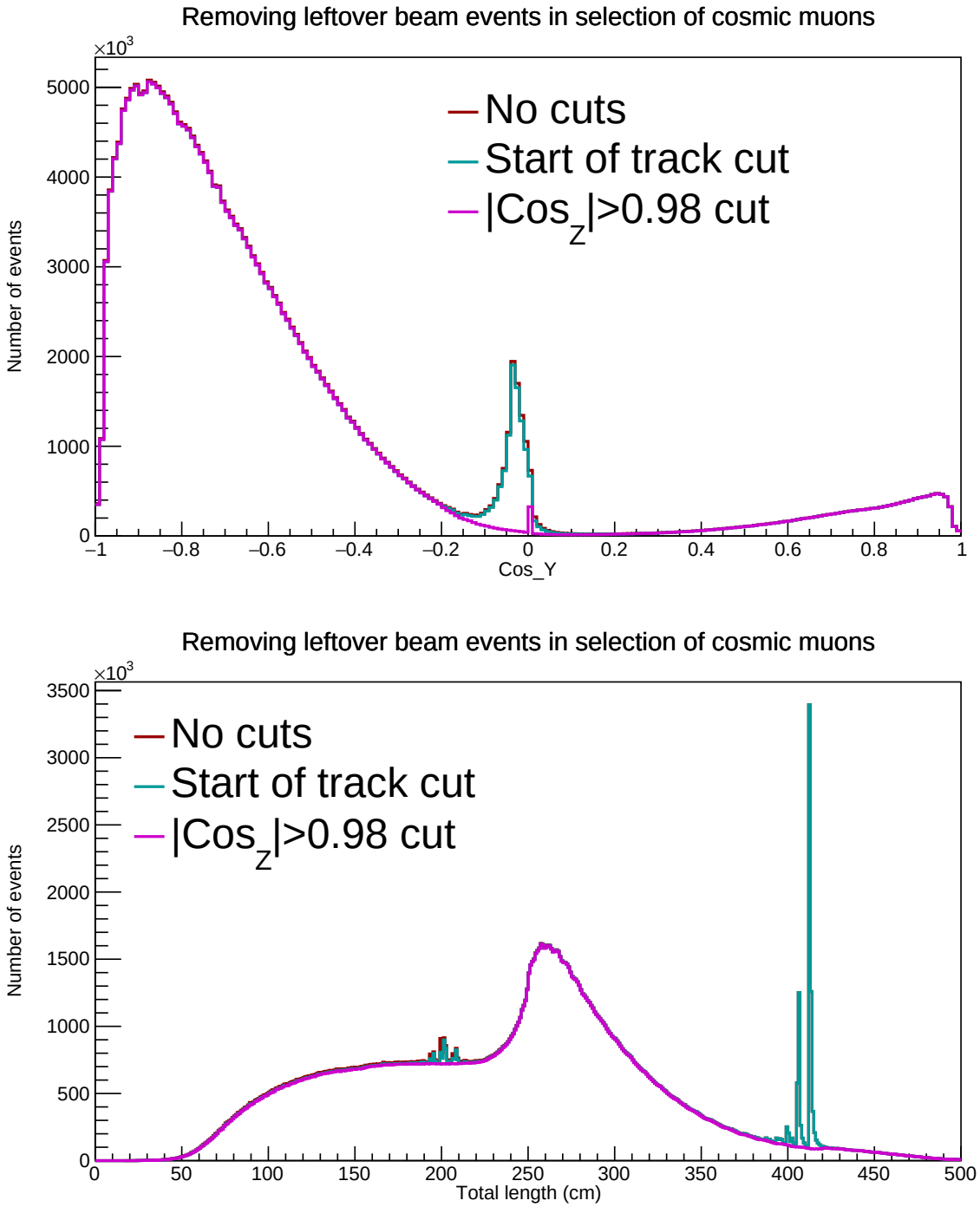

Figure 1: Comparison of event selections for the data-based simulation and of the corresponding data calibration sample. All of the distributions are made from the period 4 Test Beam data. The green line depicts the final selection of events used for the data-based simulation, while the blue line represents the selection with full PCHitsList cuts applied.

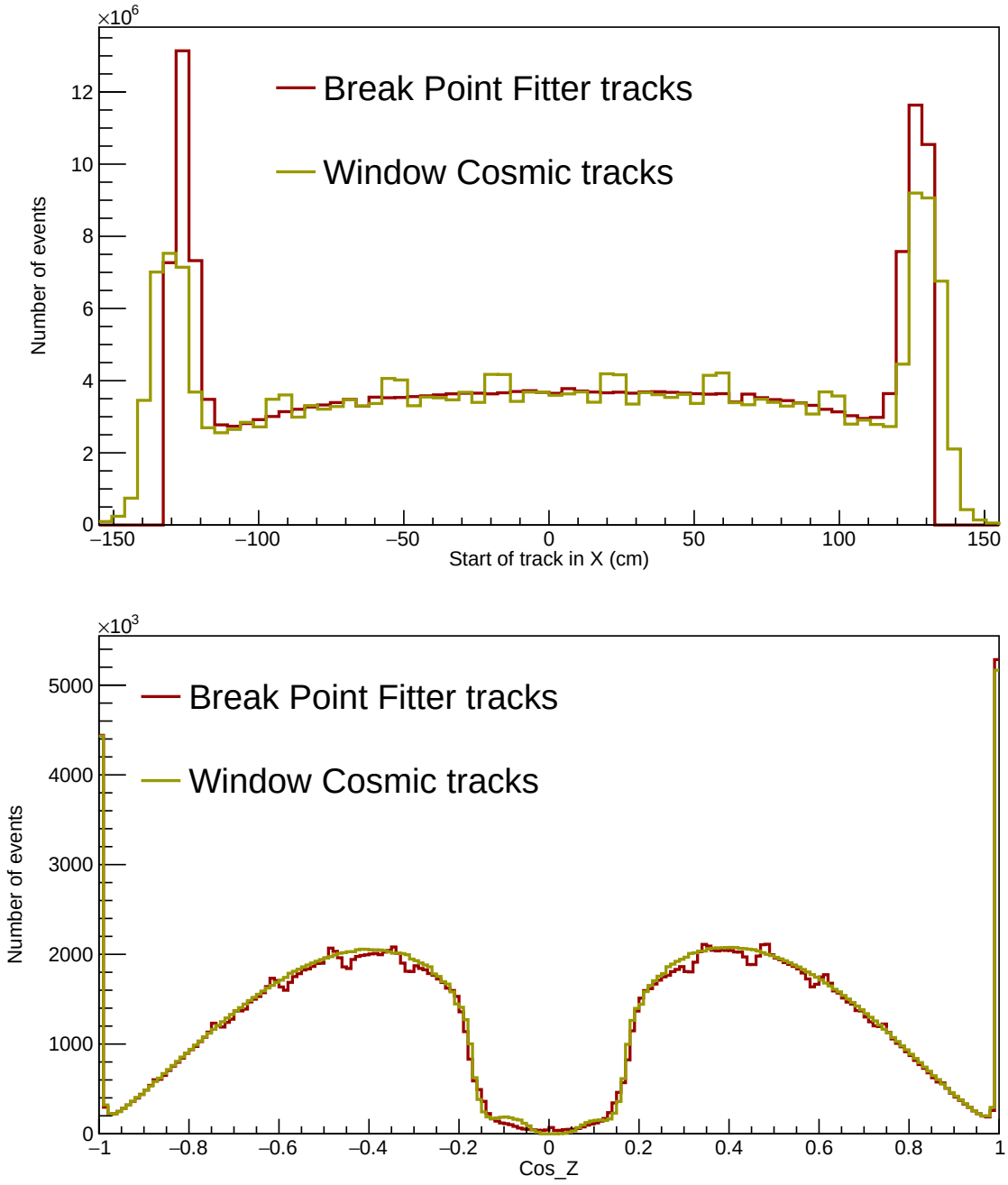

Figure 2: Difference between the tracks reconstructed with the break point fitter (BPF - used for data-bases simulation) and with the window cosmictrack (WT - used for generating the calibration samples) algorithms. Both distributions are for the period 4 Test Beam data (with removed beam spill) without applying any selection. The BPF tracks have a hard cut-off at the detector edges, whereas the WT tracks are allowed to start beyond these limits. Also, the BPF tracks have a ragged distribution in Cos_Z, which is not present for WT tracks. It is not know where does this raggedness come from.

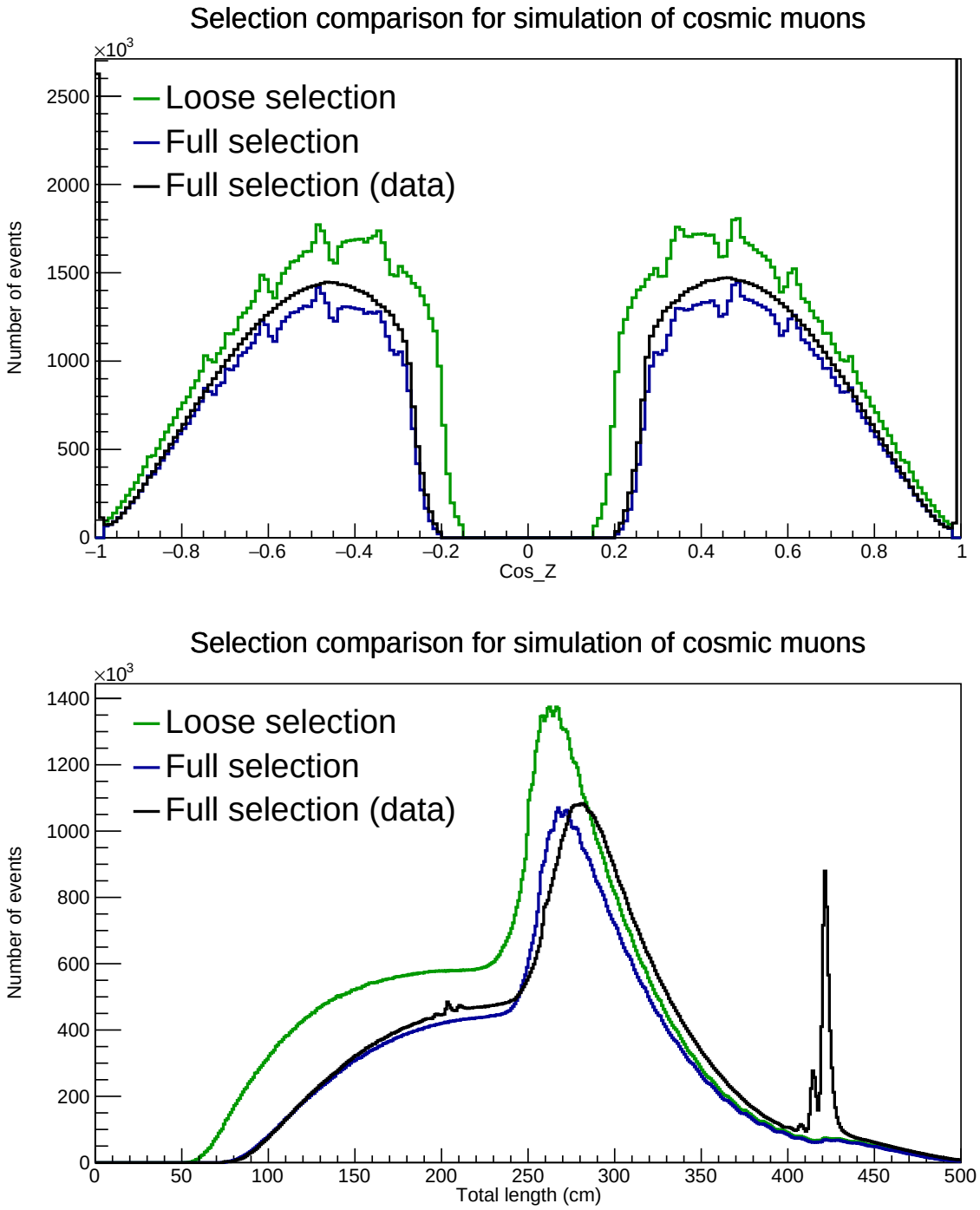

Figure 3: Comparison of the final selection...

2.2 Energy correction, charge assignment and smearing

Once we have the TTree with kinematic information for the selected events we run a python script `CosmicStudies/GenerateHEPEVTFromROOT.py`. This script performs several tasks, including correcting energies of the through-going muons, assigning a charge to each muon, applies smearing and converting the information into a `HEPEVT` format, which will be used in the `Text File Generator`.

To load the TTree ntuples into a dictionary of numpy arrays, the script utilizes the `uproot` library. It is important to note that the machine being used may not have `uproot` installed. If necessary, the command `pip install -user uproot` can be executed to install the library.

During a detector systematics planning session in Summer 2021, Mark Messier and Teresa Lackey presented an overview and a strategy for data-based simulation of cosmic muons for calibration [2]. They discussed potential improvements to the energy estimation of through-going muons, charge assignment based on energy distribution and a plan to implement data-based simulation in the Near and Far detectors.

To run the python script do:

```
python generate_hepevt_cosmic.py inFile.root outFile.txt\
                                  --niter NIterations
```

2.2.1 Energy correction

Muons that do not stop inside the detector carry away the energy not deposited in the detector and their true initial energy is therefore larger than the reconstructed energy (E_R) we get from the Break Point Fitter. In general, the energy spectrum of cosmic muons can be approximately described by a power law $E^{-\alpha}$, with $\alpha \approx 2.7$ [2, 3]. The expectation value for the "true" energy of the through-going muons can be calculated as

$$\langle E \rangle = \frac{\int_{E_R}^{E_C} E \cdot E^{-\alpha}}{\int_{E_R}^{E_C} E^{-\alpha}} = \left(\frac{\alpha-1}{\alpha-2}\right)\left(\frac{E_C^{2-\alpha} - E_R^{2-\alpha}}{E_C^{1-\alpha} - E_R^{1-\alpha}}\right) \tag{1}$$

where E_C is the critical energy chosen to be 300GeV, as we do not expect muons with higher energies to be selected due to large showers along their paths. Plot 4 shows the corrected energy distribution of our selected events and demonstrates that the choice of the critical energy does not significantly change the correction of the true energy.

We could also include angular dependence for the energy correction [3], but this would only have a negligible effect due to the small energies in our detectors.

2.2.2 Smearing

Smear the events to avoid any bias from the data. Smearing is done by randomly changing the total momentum within 2%, azimuthal angle uniformly, polar angle within 4mrad and the X/Y and Z vertex positions within the width or depth of the cell respectively.

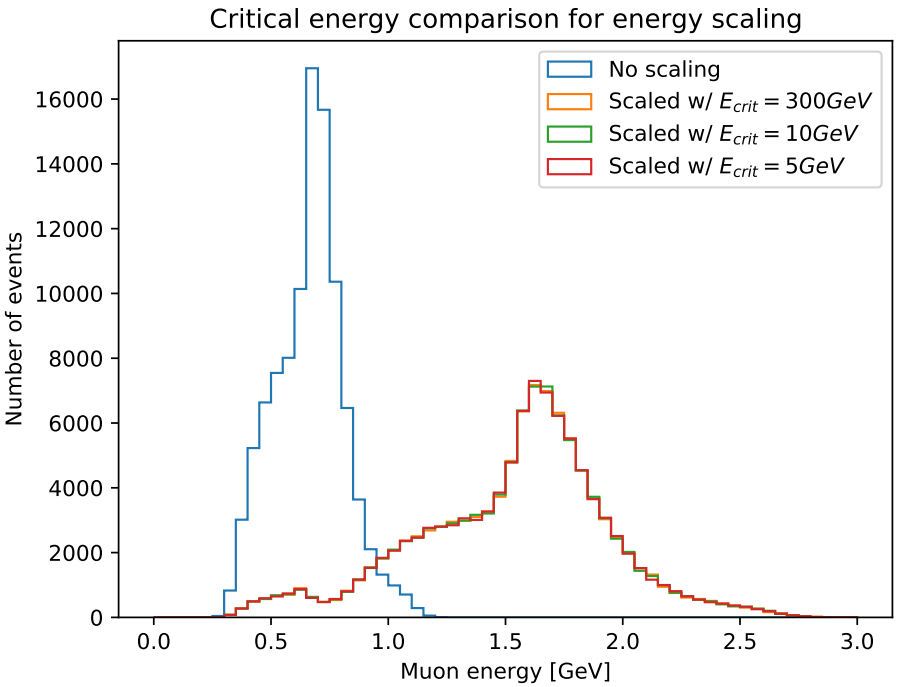

Figure 4: The effect of energy correction for through-going muons with various critical energies. No significant difference can be seen when using different critical energies.

2.2.3 Charge assignment

Text File Generator requires the charge of each muon event. Since we reconstruct muons and antimuons in the same way, we have to assign the charge based on external measurments.

Figure out where are the plots and the equation Mark/Teresa quoted from (somewhere in PDG). Describe the basis of the measurement. Here(p10): https://pdg.lbl.gov/2022/reviews/rpp2022-rev-cosmic-rays.pdf But the equation must be from one of the sources listed.

$$P_+ \simeq 0.539 + \frac{x}{34.5} - \left(\frac{x}{9.48}\right)^2 + \left(\frac{x}{8.27}\right)^3 \tag{2}$$

2.2.4 Number of events to simulate

The python script offers options to adjust the statistics of the generated simulation. We can specify the number of iterations (niter) or choose to skip events (stride). It is crucial to strike a balance between simulating an adequate number of events for thorough calibration and avoiding unnecessary computational and memory requirements.

To ensure a reliable calibration, we aim to have a sufficient number of through-going muons in each cell, view, and fiber brightness bin, with a target $\chi^2 < 0.2$.

Based on observations, approximately 50,000 tricell hits per cell, view, and brightness bin provide reliable results. This figure remains the same for the Near and Far detectors, since the binning for position inside a cell is identical for all detectors (100 bins per cell). However, we must consider the discrepancy between horizontal and vertical cells. Vertical cells, parallel to cosmic muons, receive approximately 5 times fewer hits than horizontal cells. To adequately populate the vertical cells, we multiply the target number (50,000) by 6 (1 horizontal + 5 vertical views), then by the number of fiber brightness bins (12). Finally, we multiply this by the number of cells: 390 for FD, 100 for ND, and 64 for TB. Thus, the minimum total tricell hits required is calculated as:

Detector	$50,000 \times 6 \times 12 \times$NCells
Test Beam	230,400,000
Near Detector	360,000,000
Far Detector	1,404,000,000

Since we are simulating events not hits and each event can have a vastly different number of successful tricell hits, we need to estimate the average number of tricell hits per event. This number will be different for each detector.

For Test Beam events have on average 5 tricell hits (and 37 cell hits), so we need about 46×10^6 events in our simulation calibration sample. The simulation and calibration selection processes have about 90% efficiency. **So for Test Beam we need to simulate at least** 51×10^6 **events.** From period 4 we get 159,153,260 events with loose selection and 132,420,230 events with full selection. We decided to divide this sample in half to get a smaller simulation sample in the end.

2.2.5 Print the outputs

Write the pdg code, 4-momentum components and vertex positions into a hepevt-format text file which will then be fed to the TextFileGenerator.

2.3 Submitting the simulation jobs

1. The output of `generate_hepevt_cosmic.py` script is a single large `.txt` file (let's call it `TextGen_inFile.txt`. To run it on the grid, split it into multiple subfiles, each will be sourced by a separate `FHiCL` file. Don't forget that each event is written on 2 lines in the `txt` file, so split it into an even number of lines. From experience, 125,000 events (250,000 lines) are optimal, where each job runs for only a few hours. 250,000 events could be considered if the number of created subfiles would be >1000.

```
split -d -l 250000 --additional-suffix=.txt\
TextGen_inFile.txt /path/to/new/files/TextGen_inFile_
```

2. Then create the same amount of FHiCL files, each sourcing a different text file. There is a template called `TextFileGenTBjob_template.fcl`. Take a look at it and check (not only):

   - maxEvents
   - firstRun, firstSubRun
   - physics.producers.photrans.nd/fd/tb.BrightnessFile (which brightness file is used for the simulation)

   The brightness file describes the relative differences in energy response across the different cells and planes. These differences mainly arise from the variability of each fiber's brightness, and specifically for test beam also from the different scintillators used. Since we want the simulated detectors to be functional copies of the ideal versions of the real detectors, it is important to provide a correct brightness file without any defects. More information about the brightness files and how to create them can be found on the Test Beam calibration redmine wiki page.

   In the first iteration of the data-based simulation, Teresa used a test beam brightness file created from period 2 data, which contains faulty FEBs and underfilled cells, resulting in a simulation also containing these defects. In the second iteration Robert created a new brightness file from period 4 test beam data, which are free from any irregularities and supplied that to the simulation

```
bash CreateFclsForSimulation.sh
     TextFileGenjob_template.fcl
     /path/to/TextGen_infiles_directory/
     /path/to/output/TextGen_fcl_directory
```

3. Then create a SamWeb definition out of these FHiCL files

```
sam_add_dataset -n username_CosmicGen_description
                -t username_date
                -d /path/to/FHiCL/directory
```

You can also use the option `-no-rename` if you've named your FHiCL files uniquely enough.

4. Adjust the configuration script accordingly (`njobs`, `defname` for your fcls, `dest`,...) and include all the text files with:

```
for file in $(ls -1 /path/to/TextGen_inputfiles/*); do
echo --inputfile $file >> TextFileGen.cfg; done
```

   If you need to remove the previously included text files, you can do

```
sed -i '/^--inputfile/d' TextFileGen.cfg
```

5. Submit. First with the `-test_submission` flag and if everything looks all right, comment it out.

```
submit_nova_art -f TextFileGen.cfg
```

6. This results in artdaq-stage simulation files. Move them to a suitable area for long storage (for example persistent, ask production) and investigate them. You can re-use the CosmicGenAna module from section 2.1.

7. To create the calibration samples you can ask prodction to create them for you or to point you to a corresponding job to use. For Test Beam you can use this script

```
novaprod/novaproduction/fcl/testbeam/
prod_tb_ddactivity1_pclist_mc_job.fcl
```

   This creates the simulation calibration samples. You should move them a suitable area (like persistent) and create new definitions from them. You can again ask production for help.

3 Validation

To validate whether our simulation is working properly we

1. compare it to data and

2. use the new simulated as "fake data" and repeat the simulation process to get a re-simulated events.

Show plots and decribe data-MC comparison and gen3 to gen3.5 and gen3.7 comparisons. Maybe also refer to the TB calibration technote to see how was this simulation employed.

To validate whether our solution is working we use the result of the simulation as the "fake data" input for a second iteration of simulation and repeat every same in the exact same way. If the first and the second iteration of the simulation have a shift between each other, it means there's a selection efficiency bias.

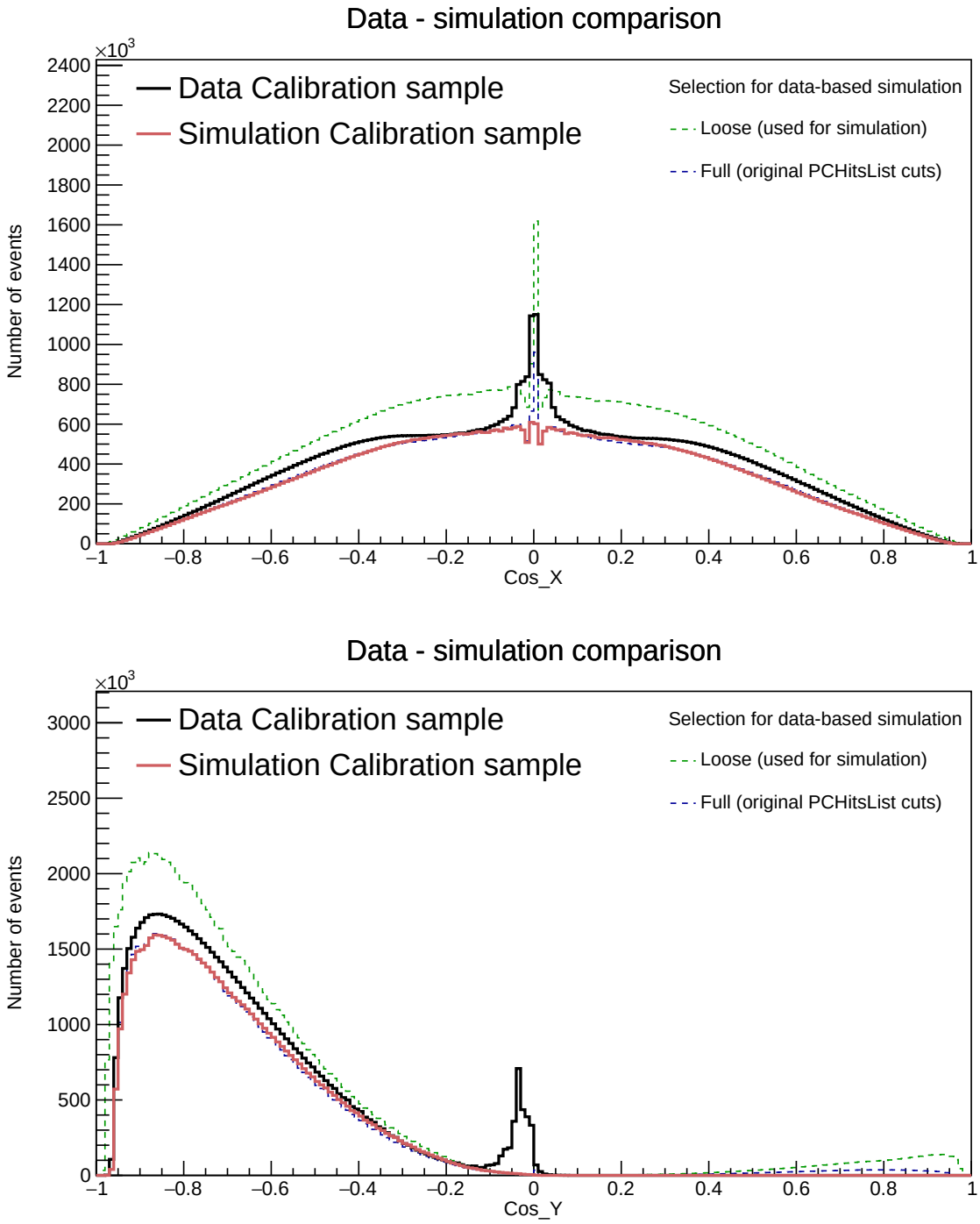

Figure 5: ...

4 Conclusions

Do we need this?

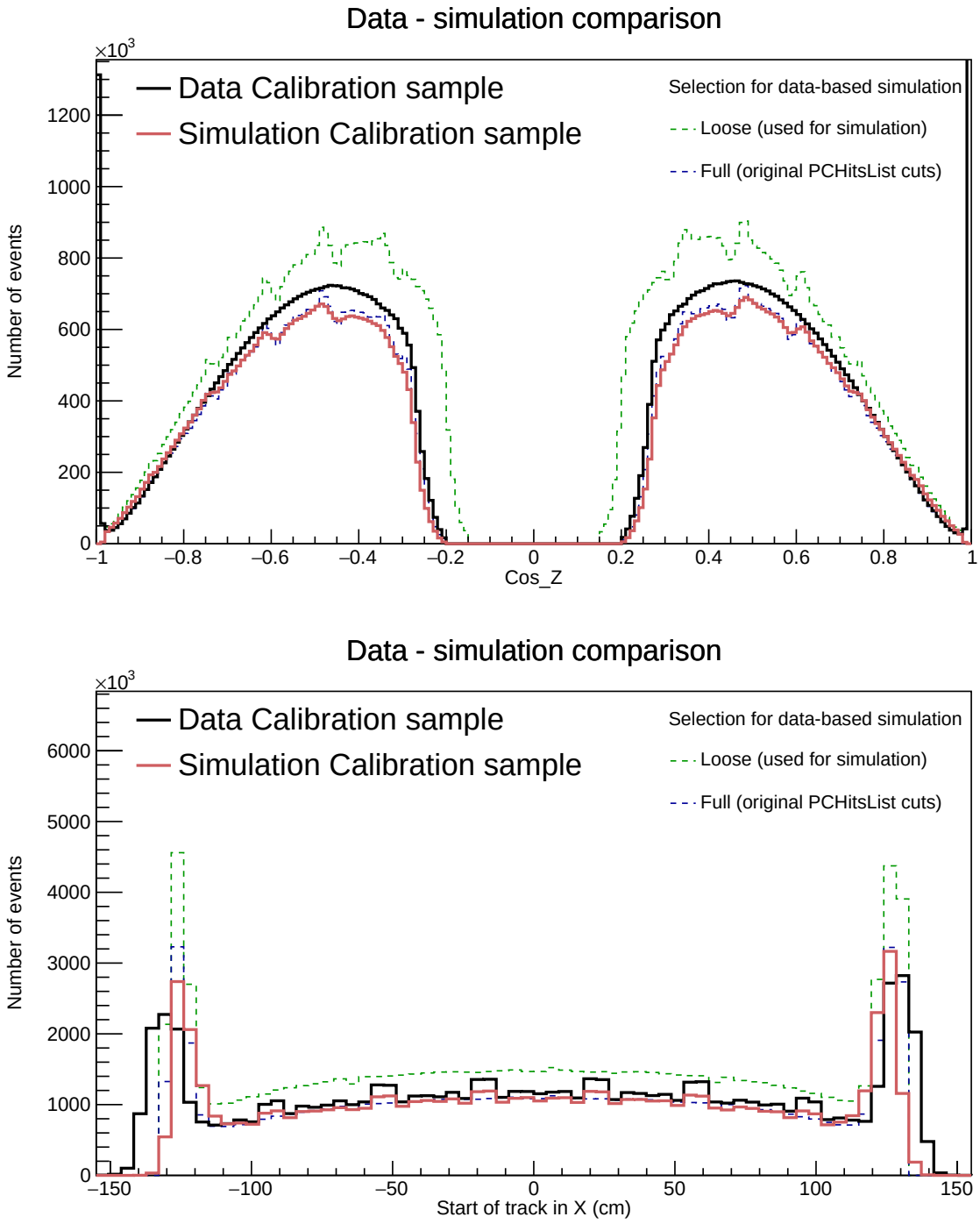

Figure 6: ...

References

[1] Kevin Mulder. Testbeam calibration update. NOVA Document 39244-v1. URL: https://nova-docdb.fnal.gov/cgi-bin/sso/ShowDocument?docid=39244.

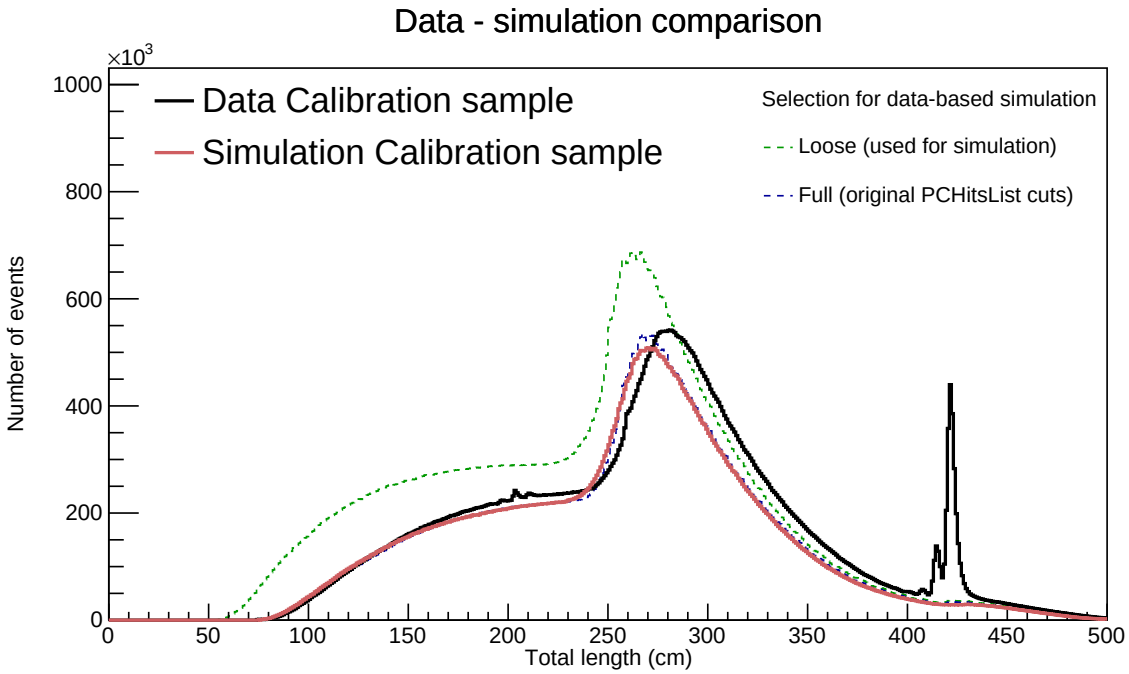

Figure 7: ...

[2] Mark Messier and Teresa Lackey. Data driven cosmic generation. NOVA Document 51327-v3, July 2021. URL: https://nova-docdb.fnal.gov/cgi-bin/sso/ShowDocument?docid=51327&version=3.

[3] R. L. Workman and Others. Review of Particle Physics. *PTEP*, 2022:083C01, 2022. doi:10.1093/ptep/ptac097.