

# PPS Progress Report

31.07.2025-25.08.2025

---

Grzegorz Jędrzejowski

25.08.2025

University of Warsaw  
Faculty of Physics

# Objectives

- Develop and implement analysis modules to apply physics and calibration cuts to the CTPPS Monte Carlo generated data.
- Create dedicated plotters to visualize the effects of these cuts and present key distributions.

# What's Been Done?

- The 2018 analysis was performed using the modified configuration file [Validation/CTPPS/test/simu/simu\\_2018\\_cfg.py](#).
- And the 2025 using the file: [Validation/CTPPS/test/simu/test\\_2025\\_man\\_cfg.py](#)

## Core Components

- **Producer:** [CTPPSGregDucer.cc](#)
- **Analyzer:** [CTPPSGregPlotter.cc](#)

## Plotting Scripts

- [plotGreg.py](#)
- [plotProtons.py](#)
- [plotTracks.py](#)

# Producer: Core Functionality

- Developed based on the existing framework:  
`SimPPS/DirectSimProducer/plugins/PPSDirectProtonSimulation.cc`.
- Its primary function is to apply a cut to the input data (HepMC).
- The cut parameters are defined in a separate [files](#) (calibration and physics), provided by [Mario Deile](#), with the following convention:
  1. The first line specifies the number of  $\phi$  steps, followed by their values (doubled per step).
  2. Subsequent lines define the cut specifics, starting with a  $\xi$  value, then  $\theta_{min}$  and  $\theta_{max}$  for each  $\phi$  step.
- The process pipeline involves three key steps:
  - The 'getCut()' method reads the cut file.
  - The 'produce()' method processes the data and interpolates it to match the cut using 'interpolate()' and 'interpolate\_step()'.
  - The 'applyCut()' method filters the data and returns the modified output.

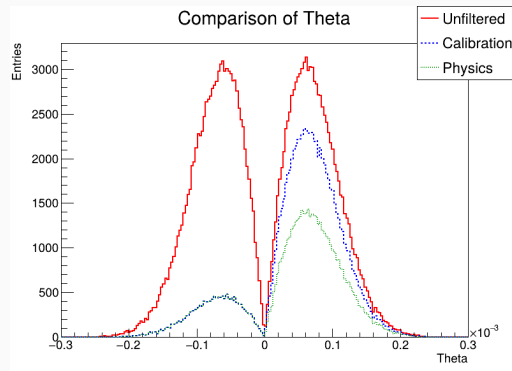
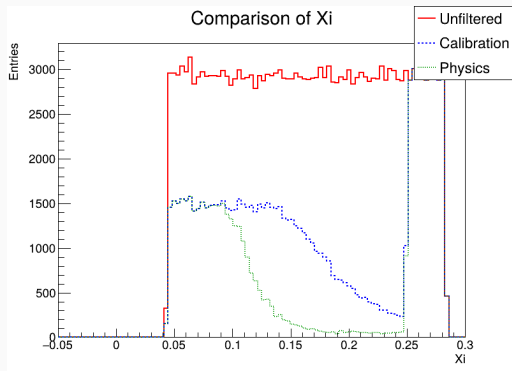
# Analyzer: Histograms and Data Flow

- Developed based on the existing framework:  
`Validation/CTPPS/plugins/CTPPSProtonReconstructionPlotter.cc`.
- The analyzer is executed twice in the workflow: once before applying the cut and again after.
- The 'analyze()' method is responsible for filling a variety of histograms:
  - **1D Histograms:**  $\phi$ , energy,  $p_T$ , and  $\xi$ .
  - **3D Histograms:**  $p_T$ - $\xi$ - $\phi$  and  $\theta$ - $\xi$ - $\phi$ .
- These 3D histograms are later projected to create 2D histograms, which can be adjusted according to different  $\phi$  slices.
- The analyzer saves the data in the `.root` file.

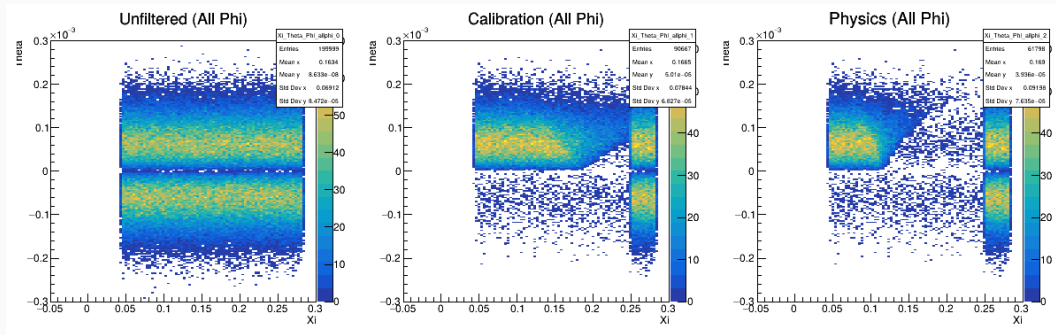
# Plotting Scripts: 'plotGreg.py'

- The 'plotGreg.py' script is designed to plot and compare three datasets:
  - **Unfiltered data:** Initial data saved by the analyzer.
  - **Calibrated data:** Data processed by the producer using a calibration file and saved using the analyzer.
  - **Physics data:** Data processed by the producer using a physics file and saved using the analyzer.
- **1D Histograms:**
  - Plots three histograms (one for each dataset) side-by-side on a single canvas.
  - These are then merged into a single histogram for better visualization and comparison.
- **3D Histograms:**
  - 2D projections are created for all  $\phi$  and plotted side-by-side.
  - For  $\phi$  slices, 12 separate histograms are displayed on a single canvas.
- All plots are saved in '.png' format within a dedicated output directory.

## Example Plots: 'plotGreg.py'

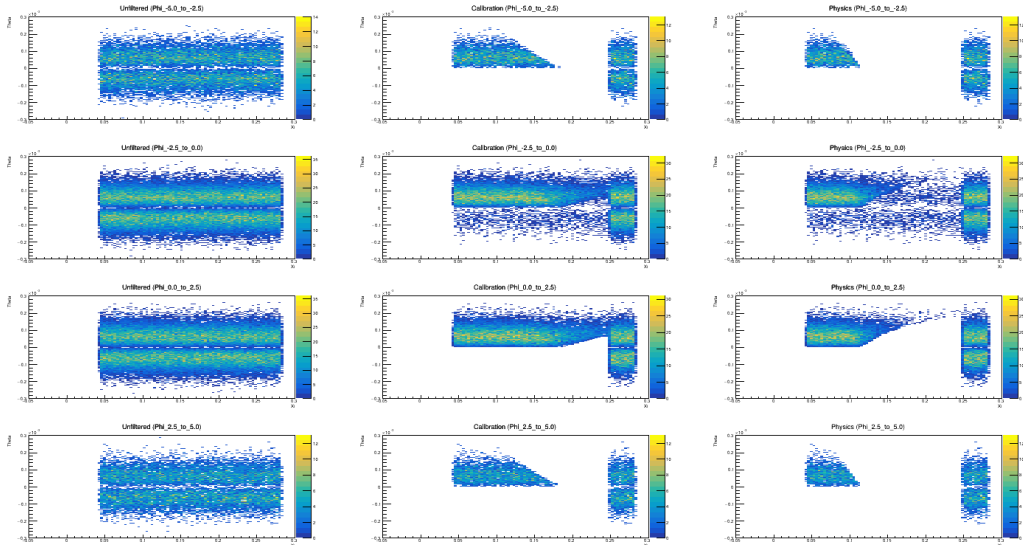


# Example Plots: 'plotGreg.py'





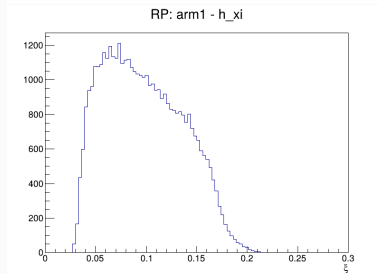
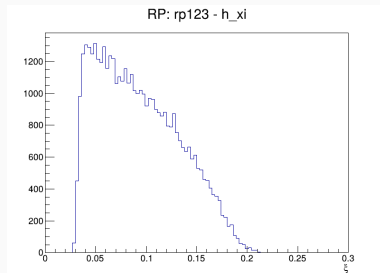
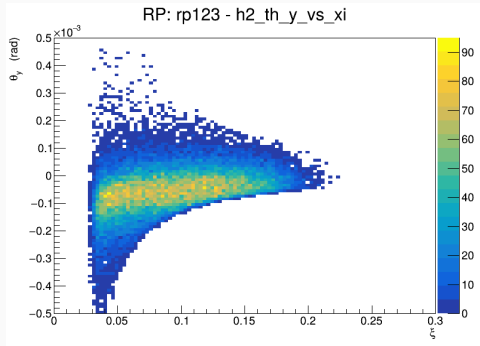
# Example Plots: 'plotGreg.py'



## Plotting Scripts: 'plotProtons.py'

- The 'plotProtons.py' script is designed to access and plot data generated by 'Validation/CTPPS/plugins/CTPPSProtonReconstructionPlotter.cc'.
- It processes data from individual Roman Pot (RP) units:
  - It accesses RPs 3, 23, 103, and 123.
  - It plots the "h\_xi" and "h2\_th\_y\_vs\_xi" histograms for each of these RPs.
- It also handles multi-RP data from both arms (Arm 0 and Arm 1), and for each arm, it draws the combined "h\_xi" histogram.
- All plots are saved to a dedicated output directory in '.png' format.

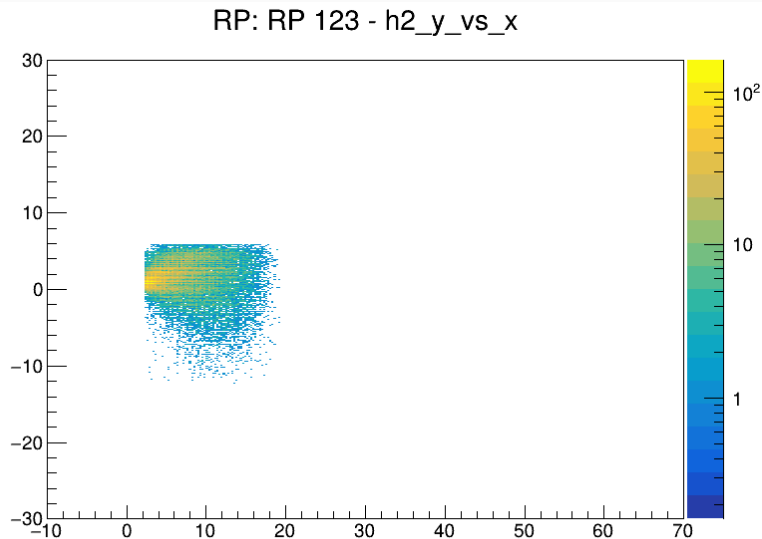
## Example Plots: 'plotProtons.py'



## Plotting Scripts: 'plotTracks.py'

- The 'plotProtons.py' script is designed to access and plot data generated by 'Validation/CTPPS/plugins/CTPPSTrackDistributionPlotter.cc'.
- It processes data from individual Roman Pot (RP) units:
  - It accesses RPs 3, 23, 103, and 123.
  - It plots the "h2\_y\_vs\_x" histogram for each of these RPs.
- All plots are saved to a dedicated output directory in '.png' format.

## Example Plots: 'plotTracks.py'



1. **Download CMSSW** e.g.: `CMSSW_15_0_11`; and in `src` run `cmsenv`.
2. **From Greg\_Jed [branch](#)** download the following packages:
  - Validation/CTPPS
  - SimPPS/DirectSimProducer
  - FWCore/AbstractServices
3. **Build the Project**
4. **Run the Analysis**
  - Navigate to the simulation directory: `cd Validation/CTPPS/test/simu/`.
  - Set up the `cmsenv`: `cmsenv` and run the simulation: `cmsRun simu_2018_cfg.py`.
  - Plot the results using the provided Python scripts, ensuring your environment can use python.

1. **Set up the 2018 compilation and execution procedure.**
2. **In src download packages and files:**
  - Geometry/VeryForwardGeometry
  - Geometry/VeryForwardData
  - [Root Files](#)
3. **Run the simulation:**
  - Go to the CMSSW\_/src directory and use the command `cmsRun ./Validation/CTPPS/test/simu/test_2025_man_cfg.py`.
4. **Plot histograms:**
  - In the CMSSW\_/src directory, run the Python script:  
`./Validation/CTPPS/test/simu/plotGreg.py`

# Questions?

If you have any questions, feel free to contact me:

[g.jedrzejows@student.uw.edu.pl](mailto:g.jedrzejows@student.uw.edu.pl)