McGill University
panagiotis.kaloyannis@physics.mcgill.ca

Detection of Wire Support Misalignment in sTGCs using Lower Measurement Sensitivity Zones

Panagiotis Kaloyannis, McGill, Montreal, Canada.

Supervised by Brigitte Vachon, McGill, Montreal, Canada.

Abstract

Due to the HiLumi upgrade of the LHC, many of the systems in the ATLAS detector are being upgraded to match the new luminosity target. Specifically, the muon system's Small Wheel (SW) has unsuitable performance for these upgrades. It is being replaced with the New Small Wheel (NSW) loaded with small Thin Gap Detectors (sTGCs). A novel technique which uses lower sensitivity regions in the detectors to detect the locations of support structures inside the detectors is developed. These could be cross validated with other analyses on the sTGCs at CERN. This report contains the documentation of the software package used as well as validation, scalability, and results from analysis of a QS3P8 quadruplet.

Montreal, Canada
October 7, 2020

1 Acknowledgments

I would like to first and foremost thank my advisor Prof. Brigitte Vachon who had put so many hours to help me get meaningful work done over the summer. Her guidance not only helped me when I was stuck, but egged me on to think of my results in new ways that lead to interesting theories!

Next I would like to of course thank the establishments that made my summer work possible. The IPP allowed me to get connected with people at CERN, in particular it led me towards Prof. Vachon. Despite the circumstances that prevented the internship that they originally offered me, the peers they allowed me to generate along the way were amazing!

My gratitudes go to NSERC for providing me the USRA needed not only for my IPP application but also that provided me funding for this project. Without them this work would not have been possible.

Finally, some thanks are due to my friends in family, without whom my journey to get to this report would have been far more bleak!

Contents

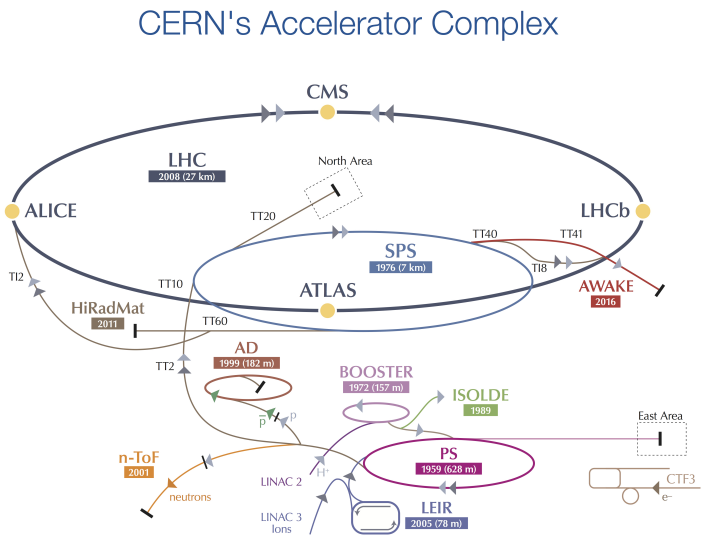

Figure 1: The CERN accelerator complex that is connected up to the LHC.[1]

2 Introduction

2.1 The Large Hadron Collider

The European Organization for Nuclear Research (CERN) is one of the world's largest and most respected centres for scientific research. It houses a plethora of particle accelerators, that is, machines that fire particles at astounding speeds and smashing them into stuff. The most notable of these accelerators is the Large Hadron Collider (LHC) which fires protons at each other at near the speed of light. The LHC is the largest machine ever built. Consisting of several nested rings that slowly bring the protons up to speed, the largest of which having a circumference of 27 km, before smashing them together. The purpose of such a machine is to probe the smallest objects we know of and test the validity of the Standard Model (SM) of particle physics. These measurements occur at four particle detectors in the ring - ATLAS, CMS, ALICE and LHCb [2].

The LHC was first brought online on September 10, 2008 and has since been a hub of discoveries. Most notably, the LHC discovered the Higgs Boson on July 4, 2012 leading to the Nobel prize in Physics of 2013 to be awarded to Peter Higgs and François Englert [3].

2.1.1 Upgrades People, Upgrades!

The LHC is currently down for High Luminosity Upgrades (LHC HiLumi), and is planned to restart its beams in early 2021. These upgrades will take place over the next

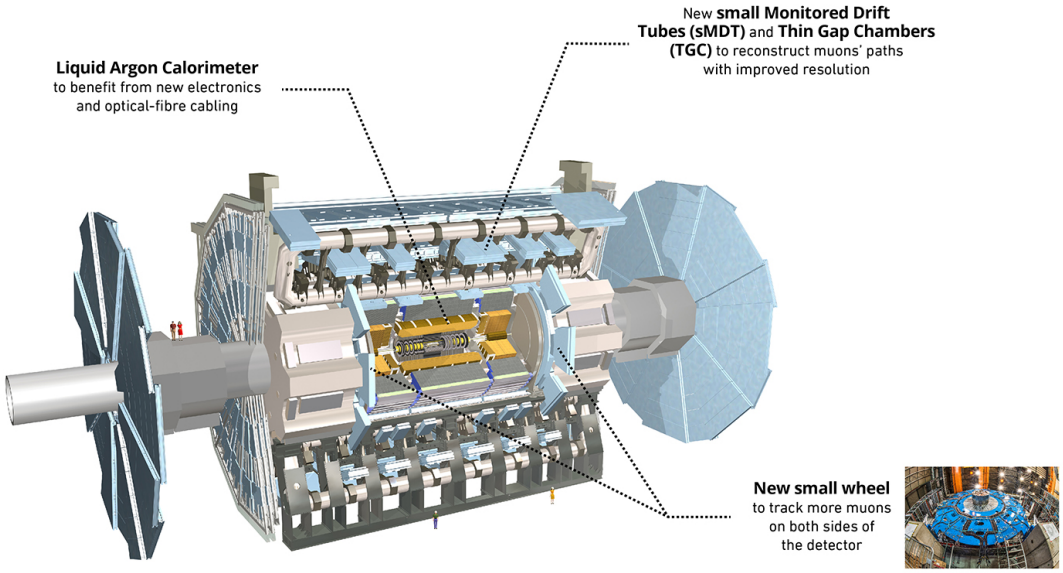

Figure 2: A cross section of the ATLAS detector pointing out the location of the NSW in the HiLumi upgrade. There are several humans for scale in the image. [5]

decade and boost the LHC's luminosity from its initial 30fb^{-1} to an estimated 300fb^{-1}. An accelerator measures statistical quantities, meaning the performance of the detector is not only proportional to the energy of the particles, but also how many collisions there are [4].

Some of the detectors originally implemented at the collision sites were not designed with such a high luminosity in mind. Thus many of them need to be replaced and upgraded. This analysis pertains to one of such systems that need to be replaced in entirely.

2.2 The ATLAS Detector

The ATLAS experiment is one of two-general purpose detectors on the LHC. It is designed to precisely measure, identify, and recreate the tracks of particles created in a collision in its center. It is comprised of 6 specialized systems organized in layers surrounding the collision site. The volume of data produced by the detector is controlled by an advanced triggering system that selects which data should be recorded. Because of the HiLumi upgrades, many systems in ATLAS are in the process of being upgraded and replaced. For this analysis in particular, we will be focusing on the muon detection system's Small Wheel being replaced by the New Small Wheel (see figure 2). A muon is essentially a heavy electron that passes through the Small Wheel ionizing layers of gas

in Thin Gas Chambers (TGCs). This ionization's strength and location is then recorded by several different electronics which can then reconstruct the trajectory that the muon has taken through the wheel. These TGCs do not have good efficiencies at the high luminosity of LHC HiLumi and thus must be replaced. The New Small Wheel (NSW) is loaded with the new small-strip Thin Gas Chambers (sTGCs) which will be the focus of this analysis.

2.3 The small-strip Thin Gas Chamber

A small-strip Thin Gas Chamber (sTGC) (see figure 3) is a trapezoidal thin gas volume of n-pentane and CO_2 in a ration of $55 : 45$ that is held together by two large PCBs. Suspended in the gas volume are small wires that represent the x-axis of the unit's coordinate system. There wires are held at 2800 V relative to the PCBs. This creates an powerful electric field inside the gas chamber and allows electrons that are ionized by muons in the gas to drift and ionize more particles causing an avalanche of ionization. This avalanche ultimately ends when the electrons reach one of the wires and are recorded as a small current pulse on that wire. One of the PCBs enclosing the volume has large copper pads on it's surface that are used for triggering when a muon passes through. The other PCB is covered with a carbon coating that spreads the positive charges generated from the ionization smoothly on its surface. This PCB also contains copper strips that run perpendicular to the wires suspended in the gas gaps and represent the y-axis of the detector's coordinate system. These strips collect the positive charges and record it as a current pulse.

These of course are missing the third axis required for reconstruction in full 3-D space, and so the sTGCs are stacked four times on top of each other and glued together to create what are called quadruplets (quads for short). The layer of the quad that the signal is read on effectively provides the z-axis of the detector, thus resulting in a full right handed coordinate system for each quadruplet. This z-axis actually runs parallel to the z-axis of the ATLAS detector, and so its poorer resolution is not problematic for track reconstruction. The x-axis has many wires in the gap at a fine resolution, but is read in wire bundles and not individually for computational reasons. The y-axis strips are all read individually and have a fine resolution, making them the finest axis in the sTGC. This is because the y-axis resolution of these detectors actually corresponds best to the resolution rapidity of the collision. The actual achievable resolution of the detector is even higher than this since the carbon coating spreads the signal over many strips, and can be best fit to for getting positions more precise than the strips resolution would allow. [6]

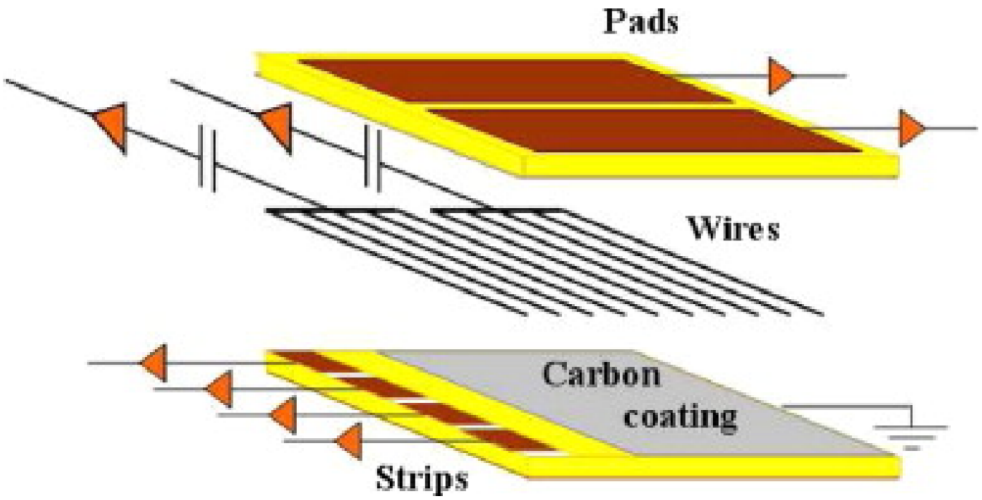

Figure 3: The general layout of an sTGC unit with the two PCBs on the top and bottom. The wire bundles suspended in the gap represent the x-axis and the strips in the bottom PCB represent the y-axis.[6]

2.4 Motivation

These structures on their own are wonderful, but they must operate in the context of the ATLAS detector. For any of this resolution to make sense, the location of the detector coordinate system must be precisely known in the ATLAS coordinate system. On top of this, any manufacturing imperfections that introduce offsets in position of any of these detector readout channels more than 100μm would effect the tracking significantly. Angular offsets between the PCB that holds the strips and the wires could cause a skewed x-y axis plane, or angular offsets in between two quadruplets could lead to rotations in the x-y plane as a function of z! All of these would be disasters for the accuracy target that the sTGC units are built for. Many efforts across many countries are all working to precisely document the performance and precise locations of each component of quadruplets. This report focuses on the development a new method to characterize some manufacturing imperfections in a quadruplet unit.

The wires in the gas gap are supported by small wire supports that run parallel to the y-axis of the detector. These supports are glued to each PCB and then pinch the wires in the middle, effectively creating a small low sensitivity zone. The supports are 7 mm in width, and in comparison to the 3.2 mm pitch of the strips, can cover about two strips at once. Precisely knowing the location of these supports may actually reveal some manufacturing imperfections in each quadruplet that can be cross-validated with several other methods in development. The goal is to use data from validation and

characterization runs of the detectors before they are sent to Switzerland from cosmic rays to precisely locate these supports along the whole detector.

3 Analysis Outline

This method is designed to operates on simplified cosmic ray data provided by the McGill sTGC lab. As inputs, it takes a root file with information of the strip ID, wire ID, ADC counts, and quadruplet layer for each muon hit. It also has pad information but this is unused in this analysis. The analysis is implemented in the wire-support position analysis package and goes as follows

1. Select a support strut and zone off angled sections of the sTGC.
2. Divide the data near that support into N bins along the x-axis that are M strips wide. Here N and M are left as free parameters.
3. Project all hits in that bin along the y-axis and best fit a function to the dip in data. Alternatively the projection can happen on simulated data that is sampled from a preset distribution.
4. Record the best fit locations and the effective width of the dip.
5. Examine these across an entire detector to determine patterns in the data.

This procedure is visualized in figure 4, but there are additional details not listed in this primitive list. Notably, many of the different settings are not mentioned. Additionally, one can best fit lines to the widths and locations determined to the effective slope of the supports in the detector. One can also create cuts in the voltages that allow for cutting off the noise to further clean and process the data.

3.1 Simulation

The primary use of the simulation built alongside the analysis was to quantify the maximum achievable statistical uncertainty on the best fit locations of the support and to verify that the results of the analysis were accurate. To do this, the sampling distribution was selected to be the inverted Gaussian, and it was fed inputs similar to what the best fit values were reporting on real data. Next, this was sampled over to fill the histograms created by the machinery of the analysis objects in a similar way to real data. Finally, these histograms were passed to the rest of the analysis machinery as if they were real data. Baked into the simulation is the ability to translate the distribution according to a preset slope so as to test ability to determine angular offsets between the support structure and axes of the detector. An extension of the simulation where it runs on a strip by strip basis and then is recombined together to for the bins used in analysis is also available but this smearing effect is negligible in real data so it is not recommended.

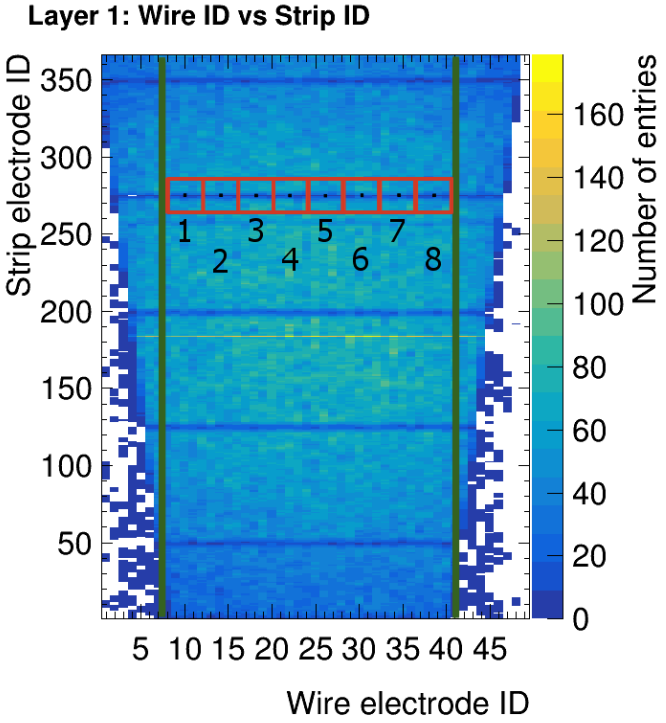

Figure 4: A visualization of the procedure taken in the analysis. The dark lines running horizontal in the plots are the low efficiency zones created by the support structures. The green lines mark the cuts on the angled sections of the sTGC. The red boxes indicate the binning of the data. The black dots in each bin correspond to the best fit location of the support within that bin.

4 Analysis Tool Description

Wire Support Position Analysis is a root software package that puts into practice the procedure outlined above. It is comprised of several C++ objects that represent different structures to be analyzed. The GitLab for this project is at `https://gitlab.cern.ch/McGill-sTGC/wire-support-position-analysis`.

4.1 Installation

To install Wire Support Position Analysis, one can begin on lxplus and download the repository. Running 'make' in the downloaded repository will compile the code. Included in this package are some slightly modified files of the DetectorGeometry class from the tgc-analysis package (`https://gitlab.cern.ch/McGill-sTGC/tgc_analysis`). The modifications are in the DetectorGeometryNSW.cpp file at lines 1018 onwards and the DetectorGeometryNSW.h file at line 80.

Alternatively, on a local machine one can compile the code by first installing root and xml2. Then running the command 'make compile' will compile the code.

4.2 Documentation

4.3 Files

- **AnalysisTemplate.cpp/.h :** Contains the AnalysisTemplate superclass for all the other classes in the package and the FuncPt datatype.
- **SupportAnalyzer.cpp/.h :** Contains the SupportAnalyzer object for analyzing/simulating a single support structure in a single quadruplet layer.
- **LayerAnalyzer.cpp/.h :** Contains the LayerAnalyzer object for analyzing/simulating a layer of a quadruplet.
- **QuadAnalyzer.cpp/.h :** Contains the QuadAnalyzer object for analyzing/simulating a quadruplet.
- **Analysis.cpp/.h :** A file containing common functions for validation and testing the classes.
- **FitFuncs.cpp/.h :** A file containing common functions that the classes use for best fitting.
- **AtlasStyle.cpp/.h :** A file containing the atlas root style. Used for creating pretty plots!

4.4 Classes

4.4.1 ANALYSISTEMPLATE

This object is entirely contained in AnalysisTempalte.cpp/.h and is the super class of the other major classes used in the analysis.
Protected Members:

- **int boxes :** Division of wire space as described in theory.
- **int width :** Width of histograms used for fitting as described in theory or for rendering (see fixedSize).
- **int npar :** Number of parameters in the fit function for correct statistics. May be deprecated.
- **int events :** Number of events per histogram for simulations. May be deprecated.
- **bool isSubObject :** Boolean noting if the object is for standalone use or if it initialized as a sub object.
- **bool fitLine :** Boolean noting if the widths and locations should have a line best fit to them.
- **bool fixedSize :** Boolean noting if histogram best fit should happen in a precise range and turns width into a rendering option only. Settings for this are on lines 282-291 in SupportAnalyzer.cpp.

- **double DataCut :** Boolean noting if the object is for standalone use or if it initialized as a sub object.
- **FuncPt fitFunc :** Pointer to a function used for the best fits. May be deprecated.
- **DetectorGeometry * Geom :** Pointer to a DetectorGeometry object from the tgc analysis package.
- **TFile * file :** Pointer to a TFile from tgc analysis package.
- **TTree * OutputTree :** Pointer to a TTree stored in file.
- **string FileInfo :** String containing the detector type and some info about it. Used primarily in folder creation.

Public Members:

- **AnalysisTemplate(int initboxes, int initwidth, int initnpar, double initfitFunction(double *, double *), const char *filename, const char *detectorType) :** A toy constructor. Not really useful.
- **AnalysisTemplate() :** The default constructor used by subObjects. Does nothing.
- **int GetBoxes() :** Gets boxes.
- **int GetWidth() :** Gets width.
- **int GetNpar() :** Gets npar.
- **int GetEvents() :** Gets events for the simulations.
- **DetectorGeometry *GetGeom() :** Gets the geometry object.
- **string GetFileInfo() :** Gets the file info.
- **double EvalFunction(double *vars, double *pars) :** Evaluates the function for fitting.
- **double GetDataCut() :** Gets dataCut.
- **bool IsSubObject() :** Gets SubObject.
- **bool IsFitLine() :** Gets fitLine.
- **bool IsFixedSize() :** Gets fixedSize.
- **string formatter(double value, double error) :** A number formatter to make nice scientific numbers with errors.
- **void folderMaker(const char *name) :** Creates folders for saving data.
- **void showData() :** Saves a very primitive plot of the initial data before analysis.
- **void filenameExtractor(const char *) :** Extracts the FileInfo from a filename.

4.4.2 SUPPORTANALYZER

The SupportAnalyzer object is the main work horse of the analysis and is a subclass of AnalysisTemplate object. The best way to visualize what this object is is an actual support on the detector designated by its layer and number (i.e. layer 1 support structure 3). Calling commands on this class are in a sense being called directly on that support structure. Thus calling the analyze() or simulate() commands described below do the actions of simulating and analyzing that support structure in particular. This object has two modes of operation: standalone where a support is being analyzed on its own or SubObject mode where the support is being analyzed in the context of the layer or entire quadruplet.
Protected Members:

- **int layer :** The layer of the Quad being analyzed from 1 to 4.
- **int supportNumber :** Integer specifying which support to analyze on a layer from 0 to 4.
- **int supportLoc :** The strip ID where the theoretical location of the support falls.
- **double StripSpacing :** The space between strips in real space.
- **vector<double> stripMax :** The maximum range for the histogram best fit functions (size 2).
- **vector<TH1I *> Hists :** A vector of histograms corresponding to each bin in the analysis that has a strip width set by the width variable (size boxes).
- **vector<TF1 *> functions :** A vector of functions used to fit the corresponding histogram for each bin in the analysis. These are made in the image of the template function provided in the initializer (size boxes).
- **vector<TFitResultPtr *> Fits :** A vector of fit results corresponding to each fitted function in functions (size boxes).
- **TF1 *dataTemplate :** The template function the simulation uses to sample a distribution from.
- **TF1 *line :** A straight line that would be used to fit the locations of the support structure.
- **TF1 *widthline :** A straight line that would be used to fit the widths of the support structure.
- **TGraphErrors *SupportPlot :** The plot of the support fit positions along the quadruplet with/without a line as set by fitLine.
- **TGraphErrors *WidthPlot :** The plot of the support fit widths along the quadruplet with/without a line as set by fitLine.
- **vector<double> bins :** A vector containing the upper and lower bounds to each bin (size boxes+1).
- **void supportFinder() :** A simple function to assign the correct supportLoc for each supportNumber.

Public Members:

- **SupportAnalyzer(int initboxes, int initwidth, int initnpar, int initlayer, int initsupportNumber, double initfitFunction(double *, double *), const char *filename, bool initFitLine = true, bool initfixedsize = false, double initdataCut = 0) :** The standalone mode initializer for the SupportAnalyzer class. The inputs are fairly self explanatory.
- **SupportAnalyzer(int initboxes, int initwidth, int initnpar, int initlayer, int initSupportNumber, double initfitFunction(double *, double *), TFile *initFile, DetectorGeometry *Geom, TTree *initOutputTree, string initFileInfo, bool initFitLine, bool initfixedsize, double initdataCut=0) :** The subobject mode initializer for the SupportAnalyzer class. The inputs are fairly self explanatory but this mode is intended for use in other objects.
- **SupportAnalyzer() :** The SupportAnalyzer destructor.
- **int GetLayer() :** Gets layer.
- **int GetSupportNumber() :** Gets supportNumber.
- **int GetSupportLoc() :** Gets supportLoc.
- **vector<double> GetBins() :** Gets bins.
- **vector<TF1 *> GetFunctions() :** Gets the fit functions.
- **vector<TH1I *> GetHists() :** Gets the histograms for each bin.
- **TF1 *GetLine() :** Gets the location best fit line.
- **TF1 *GetWidthLine() :** Gets the width best fit line.
- **TGraphErrors *GetSupportPlot() :** Gets the best fit locations plot.
- **TGraphErrors *GetWidthPlot() :** Gets the best fit widths plot.
- **void saveFunc() :** Saves the histograms for the analysis of a support structure.
- **void binner() :** Creates the correct binning for the analysis of a support structure.
- **void histmaker() :** Creates the corresponding histograms to store the data in the bins made by hist maker.
- **void histreset() :** Resets the values in the histograms for multiple simulations or analyses.
- **void filler() :** Fills the histograms with data from the file.
- **void fitter() :** Fits the provided fit function to the data in the histograms. Some tweaking must be done to the initial parameter guesses on lines 297 to 311 if one plans to change the functions.
- **void locPlot() :** Creates the plot of the locations of the support along the detector.
- **void widthPlot() :** Creates the plot of the widths of the support along the detector.
- **void analyze() :** Completes a full analysis of the support structure by combining the above functions.
- **TF1 *functionInit() :** Initializes the function the simulation will sample from.

- **void simulatedFiller(double slope) :** Fills the histograms with simulated data with a fixed slope for straight supports.
- **void simulate(int nEvents, double slope = 0.0) :** Completes a full simulation of nEvents per histogram and with a set slope of a support structure by combining many of the functions above.
- **void scaling() :** Many simulation runs that test the dependence of the statistically achievable precision with the amount of hits per histogram.
- **void rebox(int newBoxes) :** An unstable function that is able to change the number of bins on simulated data. This function primarily serves to create a blurring effect on simulations and is not recommended for use outside of that.

4.4.3 LAYERANALYZER

The LayerAnalyzer is comprised of 5 SupportAnalyzer objects in SubObject mode. It is a sub class of the AnalysisTemplate class. The best way to visualize what this object is is an actual layer on the detector (i.e. layer 1). Calling commands on this class are in a sense being called directly on all the supports on that layer. Thus calling the analyze() or simulate() commands described below do the actions of simulating and analyzing that layer in particular. This object has two modes of operation: standalone where a layer is being analyzed on its own or SubObject mode where the support is being analyzed in the context of an entire quadruplet.
Protected Members:

- **int layer :** The layer of the Quad being analyzed from 1 to 4.
- **TMultiGraph *mg :** A multigraph that contains all the TGraphErrors for the support locations of its supports. See buildMultiGraph().
- **vector<SupportAnalyzer *> supports :** A vector of SupportAnalyzer objects corresponding to each support on that layer (size 5).

Public Members:

- **LayerAnalyzer(int initboxes, int initwidth, int initnpar, int layer, double initfitFunction(double *, double *),const char *filename, bool initFitLine = true, bool initfixedsize = false, double initdataCut = 0) :** The standalone initializer for a LayerAnalyzer object.
- **LayerAnalyzer(int initboxes, int initwidth, int initnpar, int initlayer, double initfitFunction(double *, double *),TFile *initFile, DetectorGeometry *Geom, TTree *initOutputTree, string initFileInfo, bool initFitLine, bool initfixedsize, double initdataCut) :** The subobject mode initializer for the LayerAnalyzer class. This mode is intedended for use in other objects.
- **LayerAnalyzer() :** The class destructor.
- **void simulate(int nEvents, double slope = 0.0) :** This function simulates every support structure on that layer with nEvents per bin per support for a preset slope.

- **void analyze() :** A function that calls analyze() on all the supports on that layer.
- **void visualize() :** A function that visualizes the data in mg in a lovely way. Works best with fitLine=True.
- **void buildMultiGraph() :** A function that builds a TMultiGraph out of the SupportPlot from each SupportAnalyzer in the layer.
- **void rebox(int newBoxes) :** An unstable function that is able to change the number of bins on simulated data. This function primarily serves to create a blurring effect on simulations and is not recommended for use outside of that.
- **int GetLayer() :** Gets layer.
- **TMultiGraph *Getmg() :** Gets mg.
- **vector<SupportAnalyzer *> GetSupports() :** Gets the vector of support analyzers.

4.4.4 QUADANALYZER

The QuadAnalyzer is comprised of 4 LayerAnalyzer objects (or effectively 20 SupportAnalyzer objects) in SubObject mode. It is a sub class of the AnalysisTemplate class. The best way to visualize what this object is is the actual Quad. Calling commands on this class are in a sense being called directly on all the layers of that Quad. Thus calling the analyze() or simulate() commands described below do the actions of simulating and analyzing the Quad. This class does not have a SubObject mode.

Protected Members:

- **vector<LayerAnalyzer *> layers :** A vector of LayerAnalyzer objects that are running in subObject mode (size 4).

Public Members:

- **QuadAnalyzer(int initboxes, int initwidth, int initnpar, double initfitFunction(double *, double *),const char *filename, bool initFitLine = true, bool initfixedsize = false, double initdataCut = 0) :** The initializer for a QuadAnalyzer object.
- **QuadAnalyzer() :** The class destructor.
- **void simulate(int nEvents, double slope = 0.0) :** This function simulates every support structure in the Quad with nEvents per bin per support for a preset slope.
- **void analyze() :** A function that calls analyze() on all the layers in the Quad.
- **void visualize() :** A function that visualizes the data in the best fit locations of all support in the Quad in a lovely way! Works best with fitLine=True.
- **void rebox(int newBoxes) :** An unstable function that is able to change the number of bins on simulated data. This function primarily serves to create a blurring effect on simulations and is not recommended for use outside of that.
- **vector<SupportAnalyzer *> GetSupports() :** Gets the vector of layer analyzers.

4.5 Other Files and Methods

4.5.1 AtlasStyle.cpp

This is a slightly modified version of the AtlasStyle.C file that is used for creating ATLAS plots. It contains a specific TStyle pointer called AtlasStyle and a one line command to activate it.

4.5.2 FitFuncs.cpp

This function contains many template functions used by the objects in the code.

- **double invertedGaus(double *vars, double *pars) :** A simple inverted Gaussian and the main histogram fitting function used by the other classes. This function has npar $= 4$ and the parameters are {Mean,Norm,STD,C} and is computed as

$$f(x) = \text{Norm} * e^{-(\frac{x-\text{Mean}}{\text{STD}})^2} + C$$

- **double linear(double *vars, double *pars) :** A straight line function used for location best fitting and plotting. This has npar $= 2$ and the parameters are {Slope, C}. It is computed as

$$f(x) = \text{Slope} * x + C$$

- **double stepFunction(double *vars, double *pars) :** A step function used to best fit histograms. It has npar $= 4$ and the parameters are {Center, Norm, Width, C}. It is computed as

$$f(x) = \begin{cases} C & \text{Width} \leq |\text{Center} - x| \\ C - \text{Norm} & \text{Width} \leq |\text{Center} - x| \end{cases}$$

- **double summedGaus(double *vars, double *pars) :** This function is a sum of the inverted Gaussians from above to create a blurred Gaussian used for histogram fitting. It has npar $= 5$ and the parameters are {Center, Norm, STD, C, Spacing}. It is computed as

$$f(x) = \sum_{i=-1}^{1} \text{Norm} * e^{-(\frac{x-\text{Center}-i*\text{Spacing}}{\text{STD}})^2} + C$$

- **double fftStep(double *vars, double *pars) :** This is a Fourier Series for a step function with 20 terms. It is supposed to provide a slightly rounder step function for best fitting to histograms. It has npar $= 5$ and the parameters are {Center, Norm, Width, C}. It is calculated as:

$$f(x) = C + \sum_{i=1}^{20} \frac{\text{Norm}}{i} [\sin(i * \text{Center} * \text{Width}) \sin(i/2) \sin(i * x * \text{Width}) + \cos(i * \text{Center} * \text{Width}) \sin(i/2) \cos(i * x * \text{Width})]$$

4.5.3 ANALYSIS.CPP

This file is treated as the main interface between the objects and contains many useful logical operations that combines all the structures in the package. Obviously these objects can be used outside of this file but It is already set up with all the correct imports.

- **int main(int argc, char *argv[]) :** This it the main for the package, where one can actually run the analysis or many of the other functions in the Analysis.cpp file. It always must start by calling SetAtlasStyle() for the plots to come out looking correct.
- **void compareFunc() :** This function compares the performance of the invertedGaussian and summedGaussian fit functions. It outputs the result in an easy to read CSV for python.
- **double chisqCollector(QuadAnalyzer *Quadruplet) :** Collects the average reduced χ^2/ndf for an entire quadruplet.
- **void validator(const char *filename) :** This creates a primitive plot of the average statistical uncertainty on a layer of a quadruplet as a function of the number of hits per histogram.
- **void AnalyzeData(int argc, char *argv[]) :** Analysis results from both the 3100V and 2900V data for a single detector and writes it into some convenient files for further processing. One file creates convenient LaTeX tables and the other two contain the slopes and the errors in their slopes. To feed it data just feed in filenames into argv.
- **void widthHypothesis() :** Tests the hypothesis that the invertedGaussian fit performs better at the extremities of the histrograms than in the center even in simulations.
- **void boxHypothesis() :** Tests the hypothesis that there is a blurring effect making the smaller number of bins perform better.
- **void ComparePlots(int argc, char *argv[]) :** This compares point by point results between 2900V and 3100V data. To feed it data just feed in filenames into argv.
- **void LocPlotCombiner(SupportAnalyzer *Run2900V, SupportAnalyzer *Run3100V) :** A command called in ComparPlots that compares the locations.
- **void WidthPlotCombiner(SupportAnalyzer *Run2900V, SupportAnalyzer *Run3100V) :** A command called in ComparPlots that compares the widths.
- **void CompareBinPlots(const char *data) :** Called to compare the results of the same run with different bin counts.
- **void ManyBinLocCombiner(vector<SupportAnalyzer *> Supports) :** Compares the locations of many bin counts, used in CompareBinPlots.
- **void ManyBinWidthCombiner(vector<SupportAnalyzer *> Supports) :** Compares the widths of many bin counts, used in CompareBinPlots.
- **void CompareCutPlots(const char *data) :** Called to compare the results of the same run with different voltage cuts.
- **void ManyCutLocCombiner(vector<SupportAnalyzer *> Supports) :** Compares the locations of many voltage counts, used in CompareCutPlots.

- **void ManyCutWidthCombiner(vector<SupportAnalyzer *> Supports) :** Compares the widths of many voltage counts, used in CompareCutPlots.

4.6 Usage and Examples

Primarily these objects were designed to be used in the Analysis.cpp file with the following general workflow:

1. Call SetAtlasStyle() for the figure creation.
2. Initialize the analysis objects as needed for the task at hand (This has some caveats).
3. Analyze or Simulate on the object. This automatically generates a few plots for the effective fit width and effective fit locations.
4. Further process the results from the automated Analysis. These results are most often stored in the TF1s that were used to best fit the histograms or in the TGraphErrors. The results can be extracted with the getters.
5. Delete the objects. Be very careful to not delete any parts of the objects in your further processing.

Upon initializing the analysis objects, one automatically generates a primitive plot on the input data showing the number of muon hits as a function of stripID per layer as well as a plots directory with a sub folder named appropriately for each run. After running a simulation or analysis, more subfolders are generated for each layer and are filled with plots of every individual fit and plots of the fit location and width over the whole detector. One can also create comparison plots between different analysis objects by using the comparison methods in Analysis.cpp. There are other outputs possible, for example the whole quadruplet or layer plots created by visualize() may be called explicitly or some of the analytic fit quality functions in Analysis.cpp will generate some .dat files for further reading and processing.

Many examples of this are present in the Analysis.cpp document as essentially every command follows this structure for data flow. Here is an example that collects results on the performance of the invertGaussian on data from the McGill sTGC lab.

```
1  void compareFunc()
2  {
3      SetAtlasStyle(); //This is the first step to the data flow!
4
5      //now we begin our dataflow. Here we are trying to run over many
       settings for widths and boxes.
6      //here we initialize a quad analyzer and complete a full data
       flow cycle for each set of settings
7      for (int width = 9; width < 22; width += 4)
8      {
9
10
11         for (int boxes = 5; boxes < 16; boxes += 5)
12         {
13             //2. initialize
```

```
            auto Invert = new QuadAnalyzer(boxes, width, 4,
                                invertedGaus,
                                "./QL2C8_3100V_P1_TTree.root");
            //3. analyze
            Invert->analyze();
            //4. Further process by collecting the reduced chi^2 from another function and printing
            cout << chisqCollector(Invert) << endl;
            //5. delete object
            delete Invert;
        }
    }
    //thus concludes this short example that is taken as an excerpt of the compareFunc() method in Analysis.cpp.
}
```

There are a few limitations that were caused by poor foresight in the structure of these objects that frankly do not cause enough of an issue to have warranted a change. First and foremost, **a data file must be loaded into the objects even if simulating!** This can be a little annoying but it would have taken too much time to fix. Second, **one cannot delete any of the analysis objects without either analyzing or simulating on them!** This is once again a mild inconvenience since there is no reason to initialize these objects unless the intention is to process them further.

4.6.1 Implementing Custom fitting functions.

To insert your own fitting function for experimentation the procedure is fairly straightforward. Simply create a function that returns a double and has two arrays of doubles as inputs. The first array corresponds to the Y location of each bin and thus has size 1. The second array is of a user selected size npar and is the input parameters. The first input must be the determined location and the second must be the effective width. Then, passing the function to the analysis object of the users choice with the according npar will make the fit. To adjust initial parameter guesses, they are stored in SupportAnalyzer.cpp at lines 290 to 315.

5 Results and Validation

Many unexpected patterns are manifested in the results of the analysis. These prompted dropping some of the initial assumptions made. Originally, it was expected that the supports would form clean and straight lines when best fitted to but it was quickly realized this is not the case. This is still under investigation, but explains the absence of many of the plots that the analysis package is capable of creating from this report as they are nonsensical without such an assumption.

The data used in this analysis is from the McGill sTGC lab that tests and characterizes Canadian made quadruplets with cosmic rays. This data collection is done in a hodoscope that mounts the quadruplet with a PMT/scintillator on the top and the bottom.

When the scintillators measure concurrent hits, the readout electronics are triggered. Data is collected at 2900V instead of 2800V because of altitude differences between the LHC (underground) and the lab (atop a hill). There is also data collected at 3100V because the electronics in the lab are different then those being installed in the LHC and need increased gain from the detectors.

A single data run on a Quadruplet is done at 2900V or 3100V and comprises of 4 one hour runs of collection. This is so if there is a problem in the run, the entire 4 hour collection period is not wasted. Thus, for this analysis all data is the equivalent of 4 hours of data collection. Using 2900V and 3100V concurrently is similar to 8 hours of data.

5.1 Fitting Functions

The fitting functions in this code were all tested to different levels to determine how well they worked but have all been included in the original code. This section focuses on the characterization of the best fit functions performances starting with the two least impressive functions.The step function and Fourier series had the worst performance and were never tested with any rigorous methods. Just examining the fits in figure 5 by eye revealed that they were consistently bad across the board. The step function was initially added under the assumption that we would have an abrupt dead zone where the support is covering the strips. This proved to not be the case and so it and its Fourier series (which aimed to make the the step less sharp) were quickly discarded as unsuitable.

The inverted Gaussian and summed Gaussian were next in the list. In figure 5, both performed reasonably well. More analytical methods were then developed to determine the performance of each for an entire detector. Using data from QS3P8, the average $\chi 2/ndf$ was collected for the entire detector with many analysis settings. The goal was to select the fit that had $\chi 2/ndf$ that was the closest to 1 since that was an indication of a healthy fit. The table 1 shows the results of this computation and reveals two interesting patterns beyond the identification of the better performing function. It is evident that that inverted Gaussian does perform consistently better than the summed Gaussian, and so all analysis from here onward was conducted with the inverted Gaussian. The trend of the reduced χ^2 to decrease with bins and widths was next, as it could provide some insight on the performance of the functions with different settings. Two hypotheses were conjured up as to why these patterns are appearing.

First, as the strip width of the histograms increases, the fits perform better because the relatively flat area outside the dip is much easier to fit to than the dip. To test this, a simulation generated many plots of different widths for an entire quadruplet with a similar amount of data but with the sampling distribution being the exact same as the fitting function (inverted Gaussian). If this hypothesis is correct, then even fitting the the same function as the distribution should demonstrate the same effect. Table 2 shows

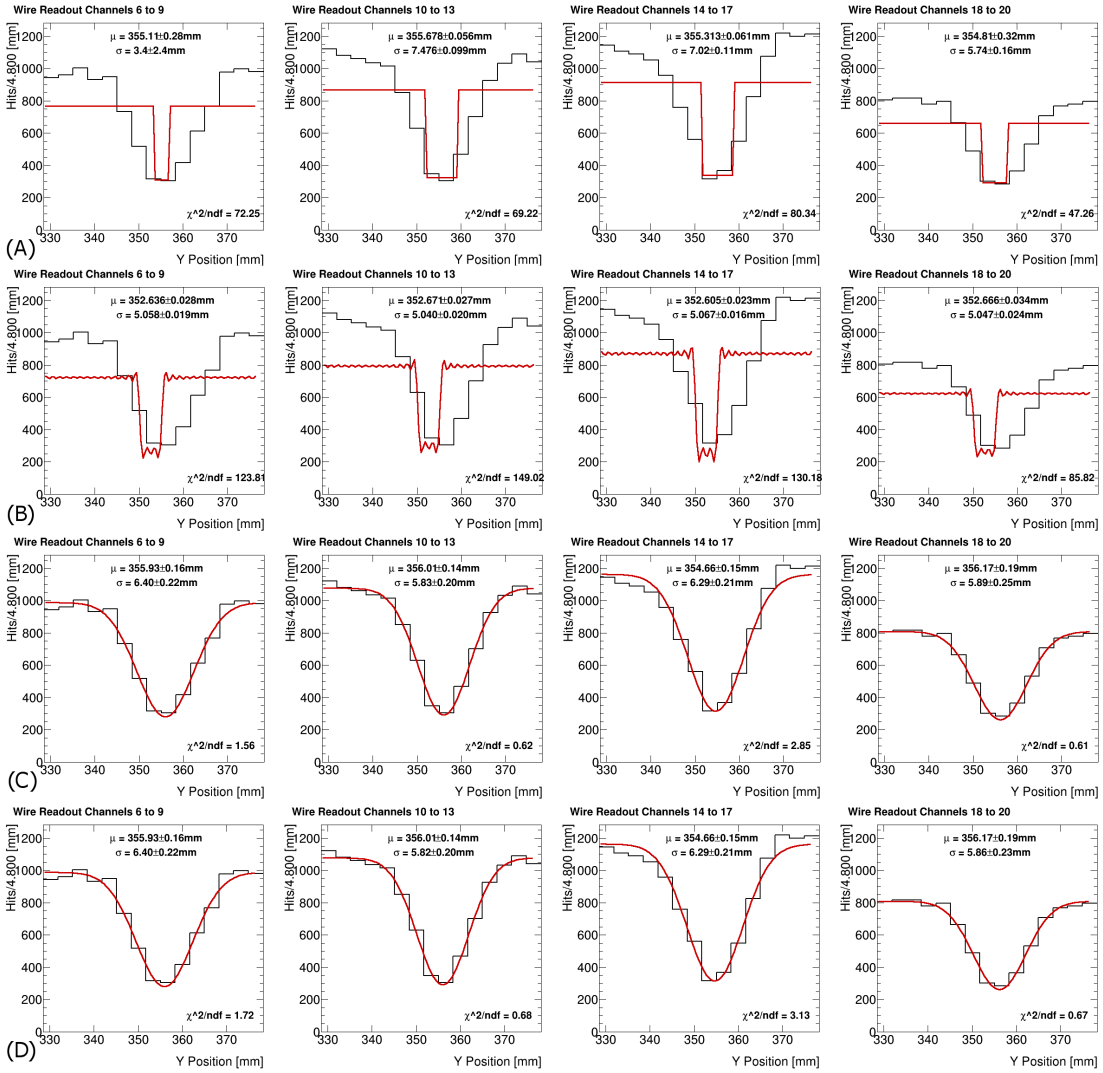

Figure 5: Four typical fits on a support from QS3P8 data by (A) the step function, (B) the Fourier series of a step function, (C) an inverted Gaussian, and (D) the summed Gaussian. This is meant to illustrate the the similar performance at fitting to the dip by C and D but the inadequacy of (A) and (B).

Type	5 Bins	10 Bins	15 Bins
Strip Width = 9			
Inverted Gaussian	3.89	2.12	1.49
Summed Gaussian	4.68	2.51	1.78
Strip Width = 13			
Inverted Gaussian	3.53	2.02	1.45
Summed Gaussian	3.79	2.15	1.56
Strip Width = 17			
Inverted Gaussian	2.95	1.77	1.32
Summed Gaussian	3.05	1.86	1.38
Strip Width = 21			
Inverted Gaussian	2.69	1.66	1.29
Summed Gaussian	2.74	1.69	1.32

Table 1: Comparison of the average reduced χ^2 for fits conducted using the inverted Gaussian and summed Gaussian functions on real QS3P8 data at 3100V. The Inverted Gaussian has values that are consistently much closer to 1 across all settings, and so it is deemed the superior functions. It is important to note that the reduced χ^2 decreases with smaller bins and with larger widths.

that this trend is indeed matched even in simulation, and that the confirms that this is a consequence of the fit being more stable at its extremities.

Width	9	11	13	15	17	19	21	23
χ^2/ndf	2.28	1.73	1.42	1.21	1.04	1.04	0.98	1.00

Table 2: The reduced χ^2 for a simulation of a QS3P quadruplet with approximately the correct amount of data per bin for each width to match reality. The fitting function here matches the sampling distribution of hits for the simulation, and so the downward trend indicates that the fit is just more stable in the extremities where the distribution is sampled more often

Second, the bin hypothesis states that a blurring effect from the location of the support moving considerably over the x-axis length of the bin actually makes the fits perform worse. To test this, an extension to the simulation capabilities of the code was built that provided a more realistic scenario. The simulation acted on a strip by strip basis, and then added all the hits together at the end to create such a blur. Presumably, the larger the slope that the support structure would have the worse this trend would be and so the simulation was ran at gradually increasing slopes. The results showed that in the ranges of slopes expected for real detectors, this effect is negligible. At this point no hypothesis has been created that can explain away this trend.

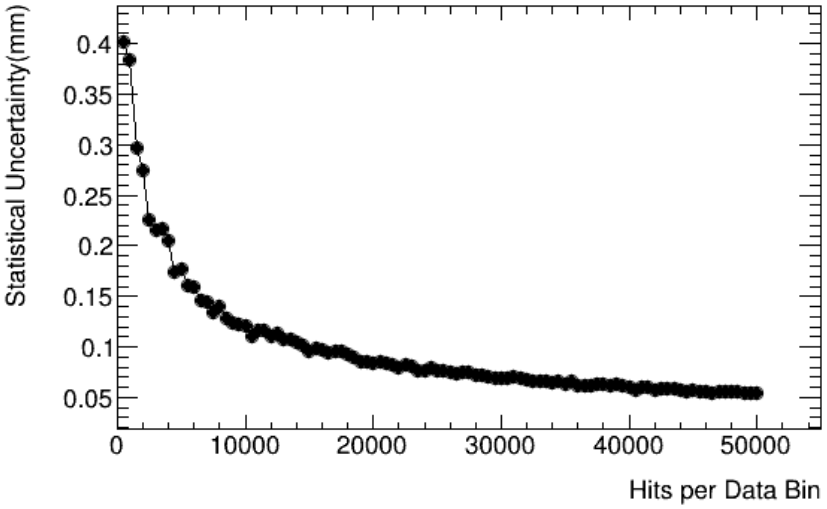

Figure 6: Demonstrating the analysis' ability to scale in statistical uncertainty as a function of amount of data per bin. This is done with a fixed bin width of 15 using a QS3P8 as a model and running scaling() from a SupportAnalyzer object.

5.2 Validation and Data Scaling

The simulation was run at various slopes, number of hits, and initial settings to determine the performance of the analysis at determining the support locations from the data. The results were quite promising for the best achievable statistical uncertainty of $\sim 150\mu$m on locations of the support along the detector for a realistic amount of data. Similarly if the assumption is made that the support is a straight line, an estimate of the slope can be made to within 10^{-3}mm/mm. Simulations quite easily can surpass these numbers even with only double the data per quadruplet per voltage (represented by an 8 hour run instead of a 4 hour run). The statistical uncertainty as a function of number of hits curves can be seen in figure 6.

Hits	Slopes 1.00E-05	1.00E-04	1.00E-03
1000 Hits	$(1.1 \pm 1.9)x10^{-4}$	$(1.5 \pm 1.8)x10^{-4}$	$(1.29 \pm 0.19)x10^{-3}$
2000 Hits	$(1.5 \pm 1.3)x10^{-4}$	$(-0.57 \pm 1.33) \times 10^{-4}$	$(1.20 \pm 0.13)x10^{-3}$
3000 Hits	$(0.21 \pm 1.05) \times 10^{-4}$	$(0.82 \pm 1.07) \times 10^{-4}$	$(1.13 \pm 0.10)x10^{-3}$
4000 Hits	$(1.65 \pm 0.92)x10^{-4}$	$(-1.9 \pm 9.2)x10^{-5}$	$(1.133 \pm 0.093)x10^{-3}$
5000 Hits	$(-0.52 \pm 8.27) \times 10^{-5}$	$(4.1 \pm 8.2)x10^{-5}$	$(1.045 \pm 0.084)x10^{-3}$
6000 Hits	$(6.4 \pm 7.6)x10^{-5}$	$(5.9 \pm 7.6)x10^{-5}$	$(9.54 \pm 0.76)x10^{-4}$
7000 Hits	$(-7.9 \pm 7.1)x10^{-5}$	$(4.6 \pm 7.0)x10^{-5}$	$(9.77 \pm 0.71)x10^{-4}$
8000 Hits	$(-5.6 \pm 6.6)x10^{-5}$	$(1.14 \pm 0.66)x10^{-4}$	$(1.022 \pm 0.066)x10^{-3}$
9000 Hits	$(3.5 \pm 6.2)x10^{-5}$	$(1.14 \pm 0.63)x10^{-4}$	$(9.67 \pm 0.63)x10^{-4}$
10000 Hits	$(-1.02 \pm 0.59)x10^{-4}$	$(1.98 \pm 0.59)x10^{-4}$	$(1.069 \pm 0.059)x10^{-3}$
11000 Hits	$(-1.1 \pm 5.6)x10^{-5}$	$(8.4 \pm 5.6)x10^{-5}$	$(9.81 \pm 0.56)x10^{-4}$
12000 Hits	$(3.7 \pm 5.4)x10^{-5}$	$(7.5 \pm 5.4)x10^{-5}$	$(1.139 \pm 0.054)x10^{-3}$
13000 Hits	$(2.4 \pm 5.2)x10^{-5}$	$(1.63 \pm 0.52)x10^{-4}$	$(8.97 \pm 0.52)x10^{-4}$
14000 Hits	$(-1.5 \pm 5.0)x10^{-5}$	$(1.09 \pm 0.49)x10^{-4}$	$(1.063 \pm 0.050)x10^{-3}$
15000 Hits	$(-1.7 \pm 4.8)x10^{-5}$	$(-1.1 \pm 4.8)x10^{-5}$	$(9.90 \pm 0.48)x10^{-4}$
16000 Hits	$(0.53 \pm 4.69) \times 10^{-5}$	$(0.65 \pm 4.66) \times 10^{-5}$	$(1.054 \pm 0.047)x10^{-3}$
17000 Hits	$(-4.8 \pm 4.5)x10^{-5}$	$(5.8 \pm 4.6)x10^{-5}$	$(1.067 \pm 0.046)x10^{-3}$
18000 Hits	$(2.6 \pm 4.4)x10^{-5}$	$(1.35 \pm 0.44)x10^{-4}$	$(1.027 \pm 0.044)x10^{-3}$
19000 Hits	$(2.3 \pm 4.3)x10^{-5}$	$(1.53 \pm 0.43)x10^{-4}$	$(9.89 \pm 0.43)x10^{-4}$
20000 Hits	$(-3.3 \pm 4.2)x10^{-5}$	$(6.6 \pm 4.2)x10^{-5}$	$(9.23 \pm 0.42)x10^{-4}$
21000 Hits	$(-1.4 \pm 4.1)x10^{-5}$	$(1.82 \pm 0.41)x10^{-4}$	$(9.56 \pm 0.41)x10^{-4}$
22000 Hits	$(-1.9 \pm 4.0)x10^{-5}$	$(9.2 \pm 4.0)x10^{-5}$	$(9.47 \pm 0.40)x10^{-4}$
23000 Hits	$(0.90 \pm 3.90) \times 10^{-5}$	$(1.41 \pm 0.39)x10^{-4}$	$(9.75 \pm 0.39)x10^{-4}$
24000 Hits	$(1.7 \pm 3.8)x10^{-5}$	$(3.5 \pm 3.8)x10^{-5}$	$(1.015 \pm 0.038)x10^{-3}$
25000 Hits	$(4.7 \pm 3.7)x10^{-5}$	$(6.9 \pm 3.7)x10^{-5}$	$(1.050 \pm 0.037)x10^{-3}$
26000 Hits	$(4.1 \pm 3.7)x10^{-5}$	$(1.55 \pm 0.37)x10^{-4}$	$(1.056 \pm 0.037)x10^{-3}$
27000 Hits	$(1.8 \pm 3.6)x10^{-5}$	$(2.9 \pm 3.6)x10^{-5}$	$(9.69 \pm 0.36)x10^{-4}$
28000 Hits	$(9.2 \pm 3.5)x10^{-5}$	$(4.6 \pm 3.5)x10^{-5}$	$(1.021 \pm 0.035)x10^{-3}$
29000 Hits	$(-0.59 \pm 3.48) \times 10^{-5}$	$(6.7 \pm 3.5)x10^{-5}$	$(1.007 \pm 0.035)x10^{-3}$
30000 Hits	$(-2.3 \pm 3.4)x10^{-5}$	$(9.8 \pm 3.4)x10^{-5}$	$(9.99 \pm 0.34)x10^{-4}$
31000 Hits	$(-2.3 \pm 3.4)x10^{-5}$	$(9.7 \pm 3.4)x10^{-5}$	$(1.017 \pm 0.034)x10^{-3}$
32000 Hits	$(-4.0 \pm 3.3)x10^{-5}$	$(1.46 \pm 0.33)x10^{-4}$	$(1.013 \pm 0.033)x10^{-3}$
33000 Hits	$(3.8 \pm 3.3)x10^{-5}$	$(1.09 \pm 0.33)x10^{-4}$	$(9.87 \pm 0.32)x10^{-4}$
34000 Hits	$(4.0 \pm 3.2)x10^{-5}$	$(1.30 \pm 0.32)x10^{-4}$	$(1.008 \pm 0.032)x10^{-3}$
35000 Hits	$(5.2 \pm 3.2)x10^{-5}$	$(8.3 \pm 3.2)x10^{-5}$	$(9.90 \pm 0.32)x10^{-4}$
36000 Hits	$(-2.6 \pm 3.1)x10^{-5}$	$(1.64 \pm 0.31)x10^{-4}$	$(9.74 \pm 0.31)x10^{-4}$
37000 Hits	$(-0.048 \pm 3.060) \times 10^{-5}$	$(9.2 \pm 3.1)x10^{-5}$	$(1.017 \pm 0.031)x10^{-3}$
38000 Hits	$(3.9 \pm 3.0)x10^{-5}$	$(9.3 \pm 3.0)x10^{-5}$	$(1.027 \pm 0.030)x10^{-3}$
39000 Hits	$(-2.9 \pm 3.0)x10^{-5}$	$(6.7 \pm 3.0)x10^{-5}$	$(1.023 \pm 0.030)x10^{-3}$
40000 Hits	$(-2.7 \pm 3.0)x10^{-5}$	$(1.37 \pm 0.29)x10^{-4}$	$(1.001 \pm 0.029)x10^{-3}$
41000 Hits	$(2.1 \pm 2.9)x10^{-5}$	$(7.5 \pm 2.9)x10^{-5}$	$(1.035 \pm 0.029)x10^{-3}$
42000 Hits	$(1.6 \pm 2.9)x10^{-5}$	$(1.18 \pm 0.29)x10^{-4}$	$(1.073 \pm 0.029)x10^{-3}$
43000 Hits	$(2.4 \pm 2.9)x10^{-5}$	$(1.20 \pm 0.28)x10^{-4}$	$(1.055 \pm 0.028)x10^{-3}$
44000 Hits	$(4.4 \pm 2.8)x10^{-5}$	$(1.44 \pm 0.28)x10^{-4}$	$(9.83 \pm 0.28)x10^{-4}$
45000 Hits	$(0.88 \pm 2.78) \times 10^{-5}$	$(7.8 \pm 2.8)x10^{-5}$	$(1.001 \pm 0.028)x10^{-3}$

Table 3: The real slope of the simulation on QS3P8 versus the predicted slope from the analysis in relation to the number of hits in each bin.

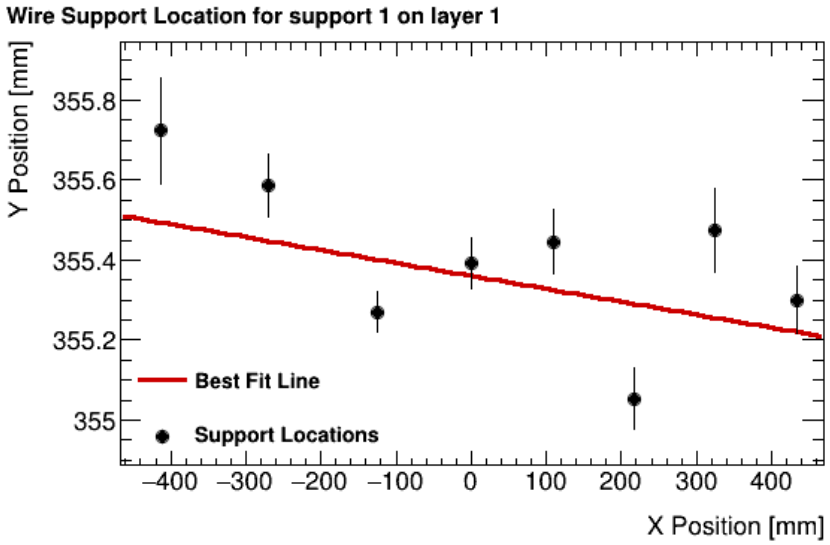

Figure 7: The typical result for best fit locations in the bins from a support in QS3P8 at 3100V. It is fairly obvious that a line is not the greatest fit here as only a single error bar intersects the line. There is a hint of a periodic effect in this and many other location plots that has yet to be explained.

To validate the accuracy of the analysis, artificial slopes in the simulated data were introduced for a layer. Then, the prediction of the slope made by the analysis was entered into table 3 with the known value and the number of hits.

5.3 Examination of a QS3P8 Quadruplet

The examination of real data was done on the QS3P8 quadruplet. It has been run through plethora of initial settings both at 2900V and 3100V. These data runs were also the primary sets used for testing and development of the analysis package, although others were used to test the functionality with different shapes of quadruplets.

Initial trials with this data shown in figure 7 reveals that the guess that the supports are straight may not be correct or that there is some external effects. Periodic effects are observed in many width and location plots but no statistics have been run to confirm their presence.

An explanation for the periodic effect comes from misalignment of the finite resolution strips (y-axis) and the support structures. Since the support is $\sim$ 7mm and the strips have a $\sim$ 3.2mm pitch, the support will go from perfectly covering 2 strips to overlapping with 3 and back again as it winds between the strip pattern. This effect would only create a disturbances in the results if the offset between the support and strips was large enough for the support to oscillate between these positions. Looking closely at the

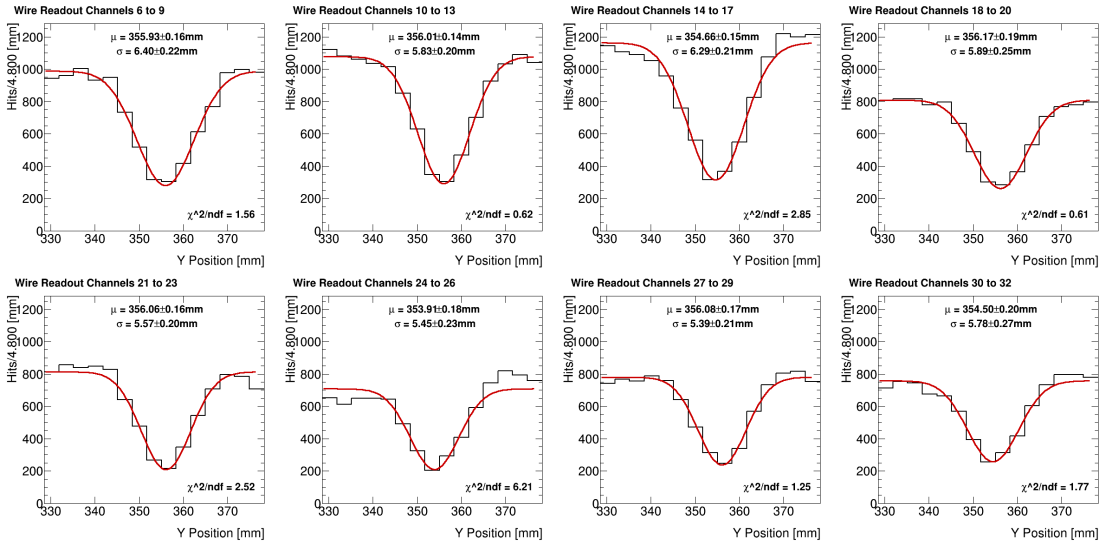

Figure 8: A sample of the best fits taken on real QS3P8 data with a strip width of 15 and 8 bins in the wire direction. The fits do perform well, but the bases of the first four and last four bins don't change shape in a way that suggests periodicity. Furthermore, plots 3 and 6 have considerable extra weight on one side of the support than another, creating a pulling effect to that side.

histogram fits in figure 8, no such oscillations in the width of the base of the dip seem present. Furthermore, this would imply that a straight support structure would have a large slope of $\sim$ 5mm/mm which is far within what a straight line support is capable of identifying. This rules out this hypothesis, but no other idea has been formulated to explain the periodicity.

Figure 8 examined by eye suggested having the histograms fit to too many strips may lead to non critical regions of the fits pulling the data left and right. The decision was made to then drop the width in hopes of reducing this "pulling" effect in the fits.

5.3.1 Stability

Here the analysis was done on both 3100V and 2900V data under different data binnings along the wireID (x-axis) direction to check if cutting the data into different bins had any effect on the results. Fortunately, figure 9 reveals that there is consistency between different binnings. This further lends credence that there is some external effect on the data that is not accounted for in this analysis.

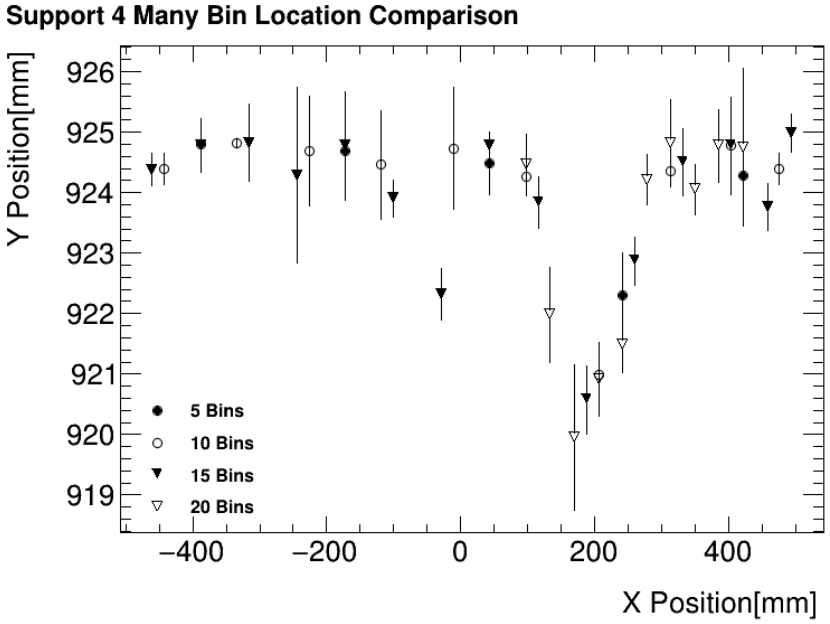

Figure 9: The best fit locations of the low efficiency zone cast by the support structures in a single data run analyzed with different bin counts. There is strong consistency between all the different binnings, indicating that the analysis is indeed stable.

5.3.2 2900 V vs 3100 V Results

A principle method used to validate the performance of the analysis was to cross check the results between 2900V and 3100V data. These do come from the same quadruplet, so naturally the results should be consistent. Figure 10 shows what initially was produced by the analysis. It displays inconsistencies on the order of a few millimeters, which is far beyond the statistical uncertainty expected at this volume of data. The widths of the 3100V data is consistently higher than that of 2900V data, which on its own invalidates the results obtained for the best fit locations. The current hypothesis as to the cause is different noise floors of 3100V and 2900V data.

5.3.3 ADC Count Cuts

To follow up the noise floor hypothesis, ADC count cuts were made to the minimum required of a recorded hit. This was to reduce the effects of the noise floor and "neighbor hits" where the system records the hit and the bin neighboring it. Figure 11 shows that there is a decrease in the best fit widths as the cuts in voltage get more aggressive. Cuts larger than 80 ADC counts for 2900V converge to a consistent width value. Furthermore, the reduction in widths hints that cuts in counts could make the 2900V and 3100V widths agree.

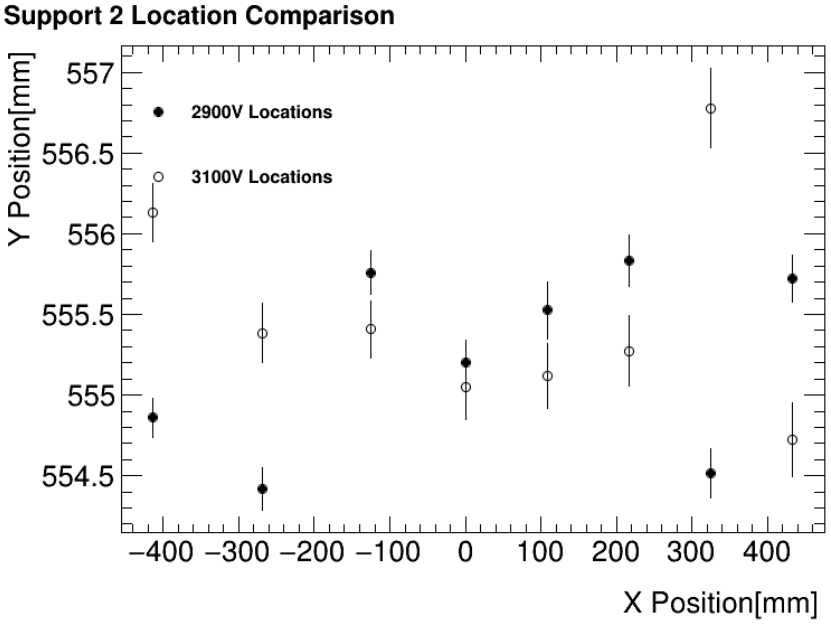

Figure 10: Comparison between the 2900V and 3100V data under the same analysis settings for a support on a QS3P8 quadruplet. There is quite wild disagreements as large as 3 mm and being on average $\sim 500\mu$m away.

On that note, 2900V and 3100V width comparisons with a 100 ADC counts has good agreement, and the location fits are in much greater agreement than earlier. Currently inconsistencies between the two types of runs are within a millimeter. This is clearly an important route forward, and there is more work to be done fine tuning the cuts needed and perhaps adjusting the fitting function to more closely match the new shape of the cut dips.

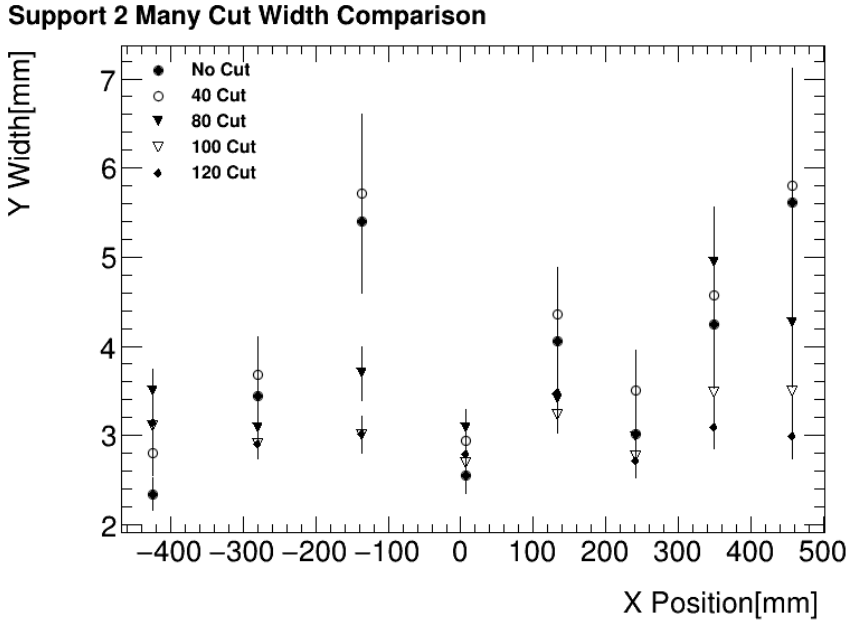

Figure 11: Comparisons between the same support on 2900V QS3P8 data for different cuts in the ADC counts. There is a clear segmentation after 80 where the voltages seem to converge to a result.

6 Conclusion

This novel technique for determining support locations is still in its adolescent stages. It is currently functional, but many of the patterns produced by it are not well understood. Once that point is reached, the stability of the results and the ever improving consistency between 2900V and 3100V data as well as the validation efforts demonstrate that it should be possible to detect some manufacturing imperfections. This process will be facilitated by working on data sets of other detectors in the future and checking if the same patterns manifest themselves or if these are unique to the QS3P8 analyzed here. Furthermore, with the introduction of cutting at the end of last section, there is a prompt to revisit the fitting functions and find one that may fit the cut shadow much better than the inverted Gaussian.

References

[1] “Facts and figures about the lhc.” [Online]. Available: https://home.cern/resources/faqs/facts-and-figures-about-lhc

[2] “The large hadron collider.” [Online]. Available: https://home.cern/science/accelerators/large-hadron-collider

[3] “The higgs boson.” [Online]. Available: https://home.cern/science/physics/higgs-boson

[4] “High luminosity lhc.” [Online]. Available: https://home.cern/science/accelerators/high-luminosity-lhc

[5] “Atlas detector technology.” [Online]. Available: https://atlas.cern/discover/detector

[6] B. Lefebvre, “Characterization studies of small-strip thingap chambers for the atlas upgrade.” [Online]. Available: http://cds.cern.ch/record/2633639/files/CERN-THESIS-2018-111.pdf