# A New Analytic Tool for Supercells in the LAr Calorimeter

Lana Glišić (Princeton University)
Supervisors: Sahibjeet Singh (University of Toronto),
Sully Billingsley (Southern Methodist University)

## 1. Introduction

The LAr calorimeter constitutes part of the ATLAS particle detector, which is located in a segment of the Large Hadron Collider at CERN. The calorimeter is a cylindrical device responsible for measuring the energies of particle collisions by measuring the energies of their products. The device can measure the energies of any electrons, photons, and hadrons produced by the collisions.

The calorimeter is composed of several layers of supercells, which are unit groups of sensors that are distributed radially and longitudinally throughout the calorimeter. Given an energy threshold and an integration time, these supercells provide the rate at which they measure energies that equal or exceed this threshold. For a functional supercell, an ideal noise-free graph of energy threshold versus rate would reveal a monotonically decreasing function whose maximum value roughly corresponds to the maximum collision rate in the LHC, and whose minimum value is zero past a sufficiently high threshold. A supercell whose behavior can be estimated by a monotonically decreasing function with these features will henceforth be referred to as "functional" or "healthy." This paper concerns only healthy supercells, as opposed to malfunctioning supercells, although studying the former kind of supercell may elucidate issues with the latter kind. These issues will be briefly discussed later in this introduction.

In the present day, CERN's computing power is insufficient to analyze all data generated by the supercells in the LAr calorimeter. Instead, each supercell is assigned a customized energy threshold to distinguish between interesting and uninteresting collisions. Data from interesting collisions is kept, and data from uninteresting collisions is discarded, thereby optimizing the use of computing power. Since low-energy collisions have already been reproduced and studied extensively, new discoveries are expected to take place at higher energies. For this reason, it is imperative that custom energy thresholds be assigned effectively so that interesting collisions are processed but computing power is conserved.

Empirical analysis reveals that for a given energy threshold, different supercells tend to measure different rates. This variation is a product of each supercell's size and its location in the calorimeter relative to collisions, as well as minute manufacturing differences between supercells. Determining the exact effect of these factors is difficult, and the calorimeter currently uses an algorithm to assign customized energy thresholds to individual supercells based on the standard deviation of their measured rates.

Despite the existence of this algorithm and perhaps partly because of it, supercell behavior remains an understudied phenomenon. It is still unknown which monotonically decreasing function describes supercell rate measurements as a function of energy threshold. Furthermore, understanding healthy supercell behavior may help refine their algorithmically assigned customized energy thresholds and explain certain erratic behaviors exhibited by malfunctioning supercells. Malfunctioning supercells constantly output significantly higher rate measurements than other supercells and consume large amounts of computing power, but provide erroneous data. These faulty supercells are "masked," or manually excluded from data analysis. Due to the irradiation of the LHC, masked supercells can only be safely repaired every few years during an LHC shutdown. It is currently unclear why supercells malfunction and how exactly their behavior differs from that of functional supercells. However, understanding healthy supercell behavior may eventually foster a successful investigation into the behavior of malfunctioning supercells, too. For this reason, fitting the appropriate function to data from healthy supercells represents an important first step to improving the measurement capabilities of the LAr calorimeter.

To better understand healthy supercell behavior, it is necessary to create a new analytic tool that is easily accessible to researchers at CERN. Such a tool should display data collected from functional

supercells and provide explanatory fitting and statistical analysis. The aim of this project is to build this tool as a webpage housed within the existing LArSoup website, a website that allows both real-time (online) and retrospective (offline) supercell data analysis. The webpage will rely on offline data to create informative graphs for use by CERN researchers. The following paper will describe the unique features and implementation of this new webpage.

## 2. Language, Framework, and Libraries

The webpage is programmed in Python and uses the Dash framework so that it can integrate with the rest of the Dash-based LArSoup website. The webpage queries offline data from CERN's P-BEAST data storage system. It uses the Scipy library for curve fitting and statistical analysis, and the Plotly library for visualizing graphs.

## 3. Data Collection

Raw supercell data is obtained by running an existing script during a time when there is a stable beam in the LHC. This encourages data consistency across multiple data collection periods. As the script steps sequentially through energy thresholds, the rates measured by the supercells are stored in P-BEAST. After the script stops running, these rates can be queried from P-BEAST using the appropriate starting and ending timestamps.

The script's stepping mechanism encompasses supercell energy thresholds in a range of 0.25 GeV to 10.25 GeV, rising in increments of 0.0125 GeV and yielding 800 thresholds total. These thresholds are stored in P-BEAST in units of counts. Stepping once through all 800 counts with an integration time of 5 seconds takes approximately eight hours. For the purposes of this project, integration times of 2 seconds and 5 seconds were used during different data collection periods.

Modification of the data collection script allows for the selection of N energy thresholds from the available range, thereby shortening the data collection period. Stepping through these N energy thresholds multiple times yields multiple "index numbers," such that for some data collection period, all data points belonging to some index number X represent the Xth rate measurement from the selected energy thresholds. Despite temporal costs, obtaining multiple index numbers per data collection period is preferable because data noise makes statistical analysis of a single index number unreliable. Furthermore, unknown factors may also make combining index numbers across data collection periods unreliable.

Another consideration besides data noise is buffer overflow. Buffer overflow impedes data collection at low energy thresholds. Collisions in the LHC occur every 25 nanoseconds during a stable beam, meaning that for low energy thresholds, a rate of approximately 40 MHz is expected. However, across all energy thresholds, the maximum rate retrieved from the supercells with an integration time of 2 seconds is 262,143 Hz, which corresponds to the overflow of an 18-bit binary number. Similarly, the maximum rate retrieved from the supercells with an integration time of 5 seconds is 209,715 Hz, which corresponds to the overflow of a 20-bit binary number. These hardware limitations make measurements at low energy thresholds inaccurate, which further complicates statistical analysis. Complications due to noise and buffer overflow are addressed via fitting and smoothing methods discussed later in this paper.

## 4. Summary of Webpage Features

The top of the webpage contains three drop-down lists that allow the user to select the ID of the desired supercell, the timestamp of the desired data collection period, and the desired index number as shown in Figure 1a. The rest of the webpage is occupied by three Plotly graphs stacked on top of each other that refresh according to the selected values from the drop-down lists. The first of these graphs displays fitted raw data for the selected index number, supercell, and data collection period. The second graph displays this data's corresponding fitted cumulative density function. The third graph displays this data's corresponding fitted probability density function. Examples of these three types of graphs are shown in Figures 1b, 1c, and 1d, respectively. For all three graphs, Plotly's inbuilt functionality allows the user to show or hide the displayed data and its fits as desired by clicking on the entries in the legend. Scrolling over a scatterplot point triggers a pop-up label with the point's x- and y- values. Meanwhile,

scrolling over a fitted function reveals its x- and y- coordinates as well as its chi squared value. In the case of the fitted probability density function, the mean and standard deviation are also shown in the pop-up label.

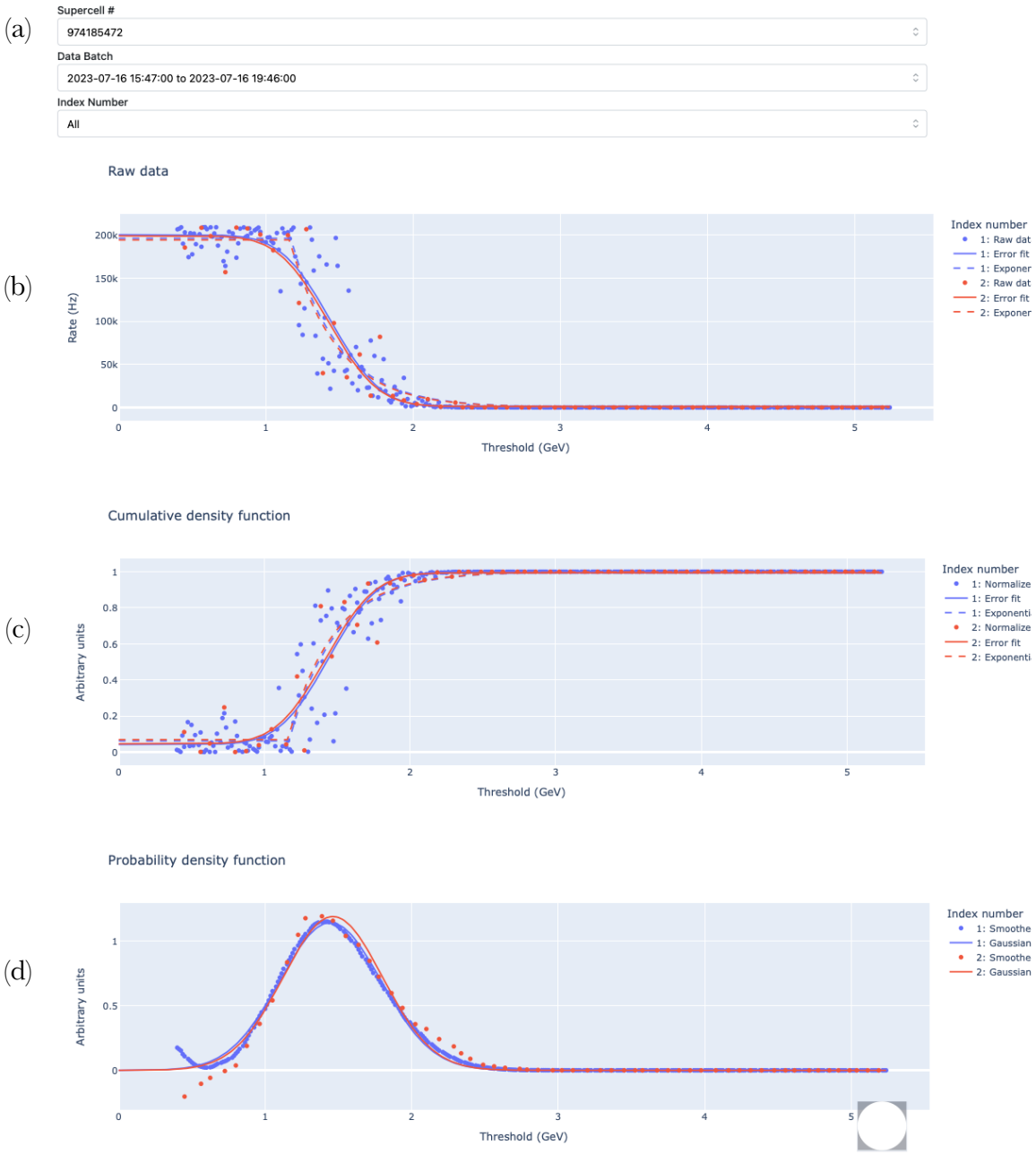


**Figure 1:** The webpage displaying graphs for two index numbers taken from the supercell with ID 974185472 during a data collection period on July 16th, 2023: (a) drop-down menu for user's selection of supercell, data collection period, and index number, (b) raw data with fitted functions, (c) cumulative density function, and (d) probability density function.

## 5. Raw Data Display

The first graph displayed on the webpage contains the raw data for the selected index number, supercell, and data collection period, as shown in Figure 1b. Its x-axis represents the energy threshold in

giga electron volts and its y-axis represents the rate in hertz. The raw data from the supercells is displayed on the graph in scatterplot format.

Two fits are provided on the raw data, the first of which is a Gaussian error function. If the rate measured by a supercell is described as a normally distributed random variable dependent on energy threshold, then a Gaussian error function describes the distribution of rates measured by a supercell across a range of energy thresholds. While a Gaussian error function may not be truly descriptive of rate as a function of energy threshold, its parameters can be adjusted such that the function fits the raw data and is monotonically decreasing. As seen in the Introduction, a monotonically decreasing function aligns with the theoretical description of a healthy supercell. It also provides a controlled slope from maximum to minimum value which, based on initial observation, closely resembles that of the raw data. Furthermore, a Gaussian error function begets a mean and standard deviation that can help CERN researchers better understand the properties of healthy supercell behavior. By studying the mean and standard deviation obtained via a Gaussian error function, the researchers may find themselves better equipped to discover a new function that describes the data more accurately. For these reasons, a Gaussian error function was selected for the first fit.

Fitting to the Gaussian error function is performed with least-squares using Scipy's curve_fit method. As discussed in Data Collection, buffer overflow impedes data collection for low energy thresholds. This means that any fits performed with a Gaussian error function are inaccurate at all x-values that are lower than some critical x-value. At this critical x-value, the measured rate becomes low enough to be representable without overflow. Since this critical x-value is unknown, the current fitting method includes all of the raw data points. Consequently, the resulting fit tends to have a maximum value approximately equivalent to the maximum value of the dataset, or the buffer overflow limit. While fitting to the buffer overflow limit for low energy thresholds is likely a significant cause of error, it may not be the case that all supercells would output 40 MHz for the lowest energy thresholds if buffer overflow were not present. Due to this uncertainty, as well as uncertainty about the x-value at which buffer overflow ceases to interfere with data, the final fit includes all data points and does not attempt to artificially increase the amplitude of the Gaussian error function. Although the mean and standard deviation obtained via this fit do not necessarily reflect true, overflow-free supercell measurements, they are nevertheless effective at describing data patterns across supercells that are known to be uniformly affected by buffer overflow.

Besides the Gaussian error function, the second fit provided on the raw data is a piecewise exponential function. This function seeks to better describe the gradually sloping behavior of the raw data, which resembles exponential decay across a majority of supercells. It also attempts to describe the presence of the buffer overflow limit for x-values lower than some critical x-value. The function does so by assigning a constant y-value to all x-values smaller than the critical x-value, and an exponential decay function to all x-values greater than or equal to the critical x-value. An ideal fit with this function would assign the buffer overflow limit to all x-values smaller than the critical x-value. Finally, in accordance with the theorized behavior of a functional supercell, this piecewise exponential function is monotonically decreasing. While this function is not used to calculate a mean and standard deviation, it provides a secondary interpretation of the data outside of the Gaussian error function. This is important because the fitted Gaussian error function does not account for the interference of buffer overflow and assumes that rate is a normally distributed random variable. Instead, the piecewise exponential fit reflects the reality of buffer overflow and brings attention to the exponentially decaying rate measurements exhibited by many supercells.

**6. Cumulative Density Function Display**

The cumulative density function display is closely related to the raw data display, as shown in Figure 1c. In fact, the cumulative density function is obtained by calculating one-minus the normalized raw data. Normalization is accomplished by dividing each raw data point by the maximum value of the raw data. Then, each normalized data point is subtracted from 1.00. The same sequence of transformations is applied to the fitted Gaussian error function and piecewise exponential function. The y-values of each fitted function are divided by the maximum value of the raw data, then subtracted from

1.00. The resulting graph is visually similar to the original raw data graph, but serves as an intermediate graph between the raw data and probability density function. It makes apparent the state of the data just before the probability density function is derived, allowing the user to immediately compare the slope of the cumulative density function with the resulting peak in the probability density function that is displayed immediately below.

**7. Probability Density Function Display**

As discussed in Raw Data Display, the fitted Gaussian error function assumes that the rate is a normal random variable dependent on energy threshold. In accordance with this, the probability density function is fitted to a regular Gaussian function, as shown in Figure 1d. Due to the extreme noisiness of the raw data observed with both a 2 second and 5 second integration time, the first derivative of the raw data jumps between large positive and negative values. Consequently, smoothing is applied to the cumulative density function before calculating the derivative. The smoothing method is a moving average that uses approximately 7% of the surrounding data points, giving equal weight to the averages obtained from a data point's left-hand and right-hand neighbors. The data points obtained from smoothing are stored in a new list. The algorithm then calculates a smoothed derivative of this new list by using 7% of surrounding data points in the new list. The algorithm subtracts the averages of each data point's left-hand and right-hand neighbors and divides this value by the average x-values of its left-hand and right-hand neighbors. The resulting probability density function is significantly smoother than the first derivative of the raw data, producing mostly nonnegative values, and enables a Gaussian fit. In its current state, the algorithm takes O(n) time in the number of data points. Although other smoothing algorithms may be employed, this algorithm adapts effectively to the varying number of N-values for index numbers across different data collection periods. After smoothing, the fitted Gaussian function yields a mean and standard deviation. These values are displayed in a pop-up label that the user triggers by scrolling over the plot of the Gaussian function on the webpage.

**8. Conclusion and Future Improvements**

For a user-selected index number, supercell, and data collection period, the webpage displays the corresponding raw data, cumulative density function, and probability density function with accompanying fits and statistical analysis. Furthermore, the fitting and smoothing algorithms are robust and able to withstand both sparse and noisy data. In addition to their usefulness in mathematical analysis, the function fits generated by these algorithms provide intuitive visual guides for CERN researchers studying the behavior of healthy supercells.

In the future, numerous improvements could be made to this webpage. First, providing error bars on the raw data would enable the user to visually determine the confidence of various function fits. Error bars may also eliminate the need for smoothing when fitting to the probability density function. Second, fitting to a regular exponential function as opposed to a piecewise exponential function would clarify the difference between measurements that are affected by buffer overflow and those that are not. Finally, fully integrating the webpage into the LArSoup website would eliminate the need for a drop-down list of supercell IDs. Inputting the desired ID via the LArSoup homepage could redirect the user to a webpage dedicated solely to the desired supercell.

**9. Acknowledgements**

I would like to give special thanks to my supervisors, Sahibjeet Singh and Sully Billingsley, for their mentorship throughout this project. I would also like to thank Huacheng Cai and Youssef Tawfik for their work in parallel with this project and development of the data collection script. In addition to this, I would like to thank the LAr calorimeter team for their guidance and kindness. I would like to thank CERN and the Princeton International Internship Program for the opportunity to work on this summer project. Finally, I would like to thank my friends at CERN for making my stay especially wonderful.