FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF BACHELOR'S THESIS

Title:	Optimization of the Matching Criteria Between the ATLAS and AFP Detectors at CERN
Student:	Petr Dostál
Supervisor:	doc. Dr. André Sopczak
Study Programme:	Informatics
Study Branch:	Web and Software Engineering
Department:	Department of Software Engineering
Validity:	Until the end of summer semester 2020/21

Instructions

The ATLAS Forward Proton (AFP) detector took large scale data in 2017 together with the ATLAS central detector at CERN. The goal of this project is to optimize the matching requirement between the two detectors.
The first task is to familiarize with the existing analysis software and the literature regarding two-photon interactions. A software development plan should be created with the focus to improve the current basic matching criteria. The uncertainties on the ATLAS central and AFP measurements should be taken into account in the matching optimization. The performance increase of the optimized matching should be compared with the performance of an existing simple matching criteria. The developed code should first be tested on a small event sample, and then applied to the whole data-set which requires the use of an user interface to grid computing. As a bonus, the method should also be tested on simulated data.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague November 6, 2019

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Optimization of the Matching Criteria Between the ATLAS and AFP Detectors at CERN

Petr Dostál

Department of Software Engineering
Supervisor: doc. Dr. André Sopczak

July 30, 2020

Acknowledgements

I am very thankful to my supervisor, doc. Dr André Sopczak, for the patience he had with me and the opportunity to be a part of a real research. Also I would like to thank my friends and my family for the continuous support in my studies.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on July 30, 2020 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Dostál, Petr. *Optimization of the Matching Criteria Between the ATLAS and AFP Detectors at CERN*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

Abstract

This thesis is a part of CERN's ongoing research of the search for axion-like-particle (ALP). It consists of a software to modify existing data to include calculated uncertainties on photon and proton measurements and a software that calculates the difference between corellated and uncorellated data. Also included is a study on matching criteria for proton-proton interactions with two detected photons. The study discusses the performance of three different matching criteria regarding the kinematics of the di-photon system with at least one detected proton. The created software fulfils all stated requirements, including execution time requirement. The main discovery from the physics point of view is that the interactions studied in the data are caused by pile-up background, thus they are random interactions with no indication of a specific physics process, which is consistent with previous simulations.

Keywords CERN, LHC, ATLAS, AFP, axion-like-particle, ROOT, data analysis, matching criteria

Abstrakt

Tato bakalářská práce je součástí výzkumu CERNu, který se snaží najít axion-like-particle (ALP). Skládá se ze softwaru, který upravuje existující data tak, aby obsahovala vypočítané odchylky (chyby) pro měření protonů i fotonů. Také obsahuje studii přiřazovacích kritérií pro proton-proton interakce se dvěmi detekovanými fotony. Studie pojednává výkon tří různých přiřazovacích kritérií ohledně kinematiky dvoj-fotonového systému s alespoň jedním detekovaným protonem. Vytvořený software splňuje všechny zadané požadavky, včetně časové složitosti. Hlavním objevem práce z fyzikálního pohledu je zjištění, že pozorované interakce vznikají díky pile-up backgroundu, tedy se jedná o náhodné interakce bez indikace fyzikálního procesu, což je konzistentní s předchozími simulacemi.

Klíčová slova CERN, LHC, ATLAS, AFP, axion-like-particle, ROOT, analýza dat, přiřazovací kritéria

Contents

List of Figures

List of Tables

Introduction

In 2017, CERN's ATLAS Central and ATLAS Forward Proton (AFP) detectors took large scale data including proton-proton interactions data. Since then, in 2019 ATLAS observed light-by-light (LbyL) scattering [1, 2, 3, 4] (a rare interaction where two highly energetic photons produce another pair of photons) by performing an analysis focused on lead-lead collisions with exactly two photons present in the system. Light-by-light scattering could also be observed in high energy proton-proton collisions, but the occurrence is very rare.

However there is a theory of existence of an axion-like-particle (ALP) [5] which could increase the occurrence rate over the expected amount coming from the Standard Model (SM) of particle physics. This thesis is a part of ongoing research that explores this ALP mediated LbyL scattering. The specific investigation is the matching of kinematic properties of the di-photon system with at least one proton detected.

There has already been an analysis [6] that serves as a proof of concept that the AFP can be used together with ATLAS Central Detector data. On the 30th of July, 2020, the ATLAS Collaboration announced at the ICHEP 2020 conference the first results of matching AFP data and ATLAS Central data using di-leptons [7]. As a former Experimental Nuclear and Particle Physics student, the opportunity to be a part of this research has been great.

The main goal of this thesis is to create software that would modify existing processed data to include systematic uncertainties and that would calculate the difference between nominal matching and matching done with uncorrelated data. Also a secondary goal of the thesis is to analyze and optimize the matching criteria in proton-proton interactions with exactly two outgoing photons observed. This analysis will provide clarity to the randomness coming from unwanted reactions to enhance the clarity of the signal coming from LbyL scattering.

Outline The thesis is organized into several chapters starting with the theoretical background and the physics needed for this thesis in Chapter 1.The software development is discussed in Chapter 2 and the observed results, both from software point of view and physics point of view, are discussed in Chapter 3.

Chapter 1

Physics framework

This chapter starts with the introduction to CERN, The ATLAS Experiment and the proton-proton interaction with ALP search The following sections define the matching and the compared matching criteria. Next are the sections that study the photon and proton systematic uncertainties. Lastly, the proton selection and matching background are explained.

1.1 CERN

CERN [8], also known as the European Organization for Nuclear Research is one of the most important contributors to particle research. The acronym CERN (Conseil Européen pour la Recherche Nucléaire) is also used for the largest particle physics laboratory in the world that is a part of the organization. Another notable information is that CERN is the birthplace of World Wide Web [9].

A very important part of the laboratory is the Large Hadron Collider (LHC) [10, 11]. As of now it is the largest particle collider in the world. It also holds the world record for the highest total collision energy at 13 TeV. The LHC consists of four major experiments – the ATLAS experiment, the ALICE experiment, the LHCb experiment and the CMS experiment. Each experiment is situated at a different part of LHC and has its own set of detectors (Figure 1.1).

1.2 The ATLAS experiment

A Toroidal LHC ApparatuS (ATLAS) is the largest experiment of LHC at CERN [13, 14, 15]. It is also the biggest set of general purpose particle detectors.

The ATLAS central detector (Figure 1.2) is a layered multipurpose particle detector. The layers are designed to increase the spectrum of particles that

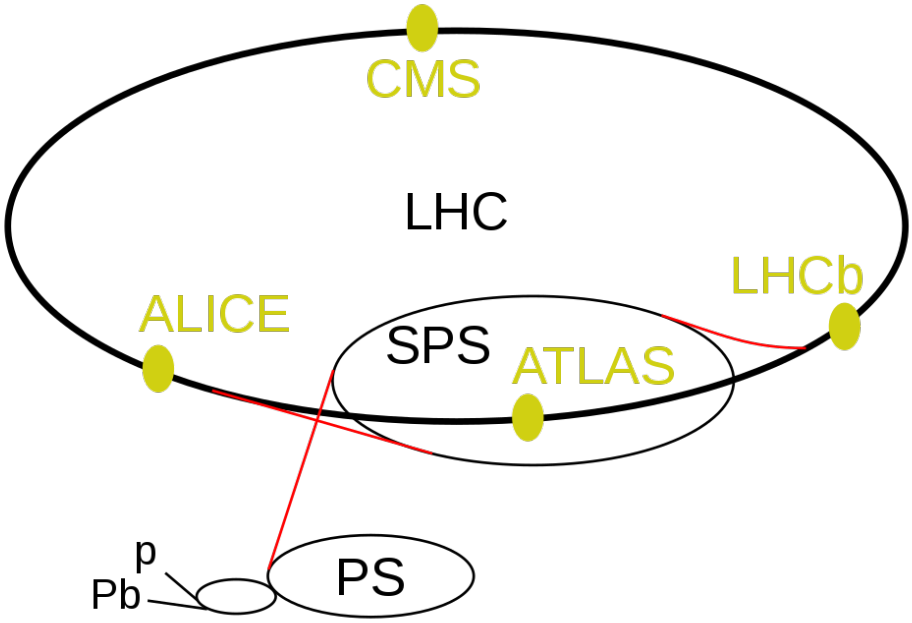

Figure 1.1: The LHC experiments and the preaccelerators. Source: [12]

can be identified. Each layer detects a specific region of pseudorapidity η of the particles. The main layers are the inner detector, the calorimeters, the muon spectrometer and the magnet system. Also a part of the ATLAS detector is the ATLAS Forward Proton (AFP) detector.

The AFP [16] is a set of pixel sensors located at around 210 m on each side of the beam from the interaction point (IP). The main goal is to detect protons that lose a fraction of their energy during an interaction in the IP (for example by photon emission). The internal structure are two stations, referred to as NEAR and FAR stations (distance from IP $\approx$ 205 m and $\approx$ 217 m, respectively). Each station consists of four 3D silicon pixel sensors that measure the trajectory of the passing protons.

1.3 Proton-proton interaction and ALP Search

In theory, when the two very energetic protons get very close together, three options can occur. Both of them remain intact, one of them remains intact (the other one is destroyed) or neither of them remain intact (both protons are destroyed). During standard proton-proton collisions the protons are destroyed. In this research the detected protons are expected stay intact. If they stay intact, the expectation is that during the interaction the protons lose some of their energy, which alters their direction according to the energy

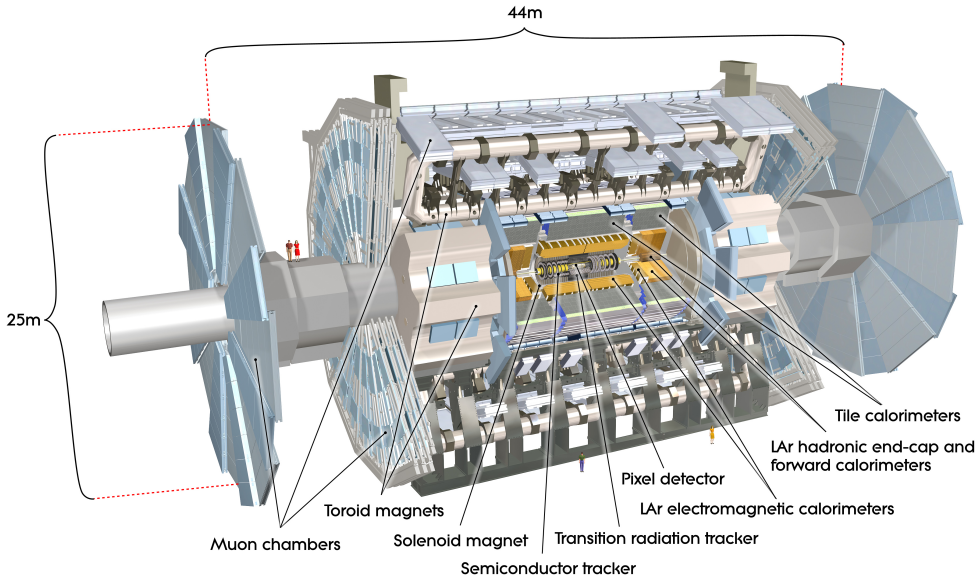

Figure 1.2: Computer generated image of the whole ATLAS detector. Source: [17]

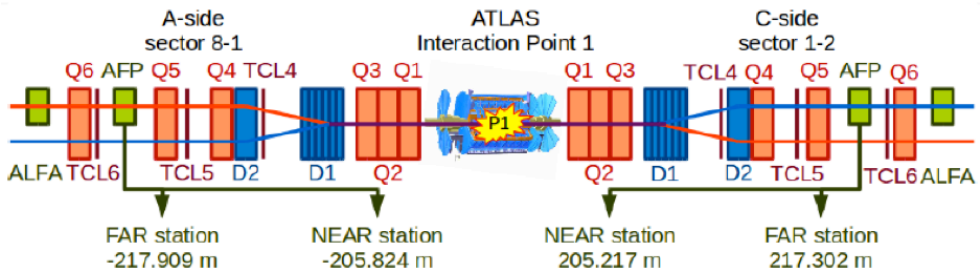

Figure 1.3: ATLAS central and AFP detectors. Q1–Q6 are magnets that focus the beam, D1–D2 are dipoles that bend the beam. Source: [18]

lost.

The direction change is caused by a set of magnets that affects particles with different electromagnetic fields differently. This electromagnetic field is directly related to the energy of the particle, thus by extension the energy is directly related to the path the protons take.

According to the law of conservation of energy the energy must be transferred or transformed. This analysis expects that during the interactions exactly two photons are created from the energy lost by the protons. In case a different amount of photons is detected in the central detector, the event is filtered out. This allows to reduce the unwanted background in the search for an axion-like-particle (ALP).

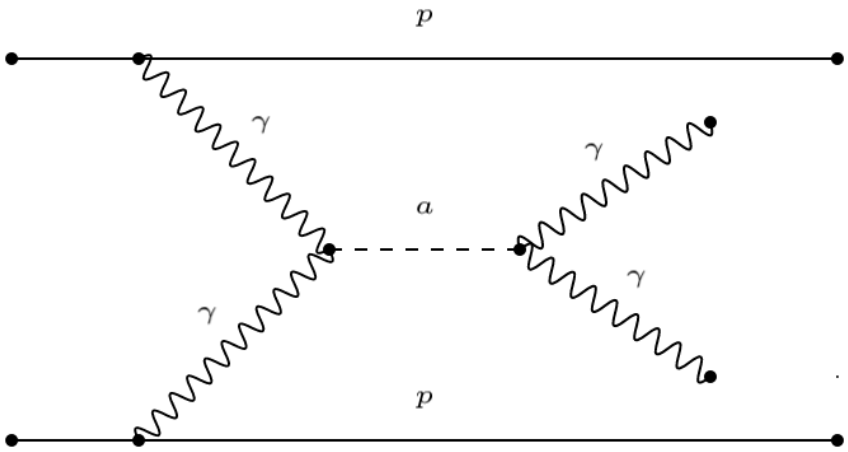

Figure 1.4: Feynman diagram illustrating light-by-light scattering mediated by an ALP (a). Source: [18]

The Feynman diagram (Figure 1.4) illustrates that when the protons get very close to each other, two photons are emitted. These photons fuse to create an ALP that shortly after decays into a pair of photons. These photons could be detected in the ATLAS central detector. The problem with the diagram is that it is not self-explanatory. It can be compared to a state diagram of particles present. Time flows from left side of the diagram to the right side. In the beginning, on the left side, two protons exist. Next they nearly collide and two photons are emitted. In the next step the photons collide and fuse, creating the ALP particle. The ALP particle quickly decays into two photons and in the final state two photons and two protons are in the system. The diagram does not suggest the details the interaction between protons or photons. In reality the protons fly against one another to achieve the electric charge needed to create the two photons, even though it is not stated by the diagram.

After the protons nearly collide and their direction is altered, there is a possibility that the protons are detected by the AFP detector. The chance of detection is determined by the amount of energy lost by the protons. The two protons do not have to lose the same amount of energy during the interaction. On the other hand, the ATLAS central detector detects all photons in the system.

This thesis is focused on pairing the detected kinematic properties (their energy) from the ATLAS central detector (detected photons) to the data from

the AFP (detected protons) to see if they match, which is called the matching. The analysis does not include only measured data, it also includes simulated data.

To obtain the simulated data a particle generator (in this case SuperChic3 [19]) is used to generate truth particles. Next, this generated data goes through a simulation process to obtain the simulated data which models the ATLAS detector. The simulation process is different for photons and protons. For photons, which are detected by ATLAS central detector, there is a dedicated group that simulates the detector. The protons, detected by AFP detector, do not have a dedicated group, but a tool called `AFPFastSim` exists to obtain the simulated data for protons.

There are many measured variables by the detectors, in order to reduce it there are so called "derivations". There are numerous derivations, each containing a different list of variables that suites the needs of the groups using them. In this analysis, STDM2 (Standard Model 2) derivation is used. The notable part about STDM2 is the inclusion of proton data measured by the AFP detector.

1.4 Matching

Matching is the pairing between the kinematics between detected protons and detected photons. To be precise, we are looking at relative proton energy loss ξ. The definitions for ξ_{AFP} (protons) and $\xi_{\gamma\gamma}$ (photons) are different (Equations 1.1, 1.2, respectively [18]). By definition both the ξ_{AFP} and $\xi_{\gamma\gamma}$ values are in interval the $\langle 0; 1\rangle$. The upper limit for $\xi_{\gamma\gamma}$ is a result of additional constraints on $m_{\gamma\gamma}$ and η coming from the law of conservation of energy.

$$\xi_{\mathrm{AFP}} = 1 - \frac{E}{\mathrm{E}_{\mathrm{beam}}} \tag{1.1}$$

$$\xi_{\gamma\gamma} = m_{\gamma\gamma} \cdot \frac{\mathrm{e}^{\pm\eta}}{2E_{\mathrm{beam}}} \tag{1.2}$$

A short explanation of the symbols used in Equations 1.1, 1.2:

- E – Energy of the proton measured by the AFP detector
- $m_{\gamma\gamma}$ – Di-photon invariant mass
- η – Di-photon pseudorapidity, defined as $\eta = -\ln\tan\frac{\theta}{2}$, where θ is the polar angle in relation to the beam
- E_{beam} – Energy of the beam (a constant value, 6.5 TeV) [20]

Since there are two independent sides of the beam (side A, side C), the matching is done for both sides separately. Without the loss of generality it

has been decided that in Equation 1.2, side A corresponds to positive η, side C corresponds to negative η.

1.5 Matching criteria

It is natural that there cannot be an expectation that the compared energy loss would be exactly the same for protons and photons. This is where the matching criteria comes in. The goal is to get the highest possible amount of matches with the lowest possible amount of false positives. The optimization of the matching criteria refers to the inclusion of systematic uncertainties on the photon and proton measurements.

The first matching criteria, also later referred to as standard (Equation 1.3), is a naive approximation that the relative difference of energy losses should be less than 10%. $\Delta\xi$ is defined as the absolute difference between proton and photon ξ ($\Delta\xi = \xi_{\text{AFP}} - \xi_{\gamma\gamma}$). Left side of the equation is divided by $\xi_{\gamma\gamma}$, it is more precise when compared to ξ_{AFP}.

$$\frac{|\Delta\xi|}{\xi_{\gamma\gamma}} < 10\% \tag{1.3}$$

The next matching criteria, labeled as 1σ matching (Equation 1.4), takes into account uncertainties on the measurements. $\sigma_{\xi_{\text{AFP}}}, \sigma_{\xi_{\gamma\gamma}}$ represent the absolute uncertainty on ξ_{AFP} and $\xi_{\gamma\gamma}$, respectively. From the theoretical point of view the values should be the same, by extension the difference between the detected numbers should be lower than the uncertainties on the measurements.

$$\Delta\xi < \sigma_{\xi_{\text{AFP}}} + \sigma_{\xi_{\gamma\gamma}} \tag{1.4}$$

There is one more matching criteria that will be taken into account, labeled as 2σ matching (Equation 1.5). It is included because of the fact that in reality there will be some energy lost from other sources than the interaction itself. But the amount of energy lost this way should be fractional compared to the total energy. To account for this additional energy loss, the threshold for match when taking uncertainties into account is doubled. By definition, $\approx 68.3\%$ of the matches should be detected by 1σ constraint and $\approx 95.5\%$ of the matches by the 2σ constraint.

$$\Delta\xi < 2\left(\sigma_{\xi_{\text{AFP}}} + \sigma_{\xi_{\gamma\gamma}}\right) \tag{1.5}$$

1.6 Photon systematic uncertainties

The determination process of $\sigma_{\xi_{\gamma\gamma}}$ is to partially derive Equation 1.2 by its variables (in this case $m_{\gamma\gamma}$ and η). It is known that the angle is measured very precisely (the uncertainty on the angle is very small), we can approximate that

there is no uncertainty on the angle. With this assumption we are left with Equation 1.6.

$$\sigma_{\xi_{\gamma\gamma}} = \sigma_{m_{\gamma\gamma}} \cdot \frac{\mathrm{e}^{\pm\eta}}{2E_{\mathrm{beam}}} \tag{1.6}$$

In order to determine $\sigma_{m_{\gamma\gamma}}$ (the absolute uncertainty on di-photon invariant mass), because $m_{\gamma\gamma}$ is calculated and not measured, we must mention how it is calculated (Equation 1.7).

$$m_{\gamma\gamma} = \sqrt{2E_1E_2(1-\cos\alpha)} \tag{1.7}$$

In this case α is the angle between the two photons. And again it is measured very precisely so we neglect the uncertainty on it. Again we partially derive by variables and add the results in a quadrature sum in order to determine the equation for $\sigma_{m_{\gamma\gamma}}$ (Equation 1.8).

$$\sigma_{m_{\gamma\gamma}} = \frac{m_{\gamma\gamma}}{2}\sqrt{\left(\frac{\sigma_{E_1}}{E_1}\right)^2 + \left(\frac{\sigma_{E_2}}{E_2}\right)^2} = \frac{m_{\gamma\gamma}}{2} \cdot \left(\frac{\sigma_{E_1}}{E_1} \oplus \frac{\sigma_{E_2}}{E_2}\right) \tag{1.8}$$

There has already been a study that mentions photon energy resolution uncertainty [21], the Equation 1.9 is taken directly from it.

$$\frac{\sigma}{E} = \frac{a}{\sqrt{E}} \oplus \frac{b}{E} \oplus c \tag{1.9}$$

In Equation 1.9, the listed variables are:

- a – Sampling term - $9-10\%$ (GeV) (10% is used)
- b – Noise term - $350\cosh\eta$ (MeV)
- c – Constant term - 0.7%
- E – Measured energy of the photon
- η – Single photon pseudorapidity

1.7 Proton systematic uncertainties

The already mentioned di-lepton study [6] includes a chapter with AFP systematic uncertainties.

Because proton energy is measured using the trajectory of the protons, the uncertainties are related to the position on the x-axis on the detector (with the exception of beam optics).

- Global alignment – $300\,\mu$m
- Beam optics – $50\,\mu$rad

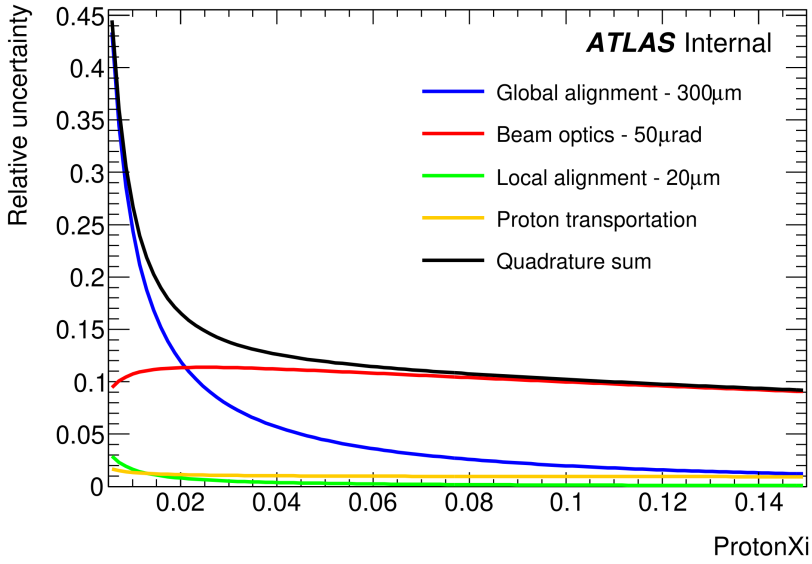

Figure 1.5: Relative proton uncertainties that are taken into account with their quadrature sum (total proton ξ uncertainty).

- Local alignment – 20 μm
- Proton transportation – 2%

In order to calculate the $\sigma_{\xi_{\text{AFP}}}$ there is an approximation that calculates ξ from the x-axis position on the detector (Equation 1.10) [22]. σ_x is defined as the uncertainty for the x-axis position. For the beam optics and the proton transportation the $\sigma_x(\xi)$ is shown in Equations 1.11, 1.12, respectively [6].

$$\sigma_\xi = \frac{\sigma_x}{-119 - 328\xi} \tag{1.10}$$

$$\sigma_{x_{\text{beam}}} = -0.02 + 15.38\xi \tag{1.11}$$

$$\sigma_{x_{\text{transport}}} = 0.00508 + 1.104\xi + 2.834\xi^2 \tag{1.12}$$

The total uncertainty $\sigma_{\xi_{\text{AFP}}}$ is calculated by quadrature sum of all its parts (Equation 1.13, Figure 1.5).

$$\sigma_{\xi_{\text{AFP}}} = \sigma_{\xi_{\text{global}}} \oplus \sigma_{\xi_{\text{beam}}} \oplus \sigma_{\xi_{\text{local}}} \oplus \sigma_{\xi_{\text{transport}}} \tag{1.13}$$

1.8 Proton selection

There is a requirement for the analyzed events to have exactly two photons. For protons there is a requirement that at least one proton has to be detected in the system. This leads to a large amount of events where there might not be a single proton or there might be more than one proton detected on either side of the beam. More than one proton on one side is possible, because the interactions are not one proton to one proton, but a bunch of protons into a bunch of protons.

The case of zero protons on either side is solved by not-matching the side to a photon. But where there are more protons only one can be chosen as a candidate to be matched.

There are 3 main strategies – choosing the proton with the lowest measured energy, the highest measured energy or the proton, which has the most similar energy loss to the compared photon. Early in this analysis the proton with highest energy loss was chosen, but since then it was found out that the highest efficiency is when a proton with the closest energy loss to the photon was selected.

1.9 Matching background

The matching background is the randomness of the matching. Because the events are uncorrelated, there are two main strategies – one is to take proton data from a different event and try to match it to the photons from current event (later referred to as "mixed case"), the other is to switch sides when matching (matching side A proton to side C photon and vice versa, later referred to as "switched sides").

After matching, the similarities or differences between the number of matches of nominal matching and mixed matching can serve as a background model. This model shows how many matches from the number of nominal matches are a random coincidence.

In order to eliminate any unwanted correlation the proton selection must be done right before matching (after mixing events or switching sides).

For the mixed case there is a statistics that can be easily made. The results should, by definition, be similar when taking proton information from event $n-1$, $n-2$, ...The number of matches in these can be fitted using a Gaussian function. The width of the Gaussian fit results in the uncertainty on the number of random matches.

In the implementation the statistics is not done using the number of matches, but using the difference between the number of nominal matches and the number of mixed matches. The width of the Gaussian is still the same in both cases.

At this time, the number of matches for all three matching criteria (Table 3.1) was known. As a result, the matching background was analyzed only for 2σ matching.

Chapter 2

Software development

In this chapter the development process is discussed. First the functional and non-functional requirements for the software are stated. This is followed by the analysis and design of the software. It follows a section with a discussion on the implementation. And lastly the software results are discussed.

2.1 Functional and non-functional requirements

The functional requirements are the correct calculation of systematic uncertainties for detected protons and photons, discussed in sections 1.6, 1.7. Another functional requirement is to have the possibility to change the proton selection (discussed in section 1.8) from "standard" to "mixed" case or "switched sides" case. An alternative option is to include the results of all three proton selections inside one ROOT file. Lastly another functional requirement is to calculate the difference of numbers of matches between "standard" proton selection and "mixed" proton selection for $n-1, \ldots, n-1000$.

The software should be split into two ROOT scripts, one calculating the uncertainties, the other to calculate the difference of numbers of matches. The most important non-functional requirement is that the software is to be run on CERN's lxplus servers. As a side note, it would be a plus if the code could be run on CERN's distributed grid network or locally. Another requirement is that the output file is a ROOT file. Also it is important that the software (or part of the software) that calculates the difference between the number of matches between "standard" and "mixed" proton selection does not exceed more than typically two hours, which relates to around 6 seconds per one iteration (measured on lxplus servers). One iteration is defined as calculating one difference between "standard" and "mixed" case. Because this software is a part of an ongoing research, it is important that the code is easily modifiable in case any parameter or calculations changes.

2.2 Analysis and design

In this section first the development strategy is discussed. It follows a short analysis on how the data is measured and processed further. Next, ROOT, the software used, and lxplus server cluster are briefly described. Lastly, the input and output file structures are analysed.

2.2.1 Development strategy

The complexity of the physics behind this software resulted in my decision to choose Iterative development over Waterfall or Agile. The main advantage of Iterative development is that the software is developed in incremental iterations where each iteration mainly includes additional functionality. In contrast, when using Agile development strategy, there are regular iterations that contain all functionality. And lastly the Waterfall strategy does not use iterations and is slow to adopt changes.

To be more specific about the reasoning behind the decision, I ruled out the Waterfall method, because during a research like this there can be new observations that result in a need to change parts of the code and the slow adoption of changes of the Waterfall method is not suitable for a project like this. The decision between Agile and Iterative was done purely according to personal preference. As I personally have very little experience with Agile, Iterative felt like the better choice.

2.2.2 From raw data to NTuples

Every time there are interactions in LHC, to identify the date and time of the interaction period, an unique identifier, `RunNumber`, is used. Each Run is split into one minute long blocks, referred to as `LumiBlock`. For each interaction the measured data is separated and to identify each interaction an identifier `EventNumber` is used. There can be multiple events with the same `EventNumber` across different `RunNumber`s.

First the measured data is saved into so called containers. The containers are collections of related variables, for example the photon container contains all measured variables related to photons. To filter the containers that are relevant for different groups, so called derivations are used. In the case of this research the used derivation is STDM2 (Standard model 2). The notable containers that STDM2 contains are the photon and AFP containers.

All the data is saved on the grid. But in order for scientists to work more efficiently, the data can be split into smaller fragments that contain a small fraction of the data can be transferred to lxplus servers or downloaded to local machines. These fragments are typically in the range of hundred megabytes to hundred gigabytes. One of the main purposes for these smaller fragments is to test software on smaller data compared to the whole data sets on the

grid. The execution time of software that processes the data for the whole data sets on the grid can typically be a few days.

The processing software for this research is the Group NTuple Production code. Its input is the STDM2 derivation. It can process both real and simulated data. The main task of the code is to filter events in which were detected exactly two photons. Another important filtering is done at this stage as well. It can sometimes happen that some detectors are malfunctioning. To remove these cases a Good Run List (GRL) is applied and only the `RunNumber`s in the GRL are processed. Each research group uses a different one since each group needs data from different detectors. In order to access the data in the containers it uses a so called `AFPToolbox` framework.

2.2.3 ROOT

ROOT [23, 24] is a multi-platform software developed by CERN with intended use of big-data storage, analysis and visualisation. The main applications are plotting and storing complex or high amounts of data.

The storage files are self-descriptive and compressed binary files. One of the main benefits of ROOT's storage solution is that one file can be split into several smaller files that are chained and accessed as a single object.an A simple file containing several entries for variables calso be called a ntuple.

The main way to control ROOT is using a command line interface (CLI). By default it uses a C++ interpreter Cling, but there is also an option to have an interactive session using a Python interpreter. This allows the creation of very complex scripts with custom classes and structures.

Creating scripts and interpreting them using the CLI is not the only way to interact with ROOT files. Because interpreting scripts can be time ineffective, there is also an option to compile the scripts or compile them as separate applications [25].

For visualization there is also a simple graphic user interface (GUI). The basic window is the `TBrowser` (Figure 2.1), which serves as a tool to view the structure of a ROOT file. It also includes a part with `TCanvas`, a canvas that is used for plotting graphs.

Both the CLI and GUI include tools to modify the visualisation of data. The visualised data can be also modified while visualising.

To keep plots and figures consistent, scripts are the best way to create them. The ATLAS Experiment group has its own visual style that modifies the font used, the style of lines, hides figure title by default, etc.

2.2.4 Lxplus

The software is to be run mainly on lxplus servers [26]. It is a cluster of Linux machines running CERN CentOS 7 in 64-bit mode. It is a load balanced cluster that is accessible by SSH. Each machine in the cluster is connected

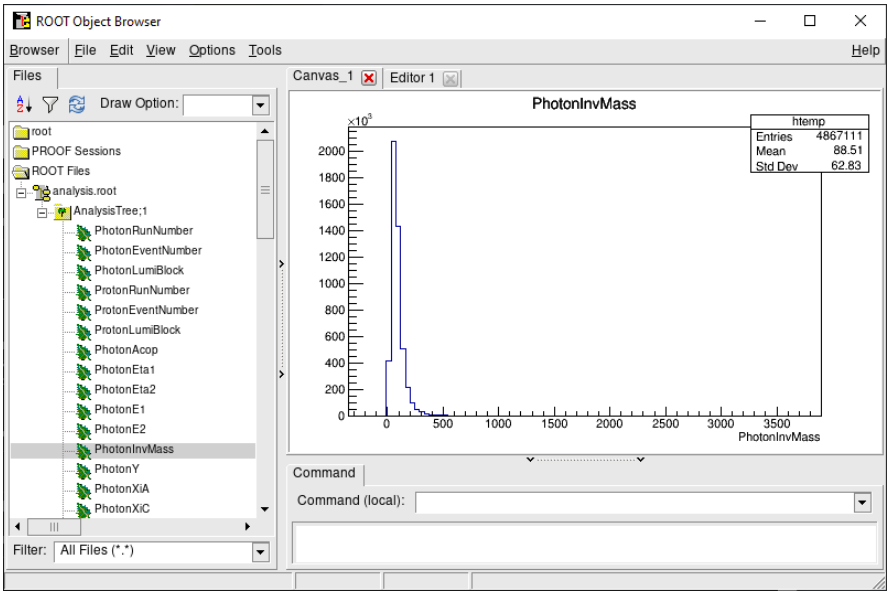

Figure 2.1: An example window of ROOT's `TBrowser` GUI.

to two file systems to share data, the AFS and EOS. Users have folders with access on both file systems. AFS is a smaller, but faster file system while EOS has bigger capacity, but is generally slower. Software on lxplus servers can be run directly or using a job system HTCondor.

2.2.5 Input structure

The input file for the developed software is a ntuple created by the Group NTuple Production code that was briefly discussed in the previous section. The input structure is the same for both real and simulated data and both are to be processed in the same way. The ntuple contains two `TTree`s, `TreeProton` and `TreePhoton`. Inside them are `TBranch`es, each representing a measured variable.

There are three variables that are the same in both `TTree`s – `RunNumber`, `EventNumber` and `LumiBlock`. It is important that in case of standard matching these three variables have the same values for each event – that the code processes protons and photons corresponding to the same event. The Group NTuple Production code first checks if an event has exactly two photons. If the event passes this check, the measured photon data is inserted into `TreePhoton`. Failure to pass skips this event and the code processes the next event. After this comes another important check, if at least one proton was detected by the AFP detector. As with photons, if this check is passed, the measured variables are written into the `TreeProton`. If not, nothing is written into `TreeProton`. This implies that for every entry in the `TreeProton` there is an entry in the

TreePhoton with the same RunNumber, EventNumber and LumiBlock.

2.2.6 Output structure

The output files are also ntuples. Because the scripts need to process only the events that have exactly two photons detected and at least one proton detected, there is no need for two separate and independent TTrees with different amount of events. As I have decided to split the software into two scripts, one calculating the uncertainties and one calculating the difference of number of matches between different proton selections, the structures of the resulting ntuples is different.

For the script that calculates the uncertainties the ntuple contains a TTree, AnalysisTree. Inside that tree are the identifying variables, RunNumber, EventNumber and LumiBlock, but as a safe measure the variables are there twice, once taken from the TreePhoton and once from TreeProton. In case there are any doubts if the proton and photon data are from the same event the variables can be compared. Also included are almost all photon and proton variables from the input file. Added were the calculated uncertainties for both protons and photons and also variables showing the number of protons detected in the whole system and in both AFP detectors.

The ntuple created by the script that calculates the difference of number of matches between the different proton selections has a simpler structure. It contains one TTree as the previous one, but only four variables – A, C, AorC, AandC – each variable corresponding to match on side A, side C, side A or side C, side A and side C of the AFP.

2.3 Implementation

In this section the implementation of the software is discussed. First a short subsection about how I worked with the input files. Next the implementation of the resources shared by both scripts is given. It follows two sections, discussing the implementation of the matching and difference scripts, respectively.

2.3.1 Working with two independent TTrees

Because of the structure of the input file, both scripts were presented with a difficulty of working with the data. Having the photon and proton data in two separate TTrees is a problem, since the TTrees are independent on one another and the scripts need to access data from both TTrees. Worth mentioning is that the code needs to process only events that are in TreeProton, because matching can be done only when at least one proton is detected.

One option would be to brute-force and for each event in TreeProton iterate over events in TreePhoton until the RunNumber and EventNumbers are

the same. This method is very inefficient and thus not suitable, especially when the input files can be very large, in realm of million events.

This problem can be solved using ROOT – there is an option to assign primary and secondary index to a `TTree` by using method `TTree:BuildIndex`. This method needs to be called only on `TreePhoton`. Another important method is `TTree:AddFriend`. When iterating over entries in `TreeProton` by using `TTree:GetEntry`, `TTree:GetEntry` is also called on friend `TTree`s, in this case `TreePhoton`. Because the `TreePhoton` has indexes set, the entry that is returned is the one with the same values in the index variables. By doing this the `RunNumber`, `EventNumber` and `LumiBlock` variables are the same inside both `TTree`s when iterating over `TreeProton`. This means that the data inside `TreeProton` corresponds to the data in `TreePhoton`.

This method cannot be used with "mixed" proton selection, because values of `RunNumber` and `EventNumber` variables are different for protons and photons.

```
auto PhotonTree = (TTree*) Data->Get("TreePhoton");
auto ProtonTree = (TTree*) Data->Get("TreeProton");
PhotonTree->BuildIndex("RunNumber", "EventNumber");
ProtonTree->AddFriend("TreePhoton");
```

2.3.2 Shared resources

In order to reduce the possibilities of errors coming from inattention, having one file with functions needed by two both scripts is an option. Both scripts need to calculate the uncertainties for protons and photons and both scripts are doing the proton selection for which a function that minimizes the difference between photon and proton xi values is needed. The definitions of the functions that are included in both scripts are shown in Listing 2.1.

```
Double_t quadratureSum ( vector<Double_t> list );

Double_t calcPhotonSigmaE ( Double_t & PhotonE, Double_t &
    PhotonEta );

Double_t calcProtonSigmaXi ( Double_t & ProtonXi );

Double_t calcDeltaXi ( Double_t & ProtonXi, Double_t & PhotonXi );

Double_t minimizeDeltaXi ( vector<Double_T> & Protons, Double_t &
    PhotonXi );
```

Listing 2.1: Definitions of the functions in `shared.cxx`.

Function `quadratureSum` (Listing 2.2) was implemented using a vector as a parameter, because functions `calcPhotonSigmaE` and `calcProtonSigmaXi` need to calculate a quadrature sum of different number of values. Other option to vector is to define the function twice, once for three doubles as parameters (as used in `calcPhotonSigmaE`) and for four parameters (as used in

`calcProtonSigmaXi`), but that would be writing two implementations instead of one. Using a vector and an accumulator provides an option to calculate the quadrature sum of any amount of numbers.

```
Double_t quadratureSum ( vector<Double_t> list ) {
  Double_t acc = 0;
  for ( auto & i : list ) {
    acc += pow( i, 2 );
  }
  return sqrt( acc );
}
```

Listing 2.2: Implementation of the function `quadratureSum`.

`calcDeltaXi` function is a very simple function that returns -1 in case `ProtonXi` equals -1 and the difference between `ProtonXi` and `PhotonXi` otherwise. Functions `calcPhotonSigmaE` and `calcProtonSigmaXi` are functions that represent Equations 1.9, 1.13, respectively.

The last function, `minimizeDeltaXi` (Listing 2.3) takes as input a vector of doubles representing the Xi values of detected protons and a Xi value of the photon the protons are to be matched with, as discussed in Section 1.8. The purpose of the function is to return the Xi value of a proton that is the closest to a given photon Xi value. This is done using a brute-force method – checking the difference of each photon Xi and proton Xi and saving the lowest value in a variable. In case the vector with proton Xi values is empty, -1 is returned. For both `calcDeltaXi` and `minimizeDeltaXi` functions the -1 return value was not chosen randomly, the value is easily distinguishable as from the physics point of view it can not happen.

```
Double_t minimizeDeltaXi ( vector<Double_t> & Protons, Double_t &
    PhotonXi ) {
  Double_t minProtonXi = -1;
  Double_t minDeltaXi = 99999.;
  Double_t currDeltaXi;
  for ( auto ProtonXi : Protons ) {
    currDeltaXi = fabs ( ProtonXi - PhotonXi);
    if ( currDeltaXi < minDeltaXi ) {
      minProtonXi = ProtonXi;
      minDeltaXi = currDeltaXi;
    }
  }
  return minProtonXi;
}
```

Listing 2.3: Implementation of the function `minimizeDeltaXi`.

2.3.3 Matching script

The main point of this script is to create a ntuple with added photon and proton uncertainties. There is no specifically required time complexity for

this script, but the code should run with linear time with respect with the number of events in the ROOT ntuple.

To satisfy the functional requirement that the code should have the possibility to change the proton selection from "standard" to "mixed" case or "switched sides" case, I have decided to have three sections, two commented and one uncommented. The two main reasons for not including all three styles are the execution time and the size of the output ntuple. Because each event is uncorrelated to one another, the script iterates over all events and for each event does the proton selection routine and calculates the uncertainties.

If I put all variables for all three selection styles inside one `TTree`, the structure of the `TTree` would not be clear. Alternative would be to separate the selection styles into three different `TTree`s, but it is not suitable either, because a big part of the data would be duplicated.

First iteration of this script opened the input ntuple and created the output ntuple. The input file is opened in `READ` mode because it does not have to be modified. The output file is opened in `UPDATE` mode, which creates a file or prepares to update already existing file. Next iteration included the creation of `AnalysisTree` and `TBranches` representing variables. I have decided to turn off `AutoSave` option of `TTree` because if it is on, the resulting ntuple contains multiple backup copies of the same `TTree`. This iteration also included the variables for all `TBranch`es. An important step of this iteration was also the inclusion of iterating loop that iterates over all events in `TreeProton`. The photon and proton variables were copied using this iteration loop from the input file to the output file.

Another iteration included the "standard" proton selection and the calculations of the uncertainties (Listing 2.4). "Switched sides" and "mixed" proton selections were included in next iteration. Follows an iteration that informs the user about the part of the code that is executed at the moment by printing lines into standard output. In this iteration were also included time measurements using `std::chrono`, but they have been removed for the final iteration.

The final iteration first opens the input file and uses the method discussed in subsection 2.3.1 to ensure that the photon and proton data accessed have the same `RunNumber` and `EventNumber`. Also the output file and its `TTree` is created. This is followed by declarations of variables and setting their branch addresses (in case the variables exist in input file). Next the creation of branches inside the output file is discussed. The iteration over all events in the `TreeProton` follows. For each event `PhotonE1` and `PhotonE2` are converted from MeV to GeV and the uncertainties are calculated. Then the memory is cleaned up and the output file is written.

```
//Conversion to GeV
PhotonE1 /= 1000;
PhotonE2 /= 1000;

//Calculating PhotonSigmas
PhotonSigmaE1 = calcPhotonSigmaE( PhotonE1, PhotonEta1 );
PhotonSigmaE2 = calcPhotonSigmaE( PhotonE2, PhotonEta2 );
PhotonSigmaInvMass = PhotonInvMass / 2 * quadratureSum( {
    PhotonSigmaE1 / PhotonE1, PhotonSigmaE2 / PhotonE2 } );
PhotonSigmaXiA = PhotonXiA / PhotonInvMass * PhotonSigmaInvMass;
PhotonSigmaXiC = PhotonXiC / PhotonInvMass * PhotonSigmaInvMass;

//Minimizing DeltaXi -- standard case
nProtonsA = ProtonCandsXiA->size();
nProtonsC = ProtonCandsXiC->size();
nProtons = nProtonsA + nProtonsC;
ProtonXiA = minimizeDeltaXi ( *ProtonCandsXiA, PhotonXiA );
ProtonXiC = minimizeDeltaXi ( *ProtonCandsXiC, PhotonXiC );
DeltaXiA = calcDeltaXi ( ProtonXiA, PhotonXiA );
ProtonSigmaXiA = calcProtonSigmaXi ( ProtonXiA );
DeltaXiC = calcDeltaXi ( ProtonXiC, PhotonXiC );
ProtonSigmaXiC = calcProtonSigmaXi ( ProtonXiC );
```

Listing 2.4: Calculations done for each event for "standard" photon-proton matching.

2.3.4 Difference script

This script calculates the difference of number of matches between "standard" matching and "mixed" matching, where the proton data is used from another event. In this case the goal is to get the statistic variation between random matching and nominal matching. In order to do that, there have to be statistical iterations of the mixed selection case. The first iteration is the proton information from previous event, in the next iteration proton information from the event "n-2" is taken, etc. In order to remove unwanted correlations, the proton selection must be done separately for each statistical iteration for each event.

This introduces a huge priority into speed optimization of the script. Because the proton selection is to be done for $n-1, \ldots, n-1000$ cases, the time to complete each iteration should be at worst constant, which results in linear total execution time with respect to the number of iterations. Anything asymptotically slower would result in a long execution time.

The number of matches for each iteration for each detector side option is kept in a variable which is changed in case of match, after each iteration the difference between the "standard" and the "mixed" matching is calculated and saved into the output file.

One option that I had to consider was the use of multi-threading. After a discussion with the supervisor I decided to treat multi-threading as a fi-

nal resort if I could not achieve the time requirement without it. The main disadvantage of multi-threading is the thread safety problem which would be present in this case.

The first idea was to check if accessing data in input ntuple has constant time complexity. The idea is that if the time complexity is constant, then for each iteration the script can copy photon data into separate variables and then access proton data from different event and calculate.

The first iteration that checked the access time complexity included the same basic layout as the matching script. First the script opened the input file and created the output file. This was followed by declaring the necessary variables and creating branches in the output file. This was followed by two loops, first one looping over the set number of iterations, in this case 1000. The second loop iterated over all events in the input file, did the proton selection, calculated necessary variables and checked the event for a match for side A and side C. Next the variables representing the number of matches in one iteration for side A, side C, side A or side C, side A and side C match were modified accordingly. After the loops only a memory cleanup and file writing were done.

The result of this test was that the execution time for each iteration was increasing even though the number of accesses was constant. To get over this problem I decided to copy the proton and photon data into structures that have constant access time (Listing 2.5). The main problem with this solution is that there is a chance that there is a failure to allocate enough memory to contain all data. I decided to put a try-catch block into the code to catch this possible exception. The selected data structure was `std::vector`. I have also considered a standard C++ array, but the ease of use of vector and the easier readability of the code were the main decision arguments.

```
for (Long64_t i = 0; i < nEntries; i++) {
  ProtonTree->GetEntry(i);
  try{
    ProtonCandsA.push_back(*ProtonCandsXiA);
    ProtonCandsC.push_back(*ProtonCandsXiC);
    Photons.push_back(Photon(PhotonAcop, PhotonE1, PhotonE2,
        PhotonEta1, PhotonEta2, PhotonXiA, PhotonXiC,
        PhotonInvMass));
  }
  catch ( exception e ) {
    cout << "Exception related to copying data occured." << endl;
    cout << e.what() << endl;
    return;
  }
}
```

Listing 2.5: Loop in the difference script that is responsible for copying data from the input file into `std::vector`.

For the proton data, the variables copied were already `vector`s of doubles,

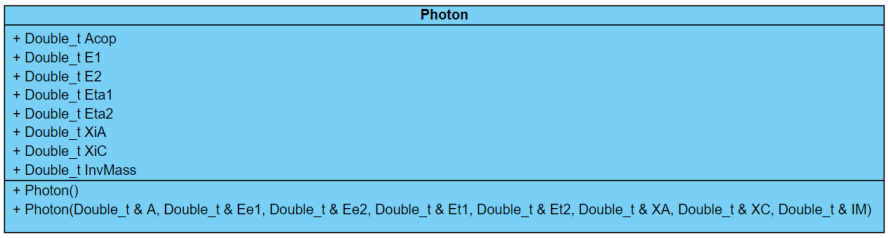

Figure 2.2: Class diagram describing class `Photon`.

but for photons there were more variables that needed to be copied in to calculate the uncertainties. I have decided to create a simple class (Figure 2.2) containing all the needed variables with an empty constructor and a constructor that assigns all the variables accordingly. It is worth mentioning that in case of memory allocation failure there is a possibility to not copy the photon data, but the downside would be increased execution time. By copying also the photon data the script goes through all events only once, copies all necessary data and then only accesses the newly created vectors with constant access time. If only proton data was copied, the script would still need to access the ntuple once for each event in each iteration.

The final iteration first opens the input file and creates the output file. After the declaration of variables and creating both `TTree` and `TBranch`es in the output file the script copies the photon and proton data into vectors. After this comes a loop that iterates over the number of total iterations +1 (the reference case, number of matches when using "standard" matching). Inside that loop is another loop that iterates over all events. Inside that are the necessary calculations and the check if there are matches on each side of the AFP detector. Once all events in the iteration are processed, the difference between reference and the iteration is calculated (in case of the reference case the values are only saved). Once all the iterations are processed, the script cleans up memory and saves the data into the output file.

2.4 Testing

This section includes a discussion on unit, time complexity and user acceptance testing. All the tests were done on lxplus servers. Because the software was run on lxplus servers at all times, no specific integration tests were made.

2.4.1 Unit testing

The purpose of unit testing is to test a very small part of code, typically one function or method. All functions in the file with shared resources can be unit tested. To check if the returned values of the functions are correct I chose to create a ROOT script to unit test. Other option that I considered was

standard C++ code, but to keep everything consistent a ROOT script was the better choice.

Most of the unit tests have trivial, edge cases and few manually created tests, none of the unit tests were randomly generated due to lack of time to properly create a generator. The only minor problem with unit testing in this case is the problem of comparing doubles, an epsilon value of $1 \cdot 10^{-6}$ was used.

```
bool test1() {
    Double_t expected = 0;
    Double_t result = quadratureSum({});

    if (fabs(result-expected) < EPS) {
        cout << "Test 1 passed" << endl;
        return true;
    }
    cout << "Test 1 failed" << endl;
    return false;
}
```

Listing 2.6: Example of unit tests used.

2.4.2 Time complexity testing

As stated in the requirements, the difference script should not exceed 6 seconds per iteration and the matching script has no explicitly stated execution time requirement. The time measurement itself was at first done by hand using a stopwatch to get a rough idea about the execution time. To get more precise times, `std::chrono` library was used with conjunction with printed lines into standard output with time splits for each part of the scripts. This method which uses `std::chrono` worked well to get the iteration times for the difference script (results were the same for stopwatch measurements), but the overall execution time was increased greatly. To be more specific, the execution time increased for the initialization part of the code. This is the reason why in the final iterations of the scripts there are no time measurements. From the point of ROOT, the scripts were both only interpreted and also pre-compiled to compare and get a rough idea about the performance benefits of pre-compiling. The scripts were executed directly, without the use of the HTCondor job system.

2.4.3 User acceptance testing

User acceptance tests are tests that verify with the end user if the software completes the set requirements. The testing was done by comparing the output to an expected result. For the matching script, the variables that were directly taken from the input file and copied into the output file should be the same. The calculated variables are checked by unit tests. The results of the difference

script were also compared to the number of matches in the matching script when using all three different proton selection methods.

Chapter 3

Results

In this chapter there is a discussion on observed software performance which is followed by section about the physics results obtained by using the software and analyzing the data produced.

3.1 Software performance

The scripts were run on the lxplus servers as mentioned before. The full 2017 data contains 12.3 million events in PhotonTree and 4.8 million events in ProtonTree. The simulation on the other hand contains 7377 events in both TTrees.

3.1.1 Matching script

The final iteration has execution time of around two minutes for the whole 2017 data. There is no difference in execution time between interpreting the code and pre-compiling the code. For the simulation the execution time was almost instant, under five seconds.

This script was the main reason why `std::chrono` was removed from both scripts. The observed execution time without `std::chrono` was around two minutes, but after adding time measurements the execution time increased to over 10 minutes.

3.1.2 Difference script

The first iteration which checked if the access time was constant had linearly increasing execution time per iteration. While interpreting the first iteration of the script, the first calculated iteration took on average 60 seconds to complete, the second one 140 seconds and the third one 200 seconds.

As this result was very unsatisfying, it was followed by the iteration of the code that copied the proton and photon data into `std::vector`. While

interpreting this iteration of the code, the statistical iterations had an average execution time of 7.8 seconds. The one downside was the increased time before the code started calculating the statistical iterations, before the initialization was almost instant, but now the software took on average 140 seconds to copy initialize and copy the required data. The execution time per statistical iteration was further reduced by pre-compiling the code to 1.5 seconds on average.

After realising that `std::chrono` has unwanted behavior of increasing the execution time, it was removed from this script as well and the execution times were roughly timed again by using a stopwatch. The initialization time was reduced to around 60 seconds and the time to calculate an iteration was the same, around 1.5 seconds which satisfies the requirement that an iteration should be calculated in less than 6 seconds. The total execution time for all 1000 statistical iterations is around 30 minutes.

3.2 Results of using the software

In this section are discussed all the results from the data created using the developed code. First, the cuts on the data are mentioned. Next, the final number of detected matches is presented and the optimization of the matching criteria is discussed. Lastly, the randomness of the matching and the ALP signal matching efficiencies are discussed.

3.2.1 Initial data selection

Important condition is that the 2017 data (not simulations) needs to be so called blinded – the photon acoplanarity must be greater than 0.01. The acoplanarity is the angle from being back-to back. The reason for blinding is to avoid any bias in the analysis which could influence the detection or exclusion of an ALP signal. It is a standard procedure for any search for a new particle.

Independent of the blinding there are also cleanup cuts $\xi_{\mathrm{AFP}} > 0.01$ and $\xi_{\gamma\gamma} > 0.01$. The reason for these cuts is the detection range of the AFP detectors.

3.2.2 Statistical uncertainties

The statistical uncertainty on the difference between ξ_{AFP} and $\xi_{\gamma\gamma}$ can be obtained from the simulated data. While the ATLAS central detector has a dedicated group towards simulation, the AFP uses a module called `AFPFastSimTool` inside the `AFPToolbox` framework to simulate the proton data. Because the STDM2 derivation of the ATLAS simulation did not include so called Truth-Proton container, a local modification of the `AFPFastSimTool` had to be done in the Group Production NTuple code to obtain simulated data for protons.

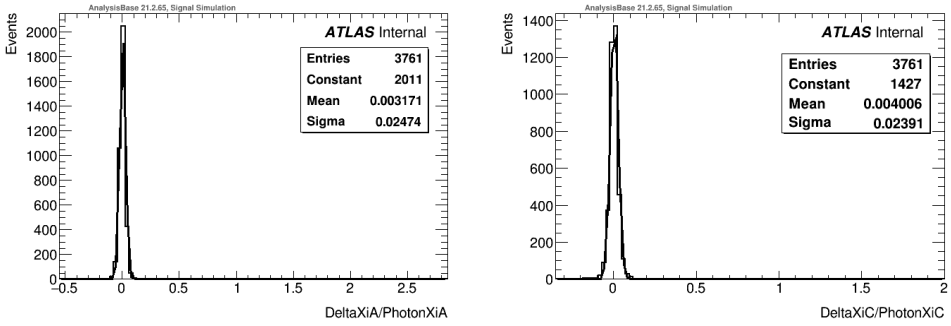

Figure 3.1: Relative $\Delta\xi$ uncertainty coming from signal simulation for side A (left) and side C (right).

The statistical uncertainty represents random fluctuations on the measurements by the detectors. The statistical uncertainty is determined by the width of the gaussian fit of the $\Delta\xi$ coming from the simulations. The determined uncertainty for the $\Delta\xi$ is around 2.5% for both sides A and C (Figure 3.1).

When comparing the statistical uncertainties to the systematic ones the conclusion is that the systematic uncertainties are greater compared to the statistical. The main part of systematic uncertainties is the proton systematic uncertainty, which is around 10%.

3.2.3 Number of matches

The number of matches in 2017 data for each side and each matching condition is shown in Table 3.1. The low amount of matches for 10% matching can be explained by the high systematic uncertainty discussed in section systematic uncertainties.

The di-photon invariant mass distributions for A and C matching for all three matching criteria is shown in Figures 3.2, 3.3, 3.4. The figures show that the shape of the di-photon invariant mass distribution look similar for all matching criteria with a peak at around 500 GeV. Clearly, the 1σ and 2σ matching conditions increased the amount of matches significantly. It is important to note that the optimized matching criteria take into account varying uncertainties event by event, while the simple 10% matching criteria does not.

3.2.4 Random matches (pile-up background)

The number of matches for "mixed" and "switched sides" cases are shown in Tables 3.2 and 3.3, respectively. The direct comparison between nominal and mixed matches with the respective uncertainty gained from $n = 1000$ statistical iterations are shown in Table 3.4.

Matching	A	C	A or C	A and C
10%	34 442	47 849	82 224	67
1σ	47 421	66 642	113 943	120
2σ	95 971	135 386	230 848	509

Table 3.1: Number of matches across side A, side C, side A or side C, and side A and side C for 10%, 1σ and 2σ matching (2017 data).

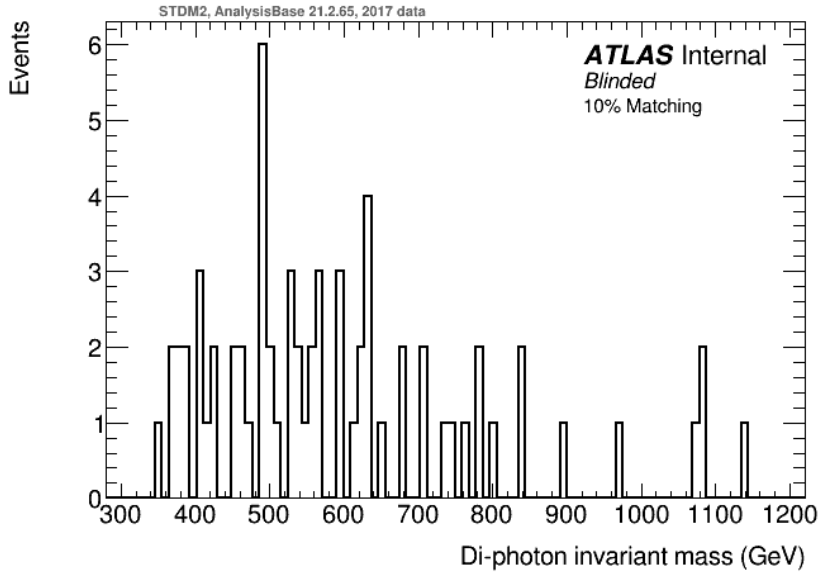

Figure 3.2: Di-photon invariant mass of the matched events on sides A and C (at the same time) for 10% matching.

In theory there is an expectation that some fraction of matches is based on randomness. The fraction of randomness is examined by taking by definition uncorrelated data and looking at the number of matches. In this case the number of nominal matches is compatible within the determined statistical uncertainties with the number of random matches (uncorrelated matches). This concludes that the matching done in this analysis was a series of random coincidences. This is consistent with previous simulation results where it was found that the dominant background comes from pile-up, not from physics reactions [5].

3.2.5 Matching efficiencies

The ALP signal matching efficiencies are analyzed by performing the matching criteria on the simulated data (Table 3.5). In this case the ξ data cleaning cut is $\xi \in \langle 0.02; 0.1 \rangle$ for both ξ_{AFP} and $\xi_{\gamma\gamma}$. As expected, the efficiencies are lower

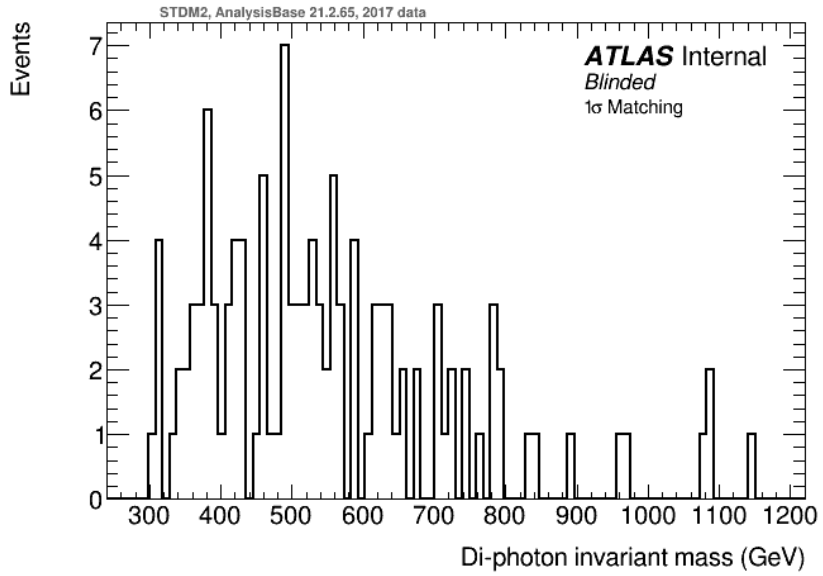

Figure 3.3: Di-photon invariant mass of the matched events on sides A and C (at the same time) for 1σ matching.

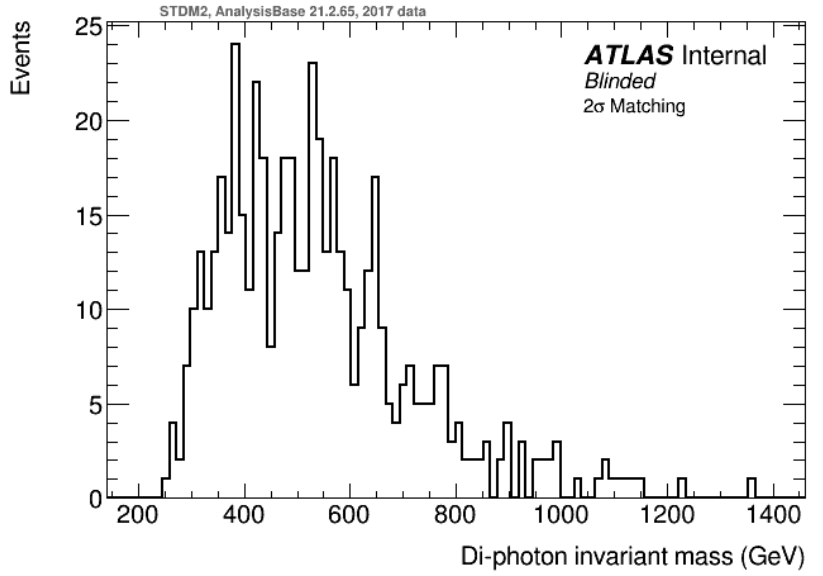

Figure 3.4: Di-photon invariant mass of the matched events on sides A and C (at the same time) for 2σ matching.

than at the generator level. These efficiencies are important to determine the sensitivity to detect an axion-like-particle with the ATLAS central and AFP

Matching	A	C	A or C	A and C
10%	34 534	47 870	82 338	66
1σ	47 338	67 000	114 209	129
2σ	95 469	135 636	230 580	525

Table 3.2: Number of matches for "mixed" case (di-photon information taken from event "n" and AFP information taken from event "n-1").

Matching	A	C	A or C	A and C
10%	34 340	48 130	82 399	71
1σ	47 296	67 068	114 224	140
2σ	95 830	136 466	231 745	551

Table 3.3: Number of matches for "switched sides" case.

Matching	Nominal (blinded)		Random matching	Uncertainty
A	95 971		95 469	± 271
C	135 386		135 636	± 307
A or C	230 848		230 580	± 415
A and C	509		525	± 22

Table 3.4: Comparison between the number of nominal matches and random matches with uncertainty (2σ matching).

detectors.

Matching	A	C	A or C	A and C
10%	3736	3725	5752	1709
	50.6%	50.5%	78.0%	23.2%
1σ	3745	3739	5770	1714
	50.8%	50.7%	78.2%	23.2%
2σ	3757	3760	5790	1727
	50.9%	51.0%	78.5%	23.4%
Generator – 10%	66.9%	66.9%	89.1%	25.0%

Table 3.5: The number of matches and their percentages related to the total number of di-photon events with at least one proton present (7377 events) for all three matching criteria. The table also lists the percentages on the generator level for 10% matching.

Conclusions

The main goal of this thesis was to create software that would modify existing data to include calculated uncertainties and that would calculate the difference between nominal matching and uncorrelated data matching. The software created fulfils all stated requirements. Another goal of this thesis was to optimize the matching criteria in proton-proton interactions with exactly two photons observed. The important extension is the that a simple matching criteria was extended to include the systematic uncertainties in the photon and proton measurements. The optimization was used to study the number of events in the recorded data and the efficiencies of a simulated ALP signal. Also another goal was to provide clarity to the randomness of the matching. The large data set provided by the ATLAS central and AFP detectors is a challenge from the point of view of Information Technology.

A notable result from the physics point of view is that the analyzed similarity between ξ_{AFP} and $\xi_{\gamma\gamma}$ is a random coincidence instead of a physics process with low randomness. This result is consistent with previous simulations. The three matching criteria were compared by the number of matches and the di-photon invariant mass distribution. Also the uncertainties on proton and photon reconstruction were discussed. It was determined that the sum of proton and photons systematic uncertainties is much larger than the statistical uncertainties. Lastly, the ALP signal matching efficiencies were compared against efficiencies determined on the generator level.

The written software can serve for similar analysis and for reproduction of these results. The code is written in a way that can be changed in case of an update of the selection of the di-photons or changes in the proton detection system. Another use for the software is to serve as an example of how to work with correlated data in two different `TTree`s.

Bibliography

[1] ATLAS Collaboration. Observation of light-by-light scattering in ultra-peripheral Pb+Pb collisions with the ATLAS detector. *Phys. Rev. Lett.*, volume 123, no. 5, 2019: p. 052001, doi:10.1103/PhysRevLett.123.052001, 1904.03536.

[2] d'Enterria, D.; da Silveira, G. G. Observing light-by-light scattering at the Large Hadron Collider. *Phys. Rev. Lett.*, volume 111, 2013: p. 080405, doi:10.1103/PhysRevLett.111.080405,10.1103/PhysRevLett.116.129901, [Erratum: Phys. Rev. Lett.116,no.12,129901(2016)], 1305.7142.

[3] Euler, H. On the scattering of light by light according to Dirac's theory. *Annalen Phys.*, volume 26, no. 5, 1936: pp. 398–448, doi:10.1002/andp.19364180503, [Annalen Phys.418,no.5,398(1936)].

[4] ATLAS Collaboration. Evidence for light-by-light scattering in heavy-ion collisions with the ATLAS detector at the LHC. *Nature Phys.*, volume 13, no. 9, 2017: pp. 852–858, doi:10.1038/nphys4208, 1702.01625.

[5] Baldenegro, C.; Fichet, S.; et al. Searching for axion-like particles with proton tagging at the LHC. *JHEP*, volume 06, 2018: p. 131, doi:10.1007/JHEP06(2018)131, 1803.10835.

[6] Beresford, L.; Bussey, P.; et al. Measurement of proton-tagged lepton pairs in photon fusion using the ATLAS Forward Proton spectrometer. Technical report ATL-COM-PHYS-2020-205, CERN, Geneva, Mar 2020, [Online; accessed April 10, 2020; public in: ATLAS-CONF-2020-041]. Available from: https://cds.cern.ch/record/2712727

[7] Looking forward: ATLAS measures proton scattering when light turns into matter. [Online; accessed July 30, 2020]. Available from: https://atlas.cern/updates/physics-briefing/looking-forward-light-matter

[8] About CERN. [Online; accessed May 28, 2020]. Available from: https://home.cern/about

[9] The birth of the Web. [Online; accessed May 28, 2020]. Available from: https://home.cern/science/computing/birth-web

[10] Evans, L.; Bryant, P. LHC Machine. *JINST*, volume 3, 2008: p. S08001, doi:10.1088/1748-0221/3/08/S08001.

[11] The Large Hadron Collider. [Online; accessed May 28, 2020]. Available from: https://home.cern/science/accelerators/large-hadron-collider

[12] Horvath, A. The LHC experiments and the preaccelerators. 2006, [Online image; accessed May 28, 2020]. Available from: https://commons.wikimedia.org/wiki/File:LHC.svg

[13] ATLAS Collaboration. The ATLAS Experiment at the CERN Large Hadron Collider. *JINST*, volume 3, 2008: p. S08003, doi:10.1088/1748-0221/3/08/S08003.

[14] ATLAS Collaboration. Performance of the ATLAS detector using first collision data. *JHEP*, volume 09, 2010: p. 056, doi:10.1007/JHEP09(2010)056, 1005.5254.

[15] About the ATLAS Experiment. [Online; accessed May 28, 2020]. Available from: https://atlas.cern/discover/about

[16] Adamczyk, L.; et al. Technical Design Report for the ATLAS Forward Proton Detector. 2015.

[17] Pequenao, J. Computer generated image of the whole ATLAS detector. 2008, [Online image; accessed May 28, 2020]. Available from: https://cds.cern.ch/images/CERN-GE-0803012-01

[18] Sopczak, A.; Bussey, P.; et al. Search for an Axion-Like Particle in Light-by-Light scattering using the ATLAS central detector and the ATLAS Forward Proton detector. Technical report ATL-COM-PHYS-2020-238, CERN, Geneva, Mar 2020, [Online; accessed June 1, 2020]. Available from: https://cds.cern.ch/record/2714416

[19] Harland-Lang, L. A.; Khoze, V. A.; et al. Photon-Photon Collisions with SuperChic. *CERN Proc.*, volume 1, 2018: p. 59, doi:10.23727/CERN-Proceedings-2018-001.59, 1709.00176.

[20] Todesco, E.; Wenninger, J. Large Hadron Collider momentum calibration and accuracy. *Phys. Rev. Accel. Beams*, volume 20, 2017: p. 081003, doi:10.1103/PhysRevAccelBeams.20.081003. Available from: https://link.aps.org/doi/10.1103/PhysRevAccelBeams.20.081003

[21] ATLAS Collaboration. Electron and photon energy calibration with the ATLAS detector using LHC Run 1 data. 2014, doi:10.1140/epjc/s10052-014-3071-4, 1407.5063.

[22] Staszewski, R. Towards optics uncertainty. 2019, [Private conversation].

[23] Brun, R.; Rademakers, F. ROOT – An object oriented data analysis framework. *Nucl. Instrum. Meth. A*, volume 389, no. 1, 1997: pp. 81 – 86, ISSN 0168-9002, doi:10.1016/S0168-9002(97)00048-X.

[24] About ROOT. [Online; accessed May 28, 2020]. Available from: https://root.cern.ch/about-root

[25] ROOT/Getting Started/Many Ways to Use ROOT. [Online; accessed May 28, 2020]. Available from: https://en.wikibooks.org/wiki/ROOT/Getting_Started/Many_Ways_to_Use_ROOT

[26] LXPLUS Service. [Online; accessed July 30, 2020]. Available from: http://information-technology.web.cern.ch/services/lxplus-service

Appendix A

Acronyms

AFP Atlas forward proton

ALP Axion-like-patricle

ATLAS A Toroidal LHC Apparatus

CERN Conseil Européen pour la recherche nucléaire

CLI Command line interface

GRL Good Run List

GUI Graphical user interface

IP Interaction point

LbyL Light-by-light

SM Standard model

Appendix B

Contents of enclosed SD card

- `readme.txt` ..... the file with SD card contents description
- `thesis.pdf` ..... the thesis text in PDF format
- `implementation` ..... implementation sources
  - `input` ..... example input files for the software
- `thesis` ..... the directory of LaTeX source codes of the thesis
  - `figures` ..... the directory with figures in .png format