Development, Testing and Future Outlook of a Penalized Spline as a Representation of the Standard Model Background in Dijet Searches

Alexander Ekman
Supervised by Caterina Doglioni and Daniel Whiteson

Thesis submitted for the degree of Bachelor of Science
Project duration: 2 months

LUND UNIVERSITY

Department of Physics
Division of Particle Physics
May 2017

Abstract

The aim of this thesis is to develop, test and evaluate a penalized basis spline as a method for representing the Standard Model background in dijet searches from proton-proton collisions at the Large Hadron Collider. Current representations of the Standard Model background are complex and dependent on current assumptions of the Standard Model. Using a P-spline is interesting because of its simplicity and independence from assumptions of the data. Its performance will be tested by fitting $p + p \rightarrow j + j$ generated events from MadGraph and ATLAS data with different penalization parameters of the spline. The test is evaluated using a χ^2 goodness-of-fit test. The distribution of χ^2 for 100 simulated dijet events peaked at 1, showing that the spline has no problem representing the Standard Model background. The spline can also perform a good fit on ATLAS data.

This thesis concludes that the P-spline can represent the Standard Model background and perform a good fit on ATLAS data. To conclude that the P-spline can be used as a tool to aid particle searches, further research needs to be done where the P-spline is tested with signal injection.

Contents

List of Abbreviations

Standard Model = SM
Standard Model Background= SMB
MadGraph5_aMC@NLO = MG
LHC = Large Hadron Collider
ATLAS = A Toroidal LHC Apparatus
B-spline=Basis spline
P-spline=Penalized Basis spline

1 Introduction

The Standard Model of particle physics (SM) attempts to describe the fundamental particles and how they interact with each other. Ever since the SM was proposed in the 1970s, new discoveries and more precise measurements have changed and improved the model [1]. The SM has been effective and precise at describing the universe's most fundamental constituents of matter thus far. Despite the model's success it has well-known flaws, for example, neither gravity nor dark matter is described in the model [1]. One way to solve these issues is to look for particles which can explain these observed phenomena. The search for dark matter particles and associated new phenomena is one of the goals for the ATLAS (A Toroidal LHC Apparatus) experiment at the Large Hadron Collider (LHC) [2][3]. This thesis is based on data collected by the ATLAS experiment in 2015 [4].

In many scientific measurements, the presence of a signal needs to be distinguished from background noise. A common way to tackle this is to estimate or measure the background separately, then collect the data which can contain both signal and the background. To make it easier to find a signal in data one then subtract the background from the data. A similar method to this is used at multiple searches for new particles, among others the dijet searches at the ATLAS experiment [5].

At the LHC, beams of protons are set to collide with one another at multiple points along the accelerator. During these energetic collisions, new particles can be created. As they have short life-times, it is currently impossible to detect them directly. The particles are instead detected indirectly through their decay products. The ATLAS detector is positioned at one of LHC's collision sites. By recording the momentum of all the decay products from the particle, it is possible to calculate the mass of the initial particle. This mass is called the invariant mass and is independent of frame of reference [6]. A large enough data set of invariant masses can then be analyzed to determine the presence or absence of a new particle.

To find the signal of a new particle in the data, scientists use models to represent what current theories, such as the SM, predict the invariant mass distribution to look like. This prediction is the estimated Standard Model background (SMB) which is then compared to the data. Any significant deviation from the SMB suggests existence of something which is unknown to current theories such as the SM.

This thesis aims to develop and test a penalized basis spline (P-spline) as a representation of the SMB in dijet searches at the ATLAS experiment. A P-spline function is continuous and piecewise-defined by polynomials which are penalized by a penalty function that enforces smoothness [7][8]. This means that it is a generally smooth, continuous function made from connected polynomials. The main method of this thesis is using a program which implements the Python library SPLINTER [9] to fit a P-spline to both simulated and experimental ATLAS data [4]. The performance of the P-spline will be determined by a χ^2 goodness-of-fit test.

The research done in this thesis was limited by the implementation of the SPLINTER library. Issues include handling of data uncertainties and zero counts in histograms. However, changes have been made to the library that could fix these issue in future research. Further limitations of this research include not using shower and detector simulations while generating SMB events.

Section 1 introduces the Standard Model, its flaws, and one of the ways it can be improved. It further introduces the ATLAS detector and how work developed in this thesis could contribute to particle searches. Section 2 gives a more detailed account of the Standard Model, a closer look at the interactions dealt with in this thesis, a closer look at the model's flaws and how finding new particles can fix some of those flaws. Chapter 3 describes the tools used in this thesis to obtain experimental data, simulated data, the basics of a P-spline and how the goodness-of-fit is determined. Section 4 describes the method and the code used to investigate the performance of the P-spline. Section 5 presents the results of the thesis. Section 6 draws conclusions from the results. Section 7 discusses the future outlook for the P-spline and what should be done in future work to further investigate its ability to work as a tool in particle searches.

2 Theory

2.1 The Standard Model

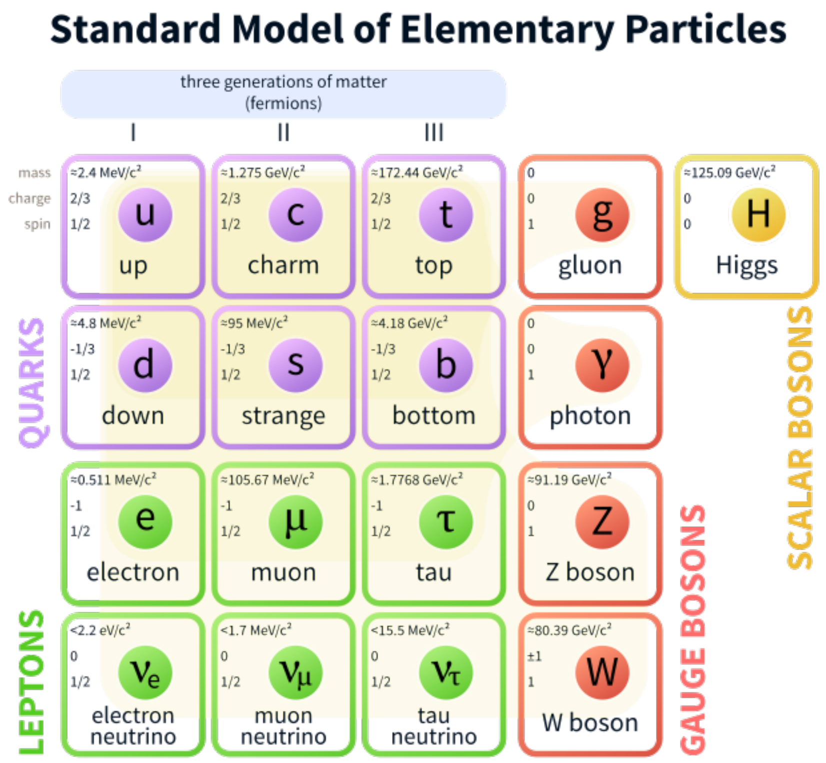

Figure 1: A visual grouping of the 4 quarks, 4 leptons, 4 force carriers and the Higgs which are represented in the Standard Model [10].

The SM attempts to describe the fundamental particles and their interactions with each other. The model currently describes the interactions between 12 matter particles with three fundamental forces and the Higgs boson [1]. The fourth force, gravity, has not yet been included. The three fundamental forces included in the SM are the electromagnetic, strong, and weak force. The particles of the SM are divided into two groups depending on their spin, an intrinsic form of angular momentum. As seen in Figure 1, the matter particles have a half-integer spin and are called fermions, whereas the particles that mediate the three fundamental forces have an integer spin, and are called gauge bosons. The Higgs boson is a scalar boson, it is not a force carrier and has a spin of 0.

When matter particles, fermions, interact with each other using a force, they are exchanging force-carriers of that force. Each force has a different carrier and is effective within a certain range. The strong force is carried by the gluon, compared to the weak and electromagnetic force it is very strong but it is only effective within a short range, 0.8 fm, such as the distance of an atomic nucleus. The electromagnetic force is carried by the photon, it has an infinite range, it is weaker than the strong force but stronger than the weak force. The weak force is carried by the Z and W bosons, and similarly to the gluon has a very short range. It is also the weakest of the three forces included in the SM.

As seen in Figure 1, the 12 fermions that interact using these forces are further divided into groups depending on their observed characteristics. The leptons do not interact with the strong force but the quarks do. There are three generations of leptons with two leptons in each generation. The first, second and third generation consist of the electron and the electron neutrino, the muon and the muon neutrino, and the tau and the tau neutrino respectively. The electron, muon and tau all have charge; Therefore they interact with the weak force and the electromagnetic force. The neutrinos, on the other hand, have no charge and only interact via the weak force.

The quarks are also split into three generations, each including two quarks. The first, second and third generation contain the up and down, charm and strange, and the top and bottom quark respectively. The quarks have mass, charge and color charge, which means that they interact via weak, strong and electromagnetic force. The gluon, which carries the strong force, also has color charge. This gives rise to some interesting behavior which will be discussed later [6].

The last particle to be discussed and most recently discovered is the Higgs boson. The Higgs boson is an excitation of the Higgs field. As particles move through the Higgs field some interact with it. Depending on the strength of the interaction with the Higgs field they are given a mass proportional to the interaction with the field [11].

2.1.1 Quantum Chromodynamics (QCD)

The strong force is described by QCD. As mentioned before, unlike the electromagnetic force, the gluon couples to color charge whilst having a non-zero color charge itself. For this reason, gluons self-interact, as well as interacting with quarks [6]. This property of self-interaction among others gives rise to what is called the running coupling constant of the strong force. The coupling constant is a measurement of the strength of a force. The strong force is weak at short distances, i.e. large momentum transfers, creating asymptotic freedom. The force is strong at longer distances, i.e. small momentum transfers, creating color confinement. Asymptotic freedom refers to the fact that at short distance interactions, the strength of the interaction between particles become asymptotically weaker. This effectively makes them free particles at short distances. Color confinement refers to the fact that all observed states have zero color charges, i.e. quarks and gluons cannot be observed as isolated free particles [6].

If we use the following process as an example:

$$q + \bar{q} \rightarrow q + \bar{q} \tag{1}$$

Where these quarks represent some quarks (either valence or sea quarks) involved in a proton-proton collision, such as those produced at the LHC. The two resulting quarks are then seemingly free isolated particles, but because of the running coupling constant of the strong force they undergo a strong interaction process called fragmentation [12]. Here the high energy resulting quarks turn into two bursts of many hadrons. Such a burst of particles arising from the hadronization and fragmentation of partons after a collision event is called a jet [13]. The fragmentation process itself is complicated and will not be covered here, but the direction of a jet is defined by the total momentum of all particles inside the jet. The following describes the momentum P of a jet as a sum of momenta of particles inside it:

$$P = \sum_i p_i \tag{2}$$

This allows us to represent the final state quarks, which cannot be observed as free isolated particles, as the single momentum vector of the jets that are actually observed.

This thesis will work with experimental and computer simulated data of $p + p \to j + j$ events, where the momentum of a jet is representing the momentum of final state gluon or quark pair from a proton collision in a proton collider [6].

2.1.2 Problems with the Standard Model

There are a number of features of the SM that are problematic. The major issues can be divided into two major categories: unexplained observations and experimental results, and theoretical problems.

As mentioned before, observations such as gravity and dark matter, are not included in the SM [14]. This is problematic because the aim of the SM is to explain how fundamental particles of matter interact and if their interactions using gravity are not included, or supposed particles such as dark matter are not present then the SM is incomplete. Astronomical observations have shown that galaxies in our universe experience a centrifugal force greater than the gravitational force exerted by its observable matter. This means that matter which has not been observed by reflection, emission, or absorption of light, is adding mass to galaxies and is effectively keeping them together. It can, therefore, be concluded that there is an unobserved type of matter, which interacts via gravity but not the electromagnetic force, thereby eliciting the name dark matter [15][16]. Another example of experimental results not explained by the SM are neutrino masses. The SM is derived under the assumption that neutrinos are massless. In 1998 the Super-Kamiokande neutrino detector discovered evidence for neutrino oscillation, which requires neutrinos to have a nonzero mass [17]. Since neutrinos in the SM do not acquire their mass through the Higgs mechanism, this mass presents an issue with the consistency of the theory [18].

Lastly, a theoretical problem in the SM is the hierarchy problem. As described earlier, according to the SM, particles obtain their mass through interactions with the Higgs field. However, the Higgs boson itself obtains large quantum corrections because of virtual particles such as virtual top quarks. To avoid a huge Higgs mass, parameters of the Higgs in the SM therefore need to be manually changed and fine-tuned, which is an unnatural process [6].

One way to solve some of the problems mentioned above is to make experimental observations of particles and phenomena that are not yet included in the SM, but predicted by theories that could complement it. For example, a direct or indirect observation of dark matter could verify a theory that would complement the SM, and explain astronomical observations. Looking for new physics phenomena confirming or falsifying theories beyond the SM can be done in different ways, section 2.2 will discuss how the dijet searches at the ATLAS experiment can help in this search for new physics. Dijet searches are one of the most direct ways of looking for new physics at a hadron collider, since if the new particle is produced from proton-proton collision through a strong interaction involving quarks and gluons, then it will have to decay into quarks and gluons which creates jets.

2.2 Finding New Physics

Discovery is to observe something that differs from what is known. In physics this is to observe a difference between the data collected and existing theory, having excluded any instrumental effects. When looking for new particles, those particles are often very unstable and decay too quickly to be detected directly. These are called resonances, particles that form as intermediate steps between a collision and the final products. To indirectly observe these resonances and calculate their mass, the energy and momentum of their decay products is measured and used to calculate what is called the resonances' invariant mass [19].

2.2.1 Four Vectors, Invariant Mass and the Standard Model Background

The vector describing a relativistic particle in space-time is represented by the four-dimensional vector (ct, x, z, y), called a four-vector, where c is the speed of light, t is time, and x, y, z are the spatial coordinates of the particle.

Similarly to space-time, energy and momentum of a relativistic particle can also be combined into a four-vector.

$$p_{\mathrm{\mu}} = (E, p_x, p_y, p_z) = (E, \bar{p}) \tag{3}$$

Where E is the energy of the particle and $\bar{p} = p_x, p_y, p_z$ is the momentum in respective spatial dimension.

The modulo square of a four-vector for a single relativistic particle is $p_{\mathrm{\mu}}^2 = E^2 - \bar{p}^{\ 2}$. When in the frame of reference of this particle, the particle is at rest and only has energy equivalent to its rest mass. Since this relation must apply in every frame of reference, whether the particle is at rest or not we get:

$$p_{\mathrm{\mu}}^2 = E^2 - \bar{p}^{\ 2} = m_0^2$$

Where m_0 is the rest mass of the particle [19]. $p_{\mathrm{\mu}}^2$ is then called the invariant mass of a particle, which is invariant of the frame of reference.

The total four-momentum of a particle collision involving i incoming particles and j outgoing particles is conserved in the initial and final state, as a consequence of energy conservation [19]:

$$\sum_i p_{\mathrm{\mu}i} = \sum_j p_{\mathrm{\mu}j} \tag{4}$$

The mass squared m^2 of a particle that decays, which is equivalent to its rest mass, is given by the four-vector addition of the four-momenta of its decay products:

$$m^2 = \sum p_{\mu j}^2$$

Where m is the invariant mass of the mother particle and $p_{\mu j}$ the four-vectors of the integer number j decay products.

In this thesis, the four-vectors of the particles involved in collision events are represented by the ROOT framework's [20] TLorentz vector, and the invariant mass was calculated by the sum of the final state particles' TLorentz vectors [21].

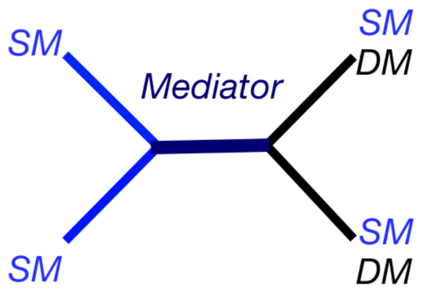

Figure 2: A process in which two colliding SM particles create an unknown mediator particle which decays into either SM particles or dark matter particles. Such a mediator could be detected as a resonance [22].

As mentioned earlier, one way to complete the SM is to make observations of dark matter. One way to look for signs of dark matter in colliders is to look for a possible mediator particle, which propagates the force connecting dark matter particles with SM particles. Such an unknown mediator, Z', could be a resonance which decays into either other dark matter particles or SM particles which eventually produce jets. This process is visualized in Figure 2. By theorizing that Z' acts as a resonance, one would therefore look for its invariant mass by observing its decay products. Z' has a possibility of being created in high energy collision events, meaning that its invariant mass can be measured with the ATLAS detector during proton collisions. Since Z' is expected to exist as an intermediate step in a $q + \bar{q} \to q + \bar{q}$ process we focus on collision processes such as $p + p \to j + j$. If the final state jets are decay products as a result from a Z' decay this will show as a sharp peak of event counts at the Z' mass in the distribution of invariant masses. If the final state jets do not come from the same decay, i.e any other process that produces two jets without coming from a decay , this will not contribute to a peak. It would rather contribute to a smooth distribution partly representing the unpredictable momentum transfer that occurs in pp and other QCD events [6][19].

The Standard Model background (SMB) is similar to the smooth distribution mentioned above. The SMB consists of a collection of QCD events produced by phenomena that are already included in the SM. QCD predicts this background to be a smooth, steeply decreasing distribution as a function of increasing collision energy. This is the SMB of dijet searches investigated in this thesis [4].

3 Tools

3.1 ATLAS

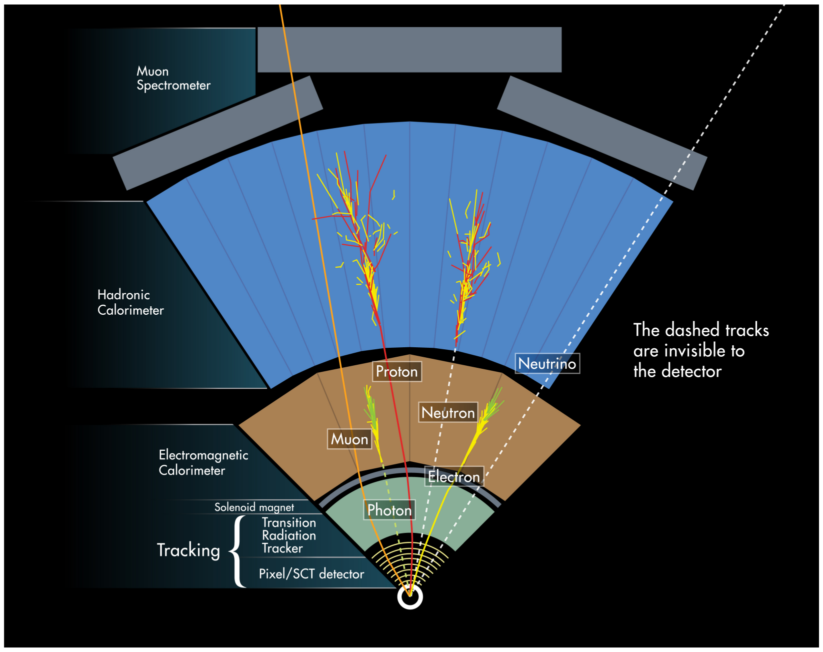

Figure 3: A cross-section view of the ATLAS detector. It shows the inner tracking detector layer, the electromagnetic calorimeter, the hadronic calorimeter and the muon spectrometer in a concentric formation. Also included are the particles detected by each layer [23].

The LHC is a circular particle accelerator which accelerates particles in two counter-rotating beams and collides them at multiple points [24]. ATLAS is a concentric detector at one of these collision points, it encloses the collision point with magnets and different layers of detector types to record the trajectory, momentum and energy of particles from the collision events [25][26]. Closest to the collision is the inner detector which uses semiconductor detectors (pixels and strips) and a transition radiation tracker to measure the momentum and charge of particles from the collisions [27]. The second concentric layer is the electromagnetic calorimeter. A calorimeter attempts to stop a particle so that its energy can be absorbed and measured. The electromagnetic calorimeter uses this method to measure the energy of photons and electrons [28]. The third layer is the hadronic calorimeter which measures the energy of hadrons in a similar way [28]. The final layer is the muon spectrometer. Muons produced at the LHC are for the most part minimum ionizing particles and therefore difficult to detect and measure with the other layers. They will therefore penetrate and reach this layer, which uses drift tubes and tracking chambers, to measure the momenta of muons [29].

In ATLAS, the calorimeters detect most of the energy from these jets. Using the calorimetry and tracking, so-calleld jet algorithms are used to group the data into four-vectors to represent the jets. This is one method that allows us to calculate the invariant mass of a decayed particle [30]. This thesis will use the invariant mass distribution in dijet events as used by Ref. [4] as observable for testing the performance of the fit, and to help generate simulated data from the expected QCD background using the MadGraph event generator[31].

3.2 MadGraph

The simulated particle collisions were created using MadGraph5_aMC@NLO (MG) [32]. MG is a framework that provides SM phenomenology that can generate simulated particle collision events [32][31]. In this thesis, MG was used to create 100 individually simulated data sets of particle collisions. These data sets are referred to as simulated collider runs. These runs generated proton-proton collision events with two jets of particles, dijet, as final products. The events were generated with a minimum invariant mass of a jet pair of 1100 GeV. The final number of events used from each run was 662681 because this was the number of events in the ATLAS data [4]. The MG event generation was performed on a cluster using the MG built-in grid-pack feature. The parameters for the event generation, such as minimum invariant was delivered by the run card file found in Appendix 10.8. The execution of the runs on the computer cluster was managed by the delphes-pilot program [33].

It is important to note that these events only generates quarks or gluons, which represent jets, as final state products. Furthermore, in a more realistic event generation this data would be passed through hadronization and fragmentation simulations which would then cluster data into actual jets. An even better simulation would also include detector simulations on top of this; to simulate the jets' interactions with the detector.

3.3 Splines

It is in the interest of a new SMB representation to make as few assumptions as possible about the shape of the background. Previous research has used a functional form of the SMB representation which had characteristics that depend on assumptions made about the data [4]. As mentioned before, the SMB without a resonance is expected to be a smooth distribution of invariant masses. Similar to other natural phenomena it is also expected to be continuous. With these criteria in place there is a need for a flexible, smooth and continuous function.

When linear, quadratic or cubic polynomials are not flexible enough to accurately represent data, one can resort to higher degree polynomials. However, at higher degrees than 3 the meaning of the polynomial is hard to interpret as an actual physical behavior. Oscillations at end points also render a high degree polynomial useless for interpolation, known as Runge's phenomenon [34]. Instead of increasing the number of degrees one assumes that linear, quadratic and cubic polynomials are sufficient to fit small ranges of data within a larger set. To represent a natural phenomenon one also wants the resulting fitting function to be smooth, often referred to as C^2 continuous. C^2 continuous means that the first and second derivatives are continuous. With this piece-wise polynomial representation, data which cannot be accurately represented by low-degree polynomials can be represented by a spline. A spline is a function, that is piece-wise defined by polynomial functions and is C^2 continuous.

The places where the polynomials connect are called knots. A higher order spline is given by:

$$y = \beta_0 + \beta_1 x + ...\beta_D x^D + \sum_{k=1}^{I-1} b_k (x - \xi_k)_+^D \tag{5}$$

Where D is the degree of the spline, K is the set of knots, I is the number of intervals, ξ is the value of the current knot and

$$(x - \xi_k)_+^D = \begin{cases} 0 & x < \xi \\ (x - \xi)^D & x > \xi \end{cases} \tag{6}$$

is a truncated polynomial of degree D. These terms only affect the fit of the function to the left of ξ. The first $D - 1$ derivatives are continuous over the entire range of data and the D^{th} derivative is continuous over the whole range except at the K knot points. This means that the spline is very flexible since the fit is allowed to be local to small regions of the data. On the other hand, truncated polynomials can lead to unstable fittings if two knots come too close in proximity. To solve this, another set of functions, B-splines, is used to replace the truncated polynomial [8].

3.3.1 B-spline Basis

Instead of fitting with truncated polynomials, a better solution is to fit with a set of functions which produce the same range of curves as the truncated polynomials. The most common set is the set of B-splines [8]. B-splines are C^2 continuous and have local support. Local support means that at any point x in the domain of the spline there are at most $D + 1$ number of basis functions that are nonzero. This property makes it easier and faster to fit and evaluate a spline. A B-spline curve defined everywhere on the real number plane can be written in the form:

$$S_{D,t} = \sum_{i=0}^{n-1} c_i B_{i,D}(x; t) \tag{7}$$

Where D is the degree of the spline, t is the knot vector and n is the number of basis functions. This is a B-spline of degree D on the domain $[t_0, t_{n+p}]$. The basis functions determine how much the curve is affected at a knot t. For B-splines with uniform knot placement, the basis functions must satisfy:

$$B_i(t) = B(t - i) \tag{8}$$

With this condition, the basis function is chosen such that a B-spline is constructed with C^2 continuity and control points which only affect the curve locally [35]. The B-splines constructed in this thesis have non-uniform knot placements. They have a special condition on the knot vector for interpolation called 'Schoenberg-Whitney nesting conditions', however, this is beyond the scope of this thesis [36].

3.3.2 Penalization

While it could be tempting to increase the number of knots in order to increase smoothness, a large number of knots will result in the data becoming over fitted. Overfitting here means that the model describes random errors and noise, not the underlying distribution. On the other hand, if the number of knots is too small the fit becomes unrepresentative of the data. Whilst keeping the number of knots constant and high, another method can be used to make the fit smooth but also making sure that the data isn't over fitted. A B-spline becomes smoother as the difference between its coefficients decreases. An effective penalization parameter will therefore penalize the difference of the coefficients. For a cubic B-spline, which is what is used in this thesis, the second difference of the coefficients is penalized. This is represented by a second-order difference penalty in the B-spline, creating a penalized B-spline called a P-spline. The details of this penalty is beyond the scope of this thesis.

P-splines like this are implemented to fit the data in this thesis using the python library SPLINTER. For information on the exact implementation, see the SPLINTER library [9]. In the SPLINTER library, the penalization is represented by a single parameter α. If α is small there is no restriction concerning smoothness and the P-spline is likely to fit all data points perfectly. If α is large the fit is highly penalized and is very close to a polynomial of the degree of the spline [35].

3.4 Chi-squared

To test if the P-spline is a good representation of the SMB or test its outlook for being used in dijet searches, it is important to know how well it fits the data. A figure of merit that can quantify the agreement between the fitted P-spline and the data is the χ^2 test. χ^2 can be used as a goodness-of-fit test where it determines whether a frequency distribution deviates from a theoretical distribution. It is therefore used here as a goodness-of-fit statistic to test how well the P-spline represents the MG simulations and the ATLAS data. Pearson's chi-squared test is defined as:

$$\chi^2 = \sum \frac{(E_i - O_i)^2}{\sigma_i^2} \tag{9}$$

Where O_i is the observed value from the experimental data in each of the bins, σ_i the experimental error and E_i the expected value, i.e. the value of the P-spline at a respective x. Since the observed values are number of events that follow Poisson statistics, the error on the observation is $\sqrt{O_i}$, so the χ^2 reduces to:

$$\chi^2 = \sum \frac{(E_i - O_i)^2}{O_i} \tag{10}$$

If the P-spline fit represents the data, then the statistic χ^2 follows the χ^2 probability density function. The χ^2 probability density function has an expectation value equal to the number of degrees of freedom, n_d. So if $\chi^2 = n_d \Rightarrow \frac{\chi^2}{n_d} = \frac{n_d}{n_d} = 1$ then the fit is good [37].

In this thesis the goodness-of-fit is determined by defining χ^2 as:

$$\chi^2 = \frac{\sum \frac{(y_{data} - y_{spline})^2}{y_{spline}}}{n-1} \tag{11}$$

The data used are histograms, so here y_{data} is the number of counts of a certain invariant mass bin from the MG or the ATLAS data, and y_{spline} is the respective value from the P-spline. In this case, for simplicity, we consider the number of degrees of freedom as the number of data points minus the number of fit parameters. Other more sophisticated methods are outside the scope of this thesis [38]. It was complicated to determine the number of fit parameters of the spline so the penalization parameter, α, represents the only fit parameter. The number of degrees of freedom is therefore the number of data points, n, minus one.

The penalization parameter, α, of a fitted P-spline is called an optimal α if its corresponding P-spline produces a $\chi^2 = 1$ with the fitted data.

4 Method

To test how well the P-spline represents the SMB and to test its future outlook for working with ATLAS dijet searches, fitting methods 1 and 2 were devised respectively. In fitting method 1, the P-spline was fitted to 100 MG runs with an α such that the average χ^2 of all runs was equal to 1. The distribution of individual χ^2 will show whether the P-spline manages to create a good fit over the whole range of MG runs. In fitting method 2, the ATLAS data and 100 MG runs were fitted using individual optimal α. A distribution of optimal α will show how different the MG runs are compared to the ATLAS data from the perspective of how flexible the spline has to be to obtain a $\chi^2 = 1$. Both fitting methods are represented as flow-charts in fig 5.

The code developed can be found in updated versions on its GitHub repository [39]. The code used in this thesis, run5.sh, test5.cpp, pspline8.py, and library.py can also be found in Appendix 10.5, 10.6, 10.7.1, and 10.7.2 respectively. The code is dependent on the SPLINTER library [9]. Further-

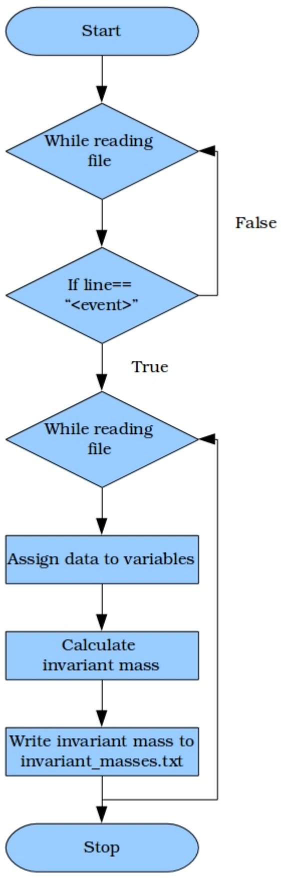

Figure 4: Flowchart of test5.cpp which is the code that calculates the invariant mass from the MadGraph simulated collisions.

more, work on this thesis used MG version 2.5.2, gcc (Ubuntu 5.4.0-6ubuntu1 16.04.4) 5.4.0 20160609, ROOT version 5.34/36, Python 3.5.2 (default, Nov 17 2016, 17:05:23) [GCC 5.4.0 20160609] on linux and SPLINTER version 3.

4.1 Software for Invariant Mass Calculation

With the simulated runs organized, the MG data was processed by the C++ program test5.cpp. This program iterated through each event in the run, obtained the four-vector for the two final state quarks/gluons, and represented them as ROOT TLorentz vector objects [21]. The invariant mass for each event was calculated by the magnitude of the sum of the two TLorentz vectors using the ROOT function .M() [21]. The magnitude of a four-vector is: $|v|^2 = v \cdot v = t \cdot t - x \cdot x - y \cdot y - z \cdot z$

This process is represented as a flowchart in Figure 4. The invariant masses of all runs are then added to the same text file "invariant_masses.txt". This file is then truncated to 662681 invariant masses into the file "invariant_masses_final.txt". 100 invariant_masses_final.txt was created, one for each run.

4.2 Spline Fitting

The python program "pspline8.py", its library library.py, and the SPLINTER library are responsible for manipulating α, fitting the P-spline, evaluating χ^2, and plotting the results. For fitting method 1, shown as a flowchart in Figure 5, pspline8 iterated through all runs and fitted them with a P-spline with an initially arbitrary α. The average χ^2 of this run was then calculated according to equation (11). If average $\chi^2 < 0.999$, then $0.5 \cdot K$ would be added to α. If average $\chi^2 > 1.001$ then $0.5 \cdot K$ would be subtracted from α. When 0.999<average χ^2<1.001, all the current χ^2 values were

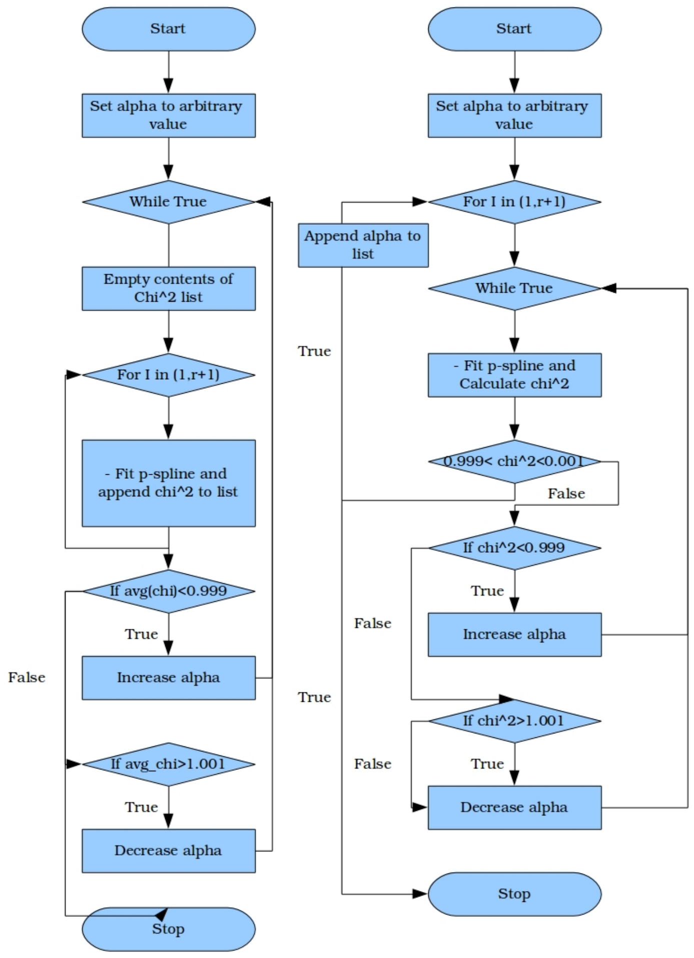

Figure 5: Flowcharts of fitting method 1 and 2, from the left. Fitting method 1 determines the distribution of χ^2 for a α that creates an average $\chi^2 = 1$. Fitting method 2 determines the distribution of optimal α. r is the number of runs

plotted in a histogram. Pspline8 was prevented from jumping back and forth between α values by decreasing the factor K each time the operator on α was changed between addition and subtraction.

Fitting method 2 created the α distribution by iterating all MG runs and finding an optimal α for each run individually. It used a similar method as in fitting method 1 to obtain all optimal α. All optimal α were stored in a list. This process is represented as a flowchart in Figure 5. The list of optimal α was then put into a histogram and plotted. The ATLAS data was also fitted using an optimal α, found in the same way. A tick mark indicates the ATLAS optimal α location in the histogram.

5 Results

Fitting method 1 was devised to determine if the P-spline is a good representation of the SMB from an MG model.

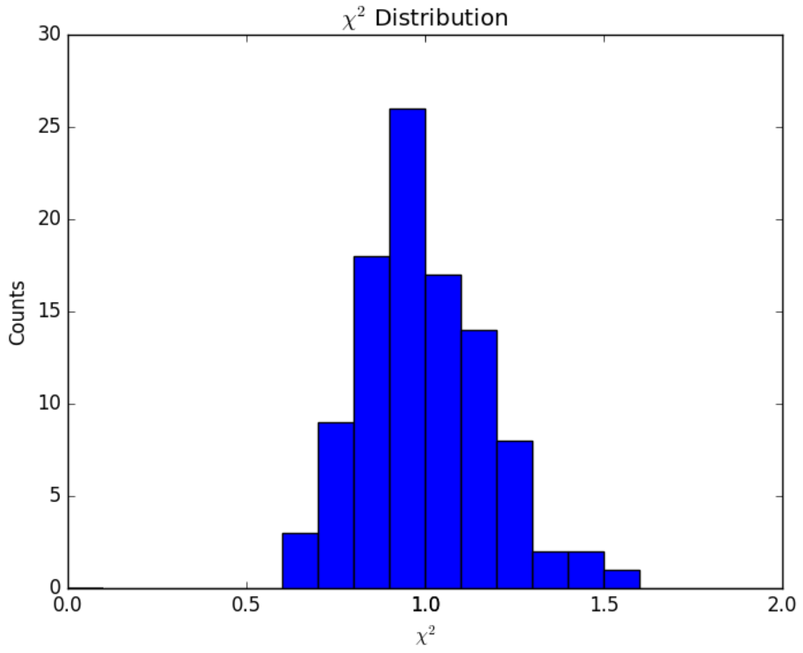

Figure 6: The distribution of χ^2 from fitting method 1. This shows that the χ^2 distribution peaks at 1, meaning that the P-spline manages to fit a large range of MG generated runs with a good fit. This was obtained with $\alpha = 1.0$.

Figure 6 shows the distribution of χ^2 from fitting method 1. This shows that the χ^2 distribution peaks at 1, meaning that the P-spline manages to fit a large range of MG generated runs with a good fit. For an average $\chi^2 = 1$ is $\alpha = 1.0$.

Fitting method 2 was devised to determine how well the P-spline behaves while fitting actual ATLAS data and compare that fit to the MG fits.

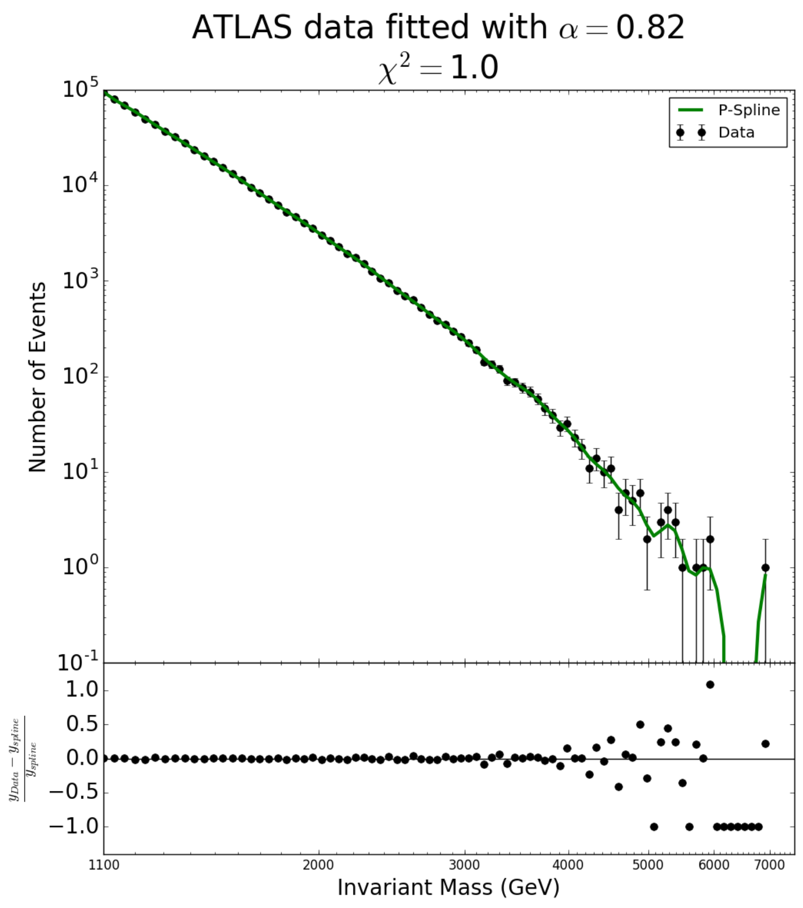

Figure 7: The ATLAS data fitted with a P-spline using its optimal $\alpha = 0.82$. The error bars are equal to $\sqrt{y_{spline}}$. The P-spline does however not take the errors into account.

Figure 7 shows the ATLAS data fitted by a P-spline with an optimal α. The errors are equal to the square root of the fit value. However this error is not recognized by the P-spline. Figure 8 shows the same fit of the ATLAS data but on a different scale and without using a logarithmic scale. This shows that the apparent discontinuity of the P-spline seen in Figure 7 is not a discontinuity, but rather an effect of using a log-scale. Knowing that the P-spline is continuous and that $\chi^2 = 1$, it can be determined that the P-spline manages to create a good fit of the ATLAS data. For the ATLAS data optimal $\alpha = 0.82$.

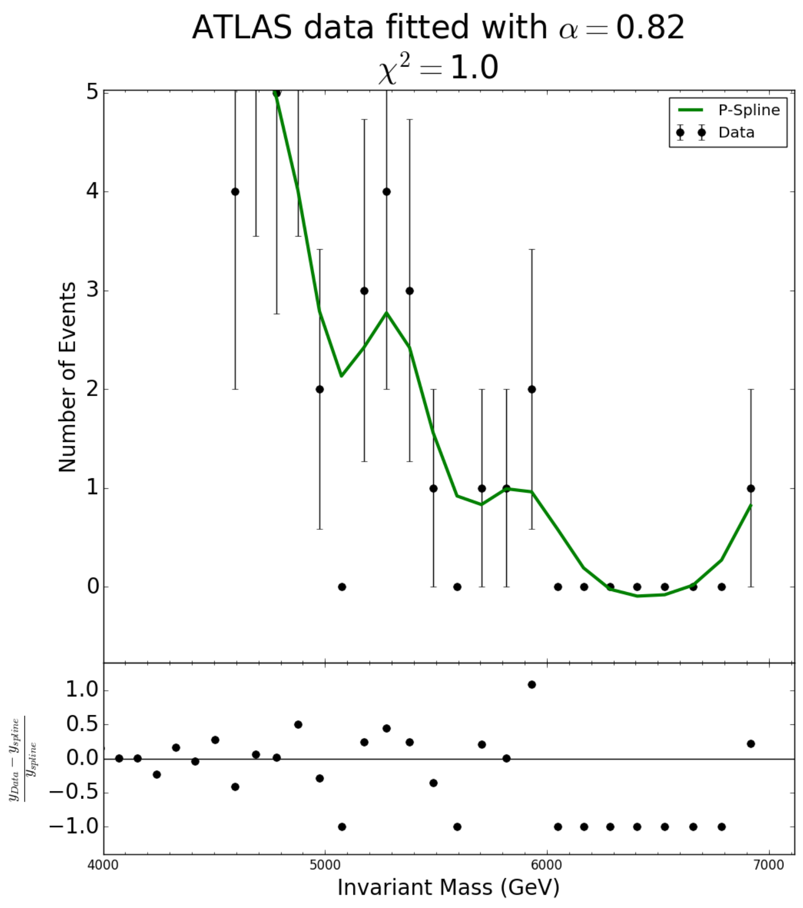

Figure 8: A closer scale without log-scale of the ATLAS data fitted with an optimal $\alpha = 0.82$. This shows that the P-spline is continuous in this range.

In Figure 8, it can be seen how the zero counts of the histogram are estimated by the P-spline as negative counts. It can also be observed that the P-spline on this scale appears to not be C^2 continuous at the knots. This is because the visual representation of the P-spline was only evaluated at the corresponding ATLAS data points to speed up the process; the P-spline is still C^2 continuous. This is demonstrated further in Appendix 10.1.

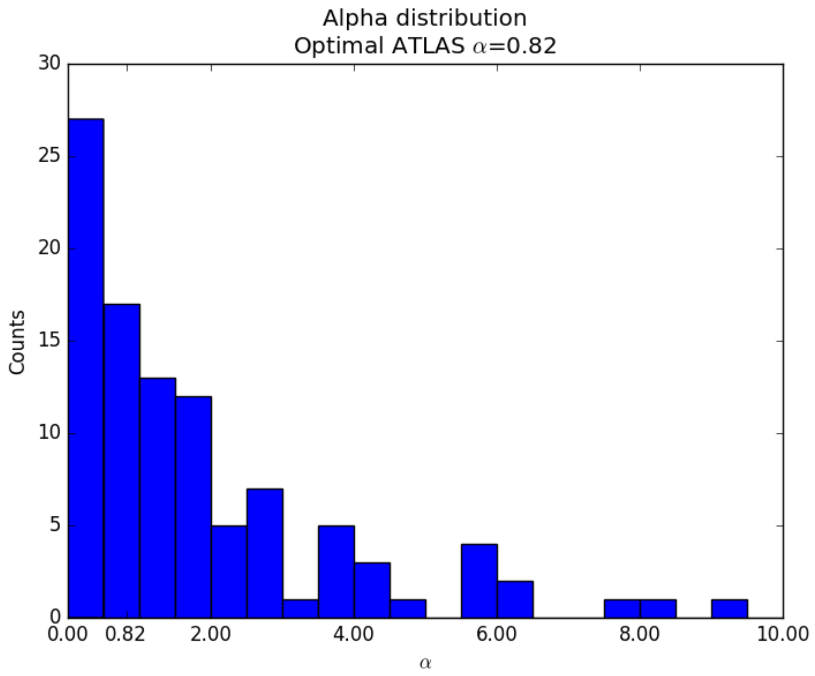

Figure 9: The distribution of optimal α for all the MG runs. A tick mark is added at the ATLAS optimal $\alpha = 0.82$.

The distribution of optimal α values for the MG runs is positively skewed as seen in Figure 9. The optimal α of the ATLAS data is marked in this distribution at 0.82. From a qualitative analysis, it is seen that the optimal ATLAS α is within the majority of the α distribution. This suggests that there is not a large deviation between the ATLAS data and the MG generated runs in how flexible the P-spline needs to be in order to create a good fit.

6 Conclusion and Discussion

The aim of this thesis was to develop and test the P-spline's ability to represent the SMB in dijet events from current MG models and ATLAS data. Since the distribution of χ^2 from all the MG runs peaks at 1, it can be concluded that the P-spline can represent the SMB well. It was further investigated how the P-spline would handle ATLAS data. The P-spline had no problem in making a good fit with the ATLAS data, and the optimal α distribution shows that for the P-spline, the MG runs and the ALTAS data are not significantly different. This final test is however not concluding if the P-spline can be used as a tool in particle searches, it only shows that it can produce a good fit with the actual data.

There are limitations to the implementation and testing of the P-spline used in this thesis. In certain situations when confronted with zero counts from histograms, the P-spline will estimate negative counts. This is unphysical. The current implementation also does not take uncertainties of the data into account nor does it use a prorperly investigated number of degrees of freedom for the goodness-of-fit test. Further limitations of this research include the generation of MG events. The events generated here did not undergo shower simulations or detector simulations. This means that they do not fully represent the current SMB. All these limitations should be noted and left for improvement in later work. Development of software used in this thesis takes us one step closer to the bigger goal of determining if a P-spline can be a useful tool in aiding the search for new particles. This thesis works as a proof-of-principle that a P-spline can be used to estimate the SMB in such searches.

This thesis concludes that the P-spline can represent the SMB and fit ATLAS data with a good fit. However, further research needs to be done where the P-spline is possibly tested with signal injection in order to conclude if the P-spline can be used as a tool to aid particle searches.

7 Outlook

As a response to the limitations of this research, improvements are left to future work, namely:

Uncertainties The biggest flaw of the P-spline implementation used in this thesis is its inability to recognize the uncertainties associated with the data points. The lack of this feature was raised to the SPLINTER developers and has now been implemented into the library as weighted least squares. For future implementation of P-splines as both representation of the SMB and tools to find signals in data, it is recommended to correlate data uncertainties to a weight, and fit the P-spline with those parameters.

Goodness-of-fit parameters This work used a simplified way of determining the number of degrees of freedom of the P-spline. This will affect the goodness-of-fit test and should in future work be researched more in depth and applied appropriately.

Estimation of negative counts Current implementation estimates negative counts for zero count values. A solution to this would be to have a histogram specific fitting mode where zero counts are dealt with in a special way. This could be implemented by using a large weight, mentioned above, for zero count data points.

Event generation and detector simulation It is important that the MG generated events are as true to the current theory as possible. It is therefore important to include proper simulations so that the data can mimic the experimental setup. This was not accomplished in this thesis since the MG events generated did not undergo particle shower nor detector simulations. In future testing of the P-spline, particle shower and detector simulations should be added to the generation of MG events.

Signal injection In order to properly test the P-splines ability to act as a tool aiding the search for new physics, the P-spline must be tested on data which has background and a known signal. A way of doing this is to create signals in background data (signal injection) and investigate the limits of the P-splines ability to not include the signal in its estimation of the background.

8 Acknowledgements

I want to thank Daniel Whiteson for making this incredible opportunity of researching with his team in the U.S.A. possible. Thank you for taking time from your busy schedule to supervise me through this research and introducing me to the field.

I want to thank Caterina Doglioni for going above and beyond to make this a great thesis. But also for being such a great lecturer and researcher, you are the reason that I got hooked on particle physics.

Huge thanks to Bjarne Grimstad, who has developed the wonderful SPLINTER library that is used in this thesis. For obvious reasons, this thesis would not be possible without your work and without your work and your support of open source. I would like to thank you further for taking time out of your busy schedule to help me both implementing the SPLINTER library, developing the library further to fit future needs and especially for helping me first hand to understand splines and correct its corresponding section in this thesis.

I want to thank Meghan Frate for being such a patient colleague, helping me understand my task at hand, helping me find the ALTAS data and for giving me her code to make pretty plots.

I want to thank Ying Wun Yvonne Ng for all the help she has given me throughout this academic year. Without you I would not even have been able to get this project off the ground.

I want to thank Kevin Bauer for saving this project after a 2 month drought. Thank you for sitting down with me to implement yours and Chase's code and get my MadGraph simulations going.

I want to thank Mo Abdullah for always helping me with the tricky questions and for the very interesting discussions in the office.

I want to thank my wonderful family Marita, Anders and Matilda Ekman for always supporting me and shaping me into who I am today. None of this would have been possible without you.

I want to thank Roger Chang for proof-reading and helping me with the style and grammar of this thesis.

Last but not least I would like to thank Jenny Lin enormously for showing great support and compassion during the ups and downs of this thesis. Further more I would like to thank you for making my year here in the U.S.A the best year of my life.

9 References

References

[1] The Standard Model (2012). URL `http://cds.cern.ch/record/1997201`.

[2] Boveia, A. Dark Matter Searches at ATLAS (2016). URL `https://indico.cern.ch/event/473193/attachments/1261288/1868948/2016426-cern-seminar-boveia.pdf`.

[3] ATLAS: The largest volume particle detector ever built (2012). URL `http://cds.cern.ch/record/1997264`.

[4] ATLAS Collaboration. Search for new phenomena in dijet mass and angular distributions from pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector. *Physics Letters B* (2015).

[5] Robert M. Harris, K. K. Searches for Dijet Resonances at Hadron Colliders. *International Journal of Modern Physics A* (2011).

[6] Martin, B. & Shaw, G. Particle Physics, third edition. *John Wiley & Sons LTD, West Sussex* (2008).

[7] Judd, K. L. Numerical Methods in Economics. *MIT Press* (1998).

[8] Schwarz, C. J. An Introduction to Splines URL `http://people.stat.sfu.ca/~cschwarz/Consulting/Trinity/Phase2/TrinityWorkshop/Workshop-material-Simon/Intro_to_splines/`.

[9] Grimstad, B. *et al.* SPLINTER: a library for multivariate function approximation with splines (2015). URL `http://github.com/bgrimstad/splinter`.

[10] Fehling, D. The Standard Model of Particle Physics: A Lunchbox's Guide. *The Johns Hopkins University* (2008).

[11] O'Luanaigh, C. The basics of the Higgs boson (2013). URL `http://cds.cern.ch/record/1998007`.

[12] Wilczek, F. QCD Made Simple. *Physics Today* (2000).

[13] Atkin, R. Review of jet reconstruction algorithms. *J. Phys.: Conf. Ser. 645 012008* (2015). URL `https://atlas.cern/discover/detector/muon-spectrometer`.

[14] Womersley, J. Beyond The Standard Model. *Symmetry* **volume 02, issue 01** (2005).

[15] Corbelli, E. & Salucci, P. The Extended Rotation Curve and the Dark Matter Halo of M33. *Monthly Notices of the Royal Astronomical Society* (1999).

[16] Dark matter (2012). URL `http://cds.cern.ch/record/1997200`.

[17] The Super-Kamiokande Collaboration, Y. F. e. a. Evidence for oscillation of atmospheric neutrinos. *Physical Review Letters* (1998).

[18] S.F.King. Neutrino Mass Models: a road map .

[19] Jönsson, L. Lectures in Particle physics, latest revision 2016 .

[20] ROOT a Data analysis Framework URL `https://root.cern.ch/`.

[21] TLorentzVector Class Reference URL `https://root.cern.ch/doc/master/classTLorentzVector.html`.

[22] Doglioni, C. & Boveia, A. EFTSimplifiedModels URL `https://commons.wikimedia.org/wiki/File:EFTSimplifiedModels.pdf`.

[23] Pequenao, J. & Schaffner, P. An computer generated image representing how ATLAS detects particles. *CERN-EX-1301009* (2013). URL `https://cds.cern.ch/record/1505342`.

[24] The Large Hadron Collider (2014). URL `http://cds.cern.ch/record/1998498`.

[25] ATLAS: The largest volume particle detector ever built (2012). URL `http://cds.cern.ch/record/1997264`.

[26] Detector Technology URL `https://atlas.cern/discover/detector`.

[27] The Inner Detector URL `https://atlas.cern/discover/detector/inner-detector`.

[28] Calorimeter URL `https://atlas.cern/discover/detector/calorimeter`.

[29] Muon Spectrometer URL `https://atlas.cern/discover/detector/muon-spectrometer`.

[30] Salam, G. P. Towards Jetography. *Eur.Phys.J* (2009). URL `https://atlas.cern/discover/detector/muon-spectrometer`.

[31] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H.-S. Shao, T. Stelzer, P. Torrielli and M. Zaro". The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. *CERN-PH-TH/2014-064* (2014). URL `https://arxiv.org/pdf/1405.0301.pdf`.

[32] Welcome to the MadGraph5_aMC@NLO Wiki URL `https://cp3.irmp.ucl.ac.be/projects/madgraph/`.

[33] Shimmin, C. Delphes-pilot URL `https://github.com/cshimmin/delphes-pilot`.

[34] Lecture 3: The Runge Phenomenon and Piecewise Polynomial Interpolation URL `https://www.math.ubc.ca/~peirce/M406_Lecture_3_Runge_Phenomenon_Piecewise_Polynomial_Interpolation.pdf`.

[35] Zorin, D. Bezeier Curves and B-splines, Blossoming. *New York University* (2002).

[36] Lyche, T. & MØrken, K. Spline Methods Draft. *Department of Informatics Centre of Mathematics for Applications University of Oslo* (2011).

[37] Cowan, G. Introduction to Statistics - Day 4. *CERN Summer Student Lectures on Statistics, 2011* (2011).

[38] Cantoni, E. & Hastie, T. Degrees-of-freedom tests for smoothing splines. *Biometrika* (2002).

[39] URL `https://github.com/exook/p-spline-SMB-representation`.

10 Appendix

10.1 Smoothness of Fit

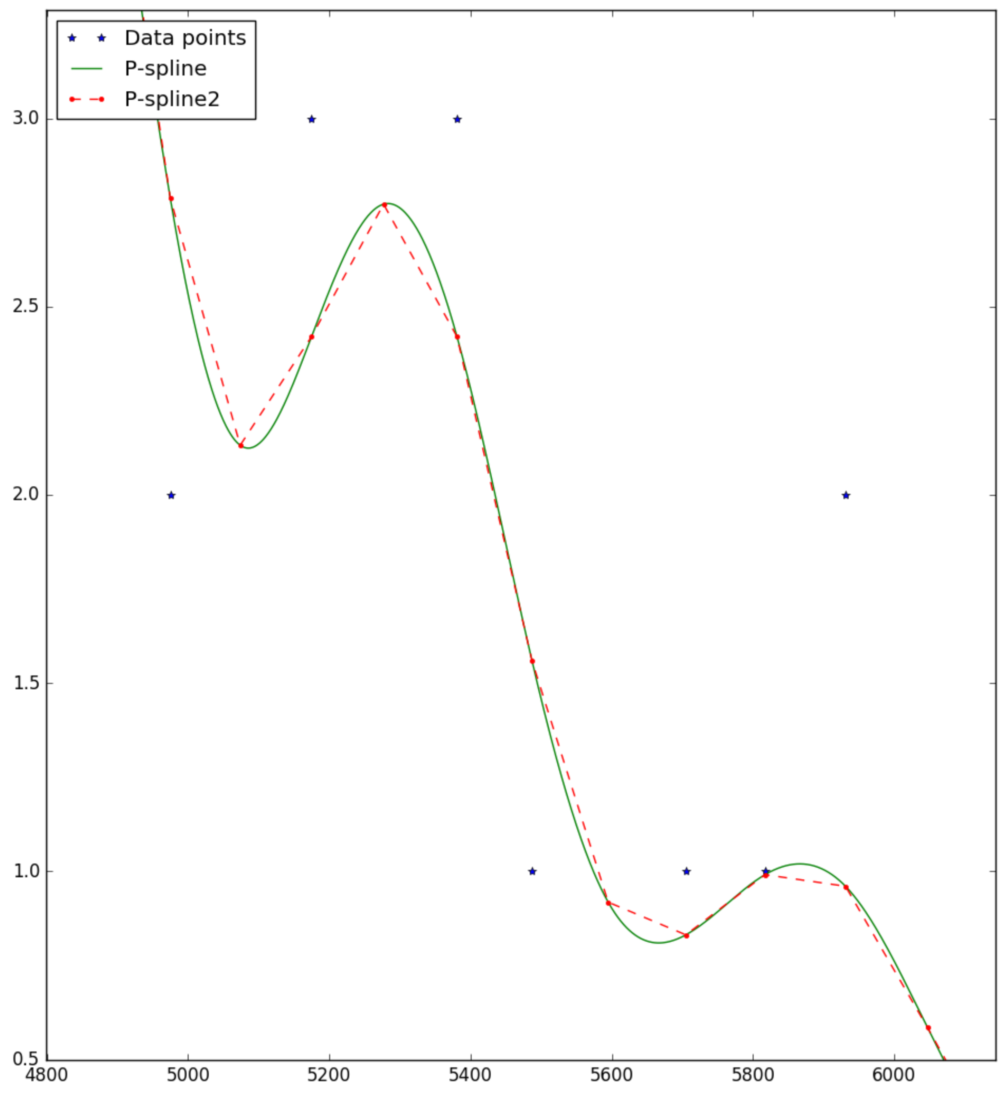

Figure 10: A closeup of the ATLAS data fitted with two equal P-splines with $\alpha = 0.82$. The dotted fit is evaluated only at the data points, the full line fit is evaluated for each integer value in the domain of the spline. This shows that even though the P-spline presented in this research is jagged, there is a proper C^2 continuous spline behind the visual representation.

10.2 SPLINTER

The SPLINTER library can be found on: `https://github.com/bgrimstad/splinter`

10.3 Delphes-pilot

The program that managed the MG event generation on the cluster can be found here: `https://github.com/cshimmin/delphes-pilot` Some minor adjustments were made to this code inorder to make it work on my cluster partition and to not run the Delphes and Pythia simulations.

10.4 Code Developed

Updated versions of all the code developed in this thesis will be found on: `https://github.com/exook/P-spline-SMB-representation` The versions of the code used to obtain the results in this thesis can be found in the following subsections.

10.5 run5.sh

This program managed the execution of test5.cpp and iterated through the specified raw data from MG.

```
#!/bin/bash
#run inside python folder

start_path=/home/alexander/Desktop/splinter/build/splinter-python/python/workarea3

#start_path=/media/alexander/INTENSO/

#run_test_iterator=0

#for j in 1 2 3 4 5 6 7 8 9 10
for j in 91 92 93 94 95 96 97 98 99 100
do



for i in 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
#for i in 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
do

echo $j,$i

cd $start_path/gp_events/run_$j/run_test_$i
```

```
gzip -d unweighted_events.lhe.gz

$start_path/test5

sed '$d' invariant_masses.txt >> $start_path/gp_events/run_$j/invariant_masses.txt
done
sed -n -e '1,662681p' $start_path/gp_events/run_$j/invariant_masses.txt > $start_path/gp

done
```

10.6 test5.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <cmath>
#include <math.h>
#include "TLorentzVector.h"
#include "TH1F.h"
#include "TFile.h"
#include <stdio.h>
using namespace std;
int main()
{
int Np;
int ID_1;
int state_1;
int mother1_1;
int mother2_1;
int color1_1;
int color2_1;
float px_1;
float py_1;
float pz_1;
float E_1;
float mass_1;

int ID_2;
int state_2;
int mother1_2;
int mother2_2;
int color1_2;
int color2_2;
float px_2;
float py_2;
float pz_2;
float E_2;
float mass_2;

int ID_3;
int state_3;
int mother1_3;
int mother2_3;
int color1_3;
int color2_3;
float px_3;
float py_3;
float pz_3;
float E_3;
float mass_3;

int ID_4;
int state_4;
int mother1_4;
int mother2_4;
int color1_4;
int color2_4;
float px_4;
float py_4;
float pz_4;
float E_4;
float mass_4;

float ParticleMass;
string start;
string a;
string b;
string c;
string d;
string e;
string f;
string g;
string h;
string i;
string j;
string k;
string l;
string m;
string n;
string o;
string p;
string q;
string r;
string s;
string t;
string u;
string v;
string w;
string x;
string y;
string z;
string a1;
string a2;
string a3;
string a4;
string a5;
string a6;
string a7;
string place1;
string place2;
string str;
str="<?>";

string data_file_number;
string line;
ifstream data("unweighted_events.lhe",ios::in);
ofstream writer("invariant_masses.txt" , ios::trunc);//write data
if(data.is_open())
{
while(data.good())

{
getline(data,line);
//writer<<line<<endl;
if (line=="<event>")
{
```

```
cout<<"it works"<<endl;
break;
}
}
while(data.good())
{
data>>start>>Np>>place1>>a>>place2>>b>>c>>
ID_1>>state_1>>mother1_1>>mother2_1>>
color1_1>>color2_1>>px_1>>py_1>>pz_1>>E_1>>d>>e>>mass_1>>

ID_2>>state_2>>mother1_2>>mother2_2>>
color1_2>>color2_2>>px_2>>py_2>>pz_2>>E_2>>f>>g>>mass_2>>

ID_3>>state_3>>mother1_3>>mother2_3>>
color1_3>>color2_3>>px_3>>py_3>>pz_3>>E_3>>h>>i>>mass_3>>

ID_4>>state_4>>mother1_4>>mother2_4>>
color1_4>>color2_4>>px_4>>py_4>>pz_4>>E_4>>j>>k>>mass_4>>

l>>m>>n>>o>>p>>q>>r>>s>>t>>u>>
v>>w>>x>>y>>z>>a1>>a2>>a3>>a4>>a5>>a6;



//Calculation
TLorentzVector particle1;//Defining lorentz vector for muon 1
  particle1.SetPxPyPzE(px_1,py_1,pz_1,E_1);//Defining variables in the vector

  TLorentzVector particle2;
  particle2.SetPxPyPzE(px_2,py_2,pz_2,E_2);

TLorentzVector particle3;
  particle3.SetPxPyPzE(px_3,py_3,pz_3,E_3);

  TLorentzVector particle4;
  particle4.SetPxPyPzE(px_4,py_4,pz_4,E_4);

TLorentzVector DecayParticle=particle3+particle4;//Creating a vector for the
ParticleMass=DecayParticle.M();//Getting invariant mass
ofstream writer("invariant_masses.txt" , ios::app);//write in data
if( ! writer)//check for errors
{
cout<<"write error"<<endl;
```

```
return -1;
}
else
{
writer<<ParticleMass<<endl;

}
}

}

data.close();
writer.close();

}
```

10.7 pspline8.py

The code that read the data, determined an appropriate α, fitted the P-spline and processed the results.

The contents of the function "create_pspline" is credited to Bjarne Grimstad and his team who included parts of this function in their examples in the SPLINTER library.

The contents of the function "makePrettyPlots" is credited to Meghan Frate

10.7.1 Main

```
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
from os import sys, path, remove
sys.path.append(path.dirname(path.dirname(path.abspath(__file__))))
import splinter
from scipy.stats import chisquare
import library

def test():
    alpha=0.8187500000000001
    atlas_x,atlas_y=library.atlas()
    (x,y,yd3,atlas_pspline)=library.create_pspline(atlas_x,atlas_y,alpha=alpha)
    chi_atlas=library.chisquare(y,yd3)
    print("chi atlas: ",chi_atlas)
    print('-----------------------------------------------------------')
    plta=library.makePrettyPlots(x, y, yd3, title="ATLAS data fitted with "
    +r"$\alpha=$"+str(round(alpha,2))+"\n"+r'$\chi^2=$'+str(round(chi_atlas,2)))
    list_of_chi_final=[]
    iteration=18
    (x_madgraph,y_madgraph)=library.MadGraph(iteration)
    a=library.create_pspline(x_madgraph,y_madgraph,alpha)
    (x_madgraph,y_madgraph,yd3_madgraph,madgraph_pspline)=a
    chi_madgraph=library.chisquare(y_madgraph,yd3_madgraph)
    plta.show()

def run():
    alpha,list_of_chi=library.average_alpha()
    pltc=library.plot_chi_spectrum(list_of_chi,1)
    pltc.show()
    #pltc=library.plot_alpha_spectrum(list_of_chi,1)
    #pltc.show()
    list_of_alpha=[]
    list_of_alpha=library.alpha_spectrum(1,100,list_of_alpha)#inclusive
    atlas_alpha=library.find_alpha(library.atlas(),0.1,2,'')
    print('Atlas has a optimal alpha of: ',atlas_alpha)
    pltb=library.plot_alpha_spectrum(list_of_alpha,atlas_alpha)
    pltb.show()
    pltd=library.fit_atlas(atlas_alpha)
    pltd.show()

if __name__=="__main__":
    run()
    #test()
```

10.7.2 Library

```
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
from os import sys, path, remove
sys.path.append(path.dirname(path.dirname(path.abspath(__file__))))
import splinter
import library
from scipy.stats import chisquare
import pspline8

def fit_atlas(alpha):
    atlas_x,atlas_y=library.atlas()
    (x,y,yd3,atlas_pspline)=library.create_pspline(atlas_x,atlas_y,alpha=alpha)
    chi_atlas=library.chisquare(y,yd3)
    print("chi atlas: ",chi_atlas)
    print('-----------------------------------------------------------')
    return library.makePrettyPlots(x, y, yd3, title="ATLAS, alpha:"
                               +str(alpha)+", chi:"+str(chi_atlas))


def average_alpha():
    number_of_events=library.number_of_files_in_folder()
    #number_of_events=15
    print("number of runs:", number_of_events)
    list_of_chi=[]
    #alpha=500.625
    alpha=1#1.078125#20:0.875
    mean=0
    factor=1
    operator=''
    while True:
        list_of_chi=[]
        for iteration in range(1,number_of_events+1):
            (x_madgraph,y_madgraph)=library.MadGraph(iteration)
            a=library.create_pspline(x_madgraph,y_madgraph,alpha)
            (x_madgraph,y_madgraph,yd3_madgraph,madgraph_pspline)=a
            list_of_chi.append(library.chisquare(y_madgraph,yd3_madgraph))

        mean=sum(list_of_chi)/number_of_events
        print('Average Chi: ',mean)
        if mean<0.999:
            if operator=="SUBTRACTING":
                factor=factor/2
            operator="ADDING"
            alpha +=0.5*factor
        elif mean>1.001:
            if operator=="ADDING":
                factor=factor/2
            operator="SUBTRACTING"
            alpha =alpha-0.5*factor
        else:
            print('-----------------------------------------------------------')
            print("Best alpha is:",alpha,"\nIt gives a average chi of: ",mean)
            print('-----------------------------------------------------------')
            break
        print("Current alpha: ", alpha)

    atlas_x,atlas_y=library.atlas()
    (x,y,yd3,atlas_pspline)=library.create_pspline(atlas_x,atlas_y,alpha=alpha)
    chi_atlas=library.chisquare(y,yd3)

    print("chi atlas: ",chi_atlas)
    print('-----------------------------------------------------------')
    #plta=library.makePrettyPlots(x, y, yd3, title="ATLAS, alpha:"+str(alpha)
    #                             +", chi:"+str(chi_atlas), ymax = 2e5)
    return alpha,list_of_chi

def alpha_spectrum(start,stop,list_of_alpha):
    number_of_events=library.number_of_files_in_folder()
    print("number of runs:", number_of_events)
    alpha=0.5
    factor=2
    operator=''
    for iteration in range(start,stop+1):
        alpha=0.1
        factor=2
        list_of_alpha.append(find_alpha(library.MadGraph(iteration),
                                        alpha,factor,operator))
    return list_of_alpha

def find_alpha(tuuple,alpha,factor,operator):
    while True:
        x,y=tuuple
        (x,y,yd3,pspline)=library.create_pspline(x,y,alpha)
        this_chi=(library.chisquare(y,yd3))
        #print('Run: ',iteration,' Chi: ',this_chi,' alpha: ',alpha)
        if 0.999<this_chi<1.001:
            break
        elif this_chi<0.999:
            if operator=="SUBTRACTING":
                factor=factor/2
            operator="ADDING"
            alpha +=0.5*factor
        elif this_chi>1.001:
            if operator=="ADDING":
                factor=factor/2
            operator="SUBTRACTING"
            alpha =alpha-0.5*factor
            if alpha<0:
                while alpha<0:
                    alpha=alpha+1.5
                    factor=factor/2
    return alpha

def plot_alpha_spectrum(list_of_alpha,atlas_alpha):
    bin_list = np.linspace(0, 10, 51)
    plt.rc("figure", facecolor="white")
    plt.hist(list_of_alpha, bins=bin_list)
    # plt.hist passes it's arguments to np.histogram
    plt.title("Alpha distribution\nOptimal ATLAS " r'$\alpha$''='
             +str(round(atlas_alpha,2)))
    plt.axis([0, 10, 0, 15])
    plt.ylabel("Counts")
    plt.xlabel(r"$\alpha$")
    extraticks=[atlas_alpha]
    plt.xticks(list(plt.xticks()[0]) + extraticks)
    print(sum(list_of_alpha)/len(list_of_alpha))
    return plt

def plot_chi_spectrum(list_of_alpha,atlas_alpha):
    plt.rc("figure", facecolor="white")
    bin_list = np.linspace(0, 2, 21)
    plt.hist(list_of_alpha, bins=bin_list)
    # plt.hist passes it's arguments to np.histogram
    plt.title(r"$\chi^2$"+" Distribution")
    plt.ylabel("Counts")
    plt.xlabel(r"$\chi^2$")

    plt.axis([0, 2, 0, 30])
    extraticks=[atlas_alpha]
    plt.xticks(list(plt.xticks()[0]) + extraticks)
    return plt


def makePrettyPlots(xs, ys, bkgs, title,ymin=1e-1, ymax = 1e5,xmin=1100):
    #This function is credited to Meghan Frate
    f, (ax1, ax2) = plt.subplots(2, sharex=True,
                                 figsize=(12,12),
                                 gridspec_kw = {'height_ratios':[3, 1]})
    f.suptitle(title, fontsize=30)
    dataPlot = ax1.errorbar(xs, ys, marker='o', ls='None',
                            yerr = np.sqrt(ys), c='black', markersize=7, label="Data")
    bkg1Plot, = ax1.plot(xs, bkgs, color='g', linewidth=3.0, label="P-Spline")

    ax1.legend()
    ax1.set_ylabel('Number of Events', fontsize=20)
    ax1.set_xlabel('m (GeV)', fontsize=20)
    ax1.set_yscale('log', nonposy="clip")
    ax1.set_xscale('log')
    ax1.set_xlim([xmin, 7500])
    ax1.set_ylim([ymin, ymax])
    ax1.tick_params(axis='y', labelsize=20)

    diff=(ys-bkgs)/bkgs
    ax2.plot(xs, diff, marker='o', ls='None', c='black', markersize=7,)
    ax2.axhline(0, color='black', lw=1)
    ax2.set_ylabel(r'$\frac{y_{Data}-y_{spline}}{y_{spline}}$', fontsize=20)
    ax2.set_xlabel('Invariant Mass (GeV)', fontsize=20)
    ax2.set_xscale('log')
    ax2.tick_params(axis='y', labelsize=20)
    ax2.set_xlim([xmin, 7500])
    ax2.set_ylim([-1.4, 1.4])


    ax2.set_xticks([xmin, 2000 ,3000, 4000, 5000, 6000, 7000])
    minor_ticks = np.arange(xmin, 7500, 100)
    ax2.set_xticks(minor_ticks, minor=True)
    labels = [str(xmin),str(xmin+900),str(xmin+1900),
              str(xmin+2900),str(xmin+3900),str(xmin+4900),str(xmin+5900)]
    ax2.set_xticklabels(labels)

    f.subplots_adjust(hspace=0)
    plt.setp([a.get_xticklabels() for a in f.axes[:-1]], visible=False)

    #plt.show()
    return plt

def atlas_no():
    read_atlas_x=open("xval_ATLAS.txt","r")
    x=[]
    for line in read_atlas_x:
        x.append(float(line[:-1]))
    read_atlas_x.close()
    x=np.array(x)

    read_atlas_y=open("yval_ATLAS.txt","r")
    y=[]
    for line in read_atlas_y:
        y.append(float(line[:-1]))
    read_atlas_y.close()
    y=np.array(y)

    return x,y

def atlas():
    ATLAS_hist=open('ATLAS_hist.txt','r')
    data=[]
    for line in ATLAS_hist:
            data.append(float(line[:-1]))

    read_bins=open("original_bins.txt","r")
    bins=[]
    for line in read_bins:
        bins.append(float(line[:-1]))

    hist=np.histogram(data,bins=bins, range=None,
                      normed=False, weights=None, density=None)
    y=hist[0]
    read_bins.close()
    ATLAS_hist.close()
    return (hist[1][:-1],y)

def create_pspline(x,y,alpha):
# This function is heavily based on a file that is part of the SPLINTER library.
# Copyright (C) 2012 Bjarne Grimstad (bjarne.grimstad@gmail.com).
#
# For this function:
# This Source Code Form is subject to the terms of the Mozilla Public
# License, v. 2.0. If a copy of the MPL was not distributed with this
# file, You can obtain one at http://mozilla.org/MPL/2.0/.
    a=splinter.BSplineBuilder(x, y, smoothing=splinter.BSplineBuilder.Smoothing.PSPLINE,
                              alpha=alpha).build()
    pspline=a
    n = len(x)
    xd = [0.]*n
    yd3 = [0.]*n
    for i,xd in enumerate(x):
        yd3[i] = pspline.eval(xd)[0]

    yd3=np.array(yd3)
    return (x,y,yd3,pspline)

def MadGraph(iteration):#with bin width division

    p=Path("/media/alexander/INTENSO/gp_events/run_"
           +str(iteration)+"/invariant_masses_final.txt")
    invariant_masses=p.open("r")

    data=[]
    for line in invariant_masses:
            data.append(float(line[:-1]))
    read_bins=open("original_bins.txt","r")

    bins=[]
    for line in read_bins:
        bins.append(float(line[:-1]))

    hist=np.histogram(data,bins=bins, range=None,
                      normed=False, weights=None, density=None)

    y=hist[0]
    read_bins.close()
    invariant_masses.close()
    return (hist[1][:-1],y)

def chisquare(y,yd3):
    chisquare=(sum(((y-yd3)**2)/yd3))/(len(y)-1)
    return chisquare

def number_of_files_in_folder():
    p=Path("/media/alexander/INTENSO/gp_events")
    return len(list(p.iterdir()))

def plot_all_mg(number_of_events,alpha):
    number_of_events=library.number_of_files_in_folder()
    for iteration in range(1,number_of_events+1):
            (x_madgraph,y_madgraph)=library.MadGraph(iteration)
            a=library.create_pspline(x_madgraph,y_madgraph,alpha)
            (x_madgraph,y_madgraph,yd3_madgraph,madgraph_pspline)=a
            pltb=library.makePrettyPlots(x_madgraph, y_madgraph, yd3_madgraph,
                            title="MadGraph run"+str(iteration)
                            +", alpha:"+str(alpha)+", chi:"
                            +str(library.chisquare(y_madgraph,yd3_madgraph)),
                            ymax = 2e5)
            pltb.show()

def plot_one_mg(run_number,alpha):
    (x_madgraph,y_madgraph)=library.MadGraph(run_number)
    a=library.create_pspline(x_madgraph,y_madgraph,alpha)
    (x_madgraph,y_madgraph,yd3_madgraph,madgraph_pspline)=a
    pltb=library.makePrettyPlots(x_madgraph, y_madgraph, yd3_madgraph,
                    title="MadGraph run"+str(run_number)+", alpha:"+str(alpha)
                    +", chi:"+str(library.chisquare(y_madgraph,yd3_madgraph)),
                    ymax = 2e5)
    return pltb
```

10.8 Run Card

This is the run card used to create the grid pack file used to generate the MG events.

```
#**********************************************************************
#                       MadGraph5_aMC@NLO                             *
#                                                                     *
#                     run_card.dat MadEvent                           *
#                                                                     *
#  This file is used to set the parameters of the run.                *
#                                                                     *
#  Some notation/conventions:                                         *
#                                                                     *
#   Lines starting with a '# ' are info or comments                   *
#                                                                     *
#   mind the format:   value    = variable     ! comment              *
#**********************************************************************
#
#******************
```

```
# Running parameters
#******************
#
#*********************************************************************
# Tag name for the run (one word)                                    *
#*********************************************************************
  tag_1 = run_tag ! name of the run
#*********************************************************************
# Number of events and rnd seed                                      *
# Warning: Do not generate more than 1M events in a single run       *
# If you want to run Pythia, avoid more than 50k events in a run.    *
#*********************************************************************
  10000 = nevents ! Number of unweighted events requested
  0 = iseed ! rnd seed (0=assigned automatically=default))
#*********************************************************************
# Collider type and energy                                           *
# lpp: 0=No PDF, 1=proton, -1=antiproton, 2=photon from proton,      *
#                                         3=photon from electron     *
#*********************************************************************
  1 = lpp1 ! beam 1 type
  1 = lpp2 ! beam 2 type
  6500.0 = ebeam1 ! beam 1 total energy in GeV
  6500.0 = ebeam2 ! beam 2 total energy in GeV
#*********************************************************************
# Beam polarization from -100 (left-handed) to 100 (right-handed)    *
#*********************************************************************
  0.0 = polbeam1 ! beam polarization for beam 1
  0.0 = polbeam2 ! beam polarization for beam 2
#*********************************************************************
# PDF CHOICE: this automatically fixes also alpha_s and its evol.    *
#*********************************************************************
  nn23lo1 = pdlabel ! PDF set
  230000 = lhaid ! if pdlabel=lhapdf, this is the lhapdf number
#*********************************************************************
# Renormalization and factorization scales                           *
#*********************************************************************
  False = fixed_ren_scale ! if .true. use fixed ren scale
  False = fixed_fac_scale ! if .true. use fixed fac scale
  91.188 = scale ! fixed ren scale
  91.188 = dsqrt_q2fact1 ! fixed fact scale for pdf1
  91.188 = dsqrt_q2fact2 ! fixed fact scale for pdf2
  -1 = dynamical_scale_choice ! Choose one of the preselected dynamical choices
  1.0 = scalefact ! scale factor for event-by-event scales
```

```
#*********************************************************************
# Type and output format
#*********************************************************************
  True = gridpack !True = setting up the grid pack
  -1.0 = time_of_flight ! threshold (in mm) below which the invariant livetime is not wr
  3.0 = lhe_version ! Change the way clustering information pass to shower.
  True = clusinfo ! include clustering tag in output
  sum = event_norm ! average/sum. Normalization of the weight in the LHEF

#*********************************************************************
# Matching parameter (MLM only)
#*********************************************************************
  0 = ickkw ! 0 no matching, 1 MLM
  1.0 = alpsfact ! scale factor for QCD emission vx
  False = chcluster ! cluster only according to channel diag
  4 = asrwgtflavor ! highest quark flavor for a_s reweight
  False = auto_ptj_mjj ! Automatic setting of ptj and mjj if xqcut >0
                                 ! (turn off for VBF and single top processes)
  0.0 = xqcut ! minimum kt jet measure between partons
#*********************************************************************
#
#*********************************************************************
# handling of the helicities:
#  0: sum over all helicities
#  1: importance sampling over helicities
#*********************************************************************
  0 = nhel ! using helicities importance sampling or not.
#*********************************************************************
# Generation bias, check the wiki page below for more information:   *
#  'cp3.irmp.ucl.ac.be/projects/madgraph/wiki/LOEventGenerationBias' *
#*********************************************************************
  None = bias_module ! Bias type of bias, [None, ptj_bias, -custom_folder-]
  {} = bias_parameters ! Specifies the parameters of the module.
#
#*******************************
# Parton level cuts definition *
#*******************************
#
#
#*********************************************************************
# BW cutoff (M+/-bwcutoff*Gamma) ! Define on/off-shell for "$" and decay
#*********************************************************************
  15.0 = bwcutoff ! (M+/-bwcutoff*Gamma)
```

```
#*********************************************************************
# Apply pt/E/eta/dr/mij/kt_durham cuts on decay products or not
# (note that etmiss/ptll/ptheavy/ht/sorted cuts always apply)
#*********************************************************************
  False = cut_decays ! Cut decay products
#*********************************************************************
# Standard Cuts                                                      *
#*********************************************************************
# Minimum and maximum pt's (for max, -1 means no cut)                *
#*********************************************************************
  20.0 = ptj ! minimum pt for the jets
  0.0 = ptb ! minimum pt for the b
  10.0 = pta ! minimum pt for the photons
  10.0 = ptl ! minimum pt for the charged leptons
  0.0 = misset ! minimum missing Et (sum of neutrino's momenta)
  0.0 = ptheavy ! minimum pt for one heavy final state
  -1.0 = ptjmax ! maximum pt for the jets
  -1.0 = ptbmax ! maximum pt for the b
  -1.0 = ptamax ! maximum pt for the photons
  -1.0 = ptlmax ! maximum pt for the charged leptons
  -1.0 = missetmax ! maximum missing Et (sum of neutrino's momenta)
#*********************************************************************
# Minimum and maximum E's (in the center of mass frame)              *
#*********************************************************************
  0.0 = ej ! minimum E for the jets
  0.0 = eb ! minimum E for the b
  0.0 = ea ! minimum E for the photons
  0.0 = el ! minimum E for the charged leptons
  -1.0 = ejmax ! maximum E for the jets
  -1.0 = ebmax ! maximum E for the b
  -1.0 = eamax ! maximum E for the photons
  -1.0 = elmax ! maximum E for the charged leptons
#*********************************************************************
# Maximum and minimum absolute rapidity (for max, -1 means no cut)   *
#*********************************************************************
  5.0 = etaj ! max rap for the jets
  -1.0 = etab ! max rap for the b
  2.5 = etaa ! max rap for the photons
  2.5 = etal ! max rap for the charged leptons
  0.0 = etajmin ! min rap for the jets
  0.0 = etabmin ! min rap for the b
  0.0 = etaamin ! min rap for the photons
  0.0 = etalmin ! main rap for the charged leptons
```

```
#*********************************************************************
# Minimum and maximum DeltaR distance                                *
#*********************************************************************
  0.4 = drjj ! min distance between jets
  0.0 = drbb ! min distance between b's
  0.4 = drll ! min distance between leptons
  0.4 = draa ! min distance between gammas
  0.0 = drbj ! min distance between b and jet
  0.4 = draj ! min distance between gamma and jet
  0.4 = drjl ! min distance between jet and lepton
  0.0 = drab ! min distance between gamma and b
  0.0 = drbl ! min distance between b and lepton
  0.4 = dral ! min distance between gamma and lepton
  -1.0 = drjjmax ! max distance between jets
  -1.0 = drbbmax ! max distance between b's
  -1.0 = drllmax ! max distance between leptons
  -1.0 = draamax ! max distance between gammas
  -1.0 = drbjmax ! max distance between b and jet
  -1.0 = drajmax ! max distance between gamma and jet
  -1.0 = drjlmax ! max distance between jet and lepton
  -1.0 = drabmax ! max distance between gamma and b
  -1.0 = drblmax ! max distance between b and lepton
  -1.0 = dralmax ! maxdistance between gamma and lepton
#*********************************************************************
# Minimum and maximum invariant mass for pairs                       *
# WARNING: for four lepton final state mmll cut require to have      *
#          different lepton masses for each flavor!                  *
#*********************************************************************
   1100.0 = mmjj ! min invariant mass of a jet pair
   0.0 = mmbb ! min invariant mass of a b pair
   0.0 = mmaa ! min invariant mass of gamma gamma pair
   0.0 = mmll ! min invariant mass of l+l- (same flavour) lepton pair
  -1.0 = mmjjmax ! max invariant mass of a jet pair
  -1.0 = mmbbmax ! max invariant mass of a b pair
  -1.0 = mmaamax ! max invariant mass of gamma gamma pair
  -1.0 = mmllmax ! max invariant mass of l+l- (same flavour) lepton pair
#*********************************************************************
# Minimum and maximum invariant mass for all letpons                 *
#*********************************************************************
   0.0 = mmnl ! min invariant mass for all letpons (l+- and vl)
  -1.0 = mmnlmax ! max invariant mass for all letpons (l+- and vl)
#*********************************************************************
# Minimum and maximum pt for 4-momenta sum of leptons                *
```

```
#*********************************************************************
  0.0 = ptllmin ! Minimum pt for 4-momenta sum of leptons(l and vl)
  -1.0 = ptllmax ! Maximum pt for 4-momenta sum of leptons(l and vl)
#*********************************************************************
# Inclusive cuts                                                     *
#*********************************************************************
  0.0 = xptj ! minimum pt for at least one jet
  0.0 = xptb ! minimum pt for at least one b
  0.0 = xpta ! minimum pt for at least one photon
  0.0 = xptl ! minimum pt for at least one charged lepton
#*********************************************************************
# Control the pt's of the jets sorted by pt                          *
#*********************************************************************
  0.0 = ptj1min ! minimum pt for the leading jet in pt
  0.0 = ptj2min ! minimum pt for the second jet in pt
  0.0 = ptj3min ! minimum pt for the third jet in pt
  0.0 = ptj4min ! minimum pt for the fourth jet in pt
  -1.0 = ptj1max ! maximum pt for the leading jet in pt
  -1.0 = ptj2max ! maximum pt for the second jet in pt
  -1.0 = ptj3max ! maximum pt for the third jet in pt
  -1.0 = ptj4max ! maximum pt for the fourth jet in pt
  0 = cutuse ! reject event if fails any (0) / all (1) jet pt cuts
#*********************************************************************
# Control the pt's of leptons sorted by pt                           *
#*********************************************************************
  0.0 = ptl1min ! minimum pt for the leading lepton in pt
  0.0 = ptl2min ! minimum pt for the second lepton in pt
  0.0 = ptl3min ! minimum pt for the third lepton in pt
  0.0 = ptl4min ! minimum pt for the fourth lepton in pt
  -1.0 = ptl1max ! maximum pt for the leading lepton in pt
  -1.0 = ptl2max ! maximum pt for the second lepton in pt
  -1.0 = ptl3max ! maximum pt for the third lepton in pt
  -1.0 = ptl4max ! maximum pt for the fourth lepton in pt
#*********************************************************************
# Control the Ht(k)=Sum of k leading jets                            *
#*********************************************************************
  0.0 = htjmin ! minimum jet HT=Sum(jet pt)
  -1.0 = htjmax ! maximum jet HT=Sum(jet pt)
  0.0 = ihtmin !inclusive Ht for all partons (including b)
  -1.0 = ihtmax !inclusive Ht for all partons (including b)
  0.0 = ht2min ! minimum Ht for the two leading jets
  0.0 = ht3min ! minimum Ht for the three leading jets
  0.0 = ht4min ! minimum Ht for the four leading jets
  -1.0 = ht2max ! maximum Ht for the two leading jets
  -1.0 = ht3max ! maximum Ht for the three leading jets
  -1.0 = ht4max ! maximum Ht for the four leading jets
#*********************************************************************
# Photon-isolation cuts, according to hep-ph/9801442                 *
# When ptgmin=0, all the other parameters are ignored                *
# When ptgmin>0, pta and draj are not going to be used               *
#*********************************************************************
  0.0 = ptgmin ! Min photon transverse momentum
  0.4 = r0gamma ! Radius of isolation code
  1.0 = xn ! n parameter of eq.(3.4) in hep-ph/9801442
  1.0 = epsgamma ! epsilon_gamma parameter of eq.(3.4) in hep-ph/9801442
  True = isoem ! isolate photons from EM energy (photons and leptons)
#*******************************************************************
# WBF cuts                                                         *
#*******************************************************************
  0.0 = xetamin ! minimum rapidity for two jets in the WBF case
  0.0 = deltaeta ! minimum rapidity for two jets in the WBF case
#*********************************************************************
# Turn on either the ktdurham or ptlund cut to activate              *
# CKKW(L) merging with Pythia8 [arXiv:1410.3012, arXiv:1109.4829]    *
#*********************************************************************
  -1.0 = ktdurham
  0.4 = dparameter
  -1.0 = ptlund
  1, 2, 3, 4, 5, 6, 21 = pdgs_for_merging_cut ! PDGs for two cuts above
#*******************************************************************
# maximal pdg code for quark to be considered as a light jet       *
# (otherwise b cuts are applied)                                   *
#*******************************************************************
  4 = maxjetflavor ! Maximum jet pdg code
#*******************************************************************
#
#*******************************************************************
# Store info for systematics studies                               *
# WARNING: Do not use for interference type of computation         *
#*******************************************************************
  True = use_syst ! Enable systematics studies
#
#**************************************
# Parameter of the systematics study
#  will be used by SysCalc (if installed)
#**************************************
```

```
#
  0.5 1 2 = sys_scalefact # factorization/renormalization scale factor
  None = sys_alpsfact # \alpha_s emission scale factors
  auto = sys_matchscale # variation of merging scale
# PDF sets and number of members (0 or none for all members).
  NNPDF23_lo_as_0130_qed = sys_pdf # separate by && if more than one set.
# MSTW2008nlo68cl.LHgrid 1  = sys_pdf
#
```