Karl-Franzens University Graz

Bachelor Thesis (BSc)

The form of the Higgs-PDF in the proton

Franziska Reiner, 01530051

supervised by

Univ.-Prof. Dipl.-Phys. Dr.rer.nat. Axel Maas

September 16, 2019

Abstract

The objective of this bachelor thesis is the study of possible Higgs-PDFs as well as the Higgs contingent in the proton, which is postulated by a quantized, non-abelian gauge theory with an active Brout-Englert-Higgs effect. This theory leads to a formulation with bound states, when treated non-perturbatively. For this reason, the general event generator HERWIG++ will be used to simulate the proton-proton-collisions at the LHC, focussing on the two final states most sensitive to the Higgs ($t\bar{t}$ and $t\bar{t}Z$). The results are then plotted and analysed in order to find the highest possible amount of Higgs in the proton and the corresponding form of the Higgs-PDF.

Contents

1. Introduction

Identifying symmetries is the aim of particle physics. Symmetries are expressed by an invariance of the action integral and Noether's theorem states that every symmetry in the action corresponds to a conserved quantity of the theory. We can distinguish between space-time symmetries and internal symmetries. Latter ones are the basis of gauge theory. The principle of gauge theory reaches back to Einstein's dream of a unification of all fundamental forces in nature. Based on his special and general theory of relativity, Hermann Weyl proposed in 1919 his idea of gauge invariance. Although his initial theory failed, the idea of a local gauge symmetry survived, since Maxwell's equations are invariant under such gauge transformations and those same turn out to be a very useful mathematical construct to simplify otherwise difficult to solve physical problems. The final breakthrough of gauge theories occurred when Geradus 't Hooft proved in the 1970s that Yang-Mills theories (non-abelian gauge theories) are renormalizable even if the symmetry is spontaneously broken. Nowadays modern particle physics is inconceivable without this mathematical stroke of genius and the all too well known standard model (SM) of particle physics is also a gauge theory with the gauge group $SU(3)\times SU(2)\times U(1)$.

In this thesis, we will be concentrating on a quantized, non-abelian gauge theory with an active Brout-Englert-Higgs effect, which requires, using non-perturbative methods, a formulation with composite states. The aim is to simulate the proton-proton-collisions at the LHC (CERN) to formulate predictions that could substantiate the thesis statement. For this reason, we will be looking at possible forms of a Higgs-PDF-function for a Higgs in the proton.

In the first "theoretical" part we will be covering the underlying concept as well as the importance of the Higgs concerning this matter. Accordingly, for this purpose, there will be given a more general insight in event generators and the concept of PDFs.

In the subsequent "technical" part the details of the simulation with HERWIG++ will be discussed and the required Feynman diagrams illustrated.

The results of our studies can be seen in chapter 4, followed by a final summary including an outlook for future theses.

2. Theoretical Part

2.1. Theory

This first part should give a brief insight in the theory and motivation behind the attempt of finding a suitable Higgs-PDF for proton-proton-collisions. For further information, I would like to refer to

[1] A. Maas, (2017), 1712.04721

[2] Master thesis; Simon Fernbach; "Higgs-PDF study in proton-proton collisions"

A common mathematical formalism to insert redundant degrees of freedom in the Lagrangian of a field theory is the formulation as a gauge theory. Since the fields themselves are not directly measurable, but some of the quantities are, different configurations of a field can lead to identical observable quantities. The transformation between such fields are called gauge transformations, and the unchanged quantities are then gauge-invariant. The physics, however, may not depend on such transformations and therefore has to be gauge-independent.

Formulating the theory to be gauge-independent is, relatively spoken, easy for abelian theories, because in this case the field strength tensor is gauge-invariant (e.g.: QED). On the other hand, non-abelian theories have a gauge-variant field strength tensor and hence the theory cannot be formulated in terms of equivalents of electric/magnetic fields and charge since, as a consequence, they are not physically observable. The standard model is such a non-abelian gauge theory with the symmetry group $SU(3) \times SU(2) \times U(1)$ and a total of 12 gauge bosons (one photon, 3 weak bosons and 8 gluons).

For quantum theories the gauge theory has to be quantized. As a result, some of the symmetries of the classical theory cannot be maintained and we obtain anomalies:

- *Scale anomaly*: breaks conformal symmetry;
- *Chiral anomaly*: violates the conservation of the axial current; e.g.: Adler-Bell-Jackiw anomalies; as a result, the neutral pion decays into two photons
- *Gauge anomaly*: have to be extinguished for a valid gauge theory

There are several methods for quantization, inter alia, canonical ones and ones based on the path integral formalism. To actually calculate any quantities of the theory one

can distinguish between perturbative schemes (cf. [3] for further information) and non-perturbative schemes (e.g.: lattice gauge theory) to do so. In our case, a path integral formulation was used. As a consequence, the elementary fields and correlation functions are gauge-variant. In perturbation theory this is fixed by the BRST symmetry, but for a non-perturbation theory, due to the Gribov-Singer ambiguity, we have to introduce composite fields/ bound states to regain gauge-independence.

A good candidate to form such composite states is the Higgs, as it is an uncharged, weak isospin-doublet particle with Spin zero. To calculate the masses of the bound state, one can apply the "gauge-invariant perturbation theory", a method developed by Fröhlich, Morchio and Strocchi (FMS). The precondition for its use is an active Brout-Englert-Higgs (BEH) effect. The basic rules of the concept can be described in the following steps [1]:

1. Formulating gauge-invariant operators and forming the correlation functions (e.g.: propagator)

$$O(x) = (\phi^\dagger \phi)(x) \text{ ..... operator with } \phi(x) \text{ the Higgs field} \tag{2.1}$$

$$\langle O^\dagger O \rangle \text{ ..... propagator} \tag{2.2}$$

2. Gauge fixing with a non-vanishing vacuum expectation value: in this case 't Hooft gauge

$$\phi(x) = v + \eta(x) \text{ ..... 't Hooft gauge with fluctuation field } \eta \tag{2.3}$$

3. Expanding the Higgs field (if contained) around fluctuations

$$\langle O^\dagger(x) O(y) \rangle = \tag{2.4}$$

$$= v^2 \langle \eta(x)\eta(y) \rangle + \tag{2.5}$$

$$+ v\langle \eta^\dagger(x)\eta^\dagger(y)\eta(y) + \eta^\dagger(y)\eta^\dagger(x)\eta(x) \rangle + \langle \eta^\dagger(x)\eta(x)\eta^\dagger(y)\eta(y) \rangle \tag{2.6}$$

4. Apply perturbation theory on the right-hand side of equation (2.4)

Non-perturbatively only the sum of (2.5) and (2.6) is physical, because of BRST symmetry breaking. Standard perturbation theory, comparatively, would mean to keep only (2.5) and expand this in a perturbative series.

Applying the gauge-invariant perturbation theory on scattering processes, this implies that we can identify the scattering states with bound states and hence they can be seen as resonances in cross-sections. While in LO (leading order) these resonances would match the elementary particle resonances, for higher orders of perturbation theory we can expect to see the impact of the bound states.

Easier than to actually calculate the right-hand side of equation (2.4) by using lattice theory, is the use of PDFs (parton density functions), which are sufficient for our purpose, since we are only interested in the question, if a Higgs exists in the proton and what the impact of its contribution would be. Because of the role it plays in the collisions at the LHC, we will take a closer look at the proton, which, as a baryon, cannot be described gauge-independently in the standard model only using quarks. The simplest way to gain gauge-independence is by adding a Higgs. For this reason, we will study proton-proton-collisions, which will be simulated using the event generator HERWIG++. In the next section there will be given a more general insight in event generators, followed by an introduction to PDFs.

2.2. Event Generator

Event Generators are specific programs that create high- energy events in a simulation. They represent an intermediate step between theoretical theses, first-principle calculations and experiments. The most popular general-purpose generators at the moment are HERWIG(++) and PYTHIA. In course of this bachelor's thesis, there will be made use of HERWIG++, which, in contrast to its predecessor version HERWIG, is written in C++ and has several improvements regarding the physics as well as the structure of the simulation. Most notable in this context is the extension of the hadron-hadron collisions.

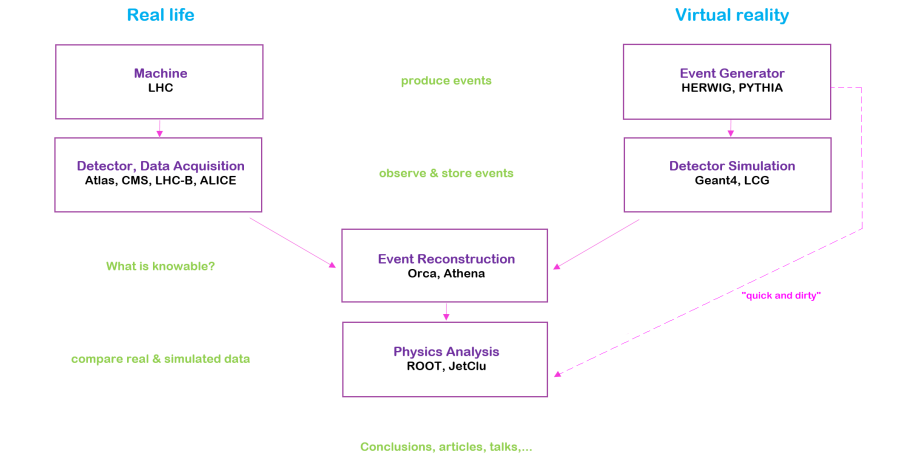

Figure 2.1.: comparison between real life and virtual reality (according to Sjöstrand, Torbjörn [4])

In Figure 2.1 the way from producing events to analyze them is shown as a comparison of reality and virtual reality. The Event Generator takes the place of the machine in producing the events. Afterwards, the produced data is handed over to a detector simulation equivalent to the detector in real life, which observes and stores the events. The stored data is then passed on to the event reconstruction software and later to programs for physics analysis, where the real and simulated data is compared. It is able to choose a direct way from the virtual event production to the analysis software, which is quicker, but also a rather dirty way. In the picture above there are some familiar examples for each step cited in black fonts.

If we take a closer look at the event production itself, we have to deal with the details of the structure of an event first. Imagine in our case two protons moving towards each other. Since protons belong to the group of hadrons (baryons, to be precise), they consist of several components: quarks, gluons and, based on the thesis considerations, a Higgs.

The distribution of these constituents of the two incoming beams is described by the parton density/distribution function.

Figure 2.2.: incoming beams

The constituents of the incoming beams react with each other by a hard subprocess, which is described by matrix elements (ME). As a result of the collision of the partons, new particles are generated depending on the type of incoming particle.

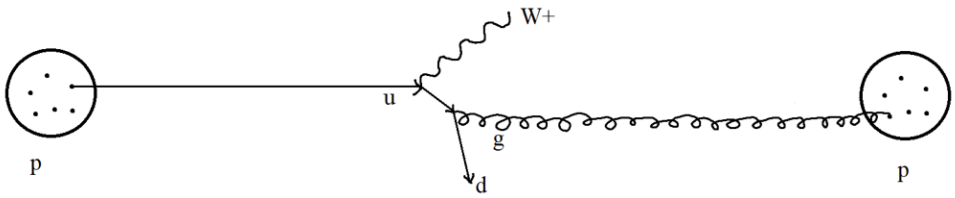

Figure 2.3.: hard subprocess between an up-quark and a gluon

Since many of these particles are unstable, they decay soon after. As a consequence, these so-called resonance decays, that are correlated with the hard subprocess, have to be considered in the calculations. This can be done perturbatively. (cf. Figure 2.4)

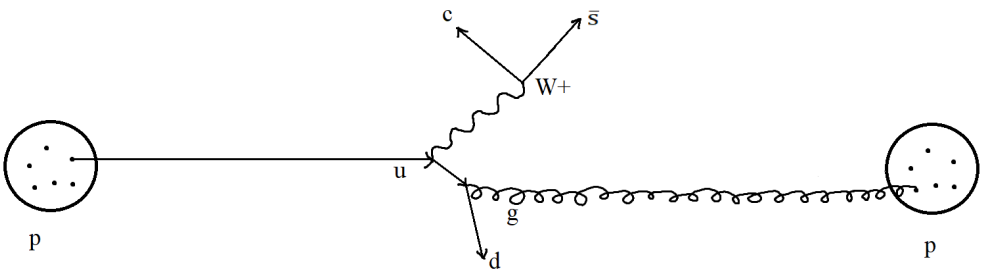

Figure 2.4.: resonance decay: The W-boson decays in a quark-antiquark-pair

This simple case is complicated by the fact, that the incoming particles carry colour charge, which results in the possibility of radiation of gluons (, just as charge-carrying particles can radiate photons). This so-called initial-state radiation produces space-like parton showers.

As well as the incoming particles, also the outgoing particles can radiate. We call this final-state radiation, which ends up as time-like parton showers.

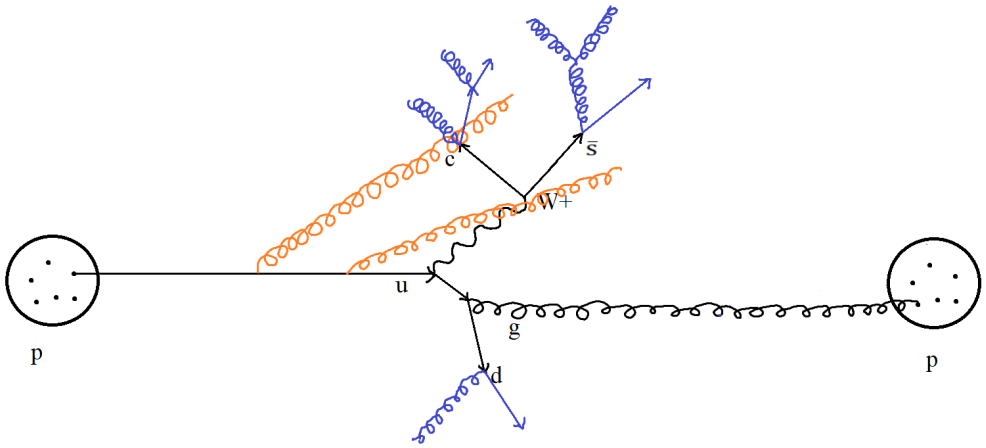

Figure 2.5.: initial- (orange) and final-state (blue) radiation

The outgoing particles (including the remnant beams) are connected by colour-confinement strings. After a limited distance, they begin to form primary hadrons with their colour-partner (hadronization). Most of these hadrons are unstable and decay further.

Because protons do contain several partons and a lot more are produced through initial- and final-state radiation, it is possible that it comes to multiple parton-parton interactions.

Quantum mechanics tells us that every possible event is going to happen, with a higher or lower probability. To generate the events as close to nature as possible, event generators make use of the Monte Carlo (MC) method. The final cross-section is then calculated from the cross-section of the hard subprocess multiplied by the total probability for the event, whereby the latter one is the product of the single probability for each of the steps described above.

$$\sigma_{\text{final state}} = \sigma_{\text{hard subprocess}} \cdot P_{\text{total; hard subprocess} \rightarrow \text{final state}} \tag{2.7}$$

where,

$$P_{\text{total}} = \prod_i P_i = P_{\text{resonance decay}} \cdot P_{\text{initial-sate radiation}} \cdot P_{\text{final-sate radiation}} \cdot$$

$$P_{\text{multiple interactions}} \cdot P_{\text{remnants}} \cdot P_{\text{hadronization}} \cdot P_{\text{decays}} \tag{2.8}$$

Sometimes it is better to choose a specialised generator, like in our case MadGraph, which produces the hard subprocess, and then use a Les Houches Interface (based on the Les Houches Accords) to include the output in the general-purpose generator (here: HERWIG++), that makes the further calculations.

In the next section, the importance of PDFs and the assumptions that have been made for the Higgs-PDF will be described.

2.3. Parton density function

PDF is short for **P**arton **D**ensity/ **D**istribution **F**unction. The name parton was introduced by Richard Feynman in the 1960s to describe the inner structure of hadrons, which turned out to be quarks and gluons. Nowadays the word parton is generally used for any constituents that make up a hadron, including virtual states, such as quark-antiquark-pairs.

Analogous to the case of elastic scattering, the cross-section for an inelastic scattering depends upon two so-called structure functions F_1 and F_2, describing the magnetic and

electrical interaction. F_2 is the Fourier transform of the charge distribution and for Spin-$\frac{1}{2}$ particles is connected to F_1 via the Callan-Gross relation:

$$F_2(x) = 2xF_1(x) \tag{2.9}$$

Here x is the Lorentz-invariant Bjorken x scaling variable, defined by:

$$x = \frac{Q^2}{2Mv} \tag{2.10}$$

describing the momentum fraction of the parton. $Q^2(= |-q^2|)$ is the squared four-momentum-transfer, M the mass and v the energy loss between the scattering particles. For the inelastic scattering, the invariant Mass W is greater than M, Q^2 is less than $2Mv$ and hence $0 < x < 1$. In the special case of elastic scattering $W = M$, $Q^2 = 2Mv$ and therefor $x = 1$.

In QCD (quantum chromodynamics) scale-independence is not necessarily given anymore and hence the structure functions will be written as $F_i(x, Q)$, with $Q = \sqrt{Q^2}$ the scaling variable. In leading-order (LO) approximation we obtain:

$$F_i(x, Q) = \sum_a c_i^a f_a(x, Q) \tag{2.11}$$

Here c_i^a is the coupling constant of a parton a and f_a is the parton density function of this parton, which describes the probability of finding a parton a at momentum fraction x and probing scale Q.

The sum over all parton momentums has to equal the total hadronic momentum and therefore we can formulate a completeness relation, that is independent of the viewed energy scale Q:

$$\sum_a \int_0^1 x f_a(x, Q) dx = 1 \tag{2.12}$$

The total cross-section is then calculated by convoluting the patronic cross-sections with the PDFs:

$$\sigma_{\text{total}} = \sum_{a,b} \int dx_1 dx_2 f_a^A(x, Q) f_b^B(x, Q) \otimes \sigma_{\text{a,b}} \tag{2.13}$$

While the evolution of the PDFs is perturbative and can be obtained by solving the DGLAP equations, the initial conditions of the PDFs are non-perturbative and not yet

calculable. Though, recently some progress has been made in the field of lattice QCD using the large-momentum effective field theory (LaMET) framework.

In general, the procedure of finding the PDF starts with a parametrisation of the same at a low scale, whose evolution is then calculated with the DGLAP scheme and fitted to the experimental data. For the parametrisation one can make a good guess or use the neural network PDF (NNPDF) methodology. Latter one, at first sight, seems to avoid an arbitrarily chosen function as a starting point but does have some ambiguity in the procedure itself.

In this Bachelor thesis we assume that the Higgs-PDF resembles the form of a gluon-PDF:

$$f_{\text{Higgs}} = x^a \cdot (1-x)^b \tag{2.14}$$

, varying the variables a from -2 to 2 and b from 0 to 2 with a step width of 0.1.

In comparison, a typical gluon-PDF looks as follows:

$$f_{\text{gluon}} = (2,62 + 9,17x)(1-x)^{5,9} \tag{2.15}$$

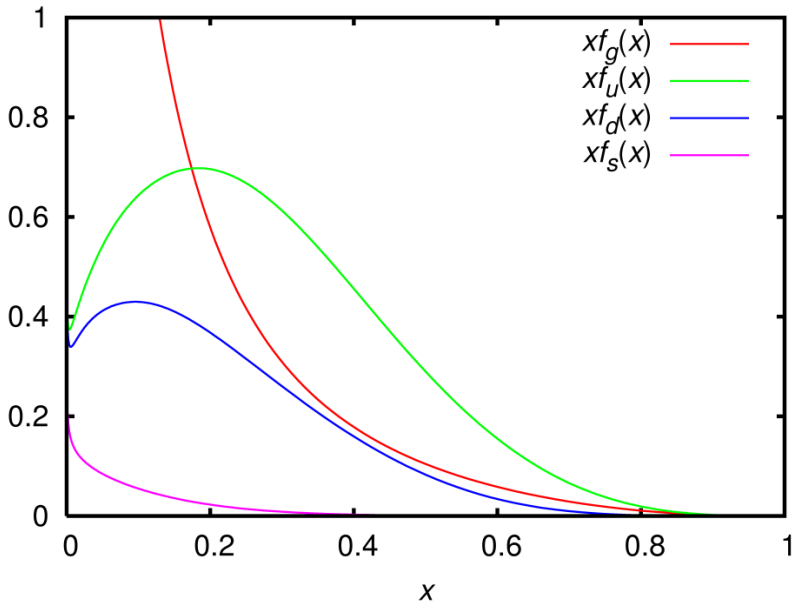

Figure 2.6.: PDFs of gluon (xf_g), up-quark (xf_u), down-quark (xf_d) and strange-quark (xf_s) [1]

[1] https://en.wikipedia.org/wiki/Parton_(particle_physics)#/media/File:CTEQ6_parton_distribution_functions.png (02.08.2019; 11:35h)

3. Technical Part

In this technical part we will be discussing the details of the usage of HERWIG++ for our purpose. As already mentioned, we are interested in proton-proton-collisions, simulating the actual events at the LHC. After HERWIG++ is running as a virtual environment on the computer, we first have to build the Rivet (Robust Independent Validation of Experiment and Theory) Analysis Plugin and then read in the input files (HH.in, GH.in, sm.in) for the different subprocesses. This produces the output files (HH.run, GH.run, sm.run) that contain all the details of the event generator we have just set up before. By running these files (command: Herwig run _.run -N number), we can finally generate the events and with the command line part -N number, the number of events can be varied.

For time-related reasons, we ran the program with only 1000 events. Even though this produces useful data, we have to take into account the high uncertainty that comes with our output when analyzing the results.

Since the Higgs strongly interacts with the top-quark [5], we will be focusing on the following process: $pp \to t\bar{t}$.

Adding a Z-boson to the final state makes the process more sensitive to the Higgs, due to a number of extra Feynman diagrams with HHH and ZH coupling. Therefore, we will also take a look at: $pp \to t\bar{t}Z$.

In the Appendix (A) one can find the code of the PYTHON script and bash file, that manage the control of the event generator as well as the calculation and plot of the cross-sections over the percentage of Higgs in the proton. In the following table, the used setting parameters are listed for a better overview.

Table 3.1.: setting parameters

process	$pp \to t\bar{t}$	$pp \to t\bar{t}Z$
Number of events	1000	
Form of the Higgs-PDF	$f_{\text{Higgs}} = x^a \cdot (1-x)^b$	
	a	[-2;2]
	b	[0;2]
Experimental error	3%	13%

In Figure 3.1 to 3.12 you can see the Feynman diagrams (LO) produced by the specialized generator MadGraph for the $t\bar{t}$ and the $t\bar{t}Z$ process, respectively.

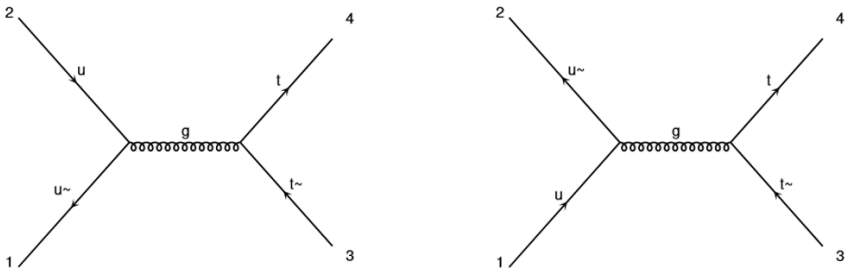

Figure 3.1.: Feynman diagrams for $u\bar{u} \to t\bar{t}$

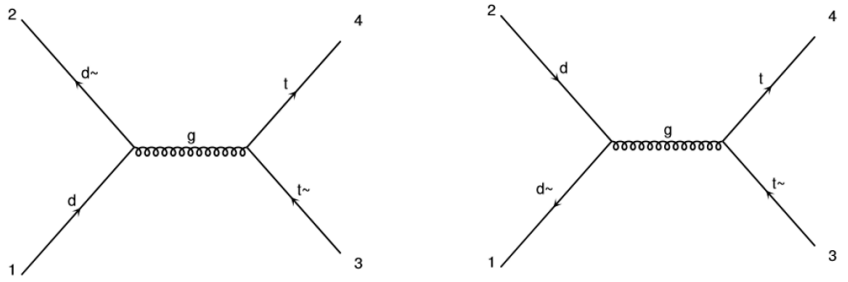

Figure 3.2.: Feynman diagrams for $d\bar{d} \to t\bar{t}$

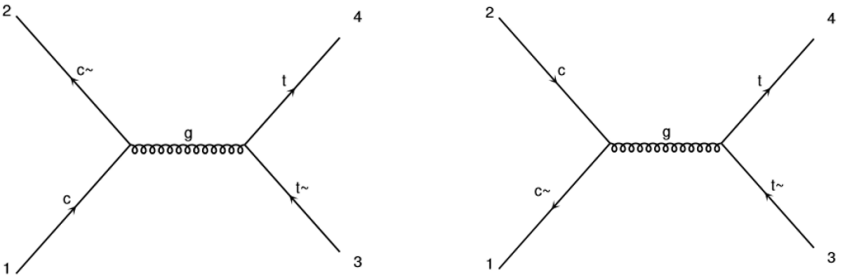

Figure 3.3.: Feynman diagrams for $c\bar{c} \to t\bar{t}$

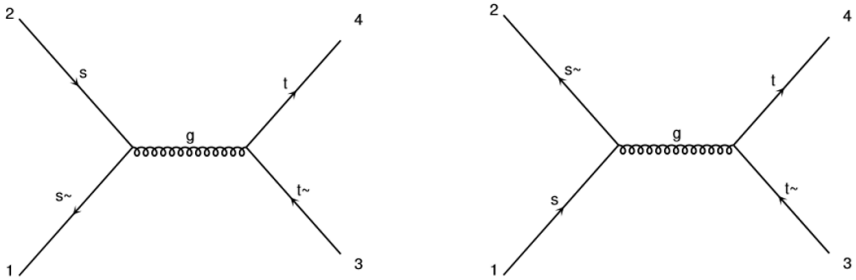

Figure 3.4.: Feynman diagrams for $s\bar{s} \to t\bar{t}$

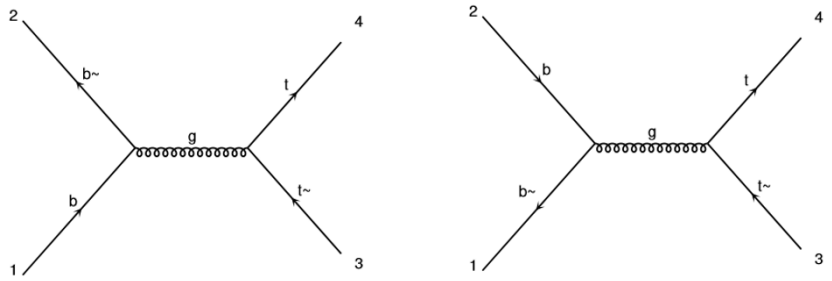

Figure 3.5.: Feynman diagrams for $b\bar{b} \to t\bar{t}$

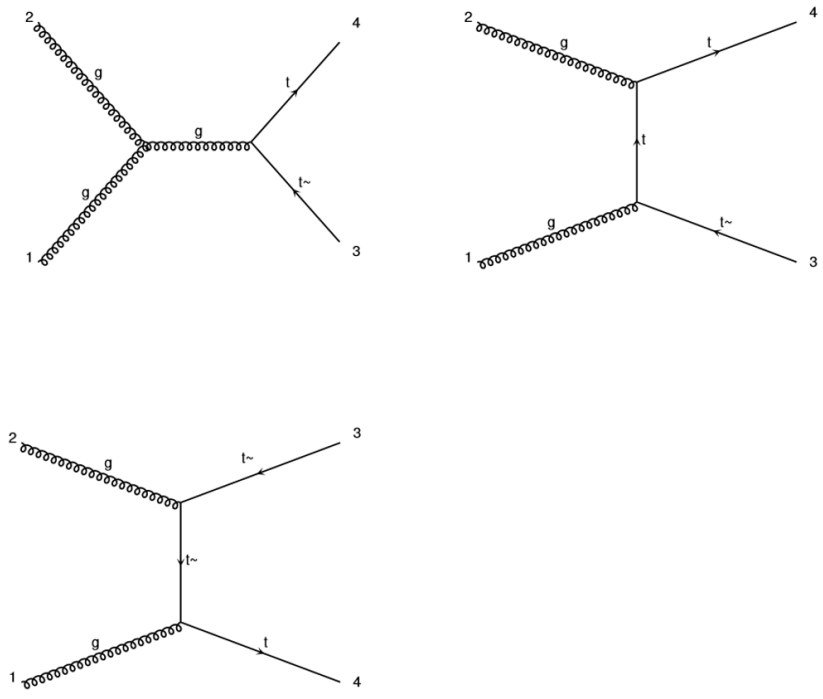

Figure 3.6.: Feynman diagrams for $gg \to t\bar{t}$

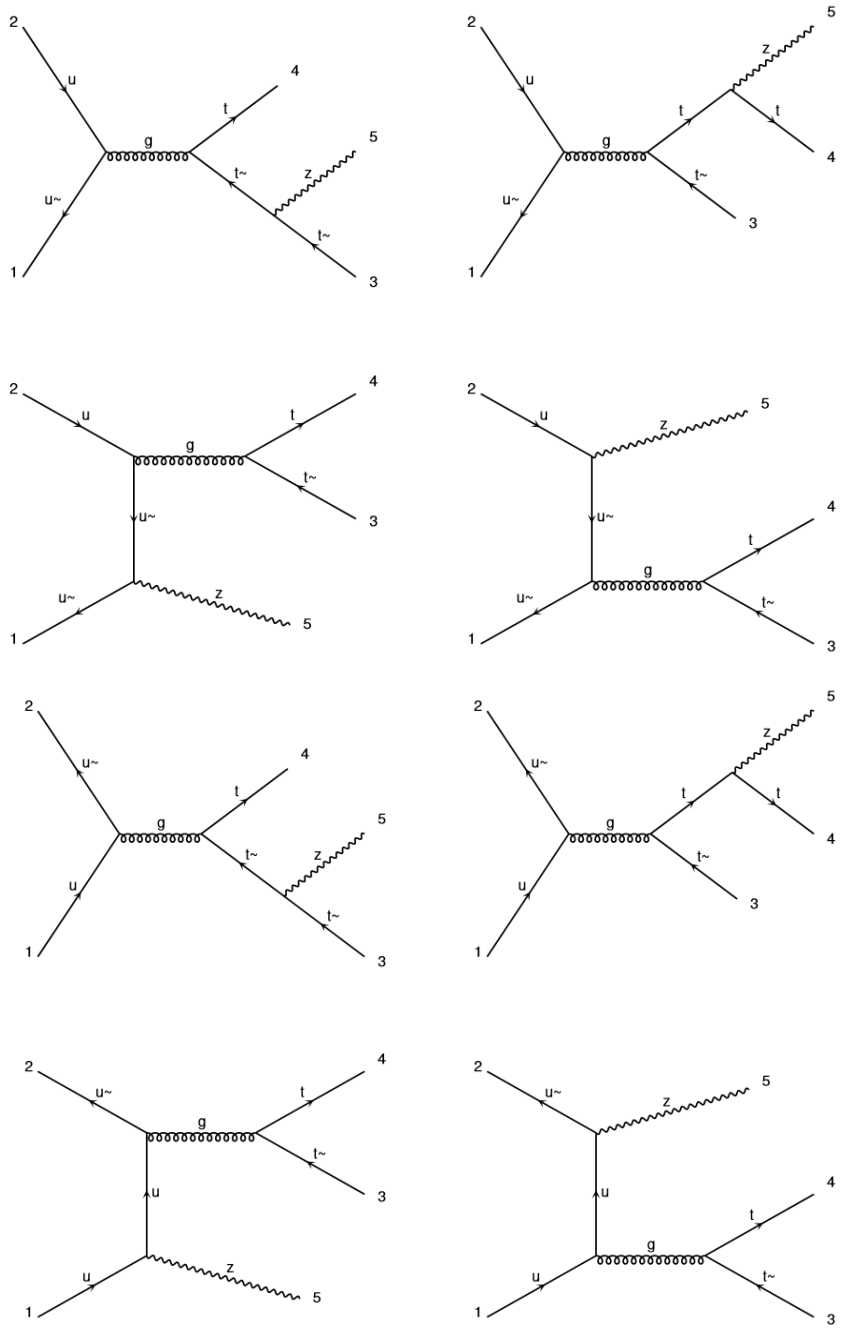

Figure 3.7.: Feynman diagrams for $u\bar{u} \to t\bar{t}Z$

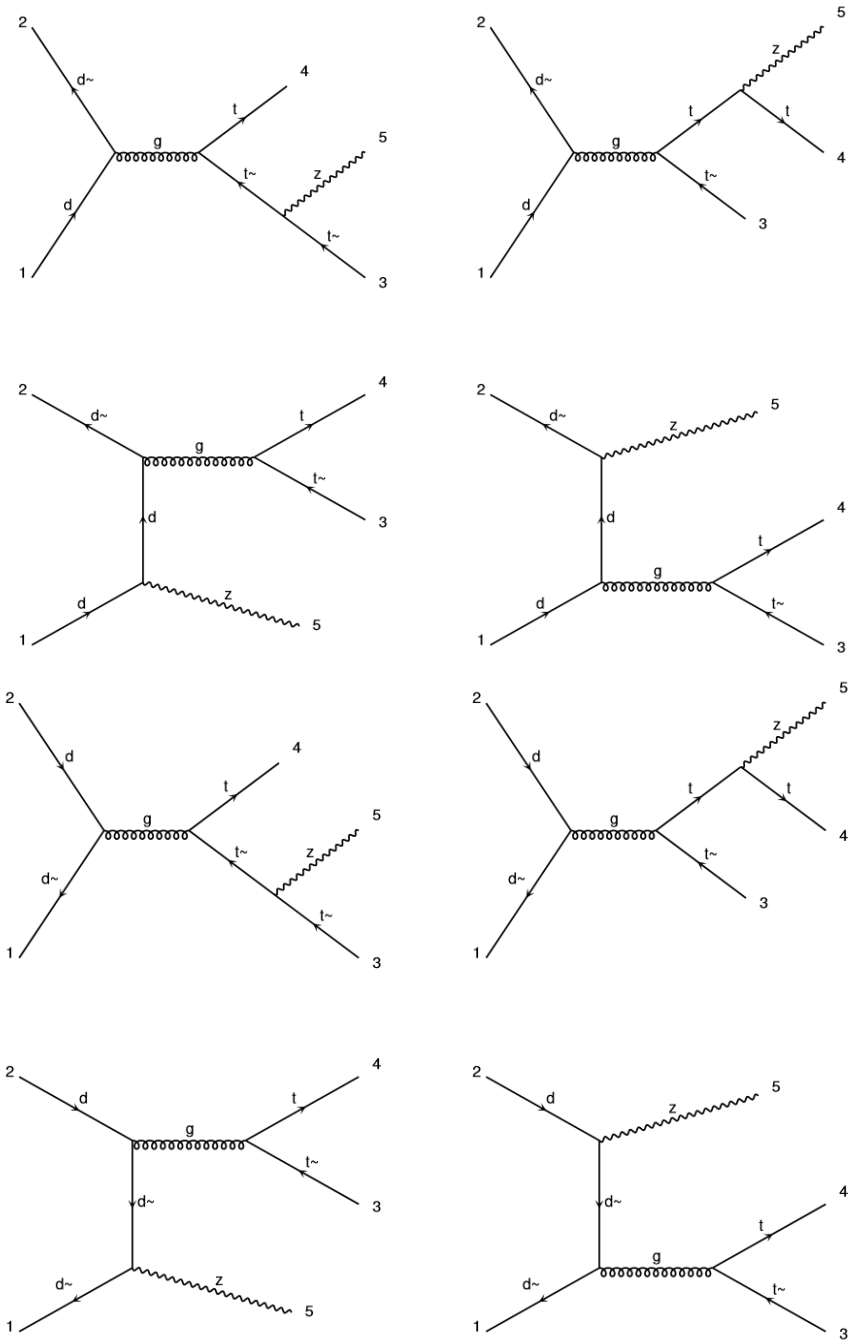

Figure 3.8.: Feynman diagrams for $d\bar{d} \to t\bar{t}Z$

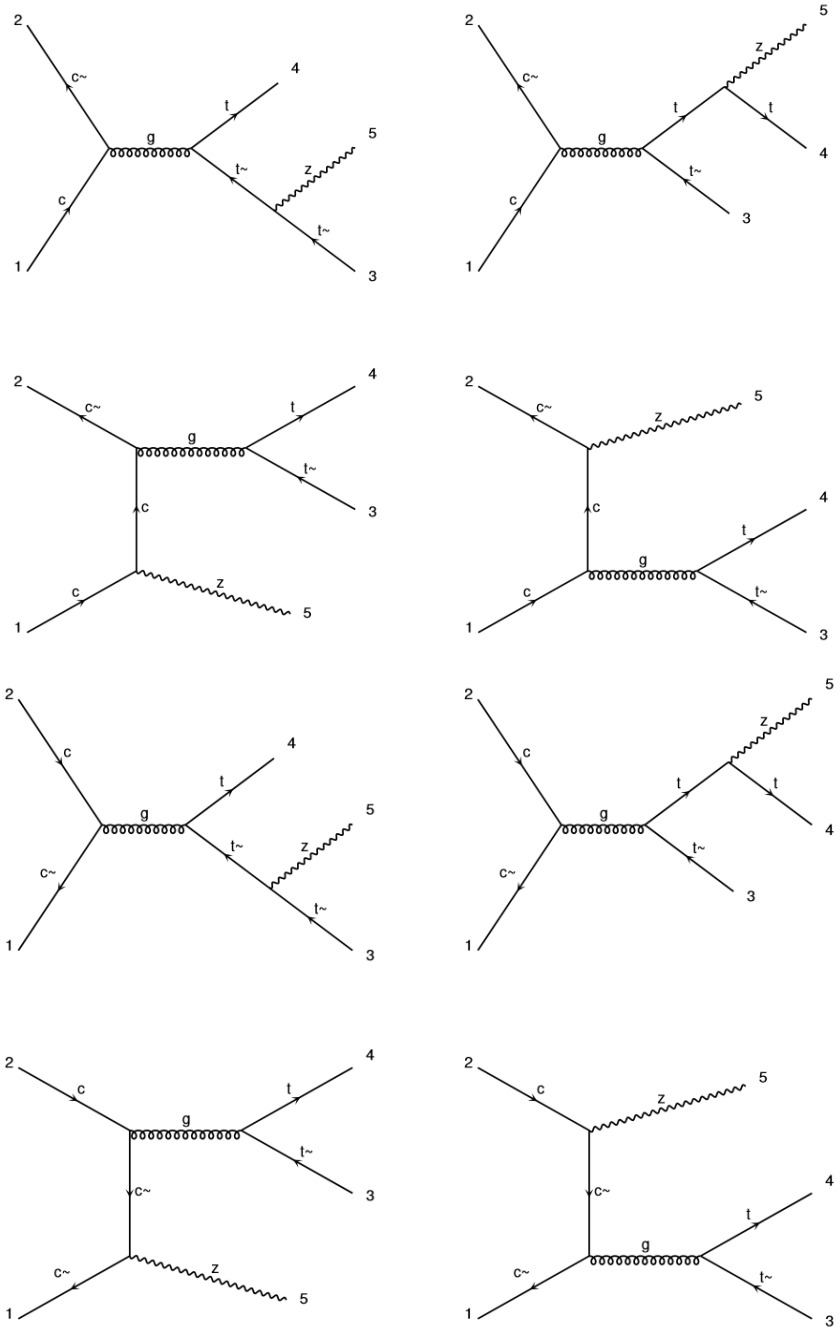

Figure 3.9.: Feynman diagrams for $c\bar{c} \to t\bar{t}Z$

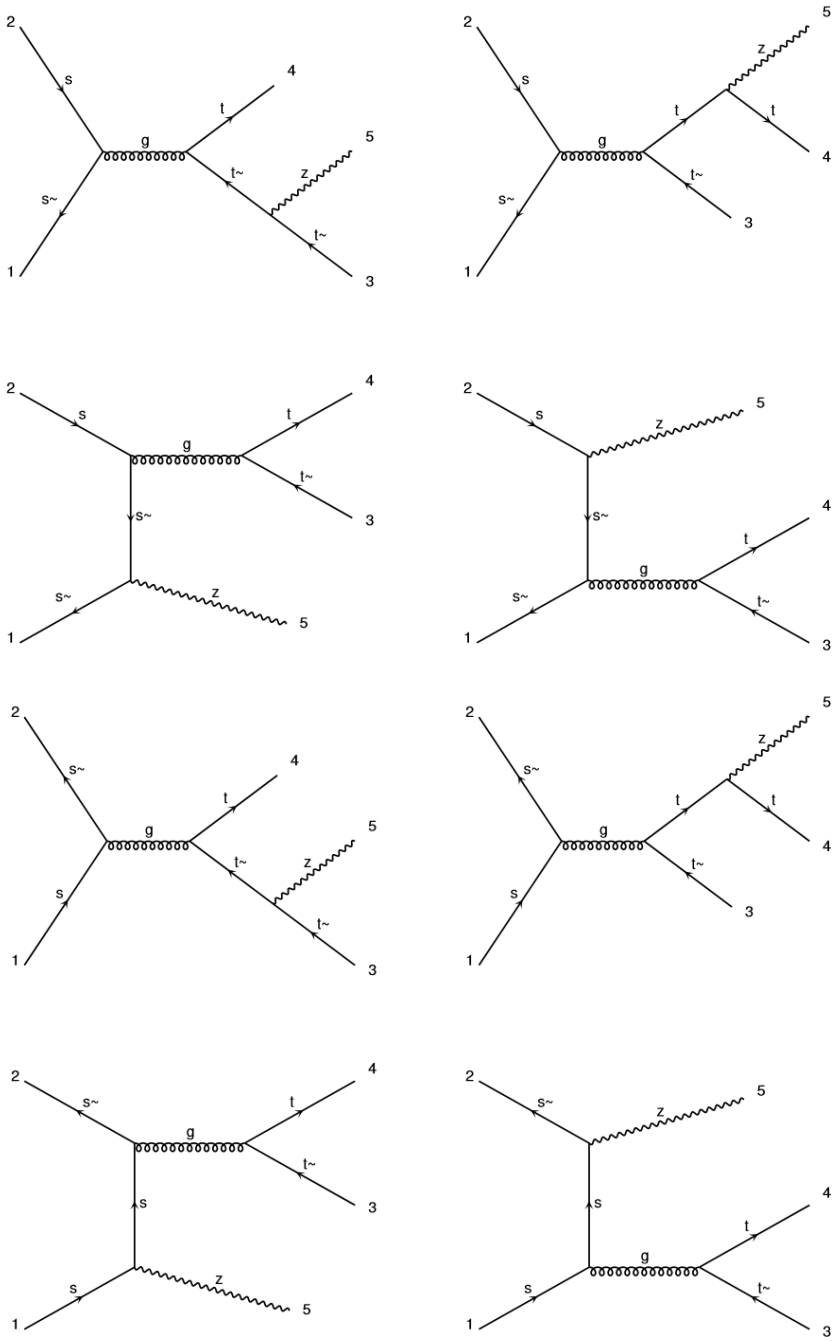

Figure 3.10.: Feynman diagrams for $s\bar{s} \to t\bar{t}Z$

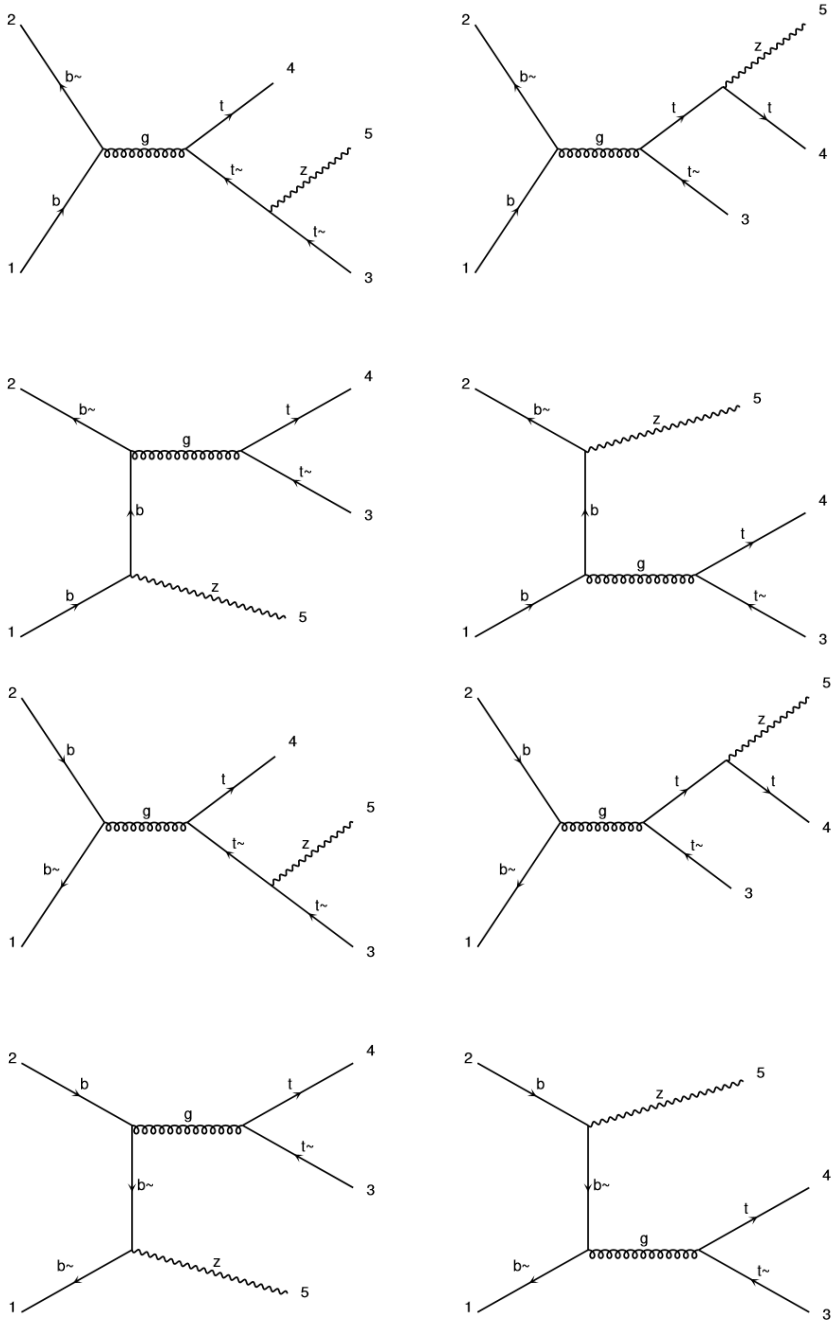

Figure 3.11.: Feynman diagrams for $b\bar{b} \to t\bar{t}Z$

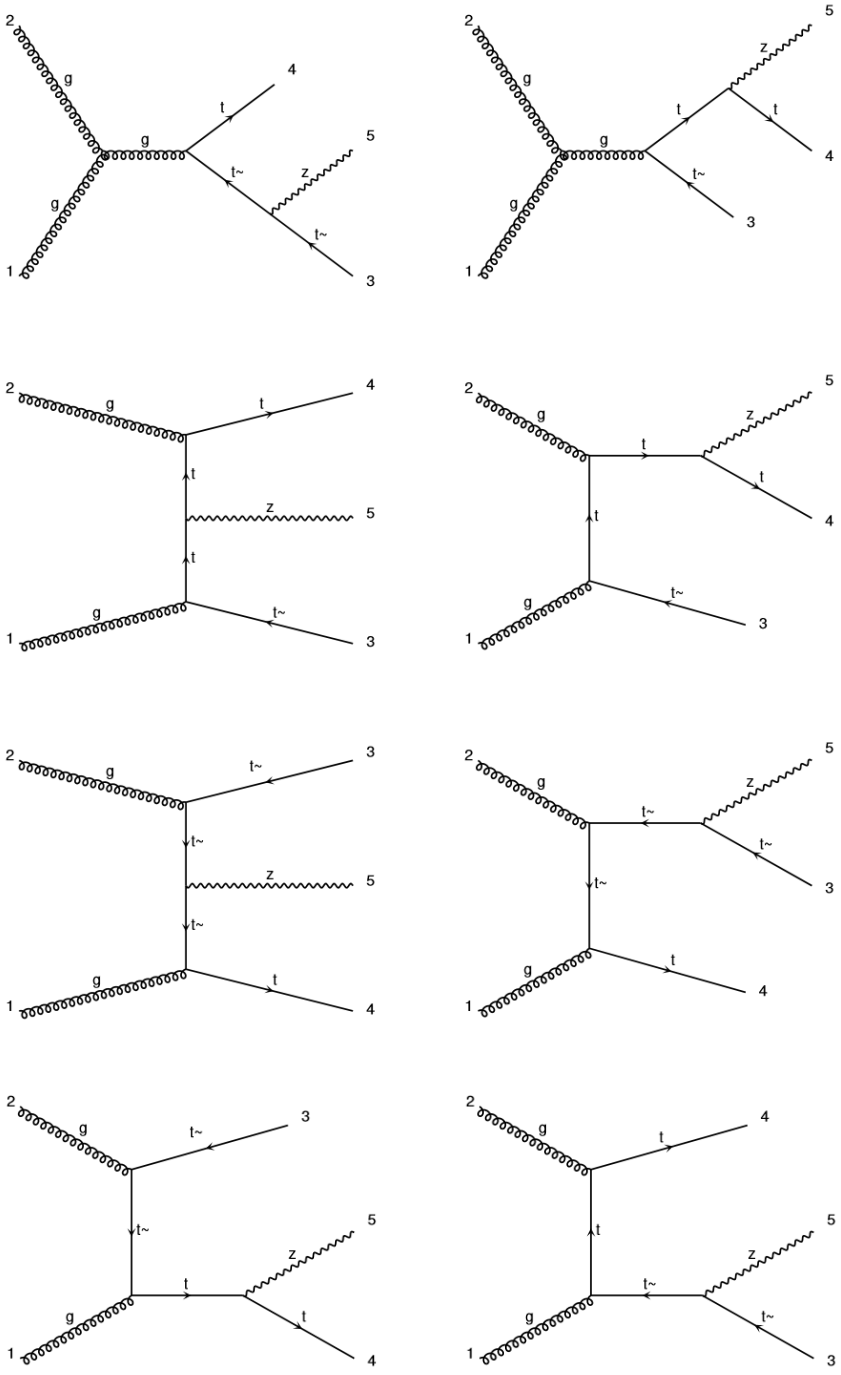

Figure 3.12.: Feynman diagrams for $gg \to t\bar{t}Z$

4. Results

From the raw output data calculated by HERWIG++, the intersections have been calculated, that mark the limit of allowed Higgs percentage within the proton without changing the cross-sections within the current experimental error bounds (c.f. Table 3.1). An exemplary plot is given in Figure 4.1 for better understanding.

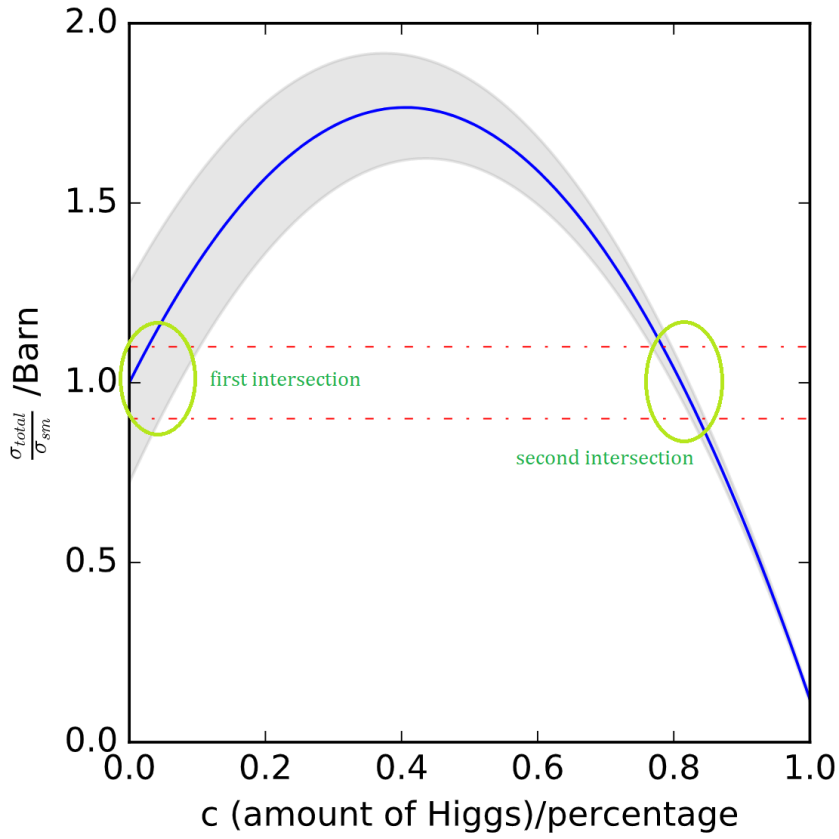

Figure 4.1.: example plot

Figure 4.1 shows an example of how the cross-section plotted over the amount of Higgs in the proton as a percentage could look like. We can see that there are two intersections in the permitted range. The two red dashed/dotted lines symbolize the experimental error

bounds, while the grey coloured band around the cross-section function represents the error made by the MC (Monte Carlo) simulation of HERWIG++. The errors occuring due to the event generator depend on a set number of events and will hereafter be referred to as "simulation errors".

4.1. Intersections

Because of the strong graphics toolbox, the three-dimensional plots have been made with MATLAB instead of PYTHON.

Let us first take a look at the results of the $t\bar{t}$-process, which are shown in the figures below.

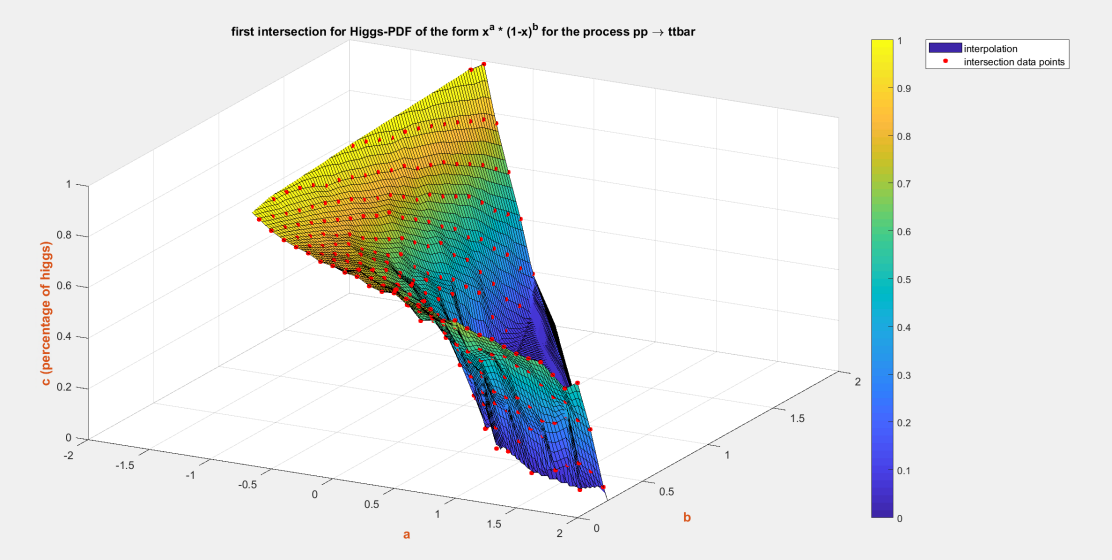

Figure 4.2.: interpolation over intersection-points for the process $pp \to t\bar{t}$

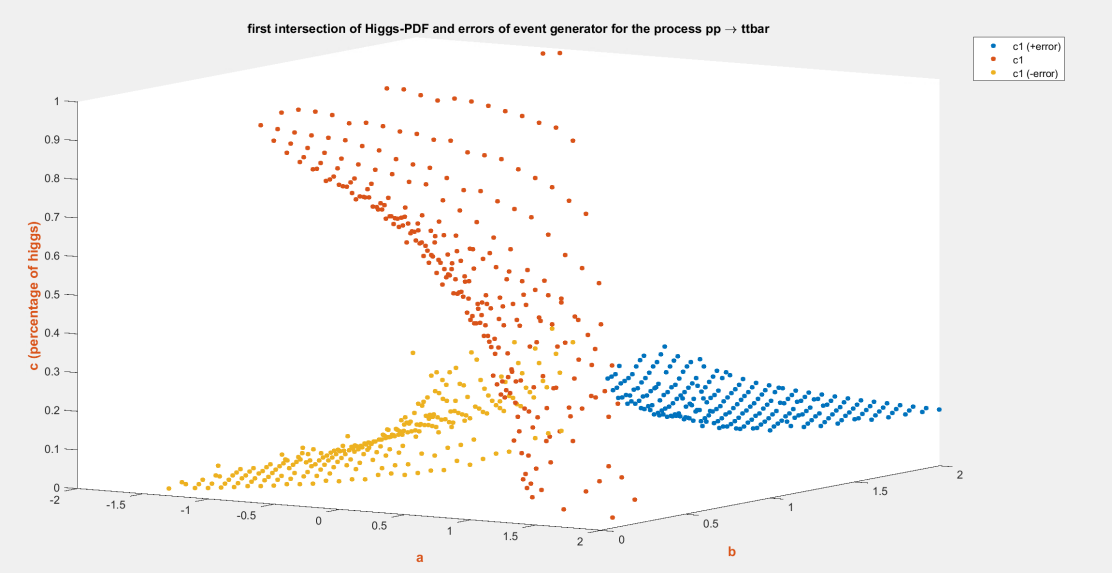

Figure 4.3.: intersection-points for the process $pp \to t\bar{t}$ including simulation errors

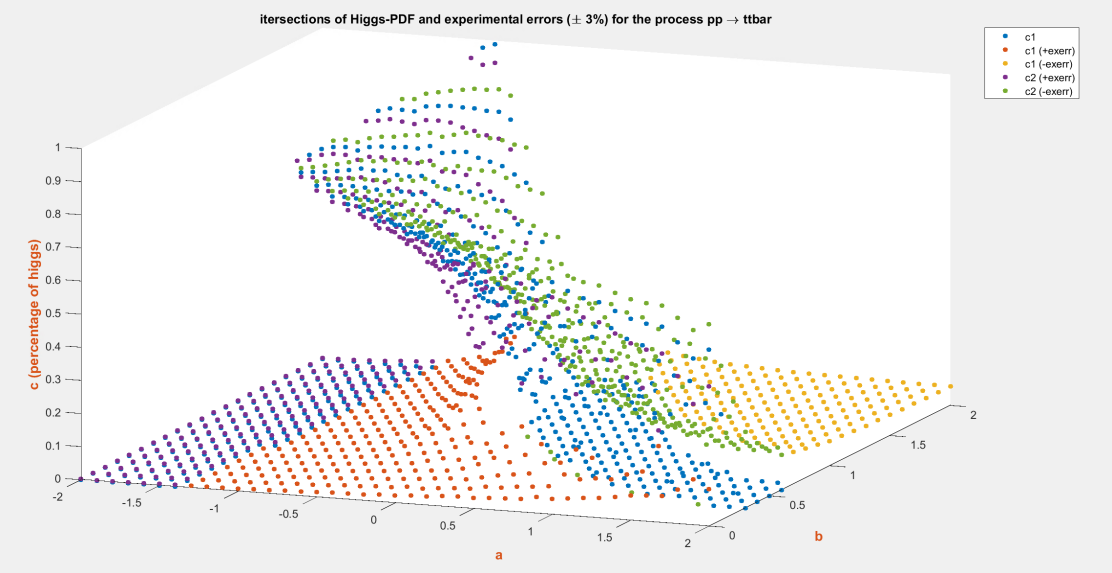

Figure 4.4.: intersection-points for the process $pp \to t\bar{t}$ including experimental errors

In order to get a better overview of the distribution of the values for allowed Higgs percentage, a histogram plot for Figure 4.3 and Figure 4.4 has been made:

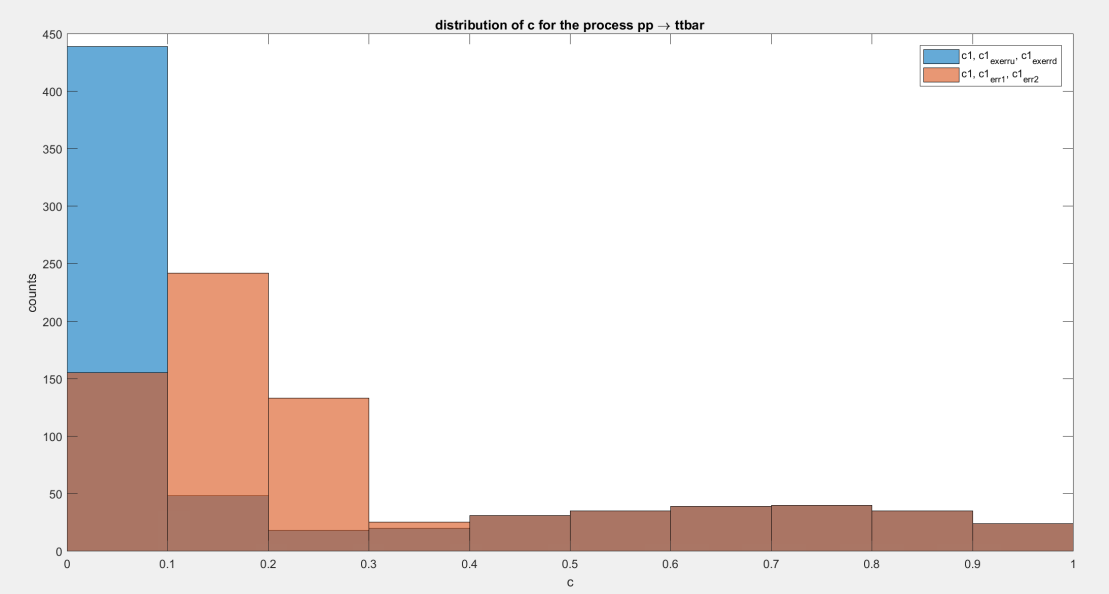

Figure 4.5.: histogram plot for the visualisation of the distribution of c-values for the process $pp \to t\bar{t}$

We can summarize the information gained from the graphical analysis thusly:

- The calculations only produce valid values for the first intersection ($0 < c < 1$).
- Most of the Higgs-PDFs yield to a Higgs percentage of 0-3%, while there exists an area where the allowed percentage reaches almost 100%.
- Within the experimental errors, a lower amount of Higgs in the proton seems to be favoured, while within the simulation errors a larger amount seems to be possible. This discrepancy can be traced back to the simulation error itself, which is still large as a consequence of the rather low chosen number of events.

After analysing the results of the process $pp \to t\bar{t}$, we can now take a look at the more sensitive process $pp \to t\bar{t}Z$.

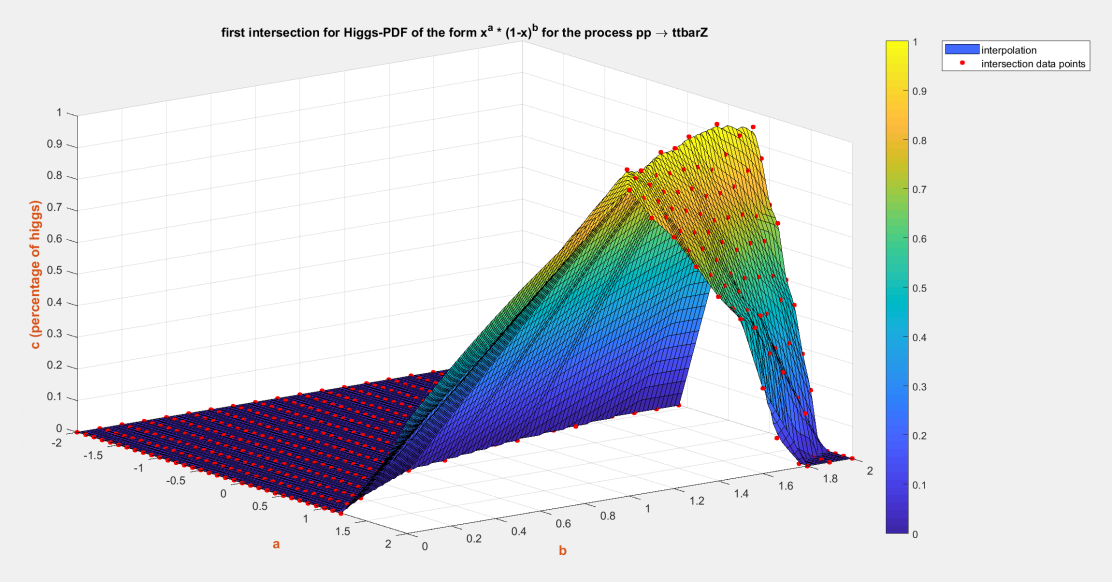

Figure 4.6.: interpolation over intersection-points for the process $pp \to t\bar{t}Z$

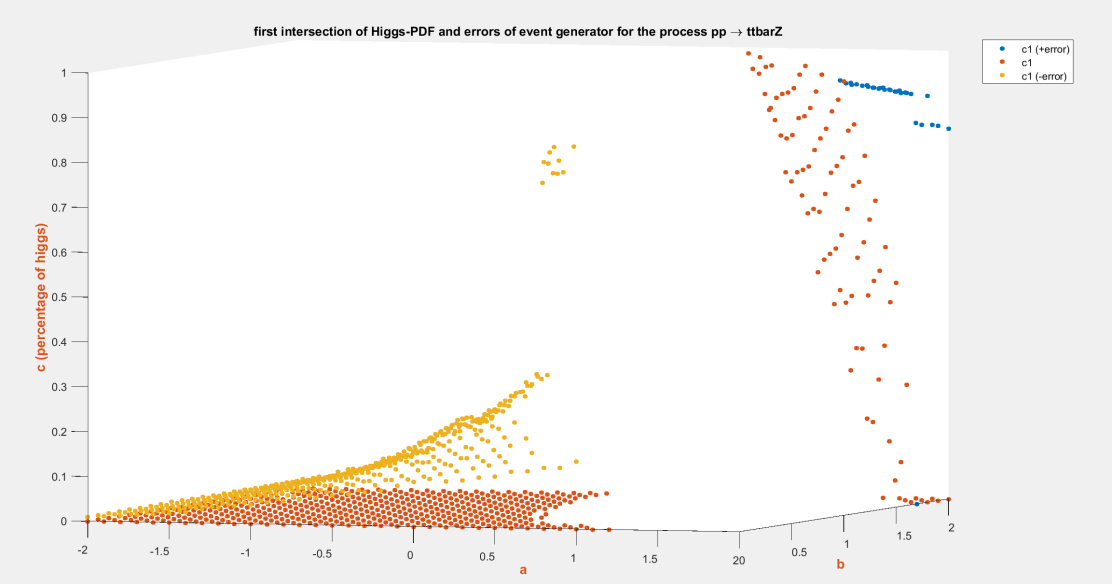

Figure 4.7.: intersection-points for the process $pp \to t\bar{t}Z$ including simulation errors

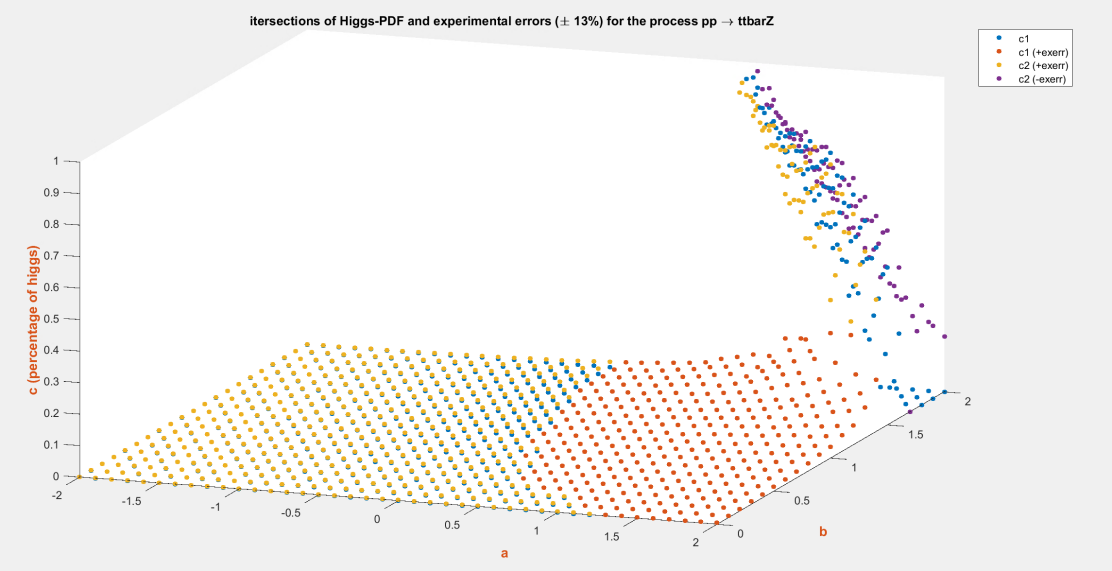

Figure 4.8.: intersection-points for the process $pp \to t\bar{t}Z$ including experimental errors

For this process as well, a histogram plot has been made for better visualisation of the distribution of c-values:

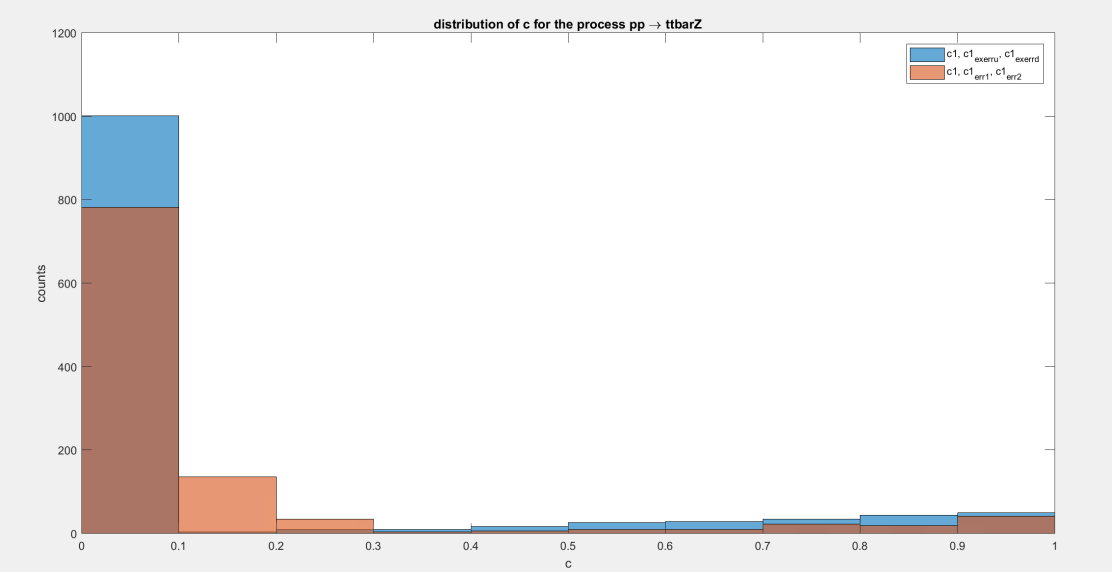

Figure 4.9.: histogram plot for the process $pp \to t\bar{t}Z$

The results of the process $pp \to t\bar{t}Z$ can be summed up to the following points:

- Again only one valid intersection.
- Histogram plot states a preference for a Higgs amount of around 0-2% for most of the used Higgs-PDFs.
- The area with a high percentage of Higgs is getting shifted to higher values for a and b compared to the former process.

Table 4.1.: summary of the most important results

process	$pp \to t\bar{t}$		$pp \to t\bar{t}Z$	
Area with the highest amount of Higgs in the proton	a	-1 to 2	a	0.5 to 2
	b	0 to 2	b	1 to 2
Most used Higgs-PDFs yield to a Higgs percentage of:	0-3%		0-2%	

4.2. Comparison between $t\bar{t}$ and $t\bar{t}Z$

Since the histogram only shows a trend of the used Higgs-PDFs regarding the distribution of c-values, to get information about the most suitable PDF and the maximum possible amount of Higgs in the proton thereof, we have to compare the two processes. In order to do so, for each data point in our set of PDFs for the two processes, the minimum c-value has been evaluated. Figuratively spoken we tried to find the intersection curve of the two interpolation surfaces ($t\bar{t}$ and $t\bar{t}Z$). Afterwards, the maximum of this set has been calculated. The results can be seen in the figures below.

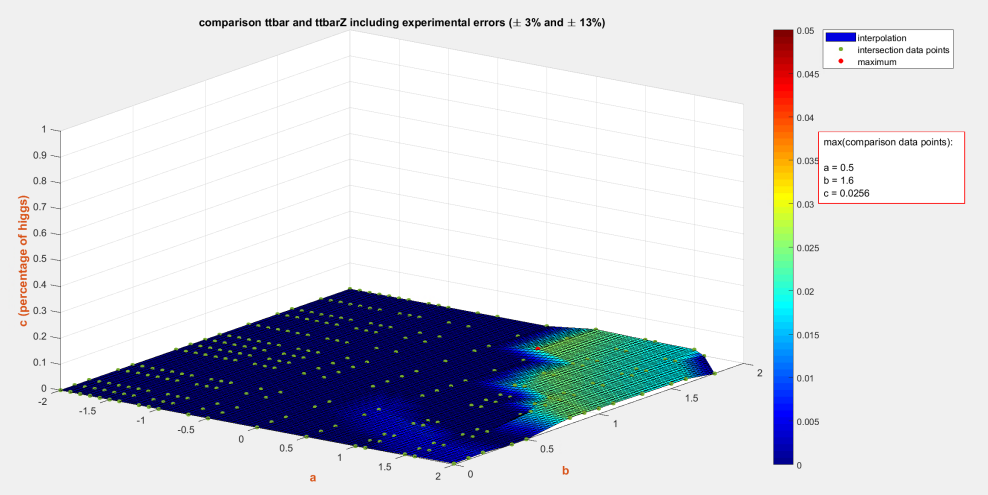

Figure 4.10.: comparison plot including the experimental errors

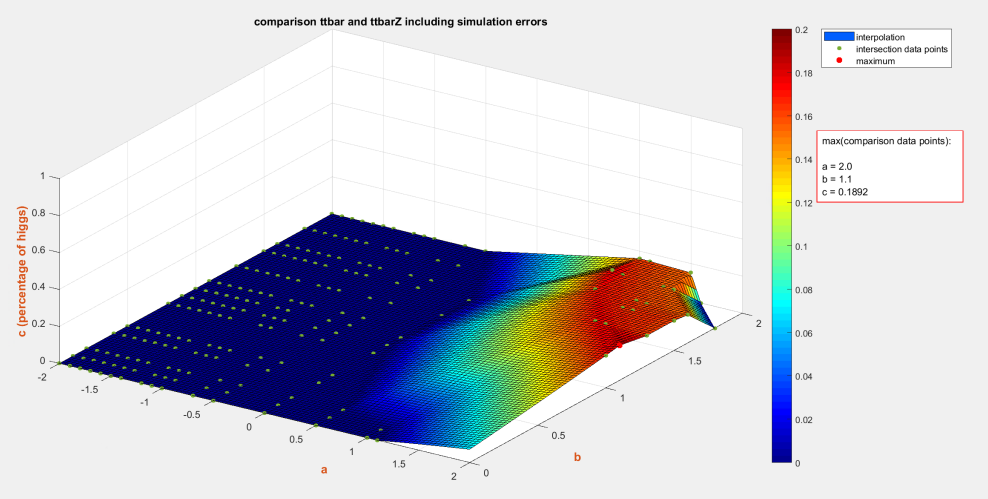

Figure 4.11.: comparison plot including the simulation errors

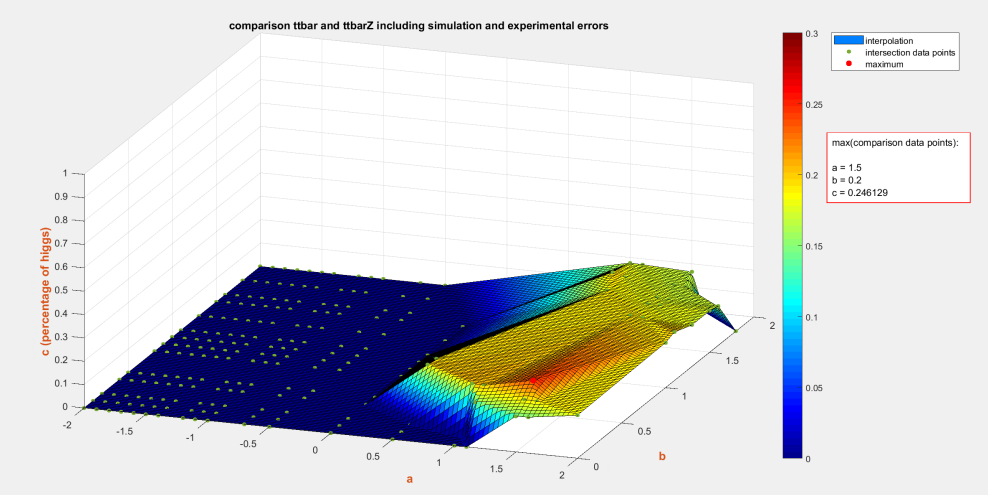

Figure 4.12.: comparison plot including experimental and simulation errors

As we can see in Figure 4.10, the comparison including the experimental errors yields to a maximum amount of Higgs of 2,56% for the Higgs-PDF $f_{\text{Higgs}} = x^{0.5} \cdot (1-x)^{1.6}$. We also notice a huge difference to the results of the comparison plot including the simulation errors (18,92%). This can be traced back to the fact that only 1000 events have been used in HERWIG++ and therefore the error is still very large. To make a more trustworthy prediction, the event generator has been run a second time with 10000 events in the area of interest, $a \in [0.5, 2]$ and $b \in [1.5, 2]$. The gained data has been analysed analogously to the description above and the resulting comparison plots can be seen below.

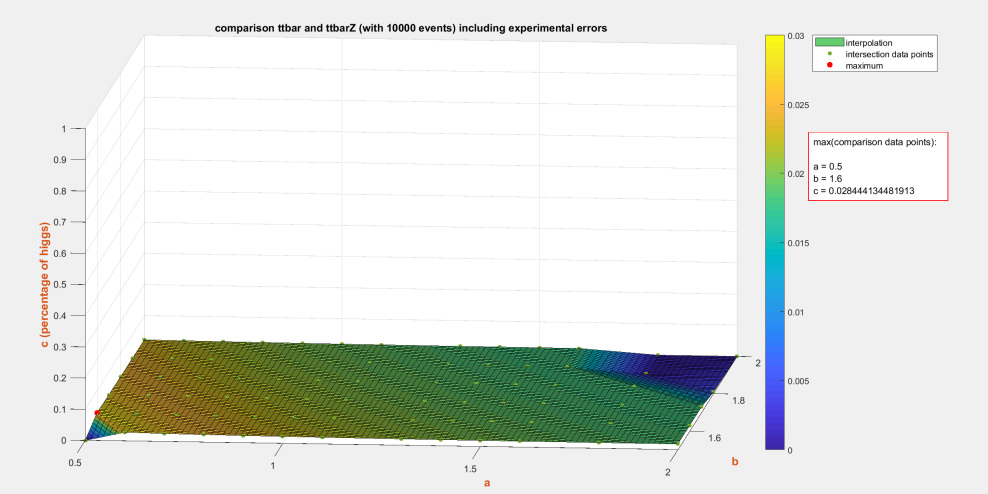

Figure 4.13.: comparison plot including the experimental errors for the second run with 10000 events

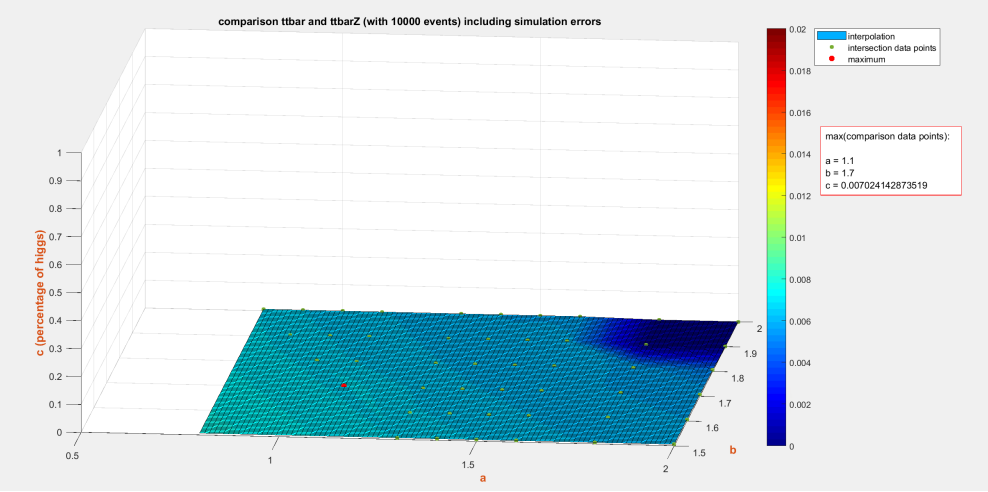

Figure 4.14.: comparison plot including the simulation errors for the second run with 10000 events

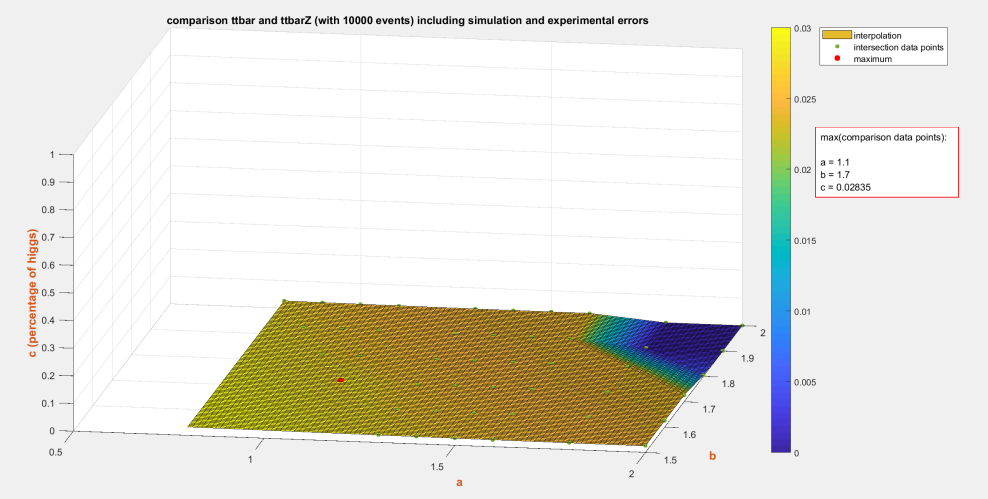

Figure 4.15.: comparison plot including experimental and simulation errors for the second run with 10000 events

As a result, we can see that the difference between the comparison plots with experimental and simulation errors has been narrowed and it seems likely that the actual Higgs percentage is about 2,8% within the current error bounds.

Assuming that the final result will not be differing too much from the one with 10000 events and included experimental errors as we increase the number of events furthermore and hence minimize the simulation error, we can note the following:

The Higgs-PDF would have the form

$$f_{\text{Higgs}} = x^{0.5} \cdot (1-x)^{1.6}$$

and yield to a maximum Higgs percentage in the proton of 2,844%.

In the following, there will be shown what the Higgs-PDF would look like in this case and in the also quiet promising case of a Higgs-PDF of the form $f_{\text{Higgs}} = x^{1.1} \cdot (1-x)^{1.7}$. Since we can still expect some deviations from the result, the nearby PDF-functions have been plotted as well.

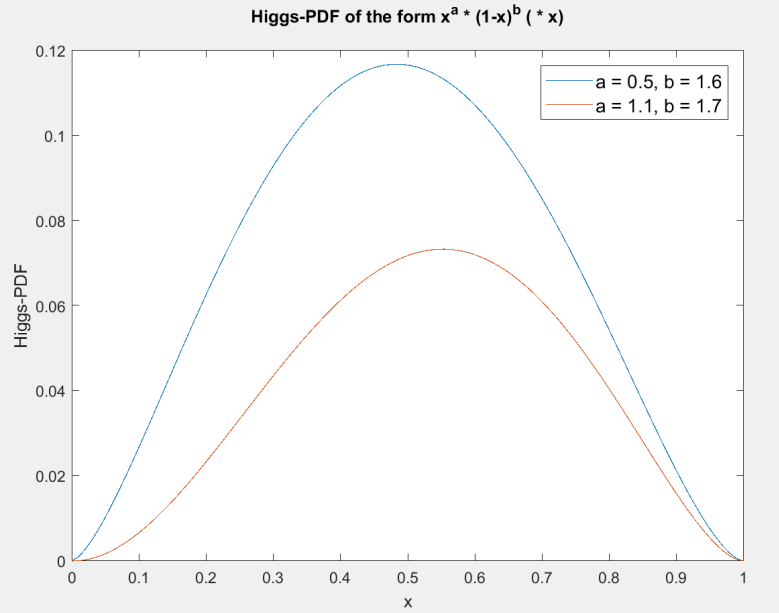

Figure 4.16.: most likely forms of the Higgs-PDF

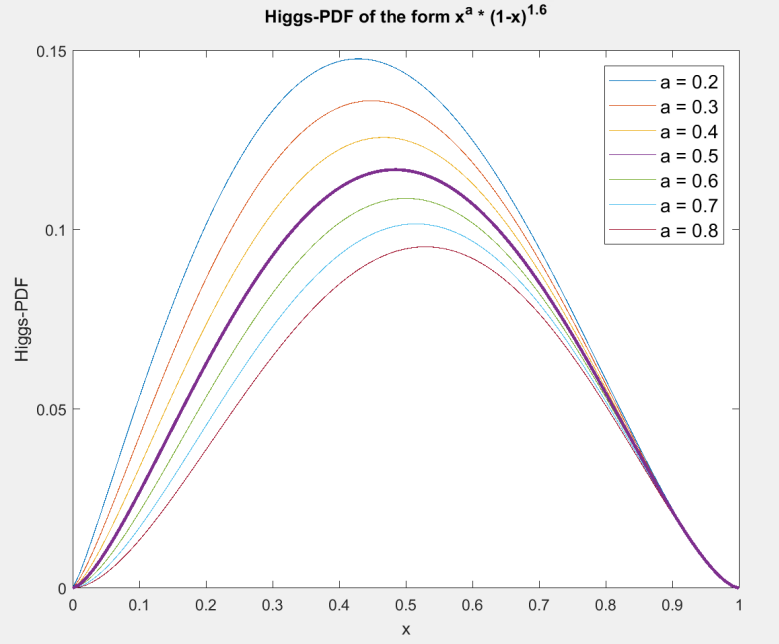

Figure 4.17.: Higgs-PDF of the form $f_{\text{Higgs}} = x^{0.5} \cdot (1-x)^{1.6}$

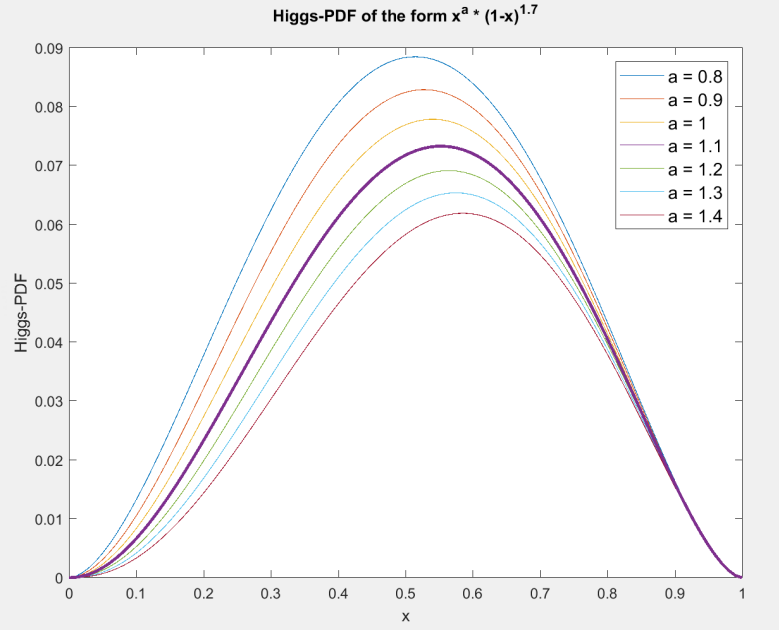

Figure 4.18.: Higgs-PDF of the form $f_{\text{Higgs}} = x^{1.1} \cdot (1-x)^{1.7}$

5. Summary

In this thesis, the form of the Higgs-PDF in the proton has been studied. For this reason, the event generator HERWIG++ has been used to simulate the proton-proton-collisions at the LHC, taking a variation of different power functions as an input parameter. The exact form of the used function as well as the varied parameters are cited in chapter 3. Since the Higgs strongly interacts with the top-quark, we focussed on the process $pp \to t\bar{t}$ as well as the process $pp \to t\bar{t}Z$ due to its higher sensitivity to the Higgs (3).

To find the most likely form of the Higgs-PDF, the intersections have been calculated from the raw data-output of the event generator, that mark the limit of the allowed Higgs percentage within the proton without changing the cross-sections within the current experimental error bound. Subsequently the results of the two processes have been compared, which led to the Higgs-PDF:

$$f_{\text{Higgs}} = x^{0.5} \cdot (1-x)^{1.6}$$

, which states a maximum Higgs percentage in the proton of 2,844%. A much higher amount of Higgs in the proton seems unlikely for this sort of Higgs-PDF.

The plot of the most promising Higgs-PDF cited above indicates, that at large x-values the up-quark stays the dominant part in the proton, while the Higgs amount evens out at a range similar to the down-quark. Depending on the exact values for a and b in the function $f_{\text{Higgs}} = x^a \cdot (1-x)^b$, the peak of the Higgs-PDF gets shifted either to a slightly higher or lower x-value.

As a consequence of limited time, there is still a lot of potential left in improving the precision of the simulation by running the event generator with more events in the area of interest ($a \in [0.5, 2]$ and $b \in [1.5, 2.0]$) and using a finer subdivision of the varied parameters of the PDF-function.

It should be mentioned, that due to some bugs in the event generators software, the exact results will differ quantitatively, but not qualitatively from the results stated here.

Future research could be focussing on a different form of a possible Higgs-PDF, taking into account the results of this thesis and the ones of Simon Fernbach's master thesis ([2]). Considering the planned building of a new circular collider at CERN (the FCC), it might become possible to reduce the current experimental limitations furthermore in future theses.

A. Code

Python Code:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed May 22 08:59:42 2019

@author: franziska
"""

import numpy as np
import os
import glob
import subprocess
import re
import matplotlib.pyplot as plt
import shutil

#%%
####################
#used functions:
####################

#checks if file is empty:

def is_non_zero_file(fpath):
    return os.path.isfile(fpath) and os.path.getsize(fpath) > 0

# calculates number of digits after decimal point

def num_after_point(x):
    st = str(x)
    if not '.' in st:
        return 0
    return len(st) - st.index('.') - 1


def findSignDig(numb):

    #returns significant digits as integer
    #if significant numbers are after '.' , a minus is added before the
    returned value
    #positive returned values are the numbers of digits before the '.'
    #also numbers in scientific notation can be put as input parameter

```

```
    numb1 = float(numb)
    numb = str(numb1)

    if not '.' in numb:
        return len(numb)
    else:

        while numb[len(numb)-1] == '0':
            numb = numb[:-1]

        if (len(numb) - numb.index('.') -1) == 0:
            numb = numb[:-1]
            return len(numb)
        else:
            return -(len(numb) - numb.index('.') - 1)

#functions for plot:

def sig_tot(c):
    return ((1-c)**2)*(sm*10**(-9)) +(1-c)*c*(GH*10**(-9)) +(c**2)*(HH
    *10**(-9))

def sig_err1(c):
    return ((1-c)**2)*((sm*10**(-9))+(sm_err*10**(-9))) +(1-c)*c*((GH
    *10**(-9))+(GH_err*10**(-9))) +(c**2)*((HH*10**(-9))+(HH_err*10**(-9))
    )

def sig_err2(c):
    return ((1-c)**2)*((sm*10**(-9))-(sm_err*10**(-9))) +(1-c)*c*((GH
    *10**(-9))-(GH_err*10**(-9))) +(c**2)*((HH*10**(-9))-(HH_err*10**(-9))
    )


# solve quadratic equation (intersections):

def quadsol(a,b,c):
    q = -0.5*(b+np.sign(b)*np.sqrt(b*b-4.0*a*c))
    x1 = q/a
    x2 = c/q
    return x1, x2

#%%

# initializing lists for intersections:

c1 = []    #intersections at 1
c2 = []
c1_exerru = []  #intersections at 1 + experimental error
c2_exerru = []
c1_exerrd = []  #intersections at 1 - experimental error
c2_exerrd = []
c1_err1 = []    #intersections +err at 1
c2_err1 = []
c1_err1u = []   #intersections +err at 1 + experimental error
```

```
c2_err1u = []
c1_err1d = []    #intersections +err at 1 - experimental error
c2_err1d = []
c1_err2 = []     #intersections -err at 1
c2_err2 = []
c1_err2u = []    #intersections -err at 1 + experimental error
c2_err2u = []
c1_err2d = []    #intersections -err at 1 - experimental error
c2_err2d = []

a_list = []
b_list = []

#%%

###############
#generate PDF
###############

#generate coeff. for PDF

start_a = -2.0
stop_a = 2.0
step_a = 0.1
i_a = int(((stop_a-start_a)/step_a)+1)
#print(i_a)

a = np.linspace(start_a,stop_a,i_a)

start_b = 0.0
stop_b = 2.0
step_b = 0.1
i_b = int(((stop_b-start_b)/step_b)+1)
#print(i_b)

b = np.linspace(start_b,stop_b,i_b)


#generate PDF x^a*(1-x)^b

PDF_new = []
ab_list = []

for an in a:
    for bn in b:

        an = round(an,1)
        bn =round(bn,1)
        ab_list.append([an,bn])

        PDF_New = '  return x*(pow(x,{:3.1f})*pow((1.L-x),{:3.1f}));\n'.
    format(an, bn)
        PDF_new.append(PDF_New)

```

```
145 #%%
146
147 ################
148 #replace PDF
149 ################
150
151
152 # call HiggsPDF.cc and read old PDF:
153
154 for outfile in glob.glob(os.path.join('/home/franziska/Dokumente/
        herwigfiles','HiggsPDF.cc')):
155     data = open(outfile,'r')
156     f = data.readlines()
157     PDF_old = f[39]
158
159
160 # replace old PDF with new one:
161
162 for outfile in glob.glob(os.path.join('/home/franziska/Dokumente/
        herwigfiles','HiggsPDF.cc')):
163     f = open(outfile,'r')
164     filedata = f.read()
165     f.close()
166
167 index = 0;
168
169 for line in PDF_new:
170     print('\033[1;34mMessage: New PDF is used! \033[1;m')
171     newdata = filedata.replace(PDF_old,line)
172
173     for outfile in glob.glob(os.path.join('/home/franziska/Dokumente/
        herwigfiles','HiggsPDF.cc')):
174         f = open(outfile,'w')
175         f.write(newdata)
176         f.close()
177
178
179 ##############################
180 # change to home directory:
181 ##############################
182
183     os.chdir('/home/franziska/Dokumente/herwigfiles/')
184
185 ###################################
186 #start Herwig and generate events
187 ###################################
188
189     #call shell script:
190     subprocess.call(['./run.sh'])
191
192 ############################
193 #stop if no data exists or file ist empty:
194 ############################
195     str_out= {'HH.out','GH.out','sm.out'}
```

```
196
197     if any(os.path.isfile('/home/franziska/Dokumente/herwigfiles/' +i) ==
        False for i in str_out):
198         index +=1
199         continue
200
201     files = ['GH.hepmc','GH.log','GH.out','GH.tex','GH.yoda','GH-EvtGen.
        log','HH.hepmc','HH.log','HH.out','HH.tex','HH.yoda','HH-EvtGen.log','
        sm.hepmc','sm.log','sm.out','sm.tex','sm.yoda','sm-EvtGen.log']
202
203     if any(is_non_zero_file('/home/franziska/Dokumente/herwigfiles/' +i)
        == False for i in str_out):
204         for doc in files:
205             try:
206                 os.chdir('/home/franziska/Dokumente/herwigfiles/')
207                 subprocess.call(['rm',doc])
208             except:
209                 continue
210         index +=1
211         continue
212     else:
213         pass
214
215 ############################
216 #save a and b:
217 ############################
218
219     a_list.append(ab_list[index][0])
220     b_list.append(ab_list[index][1])
221     index +=1
222
223 ################################
224 #make new output directory:
225 ################################
226     i = 0
227     exists = True
228
229     while exists == True:
230         i +=1
231         exists = os.path.isdir('/home/franziska/Dokumente/herwigfiles/
        output/out%i' %i)
232
233     os.chdir('/home/franziska/Dokumente/herwigfiles/output/')
234     name = 'out%i' %i
235     subprocess.call(['mkdir',name])
236
237     path = '/home/franziska/Dokumente/herwigfiles/output/'+name
238
239 ########################
240 #save PDF to txt file
241 ########################
242     path_txt = path +'/PDF.txt'
243     file = open(path_txt,'w')
244     file.write(line)
```

```
245     file.close()
246 
247 ######################
248 #read output files:
249 ######################
250 
251     # extract sigma_total in nb (!) of out-file
252 
253     for outfile in glob.glob(os.path.join('/home/franziska/Dokumente/
    herwigfiles','*.out')):
254         data = open(outfile,'r')
255 
256         f = data.readlines()
257         l = len(f)-2
258         s = f[l]
259         number = re.compile('-?\ *[0-9]+\.?[0-9]*(?:[Ee]\ *-?\ *[0-9]+)?'
    )
260         si = [float(x) for x in re.findall(number,s[61:78])]
261         print(si)
262 
263         name = s[3:7]
264         if name == 'h0,h':
265 
266             HH1 = '%ie%a' %(si[0],si[2])
267             HH,x = [float(x) for x in re.findall(number,HH1)]
268 
269             HH = si[0]*10**si[2]
270 
271             if num_after_point(HH) > abs(si[2])+10:
272                 HH = round(HH,int(abs(si[2])))          # eliminate round
    -off errors
273 
274             if not si[2] == 0:                          # calculate right
     amount of digits for error
275                 HH_err= si[1]*10**si[2]
276             else:
277                 d = findSignDig(HH)
278                 if d > 0:
279                     HH_err = si[1]*10**(d-1)
280                 else:
281                     HH_err = si[1]*10**(d)
282 
283 
284         elif name =='g,g-':
285 
286             sm1 = '%ie%a' %(si[0],si[2])
287             sm,x = [float(x) for x in re.findall(number,sm1)]
288 
289             sm = si[0]*10**si[2]
290 
291             if num_after_point(sm) > abs(si[2])+10:
292                 sm = round(sm,int(abs(si[2])))          # eliminate round
    -off errors
293 
```

```
294             if not si[2] == 0:                              # calculate right
    amount of digits for error
295                 sm_err= si[1]*10**si[2]
296             else:
297                 d = findSignDig(sm)
298                 if d > 0:
299                     sm_err = si[1]*10**(d-1)
300                 else:
301                     sm_err = si[1]*10**(d)
302
303
304         else:
305             GH1 = '%ie%a' %(si[0],si[2])
306             GH,x = [float(x) for x in re.findall(number,GH1)]
307
308             GH = si[0]*10**si[2]
309
310             if num_after_point(GH) > abs(si[2])+10:
311                 GH = round(GH,int(abs(si[2])))          # eliminate round
    -off errors
312
313             if not si[2] == 0:                              # calculate right
    amount of digits for error
314                 GH_err= si[1]*10**si[2]
315             else:
316                 d = findSignDig(GH)
317                 if d > 0:
318                     GH_err = si[1]*10**(d-1)
319                 else:
320                     GH_err = si[1]*10**(d)
321
322
323
324 ##########################
325 #calculate total sigma
326 ##########################
327     path_fig = path + '/sig_tot.png'
328
329     c = np.linspace(0,1,100)
330
331     plt.clf()
332 #     plt.plot(c,sig_tot(c))
333 #     plt.axhline(y=sig_tot(0)*0.9,color='r',linestyle='-.',linewidth =
    0.5)
334 #     plt.axhline(y=sig_tot(0)*1.1,color='r',linestyle='-.',linewidth =
    0.5)
335 #     plt.fill_between(c,sig_err1(c),sig_err2(c),color='gray',alpha=0.2)
336 #
337 #     plt.axis(xmin = 0,xmax = 1, ymin=0, ymax = 1e-9)
338 #     plt.xlabel('c (amount of Higgs)/percentage')
339 #     plt.ylabel(r'$\sigma_{total}/Barn$')
340 #     plt.savefig(path_fig)
341
342     plt.plot(c,(sig_tot(c)/(sm*10**-9)))
```

```
343     plt.axhline(y=0.97,color='r',linestyle='-.',linewidth = 0.5)
344     plt.axhline(y=1.03,color='r',linestyle='-.',linewidth = 0.5)
345     plt.fill_between(c,(sig_err1(c)/(sm*10**-9)),(sig_err2(c)/(sm*10**-9)
        ),color='gray',alpha=0.2)
346 
347     plt.axis(xmin = 0,xmax = 1)
348     plt.xlabel('c (amount of Higgs)/percentage')
349     plt.ylabel(r'$\frac{\sigma_{total}}{\sigma_{sm}}$' +' /Barn')
350     plt.savefig(path_fig)
351 
352 ####################################
353 # nb in b:
354 ####################################
355 
356     HH =  HH*10**-9
357     HH_err =  HH_err*10**-9
358     GH =  GH*10**-9
359     GH_err = GH_err*10**-9
360     sm =  sm*10**-9
361     sm_err =  sm_err*10**-9
362 
363     print('\033[1;34mMessage: generated values: \033[1;m')
364     print('HH = ', HH)
365     print('HH_err = ', HH_err)
366     print('GH = ',GH)
367     print('GH_err = ', GH_err)
368     print('sm = ',sm)
369     print('sm_err = ',sm_err)
370     print(" ")
371 
372 ######################################
373 # Calculate intersections:
374 ######################################
375     abz = [1,1.03,0.97]
376     a_arr = "(HH-GH+sm)/sm", "((HH+HH_err) - (GH+GH_err) + (sm+sm_err))/
        sm", "((HH-HH_err) - (GH-GH_err) + (sm-sm_err))/sm"
377 
378     for a in a_arr:
379         if a == "(HH-GH+sm)/sm":
380             a = (HH-GH+sm)/sm
381             b = (GH-2*sm)/(sm)
382             for i in abz:
383                 c = 1-i
384                 x1,x2 = quadsol(a,b,c);
385                 if i == 1:
386                     #print("c1 = ", x1, " c2 = ",x2)
387                     c1.append(x1); c2.append(x2);
388                 elif i == 1.03:
389                     #print("c1_exerru = ", x1, " c2_exerru = ",x2)
390                     c1_exerru.append(x1); c2_exerru.append(x2);
391                 else:
392                     #print("c1_exerrd = ", x1, " c2_exerrd = ",x2)
393                     c1_exerrd.append(x1); c2_exerrd.append(x2);
394 
```

```
395         elif a == "((HH+HH_err) - (GH+GH_err) + (sm+sm_err))/sm":
396             a = ((HH+HH_err) - (GH+GH_err) + (sm+sm_err))/sm
397             b = ((GH+GH_err) -2*(sm+sm_err))/(sm)
398             for i in abz:
399                 c = ((sm+sm_err)/(sm))-i
400                 x1,x2 = quadsol(a,b,c);
401                 if i == 1:
402                     #print("c1_err1 = ", x1, " c2_err1 = ",x2)
403                     c1_err1.append(x1); c2_err1.append(x2);
404                 elif i == 1.03:
405                     #print("c1_err1u = ", x1, " c2_err1u = ",x2)
406                     c1_err1u.append(x1); c2_err1u.append(x2);
407                 else:
408                     #print("c1_err1d = ", x1, " c2_err1d = ",x2)
409                     c1_err1d.append(x1); c2_err1d.append(x2);
410 
411         else:
412             a = ((HH-HH_err) - (GH-GH_err) + (sm-sm_err))/sm
413             b = ((GH-GH_err) -2*(sm-sm_err))/(sm)
414             for i in abz:
415                 c = ((sm-sm_err)/(sm))-i
416                 x1,x2 = quadsol(a,b,c);
417                 if i == 1:
418                     #print("c1_err2 = ", x1, " c2_err2 = ",x2)
419                     c1_err2.append(x1); c2_err2.append(x2);
420                 elif i == 1.03:
421                     #print("c1_err2u = ", x1, " c2_err2u = ",x2)
422                     c1_err2u.append(x1); c2_err2u.append(x2);
423                 else:
424                     print("c1_err2d = ", x1, " c2_err2d = ",x2)
425                     c1_err2d.append(x1); c2_err2d.append(x2);
426 
427 
428 ########################################
429 #save all generated files to out file:
430 ########################################
431     os.chdir('/home/franziska/Dokumente/herwigfiles')
432 
433     files = ['GH.hepmc','GH.log','GH.out','GH.tex','GH.yoda','GH-EvtGen.
    log','HH.hepmc','HH.log','HH.out','HH.tex','HH.yoda','HH-EvtGen.log','
    sm.hepmc','sm.log','sm.out','sm.tex','sm.yoda','sm-EvtGen.log']
434 
435     for f in files:
436         try:
437             shutil.move(f,path)
438         except:
439             continue
440 
441 
442 ##################################################
443 # store calculated intersections in .dat-file:
444 ##################################################
445 
446     print(len(a_list))
```

```
    print(len(b_list))
    print(len(c1))
    print(len(c2))
    print(len(c1_exerru))
    print(len(c2_exerru))
    print(len(c1_exerrd))
    print(len(c2_exerrd))
    print(len(c1_err1))
    print(len(c2_err1))
    print(len(c1_err1u))
    print(len(c2_err1u))
    print(len(c1_err1d))
    print(len(c2_err1d))
    print(len(c1_err2))
    print(len(c2_err2))
    print(len(c1_err2u))
    print(len(c2_err2u))
    print(len(c1_err2d))
    print(len(c2_err2d))


surfplot_1 = np.column_stack((a_list,b_list,c1,c2))
header1 = "a,b,c1, c2"
np.savetxt("/home/franziska/Dokumente/herwigfiles/Auswertung/surfplot_1.
    dat",surfplot_1,header = header1)

values1 = np.column_stack((c1_exerru,c2_exerru,c1_exerrd,c2_exerrd))
header2 = "c1_exerru, c2_exerru, c1_exerrd, c2_exerrd"
np.savetxt("/home/franziska/Dokumente/herwigfiles/Auswertung/surfplot_2.
    dat",values1,header = header2)

values2 = np.column_stack((c1_err1,c2_err1,c1_err1u,c2_err1u,c1_err1d,
    c2_err1d))
header3 = "c1_err1,c2_err1,c1_err1u,c2_err1u,c1_err1d,c2_err1d"
np.savetxt("/home/franziska/Dokumente/herwigfiles/Auswertung/surfplot_3.
    dat",values2,header = header3)

values3 = np.column_stack((c1_err2,c2_err2,c1_err2u,c2_err2u,c1_err2d,
    c2_err2d))
header4 = "c1_err2,c2_err2,c1_err2u,c2_err2u,c1_err2d,c2_err2d"
np.savetxt("/home/franziska/Dokumente/herwigfiles/Auswertung/surfplot_4.
    dat",values3,header = header4)



surfplot = np.column_stack((a_list,b_list,c1,c2,c1_exerru,c2_exerru,
    c1_exerrd,c2_exerrd,c1_err1,c2_err1,c1_err1u,c2_err1u,c1_err1d,
    c2_err1d,c1_err2,c2_err2,c1_err2u,c2_err2u,c1_err2d,c2_err2d))
header = "a,b,c1, c2, c1_exerru, c2_exerru, c1_exerrd, c2_exerrd, c1_err1
    , c2_err1, c1_err1u, c2_err1u, c1_err1d, c2_err1d, c1_err2, c2_err2,
    c1_err2u, c2_err2u, c1_err2d, c2_err2d\n"
np.savetxt("/home/franziska/Dokumente/herwigfiles/Auswertung/surfplot.dat
    ",surfplot,header = header)
```

bash file:

```
#!/bin/bash

ve() { source $1/bin/activate; }  #shell function to activate virtualenv

ve /usr/local/herwig    #activate herwig

#Plugin:
make IntrinsicHiggs.so
rivet-buildplugin TTBAR.cc
export RIVET_ANALYSIS_PATH=/home/franziska/Dokumente/herwigfiles/

find -type d -name '*scratch*' -exec rm -rf {} \;

#event generator:
Herwig read HH.in
Herwig run HH.run -N 1000

find -type d -name '*scratch*' -exec rm -rf {} \;

Herwig read GH.in
Herwig run GH.run -N 1000

find -type d -name '*scratch*' -exec rm -rf {} \;

Herwig read sm.in
Herwig run sm.run -N 1000
```

Analogue Code for the other process!

References

[1] "Brout-Englert-Higgs physics: From foundations to phenomenology"; A. Maas, 2017, (1712.04721)

[2] "Higgs-PDF study in proton-proton collisions"; Simon Fernbach; Master thesis

[3] "perturbative quantum gauge theories on manifolds with boundary"; Alberto s. Cattaneo, Pavel Mnev and Nicolai Reshetikhin; 2016; (1507.01221)

[4] http://cdsweb.cern.ch/record/794322/# (24.07.2019; 16:10h)

[5] M. Aaboud *et al.* [ATLAS Collaboration], Phys. Lett. B **784** (2018) 173 doi:10.1016/j.physletb.2018.07.035 [arXiv:1806.00425 [hep-ex]].

Literature

[A+05] S. I. Alekhin et al. DGLAP evolution and parton fits. In *HERA and the LHC: A Workshop on the implications of HERA for LHC physics: Proceedings Part A*, pages 119–159, Geneva, 2005. CERN, CERN.

[B+08] M. Bahr et al. Herwig++ Physics and Manual. *Eur. Phys. J.*, C58:639–707, 2008.

[Ber14] C. Berger. *Elementarteilchenphysik: Von den Grundlagen zu den modernen Experimenten*. Springer-Lehrbuch. Springer Berlin Heidelberg, 2014.

[HK13] S. Heusler and T. Kugo. *Eichtheorie*. Springer Berlin Heidelberg, 2013.

[Lin19] Huey-Wen Lin. Parton Distribution Functions and Lattice QCD. *EPJ Web Conf.*, 206:01003, 2019.

[Maa15] Axel Maas. Field theory as a tool to constrain new physics models. *Mod. Phys. Lett.*, A30(29):1550135, 2015.

[Maa19] Axel Maas. Brout-Englert-Higgs physics: From foundations to phenomenology. *Prog. Part. Nucl. Phys.*, 106:132–209, 2019.

[Pla11] Ringaile Placakyte. Parton Distribution Functions. In *Proceedings, 31st International Conference on Physics in collisions (PIC 2011): Vancouver, Canada, August 28-September 1, 2011*, 2011.

[SM13] Michael H. Seymour and Marilyn Marx. Monte Carlo Event Generators. In *Proceedings, 69th Scottish Universities Summer School in Physics : LHC Phenomenology (SUSSP69): St.Andrews, Scotland, August 19-September 1, 2012*, pages 287–319, 2013.

[Str15] Franco Strocchi. Symmetries, Symmetry Breaking, Gauge Symmetries. 2015.