ETP-Bachelor-KA/2022-11

Karlsruhe Institute of Technology

Monte-Carlo-Simulations for tt+X-Events at the CMS-Experiment

Bachelor Thesis

Sergey Lelyakin

At the Department of Physics
Institute of Experimental Particle Physics

Reviewer:	Prof. Dr. Ulrich Husemann
Second reviewer:	Dr. Michael Wassmer
Advisor:	Emanuel Pfeffer

Karlsruhe, 31. October 2022

This thesis has been accepted by the first reviewer of the bachelor thesis.

PLACE, DATE

.....

(**Prof. Dr. Ulrich Husemann**)

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

PLACE, DATE

.....

(**YOUR NAME**)

Contents

1 Introduction

The discipline of particle physics studies the fundamental rules of the universe on a sub-microscopic level. Insight into these is obtained by observing the interactions of fundamental particles at extremely high energies. To reach such energies and to observe the processes that occur, huge, complex and expensive arrangements of particle accelerators and detectors have to be used. The data obtained from these can then be used to support or falsify theories that seek to describe the laws governing those processes.

Almost just as important as the data obtained from real experiments is simulated data. After all, due to the inherent randomness of quantum mechanical processes, obtaining meaningful information from high-energy physics experiments is only possible by collecting and statistically analyzing large amounts of data. To compare this to theoretical predictions, similar amounts of data need to be simulated based on the theory to be tested.

Such simulations generally consist of three steps: The parton level, where the high-energy quantum mechanical processes between fundamental particles are simulated using precise analytical equations arising from underlying theory in combination with numerical methods; the generator level, where the decay and hadronization of the parton level reaction products are simulated; and the detector level, where the response of the experimental setup to the reaction products is simulated.

For each of these steps, various software packages capable of performing them exist. At the Compact Muon Solenoid (CMS) experiment, GEANT is typically used for the detector level simulation. While this allows for highly precise simulations, it has shortcomings, most notably the high computational cost required. In the case of applications for which high precision plays less of a role, a potential alternative presents itself in Delphes.

The objective of this thesis is to investigate the degree to which Delphes is usable as a substitution for GEANT, specifically in the case of tt+X events, that is, events including the associated production of a top quark-antiquark pair and some other reaction products. This means attempting to produce simulated data using Delphes which is as close as possible to the simulations made using GEANT. To do so, a workflow which isolates the effect of varying the detector level simulation is devised. Using this, all deviations between the GEANT-based results and the Delphes-based results are investigated, an attempt is made to determine their origin and, when possible, to mitigate them through adjusting the configuartion of the Delphes software. By comparing the deviation between Delphes

and GEANT across simulations of different parton level processes, the universality of the findings is gauged.

This thesis begins by introducing basic theoretical concepts in chapter 2. Next, in chapter 3, the experimental environment that is simulated is introduced. Chapter 4 gives an overview of the simulation workflow and the software used. After that, chapter 5 details the specific processes that are simulated and chapter 6 gives a technical description of the analysis parameters. Finally, chapter 7 presents the results of the research, and chapter 8 concludes the thesis by summarizing the work and giving an outlook towards further research.

2 Theoretical foundations

This chapter briefly describes the theoretical basis necessary for understanding the physical processes investigated in this thesis. As these fundamentals constitute the basis of all modern particle physics, plenty of literature exists which describes them in far greater depth and detail than is done here. One such work which can corroborate most of the information found in this chapter is [1].

2.1 The standard model

The standard model of particle physics is a theoretical model which forms the basis for all modern particle physics. Its predictions historically show good agreement with experimental data and it has predicted many discoveries of the late 20th and early 21st centuries, one of the most famous examples being the experimental discovery of the Higgs boson in 2012 [2, 3] which was predicted theoretically in 1964 [4–6].

Despite this, it should be noted that the standard model cannot describe all of physics. Most notably, it does not describe gravity in any way, being limited to only the other three fundamental forces. Further evidence for the incompleteness of the standard model include, among others, neutrino oscillations, which contradict the idea of massless neutrinos, and dark matter, for which the standard model offers no particle it could be identified with. Nevertheless, the standard model proves completely sufficient for the purposes of this thesis.

The standard model gives a description of elementary particles and their interaction via the electromagnetic, weak and strong forces based on theoretical symmetries in the model.

The fundamental forces work via exchange particles called gauge bosons. These are spin-1 particles that include the gluons, the photon, as well as the Z- and $W^{\pm}$-bosons. Gluons are exchange particles of the strong force. They have no mass or electromagnetic charge, but they do have color charge, which enables strong interactions between the gluons themselves. Photons are exchange particles of the electromagnetic force. They have no mass or charge, allowing the electromagnetic force to function at long distances. Z- and $W^{\pm}$-bosons are exchange particles of the weak interaction. Due to their high mass, the weak interaction only occurs at short distances.

To explain the properties of the weak interaction, the standard model unifies the electromagnetic and weak interaction into the electroweak interaction and introduces the

Higgs-mechanism. As a consequence of this, the existence of a spin-0 Higgs boson is predicted.

Besides the bosons, the standard model includes spin-$\frac{1}{2}$ particles called fermions, along with a corresponding antiparticle for each of them. These consist of six quarks: the up-, down-, strange-, charm-, bottom- and top-quarks, three charged leptons: the electron, muon and tauon, and three neutrinos, one corresponding to each charged lepton.

Quarks possess mass, electromagnetic charge, color charge and a weak isospin, allowing them to interact through all three interactions. Due, in part, to the self-interaction of gluons, a phenomenon known as confinement disallows the existence of non-zero color charge on macroscopic scales. This means that quarks can never be observed in isolation, but rather, form composite particles called hadrons. Any attempt to separate quarks from one another (such as imparting very high energies to them in a collider) results in a new quark-antiquark pair being generated from the strong field between them, thus producing two new hadrons.

Charged leptons possess mass, electromagnetic charge and a weak isospin, but no color charge, thus being subject to only the electromagnetic and weak interactions.

Neutrinos, finally, possess no mass, electric charge or color charge, and can thus only interact weakly.

Each of these fermions has a corresponding antiparticle of equal mass but, if applicable, opposite quantum numbers.

The specific reactions allowed between these particles can be derived based on so-called Feynman rules arising from the theory, and can be visualized using Feynman diagrams.

While the standard model gives qualitative predictions for all of this, a number of free parameters related to particle masses and coupling strengths of certain interactions remain, and must be determined experimentally. The ability to measure these in different ways and obtain consistent results is a major validation test for the standard model.

2.2 Hadron collider physics

The hadrons which are made to collide with each other in a hadron collider are not elementary particles. Instead, they consist of many partons. In the case of protons which are used in the events investigated in this thesis, these are three valence quarks, gluons arising from the strong interaction between them, as well as temporary quark-antiquark pairs called sea quarks which can appear from a gluon.

When a proton-proton collision actually happens in a hadron collider, in almost all cases, only two individual partons actually interact with one another. To determine which kind of particle these are and what momentum they carry during the interaction, parton distribution functions (PDFs) are used which give the probability of finding a given parton with a given fraction of the proton's momentum. These PDFs are not derived from theory and must be determined experimentally, most commonly using electron-proton scattering. They are further dependent on the magnitude of the momentum transfer in the scattering process, although this dependency is described analytically.

These PDFs have another use: The main high-energy hard process which is typically most interesting in a collider experiment must be separated from the soft low-energy sub-processes, such as collinear gluon radiation, happening at the same time. This separation is achieved using the factorization theorem [7], which includes the effects of the soft processes in the PDFs, while separating them from the hard process, allowing the latter to be calculated in perturbation theory as an isolated interaction between two partons.

3 The CMS Experiment

While this work deals only with simulations explicitly, these simulations ultimately aim to mimic the processes at a real hadron collider experiment: The Compact Muon Solenoid (CMS) detector at the Large Hadron Collider (LHC). This chapter aims to briefly describe the relevant technical aspects behind these.

3.1 The LHC

The LHC is a synchrotron-type particle accelerator with a circumference of 27 km, located at the European Organization for Nuclear Research (CERN). It is capable of generating proton-proton collisions at center-of-mass energies of up to over 13 TeV at four points along its circumference.

A detector is placed at each of these points: These are ALICE, ATLAS, CMS and LHCb. Each of these detectors serves a slightly different purpose and is thus built in a different way. CMS, the detector relevant for this work, is described in the next section.

The total instantaneous luminosity of the LHC, which describes the number of potential collision events per unit of time and area, is designed to reach $10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$, although twice as much was reached in reality. In practice, this translates into an actual frequency of events happening of over 1 GHz [8, 9].

3.2 The CMS detector

The CMS detector surrounds the collision points from all sides. It has a barrel-like shape, with two distinct parts: The barrel part which detects reaction products with largee scattering angles, and the two endcaps which detect almost collinear reaction products. A slice through the barrel part is shown in figure 3.1.

The innermost part of the detector consists of a tracking system. This allows for the tracking of the flight paths of any charged particle. Outside is the electromagnetic calorimeter (ECAL). It is designed to detect, stop, and measure the energy of electrons and photons. Behind the ECAL there is a hadronic calorimeter (HCAL) which does the same to any hadronic reaction products. All of that is encased inside a superconducting solenoid which produces a magnetic field inside the detector. This field allows to determine the sign of a particle's electric charge and the magnitude of its transverse momentum based on the

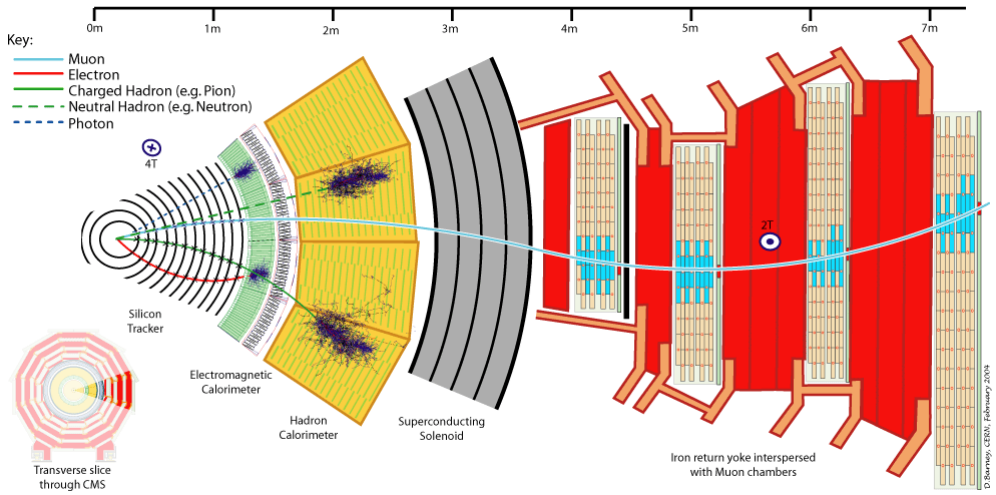

Figure 3.1: A slice of the CMS detector, including the various sub-detectors and the expected behaviour of different particles inside the detector. [10]

curvature of its flight path through the tracker. Outside the solenoid, finally, there is the muon system designed to detect muons, which pass through all aforementioned elements almost unimpeded. A more detailed description of the detector is found in [11].

3.3 Basic kinematic quantities at CMS

The convention for defining the coordinate axes in the CMS detector has the x-axis pointing towards the center of the LHC accelerator ring, the y-axis pointing straight up, and the z-axis along the beam axis. In this coordinate system, a particle's kinematic properties can be described using its 4-momentum $(p_x, p_y, p_z, E)^T$. For practical purposes, it is often useful to use the transverse momentum p_T, the pseudorapidity η, the azimuthal angle ϕ, and the mass m. The transverse momentum is the momentum in the direction orthogonal to the z-axis:

$$p_\mathrm{T} = \sqrt{p_x^2 + p_y^2}. \tag{3.1}$$

The pseudorapidity is a measure of the particle's inclination along the beam axis, defined as

$$\eta = -\ln \tan\left(\frac{\theta}{2}\right), \tag{3.2}$$

where θ is the angle between the momentum 3-vector $\vec{p}$ and the z-axis. Thus, η is 0 when $\vec{p}$ is orthogonal to the beam axis and approaches infinity or negative infinity as $\vec{p}$ becomes closer to the beam axis.

The azimuthal angle is calculated in cylinder coordinates around the z-axis. The particle's mass can be calculated as

$$m = \sqrt{E^2 - \vec{p}^2}. \tag{3.3}$$

Another two relevant observables are ΔR for a pair of objects and the invariant mass m_inv for any group of objects. The former is the cartesian distance of two objects in the

η-ϕ-plane. Note that the cylindricall symmetry of the coordinate system must be considered properly when calculating this. The latter is the total energy of a group of objects in its own center-of-mass frame.

3.4 Event reconstruction

Determining the actual final-state particles of the event from the detector data is done using the particle-flow algorithm [12]. Broadly speaking, five different kinds of particles are distinguished: Electrons are detected in the ECAL and have a matching track in the tracker system. Photons are detected in the ECAL but do not have a track. Muons pass the calorimeters almost unimpeded and are thus detected only by the tracker and the muon system. Charged hadrons leave a track and are detected in the HCAL, although they may also deposit a minor fraction of their energy inside the ECAL. Neutral hadrons, finally, leave no track and are detected inside the HCAL.

High-energy quarks that result from a hard process are unstable and radiate gluons, which themselves produce further quark-antiquark pairs. At lower energies, due to the confinement mechanism described in section 2.1, these quarks hadronize into particles with neutral color charge. These are not necessarily stable and may decay further. In total, a single parton that is a product from the hard interaction typically produces a large number of separate hadrons reaching the detector. The momenta of such a group of hadrons, however, will be correlated. Thus, these hadrons may be algorithmically clustered into so-called jets to give approximate information on the original reaction products. At CMS, the specific algorithm used for this clustering is anti-k_t [13].

Another procedure to obtain information on the reaction products which is relevant to this thesis is b-tagging. Jets originating from bottom quarks can be distinguished from other jets based on the kinematic properties of the decay products, most notably the larger decay vertex offsets arising due to the comparatively long decay times of hadrons containing b-quarks. At CMS, this is done using the neural network based DeepJet algorithm [14]. The result of the algorithm is a number (“tag”) between 0 and 1 assigned to each jet which indicates the network’s confidence in the jet being a b-jet.

4 Simulated processes

The specific events which are investigated in this thesis are the production of a top quark-antiquark pair together with a bottom quark-antiquark pair, called ttbb processes, in proton-proton collisions at the LHC at a center of mass energy of 13 TeV.

Investigations of this process are interesting due to the inherent complexity of modeling it, as it involves the two heaviest of the standard model quarks. Additionally, general ttbb events constitute a large irreducible background in analyses of ttH(bb) events. These are events in which a top quark-antiquark pair is produced together with a Higgs boson, which subsequently decays to a bottom quark-antiquark pair (figure 4.1), and they are a notable channel for Higgs boson production [15].

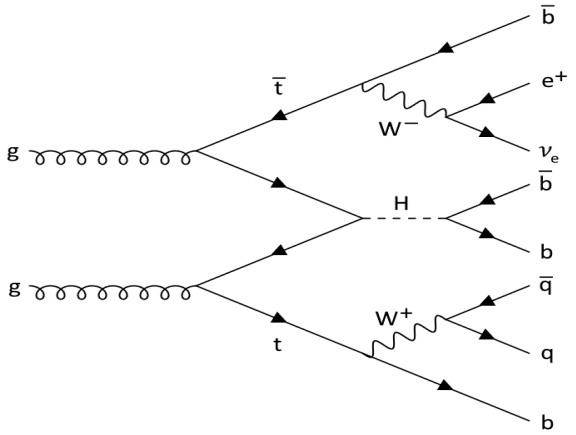

Figure 4.1: Example of a leading-order Feynman diagram for the ttH(bb) process, including subsequent top quark decays.

As this research is intended for being used for differentiating ttH events from ttbb events, ttH is also investigated for comparison with ttbb. Additionally, ttZ processes are also simulated due to their inherent similarity to ttH, with all relevant Feynman diagrams being identical save for a substitution of H bosons for Z bosons.

While it is common practice to separate all these processes into dilepton, single-lepton and full hadronic channels based on the decay modes of the W bosons that originate from the top quark decay, this is not done in this thesis, meaning that the simulations used

consider all these channels simultaneously. The methodology used, however, remains fully applicable to these more specific channels.

5 Simulation tools

Aside from Delphes, the simulation tools used in this thesis are standard for high energy particle physics simulations at CMS. Furthermore, they are the same tools used to create the simulations which this thesis uses as a reference to compare its results to.

When simulating a process in high energy particle physics, there are several stages the simulation goes through, each performed by a largely separate piece of software, as illustrated in figure 5.1.

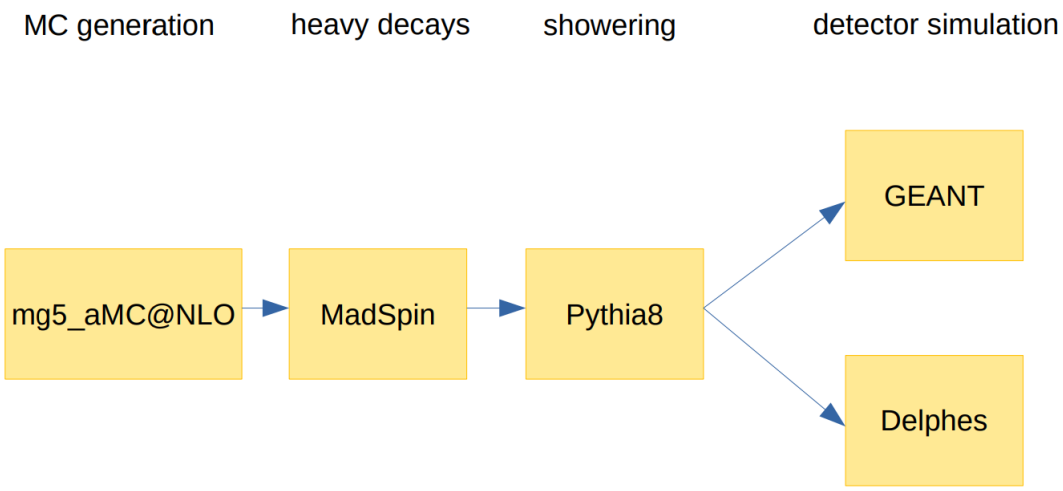

Figure 5.1: Illustration for the broad-scale data flow during the event generation process in this thesis: general steps at the top, specific software used below.

The first stage is the Monte-Carlo event generation stage. Here, parton-level events are generated using Monte-Carlo techniques. The result of this step are the kinematic parameters of the products, which, in the case of the ttbb process, are a top quark, a top antiquark, a bottom quark and a bottom antiquark. In this thesis, MadGraph5_aMC@NLO [16] (hereafter referred to as MadGraph) is used for this step (section 5.1).

The second stage deals with decaying heavy partons, that is, top quarks, and massive bosons. While technically optional, separating this step from the next allows for the retainment of much of the accuracy that would be lost when using methods of the subsequent step

for these decays. At the same time, separating it from the Monte-Carlo step allows for a significant reduction in required computation time. In this thesis, MadSpin [17] is used, which is a module built into MadGraph (section 5.2).

The third stage simulates the hadronization processes of the resulting partons ("parton shower"), simulating their decays down to hadrons and leptons which are stable on detector scale. In this thesis, PYTHIA 8 [18] is employed for this step (section 5.3).

The fourth and final stage, as well as the one which the primary focus of this thesis lies on, simulates the response of the detector to the many particles generated in the previous stages. Here, this thesis considers two different software tools: GEANT [19–21] and Delphes [22]. GEANT directly simulates the way each individual particle interacts with the detector material. While this is very accurate, it is also quite slow (section 5.4). Delphes, on the other hand, simulates the detector by using simple response functions which give a probability of detection by a tracker or an energy fraction deposited inside a calorimeter based only on a particle's basic kinematic parameters. While much faster, the accuracy of this approach naturally suffers (section 5.5). The main objective of this thesis is investigating how close to the results produced by GEANT one can get by only using Delphes on the sample case of the tt+X events detailed in chapter 4.

5.1 MadGraph

This section is based on [16], the standard reference for the current version of MadGraph.

MadGraph is the main software package used for the event generation in this thesis. It was designed specifically with user friendliness and automation in mind, such as to allow a user to be able to effectively generate events without having a particularly deep understanding of the underlying theory. Especially when the intention is to generate events within the boundaries of the standard model, just knowing the structure of the event and the basic MadGraph syntax is sufficient knowledge for running a rudimentary event generation. Correspondingly, only a surface-level overview of MadGraph's capabilities is given here.

The fundamental idea behind MadGraph is the use of meta-code which takes a physics model and a process to simulate and constructs code that, in turn, actually integrates the probabilities of the process happening in any given way and generates the events based on them. Thanks to the generality of this approach, this single software package is capable of simulating arbitrary processes in a wide variety of physics models at both leading-order (LO) and next-to-leading-order (NLO) of QCD (quantum chromodynamics) perturbation theory.

Given a Lagrangian, which is a formal description of particles and interactions in a physics model, MadGraph is capable of deriving Feynman rules for creating Feynman diagrams. From these, matrix elements which describe the transition probabilities between initial and final states of the system can be derived for computation using basic Monte-Carlo methods. It should be noted, however, that this alone is not sufficient for NLO computation due to ultra violet counter terms, which require further terms to be supplied separately. Fortunately, for the purposes of this thesis, everything needed is supplied with the standard installation of MadGraph, as it remains fully within the standard model.

This entire process is completely automated, with the user merely needing to supply the basic input parameters. Using the specific case of a ttbb process as an example, after initiating MadGraph, only four commands are necessary:

```
import model loop_sm-ckm
generate p p > t t~ b b~ [QCD]
output <output_name>
launch
```

The first line tells MadGraph which physics model to use, in this case, one of the NLO versions of the standard model supplied by the standard installation.

The second line tells MadGraph what specific process to simulate. The basic syntax consists of a list of input particles (in this case, two protons), followed by "`>`", followed by a list of output particles (here, top quark, top antiquark, bottom quark, bottom antiquark, in that order). The "`[QCD]`" option tells MadGraph to perform the simulation at NLO. There are various other possible syntactical measures to specify the process further, allowing, among other things, to specify final-state decays, exclude or force the inclusion of certain particles, or merge several processes into one simulation. Those will not be discussed here as they have seldomly or never been used during this specific work. A proper basic documentation for them can be found in [16].

The third line specifies the output directory for the simulation and initiates the actual generation of the meta-code for the event generation.

The final line initiates the integration and event generation.

While the minimum effort necessary to generate something useful for this work is, as shown, very minimal, MadGraph offers many options to adjust the process to one's needs, most of which will not be used or even mentioned here.

Upon initiating the "`launch`" command, one will be prompted to adjust several basic options, though the program will continue after a timeout regardless. Part of these options is the inclusion of MadSpin, Pythia or Herwig for showering, Delphes or PGS for detector simulation, and MadAnalysis for analyzing the final result. Thus, most of the generation process can be performed entirely inside MadGraph. Conversely, however, for example, if one is interested in the intermediate results, it is possible to separate all of these into separate steps, as well as to separate the cross-section integration from the event generation.

Next, the user will be prompted to adjust configuration text files called cards which hold parameters relevant for the generation. The two most important cards are the `run_card` and the `parameter_card`. The `run_card` contains information such as the number of events to generate, the desired precision of the integration, the parton distribution functions to use, as well as the options for customizing the basic program flow the user was prompted about before. The `parameter_card`, on the other hand, contains free parameters of the physics model, most notably particle masses and coupling strengths. Beyond these two, additional cards exist for MadSpin and showering options.

5.2 MadSpin

When considering the decay of heavy particles, specifically massive bosons and top-quarks, in the standard model, certain non-trivial effects (in particular, spin correlation) emerge which cannot be adequately simulated by regular showering algorithms. In principle, it is possible to let these decays be simulated accurately during the event generation step by

MadGraph, however, this would mean a high final-state multiplicity for the process, and thus unreasonably high computation times. MadSpin allows to alleviate this dilemma. It allows for a sufficiently good approximation in most cases, including those interesting for this work, while requiring only a fraction of the computation time that would be needed for an exact simulation. Once again, the theoretical details are of little importance for the end user, and will thus not be discussed here. They are detailed in [17].

In this thesis, MadSpin is used exclusively in conjunction with MadGraph, where all parameters relating to MadSpin can be set in the `madspin_card`. Primarily, these are simply the decays that are allowed for simulation, written in the very same syntax as the MadGraph process syntax.

5.3 Pythia

Pythia is a software package for generating high energy collision events. For a more detailed overview, refer to [18]. Historically, it began as an implementation of the Lund string model of hadronization. It has since expanded to the point of being capable of simulating the entirety of an event, including parton showering, initial and final state radiation, and even, albeit only at LO precision, the hard processes at the core of the event. It remains standard practice, however, to supply the result of the hard interaction from an external source, in the case of this work, MadGraph.

An event in Pythia is represented by the event record, which is a table of all particles involved in the event, including their basic kinematic properties along with information on the relations between particles. The process of filling this event record occurs in three stages: The process level, which simulates the hard scattering process, and is, in this work, taken over by MadGraph, the parton level, which simulates the decay of individual partons into a parton shower, and the hadron level, which simulates the effects of QCD confinement and reduces the parton shower to stable QCD-singlets (hadrons).

Once again, the specifics of any given simulation using the software can be adjusted using a large number of parameters, most of which are of little interest to this specific work. Depending on the use case, Pythia can be more of a software library than a single program. Given the specific setup used in this thesis, however, showering using Pythia can be included directly in MadGraph by turning on the option in the `run_card`, and some basic parameters can be adjusted in the `shower_card`.

5.4 GEANT

GEANT is a standard tool for detector simulation at CMS. Its fundamental idea is to directly simulate the interaction between particles and detector material. While this produces good results, it is quite slow.

The objective of this thesis is to approximate results produced by GEANT as closely as possible using Delphes. While pre-existing simulations using GEANT which are available in the CMS database are used for reference, GEANT is not actually used for event production in this work.

5.5 Delphes

The standard reference for Delphes 3, which is the current major version of Delphes used here, is [22].

Just like GEANT, Delphes is used for simulating the detector response to an event. In contrast to GEANT, it does so not by simulating, in great detail, how particles interact with the detector material, but instead, uses response functions which return the estimated detector response based on the kinematic parameters of a particle. This is inherently less precise, but it is much faster. Seeing how close one can feasibly get using Delphes to the "optimum" given by GEANT is the central question of this thesis.

Delphes has a modular structure. It consists of several modules (C++ programs that each perform a small part of the necessary calculations) which can be combined in a user-defined way using a configuration text file called a card. A card begins with a "set ExecutionPath" section which defines the order in which modules will be executed. This is followed by the definition of all the modules used. Such a definition consists of an indication of the type of module this is, out of a limited selection given by Delphes, a declaration of the module name used by the ExecutionPath, and the main body which defines all necessary parameters for the module.

Among these parameters are typically one or more input arrays, given as `source/name`, where `source` is the name of the module whose output this input is and `name` is the name given to that module's output. For many but not all modules, an output array name is also defined. The elements of these arrays used by Delphes are general "objects" which may or may not have certain properties such as basic kinematic parameters, a PDG code (a number identifying the type of particle, see [23]), and a number of possible tags. These "objects" can, in any specific case, represent individual particles, jets or calorimeter towers, but Delphes does not know of such distinction. Individual modules, however, may produce unexpected or meaningless results if the input objects do not match the module's expectation.

Aside from these arrays, the parameters of a module are actual parameters which customize the operation performed by the module. Typically, these can be integers, floats, booleans, or mathematical expressions using basic kinematic parameters of the input. The specific parameters used depend on the specific module type and are mostly self-explanatory.

Some of the module types most relevant to this work are as follows:

`Efficiency`: Takes an input array of arbitrary objects and an efficiency formula which gives a probability depending on kinematic parameters. Produces an output array which contains each entry of the input with the probability given by the efficiency formula.

`SimpleCalorimeter`: Takes an input particle array and an input particle track array and produces the response of calorimeter towers. Customizable in terms of ϕ-η-bins, expected energy fraction detected by particle and precision of the measurement.

`Isolation`: Takes an array of "candidates" and an array of "noise". Selects those candidates for which the total transverse momentum of the noise in a cone around it is within a given range of ratios compared to its own transverse momentum. This is used to determine which leptons or photons to consider isolated, as opposed to being part of a jet. In such a case, the candidates would be the final-state leptons or photons, and the noise would be the non-leptonic energy flow measured by the HCAL and ECAL.

`FastJetFinder`: Takes an input array of particles or calorimeter towers and finds jets among them. Can use several different jet finding algorithms, but this thesis is exclusively using anti-k_t [13]. Can customize input parameters necessary for the algorithm, as well as some basic cuts.

`JetFlavorAssociation`: Given an array of jets, an array of all particles and an array of unstable partons, matches parton flavor to jets based on their kinematic properties.

`BTagging`: Adds b-tags to input array based on flavor-dependent efficiency formula given.

Although it is theoretically possible to combine these and other modules in any conceivable way, the general structure for a real detector simulation is relatively rigid. Figure 5.2 illustrates a configuration akin to the ones used throughout this work, although with some simplifications.

First, the `ParticlePropagator` module propagates generator level particles through the detector's magnetic field and separates them by type. Next, for electrons, muons and charged hadrons a tracker response is simulated using `Efficiency` modules and a statistical inaccuracy is added using `MomentumSmearing` modules. After this, the two types of calorimeters are simulated. Together with several `PdgCodeFilter` modules, this produces several different types of energy flow arrays. From these, electrons and photons are reconstructed using `Efficiency` modules, whereas for muons, the tracker output is used directly. From there, `Isolation` modules select those particles that are sufficiently isolated from the remaining energy flow. In parallel to that, jets are reconstructed using `FastJetFinder`, their energy is adjusted using `EnergyScale`, and b-tagging is performed. The latter is done by first matching the generated jets to generator level unstable partons using a `JetFlavorAssociation` module before simulating the inaccuracy of real b-tagging procedures using a `BTagging` module. Finally, the `UniqueObjectFinder` module ensures that none of the high-level objects are counted twice. The `TreeWriter` module (not shown) can then write any of the arrays used in the procedure to a ROOT [24] file.

Note that this example omits several parts of the procedure that are commonly included, such as generator level jets or pile-up. Conversely, some parts shown can be omitted in a real simulation depending on the use case.

To use Delphes, while it is possible to use it directly from inside MadGraph, using a standalone version is preferred in this thesis for reasons discussed in the next section. Delphes has several executables to deal with different input file formats, the one used here is DelphesHepMC which takes HepMC [25] files as input. The command to run Delphes is as follows:

```
./DelphesHepMC delphes_card.tcl output.root input.hepmc
```

Here, `output.root` is the name of the output ROOT file Delphes will generate, `input.hepmc` is the name of the HepMC input file, and `delphes_card.tcl` is the text file that is the delphes card which customizes the detector simulation.

5.6 Gridpacks

While it would be possible to generate the neccessary tt+X events starting from only the programs described, potentially even without ever leaving MadGraph, the number of adjustable parameters is very large. This means that extreme care would need to be taken to ensure all running parameters are equivalent to those used by the existing GEANT-using simulated events in the CMS database which the events generated here are to be compared to.

A far easier and more reliable method to achieve the same is using CMS gridpacks. In the CMS database, every simulated event file is linked to the procedure used to generate it. Generally, this means a single so-called fragment file and a single console command. By adjusting the console command to stop short of performing the detector simulation, one is able to produce generator level events with the exact same running parameters as

the original event file. The fragment file, in the cases that are of interest to this work, is a Python file containing a reference to a "gridpack" which is responsible for running the MadGraph and MadSpin parts of the simulation, as well as a set of parameters for Pythia. The gridpack is simply an archive containing a full installation of MadGraph, with the preparation needed before generating events, that is, the generation of Feynman diagrams and the integration part, already done. By unpacking it and executing a shell script also included in the archive, events with reliably equivalent generation parameters can be generated.

The result of this process is a ROOT file in a format specific to the standard CMS software workflow, which is actually just a wrapper around the HepMC file produced by Pythia. As Delphes does not understand this format, a small separate program is needed to extract the original HepMC file so it can be given to Delphes to perform the custom detector simulation.

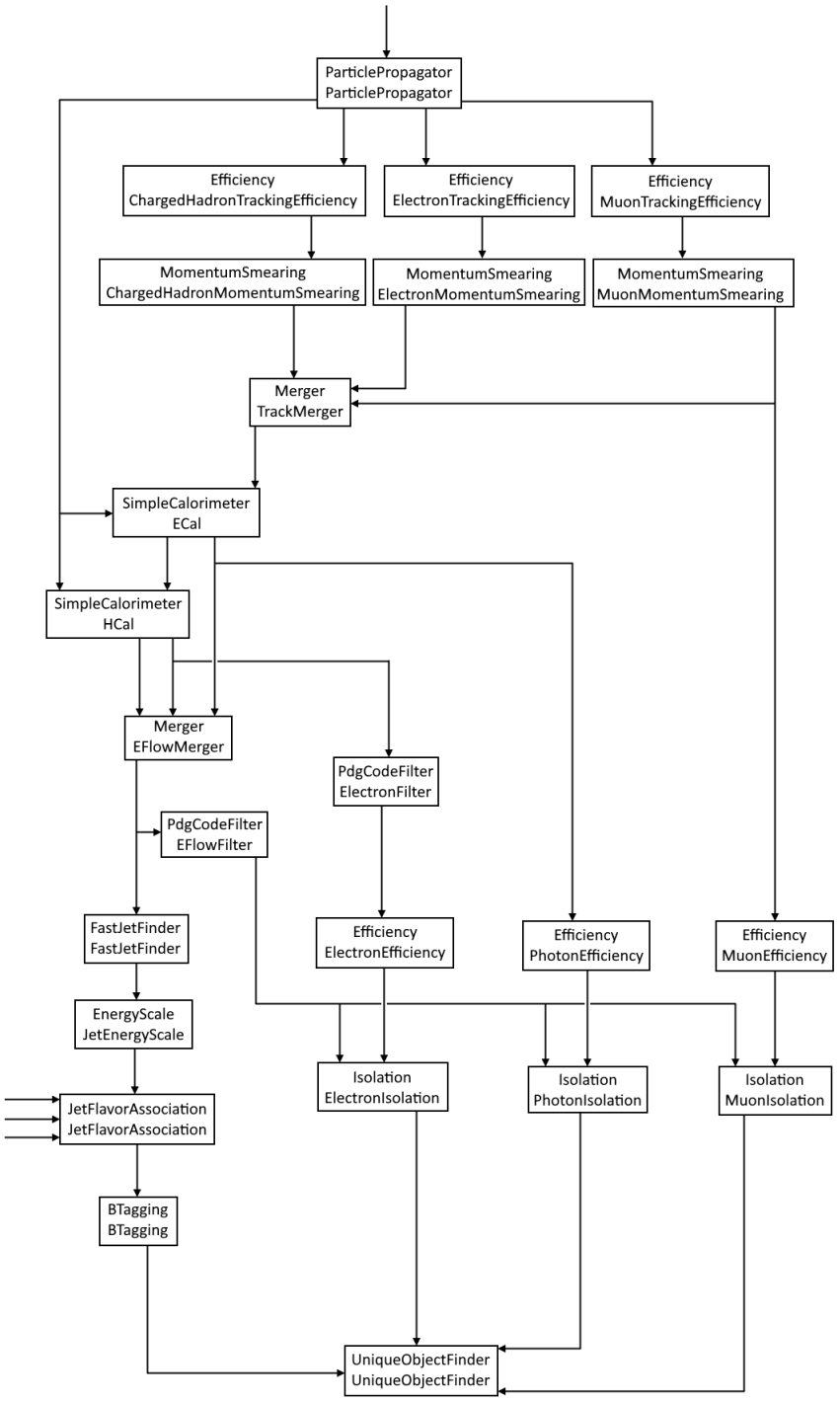

Figure 5.2: Illustration for a possible configuration in Delphes. Boxes represent modules, with their type as the first line and their name as the second. Arrows represent object arrays processed by the modules.

6 Simulation Parameters

6.1 Simulation Files

The GEANT-using event files which are used for the three processes considered in this thesis can be found under the following names in the CMS database:

643000 events from
`"/RunIISummer20UL17NanoAODv9/`
`TTbb_4f_TTToSemiLeptonic_TuneCP5-Powheg-Openloops-Pythia8/`
`NANOAODSIM/106X_mc2017_realistic_v9-v1/"`
for ttbb,

628464 events from
`"/RunIISummer20UL18NanoAODv9/`
`ttHJetTobb_M125_TuneCP5_13TeV_amcatnloFXFX_madspin_pythia8/`
`NANOAODSIM/106X_upgrade2018_realistic_v16_L1v1-v1/"`
for ttH, and

512000 events from
`"/RunIISummer20UL18NanoAODv9/`
`TTZToBB_TuneCP5_13TeV-amcatnlo-pythia8/NANOAODSIM/`
`106X_upgrade2018_realistic_v16_L1v1-v2/"`
for ttZ.

The Delphes events are generated by using the corresponding gridpacks for generator level event generation and applying Delphes using several Delphes cards, the specifics of which are detailed in the results section. The Delphes installation used is obtained by auto-installing Delphes from within a standalone MadGraph version 2.9.9 installation. The total number of events generated is 4412073 for ttbb, 2697764 for ttH and 5000000 for ttZ.

6.2 Analysis Methods

Both the pre-existing GEANT events and the Delphes results are ROOT files, however, the specific format differs, mainly by field names. To compare these, a set of three self-written programs is used.

The first of these is a Python program which extracts the relevant data from the ROOT files using the uproot Python package and stores them in a consistent ad-hoc format. The

second is a C++ program that reads the output of the first and performs calculations to extract the desired kinematic observables from the raw data. It is written in such a way as to allow for flexibility in terms of the desired output parameters. Its output is a file containing raw 32-bit floats which is used by the third program to generate histograms using the matplotlib-pyplot Python library.

The specific kinematic observables used in this work are listed in table 6.1. It is based, in large part, but not entirely, on the list used in [15]. It should further be noted that all of these parameters can also be evaluated at generator level in addition to the detector level, some of which will be used to gain further insight into what is happening at detector level.

6.3 Kinematic cuts

In practice, it is often useful to not consider certain high-level objects, particularly those with low transverse momentum or very high pseudorapidity, as the information on those is often unreliable and not very interesting.

In this work, all such selection criteria based on the object kinematics ("kinematic cuts") are dominated by those applied at the analysis stage, thus guaranteeing the equivalence in terms of cuts between the Delphes and GEANT simulations. Unless stated otherwise, these are $p_{\mathrm{T}} > 30\,\mathrm{GeV}$ and $|\eta| < 2.5$ for all high-level objects.

Table 6.1: Kinematic observables.

parameter	description
N_{Jet}	Jet multiplicity in an event.
$p_{\text{T,Jet}}$	Transverse momentum of jets (equation (3.1)).
η_{Jet}	Pseudorapidity of jets (equation (3.2)).
ϕ_{Jet}	Azimuthal angle of jets.
m_{Jet}	Invariant mass of jets (equation (3.3)).
E_{Jet}	Jet energy.
N_{BJet}	B-jet multiplicity in an event.
$p_{\text{T,BJet}}$	Transverse momentum of b-jets (equation (3.1)).
η_{BJet}	Pseudorapidity of b-jets (equation (3.2)).
ϕ_{BJet}	Azimuthal angle of b-jets.
m_{BJet}	Invariant mass of b-jets (equation (3.3)).
E_{BJet}	B-jet energy.
N_e	Electron multiplicity in an event.
$p_{\text{T},e}$	Transverse momentum of electrons (equation (3.1)).
η_e	Pseudorapidity of electrons (equation (3.2)).
ϕ_e	Azimuthal angle of electrons.
N_μ	Muon multiplicity in an event.
$p_{\text{T},\mu}$	Transverse momentum of muons (equation (3.1)).
η_μ	Pseudorapidity of muons (equation (3.2)).
ϕ_μ	Azimuthal angle of muons.
$H_{T,\text{all}}$	Scalar sum of all jet and lepton transverse momenta in an event.
$H_{T,\text{Jet}}$	Scalar sum of all jet transverse momenta in an event.
$H_{T,\text{BJet}}$	Scalar sum of all b-jet transverse momenta in an event.
$\Delta R(bb)_{\text{average}}$	Average distance between any two b-jets in the η-ϕ-plane in an event.
$\Delta R(bb)_{\text{closest}}$	Distance in the η-ϕ-plane of the two closest b-jets in an event.
$\Delta R(bb)_{\text{leading}}$	Distance in the η-ϕ-plane of the two leading b-jets in an event.
$m(bb)_{\text{closest}}$	Invariant mass of the two closest b-jets in an event.
$m(bb)_{\text{leading}}$	Invariant mass of the two leading b-jets in an event.
$p_{\text{T}}(bb)_{\text{closest}}$	Total transverse momentum of the two closest b-jets in an event.
$p_{\text{T}}(bb)_{\text{leading}}$	Total transverse momentum of the two leading b-jets in an event.
$p_{\text{T},1^{\text{st}}\text{BJet}}$	Transverse momentum of the leading b-jet in an event.
$p_{\text{T},2^{\text{nd}}\text{BJet}}$	Transverse momentum of the second b-jet in an event.
$p_{\text{T},3^{\text{rd}}\text{BJet}}$	Transverse momentum of the third b-jet in an event.
$p_{\text{T},4^{\text{th}}\text{BJet}}$	Transverse momentum of the fourth b-jet in an event.

7 Results

After performing a generator level comparison between the results produced by Delphes and those produced by GEANT in section 7.1, the detector level study starts with observing the ttbb simulation results produced by the default Delphes CMS card supplied with the Delphes installation and identifying flaws in section 7.2. After this, potential causes and remedies for these inaccuracies are investigated in sections 7.3-7.8. Finally, after adjusting the Delphes card accordingly, it is applied to both ttH and ttZ samples to infer how well the conclusions made on the basis of ttbb transfer to other processes in section 7.9.

7.1 Generator level comparison

As both the GEANT and Delphes samples use the same procedure to produce their generator level events, it is to be expected that both final samples contain statistically equivalent information. To verify this, basic kinematic parameters for jets, b-jets, electrons and muons on generator level are plotted and compared.

For selecting b-jets on generator level, the flavor association field of generator level jets is used directly. It is worth noting that the default Delphes CMS card does not perform flavor association on generator level, requiring a corresponding adjustment of adding a second `JetFlavorAssociation` module to the workflow to be made.

Theoretically, the generator level information should be equivalent between Delphes and GEANT save for statistical variations, because the generator level simulation is exactly identical. The aggregation of generator level particles into generator level jets, however, is performed during the detector simulation. Thus, the only observables which may potentially be different between the two samples are those which are connected to jets. The jet finding algorithm, however, is also the same (anti-k_{T}), meaning that that the jet observables, too, should be equivalent. In fact, reality matches these expectations (figures B.35-B.54). Unfortunately, however, significant deviations are observed when considering the b-jet multiplicity (figure 7.1).

Delphes produces significantly more events with very high b-jet multiplicities. As the parameters for all jets show no such deviation, this implies that the jet flavor association algorithm subtly differs between Delphes and GEANT. This is unfortunate, especially because events with high b-jet multiplicity are highly interesting for analyses such as [15]. Unfortunately, the flavor association module in Delphes lacks any parameters to adjust in order to attempt to remedy this issue.

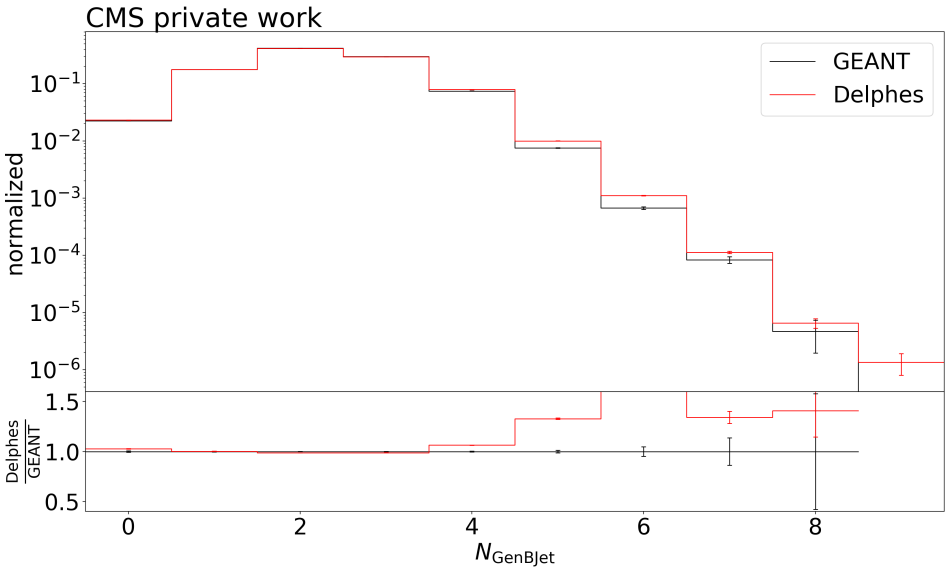

Figure 7.1: Histogram of the multiplicity of b-jets on generator level for ttbb events, normalized to an integral of 1.

7.2 Initial results

The natural starting point for this analysis is the result produced by the default Delphes CMS card. While this card is indeed the default, it makes no claim to being particularly accurate. In fact, deviations from the "target" result produced by GEANT can be observed within all but a few of the kinematic observables considered. The only parameters which immediately yield near-flawless results are η_μ and ϕ_μ (figures 7.2 and 7.3).

All other parameters show statistically significant deviations from the target. These will each be discussed in detail separately in sections 7.3-7.8. Nevertheless, most qualitative properties remain intact. For example, minor, wide peaks in the jet mass distributions at vector boson and top quark masses (figure 7.4), or the distinctive shape of the $\Delta R(bb)$ distributions are clearly distinguishable.

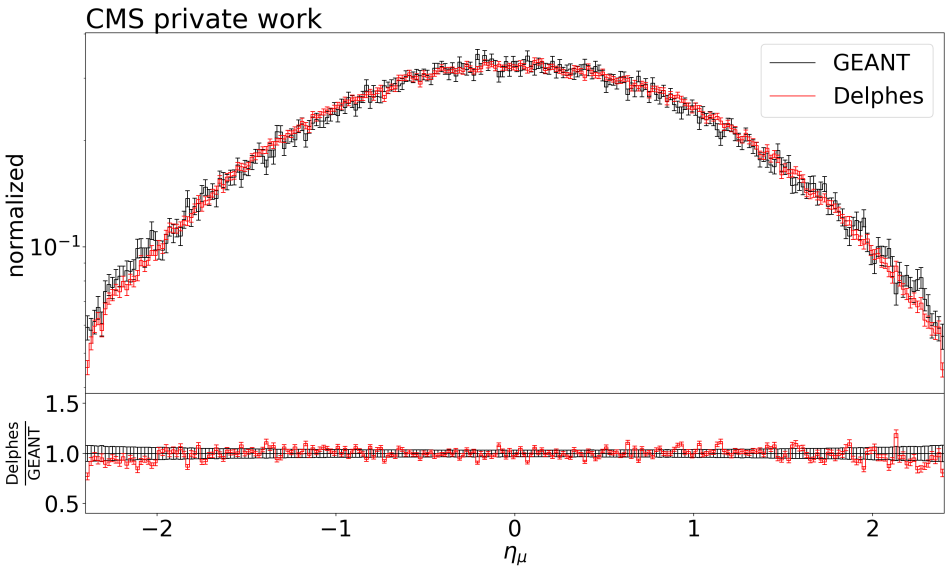

Figure 7.2: Histogram of the muon pseudorapidity distribution for ttbb events, using the default Delphes card, normalized to an integral of 1.

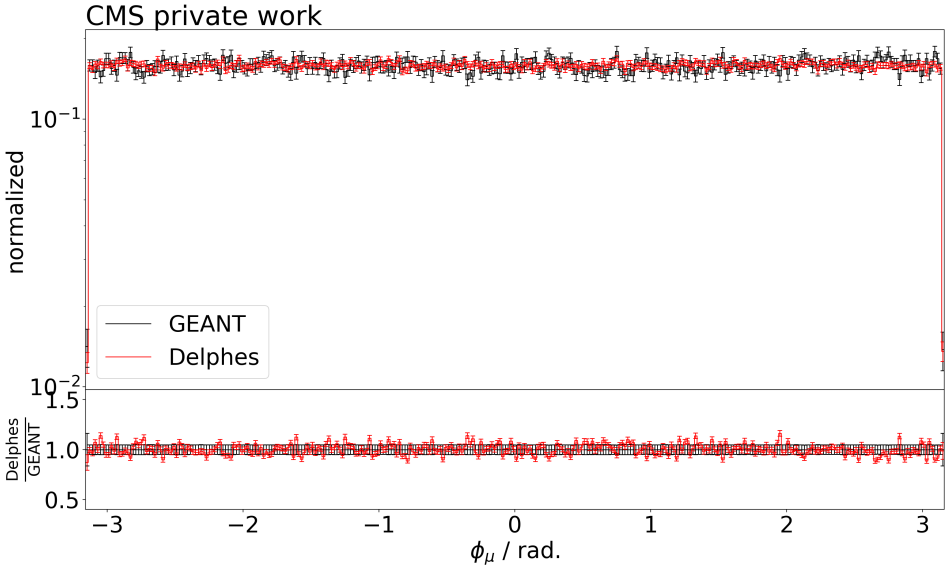

Figure 7.3: Histogram of the muon azimuthal angle distribution for ttbb events, using the default Delphes card, normalized to an integral of 1.

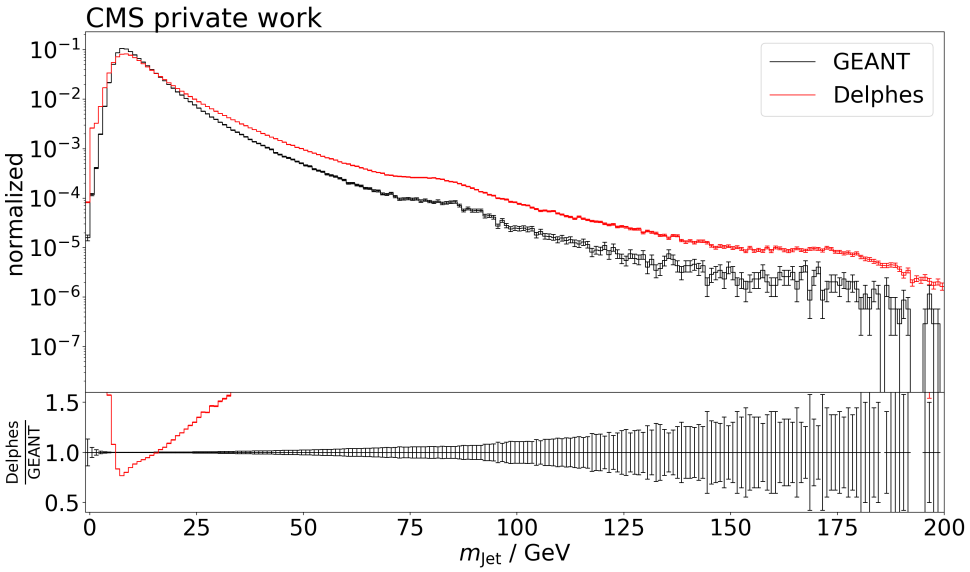

Figure 7.4: Histogram of the invariant mass of jets distribution for ttbb events, using the default Delphes card, normalized to an integral of 1.

7.3 Lepton isolation

One of the most noticeable deviations occurs for electrons and muons at low p_T. Seemingly, Delphes significantly underestimates the number of leptons with low transverse momenta (figures 7.5 and 7.6).

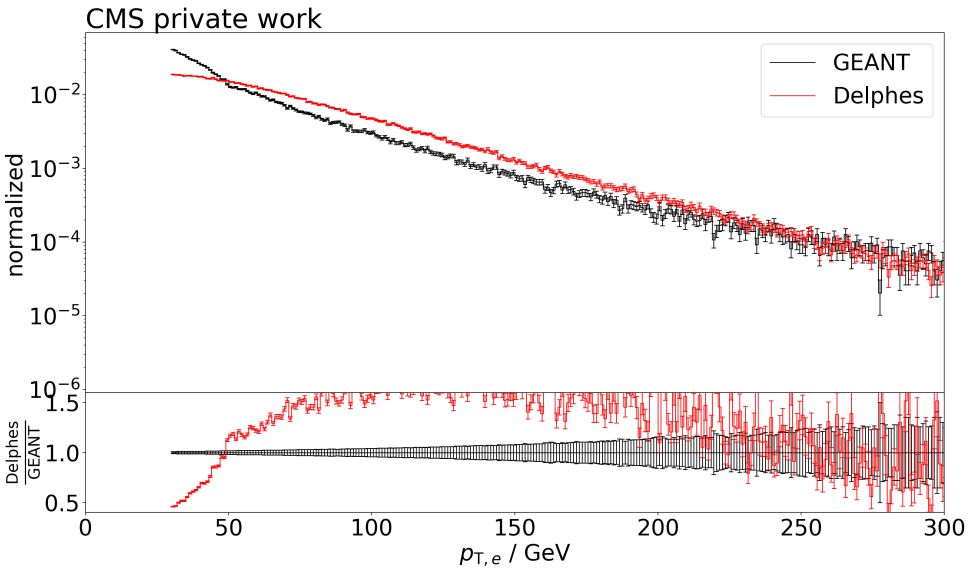

Figure 7.5: Histogram of the electron transverse momentum distribution for ttbb events, using the default Delphes card and without accounting for isolation, normalized to an integral of 1.

An investigation has shown, however, that this problem lies not with Delphes or the Delphes card, but rather, with the analysis: The output which Delphes generates for leptons only contains isolated electrons and muons, whereas the GEANT sample contains all leptons. When considering leptons as high level objects, it is useful to separate those that occur as part of a hadronic showering process, and thus become part of a jet, from those that occur as direct products of the hard process or from early decays of unstable partons, and are thus "isolated" from jets. Delphes does this by using a dedicated module to filter the array of detected leptons. GEANT calculates so-called isolation variables for each lepton instead, and leaves the filtering to subsequent analysis programs. Thus, the Delphes output, unlike GEANT, does not contain non-isolated leptons.

Simply turning off the corresponding "`Isolation`" modules in Delphes almost completely remediates this issue. However, this produces some problems for the jet-finding procedure in Delphes. While it is not difficult to adjust the Delphes card in such a way as to fix both issues, the choice taken here is slightly different: Instead of adjusting the Delphes card, the analysis procedure is adjusted to incorporate lepton isolation variables and use them to discard non-isolated leptons from the GEANT samples. This makes sense insofar as that the isolated leptons are the only ones most further analyses would be interested in.

Of note here are the specific isolation criteria applied, as they are different from those in the default Delphes card: A relative isolation parameter of $I_{\text{rel}} = 0.12$ within a $\Delta R = 0.3$ cone for electrons (as well as photons, although photons are not explicitly investigated in this work), and a relative isolation parameter of $I_{\text{rel}} = 0.25$ within a $\Delta R = 0.4$ cone for muons. This means that the total p_T of all non-lepton particles within that cone around an isolated lepton should not exceed the product of I_{rel} with the p_T of the isolated lepton.

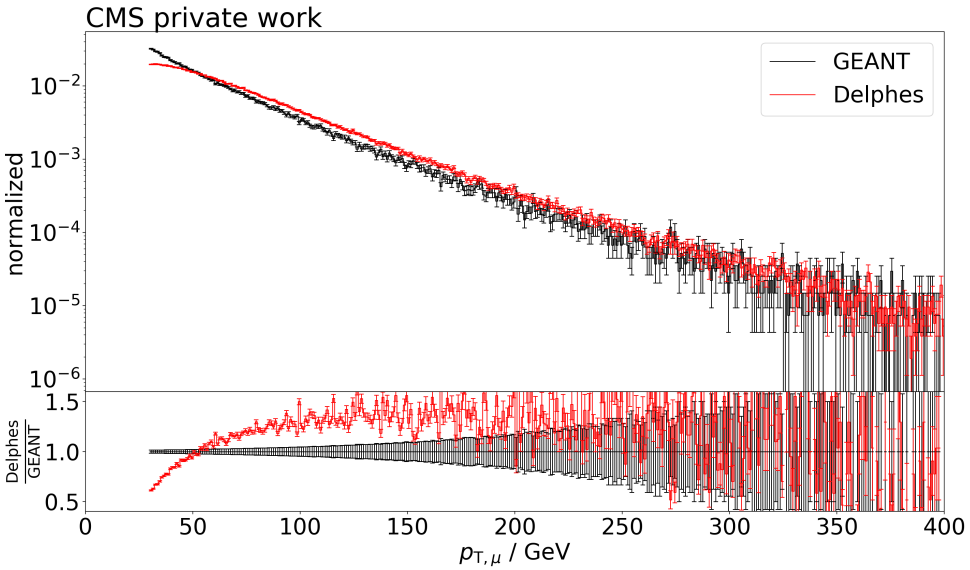

Figure 7.6: Histogram of the muon transverse momentum distribution for ttbb events, using the default Delphes card and without accounting for isolation, normalized to an integral of 1.

These criteria are applied consistently to both GEANT and Delphes simulations, however, they are not entirely equivalent to the standard criteria used at CMS due to a limitation by Delphes which requires the isolation variable to be a constant. This may have a small effect on the jet reconstruction procedure, although, if it exists, it appears to be negligible.

The results, shown in figures 7.7 and 7.8, are satisfactory.

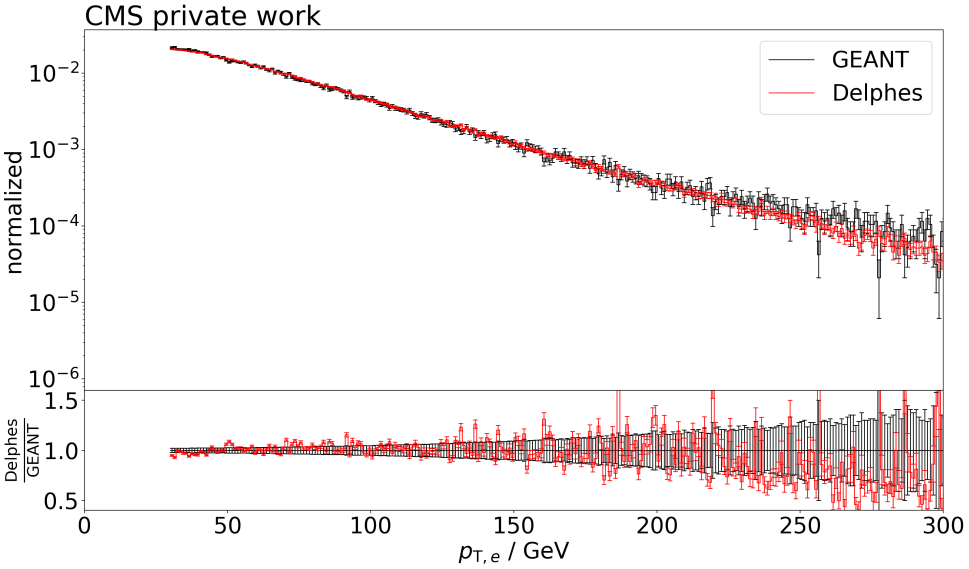

Figure 7.7: Histogram of the electron transverse momentum distribution for ttbb events, using the final Delphes card and accounting for isolation, normalized to an integral of 1.

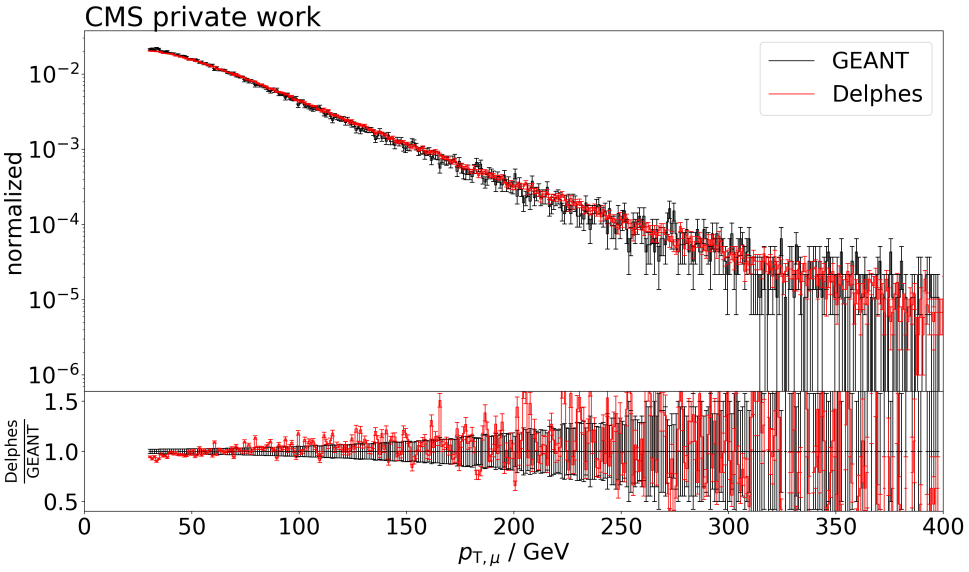

Figure 7.8: Histogram of the muon transverse momentum distribution for ttbb events, using the final Delphes card and accounting for isolation, normalized to an integral of 1.

7.4 Lepton efficiencies

While the p_T-distribution for leptons is satisfactory after the changes described in section 7.3, this is not the case for η_e (figure 7.9) or the multiplicities for both flavors of leptons (figures 7.10 and 7.11).

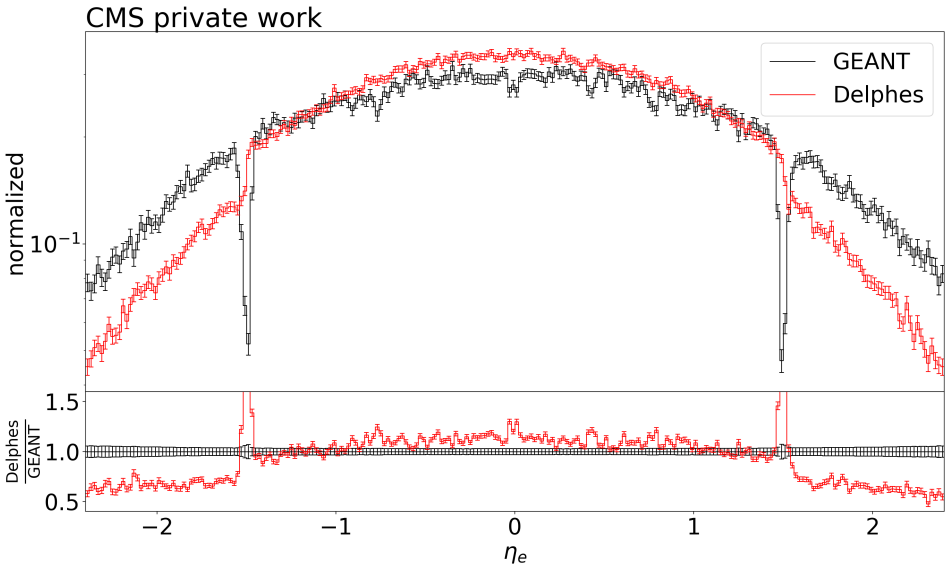

Figure 7.9: Histogram of the electron pseudorapidity distribution for ttbb events, using the default Delphes card, normalized to an integral of 1.

In the case of η_e, the gap between the detector barrel and end-cap at approximately $\eta = 1.5$ is not properly simulated. Additionally, the overall detection efficiency appears to be underestimated in the end-cap region. Clearly, this is an issue with the electron efficiency formula used by the default Delphes CMS card.

Using an alternative efficiency formula found in [26] (see "`ElectronEfficiency`" in section A in the Appendix) instead significantly improves the result (figure 7.12). The p_T-distribution appears largely unaffected, and thus not worsened.

The gap is now properly accounted for, although the number of detected electrons is still overestimated inside the gap. The end-cap regions are also improved, though the deviations remain clear. This result could likely be further improved by conducting a separate investigation into the true efficiency formulas of the CMS detector as predicted by the GEANT simulation, but this lies beyond the scope of this work.

The other problem related to lepton efficiencies is the disparity between lepton multiplicities in the GEANT and Delphes samples. The observation that Delphes significantly underestimates the number of leptons detected has been made before [27].

This appears to be a problem inherent to Delphes itself. Even when the relevant efficiencies are set to the perfect value of 1.0 which should be impossible in a real detector, Delphes produces too few electrons and muons. Turning off lepton isolation in addition to that reveals that GEANT produces more leptons on detector level than there are on generator level (figure 7.13). There appear to be two possible sources for these leptons: Fake leptons, that is, non-lepton particles mistakenly identified as leptons, and secondary particles, that is, leptons originating from interactions between generator level particles and the detector materials. Neither effect can be simulated by Delphes, and may thus be responsible for the

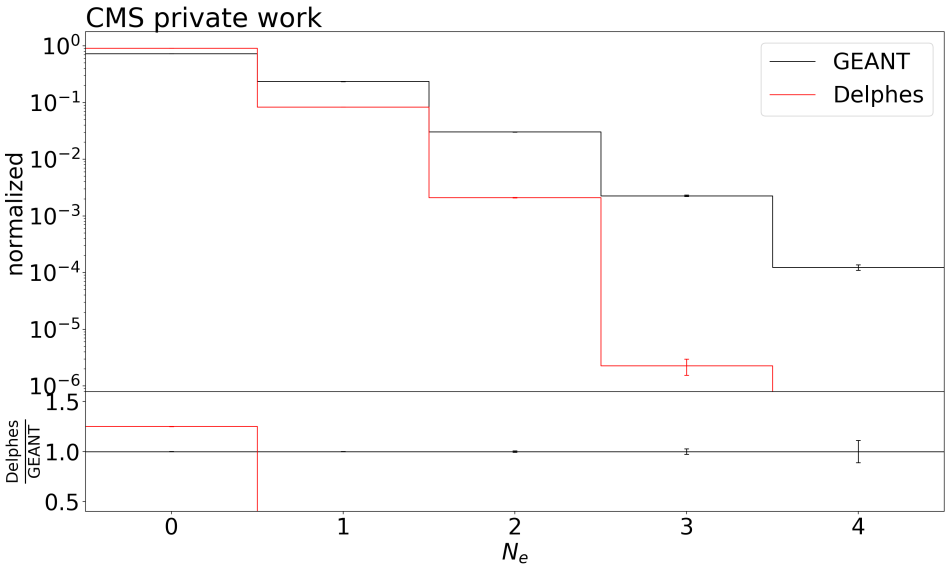

Figure 7.10: Histogram of the electron multiplicity for ttbb events, using the default Delphes card, normalized to an integral of 1.

observed discrepancy. While another possible source for leptons beyond those present on generator level is pile-up, this is unlikely to be the main issue here due to the findings of section 7.7.

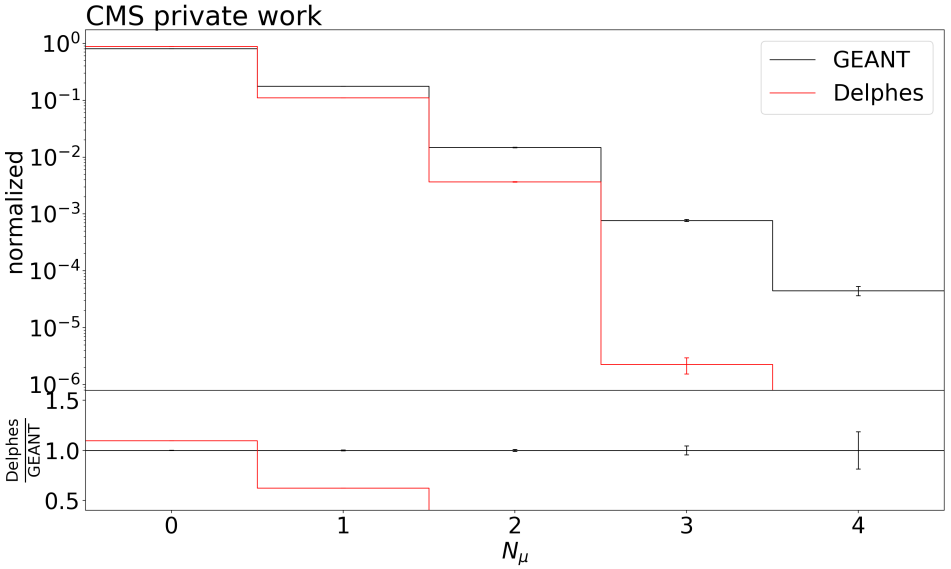

Figure 7.11: Histogram of the muon multiplicity for ttbb events, using the default Delphes card, normalized to an integral of 1.

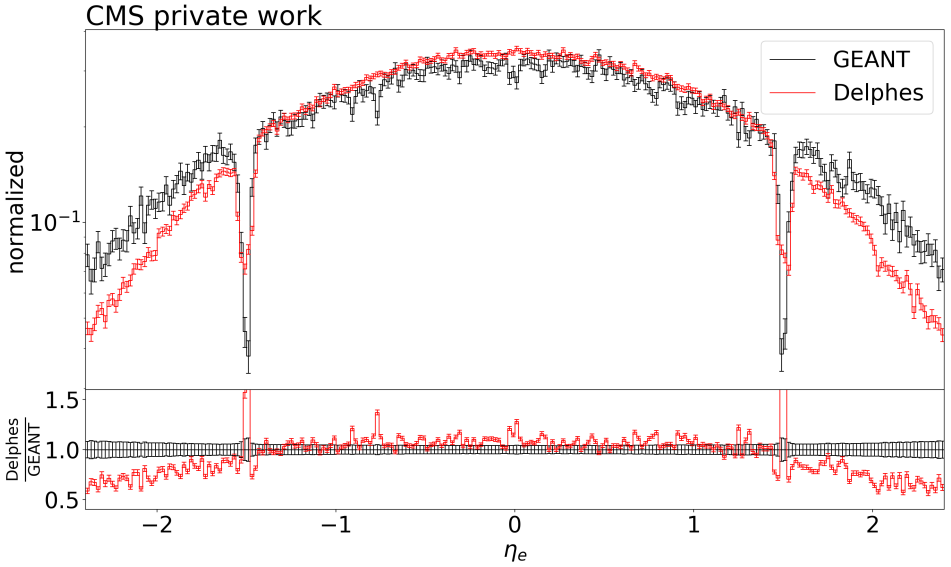

Figure 7.12: Histogram of the electron pseudorapidity distribution for ttbb events, using the final Delphes card, normalized to an integral of 1.

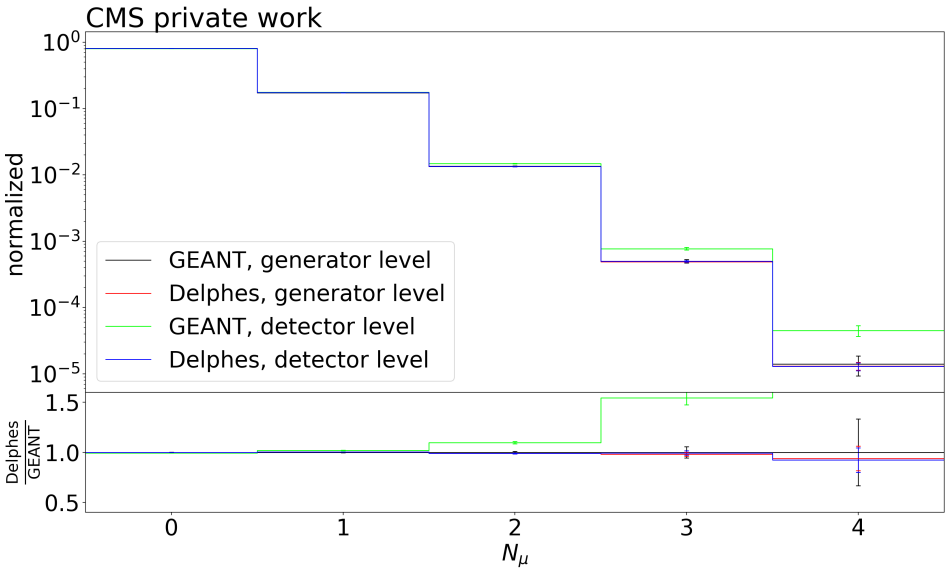

Figure 7.13: Histogram of the muon multiplicity for ttbb events, using the Delphes card with perfect muon efficiency and tracker efficiency, and no muon isolation, normalized to an integral of 1.

7.5 B-tagging efficiencies

The $p_{\mathrm{T,BJet}}$ distribution shows a significant deviation from the target (figure 7.15). Crucially, this deviation is significantly larger than the deviation of $p_{\mathrm{T,Jet}}$ (figure 7.14). This suggests that the simulation of b-tagging in Delphes is inaccurate.

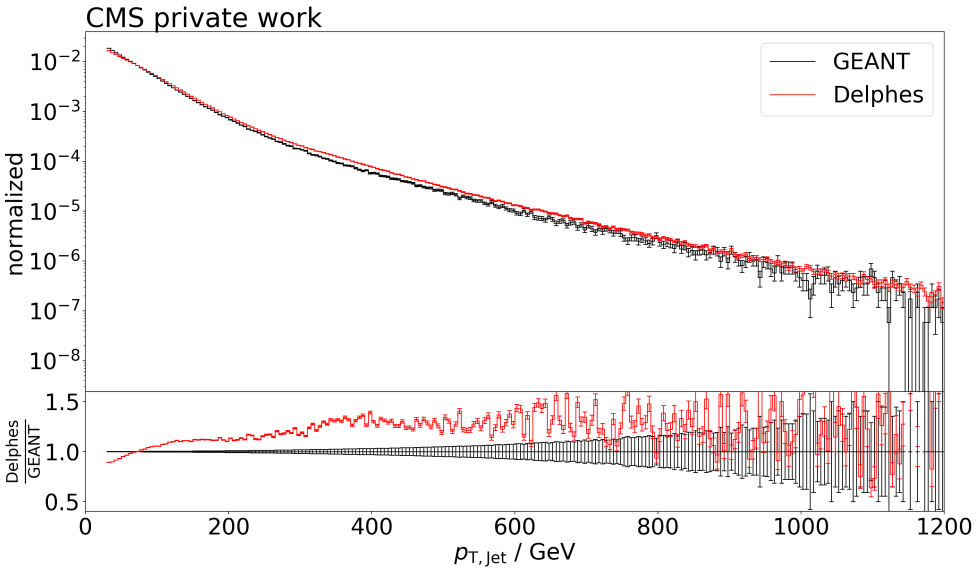

Figure 7.14: Histogram of the jet transverse momentum distribution for ttbb events, using the default Delphes card, normalized to an integral of 1.

The b-tagging procedure in Delphes functions on a simple basis: A jet is associated with an unstable parton, and the b-tag rate is given as a mapping of parton flavor to tag or mistag rate. In the default card, this includes the b-tagging efficiency for bottom-flavored jets, the mistag rate for charm-flavored jets, and the default mistag rate for all other flavors.

These formulas can easily be adjusted to reflect reality better than they do by default. In this work, this has been done based on [28] (see "`BTagging`" in section A in the Appendix). The binwise tag rates from figure 4 of [28] are used as the bottom-flavor efficiency, and a flat mistag rate of 0.18 is chosen for charm-flavored jets based on figures 2 and 3 of [28]. The result is shown in figure 7.16.

Overall, an improvement has been achieved. However, the discontinuity of the binwise efficiency function shows, especially at $p_{\mathrm{T,BJet}} = 75\,\mathrm{GeV}$. Further research is necessary to optimize the function further.

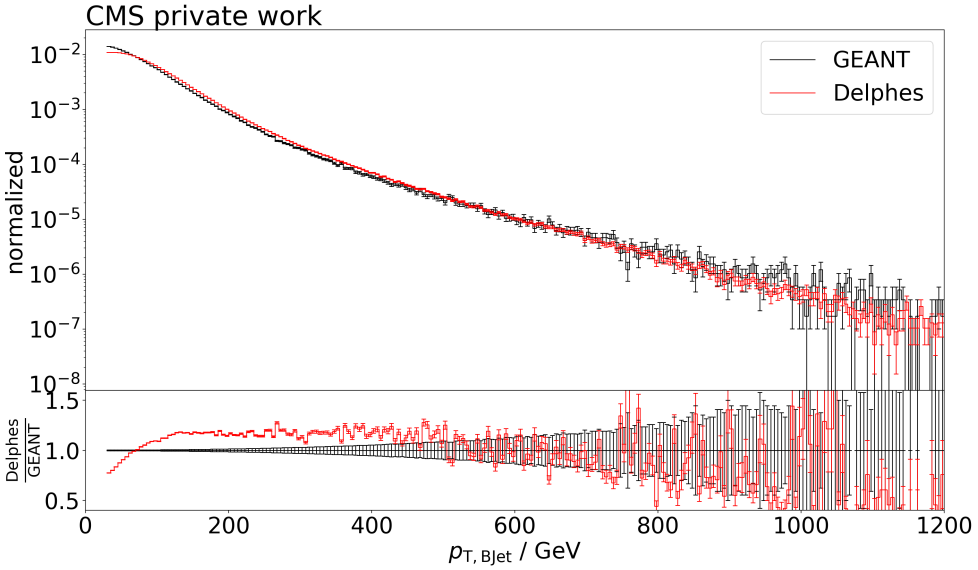

Figure 7.15: Histogram of the b-jet transverse momentum distribution for ttbb events, using the default Delphes card, normalized to an integral of 1.

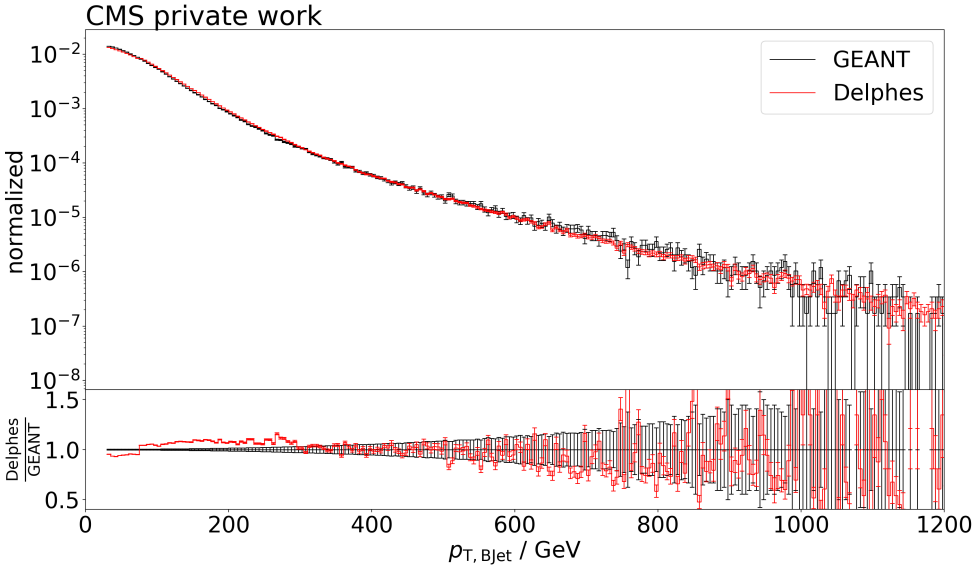

Figure 7.16: Histogram of the b-jet transverse momentum distribution for ttbb events, using the final Delphes card, normalized to an integral of 1.

7.6 Calorimeter binning

Looking at the distributions of η_{Jet} (figure 7.17) and ϕ_{Jet} (figure 7.18), as well as, to a lesser extent, those of η_{BJet} and ϕ_{BJet}, a periodicity can be observed in the Delphes sample. While its amplitude is small, this is clearly unnatural, especially for the ϕ distributions.

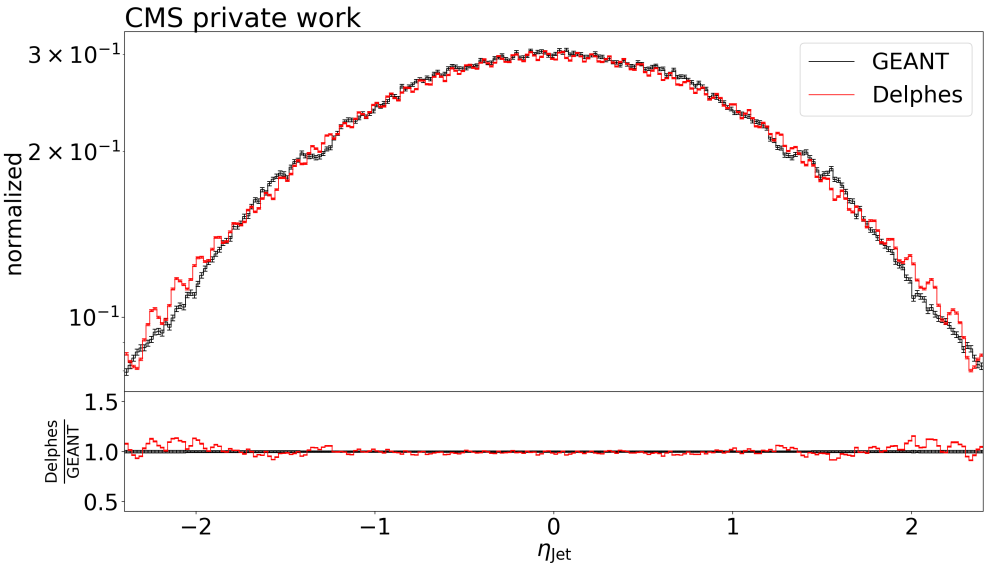

Figure 7.17: Histogram of the jet pseudorapidity distribution for ttbb events, using the default Delphes card, normalized to an integral of 1.

Investigating this periodicity closer shows that it aligns with the HCAL tiling used by the Delphes card. Evidently, the aggregation of particle momentum and energy into calorimeter towers is imperfect in Delphes. While the "`SimpleCalorimeter`" module has an option to smear the location of tower centers, it is on in the default Delphes card and insufficient for fully mitigating the effect. Turning it off, as would be expected, makes the periodicity more pronounced.

By reducing the size of the η-ϕ-bins used by Delphes, this problem can be fully mitigated (figures 7.19 and 7.20). Naturally, however, this makes the tower-based information produced by Delphes completely incompatible with real data. Thus, whether such adjustment is worth it in any given case depends on the situation. The plots in the appendix do use this adjustment as this work only concerns itself with high-level objects.

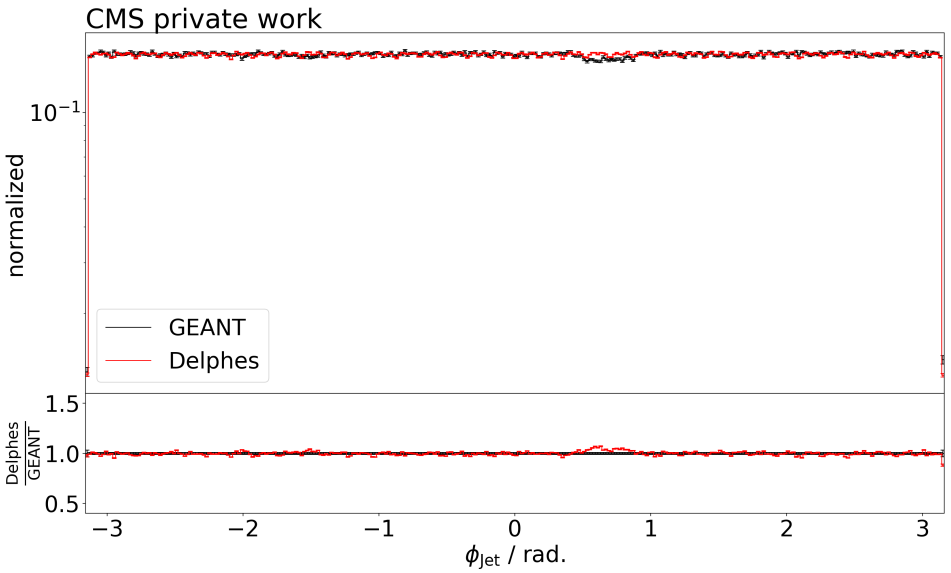

Figure 7.18: Histogram of the jet azimuthal angle distribution for ttbb events, using the default Delphes card, normalized to an integral of 1.

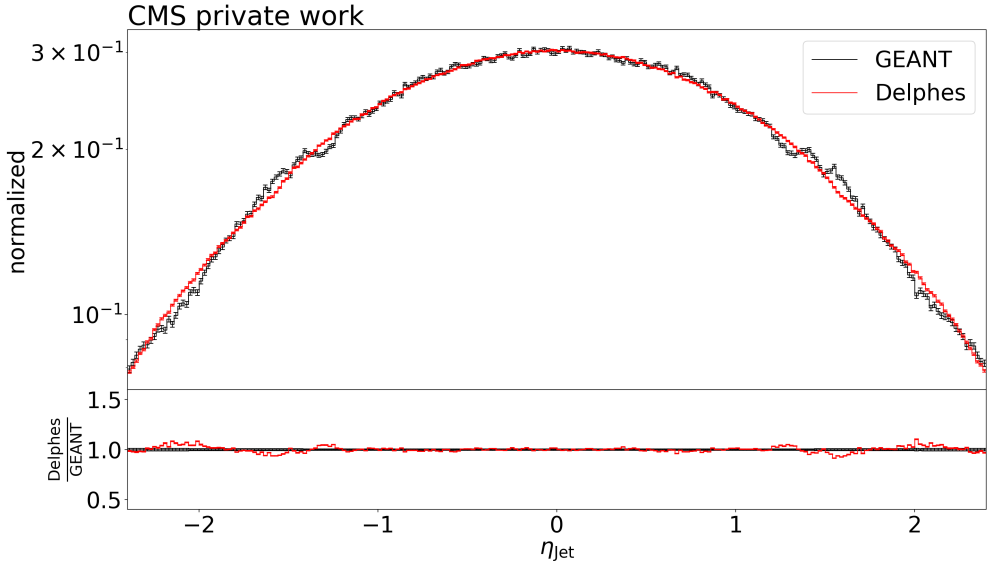

Figure 7.19: Histogram of the jet pseudorapidity distribution for ttbb events, using the final Delphes card, normalized to an integral of 1.

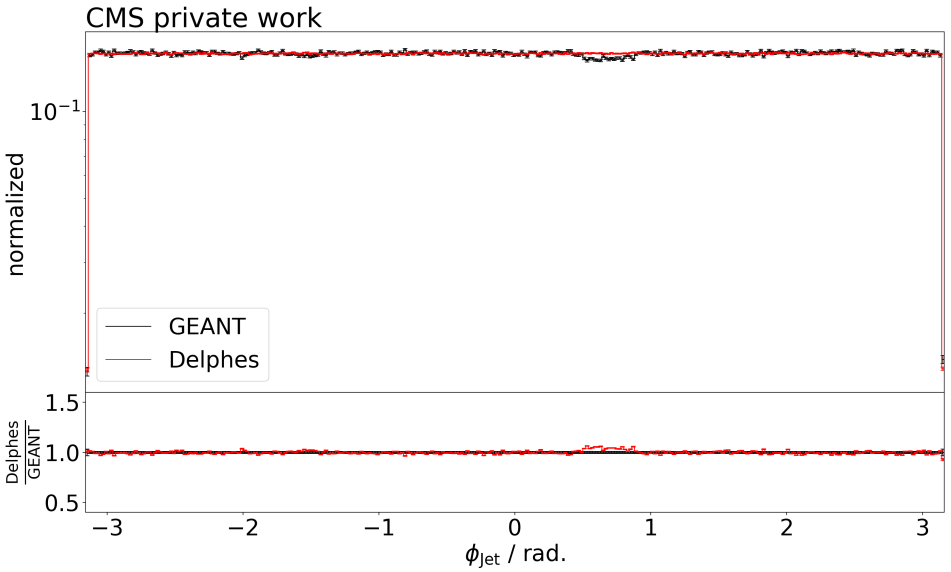

Figure 7.20: Histogram of the jet azimuthal angle distribution for ttbb events, using the final Delphes card, normalized to an integral of 1.

7.7 Pile-up

Pile-up is the process of multiple hadron collisions happening simultaneously in the collider, with every collision other than the highest-energy one being treated as background noise. While there are ways to significantly reduce the effect this has on the final data algorithmically, a certain distortion remains. This process is simulated by GEANT, but not by the default Delphes card. The possibility of including pile-up is offered by Delphes, and the default installation comes with a card for CMS which includes a simulation of pile-up.

The pile-up simulation in Delphes consists of adding the pile-up to the generator level input in the very beginnning of the simulation and subtracting its estimated effects on detector level at the end. Adding the pile-up is done using the `PileUpMerger` module, which takes an external file containing generator level events which are common for proton-proton collisions at the given center-of-mass energy of 13 TeV. A few other parameters define the mean number of pile-up events (in this case, 50) as well as the spread of the events in time and along the beam axis. The subtraction is performed by the `JetPileUpSubtractor` module which reduces the transverse momentum by the average expected transverse momentum originating from pile-up in the jet's area.

After applying the adjustments discussed in the previous sections to this pile-up card, it is used to gauge the effects pile-up has on the simulation. The results are mixed: While the agreement of the jet p_T distributions is improved significantly (figure 7.21), jet multiplicity is underestimated much more than previously (figure 7.22) and a significant portion of the jets are calculated to have negative masses (figure 7.23), which is not physical.

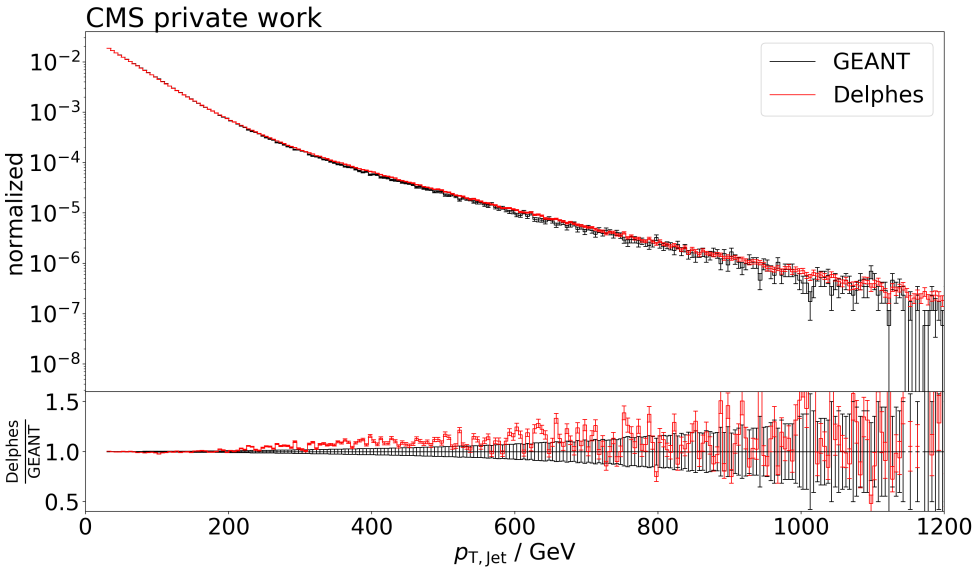

Figure 7.21: Histogram of the jet transverse momentum distribution for ttbb events, using a Delphes card which includes pile-up, normalized to an integral of 1.

Further worth noting is that, while pile-up did increase the number of detector-level leptons in the Delphes samples, the effect is approximately an order of magnitude smaller than the discrepancy discussed in section 7.4.

Due to the dubious effects which including pile-up had on the result, it was decided not to include it for the remaining study.

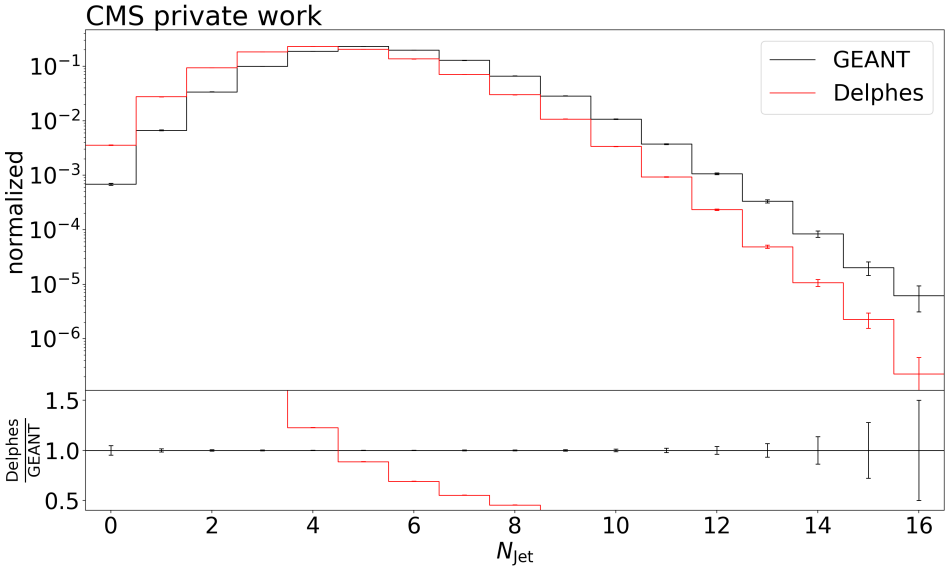

Figure 7.22: Histogram of the jet multiplicity for ttbb events, using a Delphes card which includes pile-up, normalized to an integral of 1.

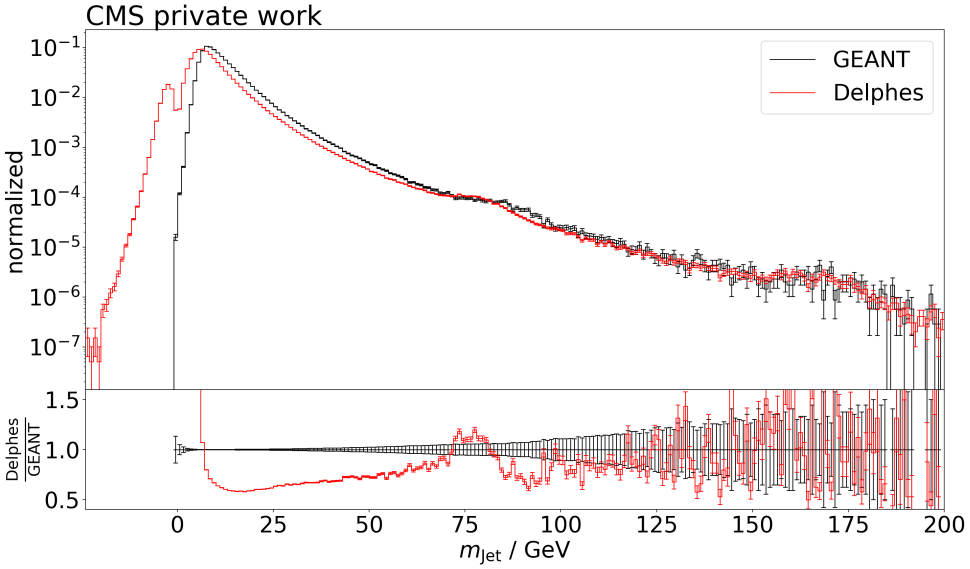

Figure 7.23: Histogram of the jet invariant mass distribution for ttbb events, using a Delphes card which includes pile-up, normalized to an integral of 1.

7.8 Other deviations

Several further issues remain entirely unresolved.

One of these is the ϕ-asymmetry of the detector. As can be most clearly seen in the ϕ_{BJet} distribution (figure 7.24), the detection efficiency is slightly reduced in a certain direction (towards the center of the accelerator ring, approximately 40 ° up). As Delphes is not designed to work with non radially symmetric situations, it is incapable of simulating this. Attempting to mediate this would require changing the code of Delphes, which goes beyond the scope of this thesis.

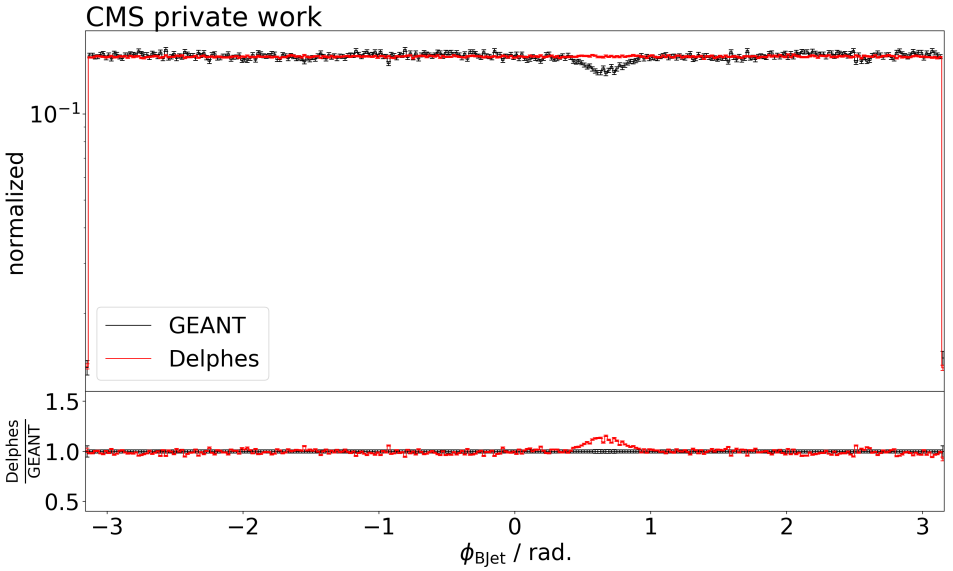

Figure 7.24: Histogram of the b-jet azimuthal angle distribution for ttbb events, using the final Delphes card, normalized to an integral of 1.

Another issue is the inaccuracy in the jet mass plots (figures 7.25). Overall, it appears that the entire distribution is shifted by about 5 GeV. Potentially, this can be remedied by adjusting the "`EnergyScale`" module which adjusts jet energies. However, due to the lack of a clear alternative formula to use and the relatively minor impact of such adjustment, this is not attempted here.

Further, the multiplicities of jets and b-jets deviate from the target significantly (figures 7.26 and 7.27). The overabundance of events with high b-jet multiplicity can be attributed to the same discrepancy found on generator level, discussed in section 7.1. The origin of all other discrepancies remains unclear.

Finally, when looking at the $\Delta R(bb)$ distributions (figure 7.28), while most features appear to be modeled accurately, there is a noticeable overabundance of close pairs of b-jets. One feasible hypothesis is that this is related to the overabundance of events with high b-jet multiplicity, as more b-jets in an event mean less distance between them, on average. This hypothesis is supported by the fact that this discrepancy remains even on generator level (figure 7.29).

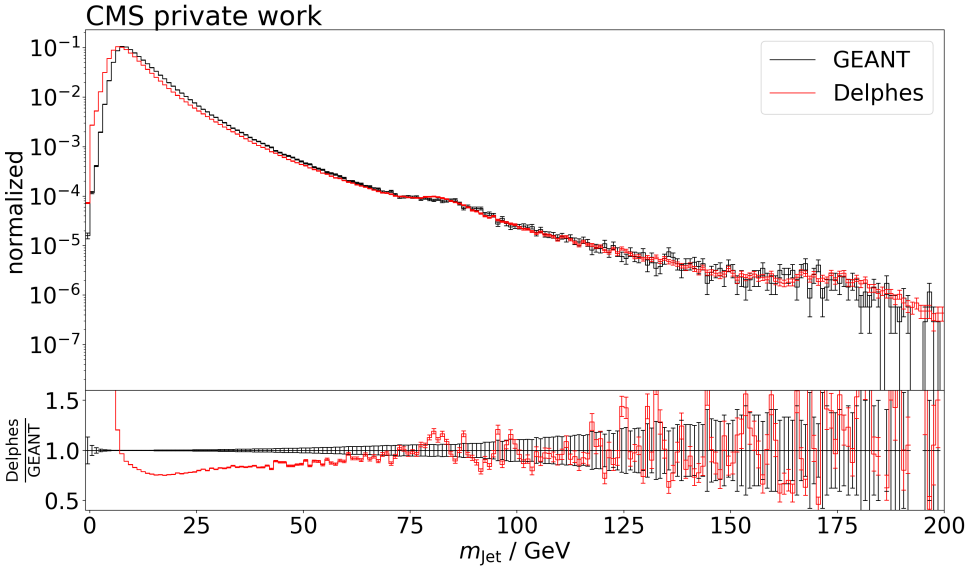

Figure 7.25: Histogram of the jet invariant mass distribution for ttbb events, using the final Delphes card, normalized to an integral of 1.

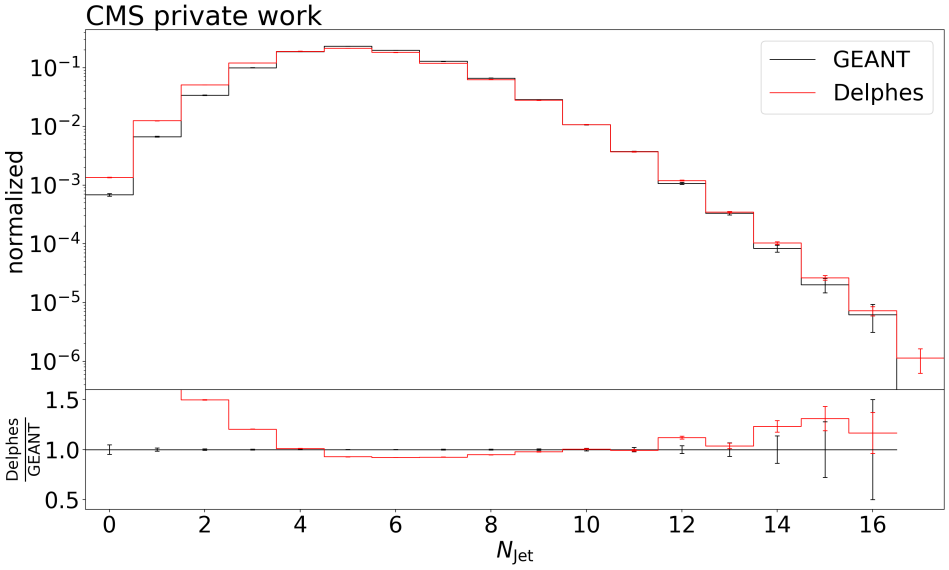

Figure 7.26: Histogram of the jet multiplicity for ttbb events, using the final Delphes card, normalized to an integral of 1.

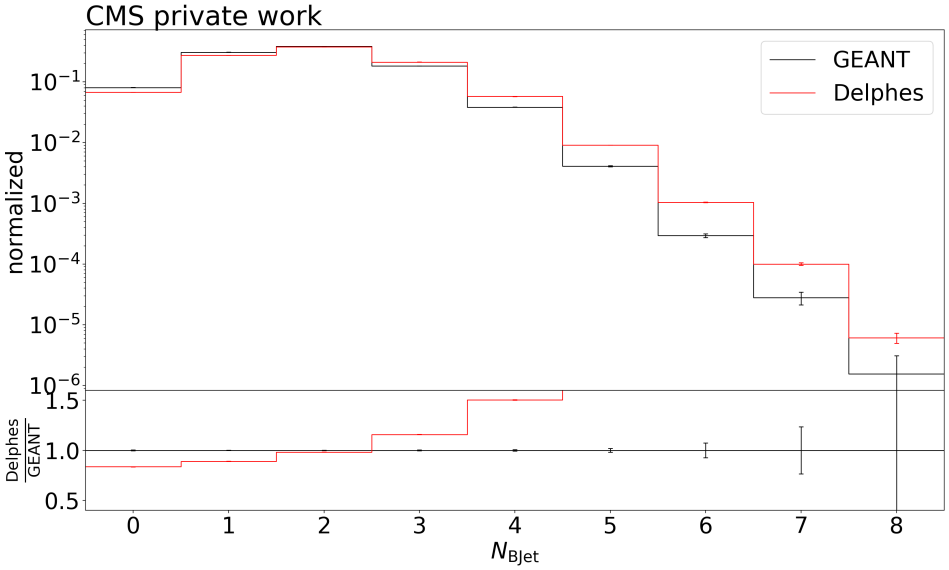

Figure 7.27: Histogram of the b-jet multiplicity for ttbb events, using the final Delphes card, normalized to an integral of 1.

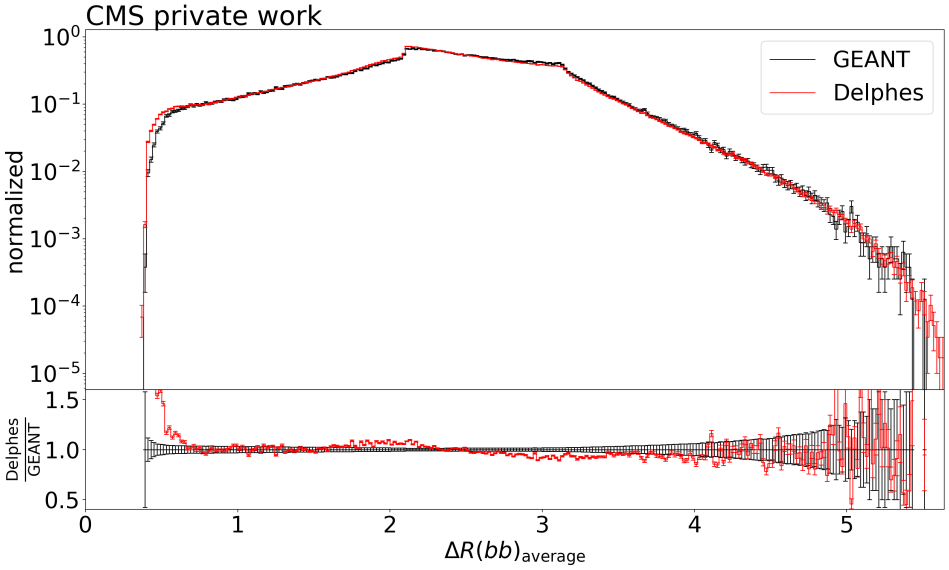

Figure 7.28: Histogram of the average ΔR of b-jet pairs for ttbb events, using the final Delphes card, normalized to an integral of 1.

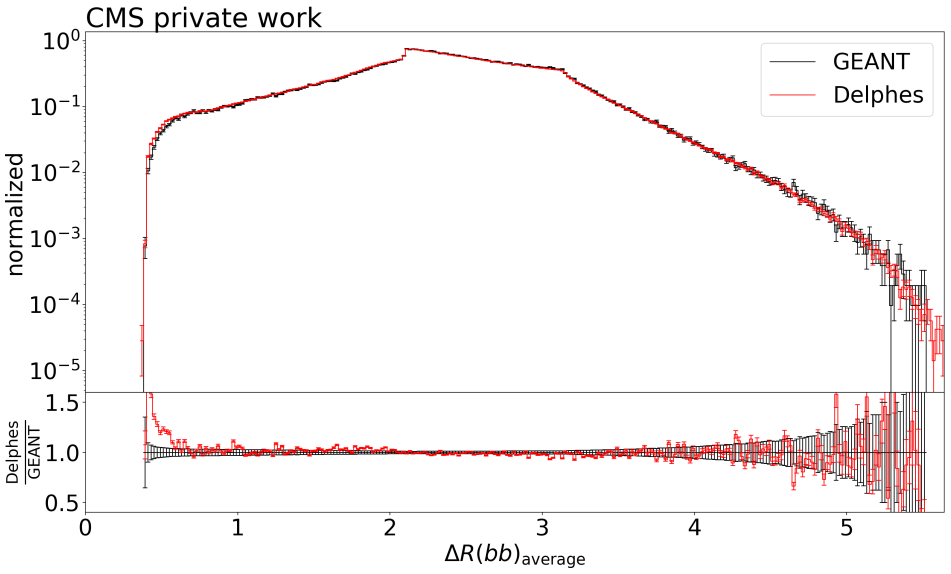

Figure 7.29: Histogram of the average ΔR of b-jet pairs for ttbb events at generator level, using the final Delphes card, normalized to an integral of 1.

7.9 Application to ttH and ttZ

After adjusting the Delphes card based on the ttbb simulation, it is now applied to ttH and ttZ simulations. As the detector properties and the specific process investigated should be almost completely independent from one another, the expectation is that the deviations between GEANT and Delphes samples will be the same independent of the specific process.

This expectation indeed holds up well. For most of the kinematic observables considered, the alignment of the GEANT-Delphes factor between the three processes is perfect or almost perfect save for statistical deviations (figure 7.30 as an example, section B of the appendix contains similar histograms for all observables considered).

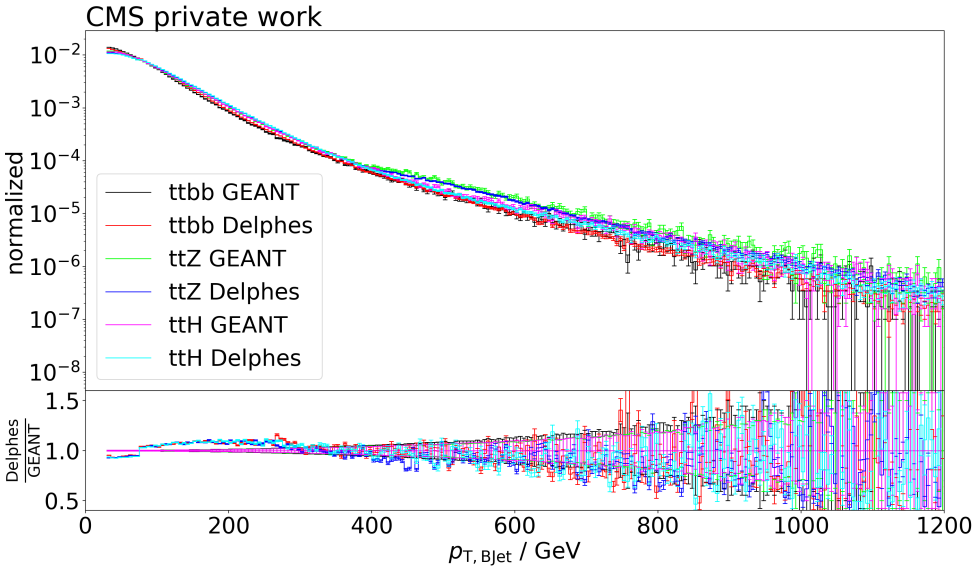

Figure 7.30: Histogram of the b-jet transverse momentum distribution for ttbb, ttH and ttZ events, using the final Delphes card, normalized to an integral of 1.

The one exception are lepton multiplicities (figures 7.31 and 7.32). While ttH and ttZ align very well with one another, they do not align quite as well with ttbb, likely because the event topologies of ttH and ttZ are very similar, whereas ttbb differs significantly from both, likely due to including a much wider array of event topologies, including many Feynman diagrams without an analogon in ttH and ttZ. Because the lepton multiplicities are also the observables for which the deviations between GEANT and Delphes remain the largest, it is likely that the deviations between ttbb, ttH and ttZ are related to this.

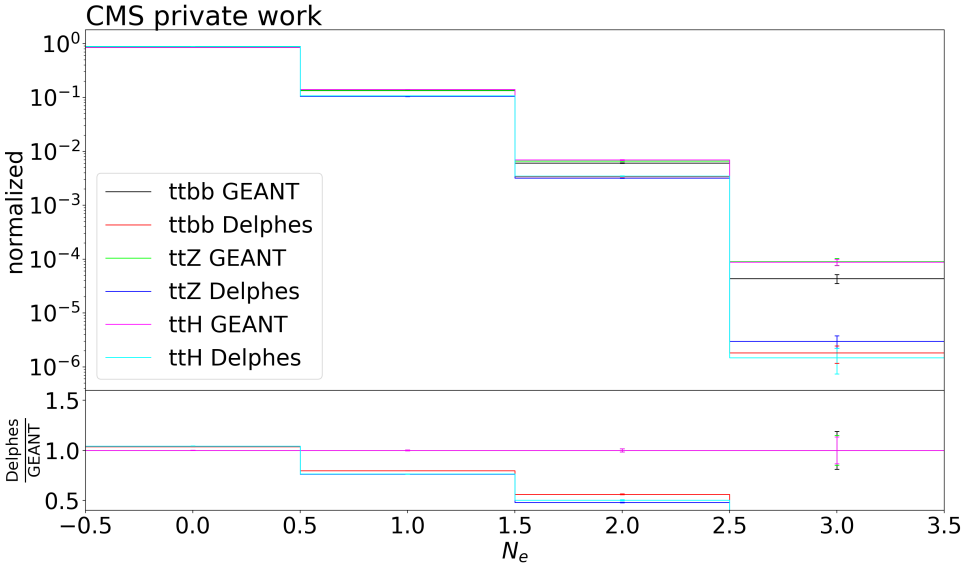

Figure 7.31: Histogram of the electron multiplicity for ttbb, ttH and ttZ events, using the final Delphes card, normalized to an integral of 1.

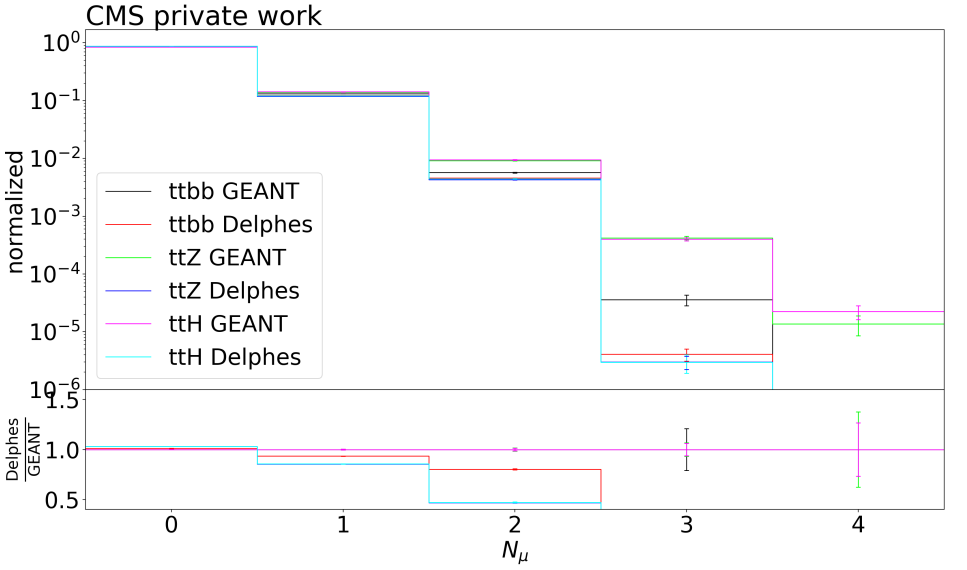

Figure 7.32: Histogram of the muon multiplicity for ttbb, ttH and ttZ events, using the final Delphes card, normalized to an integral of 1.

8 Conclusion

Simulations of collider events are an integral part of modern particle physics research. At the CMS experiment at the CERN LHC, it is standard for the detector simulation to be performed using the GEANT software package. This, however, may not be ideal under certain circumstances due to its resource-intensiveness. For certain applications with lower requirements to the precision of final results, the open-source software Delphes can be better suited.

This thesis seeks to investigate the feasibility of using Delphes for certain specific processes. Under consideration are processes with an associated production of a top quark-antiquark pair together with either a bottom quark-antiquark pair, a Higgs-boson or a Z-boson.

After considering the tools used for the simulation and devising a workflow which allows for maximum isolation of the discrepancies in detector simulation software from the event generation and the parton showering simulation, a comparison between the results is performed for ttbb processes.

A generator level comparison verifies the fact that, save for statistical variation, both softwares deal with equivalent particle-level information, while at the same time uncovering a discrepancy in the flavor association algorithms used for generator level jets.

A comparison on detector level identifies many discrepancies. Attempts are made to remedy each identified problem individually. Some, especially problems with handling lepton and photon isolation, are successfully mitigated, others, such as suboptimal efficiency formulas, only in a limited capacity. Yet others are found to be caused by fundamental insufficiencies in Delphes, most notably the lack of lepton misidentification simulation, the lack of secondary particles simulation, and an inconsistency between the jet flavor association algorithms of Delphes and GEANT.

An attempt is made to include pile-up in the simulation. While results are promising in some areas, they are associated with highly inadequate results in other areas.

Overall, the results are workable, with basic kinematic observable distributions deviating by at most 10 % in relevant sections.

Next, the Delphes simulation is applied to ttH and ttZ processes. The results show that most relative deviations between Delphes and GEANT do not depend on the specific process simulated, at least not in the case of the processes considered in this work. From

this, one can conclude that most of the deviations that remain can be further reduced by adjusting numerical parameters for Delphes more precisely.

Noteworthy for an outlook towards future research are the two issues which are responsible for significant deviations in this work while being properties of Delphes itself, rather than being something that can be adjusted through configuration. The first is the jet flavor association algorithm. It may prove worthwhile to investigate the discrepancies between Delphes and GEANT in this regard more closely, and potentially create a method to mitigate them. As this discrepancy is present on generator level, where the actual detector simulation has yet to take effect, it is likely that such an adjustment is possible, though it will likely require changing the internal code of Delphes. The second is the simulation of particle misidentification. According to [22], the creation of a new Delphes module for this is possible in principle. Third is the simulation of secondary interactions and particles, although it is unclear if and how this could be implemented in Delphes.

A further avenue for improvements to this work's results is to derive more accurate efficiency formulas, especially for leptons and b-tagging. While an improvement from the default efficiencies has been achieved here, noticeable deviations remain.

The investigation of Delphes' capabilities to simulate pile-up is rather basic in this thesis. As part of the results here show notable promise, and as pile-up increases in importance with increasing collider luminosities, finding a way to circumvent the problems encountered in this work can be a highly valuable objective.

Finally, investigating the effect of various selection criteria on the simulation can be beneficial. While this is not done here due to the high degree of independence of the detector simulation from the generator level simulation, a possibility of unexpected findings remains and should be investigated.

Bibliography

[1] B. Povh. *Particles and Nuclei : An Introduction to the Physical Concepts*. Ed. by K. Rith et al. Berlin, Heidelberg, 2015.

[2] G. Aad et al. "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". In: *Physics Letters B* 716.1 (Sept. 2012), pp. 1–29. DOI: 10.1016/j.physletb.2012.08.020.

[3] S. Chatrchyan et al. "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC". In: *Physics Letters B* 716.1 (Sept. 2012), pp. 30–61. DOI: 10.1016/j.physletb.2012.08.021.

[4] F. Englert and R. Brout. "Broken Symmetry and the Mass of Gauge Vector Mesons". In: *Phys. Rev. Lett.* 13 (9 Aug. 1964), pp. 321–323. DOI: 10.1103/PhysRevLett.13.321.

[5] P. W. Higgs. "Broken Symmetries and the Masses of Gauge Bosons". In: *Phys. Rev. Lett.* 13 (16 Oct. 1964), pp. 508–509. DOI: 10.1103/PhysRevLett.13.508.

[6] G. S. Guralnik, C. R. Hagen, and T. W. B. Kibble. "Global Conservation Laws and Massless Particles". In: *Phys. Rev. Lett.* 13 (20 Nov. 1964), pp. 585–587. DOI: 10.1103/PhysRevLett.13.585.

[7] J. C. Collins, D. E. Soper, and G. Sterman. "Factorization of Hard Processes in QCD". In: (2004). DOI: 10.48550/ARXIV.HEP-PH/0409313.

[8] S. Fartoukh et al. *LHC Configuration and Operational Scenario for Run 3*. Tech. rep. Geneva: CERN, 2021. URL: http://cds.cern.ch/record/2790409.

[9] V. Khachatryan et al. "The CMS trigger system". In: *Journal of Instrumentation* 12.01 (Jan. 2017), P01020–P01020. DOI: 10.1088/1748-0221/12/01/p01020.

[10] S. R. Davis. *Interactive Slice of the CMS detector*. 2016. URL: https://cds.cern.ch/record/2205172.

[11] S. Chatrchyan et al. "The CMS Experiment at the CERN LHC". In: *JINST* 3 (2008), S08004. DOI: 10.1088/1748-0221/3/08/S08004.

[12] F. Beaudette. "The CMS Particle Flow Algorithm". In: *Proceedings of the CHEF2013 Conference*. arXiv, 2014. DOI: 10.48550/ARXIV.1401.8155.

[13] M. Cacciari, G. P. Salam, and G. Soyez. "The anti-k_t jet clustering algorithm". In: *Journal of High Energy Physics* 2008.04 (Apr. 2008), pp. 063–063. DOI: 10.1088/1126-6708/2008/04/063.

[14] M. Stoye and on behalf of the CMS collaboration. "Deep learning in jet reconstruction at CMS". In: *Journal of Physics: Conference Series* 1085 (Sept. 2018), p. 042029. DOI: 10.1088/1742-6596/1085/4/042029.

[15] E. Pfeffer. "Studies on tt+bb production at the CMS experiment". MA thesis. Karlsruhe Institute of Technology (KIT), 2021. URL: https://publish.etp.kit.edu/record/22046.

[16] J. Alwall et al. “The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations”. In: *Journal of High Energy Physics* 2014.7 (July 2014). DOI: 10.1007/jhep07(2014)079.

[17] P. Artoisenet et al. “Automatic spin-entangled decays of heavy resonances in Monte Carlo simulations”. In: *Journal of High Energy Physics* 2013.3 (Mar. 2013). DOI: 10.1007/jhep03(2013)015.

[18] C. Bierlich et al. *A comprehensive guide to the physics and usage of PYTHIA 8.3*. 2022. DOI: 10.48550/ARXIV.2203.11601.

[19] S. Agostinelli et al. “Geant4—a simulation toolkit”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 506.3 (2003), pp. 250–303. ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(03)01368-8.

[20] J. Allison et al. “Geant4 developments and applications”. In: *IEEE Transactions on Nuclear Science* 53.1 (2006), pp. 270–278. DOI: 10.1109/TNS.2006.869826.

[21] J. Allison et al. “Recent developments in Geant4”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 835 (2016), pp. 186–225. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2016.06.125.

[22] J. de Favereau et al. “DELPHES 3: a modular framework for fast simulation of a generic collider experiment”. In: *Journal of High Energy Physics* 2014.2 (Feb. 2014). DOI: 10.1007/jhep02(2014)057.

[23] R. L. Workman et al. “Review of Particle Physics”. In: *PTEP* 2022 (2022), p. 083C01. DOI: 10.1093/ptep/ptac097.

[24] R. Brun and F. Rademakers. “ROOT - An Object Oriented Data Analysis Framework”. In: *AIHENP’96 Workshop, Lausane*. Vol. 389. 1996, pp. 81–86.

[25] M. Dobbs and J. B. Hansen. “The HepMC C++ Monte Carlo event record for High Energy Physics”. In: *Computer Physics Communications* 134.1 (2001), pp. 41–46. ISSN: 0010-4655. DOI: https://doi.org/10.1016/S0010-4655(00)00189-2.

[26] S. Zenz. *CMS "short-term" version of Delphes - starting with ECFA_v2*. https://github.com/sethzenz/Delphes. 2015. (Visited on 10/04/2022).

[27] J. Hornung. “Studien zur schnellen Ereignissimulation mit Delphes”. BA thesis. Karlsruhe Institute of Technology (KIT), 2021. URL: https://publish.etp.kit.edu/record/22082.

[28] E. Bols et al. “Jet flavour classification using DeepJet”. In: *Journal of Instrumentation* 15.12 (Dec. 2020), P12012–P12012. DOI: 10.1088/1748-0221/15/12/p12012.

Appendix

A Final Delphes card

```
set ExecutionPath {
ParticlePropagator
ChargedHadronTrackingEfficiency
ElectronTrackingEfficiency
MuonTrackingEfficiency
ChargedHadronMomentumSmearing
ElectronMomentumSmearing
MuonMomentumSmearing
TrackMerger
ECal
HCal
Calorimeter
EFlowMerger
EFlowFilter
PhotonEfficiency
PhotonIsolation
ElectronFilter
ElectronEfficiency
ElectronIsolation
ChargedHadronFilter
MuonEfficiency
MuonIsolation
MissingET
NeutrinoFilter
GenJetFinder
GenMissingET
FastJetFinder
FatJetFinder
JetEnergyScale
JetFlavorAssociation
GenJetFlavorAssociation
BTagging
```

```
TauTagging
UniqueObjectFinder
ScalarHT
TreeWriter
}

module ParticlePropagator ParticlePropagator {
set InputArray Delphes/stableParticles
set OutputArray stableParticles
set ChargedHadronOutputArray chargedHadrons
set ElectronOutputArray electrons
set MuonOutputArray muons
set Radius 1.29
set HalfLength 3.00
set Bz 3.8
}

module Efficiency ChargedHadronTrackingEfficiency {
set InputArray ParticlePropagator/chargedHadrons
set OutputArray chargedHadrons
set EfficiencyFormula {                                (pt <= 0.1) * (0.00) +
                  (abs(eta) <= 1.5) * (pt > 0.1   && pt <= 1.0) * (0.85) +
                  (abs(eta) <= 1.5) * (pt > 1.0)                * (0.97) +
(abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 0.1   && pt <= 1.0) * (0.85) +
(abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 1.0)                * (0.90) +
(abs(eta) > 2.5)                                                * (0.00)}
}

module Efficiency ElectronTrackingEfficiency {
set InputArray ParticlePropagator/electrons
set OutputArray electrons
set EfficiencyFormula {                                  (pt <= 0.1)   * (0.00) +
                  (abs(eta) <= 1.5) * (pt > 0.1   && pt <= 1.0)   * (0.85) +
                  (abs(eta) <= 1.5) * (pt > 1.0   && pt <= 1.0e2) * (0.97) +
                  (abs(eta) <= 1.5) * (pt > 1.0e2)                * (0.99) +
(abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 0.1   && pt <= 1.0)   * (0.85) +
(abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 1.0   && pt <= 1.0e2) * (0.90) +
(abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 1.0e2)                * (0.95) +
(abs(eta) > 2.5)                                                  * (0.00)}
}

module Efficiency MuonTrackingEfficiency {
set InputArray ParticlePropagator/muons
set OutputArray muons
set EfficiencyFormula {                                  (pt <= 0.1)   * (0.00) +
                  (abs(eta) <= 1.5) * (pt > 0.1   && pt <= 1.0)   * (0.998) +
                  (abs(eta) <= 1.5) * (pt > 1.0)                  * (0.9998) +
(abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 0.1   && pt <= 1.0)   * (0.98) +
(abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 1.0)                  * (0.98) +
(abs(eta) > 2.5)                                                  * (0.00)}
}
```

```
module MomentumSmearing ChargedHadronMomentumSmearing {
set InputArray ChargedHadronTrackingEfficiency/chargedHadrons
set OutputArray chargedHadrons
set ResolutionFormula {(abs(eta) <= 0.5) * (pt > 0.1) * sqrt(0.06^2 + pt^2*1.3e-3^2) +
     (abs(eta) > 0.5 && abs(eta) <= 1.5) * (pt > 0.1) * sqrt(0.10^2 + pt^2*1.7e-3^2) +
     (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 0.1) * sqrt(0.25^2 + pt^2*3.1e-3^2)}
}


module MomentumSmearing ElectronMomentumSmearing {
set InputArray ElectronTrackingEfficiency/electrons
set OutputArray electrons
set ResolutionFormula {(abs(eta) <= 0.5) * (pt > 0.1) * sqrt(0.03^2 + pt^2*1.3e-3^2) +
     (abs(eta) > 0.5 && abs(eta) <= 1.5) * (pt > 0.1) * sqrt(0.05^2 + pt^2*1.7e-3^2) +
     (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 0.1) * sqrt(0.15^2 + pt^2*3.1e-3^2)}
}


module MomentumSmearing MuonMomentumSmearing {
set InputArray MuonTrackingEfficiency/muons
set OutputArray muons
set ResolutionFormula {(abs(eta) <= 0.5) * (pt > 0.1) * sqrt(0.01^2 + pt^2*1.0e-4^2) +
     (abs(eta) > 0.5 && abs(eta) <= 1.5) * (pt > 0.1) * sqrt(0.015^2 + pt^2*1.5e-4^2) +
     (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 0.1) * sqrt(0.025^2 + pt^2*3.5e-4^2)}
}


module Merger TrackMerger {
add InputArray ChargedHadronMomentumSmearing/chargedHadrons
add InputArray ElectronMomentumSmearing/electrons
add InputArray MuonMomentumSmearing/muons
set OutputArray tracks
}


module SimpleCalorimeter ECal {
set ParticleInputArray ParticlePropagator/stableParticles
set TrackInputArray TrackMerger/tracks
set TowerOutputArray ecalTowers
set EFlowTrackOutputArray eflowTracks
set EFlowTowerOutputArray eflowPhotons
set IsEcal true
set EnergyMin 0.5
set EnergySignificanceMin 2.0
set SmearTowerCenter true
set pi [expr {acos(-1)}]
set PhiBins {}
for {set i -180} {$i <= 180} {incr i} {
  add PhiBins [expr {$i * $pi/180.0}]
}
for {set i -85} {$i <= 86} {incr i} {
  set eta [expr {$i * 0.0174}]
  add EtaPhiBins $eta $PhiBins
}
```

```
set PhiBins {}
for {set i -180} {$i <= 180} {incr i} {
  add PhiBins [expr {$i * $pi/180.0}]
}
for {set i 1} {$i <= 84} {incr i} {
  set eta [expr { -2.958 + $i * 0.0174}]
  add EtaPhiBins $eta $PhiBins
}
for {set i 1} {$i <= 84} {incr i} {
  set eta [expr { 1.4964 + $i * 0.0174}]
  add EtaPhiBins $eta $PhiBins
}
set PhiBins {}
for {set i -18} {$i <= 18} {incr i} {
  add PhiBins [expr {$i * $pi/18.0}]
}
foreach eta {-5 -4.7 -4.525 -4.35 -4.175 -4 -3.825 -3.65 -3.475 -3.3 -3.125 -2.958
             3.125 3.3 3.475 3.65 3.825 4 4.175 4.35 4.525 4.7 5} {
  add EtaPhiBins $eta $PhiBins
}
add EnergyFraction {0} {0.0}
add EnergyFraction {11} {1.0}
add EnergyFraction {22} {1.0}
add EnergyFraction {111} {1.0}
add EnergyFraction {12} {0.0}
add EnergyFraction {13} {0.0}
add EnergyFraction {14} {0.0}
add EnergyFraction {16} {0.0}
add EnergyFraction {1000022} {0.0}
add EnergyFraction {1000023} {0.0}
add EnergyFraction {1000025} {0.0}
add EnergyFraction {1000035} {0.0}
add EnergyFraction {1000045} {0.0}
add EnergyFraction {310} {0.3}
add EnergyFraction {3122} {0.3}
set ResolutionFormula {(abs(eta) <= 1.5) *
             (1+0.64*eta^2) * sqrt(energy^2*0.008^2 + energy*0.11^2 + 0.40^2) +
     (abs(eta) > 1.5 && abs(eta) <= 2.5) *
(2.16 + 5.6*(abs(eta)-2)^2) * sqrt(energy^2*0.008^2 + energy*0.11^2 + 0.40^2) +
     (abs(eta) > 2.5 && abs(eta) <= 5.0) *
                              sqrt(energy^2*0.107^2 + energy*2.08^2)}
}


module SimpleCalorimeter HCal {
set ParticleInputArray ParticlePropagator/stableParticles
set TrackInputArray ECal/eflowTracks
set TowerOutputArray hcalTowers
set EFlowTrackOutputArray eflowTracks
set EFlowTowerOutputArray eflowNeutralHadrons
set IsEcal false
set EnergyMin 1.0
```

```
set EnergySignificanceMin 1.0
set SmearTowerCenter true
set pi [expr {acos(-1)}]
set PhiBins {}
for {set i -180} {$i <= 180} {incr i} {
  add PhiBins [expr {$i * $pi/180.0}]
}
for {set i 1} {$i <= 440} {incr i} {
  set eta [expr { -4.4 + $i * 0.02}]
  add EtaPhiBins $eta $PhiBins
}
set PhiBins {}
for {set i -9} {$i <= 9} {incr i} {
  add PhiBins [expr {$i * $pi/9.0}]
}
foreach eta {-5 -4.7 -4.4 4.7 5} {
  add EtaPhiBins $eta $PhiBins
}
add EnergyFraction {0} {1.0}
add EnergyFraction {11} {0.0}
add EnergyFraction {22} {0.0}
add EnergyFraction {111} {0.0}
add EnergyFraction {12} {0.0}
add EnergyFraction {13} {0.0}
add EnergyFraction {14} {0.0}
add EnergyFraction {16} {0.0}
add EnergyFraction {1000022} {0.0}
add EnergyFraction {1000023} {0.0}
add EnergyFraction {1000025} {0.0}
add EnergyFraction {1000035} {0.0}
add EnergyFraction {1000045} {0.0}
add EnergyFraction {310} {0.7}
add EnergyFraction {3122} {0.7}
set ResolutionFormula {(abs(eta) <= 3.0) * sqrt(energy^2*0.050^2 + energy*1.50^2) +
     (abs(eta) > 3.0 && abs(eta) <= 5.0) * sqrt(energy^2*0.130^2 + energy*2.70^2)}
}

module PdgCodeFilter ElectronFilter {
set InputArray HCal/eflowTracks
set OutputArray electrons
set Invert true
add PdgCode {11}
add PdgCode {-11}
}

module PdgCodeFilter ChargedHadronFilter {
set InputArray HCal/eflowTracks
set OutputArray chargedHadrons
add PdgCode {11}
add PdgCode {-11}
add PdgCode {13}
```

```
add PdgCode {-13}
}


module Merger Calorimeter {
add InputArray ECal/ecalTowers
add InputArray HCal/hcalTowers
set OutputArray towers
}


module Merger EFlowMerger {
add InputArray HCal/eflowTracks
add InputArray ECal/eflowPhotons
add InputArray HCal/eflowNeutralHadrons
set OutputArray eflow
}


module PdgCodeFilter EFlowFilter {
set InputArray EFlowMerger/eflow
set OutputArray eflow
add PdgCode {11}
add PdgCode {-11}
add PdgCode {13}
add PdgCode {-13}
}


module Efficiency PhotonEfficiency {
set InputArray ECal/eflowPhotons
set OutputArray photons
set EfficiencyFormula {                (pt <= 10.0) * (0.00) +
                  (abs(eta) <= 1.5) * (pt > 10.0)  * (0.9635) +
(abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 10.0)  * (0.9624) +
(abs(eta) > 2.5)                                   * (0.00)}
}


module Isolation PhotonIsolation {
set CandidateInputArray PhotonEfficiency/photons
set IsolationInputArray EFlowFilter/eflow
set OutputArray photons
set DeltaRMax 0.3
set PTMin 0.5
set PTRatioMax 0.12
}


module Efficiency ElectronEfficiency {
set InputArray ElectronFilter/electrons
set OutputArray electrons
set EfficiencyFormula {                                (pt <=  4.0) * (0.00) +
(abs(eta) <= 1.45 )                  * (pt >  4.0 && pt <=  6.0) * (0.50) +
(abs(eta) <= 1.45 )                  * (pt >  6.0 && pt <=  8.0) * (0.70) +
```

```
(abs(eta) <= 1.45 )                    * (pt >  8.0 && pt <= 10.0) * (0.85) +
(abs(eta) <= 1.45 )                    * (pt > 10.0 && pt <= 30.0) * (0.94) +
(abs(eta) <= 1.45 )                    * (pt > 30.0 && pt <= 50.0) * (0.97) +
(abs(eta) <= 1.45 )                    * (pt > 50.0 && pt <= 70.0) * (0.98) +
(abs(eta) <= 1.45 )                    * (pt > 70.0 )              * (1.0) +
(abs(eta) > 1.45 && abs(eta) <= 1.55) * (pt >  4.0 && pt <= 10.0) * (0.35) +
(abs(eta) > 1.45 && abs(eta) <= 1.55) * (pt > 10.0 && pt <= 30.0) * (0.40) +
(abs(eta) > 1.45 && abs(eta) <= 1.55) * (pt > 30.0 && pt <= 70.0) * (0.45) +
(abs(eta) > 1.45 && abs(eta) <= 1.55) * (pt > 70.0 )              * (0.45) +
(abs(eta) >= 1.55 && abs(eta) <= 2.0) * (pt >  4.0 && pt <= 10.0) * (0.75) +
(abs(eta) >= 1.55 && abs(eta) <= 2.0) * (pt > 10.0 && pt <= 30.0) * (0.85) +
(abs(eta) >= 1.55 && abs(eta) <= 2.0) * (pt > 30.0 && pt <= 50.0) * (0.95) +
(abs(eta) >= 1.55 && abs(eta) <= 2.0) * (pt > 50.0 && pt <= 70.0) * (0.95) +
(abs(eta) >= 1.55 && abs(eta) <= 2.0) * (pt > 70.0 )              * (1.0) +
(abs(eta) >= 2.0  && abs(eta) <= 2.5) * (pt >  4.0 && pt <= 10.0) * (0.65) +
(abs(eta) >= 2.0  && abs(eta) <= 2.5) * (pt > 10.0 && pt <= 30.0) * (0.75) +
(abs(eta) >= 2.0  && abs(eta) <= 2.5) * (pt > 30.0 && pt <= 50.0) * (0.85) +
(abs(eta) >= 2.0  && abs(eta) <= 2.5) * (pt > 50.0 && pt <= 70.0) * (0.85) +
(abs(eta) >= 2.0  && abs(eta) <= 2.5) * (pt > 70.0 )              * (0.85) +
(abs(eta) > 2.5)                                                  * (0.00)}
}


module Isolation ElectronIsolation {
set CandidateInputArray ElectronEfficiency/electrons
set IsolationInputArray EFlowFilter/eflow
set OutputArray electrons
set DeltaRMax 0.3
set PTMin 0.5
set PTRatioMax 0.12
}


module Efficiency MuonEfficiency {
set InputArray MuonMomentumSmearing/muons
set OutputArray muons
set EfficiencyFormula {           (pt <=  2.0) * (0.00) +
(abs(eta) <= 2.40) * (pt >  2.0 && pt <=  3.0) * (0.51) +
(abs(eta) <= 2.40) * (pt >  3.0 && pt <=  4.0) * (0.85) +
(abs(eta) <= 2.40) * (pt >  4.0 && pt <= 11.0) * (0.93) +
(abs(eta) <= 2.40) * (pt > 11.0 && pt <= 50.0) * (0.96) +
(abs(eta) <= 2.40) * (pt > 50.0 && pt <= 70.0) * (0.98) +
(abs(eta) <= 2.40) * (pt > 70.0 )              * (1.00) +
(abs(eta) > 2.40)                              * (0.00)}
}


module Isolation MuonIsolation {
set CandidateInputArray MuonEfficiency/muons
set IsolationInputArray EFlowFilter/eflow
set OutputArray muons
set DeltaRMax 0.4
set PTMin 0.5
```

```
set PTRatioMax 0.25
}


module Merger MissingET {
add InputArray EFlowMerger/eflow
set MomentumOutputArray momentum
}


module Merger ScalarHT {
add InputArray UniqueObjectFinder/jets
add InputArray UniqueObjectFinder/electrons
add InputArray UniqueObjectFinder/photons
add InputArray UniqueObjectFinder/muons
set EnergyOutputArray energy
}


module PdgCodeFilter NeutrinoFilter {
set InputArray Delphes/stableParticles
set OutputArray filteredParticles
set PTMin 0.0
add PdgCode {12}
add PdgCode {14}
add PdgCode {16}
add PdgCode {-12}
add PdgCode {-14}
add PdgCode {-16}
}


module FastJetFinder GenJetFinder {
set InputArray NeutrinoFilter/filteredParticles
set OutputArray jets
set JetAlgorithm 6
set ParameterR 0.4
set JetPTMin 10.0
}


module Merger GenMissingET {
add InputArray NeutrinoFilter/filteredParticles
set MomentumOutputArray momentum
}


module FastJetFinder FastJetFinder {
set InputArray EFlowMerger/eflow
set OutputArray jets
set JetAlgorithm 6
set ParameterR 0.4
set JetPTMin 10.0
}
```

```
module FastJetFinder FatJetFinder {
set InputArray EFlowMerger/eflow
set OutputArray jets
set JetAlgorithm 6
set ParameterR 0.8
set ComputeNsubjettiness 1
set Beta 1.0
set AxisMode 4
set ComputeTrimming 1
set RTrim 0.2
set PtFracTrim 0.05
set ComputePruning 1
set ZcutPrun 0.1
set RcutPrun 0.5
set RPrun 0.8
set ComputeSoftDrop 1
set BetaSoftDrop 0.0
set SymmetryCutSoftDrop 0.1
set R0SoftDrop 0.8
set JetPTMin 200.0
}

module EnergyScale JetEnergyScale {
set InputArray FastJetFinder/jets
set OutputArray jets
set ScaleFormula {sqrt((2.5 - 0.15 * (abs(eta)))^2 / pt + 1.0 )}
}

module JetFlavorAssociation JetFlavorAssociation {
set PartonInputArray Delphes/partons
set ParticleInputArray Delphes/allParticles
set ParticleLHEFInputArray Delphes/allParticlesLHEF
set JetInputArray JetEnergyScale/jets
set DeltaR 0.4
set PartonPTMin 1.0
set PartonEtaMax 2.5
}

module JetFlavorAssociation GenJetFlavorAssociation {
set PartonInputArray Delphes/partons
set ParticleInputArray Delphes/allParticles
set ParticleLHEFInputArray Delphes/allParticlesLHEF
set JetInputArray GenJetFinder/jets
set DeltaR 0.4
set PartonPTMin 1.0
set PartonEtaMax 2.5
}

module BTagging BTagging {
set JetInputArray JetEnergyScale/jets
```

```
set BitNumber 0
add EfficiencyFormula {0} {0.01}
add EfficiencyFormula {4} {0.18}
add EfficiencyFormula {5} {(pt <=  75.0)  * (0.76) +
            (pt >   75.0 && pt <= 150.0)  * (0.83) +
            (pt >  150.0 && pt <= 225.0)  * (0.85) +
            (pt >  225.0 && pt <= 300.0)  * (0.85) +
            (pt >  300.0 && pt <= 375.0)  * (0.82) +
            (pt >  375.0 && pt <= 450.0)  * (0.79) +
            (pt >  450.0 && pt <= 525.0)  * (0.79) +
            (pt >  525.0 && pt <= 600.0)  * (0.78) +
            (pt >  600.0 && pt <= 675.0)  * (0.76) +
            (pt >  675.0 && pt <= 750.0)  * (0.73) +
            (pt >  750.0)                 * (0.66)}
}

module TauTagging TauTagging {
set ParticleInputArray Delphes/allParticles
set PartonInputArray Delphes/partons
set JetInputArray JetEnergyScale/jets
set DeltaR 0.4
set TauPTMin 1.0
set TauEtaMax 2.5
add EfficiencyFormula {0} {0.01}
add EfficiencyFormula {15} {0.6}
}

module UniqueObjectFinder UniqueObjectFinder {
add InputArray PhotonIsolation/photons photons
add InputArray ElectronIsolation/electrons electrons
add InputArray MuonIsolation/muons muons
add InputArray JetEnergyScale/jets jets
}

module TreeWriter TreeWriter {
add Branch Delphes/allParticles Particle GenParticle
add Branch TrackMerger/tracks Track Track
add Branch Calorimeter/towers Tower Tower
add Branch HCal/eflowTracks EFlowTrack Track
add Branch ECal/eflowPhotons EFlowPhoton Tower
add Branch HCal/eflowNeutralHadrons EFlowNeutralHadron Tower
add Branch GenJetFinder/jets GenJet Jet
add Branch GenMissingET/momentum GenMissingET MissingET
add Branch UniqueObjectFinder/jets Jet Jet
add Branch UniqueObjectFinder/electrons Electron Electron
add Branch UniqueObjectFinder/photons Photon Photon
add Branch UniqueObjectFinder/muons Muon Muon
add Branch FatJetFinder/jets FatJet Jet
add Branch MissingET/momentum MissingET MissingET
add Branch ScalarHT/energy ScalarHT ScalarHT
}
```

B All generated histograms for all processes using the final Delphes card

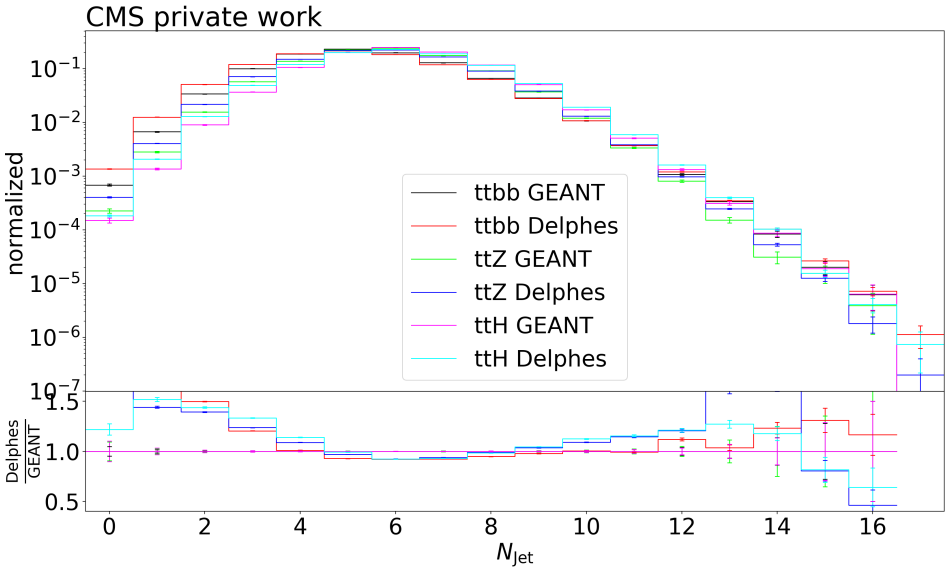

Figure B.1: Histogram of N_{Jet}.

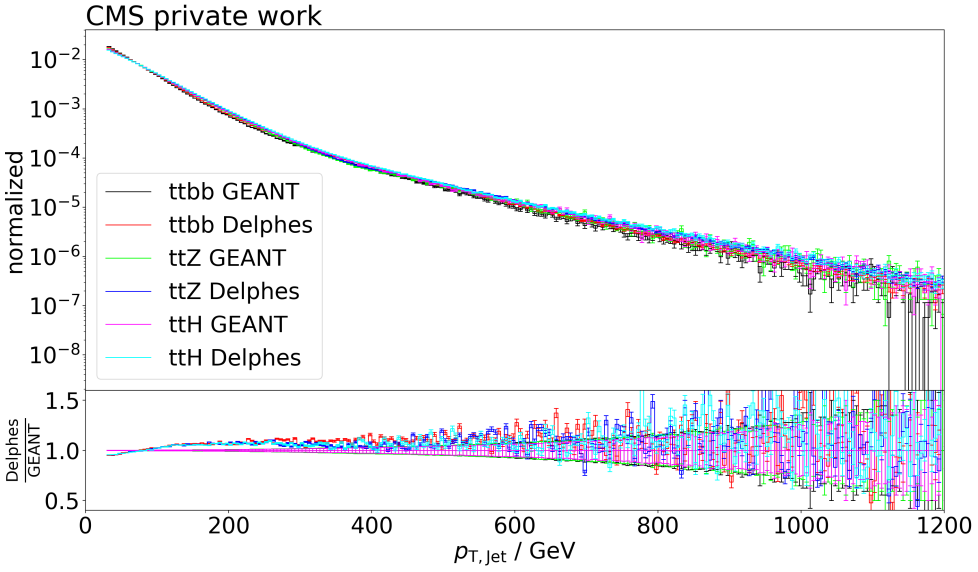

Figure B.2: Histogram of $p_{\text{T,Jet}}$.

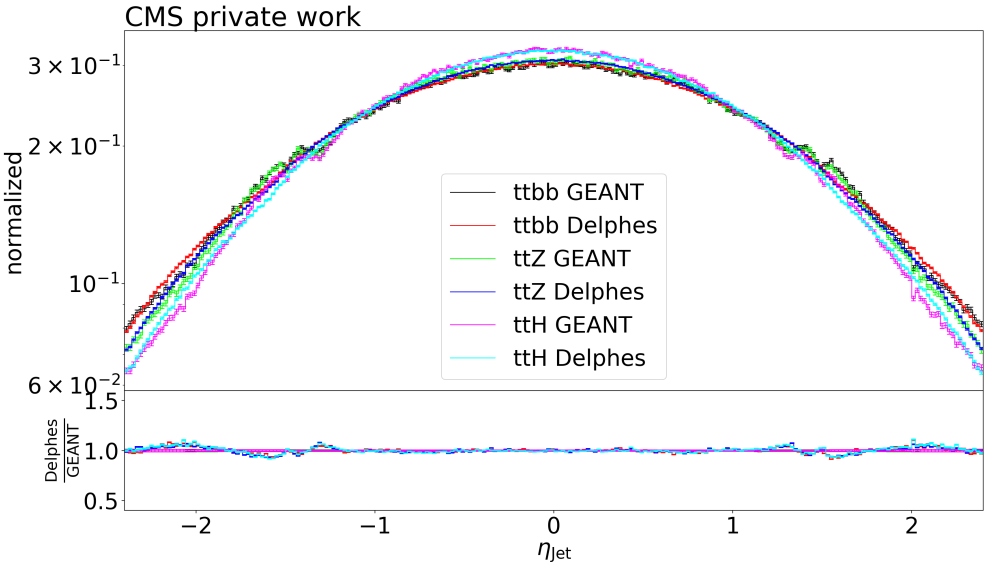

Figure B.3: Histogram of η_{Jet}.

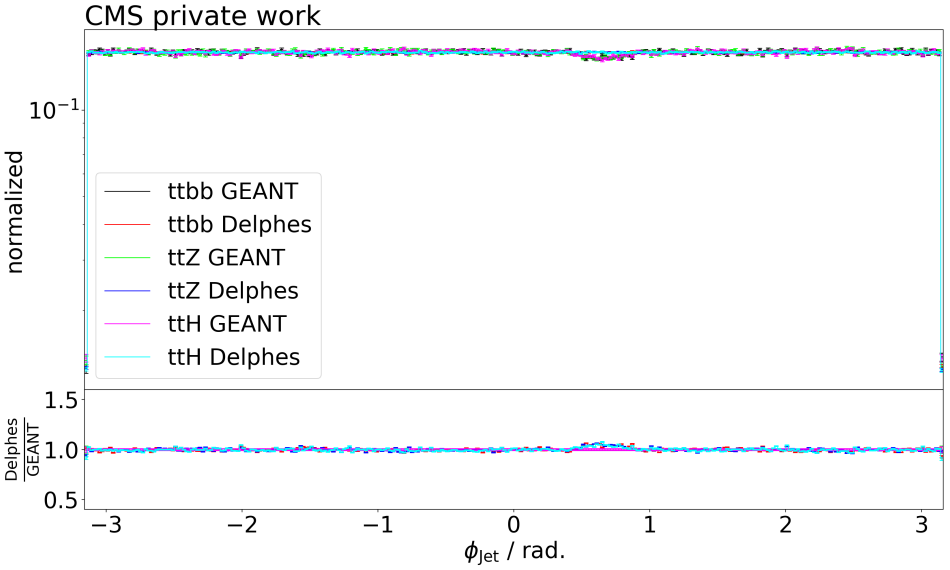

Figure B.4: Histogram of ϕ_{Jet}.

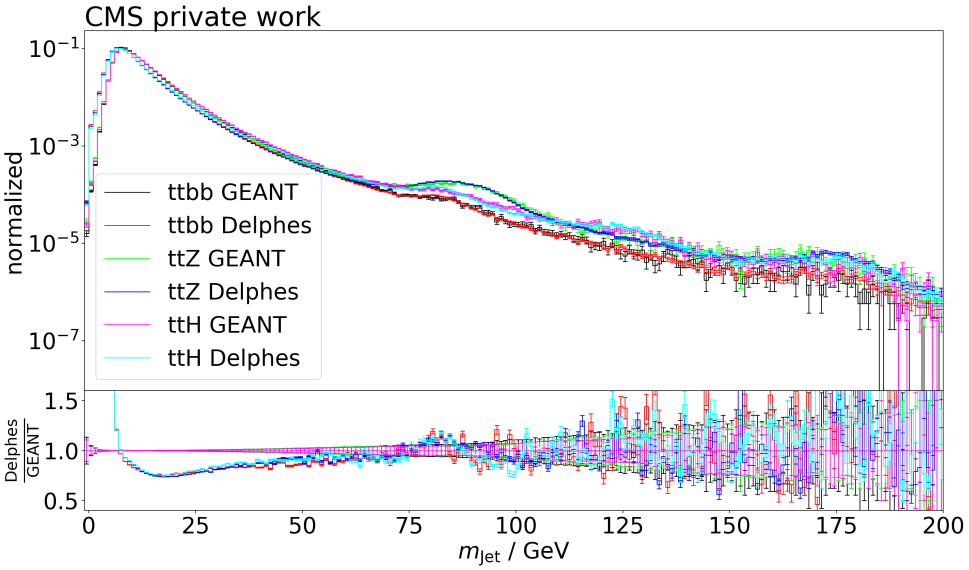

Figure B.5: Histogram of m_{Jet}.

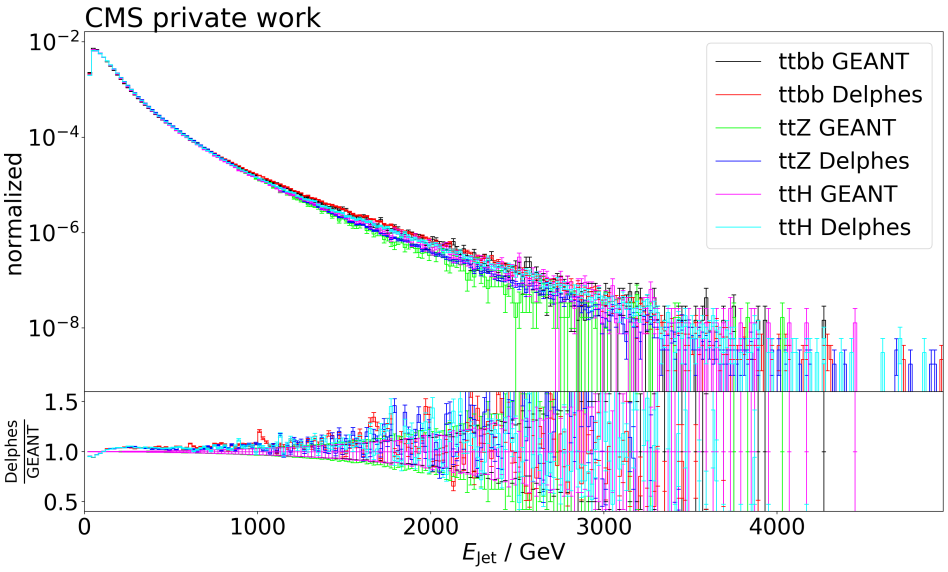

Figure B.6: Histogram of E_{Jet}.

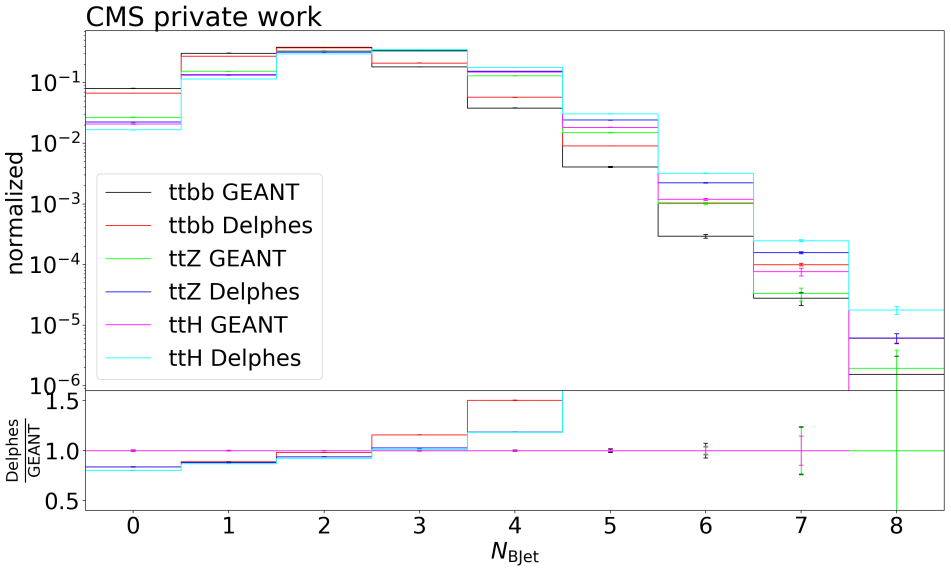

Figure B.7: Histogram of N_{BJet}.

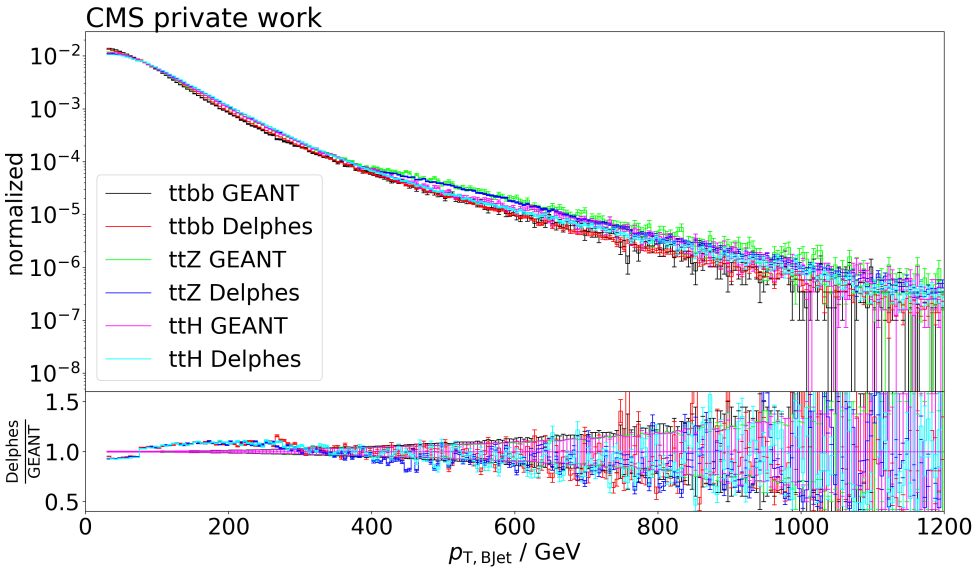

Figure B.8: Histogram of $p_{\mathrm{T,BJet}}$.

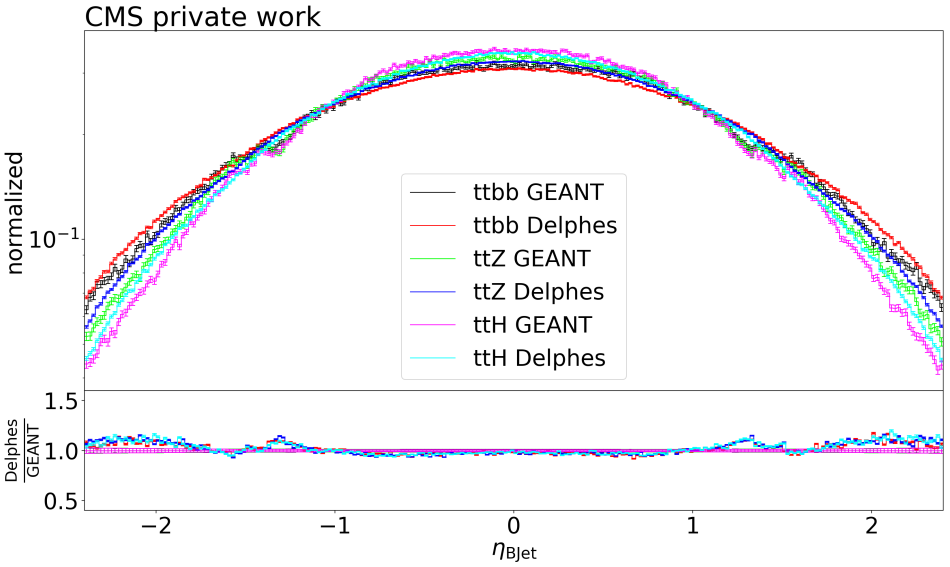

Figure B.9: Histogram of η_{BJet}.

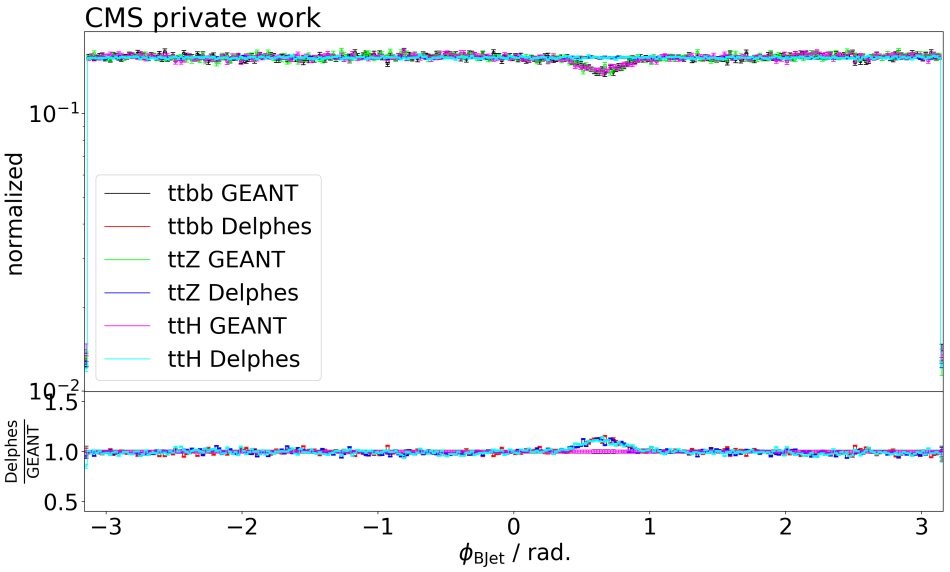

Figure B.10: Histogram of ϕ_{BJet}.

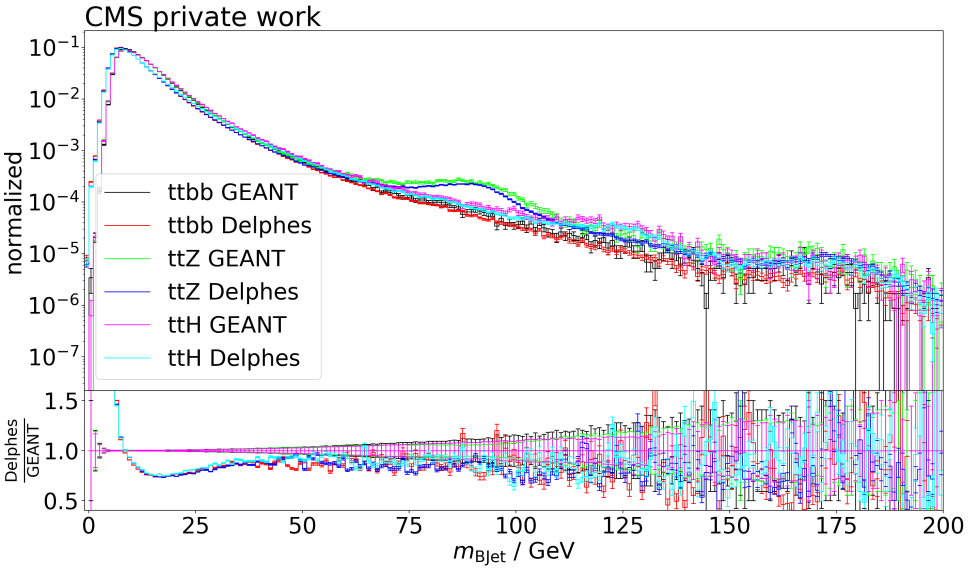

Figure B.11: Histogram of m_{BJet}.

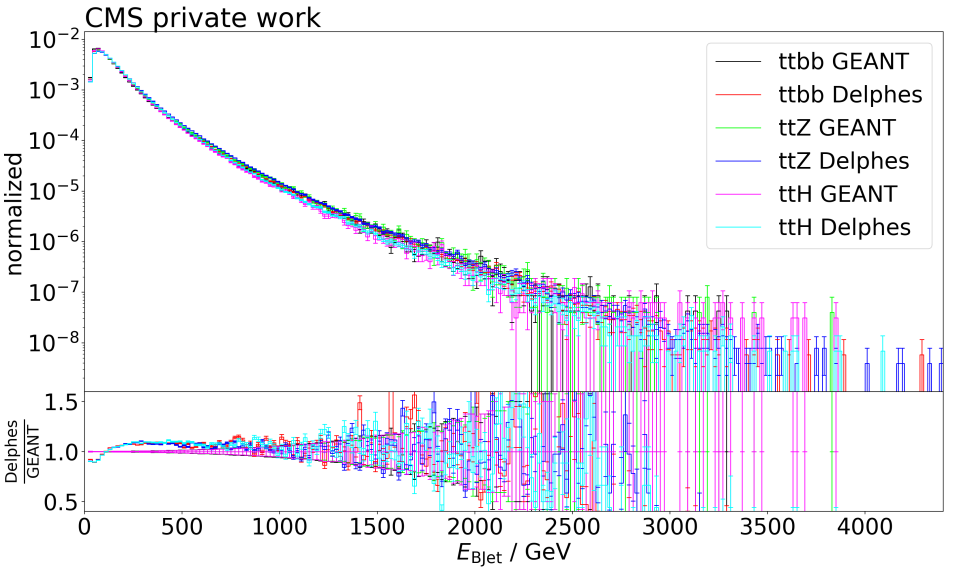

Figure B.12: Histogram of E_{BJet}.

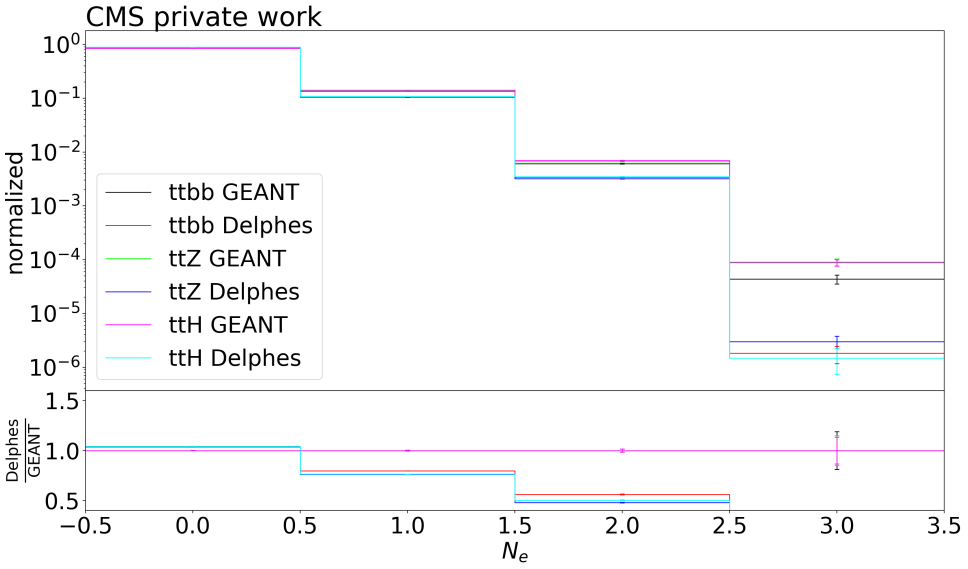

Figure B.13: Histogram of N_e.

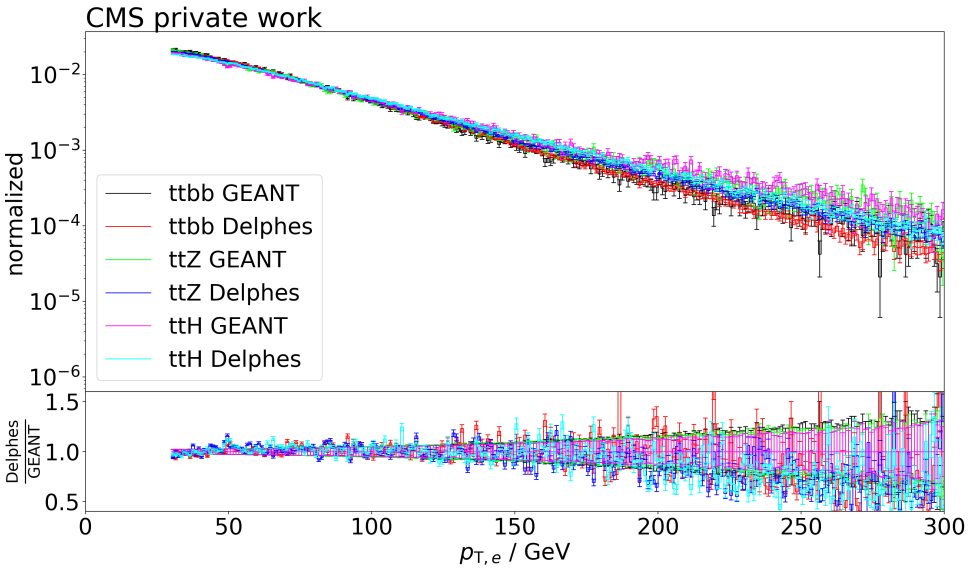

Figure B.14: Histogram of $p_{T,e}$.

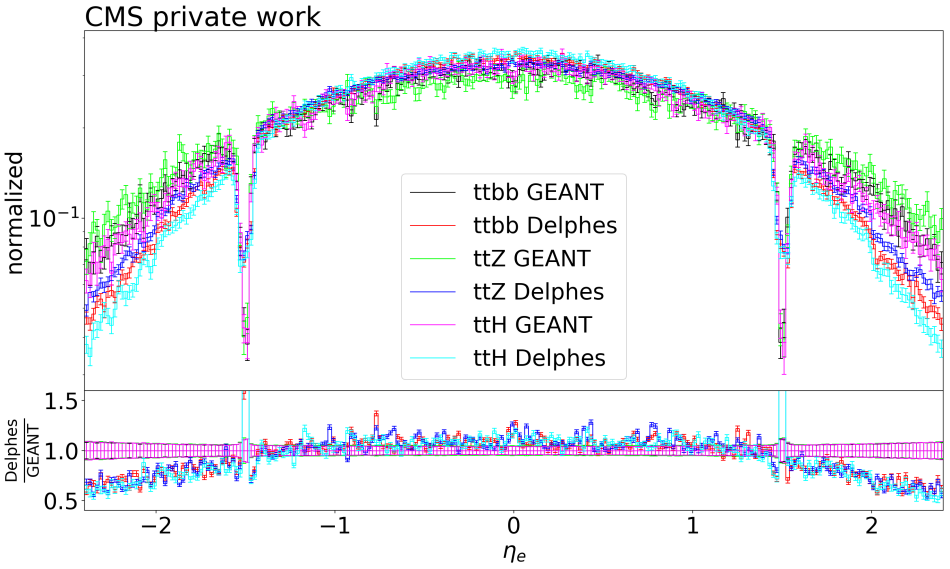

Figure B.15: Histogram of η_e.

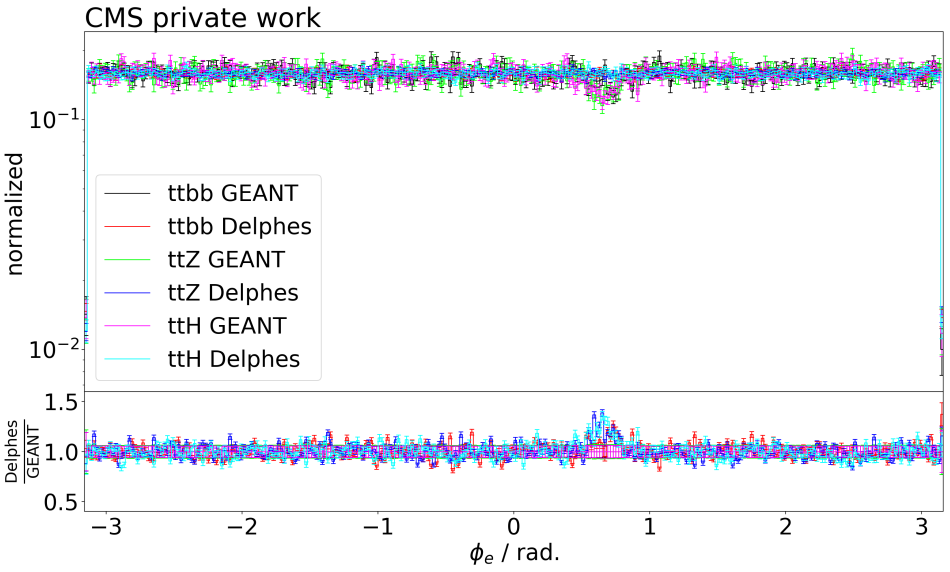

Figure B.16: Histogram of ϕ_e.

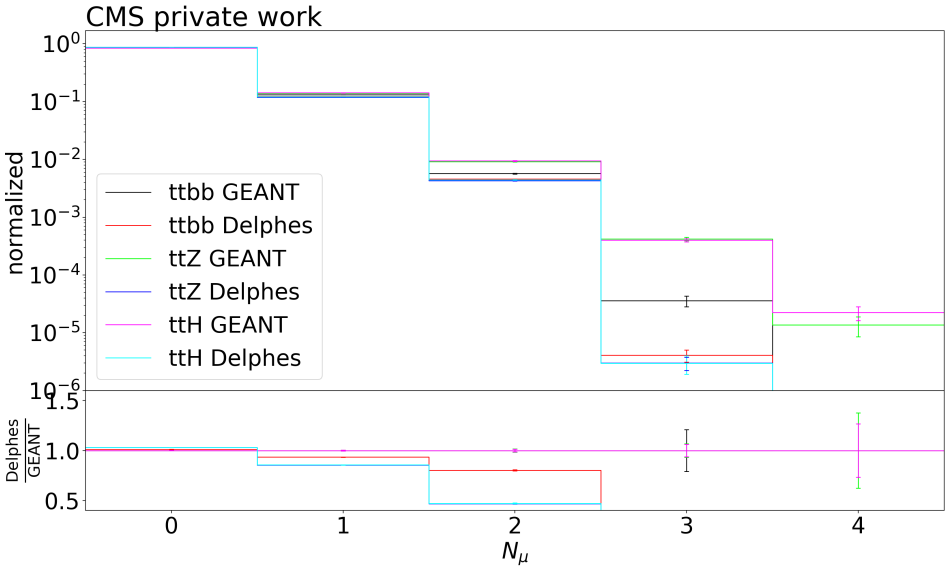

Figure B.17: Histogram of N_μ.

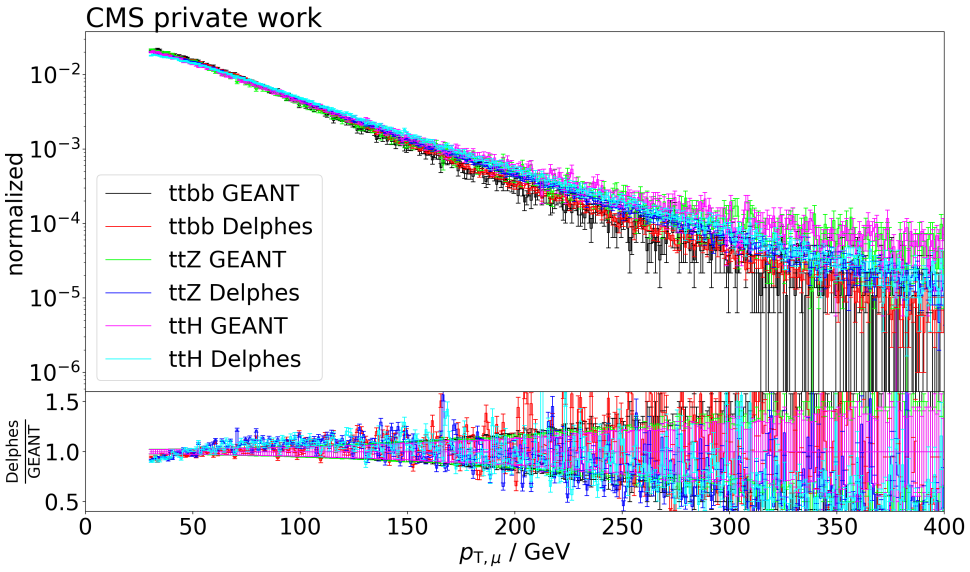

Figure B.18: Histogram of $p_{T,\mu}$.

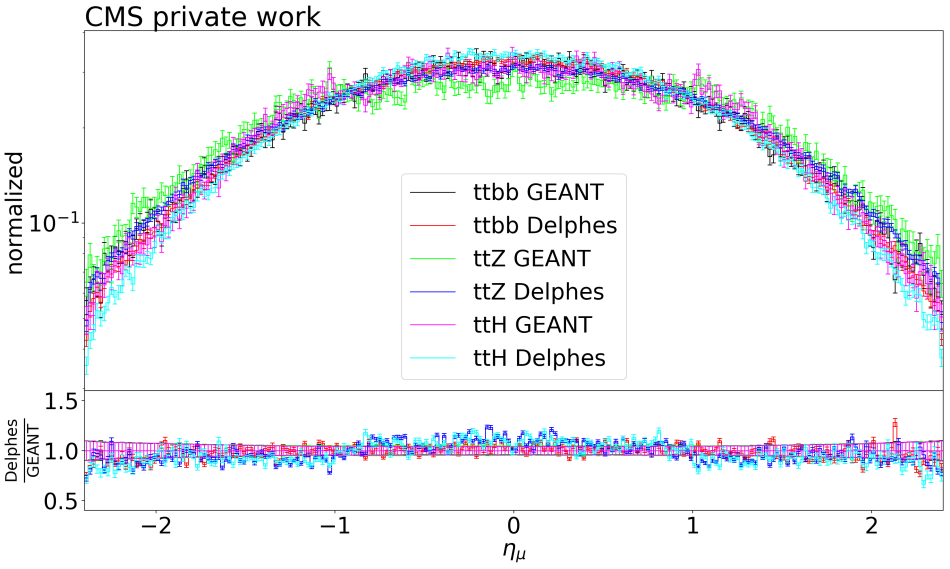

Figure B.19: Histogram of η_μ.

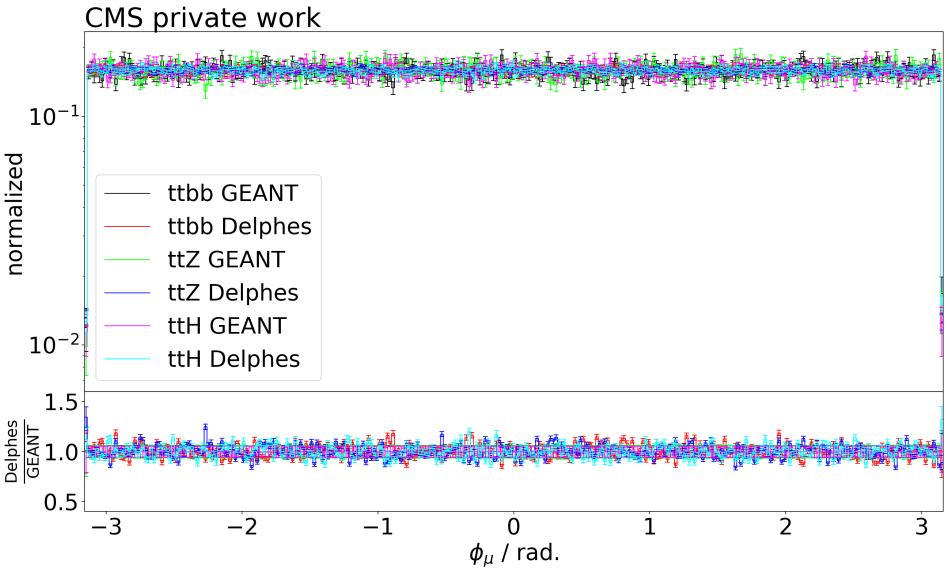

Figure B.20: Histogram of ϕ_μ.

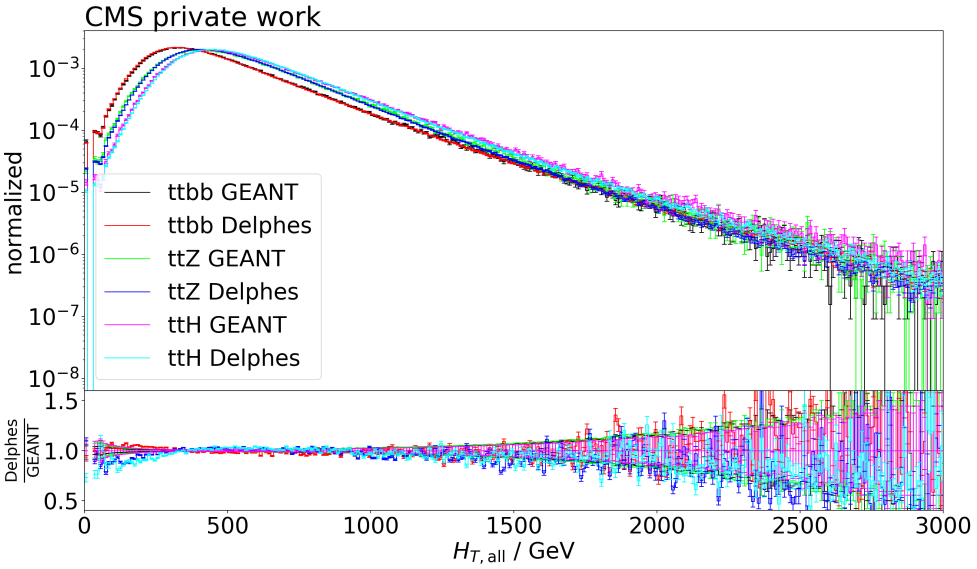

Figure B.21: Histogram of $H_{T,\mathrm{all}}$.

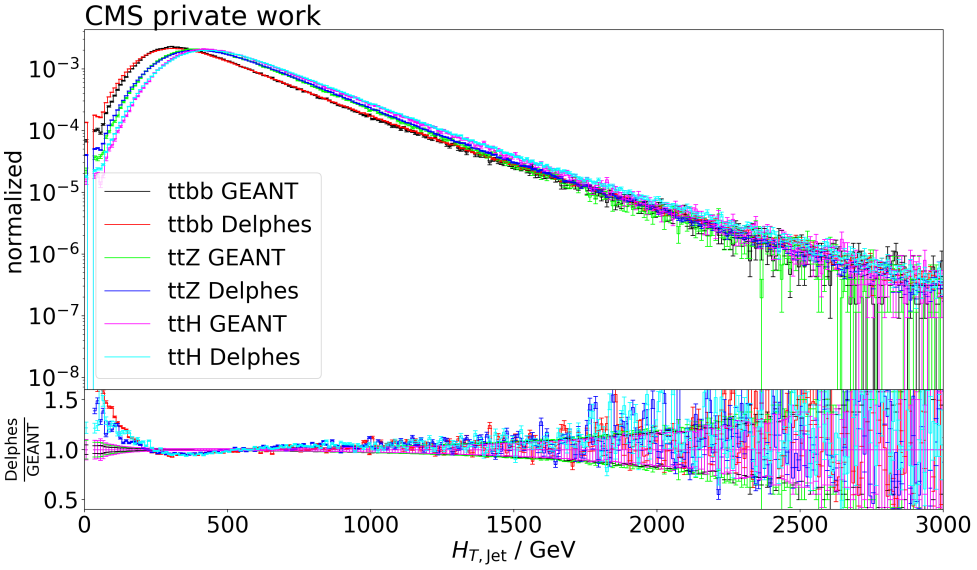

Figure B.22: Histogram of $H_{T,\mathrm{Jet}}$.

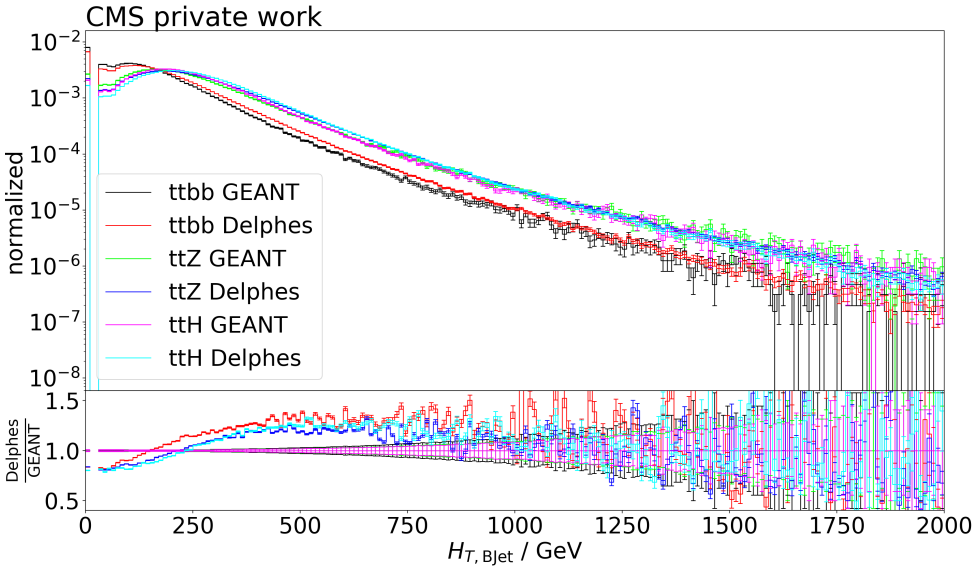

Figure B.23: Histogram of $H_{T,\mathrm{BJet}}$.

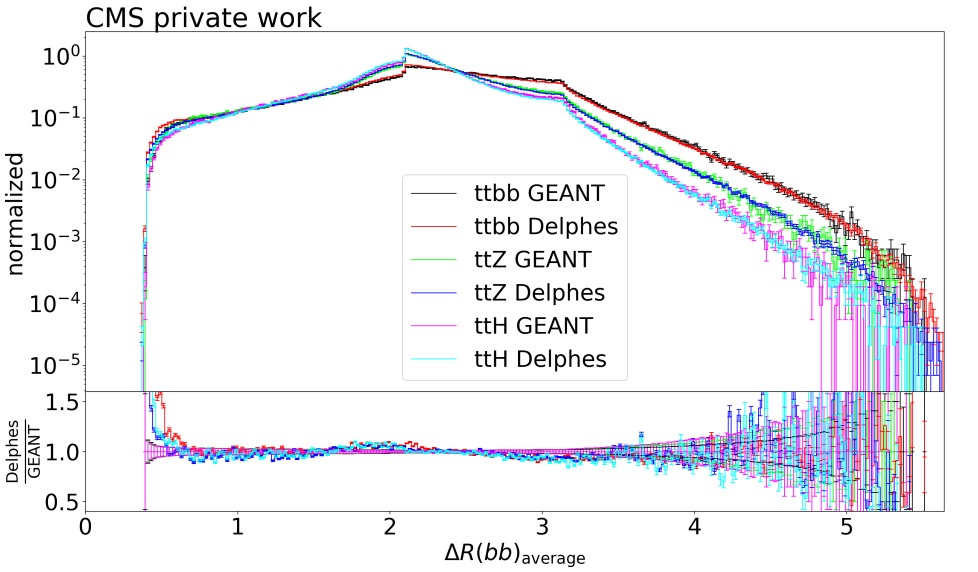

Figure B.24: Histogram of $\Delta R(bb)_{\mathrm{average}}$.

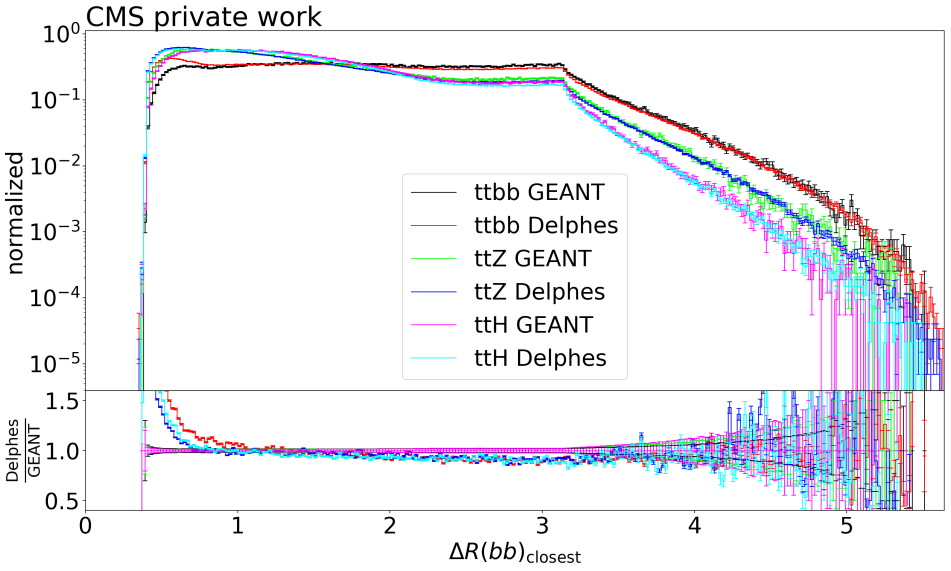

Figure B.25: Histogram of $\Delta R(bb)_{\text{closest}}$.

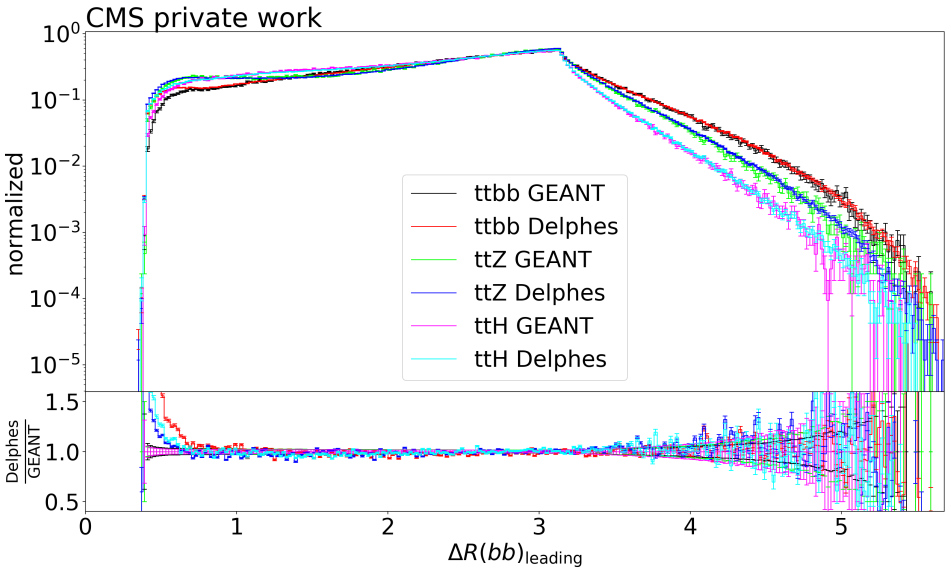

Figure B.26: Histogram of $\Delta R(bb)_{\text{leading}}$.

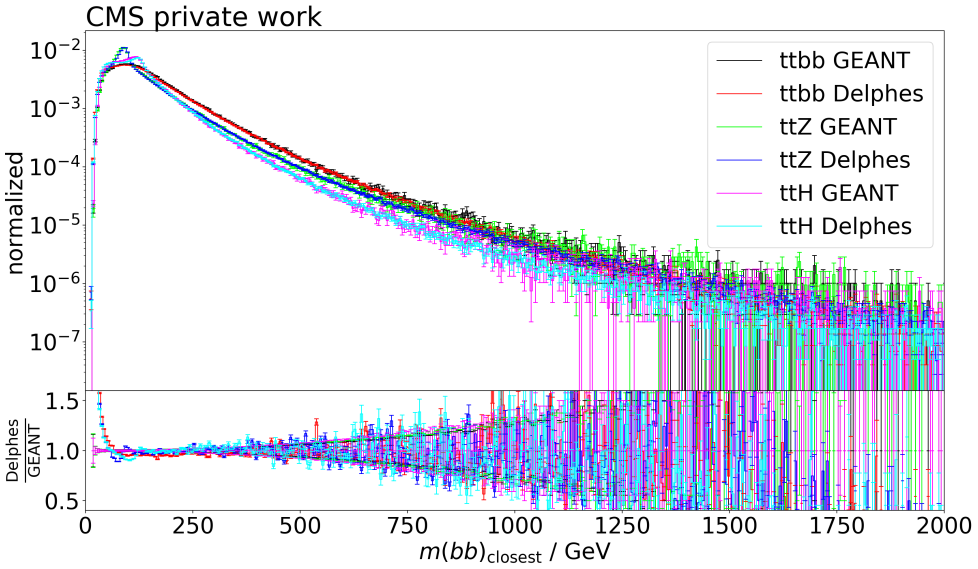

Figure B.27: Histogram of $m(bb)_{\text{closest}}$.

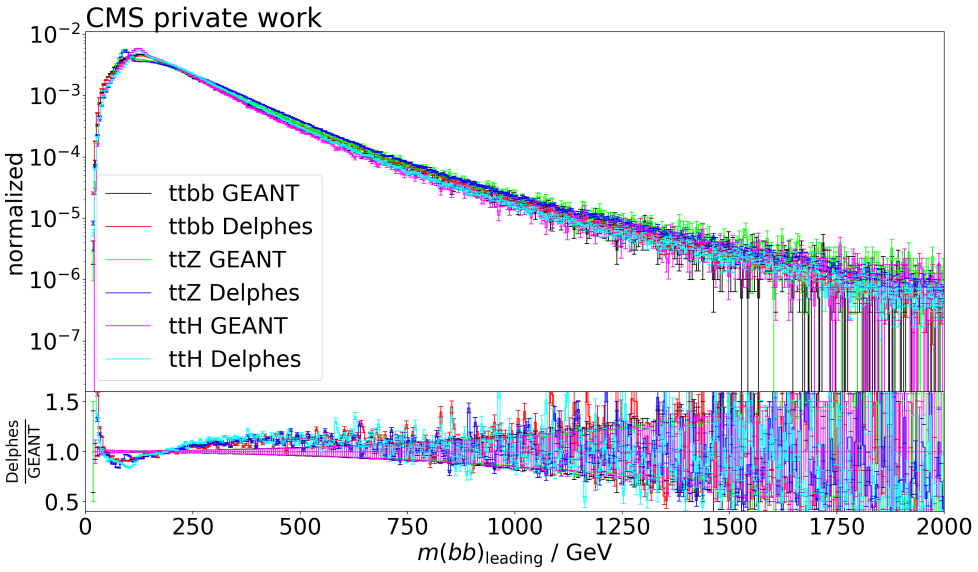

Figure B.28: Histogram of $m(bb)_{\text{leading}}$.

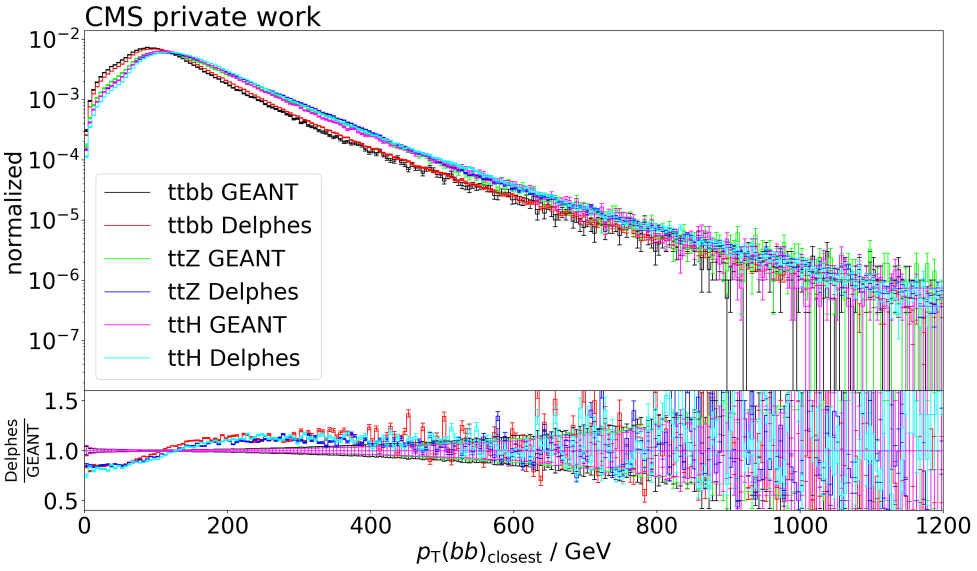

Figure B.29: Histogram of $p_T(bb)_{closest}$.

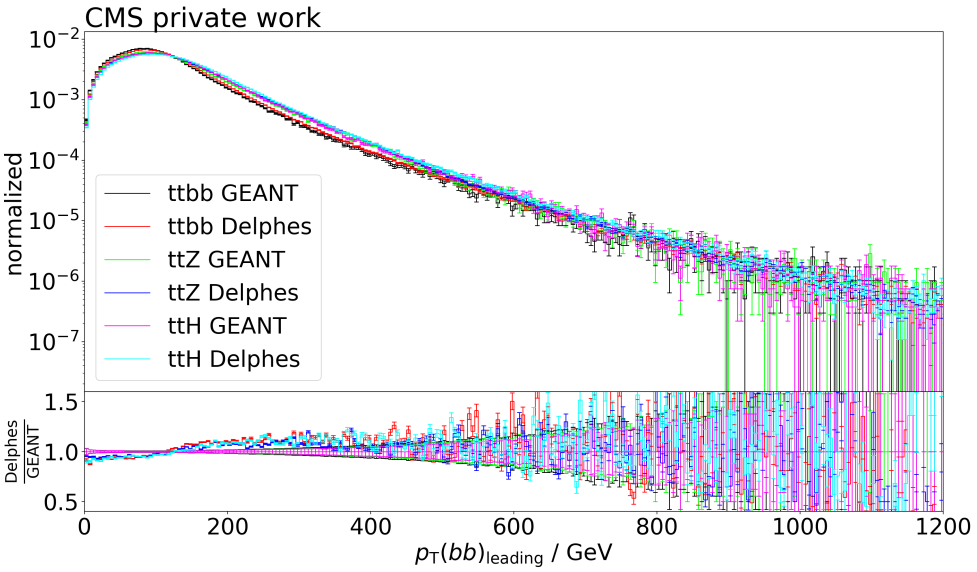

Figure B.30: Histogram of $p_T(bb)_{leading}$.

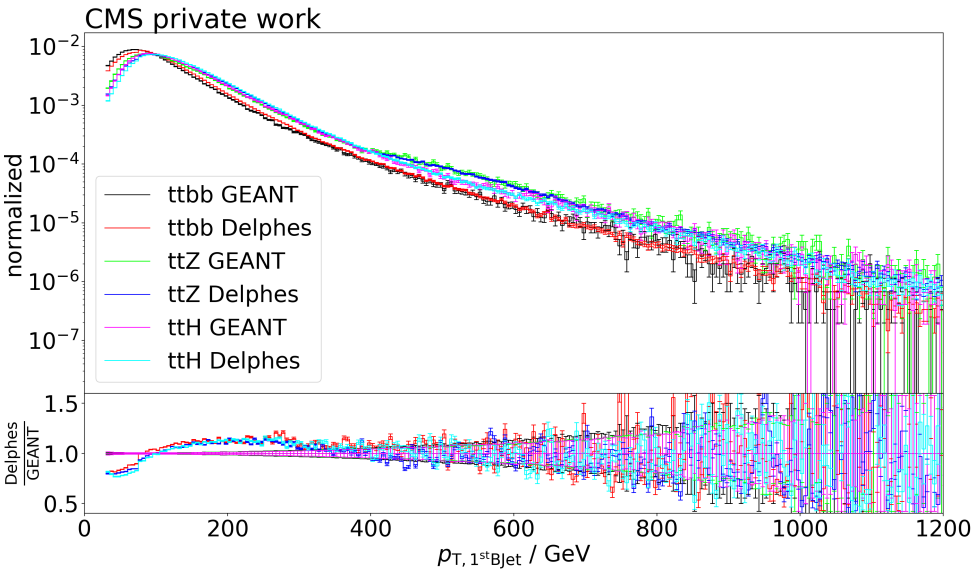

Figure B.31: Histogram of $p_{\mathrm{T},1^{\mathrm{st}}\mathrm{BJet}}$.

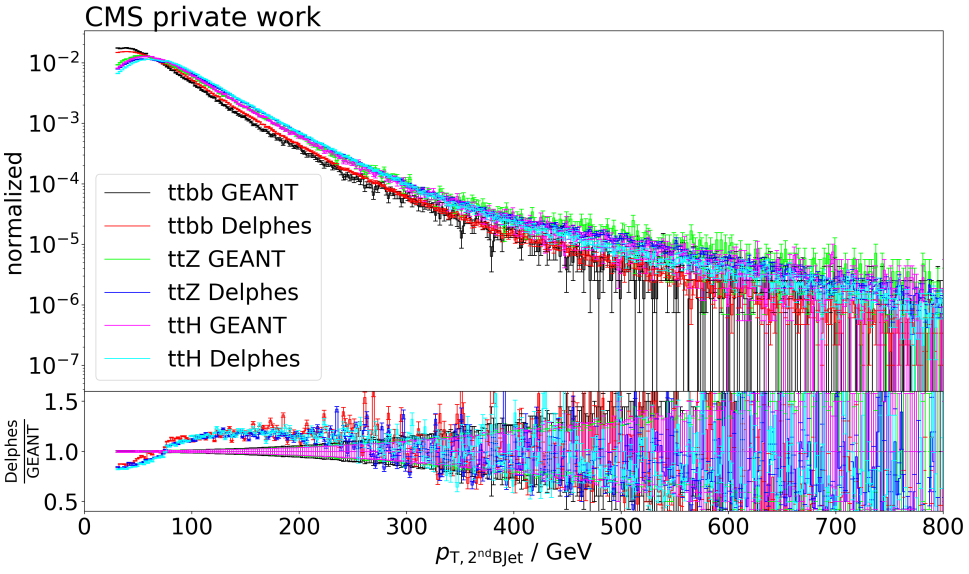

Figure B.32: Histogram of $p_{\mathrm{T},2^{\mathrm{nd}}\mathrm{BJet}}$.

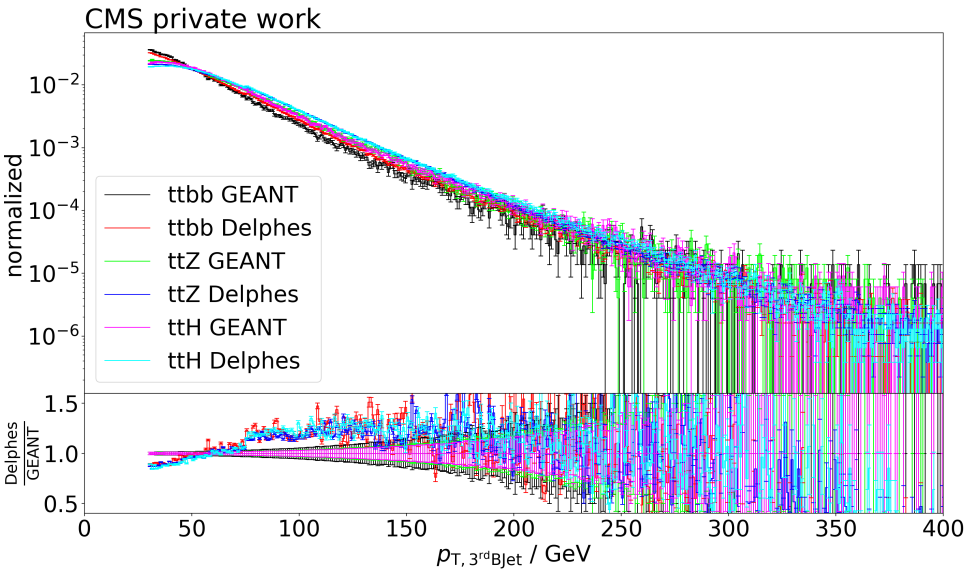

Figure B.33: Histogram of $p_{\mathrm{T},3^{\mathrm{rd}}\mathrm{BJet}}$.

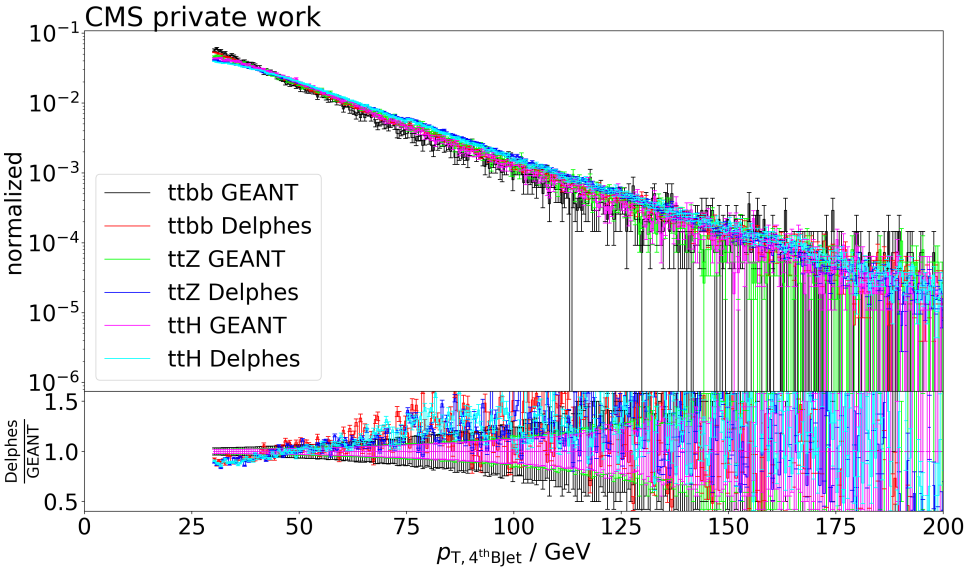

Figure B.34: Histogram of $p_{\mathrm{T},4^{\mathrm{th}}\mathrm{BJet}}$.

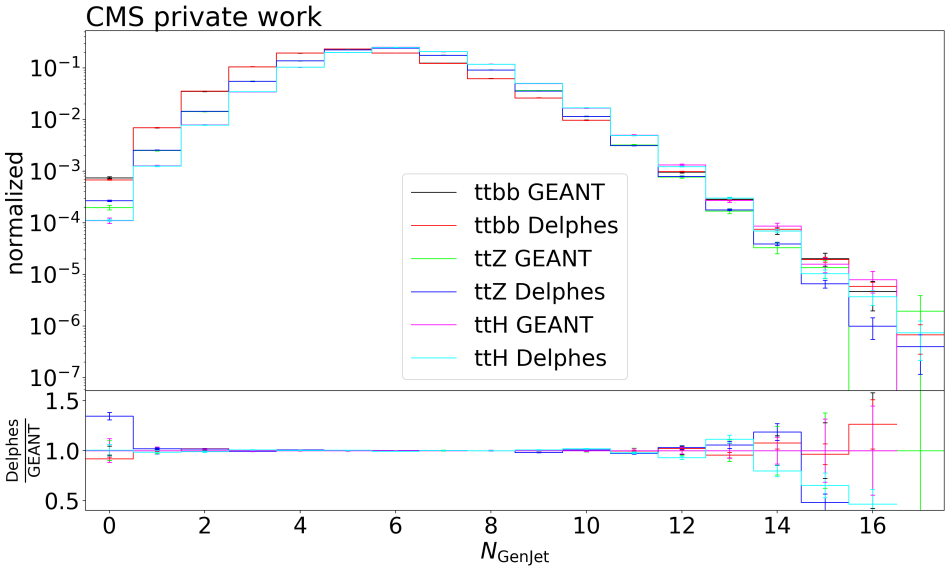

Figure B.35: Histogram of N_{Jet} on generator level.

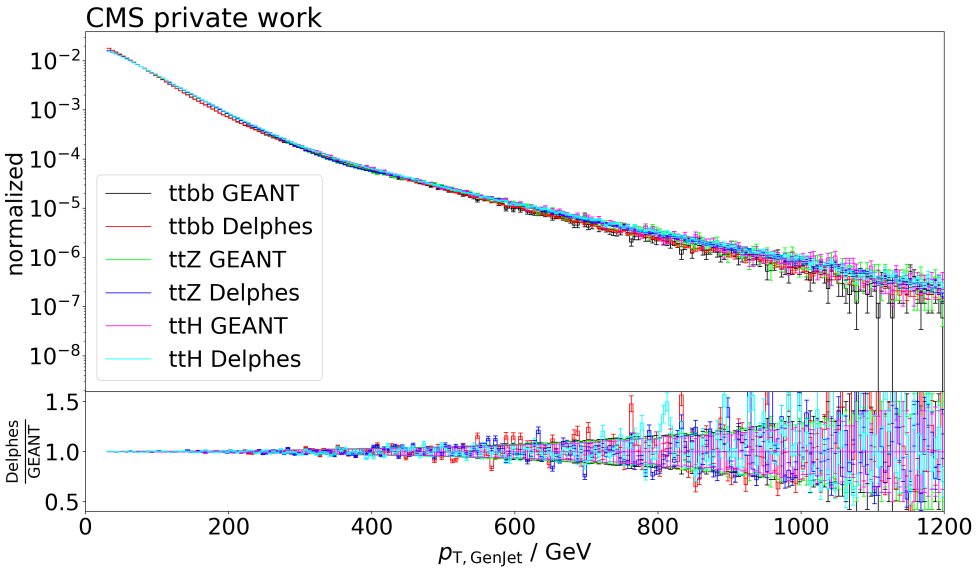

Figure B.36: Histogram of $p_{\text{T,Jet}}$ on generator level.

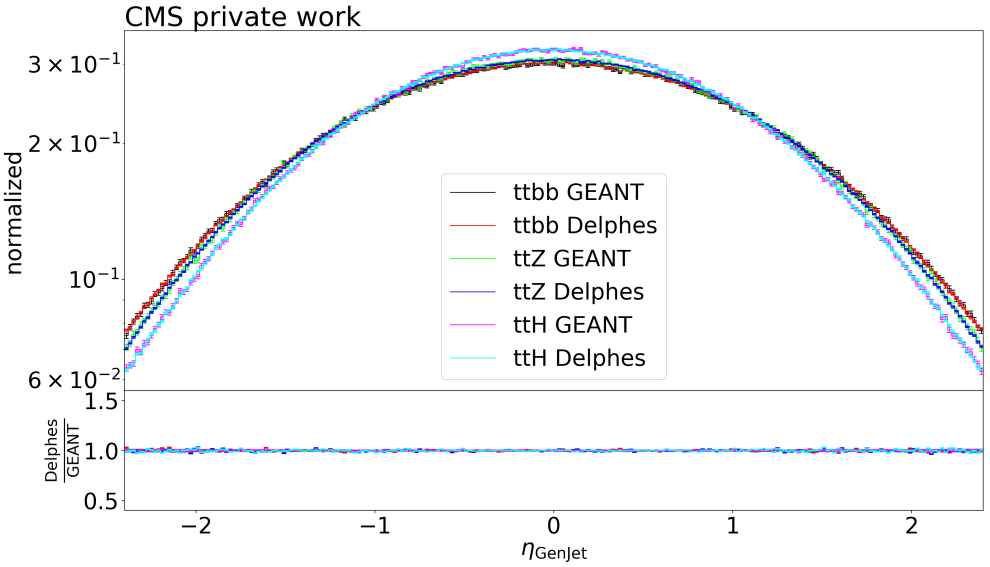

Figure B.37: Histogram of η_{Jet} on generator level.

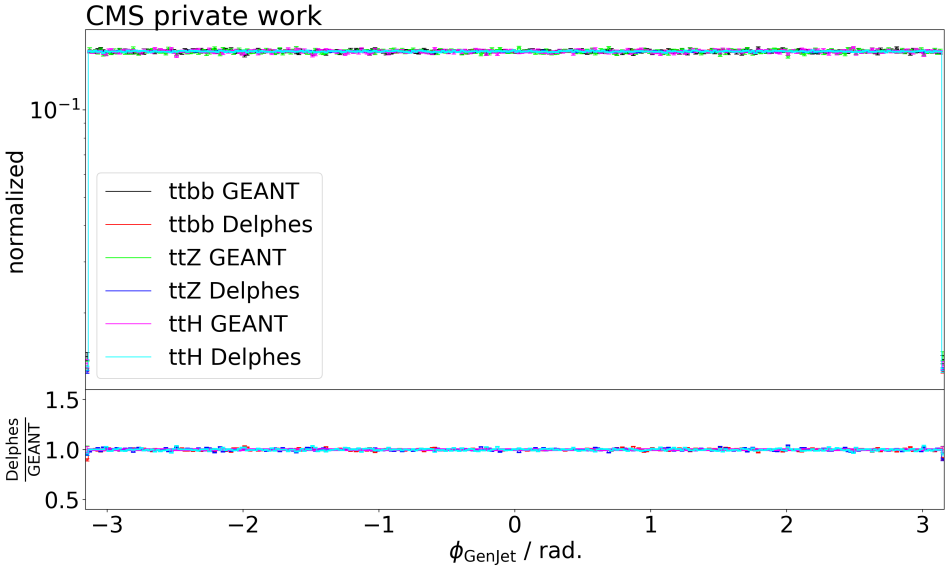

Figure B.38: Histogram of ϕ_{Jet} on generator level.

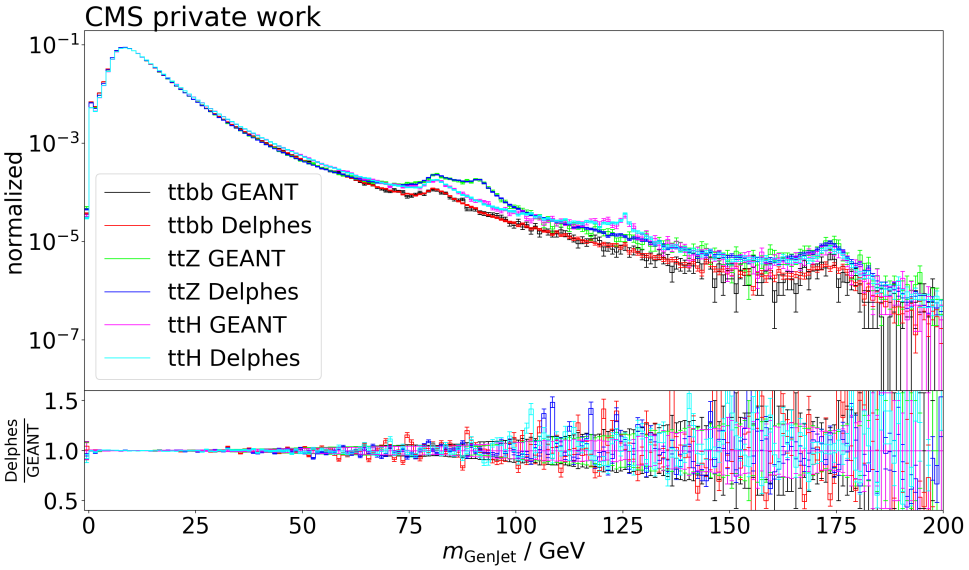

Figure B.39: Histogram of m_{Jet} on generator level.

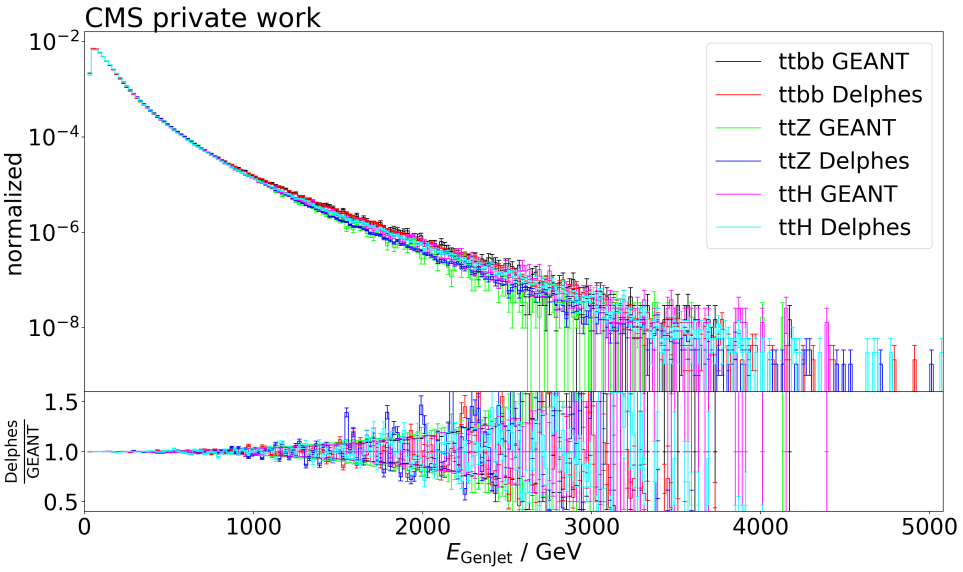

Figure B.40: Histogram of E_{Jet} on generator level.

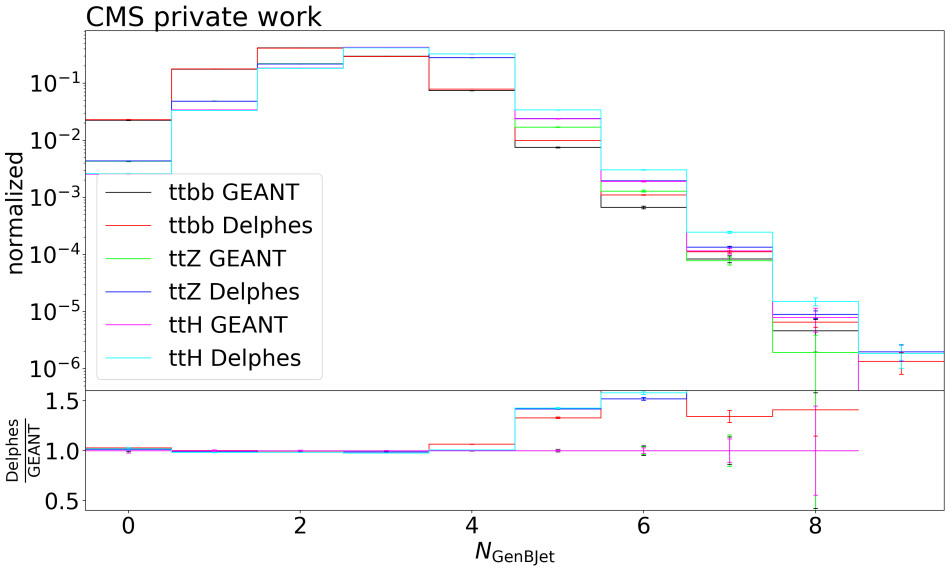

Figure B.41: Histogram of N_{BJet} on generator level.

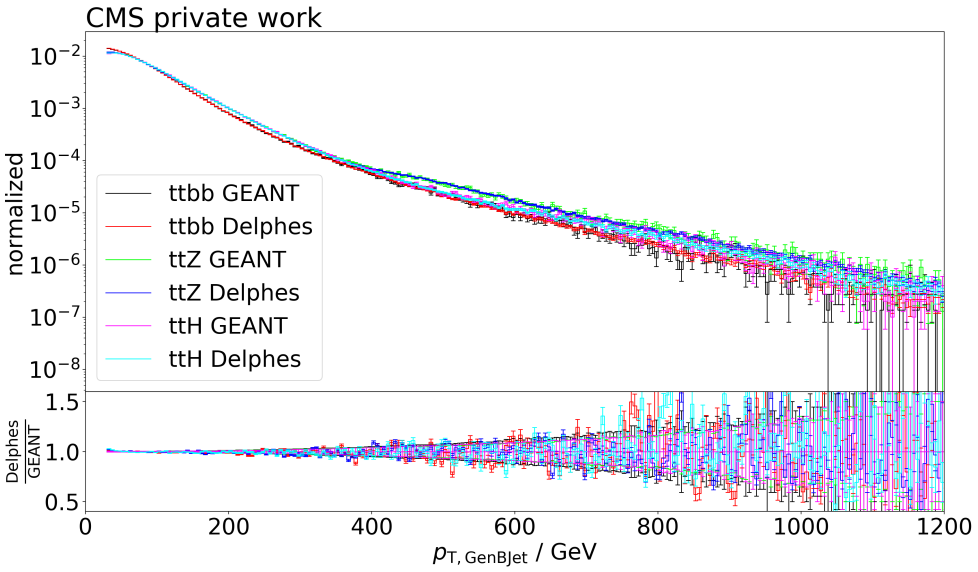

Figure B.42: Histogram of $p_{\text{T,BJet}}$ on generator level.

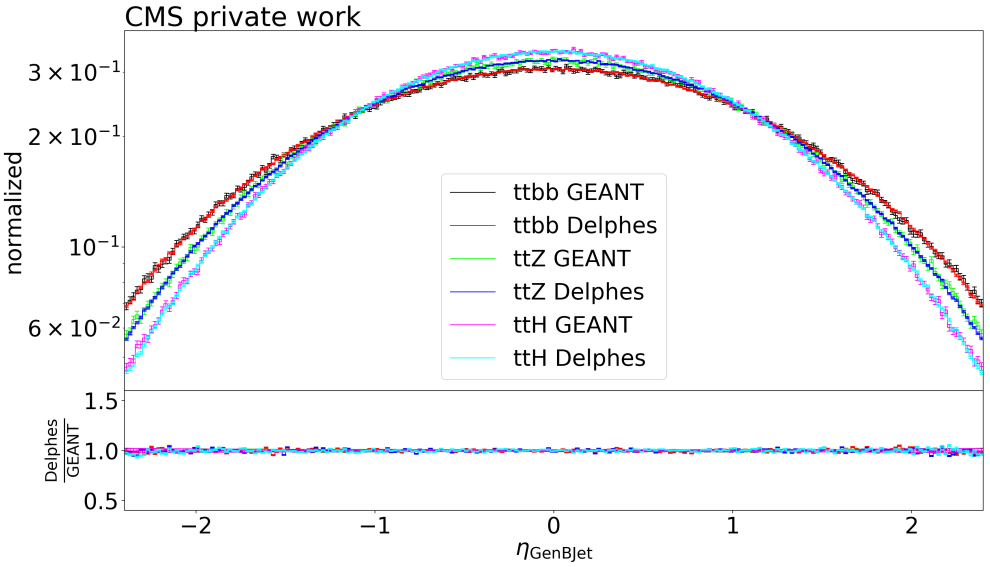

Figure B.43: Histogram of η_{BJet} on generator level.

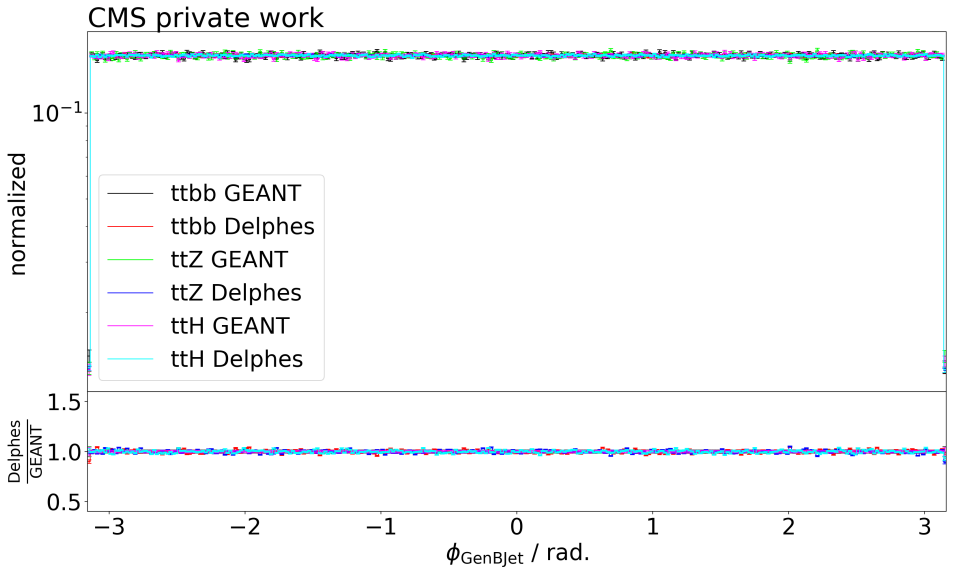

Figure B.44: Histogram of ϕ_{BJet} on generator level.

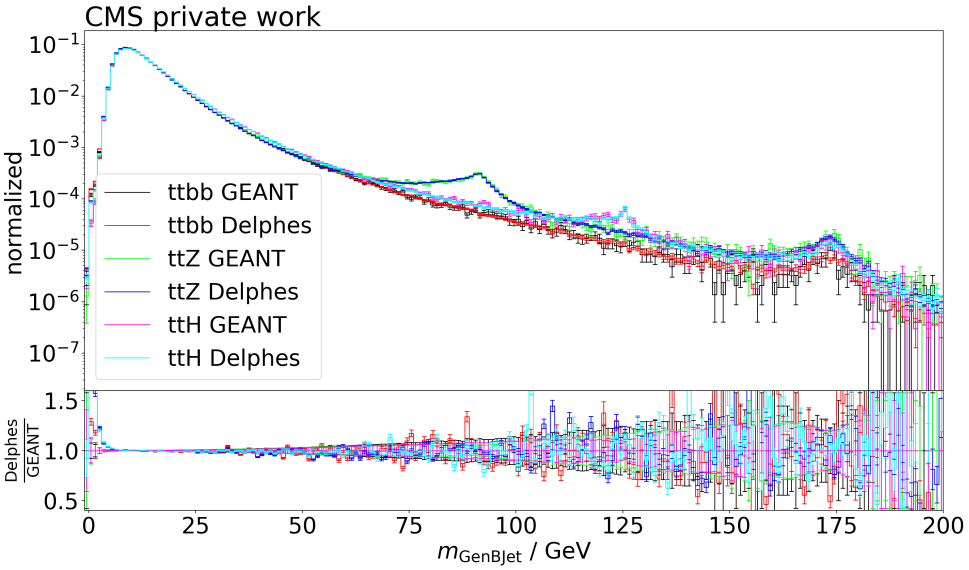

Figure B.45: Histogram of m_{BJet} on generator level.

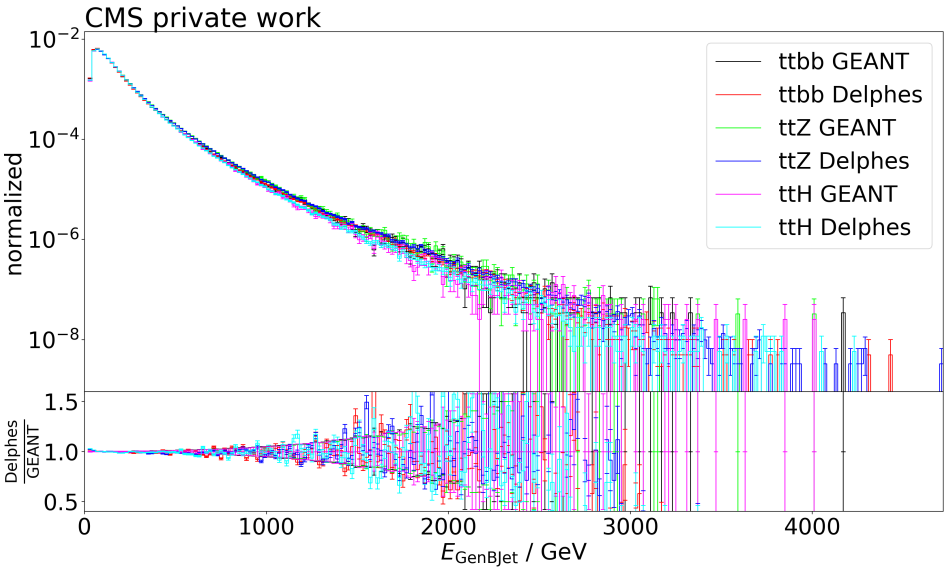

Figure B.46: Histogram of E_{BJet} on generator level.

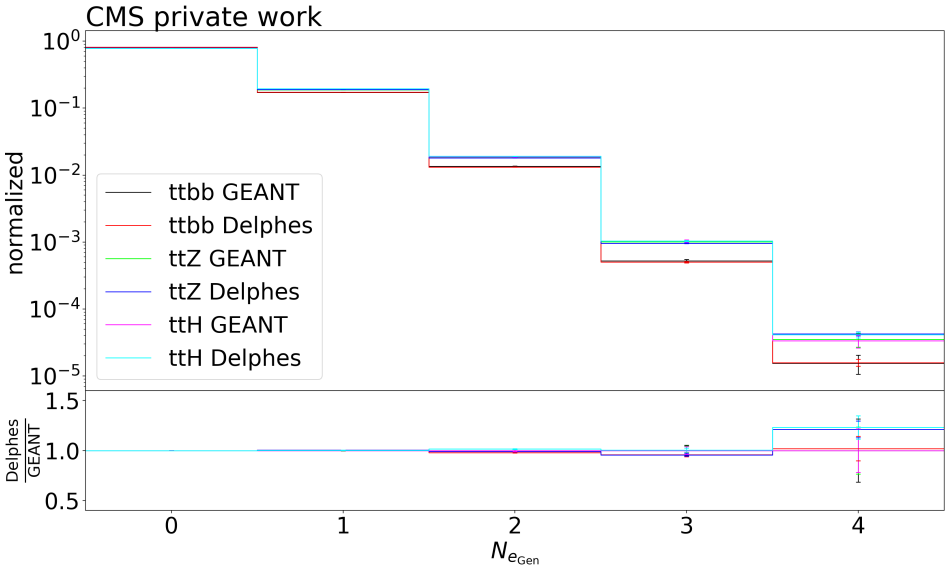

Figure B.47: Histogram of N_e on generator level.

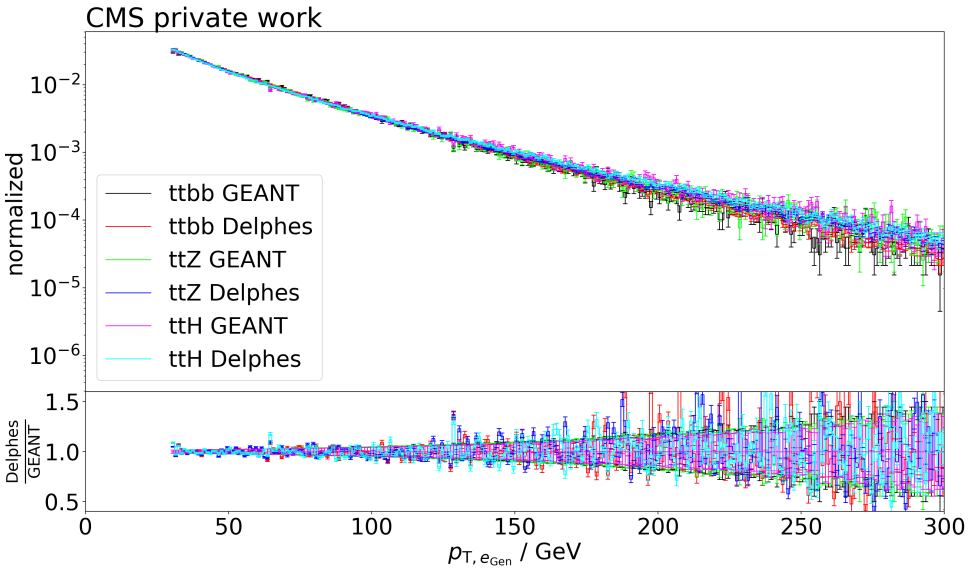

Figure B.48: Histogram of $p_{T,e}$ on generator level.

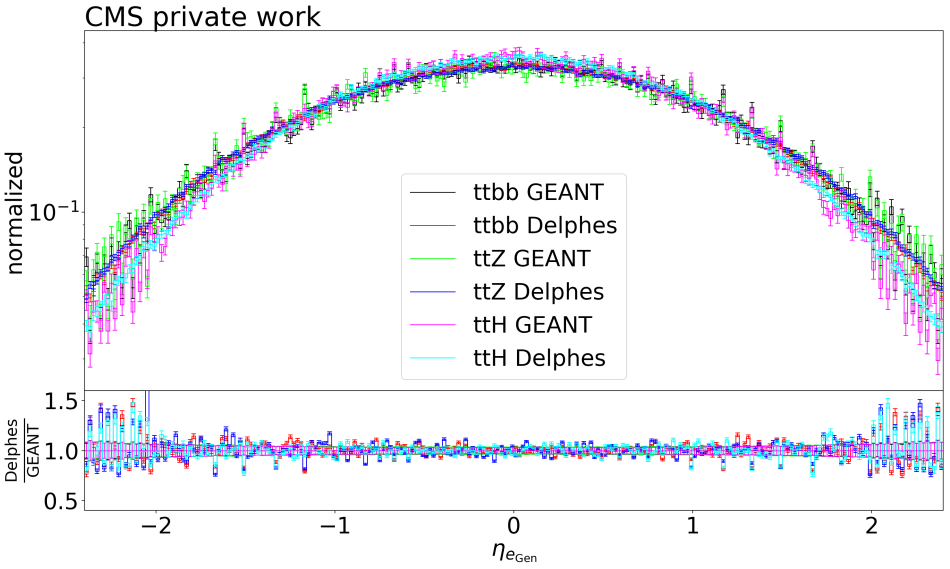

Figure B.49: Histogram of η_e on generator level.

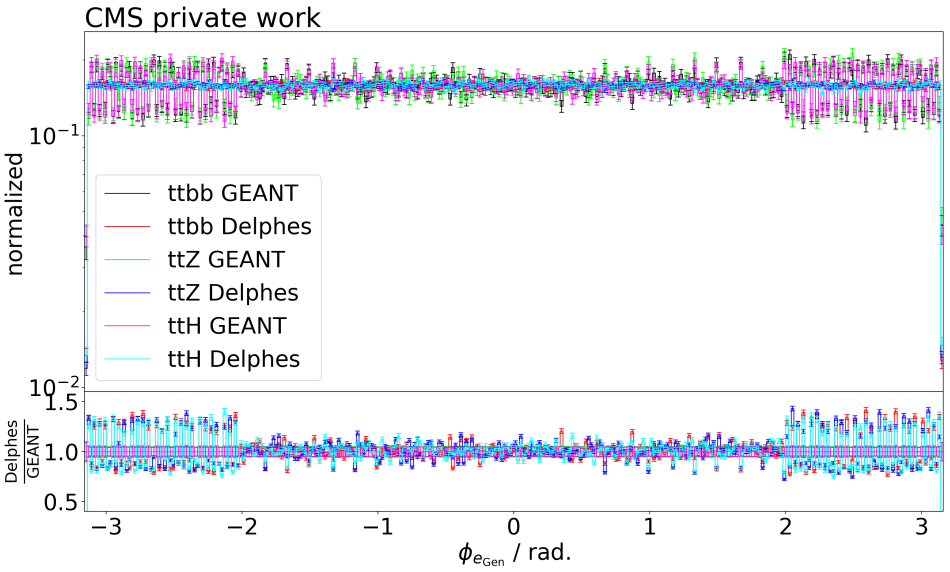

Figure B.50: Histogram of ϕ_e on generator level.

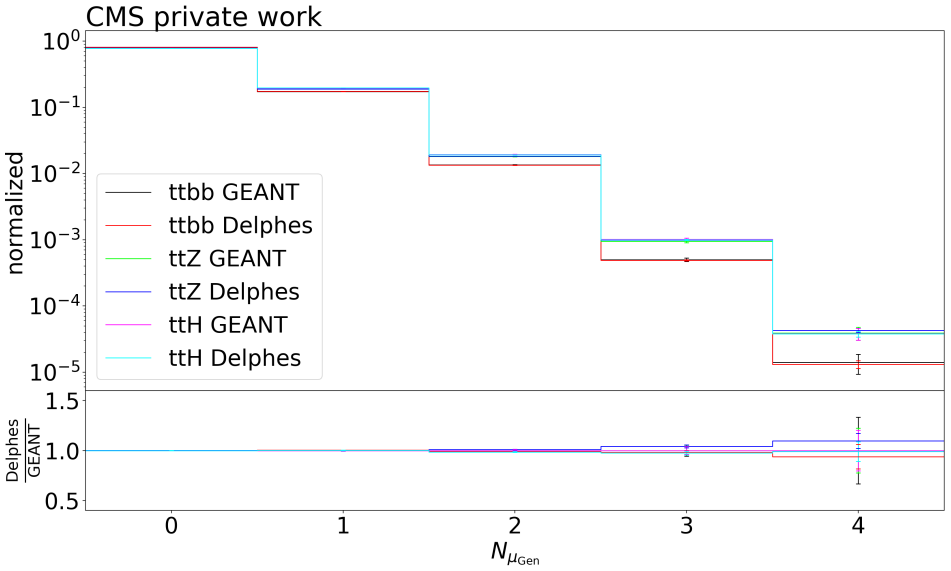

Figure B.51: Histogram of N_μ on generator level.

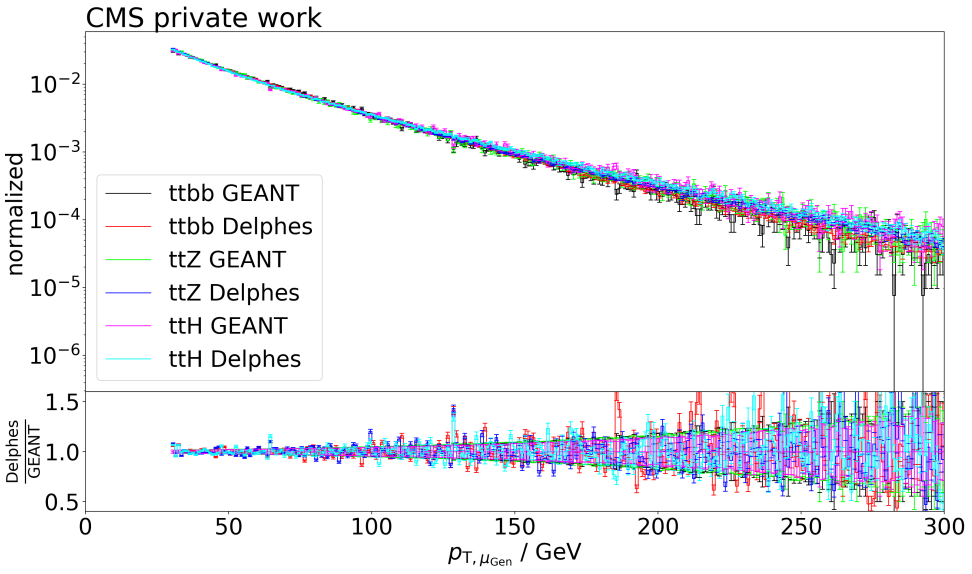

Figure B.52: Histogram of $p_{\mathrm{T},\mu}$ on generator level.

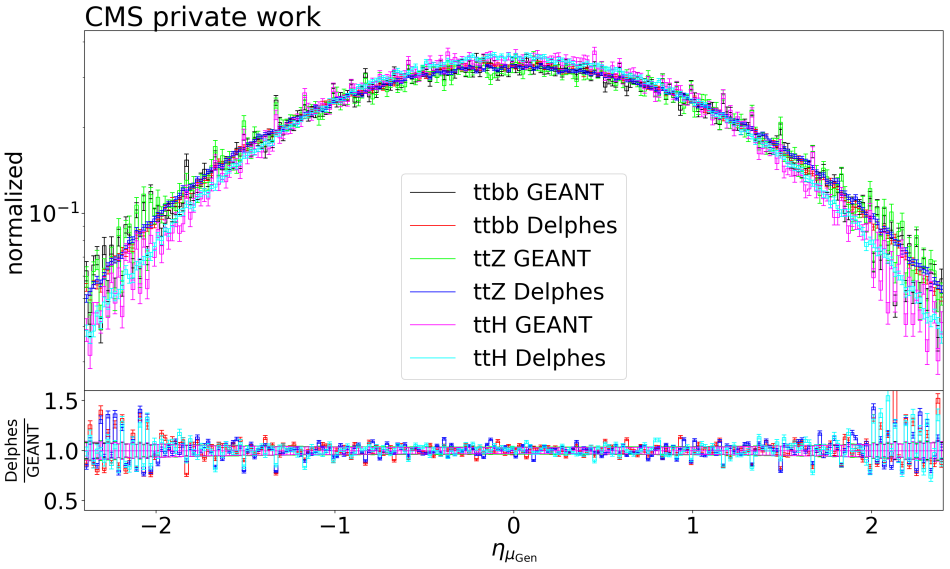

Figure B.53: Histogram of η_μ on generator level.

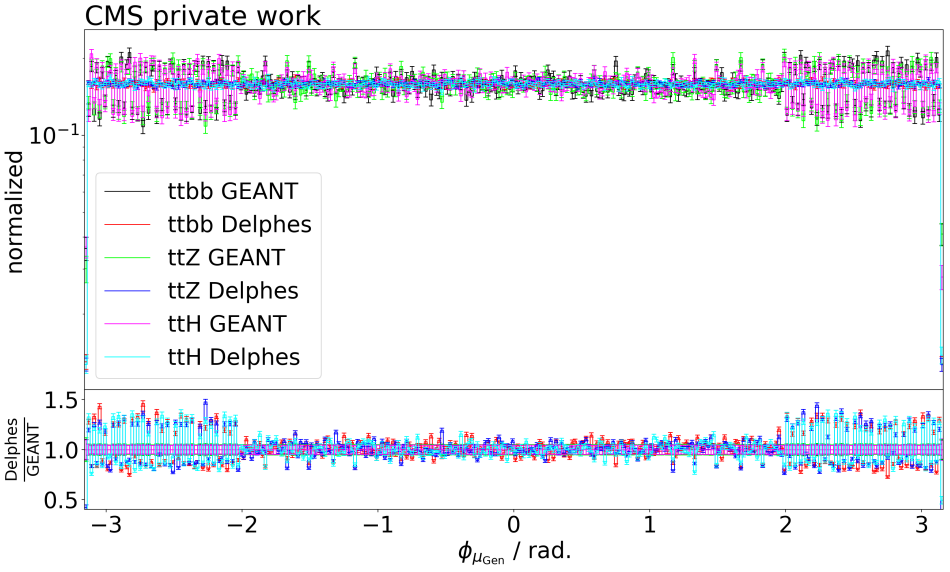

Figure B.54: Histogram of ϕ_μ on generator level.