CS 4632: Modeling and Simulation
Milestone 7: Validation and Verification

Casey Hampson (ID: 001079192)

28 April, 2025

Contents

Abstract

Event generators and simulations of proton-proton collisions in high-energy/particle physics are an integral component in the advancement of our knowledge of the fundamentals of the universe. They and the theoretical frameworks they are based on provide confirmation for what is observed in experiment and provide a means to make predictions once the framework and models are validated, providing a direction for future experiments to go in to make discoveries. The monumental discovery of the Higgs boson in 2012 at the Large Hadron Collider (LHC) in Geneva, Switzerland, was directly fueled by results from simulations. I have implemented a simplified version of such an event generator simulation, focusing only on leading order results, in which I will model two subprocesses: the hard scattering and parton showering processes. The hard scattering process centers around the Monte Carlo algorithm as a method to estimate the main cross section integral and provide a means to generate physically valid events, and the parton shower revolves around the Sudakov form factor, a means of predicting when the quark will radiate away a gluon. We find very high accuracy in the cross section calculation and strong qualitative agreement for the parton showering results with other models in the field and general physical expectations.

Introduction

In Sections 1 and 2, we present the motivation for constructing simulations of high-energy proton-proton collisions, as well as the required theory for the physics behind the collision and the mathematical tools with which the process is simulated. In Section 3 we highlight the main implementation details for the model, including the programming language of choice, UML diagrams, and essential compilation/running instructions. In Section 4 we present results from multiple runs of our simulation model. In Section 5 we do a sensitivity and scenario analysis to capture the solidity and stability of the model, and in Section 6 this is elaborated on further by conducting unit tests on the various components of the model and comparing previously presented results with those from other models. Lastly, in Section 8 we briefly summarize the key findings/results from the model and highlight improvements and next steps.

Disclaimer

It has been described via email and a number of times in previous milestones, and if it is starting to get annoying, I apologize, I just want to ensure that the unique circumstances surrounding my model are made clear and understood, because I have put a lot of work into this project, and I do not want to get points taken off simply for having a non-traditional model. This section will be almost identical to that from the previous assignment, Milestone 7, verification and validation, but with a few better explanations (I hope).

The point I want to get across about my model is that it is not similar to ordinary models like traffic models, economical models, etc. In particular, those types of models are meant to describe a wide variety of possible input/output conditions with specific ties to real-world events. For the traffic model, there are a number of input parameters to change, such as the number of cars, number of lanes in the street, speed limit, etc., and all possible variations (within reason) can relate directly to a real world possibility. Similarly, around the world and throughout time, countries' economies have varied extremely, so economical models have a high degree of freedom regarding ties to the real world.

My model, of course, is still directly tied to the real world. However, there is significantly less freedom in my choices of input parameters, scenarios, and so on, because the theory that governs proton-proton collisions (and quantum mechanics in general) is extremely tight and rigorous. One mighty popular result with which you may be familiar is the Heisenberg Uncertainty Principle, which essentially states that we cannot know both the position and the velocity of a particle at the same instant. This, and the dozens of theorems that I cannot even hope to name, highly restrict scenarios and situations which may occur in real life, and therefore limit the real-world ties present in my model.

Because of this, the latter milestones and latter sections of this project may be limited in their scope. In the context of my model, there is either: I specify particular input parameters that are accepted by the theory, and as such, get results that match what would happen in the real world, or I can specify different parameters which may break assumptions or violate the theory, in which case the results should be entirely ignored, as they not only don't represent what happens in real life, but may be simply mathematically invalid.

To reiterate, though, my model is absolutely still representative of a real world process, and as I will describe in Section 1, there are strong and relevent motivations for making such models. It's just that, to make an analogy with traffic systems, instead of modeling generic traffic patters that could be valid anywhere, I am focusing specifically on a single intersection in one part of the world, and trying to consider all possible factors related to that specific intersection, which may be more limited than a generic traffic model.

1 Background

1.1 Proton-Proton Collisions

The goal of high-energy/particle physics is to try and understand how the universe works on a fundamental scale. One of the main ways we do this is by accelerating protons to near the speed of light, and colliding them together. This is done at CERN in Geneva, Switzerland, in what is called the Large Hadron Collider (LHC). The purpose of doing these extremely energetic collisions is to try and analyze the thousands of particles that fly out in all different directions and their subsequent decays and interactions to try and gauge

what exactly happened in the collision. By analyzing the final-state stable particles, we can reconstruct various quantities and make plots and other predictions. As an example, if a heavy particle decays into two new, lighter, and more stable particles, we can analyze their energy and their paths once they reach the detector and determine (roughly) where/when the original particle decayed and how massive it was. This is the essential recipe for discovering new particles, which is one of the main goals of the LHC.

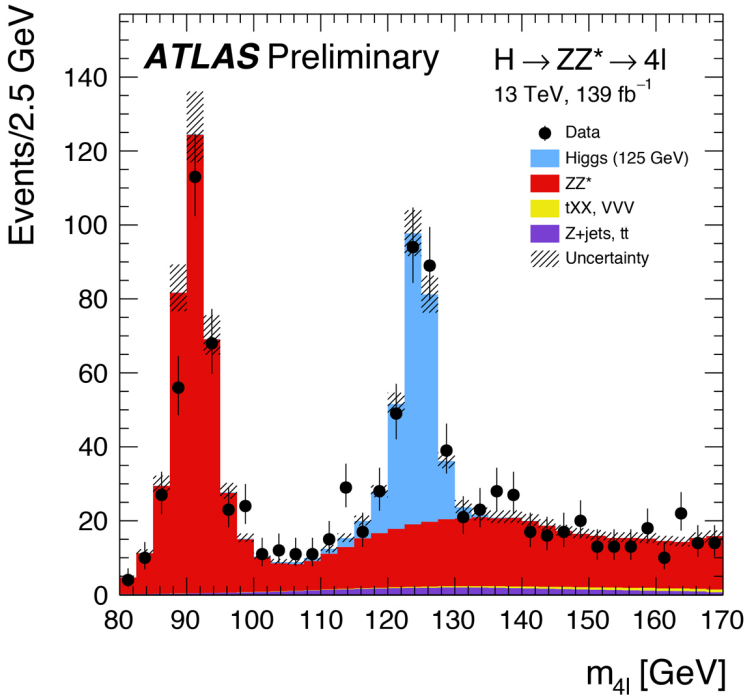

Figure 1.1: Invariant 4-lepton invariant mass of reconstructed events in the ATLAS detector for $H \to ZZ* \to 4\ell$ process, the famous plot signifying the presence of the Higgs boson.

An example of one such plot is given in Fig. 1.1. A simulation (rather, a ridiculously large number of them) was carried out that modeled the creation of a Higgs boson, its decay into a two Z bosons, and their decay into four charged leptons, and the resultant distributions of the center of mass of the four leptons were recorded and plotted. The normal production of the two Z bosons is shown in red (i.e. without a Higgs boson), and the process involving the Higgs is shown in blue. This comparison between the *signal*, the process in question, and the *background*, the expected results from the process without the new particle, is what illuminates new particles. The mass at the blue peak in this plot is equal to the mass of the Higgs boson, and the statistical uncertaintly is good enough to conclude, beyond a shadow of a doubt, that this particle exists. This particular plot came from more recent studies into "cleaner" Higgs production processes, but a similar-looking distribution was found back in 2012, and similar, real detector data was produced shortly afterwards that confirmed the existance of the Higgs boson. See Ref. [1] for more information.

Evidently, it is extremely helpful to have a theoretical framework that is able to match what is seen in the detectors so that we can make predictions with that framework. This calls for the usage of complex simulation programs that model the entire chain of events starting from the "hard scattering" process, which is the initial collision of the protons, all the way to simulating the detector structure and how it behaves when the particles from the collision fly through it. We then read the output that the simulated detector shows us, and we can analyze the exact same things as done in experiment to make very accurate comparisons and predictions.

The reason that this must be *simulated*, rather than just plugged into a calculator or some mathematical subroutine, is due to the myriad considerations that must be taken into account when talking about particle interactions at such a small scale. There are a number of different forces at play, namely the strong nuclear force, the weak nuclear force, and the electromagnetic force. Additionally, the sheer number of particles

present in the system at any given time is large enough that our analytical equations simply cannot be solved by any known method without taking insanely crude approximations, which would ruin the main goal of acquiring physically sensible results.

1.2 Physics Challenges

As one would expect, though, simulating these processes is still a huge challenge. As mentioned before, it is not simply a matter of momentum/energy conservation and other simple properties; there are a multitude of quantum mechanical phenomena that occur on this energy scale, the most notable of which come from Quantum Chromodynamics (QCD), or the strong force. Essentially, the constituents of protons, which are called *quarks*, experience a force similar to that of normal electromagnetism, but opposite in some respects. For instance, bringing two electrons closer together increases the repulsion between them, since like charges repel. Similarly, bringing a proton and an electron closer together will increase the attraction between the two. Quarks, on the other hand, have the opposite property, where the closer you bring them together the less they feel like doing anything at all, and the further you bring them apart, the stronger the force between them becomes. This force is so strong that they actually cannot exist on their own: if you try and break two of them apart, at some point the energy required will be so high that two entirely new quarks will spontaneously form and bind with the two existing quarks and create new states called *hadrons*, of which protons and neutrons are examples. This is manifested mathematically in that the strong/QCD coupling, α_s, when evaluated at larger energy scales is smaller than when it is evaluated at smaller energy scales.

The reason this is an important consideration is that such a property admits significantly more complicated behavior during a collision. The protons are brought to such a high energy that, upon colliding and splitting particles out in all directions, the strong coupling is sufficiently small to consider the quarks as "free", meaning they don't form hadrons. In this respect, they are essentially just a bunch of electrons. However, after some more time passes, the changing distance/energy scales begin to increase the value of the coupling. At this stage, the quarks are no longer free, and they begin to hadronize. The behavior of interacting hadrons is no longer anywhere near as simple as the interacting free particles, since the substructure of the hadrons must be taken into account.

Further, before hadronization occurs, the free quarks will radiate gluons, particles that are similar to photons but also feel the strong force like the quarks. Often, due to the energetic nature of the collision, there will be a large number of emitted gluons with high energy, that then may decay into quark/anti-quark pairs, which may then subsequently radiate further. Due to this, there are often a very large quantity of quarks/anti-quarks present once hadronization begins, further exacerbating the problem.

To avoid being too pessimistic, I will make one final note about what makes this so difficult. Currently, our frameworks in theoretical physics admit solutions that are represented by an infinite series of terms, called *perturbation theory*. Essentially, this involves writing a function as an expansion in some small parameter λ like so:

$$f(x) = a_0 + \lambda a_1(x) + \lambda^2 a_2(x) + \lambda^3 a_3(x) + \dots. \tag{1.1}$$

Usually, λ is the coupling for particles involved in a particular reaction. For a single interaction, there is only one factor, but as you consider more interactions, more factors of the coupling will be included. For instance, considering a quark emitting one single gluon, there is only 1 interaction, that being the interaction between the quark and the gluon. However, if we consider a chain of quark radiation, there are multiple decays and interactions between the quarks and the gluons. Any part of given process technically has a non-zero probability of occurring, meaning the full infinite series is the "full," "correct" answer. Fortunately, the "small parameter" is usually sufficiently small such that we can only consider the first few terms and have the result be moderately accurate. In other words, considering only the single interaction is usually a good enough estimation for the entire process containing the infinite number of interactions.

1.3 Project Goals

The previous section was very general, encompassing the general idea of what should be considered to model such processes (and make calculations in general). Of course, I cannot include everything in my project, so I will make a few additional considerations.

First, for a simulation such as this, I will start by including only interactions that are represented by the first term in the aforementioned perturbative series. For instance, instead of a quark radiating a gluon which then splits into a quark/anti-quark pair and so on, I will consider only the radiation of a single gluon at a time. By virtue of perturbation theory, this will already be a decent approximation itself, at least in the high energy regime where the coupling actually remains small.

Second, modeling the entire process start to finish, i.e. from hard scattering to simulating the detector, would be an absolutely monstrous task. Most if not every program out there for simulating these processes only focuses on a subset of the full run. For instance, MadGraph5 largely focuses on the hard scattering process, i.e. generating events that occur immediately after the collision and before any quark radiation. Pythia8 and Herwig++ can do this too, as well as taking into account the parton showering elements. Geant4 focuses only on the detector simulation, and takes as input the output of Pythia8, Herwig7, or any other parton showering program. My project will focus primarily on the hard scattering process as well as the parton showering, before hadronization. I will not attempt to simulate the detector structure.

Unfortunately, due to the nature of the process I chose to simulate, namely $pp \to Z/\gamma^* \to \mu^+\mu^-$, which corresponds to the two protons interacting, exchanging a Z-boson or photon, then emitting two muons, is incompatible with the leading order parton showering algorithm, specifically with regards to using the outputs from the former for the latter. In principle, if I was considering higher orders, I could generate events with the muons then shower them by having them emit *photons*, but the leading order (and simplest) method for implementing parton showering is by *quarks* radiating *gluons*. Therefore, the hard scattering and parton showering parts will simply be separate, but they will still be valid in their own regimes.

2 Theory/Literature Review

Many of the individual things listed in this section are, in the land of computational/theoretical physics, very well established, meaning that no individual results are necessarily worthy of their own reference. Rather, see Ref. [2], for instance, for descriptions of the computational techniques used and Ref. [3] for descriptions of the physics theory.

2.1 The Hard Scattering Process

2.1.1 Monte Carlo-Based Event Generators

The driving force behind these types of simulation programs is the Monte Carlo algorithm, whose main purpose is to solve multi-dimensional integrals by averaging the integrand via randomly sampling points in the domain. Mathematically, what we are achieving is, given some integral

$$I = \int_{t_1}^{t_2} \mathrm{d}t \; f(t), \tag{2.1}$$

we can approximate the result by

$$I \approx \Delta t \, \langle f(t) \rangle = \frac{\Delta t}{N} \sum_{\tau=1}^{\infty} f(\tau_i), \tag{2.2}$$

where the τ_i lie in the range (t_1, t_2). At this stage, we simply *generate* τ_i's randomly, and the approximation will converge to the result after a sufficient number of evaluations. It's worth pointing out that we can randomly generate a number in the range $(0, 1)$, then scale it to the domain, something like

$$\tau_i = (t_2 - t_1)\rho_i + t_1, \tag{2.3}$$

where ρ_i is the random number, since the random number interface in most programming languages spit out a number between 0 and 1.

Of course, there are myriad other ways to solve integrals, but in particle physics, our problems and integrals are often formulated such that the dimensionality of the domain is large enough to dissuade the usage of quadrature-based integration schemes. Such integration schemes are more efficient in lower dimensions as they converge faster, but get much slower compared to Monte Carlo integration, whose rate of convergence

increases in "constant time" in relation to the dimension. Of course, as an additional note, the reason this is a requirement at all is that the integrals we are faced with are very often mathematically impossible to solve analytically; e.g. the answer cannot be expressed in closed form.

Considering the physical process of the proton-proton collision, the starting point is the hard scattering process, which occurs immediately after the collision. The quantity we wish to calculate is the *cross section* σ for a particular relation which is, more or less, the probability for that reaction to occur. This is a quantity of interest to calculate because it can be measured and determined in the detectors, giving us a way to compare our theory with what is observed in real life.

To do this calculation, there are a few assumptions one must make. To understand this, we first know that protons are made up of quarks; one can think of the proton as a little ball with several quarks bouncing around inside. In the super high-energy collisions at the LHC, the assumption we make is that the proton's structure, i.e. *where* the quarks are and their *longitudinal* momenta (the portion of momenta of the quark that is in the direction of the proton's momenta) can be completely separated from the interactions between the individual proton's quarks. The point of doing this is that we are essentially ignoring the proton itself and only focusing on the interactions between the individual quarks. The location and momenta information is stored in what are called **parton distribution functions** (PDFs), and the information relative to the quark-quark interactions is stored in the *partonic* cross section $\hat{\sigma}$, denoted with a hat instead to differentiate it from the total cross section.

To put this more formally, the total cross section is related to the sum of all possible interactions via the various quarks within the proton, proportional to the fraction of momentum they have compared to their host proton. Mathematically this looks like:

$$\sigma(p(P_1)+p(P_2)\to X)=\int_0^1 \mathrm{d}x_1\int_0^1 \mathrm{d}x_2\ \sum_{i,j} f_{f_i}(x_1,\mu^2)f_{f_j}(x_2,\mu^2)\cdot\hat{\sigma}(f_i(x_1P_1)+f_j(x_2P_2)\to X). \quad (2.4)$$

This is a little dense, but here are what all the symbols mean: The left side is the cross section, representing two protons, one with momentum P_1 and the other with momentum P_2 colliding and leaving a generic final state X. On the right are the functions f_i and f_j, which are the PDFs for quark i and quark j. These are multiplied by the partonic cross section with quark f_i (containing total momentum x_1P_1, i.e. the fraction x_1 of the total momentum of its host proton P_1) interacting with quark f_j and producing the aforementioned final state X. μ^2 is energy scale of the process, since different quarks have different momentum fractions at different energy scales. Finally, these are convoluted together to get the final result.

The (differential) partonic cross section itself is given by:

$$\mathrm{d}\hat{\sigma}(f_i(x_1P_1)+f_j(x_2P_2)\to X)=\frac{1}{2\hat{s}}|\mathcal{M}_{f_if_j\to X}|^2\left(\prod_{k=1}^{N}\frac{\mathrm{d}^3p_k}{(2\pi)^3 2E_k}\right)(2\pi)^4\delta^{(4)}(p_1+p_2+\sum_{k=1}^{N}p_k). \quad (2.5)$$

Here, E_i and p_i are the energy and momentum of the ith parton, and the sum/products are over the N different particles that may possibly be in the final state X. For this project, we are only considering $2\to 2$ processes, in which two quarks interact and only two particles are produced in the final state. The calligraphic $\mathcal{M}$ is the *amplitude*, and is specific to the process under consideration. Later, upon introducing the process we are modeling, the exact form of this amplitude will be shown. Everything apart from the amplitude in the expression is called the **phase space**, and essentially covers all energy/momentum configurations and inputs/outputs. In relation to the Monte Carlo algorithm, it is these phase space points that we randomly generate in the evaluation of this integral.

2.1.2 The "Hit-or-Miss" Method

In the event generation, when we are generating random phase space points and calculating our quantities of interest, we want to make sure that the events we are generating are in accordance with what would happen in the real world. In particular, if a phase space point gives a cross section significantly lower than others, it shouldn't necessarily be considered on the same footing as other events which are in principle far more

likely to actually occur. One solution is to drag around every event's cross section so that we can take it into account later on. However, this extra baggage and additional computation can be inefficient.

One highly popular solution is dubbed the "Hit-or-Miss" method. This method involves finding the maximum "weight", which is the value of the integrand, during the calculation of the cross section; we shall call it $W_{\max}$. Then, when generating phase space points for the actual events, we only *accept* an event (it "hits") with a probability of $w/W_{\max}$, where w is the weight of that event (otherwise it "misses"). One caveat to this method is that it doesn't work well if the function is not flat. One way to imagine this is to consider a simple function/curve on the 2D plane and generate random x-values. Hits are those that land in/below the curve, and misses land outside/above it. Clearly, if the function isn't very flat, this method doesn't work well.

Sometimes, we are able to circumvent this by applying a transformation/change of variables. One particularly good example is the Breit-Wigner peak, modeled by the function

$$F(m^2) = \frac{1}{(m^2 - M^2)^2 + M^2\Gamma^2}. \tag{2.6}$$

This function is used to model resonances, which are essentially unstable particles, where m^2 is the center of mass energy that produces the resonance, M is the actual mass of the resonance, and Γ is the decay width of the resonance, which is inversely proportional to the lifetime of the particle. We are often interested in integrals of this function:

$$I = \int_{M^2_{\min}}^{M^2_{\max}} \mathrm{d}m^2 \, \frac{1}{(m^2 - M^2)^2 + M^2\Gamma^2}. \tag{2.7}$$

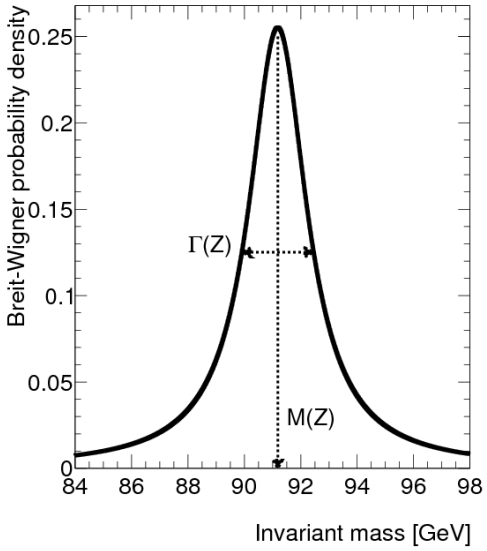

Figure 2.1: Example of the Breit-Wigner distribution.

An example plot is given in Fig. 2.1; the peak is at 91 GeV, which is the mass of the Z-boson, and the width Γ is, again, related to its lifetime. Obviously, such a function would be a particularly bad candidate for the hit-or-miss method. However, if we made the change of variables to ρ like so:

$$m^2 = M\Gamma \tan\rho + M^2, \tag{2.8}$$

our integral turns into

$$I = \frac{1}{M\Gamma} \int_{\rho_{\min}}^{\rho_{\max}} \mathrm{d}\rho, \tag{2.9}$$

which is perfectly flat, thus removing any variance and making it a perfect candidate for the hit-or-miss method. Unfortunately, we are probably never going to be able to find a transformation that is perfect like this one, but we can often get very close, which is more than good enough, especially considering that we are already truncating things at leading-order anyways.

At this point, once we have a set of phase space points that passed the hit-or-miss selection, we can generate events by simply generating some momenta in accordance with the phase space information. These will correspond to particles that, in a real simulation, would be produced immediately after the collision.

2.2 Parton Showering/Hadronization Process

The calculation of hard scattering cross section and generation of events in accordance with the cross sections is a bit more simple than the parton showering process, since the former involves a very general Monte Carlo algorithm. Parton showering is more directly based within the physics itself. As such, the results presented in this section are a conglomeration of results taken from the PYTHIA8 and HERWIG7 manuals [4], [5], as well as from a condensed "tutorial" given in [6].

The main physics behind parton showering is the idea that, immediately after the main collision, quarks will fly out with super high energies. In general, quantum mechanics dictates that particles want to release their excess energy and drop down to their "ground" state. The highly energetic quarks are certainly not in their ground state, and one method by which they release this additional energy is by emitting gluons. In general, any given quark can do this several times, and the gluon may decay into a quark/anti-quark pair, which may emit gluons again, and so on. This leads to a "shower" or "jet" of particles in the final state.

We consider the simple case of a single quark emitting single gluons one at a time, because anything further becomes extremely complex. The main mathematical structure related to the emission of gluons/photons is the **Sudakov form factor**, defined like so:

$$\Delta(t_0, t) \equiv \exp\left[-\int_{t_0}^{t} \mathrm{d}t'\, \Gamma(t')\right], \tag{2.10}$$

where t_0 is the quark's initial energy, and t is its final energy when it can no longer emit gluons. The function $\Gamma(t')$ is defined as:

$$\Gamma(t') \equiv \frac{1}{t'} \int_{z_-}^{z_+} \mathrm{d}z\, \frac{\alpha_s}{2\pi} P_{g\leftarrow q}(z), \tag{2.11}$$

where the t' is simply to differentiate from the main integration variable. α_s is the strong coupling constant, characterizing how strongly quarks and gluons interact with each other, z^- and z^+ are arbitrary limits on the max/min energy of the quark, and P is called the **splitting function**, which essentially gives the probability for the quark to emit the gluon (it is similar-ish to PDFs, but a bit simpler). It is defined like so:

$$P_{g\leftarrow q}(z) = C_F \frac{1+z^2}{1-z} \tag{2.12}$$

where $C_F = 4/3$ is a group-theoretical constant[1] and z is the momentum fraction of the gluon (analogous to x from the PDFs). The interpretation of the Sudakov form factor is that it gives the probability that the quark doesn't emit a gluon as we evolve from the initial scale t to the cutoff scale t_0. By evolve, we simply mean as the quark loses energy via anything else that may happen in the reaction, such as collisions with other particles, for instance. It may seem like this is reversed: why are we calculating when it *doesn't* emit a gluon?

Too see why we do this, we consider how this is implemented in a program. What we do is scale the Sudakov factor to the range $[0, 1]$, calculate it for a given t and t_0, then generate a random number also in the range $[0, 1]$. The larger the value of the Sudakov factor, the less likely it is to emit a gluon. So, if our random, number lies *above* the value of the Sudakov, we *do* have an emission, otherwise we don't. We then solve for t by inverting $\Delta(t, t_0)$ to calculate at what point this emission occurs, then continue on from there,

[1] Physicists, for one reason or another, often like to leave things like this as general as possible. The value it takes is nearly always 4/3.

where now the quark's energy has been reduced to whatever energy it emitted the gluon at. Further, we need to determine/consider the kinematics of the emitted gluon, specifically z, the fraction of the parton's momentum it carries. This is done by solving for z by inverting $\Gamma(t)$.

There are a few issues with solving for t and z, and that is the fact that both require evaluating inverses or integrals of functions which are, in general, very hard to integrate or invert. There is an algorithm to largely resolve this, called the *Sudakov veto algorithm*, which resembles the hit-or-miss method discussed in the previous section. The main idea is that for the functions that we have to invert, we take away some of the complexity by defining new variables that are *overestimates* of the original variable for the entire domain such that the function is more easily invertable. Of course, this implies a higher "acceptance" of events or of emissions – the probabilities are then "fixed" by only accepting proposed events according a ratio of the original variable's value to the overestimate's value.

As an example, an overestimate of the splitting function would be given by

$$\hat{P}(z) = C_F \frac{2}{1-z}, \tag{2.13}$$

since $z \in [0, 1]$ so the numerator in Equation (2.12) will never be larger than 2, meaning that Equation (2.12) overestimates the splitting function everywhere. This overestimate is now much easier to invert and solve for z.

2.2.1 The Cutoff t_0

Lastly, we explain why we need some cutoff scale. We already mentioned beforehand that the quark emits gluons so that it returns to its ground state. This isn't *wrong*, in general, but for quarks, it is a little different. In the introduction, it was mentioned that the force quarks experience is the reverse of the electromagnetic force: it becomes more strongly attracted to quarks at *larger* distances. Energies, in general, in the quantum world, are directly related to distances (hence why we say we are probing the universe at high energies, as this is equivalent to saying we are probing the universe at the smallest distance scales). So, once the quark releases enough energy, the distance scales associated with the system are sufficent that the coupling between quarks grow enough to hardonize, e.g. form protons. There is a universally understood energy(/distance) scale at which this occurs, roughly $\sim$1 GeV [7].

At this point, then, our perturbation theory approach breaks down, as the series in Equation (1.1) no longer converges since λ, which is the coupling strength, is so high. Therefore, the entire theory on which this method is based breaks down,[2] and we cannot continue calculations below this limit without producing results that are inconsistent with the real world. Additionally, it is obvious that if we let t get too small, the factor of $1/t$ in Equation (2.11) will diverge, which we must avoid.

3 Model/Implementation Details

3.1 Programming Language

This is programmed in C++, as the C family of languages are what I am familiar with, and speed is the name of the game here, due to the immense number of calls to complex mathematical routines.

3.2 Dependencies

There are two main dependencies that are used, and those are Gnuplot and LHAPDF. Gnuplot is a simple command-line tool used for plotting data, and it will allow me to visualize results very easily and nicely, and have them be comparable to results from other frameworks. LHAPDF is a package used to interface with PDFs. The reason why this must be done with a library like this is because PDFs are not calculable: they must be fitted from experimental data. Also, there is one caveat with this library, and it is that it cannot be built/used on Windows, unless you use Windows Subsystem for Linux.

[2] The splitting functions(s) (Equation (2.12)) are calculated via perturbation theory; they would be completely invalid.

3.3 Structure and Diagrams

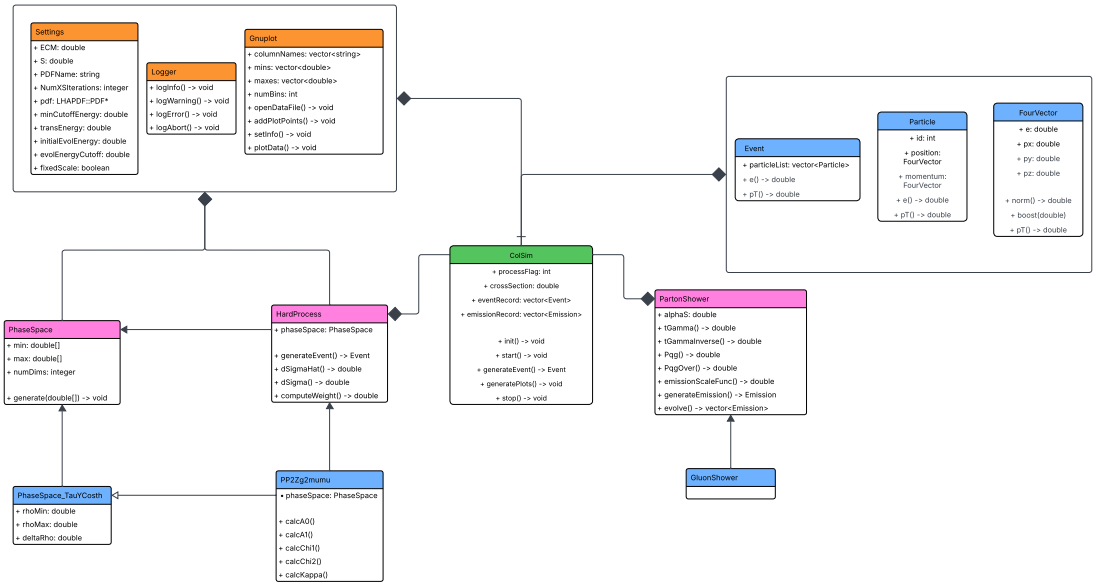

Figure 3.1: Basic UML diagram describing the interaction/relationships between the different classes.

Given in Fig. 3.1 is a UML diagram representing the main interacting parts of the program. The blocks listed in orange are singleton classes containing functionality that should be available throughout the program via a single interface. The `Logger` is a class designed at nicely formatting output to a log file and/or standard output/error. In particular, output is given a prefix stating the severity/priority of the message, since some messages are simply messages whereas some are warnings or errors, as well as a time it was outputted, and if your terminal supports it, colors. Further, if an error is serious enough, `logAbort()` will stop program execution.

The `Settings` class contains things related to reading in data from the configuration file, parsing it, and making it available throughout the program. It also contains other generic, configurable parameters.

The blue blocks are separate classes with no dependencies themselves as they are simple organization of numbers. In particular, this is where data related to particles such as ID, momentum, and so on are stored as well as groups of particles, weight information, and so on. These are referenced throughout the program, most notably in the `Event`/`Emission` classes, which are organized lists of particles.

The pink blocks are base classes representing generic behavior for that type of object. The `PartonShower` class defines common behavior such as the implementation of the Sudakov form factor which should be common for all types of parton showering, should more be implemented in the future. Similarly, the `HardProcess` and `PhaseSpace` classes contain shared information for all hard process and all of their phase spaces.

Lastly, the single green block, representing the `ColSimMain` class, is the main entry point for the user, containing simple methods to initialize the program, generate events, and view output.

3.3.1 Brief Implementation Demonstration

Here I briefly show a specific instance of some of the implementations described above and shown in the UML diagram in Figure 3.1, I show here the behavior for the calculation of the hard scattering cross section. By defining the base `HardProcess` class, additional processes are able to simply inherit from this class, making it very simple to add new processes (unfortunately the process itself is almost certainly going to be a challenge to implement) so long as they return the required information, namely the cross section, the error, the maximum value of the cross section (to be used for the hit-or-miss method), and the phase space points corresponding to the maximum cross section.

It is done this way so that, in `ColSimMain`'s `start()` method, all that we need to include is the code given in Listing 3.1.

```
// set phase space ranges according to input parameters
hardProcess->getPhaseSpace().setRanges();

// calculate the cross section
LOGGER.logMessage("Calculating cross section via Monte Carlo integration...");
HardProcessResult res = hardProcess->calculate();
LOGGER.logMessage("Finished!");
LOGGER.logMessage("Result is: %.9lf +- %.9lf pb (picobarns)",
                                  res.result, res.error);

// store values, errors, and phase space points
// corresponding to the maximum weight achived
crossSection = res.result;
crossSectionError = res.error;
maxWeight = res.maxWeight;
maxPSPoints = res.maxPoints;

LOGGER.logMessage("Maximum weight achieved: %.9lf", maxWeight);
```

Listing 3.1: The main code run within `ColSimMain`'s `start()` function to calculate the cross section.

The code for generating parton showering events is of course different, but still simple and allows for the implementation of additional events, so long as they return an array of `Emissions` encoding the quark's energies at each emission and the kinematics of the emitted gluons (which, in general, could be photons as well).

It is worth noting that the Logger functionality is also present here; the capital `LOGGER` is a macro defined like so:

```
#define LOGGER Logger::getInstance()
```

As a singleton, then, the logger functionality is now available, as a single instance, to the entire program, avoiding duplications or truly global variables. The Settings functionality is encoded in a very similar way.

We are also able to see similar behavior for the phase space: different cross sections can be encoded in terms of different phase space variables; sometimes variable transformations may make calculations easier. Either way, the phase space functionality is implemented in a nearly identical way: the main `PhaseSpace` class contains the names, minimums, maximums, and ranges for all of the that particular phase space's variables. It interfaces with the `Settings` singleton class to update ranges whenever values are input parameters are changed, and spits out randomly generated values (scaled appropriately) to be used in the calculation of the cross section.

3.4 Execution Instructions

ColSim was made as a library, so it can be interfaced with via a C++ program, as opposed to a single executable. An example main file is given in Listing 3.2.

The user first creates a `ColSimMain` object and initializes it for hard scattering. In between `start()` and `stop()` commands, we generate 100000 events and also generate plots. To generate events for parton showering, one can pass in the flag `ColSimMain::PARTON_SHOWER`. Both types of outputs are described in a subsequent section.

3.4.1 LHAPDF

Before continuing, LHAPDF must be installed on your system. As described previously, it is a library to allow for interfacing with PDFs, and is required for ColSim. Installation instructions are on their website,

```cpp
#include "ColSim/ColSim.hpp"
using namespace ColSim;
using namespace std;

#include <iostream>

int main() {
  // create the main object
  ColSimMain colsim;

  colsim.init(ColSimMain::HARD_PROCESS);

  // start/initialize generation
  colsim.start();

  // generate events
  colsim.generateEvents(100000);

  // generate the plots
  colsim.generatePlots();

  // stop/deinitialize generation
  colsim.stop();

  return 0;
}
```

Listing 3.2: An example main program interfacing with ColSim and generating 100000 hard scattering events.

and are relatively straightforward. To reiterate: **this library can only be build on Unix-like systems; it cannot be used on Windows without WSL.** Once the library is installed and your system paths are set up appropriately, PDFs can be installed by using the `lhapdf` command line tool. For instance, to install the CT18NNLO PDF set (the default for ColSim), one can run

```
lhapdf install CT18NNLO
```

in their shell.

3.4.2 Gnuplot

Gnuplot is far easier to install, as it is included in most package managers. For instance, Debian-based systems need only type

```
sudo apt-get install gnuplot
```

to have all required functionality. A quick check would be to ensure the `gnuplot` command line tool works by running

```
gnuplot --help
```

3.4.3 Compiling

Clone the repository somewhere, and cd into it. You can then follow standard CMake building:

```
mkdir build
cd build
```

```
cmake .. -DLHAPDF_ROOT_DIR=<path-to-LHAPDF>
cmake --build .
cmake --install .
```

The default path for binary installation is `/path/to/ColSim/install`, which lies within the project directory. This way, building and running the examples are very simple. Again, ensure that CMake can find LHAPDF's files. If they were not installed to standard system paths, you can set the `LHAPDF_ROOT_DIR` variable in your shell's environment or by passing its installation prefix to CMake, like above.

3.4.4 Running

Once ColSim is built and installed to your location of choosing, it can be used like any other library. In the `examples` folder in the repository is an example called "basic" which simply generates 100 000 hard scattering events and plots the kinematics (these will be described in Section 4). It it also built via the standard CMake building procedure, but of course without the installation step. An executable titled `basic` will be spit out in the build folder.

3.4.5 Configuration File

In the `res` directory at the top level of the project is a file titled `config.in`, which is a configuration file containing the default values for all of the input parameters for the model. Since these are all defaults, ColSim does not require a configuration file, but it can be copied to the same directory as the executable for the given example, and, if passing the name of the file to `ColSimMain`'s constructor, will read values from the file. One can also use the `readString()` method in the `Settings` class within the code to change input parameters if one does not want to use a configuration file. The entire file is given in Listing A.1 in Appendix A, as it is a bit large to display here. Also in the listing are descriptions for each variable, but for ease of viewing, we also present them here in the next (sub)section with a bit more information than what is included in the default configuration file.

3.4.6 Configurable Parameters

There is one generic parameter used for both hard scattering and parton showering:

- **PDFName**: The name of the parton distribution function (PDF) set to use. Different PDFs report information with different accuracy, including different effects, and so on. For the purposes of this program, most any popular PDF set is valid; the default is set to the CTEQ collaboration's NNLO set: `CT18NNLO`. If one wants to use other PDF sets, navigate here to see the available list and invoke the command line tool like so:

  ```
  lhapdf install <PDF-set-of-choice>
  ```

The hard scattering process has four configurable parameters:

- **NumXSIterations**: The number of Monte Carlo iterations to do in the computation of the cross section. This parameter of course carries no physical significance, but higher numbers reduce error and vice versa. The default is one million, but on my machine this already runs in under a second, so higher iterations are very feasible to give very accurate results.
- **ECM** (E_{CM}): The center-of-mass energy for the proton-proton collision, or in other words, each proton carries $E_{\text{CM}}/2$. This is set to 14.0 TeV as this is the current E_{CM} that the LHC in CERN is running collisions at. Higher E_{CM}s shift most energy/momentum-based distributions to the right.
- **MinCutoffEnergy**: The energy cutoff imposed during the cros section calculation. There is an absolute cutoff before the entire theory on which these calculations are based breakdown; this is around $\mathcal{O}(\sim 1.0\,\text{GeV})$, as described in Section 2.2.1. Due to the larger energy scale of the main interaction, we can afford to set this cutoff a bit higher to increase computational efficiency, but this can in principle be set as low or high as one desires, with accuracy impacts.

- **TransformationEnergy**: An intermediate calculational parameter that is usually kept around ∼60 GeV, or roughly around the minimum cutoff energy. As we will see in Section 5, values too much lower/higher cause odd behavior and divergences. This value is introduced as a transformation parameter as described in Section 2.1.2 in order to use the hit-or-miss method more easily. Hence, it has no physical meaning, but is left configurable.

The parton showering parameters are as follows:

- **InitialEvolEnergy**: The initial energy that the quark has before undergoing its evolution. This is usually set to the be roughly the energy scale of a generic subprocess of the main proton-proton collision which is roughly $\mathcal{O}(\sim 1\,\text{TeV})$; this is the default value.
- **FixedScale**: This is a Yes/No parameter. The value of the coupling term α_s for the interaction between the quarks and the gluons in principle changes with the energy scale, but not enough in this regime to substantially impact the physics. Of course, letting it vary with the quark's energy scale leads to slightly higher accuracy, but again, the impact is not significant enough to motivate me to keep it one way or the other.
- **EvolutionEnergyCutoff**: As described earlier, this cutoff is due to the fact that at lower energies than this our theory breaks down. Here, since we are directly evolving the quark to these low energies we want to have this cutoff be at the absolute minimum to capture as many emissions as possible. The code will actually error out if the user tries to set this to a value lower than 1 GeV and warns for a value >10 GeV.

4 Results

4.1 Hard Scattering

The main `basic` example outputs a number of different files, assuming the main program was not modified. In particular, for either hard scattering or parton showering there are `.dat` files produced containing the kinematic variables for all of the events.

```
#cos(theta) rho         y           Q         x1
-0.951787   1.5357      0.498004    325.786   0.0229237
-0.0794975  0.846551    0.0443711   87.5763   6.13877e-05
-0.83998    1.21204     0.64568     114.893   0.0332578
-0.138378   1.36989     0.107851    145.865   0.00029054
-0.164968   0.418056    0.0662585   72.106    5.3322e-05
-0.737887   0.668366    0.0813915   80.2655   7.61546e-05
-0.4515     0.854714    0.667957    87.9612   0.0344976
0.565371    0.49181     0.186333    74.3544   0.000198658
-0.194654   0.830988    0.885909    86.8572   0.313566
```

Listing 4.1: The first ten lines of `events.dat`, the output file from the hard scattering simulation.

As an example, running the hard scattering simulation spits out a file titled `events.dat`, the first ten lines of which look like that given in Listing 4.1. The variables, in order from left to right, are $\cos\theta$, the angle between the two final state particles, ρ, an intermediate calculational variable with no direct physical significance, y, the rapidity, a measure of the speed of the outgoing particles, Q, the center-of-mass energy of the process[3], and x_1, the fraction of momentum of the proton the first quark carried.

In Figure 4.1 are distributions for all of these variables except for ρ, since it is an intermediate variable. It is important to note that all of these distributions make complete physical sense. First, the most easy to explain is the $\cos\theta$ distribution. It would make complete sense that scattering angles are possible; limits

[3]This is not the center-of-mass energy of the proton-proton collision! Again, the subprocess involving the quarks occurs at a different energy related to the momentum fraction each quark carries.

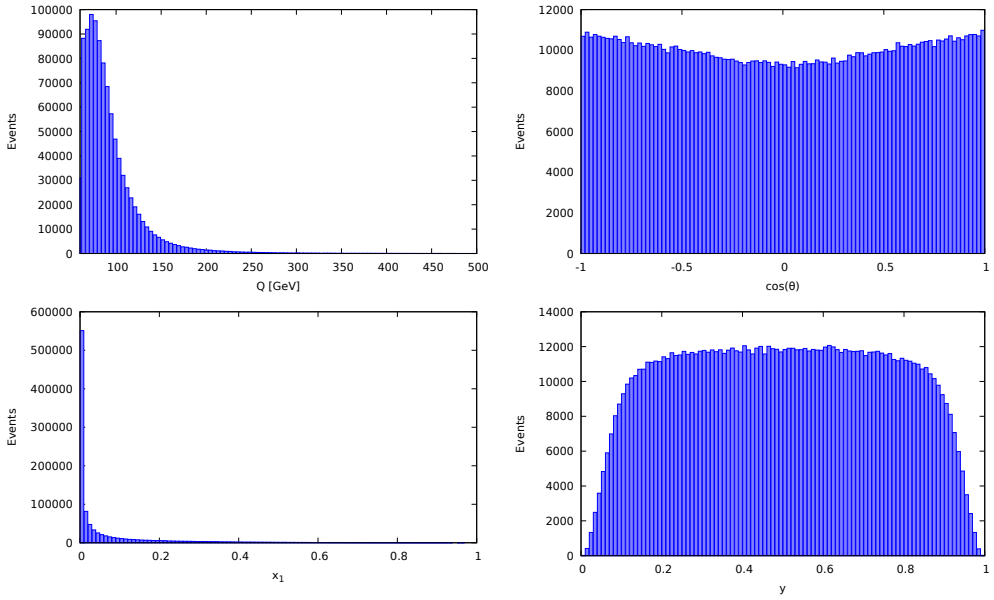

Figure 4.1: Outplot plots for distribution of the kinematic/phase space variables for the generated hard scattering events.

should instead start to pop up when looking at energies, rather than just the scattering angle. The Q distribution tells us that lower energy events are significantly more likely than higher energy ones. This is not saying that the higher energy one goes the more likely lower energy events will show up, rather it is simply reflecting the fact that the phase space for the lower energy events gives a higher cross section. As we will see in the sensitivity and scenario analysis, as we increase the center-of-mass energy of the proton-proton collision, this curve shifts to the right, reflecting the opposite of our naive assumption.

Additionally, in Figure 4.2 I have included another plot of the x_1 distribution to fully reflect the interesting fact that the majority of the events are with quarks that have a very small momentum fraction of the main proton. This is particularly interesting, but considering that, in the PDFs, there is a higher probability of finding quarks with lower momentum fractions, this checks out.

The rapidity distribution in Figure 4.1 reflects similar physics. I mentioned before that rapidity was a measure of speed, which is true, but it is also a kind of measure of the angle between the particle and the beam axis, as well. The higher the rapidity, the faster the particle and the less of an angle it has against the beam axis. Therefore, based on the observation that the lower energy events have the higher probability for occuring at a particular center-of-mass energy, we would expect fewer of the super high speed and low angle events, which is exactly what we observe in the rapidity distribution. This also explains the slight dip for smaller scattering angles, as this would reflect particles with higher rapidity.

4.2 Parton Showering

Similarly, when running the parton showering part of the simulation, it spits out a file titled `emissions.dat`, the first ten lines of which look like Listing 4.2. From left to right, the variables are t, the energy at which the emission occurred, p_T, the transverse momentum (perpendicular to the direction of the quark) of the quark and gluon, and the virtual mass of the quark and gluon. The virtual mass is interpreted as what the mass of a single particle with the same amount of energy traveling in same direction as the center-of-mass of the quark and gluon would be.

In Fig. 4.3 are the generated plots for the t, pT, and m_{virt} distributions for the generated parton showering events. We notice that all the curves follow a roughly similar distribution and for the same reasons as the Q distribution from the hard scattering case; that is, they peak at lower energies and fall off as the energy increases. This is most apparent in the p_T plot. The m_{virt} plot has a less pronounced peak compared to the p_T distribution, but this makes sense because this is taking into account the momenta of both the quark and

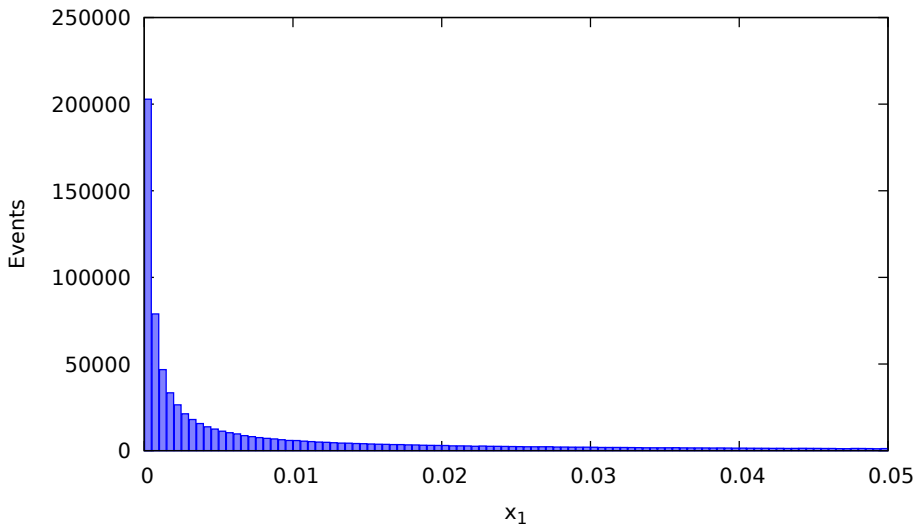

Figure 4.2: x_1 distribution with modified x-limits to reflect its range.

```
#t      pT      m
31.1053 10.1835 17.7978
14.6723 6.68227 9.90174
25.9178 6.50954 12.9889
18.2344 2.75489 7.08758
18.2067 6.32354 10.7299
22.1597 7.11254 12.5543
18.3092 3.78753 8.32746
25.3827 6.36348 12.7091
20.8469 4.96856 10.1774
```

Listing 4.2: The first ten lines of 'emissions.dat,' the output data file for parton showering simulation.

the emitted gluon, whereas p_T is in reference only to the emitted gluon. In the former, the emitted gluon is allowed to have a lower p_T, but if the quark still has a high energy, the virtual mass will remain higher. Of course, there is only so much energy to go around, so we still see it decrease in a similar manner, but the rounded peak is still expected.

Additionally, as a note, the $\sqrt{t}$ distribution is cutoff hard at $\sqrt{1000}\,\text{GeV}$, because the initial evolution scale of the parton showering was 1000 GeV in the simulation runs for which these plots were generated. Further, in all of the plots, we are able to notice the impact of the evolution cutoff parameter, set at 1 GeV for these runs. The importance/sensitivity of this parameter is discussed further in the next section.

5 Sensitivity Analysis

In this section, we consider all of the input parameters and how they impact the model's performance and the generated outputs. This sensitivity analysis helps define minimum and maximum values for the input parameters in terms of real-world significance and the model's efficiency/output.

5.1 Hard Scattering

5.1.1 Variables and Variation Plan

Here I list the hard scattering variables and the plan for varying their inputs:

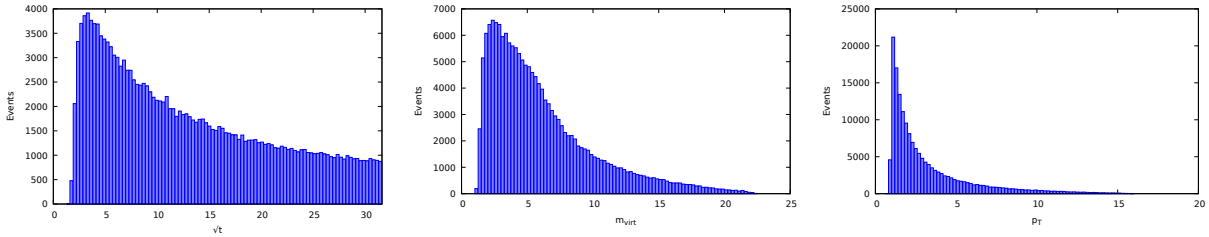

Figure 4.3: Output plots for t, m_{virt}, and p_T for the generated parton showering events.

- **PDFName**: Different PDF sets include different higher order corrections, so it will be good to pick a few of the more popular ones (that aren't too specialized for particular applications) to see how the results are effected.
- **ECM**: Higher energy collisions give rise to different cross section values, and hence a different distribution of final events. This will be varied continuously over a moderately wide range. We have to keep in mind, though, that we are attempting to model real world collisions, so we don't want to deviate too much from the 14 TeV used at CERN.
- **MinCutoffEnergy**: Like mentioned before, this value is kept moderately above the absolute 1 GeV cutoff to improve efficiency, but we can vary it close down to the level and much higher to gauge how it effects results.
- **TransformationEnergy**: This is a purely intermediary calculation variable. It does, however, rely on some of the above input parameters, but itself carries no physical significance. We can vary this widely and see how it effects results.

5.1.2 Results

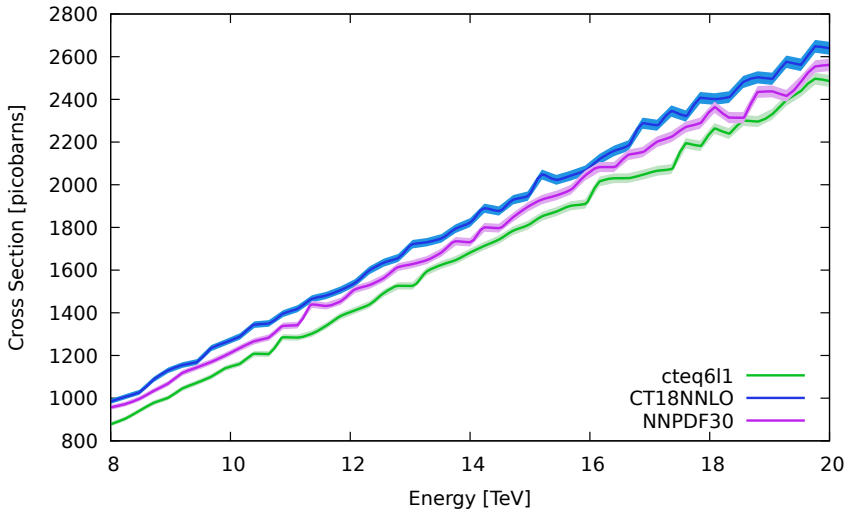

Figure 5.1: Cross section dependence on center-of-mass energy for the three different candidate PDF sets.

Figure 5.1 contains the curves depicting the dependence of the cross section on the center-of-mass energy, where the shaded region indicates the error. Every other parameter was kept fixed, and we dropped the number of Monte Carlo iterations from one million down a factor of 10 to one hundred thousand (the shading around the lines indicates the error, so one hundred thousand is still more than sufficient for this part). As we expect, the cross section grows monotonically with the ECM. Additionally, we can tell that the most accurate

PDF sets increase the total value of the cross section due to the fact that they contain more corrections to the main process. It is interesting how different the curves look, but this is due to vast differences in how different PDF sets are made, so it is not concerning.

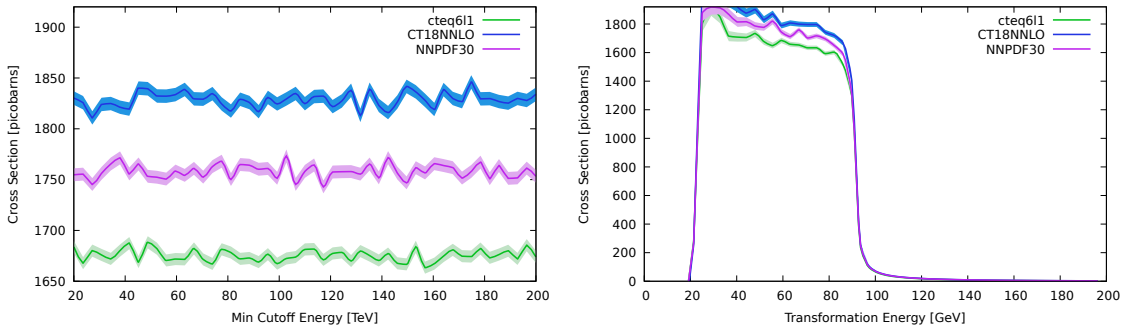

Figure 5.2: Cross section dependence on the minimum cutoff energy (left) and transformed mass/width (right) for the three different candidate PDF sets.

Figure 5.2 contains similar curves as Fig. 5.1 but for the minimum cutoff energy and transformation energy. The E_{CM} was kept fixed at 14 TeV and the number of Monte Carlo iterations was kept same. Notably, the cutoff energy seems to not have that much of an impact on the final cross section result as I might have expected, but this may be because of the limited range that I have used. I do know, from my physics knowledge, that the cutoff can only worsen the result. Keeping it within this range, then, is simply a choice the user can make without substantial impact on the final result.

The transformation energy, on the other hand, gives very interesting results. It seems the range of valid results is highly limited between 40 and 80 GeV, and outside this range, there are serious divergences and computational issues. This is not at all what was expected originally, but upon reviewing the places this transformation energy enters, it becomes clear where the issues arise. The center of mass energy for the quark-quark process is denoted $\hat{s}$ and determined by

$$\hat{s} = M_{\text{tr}}\Gamma_{\text{tr}} \tan\rho + M_{\text{tr}}^2, \tag{5.1}$$

where $M_{\text{tr}} = \Gamma_{\text{tr}} \equiv E_{\text{tr}}$, with E_{tr} being the transformation energy input parameter[4]. The range for ρ is given by

$$\rho_{\min} = \tan^{-1}\left(\frac{Q_{\min}^2 - M_{\text{tr}}^2}{\Gamma_{\text{tr}} M_{\text{tr}}}\right), \tag{5.2}$$

$$\rho_{\min} = \tan^{-1}\left(\frac{S - M_{\text{tr}}^2}{\Gamma_{\text{tr}} M_{\text{tr}}}\right), \tag{5.3}$$

where this capital, un-hatted S is defined as E_{CM}^2. Evidently, being governed by the $\tan^{-1}$, the two problematic points arise when $E_{\text{tr}} \approx Q_{\min}$ and $E_{\text{tr}} \approx \sqrt{S}$, which roughly corresponds to the two places the plot diverges. Therefore, it is understood that this value can and should only be changed within this range, and the program should exit upon specifying the parameter to be outside this range.

We next show the dependence of the distributions for the events that were generated after varying the ECM.

This is done in Figure 5.3 for the energy Q of the quark-quark process. It is surprisingly hard to notice, but the lower energy distribution has a sharper peak at a lower energy, while the higher energy distribution has a slightly more spread out peak but is noticeably shifted to the right. The extent to which it is shifted is much smaller than I would have expected, and this is something I was never fully able to figure out. I comment on this again in Section 8. However, despite the small difference, it is still absolutely a valid difference, and reflects (at least somewhat) the accurate picture of the physics of the interaction.

[4]In principle, one can specify different values for the width/mass, but they always enter the calculation together, so it is not particularly necessary to see how they vary individually.

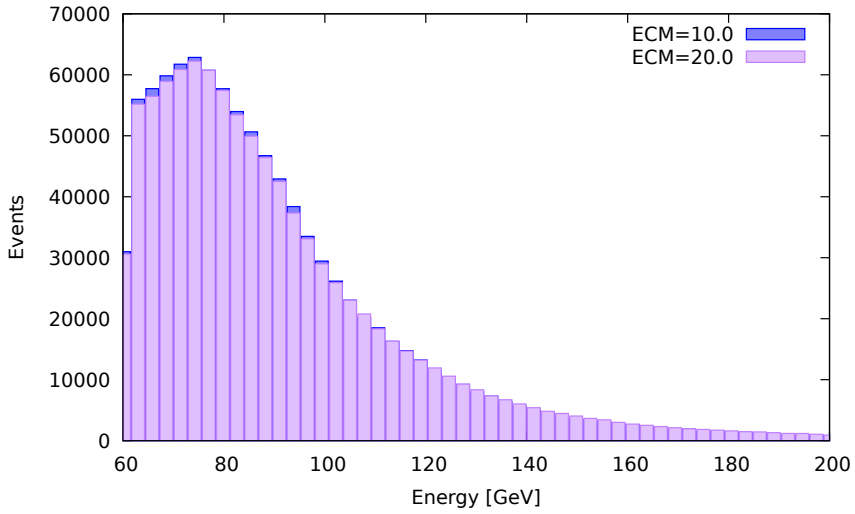

Figure 5.3: Q distributions after varying the center of mass energy.

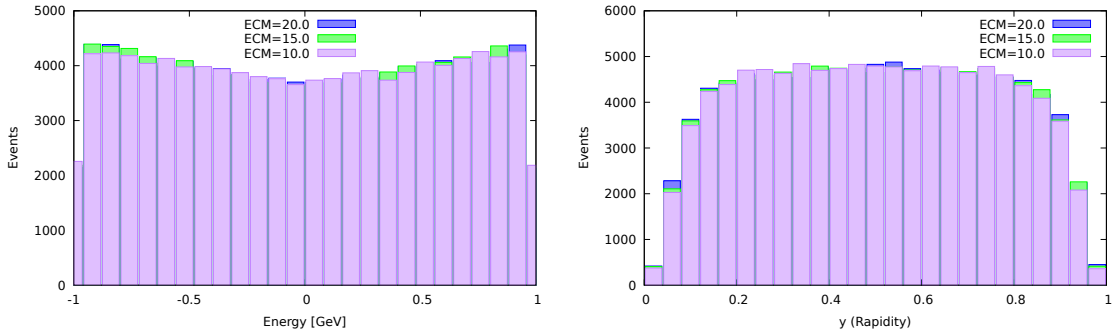

Figure 5.4: $\cos\theta$ and y (rapidity) distributions for varying the E_{CM}.

In Figure 5.4 are the distributions for the scattering angle, $\cos\theta$ and the rapidity, y, after varying the E_{CM}. These are separated from the main Q distribution since they are symmetric and aren't necessarily related to the absolute value of the E_{CM}, and indeed, this is what we find: there is no particular difference between the distributions at the different energy scales.

5.2 Parton Showering

5.2.1 Parameters and Variance Plan

Here I list the parton showering variables and the plan for varying their inputs:

- **InitialEvolEnergy**: This energy is meant to be on the order of 1 GeV, which is the average energy that a hard scattering process will spit a quark out at for an initial E_{CM} of 14 TeV. Therefore, this can be varied a bit above and below, but not too terribly far, since otherwise it would be quite unlikely in a real collision.
- **FixedScale**: The other two parton showering variables will both be varied twice, once with this set to Yes and again with it set to No to try and gauge the effects.
- **EvolutionEnergyCutoff**: The default of $\sim$1 GeV is the absolute minimum, so it won't be varied any lower. Further, we want to capture as many emissions as possible (as we saw in Section 4, the p_T distribution, for instance, was peaked at these lower energyes), so it won't be varied much higher.

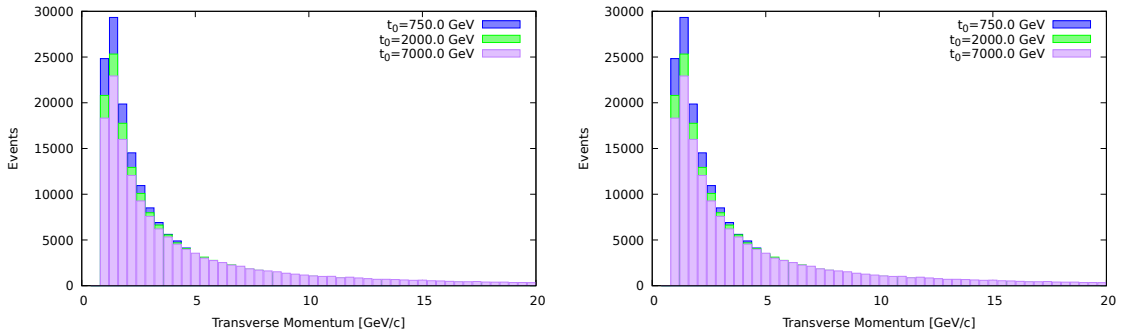

Figure 5.5: p_T distributions for the generated parton showering events by varying the initial evolution energy for a fixed scale for the coupling (left) and a variable scale for the coupling (right).

In Figure 5.5 are two plots containing the output transverse momentum p_T distributions for the generated parton showering events. On the left, the scale for the coupling α_s is kept fixed (and hence its value is also constant), and on the right, it is varied throughout each evolution.

The results here are a little bit more easy to see, and are as expected if we again consider the definition of the Sudakov factor. This factor determined the probability that the quark does *not* emit a gluon from some initial scale t_0 to a final scale t. This means that for a particular evolution starting at a t_0 which is higher compared to another at t_0', it is essentially just as probable for the former evolution to emit a quark at a t which is higher compared to the latter evolution at t'. The point is that it is more likely to emit gluons at a higher evolution scale when starting the evolution itself at a higher value (phrasing it this way makes it seem very intuitive, but it helps to have a mathematical grasp).

This is reflected in the plots since we see that the distributions corresponding to higher initial evolution scales are shifted to the right and are a bit more spread out with more events occurring at higher energies, corresponding to the fact that higher p_T events are more likely. Evidently, the model is performing as expected in this regard. Another thing that we notice is that varying the coupling vs. keeping it fixed seems to have very little effect. This is expected: the fixed coupling is chosen to be at the scale of half of the initial evolution scale, meaning that, on average, it *roughly* captures what the variable coupling would capture. Only super minor effects would matter, and those are impossible to tell in the graph. Further, the "additional effects" are only NLO effects, not NNLO or N^{3}LO (the order to which corrections to the couplings are known), meaning they are already very small.

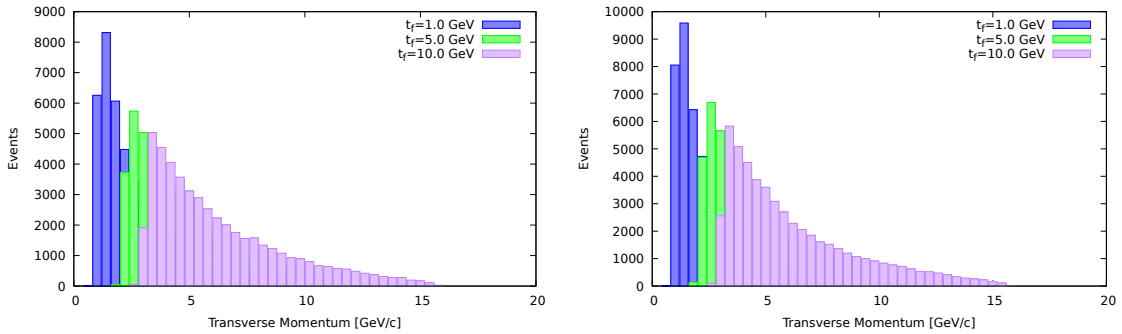

Figure 5.6: p_T distributions for the generated parton showering events by varying the evolution cutoff energy for a fixed scale for the coupling (left) and a variable scale for the coupling (right).

In Figure 5.6, I have done the same but this time by varying the final cutoff energy. This had the most notable effect of dramatically reducing the number of emissions that were generated by the end for the cases with higher cutoff. Further, the shape of each distribution is essentially identical, which makes sense. This is exactly as expected and shows further that the model works as expected.

5.3 Scenario Analysis

From my understanding of scenarios, as they are used in the context of this assignment, I believe that my model is incompatible with them. My reasoning ties back to what has been described about my model's specifics previously. In particular, the nature of my model simply doesn't allow for "optimistic" or "pessimistic" or any type of scenario at all, because there is nothing necessarily good or bad to expect. The example in the assignment gives things like higher or lower demand in the markets, which have good/bad real-world consequences. My model, on the other hand, is a model of a process in quantum physics. The behavior and mathematics the underpin it very tightly bind the process to behave in only a specific way, a part of which I attempt to capture with my model/simulation.

What I am trying to convey is that there are only two concrete scenarios for my simulation: either I follow the theorems and implement what is known in the current literature and accurately simulate what happens in the real world, or I violate any one of the dozens of theorems and assumptions in place that make modeling/simulating the model possible in the first place. Further, as a model that doesn't necessarily have the same effects directly on the everyday lives of people as traffic models or economical models, it is significantly harder to quantify what is even "optimistic" or "pessimistic" in the first place. The physics either occurs as expected, or doesn't – there's nothing else to it.

Therefore, I believe that it is not beneficial (or possible, really, without really stretching some things) to define scenarios and carry out this particular part of the procedure. I feel that, in the previous section, I have done as much as I can in terms of validating how the model performs under the changing of the various input parameters. I have also done my best to tie the changing of the input parameters to their physical interpretation in order to try and match at least some of the requirements of this section.

5.4 Final Parameter Ranges

Variable	Default	Min	Max	Notes
E_{CM}	14 TeV	−6 TeV	6 TeV	None
E_{min}	60 GeV	−50 GeV	440 GeV	None
E_{tr}	60 GeV	−20 GeV	20 GeV	Tied to E_{CM} and E_{min}.
t_0	1000 GeV	−250 GeV	5000 GeV	None
t_f	1 GeV	0 GeV	10 GeV	Hard minimum cutoff.

Table 1: Definitive table defining the absolute sensitivities of the possible input parameters.

In Table 1 is a definitive list of all the min and max values each input parameter can take. The PDF name and fixed α_s scale are omitted since those are discrete variables, and in principle, any recent PDF can work and either fixed or a variable scale is perfectly valid for the simulation.

6 Validation

In this section, we attempt to validate the input variables and other components of our model by ensuring that they are physically consistent, and their error does not impact the performance or output of the model. We do statistical tests for the Monte Carlo algorithm and compare our results to MadGraph5 and Pythia8 to ensure our model is giving consistent results. We also verify the individual components of our model via unit tests to ensure that all the moving pieces of the model are performing as expected and allowing the simulation to perform without issue.

6.1 Values of Hard-Coded Constants in the Literature

The simplest starting point for validation is to check the hard-coded variables present in the `Constants.hpp` header file. The Particle Data Group (PDG) [8] contains listings for all of the values and their current known accuracy that I have used for my program. These can be found online at this link. They are listed here, with comments on their precision:

- N_C, T_R, C_A, and C_F: These are group-theoretical constants, and are exact, i.e. not calculable or anything. They are not contained within PDG tables, but a Quantum Field Theory textbook usually contains the values for these variables; see, for instance, Chapter 16/17 of [3].
- α: The QED coupling constant, otherwise known as the fine-structure constant. This is calculable and known to a very high precision.
- m_Z, Γ_z: The mass and width (lifetime, essentially) of the Z-boson. This is often used as a reference scale for many calculations.
- m_C, m_B: The masses of the charm and bottom quarks, which are often higher than the lowest cutoff scale but lower than final evolution scales, so these are used as intermediate references in some cases.
- "Magic Factor": The magic factor is a simple unit conversion factor. In our calculations, we often set $\hbar = c = 1$ to simplify things, but of course, we must restore their values at the very end. This is done via this factor. As a unit conversion, there is no error associated with this value.
- G_F, θ_W: The Fermi coupling constant and Weinberg angle (often called the *weak-mixing angle*), are values used in calculations of the cross section.
- $\alpha_s(m_Z)$: The value of the strong coupling constant evaluated as the scale associated with the mass of the Z-boson. As mentioned, the Z-boson mass is very often used as a reference scale, and therefore so is the value of the strong coupling constant at that scale.

All of these values are *constants* in the very sense of the word: they must be hard-coded, otherwise, if they were to be configurable and changed from what they are currently set to, we would be simulating non-physical processes. Importantly, all of these quantities have been calculated to a very high precision, as one can see by examining the PDG tables. Therefore, it is reasonable to assume that the error incurred on the model output by the choice of these parameters is negligible.

6.2 Monte Carlo Convergence

It is of interest to ensure that the Monte Carlo algorithm performs well, particularly in the residual statistical error incurred via the inherent limited number of iterations we can compute. In general, we can assess the error via standard statistical methods, such as computing the variance and standard deviation. The basic definition for the variance, denoted by σ^2, of a quantity X is

$$\sigma^2 = \langle X^2 \rangle - \langle X \rangle^2 , \tag{6.1}$$

or, in discrete terms:

$$\sigma^2 = \frac{1}{N} \sum_i X_i^2 + \left(\frac{1}{N} \sum_i X_i \right)^2 . \tag{6.2}$$

The standard deviation is simply the square root of the variance, hence it is denoted σ. By running the Monte Carlo simulation for enough iterations, we can reduce this standard deviation to within acceptable bounds. In particular, Table 2 contains, for a few values of the center-of-mass energy E_{ECM}, the calculated cross section and corresponding standard deviation measurements, using one million Monte Carlo evaluations.

E_{ECM}	Cross Section (picobarns)	Error
12.0 TeV	1541.2249 pb	±5.5898
13.0 TeV	1685.1843 pb	±6.0952
14.0 TeV	1829.2264 pb	±6.6145
15.0 TeV	1972.5091 pb	±7.1462

Table 2: Values of the cross section calculation along with standard deviations for different values of E_{ECM}, using one million Monte Carlo evaluations.

From the results, we can tell that the error is already very small for one million iterations. The code takes around 1 second to do these computations, so for comparisons sake, Table 3 contains the same data but instead taking ten million Monte Carlo iterations, which increase the runtime by a factor of 10 (as one would expect).

E_{ECM}	Cross Section (picobarns)	Error
12.0 TeV	1550.8109 pb	±1.7763
13.0 TeV	1687.4082 pb	±1.9312
14.0 TeV	1823.7320 pb	±2.0855
15.0 TeV	1963.3334 pb	±2.2437

Table 3: Values of the cross section calculation along with standard deviations for different values of E_{ECM}, using ten million Monte Carlo evaluations.

As one would expect, increasing the number of iterations reduces the standard deviation associated with the calculation. However, one million iterations still gives errors of around $\sim 0.3\,\%$, which is more than good enough. There is nothing to compare to the literature in this regard, as this is a simple statistical error test.

6.3 Comparison of Cross Section Calculation with MadGraph5

MadGraph5 is a tool that is used by and large for cross section calculations and hard scattering event generation. It provides a bit of a simpler interface for the direct calculation of the cross section, so for this comparison, we will use it. In Table 4 are listed values from $E_{\text{ECM}} = 12\,\text{TeV}$ to $15\,\text{TeV}$, comparing the results obtained from MadGraph5 and ColSim.

E_{ECM}	ColSim σ_{xs}	ColSim σ_{sd}	MadGraph5 σ_{xs}	MadGraph5 σ_{sd}	Relative Error
12 TeV	1550.8109 pb	±1.7763	1553	±1.244	0.14 %
13 TeV	1687.4082 pb	±1.9312	1689	±1.561	0.12 %
14 TeV	1823.7320 pb	±2.0855	1832	±1.435	0.49 %
15 TeV	1963.3334 pb	±2.2437	1972	±1.623	0.44 %

Table 4: Comparison of cross section calculations between MadGraph5 and ColSim for selected center of mass energies.

From the results, we can tell that this portion of the simulation holds up extremely well, with the relative error between my results and results from MadGraph5 being less than half a percent. Higher energies start yielding larger errors, but the error is still very manageable and reflects the performance of the model.

6.4 Comparison of Phase Space Kinematic Distributions with Pythia8

The other part of the program consists of the parton showering. Pythia8 is one of the more popular parton showering algorithms out there. However, there is one issue, and it is that the parton showering model implemented in ColSim is extremely simplistic, and even after cutting out as many additional features from the Pythia8 parton showering options, I am still unable to reduce it to a simple quark emitting gluons from some initial scale to a final scale.

Therefore, there is no reason to directly numerically compare the results from the distributions and determine any sort of relative errors. The only possible thing that can be done is a qualitative comparison of the shapes of the output distributions, and ensure that the distributions that I generate with ColSim match both the general structure from that of Pythia8 as well as the intuitive physics understanding.

In Figure 6.1 I have given the plots for the distributions of the invariant mass for the generated events. There are some discrepancies, most notably, Pythia8 contains some additional events in the higher Q range. Figure 6.2 contains the rapidity distributions for the generated events (rapidity is a measure of speed, roughly speaking). In my code, this value is normalized a bit differently, but we notice that it follows the same pattern, i.e. higher (absolute value of) rapidity events are less likely. Lastly, in Figure 6.3 are the

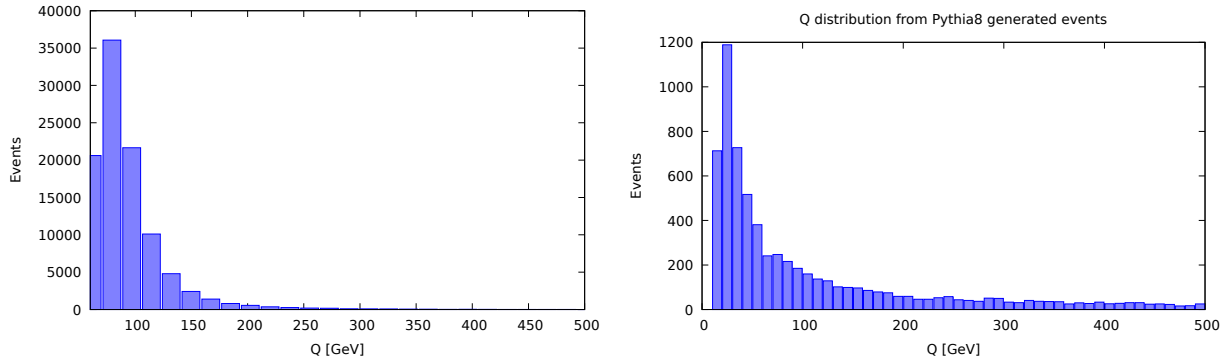

Figure 6.1: Comparison between parton showering invariant masses between ColSim(left) and Pythia8(right).

momentum fractions the quark in the initial reaction contains. These are remarkably equivalent; evidently, the quarks involved in the showering process represent those with a small momentum fraction in the proton.

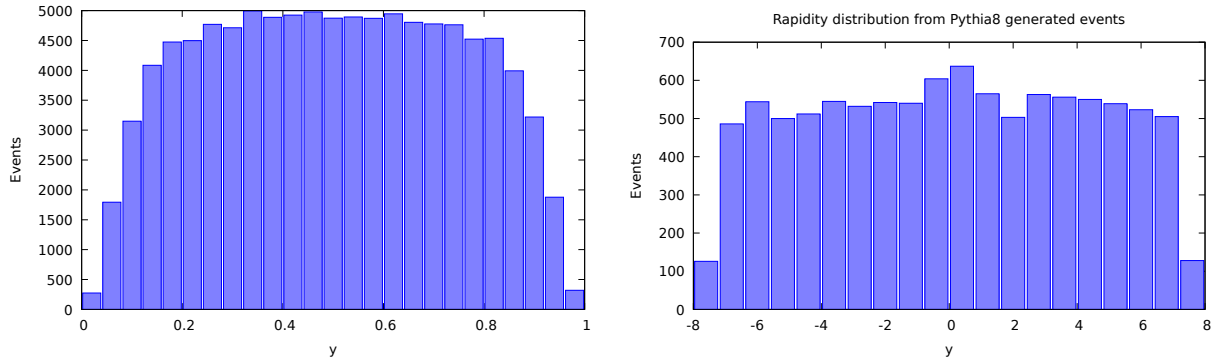

Figure 6.2: Comparison between rapidity between ColSim(left) and Pythia8(right).

6.5 Face Validation

My understanding of this section is that half of it is meant to be comparing the model outputs with what is expected by field experts. Of course, if my model matches the behavior both quantitatively in the case of the cross section calculation and quantitatively in terms of the parton showering, and the comparisons are drawn between programs written by field experts, then I have more or less already satisfied the requirements for this section.

Further, a few implementation specifics have been done with assistance from a particle theorist here at KSU who specializes in Monte Carlo algorithms and event generators. I have nothing to cite in this regard, but considering again that the behavior of the model matches so well with current literature and existing models I think that I have satisfied this section.

6.6 Historical Validation

In the case of my model, there is not much of a reason to compare with historical data. In principle, the results themselves have not drastically changed, but rather our computational tools and theories have grown to more or less reduce the error/uncertainty with the actual calculation of values. For instance, the value of all the input parameters have more or less remained the same since their conception, but have simply had their uncertainty reduced. All I'd be doing, then, is comparing against results that effectively represent the incomplete theory and incomplete computational results, which is not instructive.

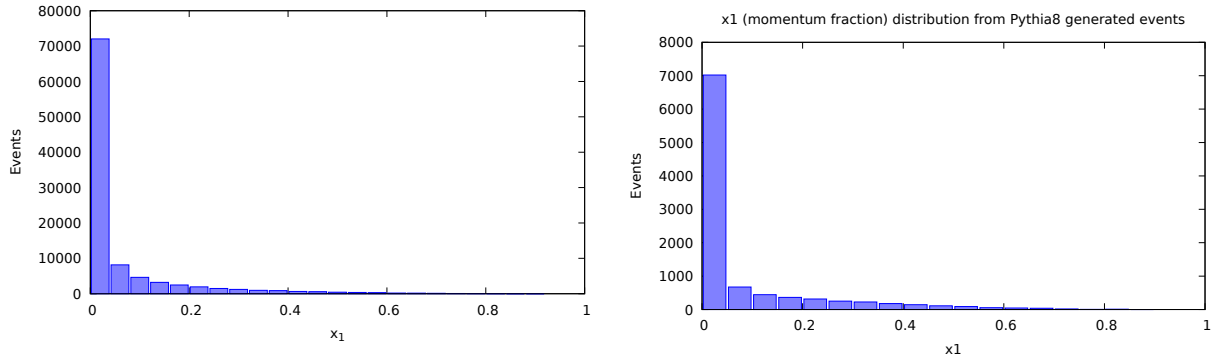

Figure 6.3: Comparison between the momentum fraction between ColSim(left) and Pythia8(right).

7 Verification

7.1 Core Components Unit Tests

Here I give some small unit tests for some of the smaller core components of the program, things like the Logger and some of the physics objects like four-vectors.

7.1.1 Logger

The unit tests here are minimal; all we need to test is that the four types of errors print as they should and handle the printing of various data types.

```
04/23/25 14:40:03 [INFO] This is a message.
04/23/25 14:40:03 [WARNING] This is a warning.
04/23/25 14:40:03 [ERROR] This is an error that will NOT abort the program.
04/23/25 14:40:03 [INFO] String: this is a string!
04/23/25 14:40:03 [INFO] Integer: 42
04/23/25 14:40:03 [INFO] Double: 5.67912
04/23/25 14:40:03 [ERROR] This is an error that will abort the program.
```

Figure 7.1: Output from the code given in Listing 7.1

The code used to test the logger is given in Listing 7.1 and the output is given in Figure 7.1. Evidently, the logger is performing as expected.

7.1.2 Settings

The Settings functionality will also be similar to test. All input variables have a default value, so the behavior for them all is that if they are neglected from the config file, no warnings/errors occur, rather the default value is set for that parameter. This can be illustrated by a simple test case as shown in Table 5. A few variables, such as the aforementioned cutoff scales, have hard limits and cannot be configured below 1.0, otherwise the program will crash. It also will emit warnings if the the cutoff is set too high, as this often reduces the accuracy of the calculation. These cases are given in Table 6.

Action	Keeping/removing E_{ECM} from the config file.
Expected Output	If omitted, keeps 14 TeV, otherwise uses specified value.
Actual Output (Omitted)	`[INFO] Using ECM=14000.000000`
Actual Output (Specifying 12.5)	`[INFO] Using ECM=12500.000000`

Table 5: Test case for omitting E_{ECM} from the configuration file.

```cpp
#include <ColSim/ColSim.hpp>
using namespace ColSim;

#include <string>
using namespace std;

int main() {

  LOGGER.logMessage("This is a message.");
  LOGGER.logWarning("This is a warning.");
  LOGGER.logError("This is an error that will NOT abort the program.");

  string str = "this is a string!";
  LOGGER.logMessage("String: %s", str.c_str());

  int x = 42;
  LOGGER.logMessage("Integer: %d", x);

  double y = 5.679122;
  LOGGER.logMessage("Double: %.5lf", y);

  LOGGER.logAbort("This is an error that will abort the program.");

  return 0;
}
```

Listing 7.1: Test program to testing logging functionality.

Action	Specifying a value < 1.0 and > 50.0 for the parton evolution cutoff scale.
Expected Output	For < 1.0, aborts, for > 50.0, gives warning.
Actual Output (< 1.0)	`[ERROR] Minimum cutoff energy for parton evolution MUST be greater than 1.0.`
Actual Output (> 50.0)	`[WARNING] Specified cutoff energy for parton evolution is abnormally high.`

Table 6: Test case for violating ranges for parton evolution cutoff energy.

Evidently, the settings interface seems to be performing as expected, and aborts upon specifying parameters that are completely incompatible with the model (and underlying theory), and warns if parameters are specified that are in principle allowed, but may lead to less accurate results.

7.1.3 Four-Vectors

The testing for four-vectors is relatively simple: the only non-trivial functionality of four-vectors is to ensure that the calculation of the transverse momentum components, i.e. $p_T^2 = p_x^2 + p_y^2$, is performed correctly, the calculation of the norm $p_\mu p^\mu \equiv p_0^2 - p_x^2 - p_y^2 - p_z^2$ is performed correctly, and boosts along the z-axis (the beam axis). A boost is defined by a parameter β which is the fraction of the speed of light to boost the particle. Defining

$$\gamma = \sqrt{\frac{1}{1-\beta^2}}, \tag{7.1}$$

the boost is given by

$$p^\mu \to p^{\mu\prime} = \begin{pmatrix} \gamma(p_0 - \beta p_z) & p_x & p_y & \gamma(p_z - \beta p_0) \end{pmatrix}. \tag{7.2}$$

We choose some random values for the components:

$$p^\mu = \begin{pmatrix} 89.0 & 5.9 & 3.4 & 9.8 \end{pmatrix}. \tag{7.3}$$

Quantity	Expected	Actual
p_T^2	7778.59	7778.59
$p_\mu p^\mu$	46.37	46.37
Boost of $\beta = 0.5$	$p^\mu = (42.05,\ 5.9,\ 3.4,\ 412.85)$	$p^\mu = (42.05,\ 5.9,\ 3.4,\ 412.85)$

Table 7: Calculated and expected values for calculated quantities involving four-vectors.

Table 7 contains the result from this test. Clearly, we are computing the correct values for four-vectors.

7.1.4 Gnuplot

The interface for plotting things with Gnuplot is less of a concrete test but rather that it can handle plotting some generic function. Due to the histogram format of the plotting that is done for the rest of the program, I do a mini Monte-Carlo to essentially plot $sin^2(x)$ between $0 - \pi$. The listing is a little long, so it given as Listing B.1 in Appendix B.

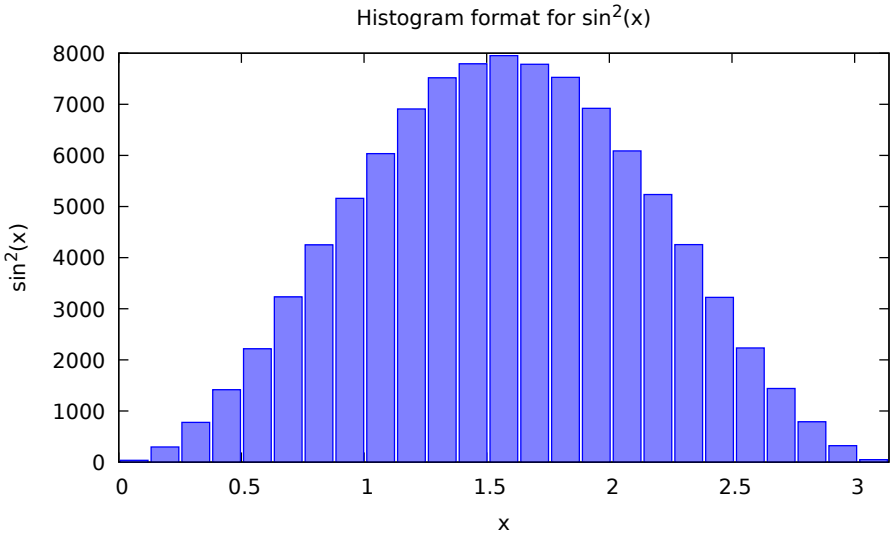

Figure 7.2: Histogram form for the plot of $sin^2(x)$.

The plot for the resulting distribution is given in Figure 7.2. The code for the Gnuplot interface therefore works as expected, and is relatively easy to use.

7.2 Monte Carlo Unit Test

A simple unit test of the Monte Carlo algorithm is conducted by running the algorithm using a few simple functions that can be calculated analytically. Since there is no reason for the algorithm to work for simple functions but fail for complex functions, I have selected a few easy functions that are easy to integrate analytically (or otherwise have exact known results):

The results are shown in Table 8. The first two functions are simple analytic functions and they yield less than a tenth of a percent relative error between the exact values and those determined from the Monte Carlo algorithm. The third function is a Gaussian, and due to the nature of the limits being $\pm\infty$, it is much harder to accurately assess the integral with smaller numbers of iterations. There are techniques to mitigate this, but the integrals that I compute do not have this feature, so I do not think it is necessary to fix anything. It is interesting to see, though, where the Monte Carlo algorithm breaks down. Despite this,

Integral	Monte Carlo Result	Error	Exact Value	Relative Error
$\int_0^1 \mathrm{d}x\ 2x$	1.0002	$\pm 5.774 \times 10^{-4}$	1.0	0.022 %
$4\int_0^1 \frac{\mathrm{d}x}{1+x^2}$	3.14164	$\pm 6.435 \times 10^{-4}$	π	1.42×10^{-3} %
$\int_{-\infty}^{\infty} \mathrm{d}x\mathrm{d}y\mathrm{d}z\ e^{-(x^2+y^2+z^2)}$	5.8080	± 0.122	$\pi^{3/2}$	6.59 %

Table 8: Test functions to ensure Monte Carlo convergence.

like I just mentioned, the integrals I compute are finite, meaning the algorithm's convergence is very good and it works as intended.

These are the main sub-components within the project. There are a few more super minor utility functions and other things, but these are insignificant enough to not necessarily require unit testing.

7.3 Physics Components

The issue with trying to unit test the physics components is that, at the end of the day, the main part of the hard scattering process revolves directly around the Monte Carlo algorithm. We have shown that that works very well. The remaining physics component is, in the context of this milestone, implementing the equations that are calculated with the Monte Carlo algorithm. Therefore, there really is no remaining unit testing to be done with respect to the hard scattering.

With respect to the parton showering, the Sudakov form factor is the main driving component. This, however, is less of a general algorithm like Monte Carlo, but instead is implemented directly for this case. Therefore, the only testing that can be done is on the entire unit as a whole, which has already been done in the previous main section.

Therefore, this concludes the unit testing for the main sections of the progam. It is evident that the individual components work well on their own and can withstand different inputs being presented.

7.4 Regression Testing

I am unsure what to write for this section. The behavior of the above individual components have worked from the get-go, with only interface changes; i.e. the Monte Carlo algorithm has always worked, the Gnuplot interface has always plotted the functions, with, again, only minor API changes. Therefore, there is nothing to really document for this testing case, but rather, the documentation changes will be reflected in the main program documentation.

7.5 Edge Case Testing

This has already been conducted in the previous section and the previous milestone, specifically in the case of the Settings testing and varying of the input parameters. Specific limits have already been tested and described thoroughly in the previous section, which are the edge cases this section refers to. Therefore, refer to that section and Milestone 6 for the edge case behavior.

7.6 Integration Testing

Integration testing is challenging, because any integrations of the above individual components are already present in the main output that I have already extensively discussed. Therefore, considering that the model performance overall considering both the numerical results and also the simple fact that the output is readable, nothing crashes, etc., I think the integration testing is more or less satisfied.

7.7 Code Review

A big (and apparently controversial) opinion of mine is that code readability is entirely subjective. If one is working on a large project with a number of collaborators, it's must consider the practices of the collaborators,

but this project is one I am programming myself. Therefore, readability is entirely my own preference, and has been written from the get-go with my own best practices I have been refining throughout my career.

In terms of actual coding practices as they relate to C++ programming, I have enabled all possible warnings and extra warnings during the compilation, i.e. via the `-Wall -Wextra` flags to the `g++` compiler. These have ensured that improper (not *unreadable*) code is avoided as much as possible. The current state of the code compiles in its entirety with zero warning (and of course zero errors).

7.8 Version Control

The code can be found on GitHub here. The version controlling has been very poor recently, as the majority of the work has been spent running tests and whatnot. Further, my version controlling via Git has not been the best in general, which is something that I have to improve on.

8 Discussion

Before making some definitive conclusions, it is worthwhile to discuss about some of the results presented in the final few sections. First, in Section 4, we gave plots for all of the kinematic variables for both the hard scattering and parton showering cases, as well as physical rationale for why the distributions look the way that they do. It was mentioned repeatedly that lower energy results are more likely, but this may be counterintuitive, so we briefly describe why this should be the case.

One would expect that, for a very high center-of-mass energy collision between the two protons, the resulting events are also quite high energy. Relative to other energy scales, this is true, as we saw in Section 5. However, the reason why the curves themselves aren't reversed, i.e. slowly increase up to the main center-of-mass energy and peak there, is because its easier to take smaller components of the main energy and have process occur, and its harder to put all of it into one process. Specifically in the case of $pp \to Z/\gamma^* \to \mu^+\mu^-$, this process is not as likely to occur with higher portions of the main energy because other processes (that I have not modeled) dominate in that regime. There are particles and other interactions that are simply more likely at higher energies.

All this is to simply to say that the shapes of the distributions given in the results section are exactly what we would expect in general for any process, but also specifically for this process.

There was one exception, that being the super small shifting of the Q distribution for the generated hard scattering process. As mentioned in that section, the shift was noticeable and definitely present (after having generated well over 10 million events multiple times, so there was no statistical fluctuations that led to the difference), but much smaller than expected. This is something that I was never able to figure out, because I would have expected similar distributions to the what p_T distributions looked like for varying the initial evolution energy in the parton showering process, which was far more pronounced. My only estimation is that I am measuring a different Q than I think, and it is simply following a similar pattern as other energy/momentum-based variables. Again, however, there was an expected difference, despite it being small, meaning it is still performing *something* correctly.

9 Conclusion

To conclude, I have presented here a thorough report on ColSim, a program that was developed to model and simulate proton-proton collisions at large colliders like the LHC by implementing the calculation of the cross section for $pp \to Z/\gamma^* \to \mu^+\mu^-$, as well as the generation of events for that process. Further, I implemented the simulation of parton showering and generating events for this case as well. We found that the results from the simulation in the hard scattering sector agree very well with other tools such as MadGraph5, and the parton showering results agree very well with Pythia8, though these comparisons were drawn qualitatively due to incompatibilities between ColSim and Pythia8. Evidently, the model performs very well and achieved its purpose.

There are many directions in which to further progress this model. One of the main pieces would be the implementation of different process, such as those governed by Quantum Chromodynamics (QCD), as those play a big role at higher energies. Further, upon successful implementation of such a process, specifically

one with quarks in the final state, it will be possible to then directly connect the hard scattering and parton showering components of the simulation and get coherent results for an entire process chain. After additional things like this, it would be more reasonable to compare results numerically with Pythia8 and get a better idea of the accuracy of the model.

Other improvements that could be made are related to the code itself, its versioning and documentation, as well as making it more functional for the end user. While I have relinquished as much control to the user as possible in terms of the physical parameters that go into the simulation, it could be beneficial to allow the user to fine tune, for example, the plotting, and allow him/her to customize the appearance of the plots more. An (statically linked) executable form of ColSim would also be nice, removing the need for the user to worry about linking or compiling.

Despite all of these possible improvements, I feel that ColSim was very successful and largely accomplished both the goals I set out to achieve as well as the requirements for this course. It may be hard to quantify this since the model is much different than what I reckon most of the other models were like, but I spent a lot of time reading, programming, and doing more reading to understand the results, and I am proud of the work that has been done.

References

[1] T. A. Collaboration, "Higgs boson production cross-section measurements and their eft interpretation in the 4ℓ decay channel at $\sqrt{s} = 13$ tev with the atlas detector," *The European Physical Journal C*, vol. 80, no. 10, Oct. 2020, ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-020-8227-9. [Online]. Available: http://dx.doi.org/10.1140/epjc/s10052-020-8227-9.

[2] W. H. Press, *Numerical recipes: the art of scientific computing*, eng, 3rd ed. Cambridge, UK: Cambridge University Press, 2007, ISBN: 978-0-511-33555-6.

[3] M. Peskin and D. Schroeder, *An Introduction to Quantum Field Theory*. CRC Press, 2019.

[4] C. Bierlich et al., *A comprehensive guide to the physics and usage of pythia 8.3*, 2022. arXiv: 2203.11601 [hep-ph]. [Online]. Available: https://arxiv.org/abs/2203.11601.

[5] M. Bähr et al., "Herwig++ physics and manual," *The European Physical Journal C*, vol. 58, no. 4, pp. 639–707, Nov. 2008, ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-008-0798-9. [Online]. Available: http://dx.doi.org/10.1140/epjc/s10052-008-0798-9.

[6] A. Papaefstathiou, *Pyresias: How to write a toy parton shower*, 2024. arXiv: 2406.03528 [hep-ph]. [Online]. Available: https://arxiv.org/abs/2406.03528.

[7] A. H. Hoang, O. L. Jin, S. Plätzer, and D. Samitz, *Matching hadronization and perturbative evolution: The cluster model in light of infrared shower cutoff dependence*, 2024. arXiv: 2404.09856 [hep-ph]. [Online]. Available: https://arxiv.org/abs/2404.09856.

[8] S. Navas et al., "Review of particle physics," *Phys. Rev. D*, vol. 110, no. 3, p. 030 001, 2024. DOI: 10.1103/PhysRevD.110.030001.

A ColSim Configuration File

```
# ----------------------------
# ---- General Information ----
# ----------------------------
# pdf set to use
PDFName=CT18NNLO

# --------------------------
# ----- Hard Scattering -----
# --------------------------
# number of evaluations of the differential cross section in the Monte Carlo integration
NumXSIterations=1000000

# Center-of-Mass energy: measured in TeV
ECM=14.0

# the lowest energy used in calculating the cross section
MinCutoffEnergy=60.0

# these are intermediate calculated parameters used in a change-of-variables transformation
# during the cross section calculation -- there is no real physical significance
TransformationEnergy=60.0

# -------------------------
# ---- Parton Showering ----
# -------------------------

# the initial energy that the quark carries before starting evolution (in GeV)
# usually set to be characteristic of a proton-proton subprocess, ~1 GeV
InitialEvolEnergy=1000.0

# whether to fix the factorization scale throughout the
# evolution or vary it with the evolution scale
# NOTE: fixing the scale misses out on some higher order effects
# Yes or No
FixedScale=Yes

# the energy cutoff reached before evolution stops (in Gev)
# this is the energy scale in which perturbative QCD stops being valid
# cannot go lower than 1.0
EvolutionEnergyCutoff=1.0
```

Listing A.1: ColSim's default configuration file, providing the default values for all input parameters.

B Listing for Gnuplot Test

```
// required for setup of random number generator
ColSimMain _;

// create gnuplot object
Gnuplot plot;

// specify plot information
vector<double> min{0.0}, max{3.14}, delta{3.14};
double numBins = 25;
plot.setHistInfo(min, max, delta, numBins);

// give col/file name, open the datafile
vector<string> colNames{"sin2_x"};
plot.openDataFile("data.dat", colNames);

// grab values of sin(x)
// in general we take a vector of vectors
// for multiple variables,
// that's the reason for adding a vector
uint numPoints = 0;
while (numPoints < 100000) {
        double x = randDouble() * 3.14;
        double r = randDouble();
        double val = sin(x)*sin(x);
        double ratio =  val / 1.0;
        if (r < ratio) {
            plot.addDataPoint(vector<double>{x});
                numPoints++;
        }
}

// define the x and y labels
vector<string> xlabel{"x"}, ylabel{"sin^2(x)"}, title{"Histogram format for sin^2(x)"};
plot.setTitles(title);
plot.setXLabels(xlabel);
plot.setYLabels(ylabel);

// make the plots
plot.plot();
```

Listing B.1: Listing for the plotting of $\sin^2(x)$. Given without a main function or includes.