ColSim: A Monte-Carlo event generator for proton-proton collisions
CS 4632: Modeling and Simulation

Casey Hampson

February 11, 2025

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 plan to implement 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. Considerations of quantum mechanical phenomena such as quantum chromodynamics (QCD) are required to implement an accurate simulation. I outline the tools I plan to use, including LHAPDF, a tool to extract parton distributions data, Gnuplot for plotting data, and GSL for mathematical subroutines, all of which will be implemented in C++. I conclude with a descripion of the model/file structure of the project and a basic timeline for what can be done next.

1 Introduction

In Sec. 1.1 and Sec. 1.2 I briefly describe the motivation for wanting to simulate proton-proton collisions and the theory behind the considerations that must be taken to model such a process. In Sec. 2, I give a brief literature review and discussion of the mathematical structure underpinning these types of simulations, as well as some initial basic results and comparisons between MadGraph5. In Sec. 3 I give a description of the model structure as it is currently laid out, as well as the external tools and programming language I will use. Lastly, in Sec. 4 I will give an overview of the current timeline and what things will be implemented next.

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 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.

An example of one such plot is given in Fig. 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 invariant mass of the four leptons were recorded and plotted. The normal production of the two Z bosons is shown in red, and the process involving the preceding 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 invariant 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 after 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 in the same as well as adjacent contexts. 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, all the way to simulating the detector structure and how it behaves when the particles from the collision fly through it using this theoretical framework. We

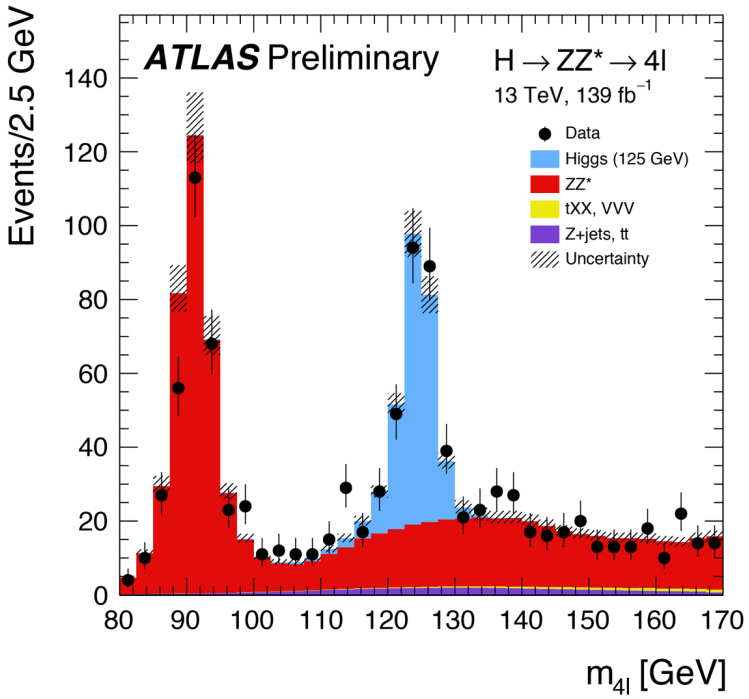

Figure 1: Invariant 4-lepton invariant mass of reconstructed events in the ATLAS detector for $H \to ZZ* \to 4\ell$ process, the famous plot signifying the presence of the Higgs boson.

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 nuclear 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 Simulation Considerations

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:

$$f(x) = a_0 + \lambda a_1(x) + \lambda^2 a_2(x) + \lambda^3 a_3(x) + \dots. \tag{1}$$

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 Main Goal(s) for the Project

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. Once these core processes are implemented, subsequent terms in the series corresponding to new interactions can be taken into account. Immediately at second-order, though, things get challenging (for instance, calculating the probability of an electron emitting and reabsorbing a photon evaluates to infinity), so only a few of the most important second-order process will be considered.

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. This is done via the Les Houches Event (LHE) file format [2], which I will describe later. 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.

As a first step, the main goal will be to produce results that are similar that produced by MadGraph5 for the hard scattering process and Pythia8/Herwig7 for the parton showering process (or, since the latter two programs can also do the hard scattering process, I may keep it simple and compare to only one for consistency's sake). This includes things like kinematical distributions of the "final-state" particles, such as

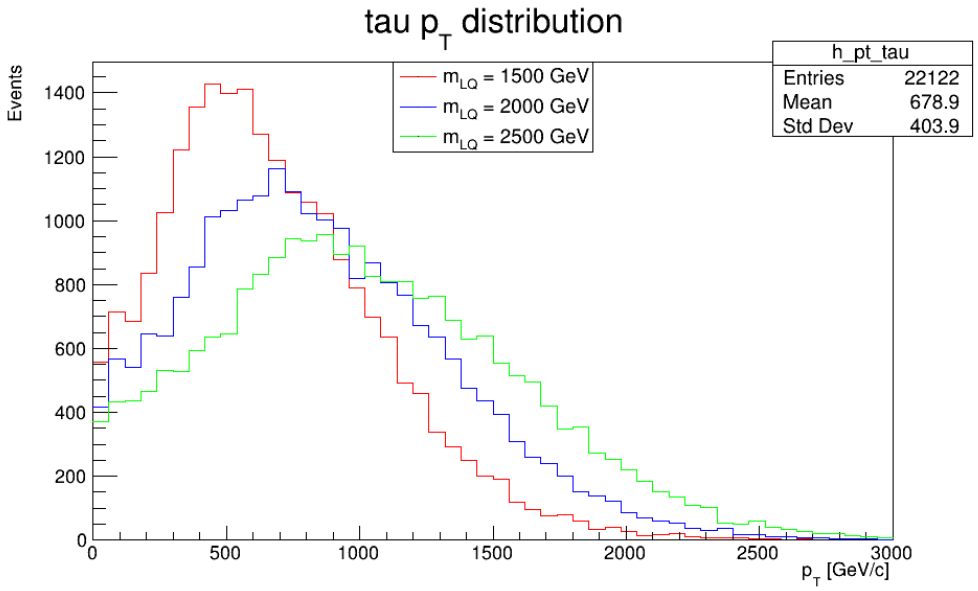

Figure 2: Distribution of intermediate-state τ lepton for a process that creates an exotic particle called the *leptoquark*.

transverse momentum, energy, and so on. For instance, an output of an event generator pipeline I worked with over the previous summer at my intership at CERN would look like Fig. 2. Essentially, the process I was simulating was considering the production of a possible new particle called a *leptoquark*. The leptoquark decays into a variety of particles including the τ lepton, and as a sanity check to ensure that my pipeline was working, I plotted the distribution of the τ leptons that were made as a result of the leptoquark decay. Such distributions are what I am attempting to create and compare with that of existing programs.

In principle, I'd like to be able to generate and shower events using my program, and have the results be comparable as if I did the same with just Pythia8/Herwig7, or if I instead generated with MadGraph5 and then showered with my program, or any combination of the event generators. If I am able to fully interop with these programs and produce sensible results, I will consider it a huge success.

2 Theory/Literature Review

Many of the individual things listed in this section are, in the land of computational physics, well known, meaning that many individual results aren't themselves profound results from any single paper. Rather, the entire suite of techniques presented in this section are all implemented to some extent

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

$$I = \int_{t_1}^{t_2} \mathrm{d}t \; f(t), \tag{2}$$

we can approximate the result by

$$I \approx \Delta t \, \langle f(t) \rangle = \frac{\Delta t}{N} \sum_{\tau=1}^{\infty} f(\tau_i), \tag{3}$$

where the τ_i lie in the range (t_1, t_2). At this stage, we simply *generate* τ_i's repeatedly, and the approximation will converge to the result. In practice, we can randomly generate a number in the range $(0, 1)$, then scale it to the domain, something like

$$\tau_i = (t_2 - t_1)\rho_i + t_1, \tag{4}$$

where ρ_i is the random number.

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; e.g. the answer cannot be expressed in closed form.

Considering the physical process of the proton-proton collision, as mentioned before, the starting point is the hard scattering process, which occurs immediately after the collision. We make one assumption (admittedly a highly non-trivial one, but one that I have nowhere near enough time to explain) that at this energy scale, the physics related to the interaction amongst the quarks is independent of how the quarks are structured within the proton itself. In other words, we only care about the fact that the individual quarks carry only a fraction of the total momentum of the proton, and we can discard any other piece of information related to any other hadronic structure. With this assumption in mind, we can then write an expression for the *cross section* σ for proton-proton collisions. The cross section is, roughly speaking, the probability for a certain event to occur. It has units of area, so another way that this can be thought of is how much of a cross-sectional area the target provides to the other object (this is actually pretty much exactly what the cross section is). Intuitively, this still is roughly a probability: if the target is larger, i.e. the cross section is higher, there is a larger chance for the other particle to hit the target. The full expression is given as:[1]

$$\sigma(p(P_1) + p(P_2) \to X + Y) = \int_0^1 \mathrm{d}x_1 \int_0^1 \mathrm{d}x_2 \sum_i f_{f_i}(x_1, \mu^2) f_{f_j}(x_2, \mu^2) \cdot \hat{\sigma}(f_j(x_1 P_1) + f_i(x_2 P_2) \to X). \tag{5}$$

Here, P_1 and P_2 are the incoming momenta of the protons p, X is the immediate result of the scattering of the two partons f_1 and f_2, Y is the generic hadronic state (which comes later), x_1 and x_2 are the momentum fractions that the parton carries with respect to its containing proton, and μ is an energy scale chosen to be characteristic of the reaction. The functions f_{f_i} are the parton distribution functions mentioned earlier, and give probabilities that the parton f_i will be found in the quark with momentum fraction x_i; $\hat{\sigma}$ is the *partonic* cross section, which contains information related only to the interactions between the partons themselves. As I mentioned earlier, we have *factorized* the information related to proton structure, that being the PDF factors, apart from the partonic structure, given by $\hat{\sigma}$.

The partonic cross section itself is given by the following integral:

$$\mathrm{d}\hat{\sigma}(f_i(x_1 P_1) + f_j(x_2 P_2) \to X) = \frac{1}{2\hat{s}} |\mathcal{M}_{f_i f_j \to X}|^2 \left(\prod_{k=1}^{N} \frac{\mathrm{d}^3 p_k}{(2\pi)^3 2E_k} \right) (2\pi)^4 \delta^{(4)}(p_1 + p_2 + \sum_{k=1}^{N} p_k). \tag{6}$$

Here, N is the number of particles in the final state of the partonic subprocess, and $\hat{s} = x_1 x_2 s$ where s is the center-of-mass energy for the partonic subprocess. $\mathcal{M}_{f_i f_j \to X}$ is the *matrix element* for the process, and is specific to the chosen process. This is also where considerations of how to incorporate corrections at higher orders will take place, as it is this matrix element that we must calculate perturbatively. Everything past this term in the above expression is called the *phase space*, and for high N, we can see the high dimensionality of our domain.

Despite the main integral's intricacy, we can analytically calculate the individual components and then use Monte Carlo integration to get an approximate result. This is the main goal for the hard scattering process.

[1] This formula, along with many of the following general results, can be found in any book on quantum field theory; see, for instance, Ref.[3].

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 $M_{\max}$. 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

$$F(m^2) = \frac{1}{(m^2 - M^2)^2 + M^2\Gamma^2}. \tag{7}$$

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:

$$I = \int_{M^2_{\min}}^{M^2_{\max}} \mathrm{d}m^2 \, \frac{1}{(m^2 - M^2)^2 + M^2\Gamma^2}. \tag{8}$$

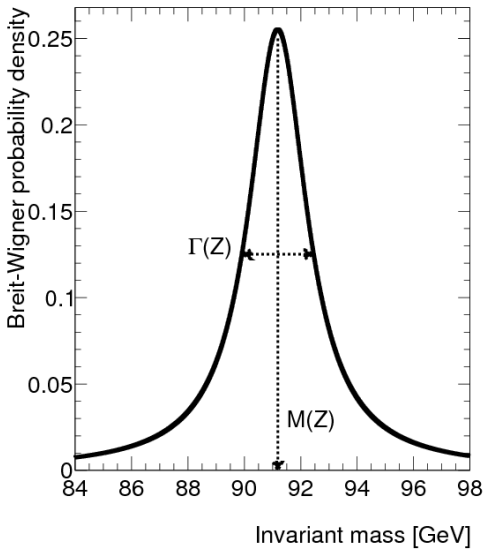

Figure 3: Example of the Breit-Wigner distribution.

An example plot is given in Fig. 3. 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:

$$m^2 = M\Gamma \tan\rho + M^2, \tag{9}$$

our integral turns into

$$I = \frac{1}{M\Gamma}\int_{\rho_{\min}}^{\rho_{\max}} \mathrm{d}\rho, \tag{10}$$

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. After this, we proceed to the parton showering part of the simulation.

2.2 Parton Showering/Hadronization Process

The calculation of hard scattering matrix elements and generation of events in accordance with the matrix elements is far more simple than the parton showering process, since the former is really an exercise in evaluating integrals with Monte Carlo algorithms. Parton showering is a little more based directly 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].

Once we have some reasonable particles from the hard scattering process, the next thing to consider is the possibility for these particles to radiate gluons/photons, called "parton showering." In the leading-order limit, which I am considering at first, this is all that is involved; I will not be considering subsequent decays/annihilations.

The main mathematical structure related to the emission of gluons/photons is the *Sudakov form factor*, defined like so:

$$\Delta(t_0, t) \equiv \exp\left[-\int_{t_0}^{t} \mathrm{d}t'\, \Gamma(t')\right], \tag{11}$$

where

$$\Gamma(t') \equiv \frac{1}{t'}\int_{z_-}^{z_+} \mathrm{d}z\, \frac{\alpha_s}{2\pi} P_{g\leftarrow q}(z). \tag{12}$$

Here, t' or t is a generic evolution variable, usually an energy scale, and t_0 is a cutoff scale, since if we get to too low of an energy, i.e. the gluon we emit is too closely collinear or soft, meaning low energy, our integrals begin to diverge. z is a momentum fraction, like x_1 or x_2 defined above, and the limits z_+ and z_- are usually determined from the specific kinematics of the process under consideration. α_s is the strong coupling constant, and $P_{g\leftarrow q}(z)$ is called a *splitting function* (specifically for a gluon splitting from a quark), defined like so:

$$C_F \frac{1+z^2}{1-z}, \tag{13}$$

where $C_F = 4/3$. The Sudakov form factor corresponds to the probability that the parton (quark in this case) *doesn't* emit a gluon as we evolve from the initial scale t to the cutoff scale t_0. This may seem backward, so to be sure: we are starting from the higher energy of the hard process t, and lowering the energy to the cutoff scale t_0.

So, one way that we can do the showering is by generating a random number in the range $[0, 1]$, and comparing it to value of the Sudakov form factor for the range $(t_0, t_{\max}$, where $t_{\max}$ starts at the initial scale of the hard process). If we find an emission, the scale that it occurs becomes the new $t_{\max}$. In case our random number is greater, then that means we will have an emission, so we numerically solve the equation

$$R = \Delta(t, t_{max}) \tag{14}$$

for t, which determines the evolution scale that the emission occurs at. Then, as I just mentioned, this becomes the new $t_{\max}$ and we continue the evolution until we reach our cutoff scale. 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 from $\Gamma(t)$ defined above.

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*. 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 "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

$$\hat{P}(z) = C_F \frac{2}{1-z}, \tag{15}$$

and this function is more easily invertable.

In terms of actual functioning parts from within the algorithm that would be directly implemented in the code, we would be considering solving for t (assuming a random R greater than the value of the Sudakov form factor) from $R = \Delta(t, t_{\max})$. After doing some rearranging, we find the function

$$E(t) = \log \frac{t}{t_{\max}} - \frac{1}{\rho} \log R, \tag{16}$$

where

$$\rho = t\hat{\Gamma}(t), \tag{17}$$

where the hat indicates the overestimated quantity. The values of t that we are interested in involve $E(t) = 0$, so we can employ traditional root-finding algorithms. One thing we can do to make it more efficient is use the entire logarithm term as the independent variable, then exponentiate and multiply by $t_{\max}$ to obtain t, rather than trying to make the root-finding algorithm do it. To find z, we can do some more rearranging to find

$$z = \tilde{\rho}^{-1} \left(\tilde{\rho}(\hat{z}_- + R'[\tilde{\rho}(\hat{z}_+) - \tilde{\rho}(\hat{z}_-)] \right), \tag{18}$$

where

$$\tilde{\rho}(z) = \int^z \mathrm{d}z \, \frac{\hat{\alpha}_s}{2\pi} \hat{P}(z), \tag{19}$$

and again, the hats indicate the overesimated quantities.

To recap: we generate random numbers and compare to the Sudakov form factor to determine whether an emission occurs or not. If one does occur, we numerically solve for the value of t using Eq. (16). We generate the gluon and give it a momentum fraction z via Eq. (18). We then accept this new event based upon the ratio of the original to the overestimated quantities. Then, the scale t that the emission occurred at becomes the new $t_{\max}$, and we continue until we reach our cutoff.

We are then left with some number of emitted gluons along with corresponding momenta and other kinematics. In higher-order limits, these would potentially scatter with other gluons or quarks, or possibly decay into quark/anti-quark pairs, and so on. After this, they would hadronize.

2.3 Some Basic Results

At the time of writing, I have implemented the numerical form of the partonic cross section for the process $q\bar{q} \to Z/\gamma \to \mu^+\mu^-$. In word form, a quark/anti-quark pair interact via either a Z boson or a photon (in this case we will consider the Z boson), and create a muon/anti-muon pair. The muon is roughly the same as the electron in terms of physical properties with the one exception that it is significantly heavier: the electron has a mass of 0.511 MeV and the muon has a mass of 105.66 MeV. The diagrammatical representation

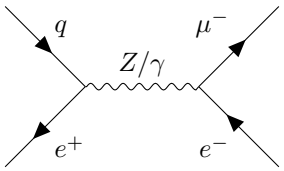

Figure 4: Feynman diagram for the process $q\bar{q} \to Z/\gamma \to \mu^+\mu^-$.

is given in Fig. 4, which is called a *Feynman diagram*. Time is read left-to-right, and while this pictoral representation doesn't fully encode exactly what physics happens, it is a good way to visualize it, and there are rules associated with them that allow one to go from the diagram to the exact mathematical expression for the cross section.

The partonic cross section is given by

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\cos\theta} = \frac{2\pi\alpha_s}{4\hat{s}}[A_0(1+\cos\theta^2) + A_1\cos\theta], \tag{20}$$

where we have the following definitions:

$$A_0 = Q_f^2 - 2Q_f V_\mu V_f \chi_1 + (A_\mu^2 + V_\mu^2)(A_f^2 + V_f^2)\chi_2, \tag{21}$$
$$A_1 = -4Q_f A_\mu A_f \chi_1 + 8A_\mu V_\mu A_f V_f \chi_2, \tag{22}$$
$$\chi_1(\hat{s}) = \kappa\hat{s}(\hat{s} - M_Z^2)/((\hat{s} - M_Z^2)^2 + \Gamma_Z^2 M_Z^2), \tag{23}$$
$$\chi_2(\hat{s}) = \kappa^2\hat{s}^2/((\hat{s} - M_Z^2)^2 + \Gamma_Z^2 M_Z^2), \tag{24}$$
$$\kappa = \sqrt{2}G_f M_Z^2/(4\pi\alpha_s). \tag{25}$$

Here, Q_f is the charge of the initial quark, the V's are constants determined based on the type of the particle with the f subscript corresponding to the incoming quark and the μ subscript corresponding to the final state muons, $\hat{s}$ is the center of mass energy of this hard process, Γ_Z is the decay width of the Z boson which is related to its lifetime, and M_Z is its mass. Lastly, G_f is the Fermi constant.

With some redefinitions of the kinematic variables present in the formula for the cross section given in Eq. (5), we can write

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\tau\mathrm{d}y\mathrm{d}\cos\theta} = \sum_{q,q'} f_q(x_1 = \sqrt{\tau}e^y, Q = \hat{s} = \tau S) f_{q'}(x_2 = \sqrt{\tau}e^{-y}, Q = \hat{s} = \tau S) \cdot \frac{\mathrm{d}\hat{\sigma}}{\mathrm{d}\cos\theta}, \tag{26}$$

where S is the center of mass energy of the two protons. We can perform this integral via Monte Carlo integration as described above, by randomly sampling values of τ, y, and $\cos\theta$. The actual expressions for the domains of these variables takes a bit of work since some are dependent on other variables, so I won't include them; the basic principle should be apparent. After running the code, we get a cross section of

$$\sigma \approx 1675.1090 \pm 5.9439\text{pb} \tag{27}$$

where the unit pb is the picobarn, a unit of area. Running MadGraph5 with as close to the same settings as possible gets us a value of 1684.0 ± 1.3pb, so we can tell already that our leading order approximation works very well, as does the program in general.

3 Model/Implementation Details

3.1 Programming Language

I plan to program this in C++, as that is one of the languages that I am more familiar with (along with C), and it is also one of the main languages that is used in physics nowadays. Further, interpreted languages like Python usually end up being too slow for intense simulations like this. For this project it likely would

have been fine, especially if I was able to implement anything on the GPU, but C++ has GPU programming toolets anyway (like CUDA), though a bit more cumbersome to use. Further, the existing tools I plan to use for my project have their main API in C++ with the Python version usually being more of a side-thought, so it is more favorable in that regard as well.

3.2 Other Tools and Frameworks

There are two external tools that I know I will be using, and those are the 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 *parton distribution functions* (PDFs). These are used to describe the structure of the proton, specifically the probability of finding different quarks within the proton given some momentum fraction x. This has to be done this way, by which I mean interfacing with some external package, because the PDFs are not able to be calculated on their own; they have to be determined from experiment and encoded in data file. This is required in the calculational side of things. There is one caveat with this library, and it seems that it can only be built on Unix-like distributions.

3.2.1 The GNU Scientific Library (GSL)

There are a number of mathematical routines that are required to implement this simulation. There are few enough that I would likely be able to program them on my own, but I would like to avoid spending too much time on such subroutines since they are not the main focus of the project. Additionally, I'd like any of my naive implementations of the algorithms to not interfere with the main simulation process. Therefore, at least at first, I may rely on the GNU Scientific Library (GSL). This is C library providing a large number of subroutines, most importantly numerical integration, root-finding, and others. It's a moderately large library, and as I'd only be using a few of the subroutines, I'd like to eventually phase it out with some of my own implementations of the algorithms. Monte Carlo integration, for instance, is something I've already implemented. However, like I just mentioned, at least for now it will be a dependency so I can focus on the core functionality.

3.3 Model Structure/Diagrams

UML diagrams will be shown shortly, but I first give a prose version of how the model will be laid out and how the code is structured. Inspiration is taken from the Pythia8 library. The name of my own library will be called ColSim, short for Collision-Simulation.

The basic directory structure for the project is organized like so:

```
CMakeLists.txt
Source
    CMakeLists.txt
    Core
        CMakeLists.txt
        inc
            ColSim
                Core
                    ...
        src
            ...
    Math
    Physics
    Plotting
    ...
res
    config.in
    manual
        ...
```

```
share
    cmake
        FindLHAPDF.cmake
        FindGSL.cmake
Tests
    CMakeLists.txt
    Test1
        CMakeLists.txt
        main.cpp
```

From the root directory, there is a global CMake file to which the other folders are added as subdirectories. The `share` folder contains other CMake files used to aid in finding the host system's versions of LHAPDF and GSL. The `Tests` contain at the moment only one test equipped with a main function with which to test the functionality of the main program. The `Source` contains all of the subdirectories of the project, where different functionality can be contained. For instance, all functionality related to the interface with Gnuplot is contained within the `Plotting` directory. Each of these directories are equipped with a CMake file, and include, and source directory in which headers and C++ source files are located. Lastly, back in the root directory of the project is the `res` folder containing the config file and some other files related to the project documentation contained in the `manual` subdirectory.

In terms of the actual model implementation, the basics start in the `Core` subproject, in which all of the core functionality that is not dependent on anything else but the standard library or other Core elements is implemented. For instance, the logger that logs to standard out/err and/or a dedicated log file is stored here, as well as some utility functions for strings and vectors which, for some reason, aren't present in the STL. Most importantly is the class `ColSimBase`, which contains a static pointer to the logger and other things like settings/options specified within the configuration file. Every other class that I make in the rest of the project will inheret from this class so that they have access to these things.

The one exception is the `Math` subproject, which contains all of the mathematical subroutines used in the simulation. These function signatures take inspiration from GSL. Since they are purely mathematical and have no relation to any specific physical process, they don't need to inherent from anything.

The `Physics` subproject is the most important, and contains all of the physics-related code relevent for the simulation. For instance, all physical constants are defined in a header file in this subproject, as well as classes for all of the different types of processes, along with the master `ColSim` class used to actually create the process, read in the configuration file, and execute the relevant subroutines.

As just mentioned, each process has its own class, inhereted from the main `ProcessBase` class, which provides members and functions that every class should use such as methods to determine phase space points and methods to simply return the evaluation of the partonic cross section at a specified phase space point. The full parton-shower core functionality has yet to be implemented, but it will follow a similar style, where each type of process will be sectioned off into its own class that inherents from some overall `PartonShowerBase` class.

Also contained in the `Physics` library are classes for kinematics such as the `FourVector` class, from which specific four vectors like the four-momentum and four-position are implemented. These contain some of the kinematic information of each particle.

Once the event generation starts and the cross section is computed, events will be created and stored in the `Event` class, which will contain a record of all the particles, represented by the `Particle` class. Each particle will contain its own momentum, any other necessary kinematical variables, as well as identifying information that ties it to its event. For emitted particles during the parton showering step, additional information will need to be contained tying the emitted particle to its "parent" particle.

3.4 Graphical Diagrams

Given in Fig. 5 is the current status of the program in terms of the interactions/relationships between the different classes expressed as a UML diagram, with the exception of the `Math` subproject, since, as mentioned earlier, it is separate from the physics processes. This is simply a graphical representation of what has already been described, so I won't discuss much about it, apart from the fact that some classes don't have fully fleshed out members/methods. The current results of the progam, for instance, were retrieved via directly using the

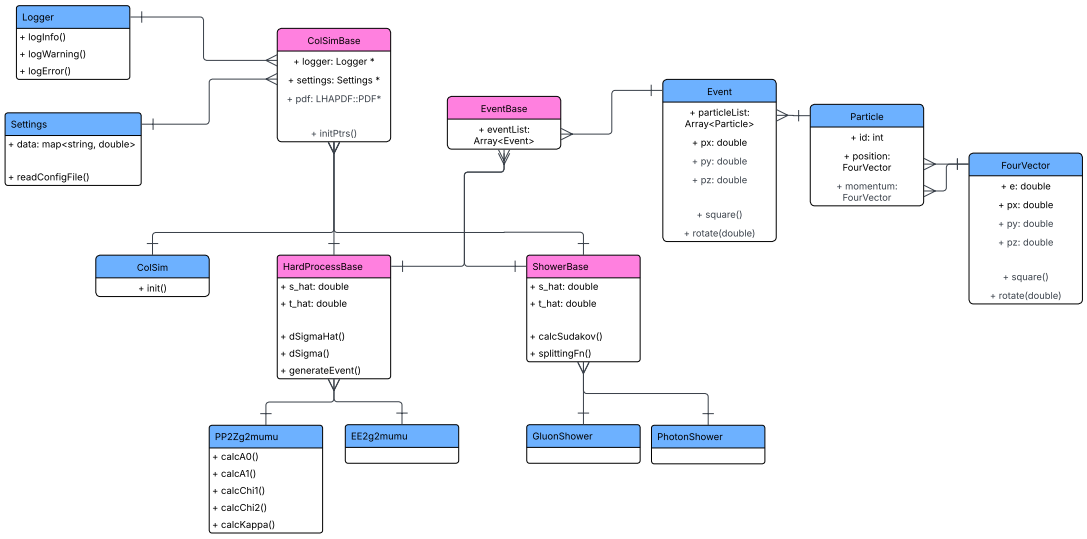

Figure 5: Basic UML diagram describing the interaction/relationships between the different classes.

methods in the `PP2Zg2mumu` class without first going through the main `ColSim` class. One more processes start being fully implemented, this more object-oriented structure will start to take hold.

3.5 Current Challenges

At the moment, one of the largest challenges is that all of the documentation on the internet for event generators like these is either extremely low-level, spanning hundred of pages and often jumping right to NLO or NNLO implementations of things, or it is extremely high-level, where it is formed almost like a tutorial of sorts, and the end product is the implementation of one single process at one order with a very small subset of observable results. These tutorials have provided a fantastic source of some basic implementation for some specific processes, but any sort of generality is left to me.

Fortunately, there is a professor here at KSU who is quite familiar with these tools, and has actually worked quite a lot on Herwig7. He works in an adjacent department to the one I do research in, and I have had two classes with him and had many discussions on things after/before class, so he will be a very approachable resource for information on this topic.

4 Timeline

The core timeline remains the same: the literature review and knowledge of the physics behind what I'm implementing is taking by far the longest time. At this stage, I am starting to highly regret choosing such a complicated process to model, despite it being phenomenally interesting to me. Despite this, I largely understand the core components behind the hard scattering process and the parton showering, at least to leading-order. This roughly corresponds to the Monte Carlo integration for the former and the Sudakov form factor for the latter. If I implement each individual process separately, then all I will need to do is finish the object-oriented structure and plotting of results, and I will have a functioning program. This is far more simple than it sounds, but it is accurate.

The immediate next step is to finish the basics behind the parton showering component. After this point, I will likely approach the aforementioned professor, Dr. Papaefstathiou, and inquire further about a few specifics, during which I will implement the partonic cross sections for a few other processes.

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] J. Alwall et al., "A standard format for les houches event files," *Computer Physics Communications*, vol. 176, no. 4, pp. 300–304, Feb. 2007, ISSN: 0010-4655. DOI: 10.1016/j.cpc.2006.11.010. [Online]. Available: http://dx.doi.org/10.1016/j.cpc.2006.11.010.

[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.