Project Proposal
CS 4632: Modeling and Simulation

Casey Hampson

January 23, 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 experiement 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, and Gnuplot for plotting data, all of which will be done in C++. I describe the basic model structure including the representation of events and particles via their own classes, along with the implementation of the required mathematical subroutines. I then give a brief timeline of when these things are expected to be completed.

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 describe the methodology and implementation strategies I plan to use, including the programming language of choice, external tools, as well as a very brief overview of how I plan to implement the model in code. In Sec. 3, I roughly estimate the timeline of when parts of this project are completed.

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.

It is often 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 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.

1.2 Simulation Considerations

As one would expect, though, simulating these processes is immensely challenging. 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). 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.

These effects manifest during a collision, where quarks will fly apart due to the immense amount of energy used to accelerate the protons but due to the nature of their interactions via QCD, once they get far enough apart, they will *hadronize*, by which two new hadrons are formed. These can further collide with other quarks or hadrons, and also decay into other particles (potentially quarks again!) if they are unstable. This creates a multitude of essentially simultaneous processes that must all be considered. The name given to this part of the simulation after the hard scattering is called "parton showering" or "hadronization."

To make matters worse, in particle physics (and often physics in general), we will never be able to calculate the full answer to any given problem. This isn't an exaggeration; it's not just that the equations are super long (see [1] for instance), it's that the best way we know how to formulate things is in terms of an *infinite* series of terms. In most cases, fortunately, these terms get significantly smaller to the point that we can get extremely reasonable estimates after only 3 terms. So, we can calculate a handful of terms, say 3, then just ignore the rest and have a very nice estimate. Unfortunately, calculating subsequent terms gets exponentially harder, and entirely new mathematical methods often have to be invented to solve the new class of problems that the next order imposes. Further, these new methods have to be implemented in code, and it must be fast, since we don't have years to wait around for a computer to go through calculations. As a note, this concept of writing things in terms of an infinite series of solutions is called *perturbation theory*.

1.3 Main Goal(s) for the Project

For the purposes of my project, I will focus on leading order calculations, meaning I calculate only the first term of the aforementioned infinite series, making my life significantly easier, and providing still decent approximations. If I have time, I will consider how to implement some next-to leading order (NLO) components, but I reckon that will be very challenging. Further, there are few if any programs out there that actually do the entire process from hard scattering to detector simulation all in one. For instance, MadGraph5 only focuses on the hard scattering; Pythia8 focuses on the parton showering (though it can do hard scattering as well); and Geant4 focuses on simulating the detector. So, in my program, I will focus on the hard scattering and parton showering processes, most similarly to Pythia8, and leave out detector simulations and reconstruction algorithms.

The goal is to be able to make distributions of some basic kinematic variables such as (transverse) momentum, energy, and so on, and have it be comparable to the output of Pythia8 and others given roughly the same set of inputs. These plots will be of the final-state, stable particles after the hadronization has finished that would then fly through the detector.

Even though this is technically already an expectation for the project, I plan to have a sophisticated "housekeeping" structure including logging, data storage, and general information storage so that the user has access to everything in a readable format whenever they want. Further, as I will discuss a bit more in the implementations section, I will also have some sort of communication with some other framework such that I can draw plots to the screen, like Gnuplot. In principle, the plots and various other generated data should be compatible and comparable with the other simulation suites out there, most specifically to Pythia8. One way this can be done is by storing my results in the Les Houches event file (LHEF) format, which is a universal text-based XML-style file format recognized by most simulation suites in use today (see [2] for a super brief description, particularly section 4). These file formats allow direct translation from one simulation suite to another in many cases. For instance, I could generate data corresponding to the hard scattering process with MadGraph5 in the LHEF format, then pass it directly to Pythia8 to do the parton showering simulation. My point is that one of my goals is to have such a file be the output of my program, among various plots and such.

2 Methodology/Implementation

2.1 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.

2.2 Other Tools and Frameworks

I would really like to interface with ROOT, a tool used by CERN for data processing, but it is an absolutely massive package, weighing it at several gigabytes after unpacking and compiling. Since I'd only use it for a small subset of its data storage and histogram functionality, I will instead implement my own method of data storage and interface with Gnuplot, a much more lightweight package intended only for plotting things.

I will also use LHAPDF, 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. I could, in princple, make my own parser for these data files, but that would be besides the point of this project and would likely take too long anyway.

2.3 Basic Model Approach/Design

I currently only have a rough idea of the model, since I am very busy with delving into papers, articles, and documentation of similar simulation suites like Pythia8 ([3]) to wrap my head around the process and the physics behind it; it's a bit more challenging of a process to model than something like the carwash, so I haven't had time to really formulate much of the model itself yet.

My current idea is that I will model each event that occurs with an `Event` class, and it will contain information related to the number of particles that were involved; the energy of the collision or decay; if it was a decay, or any process by which the states of the particles changed in any way, it will store that as well as energy absorbed/released, and many other kinematic variables. Each particle will also have its own class, with information specific to that particle as well as its role in the process and each event. Different types of particles, like the leptons, quarks, and bosons, will likely have their own class, perhaps inheriting from some central `Particle` class.

Particles and events will have methods that interface with either a class with static methods or a namespace of functions in which I implement all of the mathematical subroutines required for determining quantities of interest like energy and so on. This separate translation unit will be for the purely mathematical functions that have no particular relation to event generators such as interpolation routines and so on. Other mathematical functions relating specifically to event generators like Monte Carlo integration routines will be included via a separate class/static methods or namespace/functions that is more closely tied to the simulation process/class.

During all of this, I will be storing all of the data for each individual event, taking into account many of the things present in the event class. This may take the form of a method in the event class called `store()` or something, where it consolidates the information about the event and places it in another class I'll probably call an `Ntuple`. This is the name of the class that ROOT uses to store floating point data. Further, there will also be logging going on, where each event will trigger the main process class to output information regarding the status of the simulation to the console and/or some dedicated log file.

Lastly, once all of this has finished and data has been consolidated in any number of ntuples, I will interface with Gnuplot to plot the data. Gnuplot, in particular, is primarily a command-line tool, so I will likely create some class structure to generate commands and data files and then fork a process that calls

Gnuplot with those commands (that terminology may not be correct, but hopefully it captures my point). Also at this time data will be consolidated into an LHEF file.

2.4 Graphical Diagrams

As of writing, I have not formulated enough parts of the project to confidently put together any sort of UML or other graphical diagram to illustrate the model structure.

2.5 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. Of course, I will be unable to extensively use and rely on the latter documentation, and I will have to continue searching as well as dissecting the former documentation for clues on how to proceed.

Fortunately, there is a professor here at KSU who is quite familiar with these tools, and has actually worked quite a lot on an event generator and parton showering program called 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.

3 Timeline

I reckon that the initial phase of literature review and model formulation will take a substantial amount of time considering the relative complexity of the process I intend to model and simulate. As I mentioned above, The relevent documentation is very extensive, and it will take time to extract only the relevant components that are feasible to implement in this project. As a rough estimate, this will take a few weeks to complete.

In the mean time, I plan on implementing all of my so-called "housekeeping" functionality, since it is largely independent of how I choose to implement the model. I have already completed a system for logging information to the console and a dedicated log file, whose purpose will be mostly for debugging at first, but once the simulation has been implemented it will serve as an extensive record of the steps the simulation took during its execution. By the time the literature review is done, I will plan on having the data storage and Gnuplot interface completed, having tested it with fake data. This will have to be a bit more fluid though, since it will have to account for the actual output of the simulation and which variables I end up choosing to implement.

Having this implemented early will be a huge help, particularly the debugging/logging, as I move forward into implementing the actual physics simulation, since it will hopefully allow me to largely avoid some nasty bugs by being able to constantly visualize what my program is doing. The actual implementation of the model, once I have it designed, should hopefully be very straightforward. Several mathematical routines that I imagine I will need to use I have already programmed at some point in the past (though in either Python or C), and if not, I have extensive resources regarding how to implement them.

References

[1] P. Mathews, V. Ravindran, K. Sridhar, and W. van Neerven, "Next-to-leading order qcd corrections to the drell–yan cross section in models of tev-scale gravity," *Nuclear Physics B*, vol. 713, no. 1, pp. 333–377, 2005, ISSN: 0550-3213. DOI: https://doi.org/10.1016/j.nuclphysb.2005.01.051. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0550321305001045.

[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] 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.