12th July 2017

Monte Carlo and Jet Tutorial

Samuel Meehan[a]

[a]*University of Washington*

The use of numerical simulation and Monte Carlo techniques is ubiquitous in the field of high energy particle physics. This tutorial is meant to introduce students to these concepts in the context of studying hadronic jets at the Large Hadron Collider. It is written with the intent of being a self-guided introduction to this topic and more-over, introduce undergraduates and young graduate students to the topic of research and data analysis in high energy physics.

1 Introduction

One of the most commonly used tools to construct models in high energy physics (HEP) is Monte Carlo (MC) simulation. This form of model building is applicable from the description of fundamental processes to the behavior of large detector systems and electronics. The reason for using Monte Carlo simularion is rooted both in the inherent random nature of the systems under study as well as the ability to describe a complex system by the composition of many smaller processes.

There is extensive literature concerning the design of various types of Monte Carlo models, and extending what you learn here to those applications is your responsibility. In this tutorial, an introduction to MC modelling is explored in the context of describing the physics of *jets* [1, 2] and the Large Hadron Collider (LHC) [3] and how we study them using detectors called calorimeters [4].

This work is intended to be used as an educational tool by teachers and curious students who are interested in either the physics of the Large Hadron Collider or the topic of Monte Carlo simulation. The text should be viewed as a means of guiding your understanding with portions devoted to focused exercises that build upon each other. As the tutorial progresses, less explicit explanations of how to perform exercises are given such that you are encouraged to interpret and determine the best course of action as in a research context. It is strongly recommended that the interested individual diligently follow all portions of the tutorial and carry out all exercises to have the full benefits of the work. Typically, this takes between two and three weeks for an average student with an undergraduate level background in physics with periodic supervision by an expert. At the end of the tutorial, there are suggested project extensions that can be used as full projects in the context of a course or to begin research on a focused topic of your own.

2 What is a Jet?

The LHC is a particle accelerator designed to collide protons at extremely high energies, where the kinetic energy of the proton is ~ 1012 times as great as its rest mass energy. At such high energies, when two protons collide, the interaction that occurs takes place between the parton constituents of the proton (i.e. quarks (q) and gluons (g)). When such a collision occurs, a large number of processes may result. The exact process that occurs depends on the configuration of the initial state of the collision from proton-1 and proton-2 (qq, qg, gg) and the energy of each of these partons[1]. However, there are other aspects of the collision process that proceed differently on a collision-by-collision basis even if the initial state of the collision is the same. In some cases, the incoming partons may produce a Z boson or a photon. In other cases, it may produce a Higgs boson which decays to two photons. It is precisely this random nature of what is produced that is the aim of the type of quantum physics being studied at the LHC and is described by what is called quantum field theory.

In quantum field theory, the exact process which occurs is unpredictable but will almost assuredly be accompanied by the associated production of additional quarks and gluons. Unfortunately, describing the behavior of quarks and gluons is a sub-field of quantum field theory, called quantum chromo-dynamics (QCD), that is very challenging, primarily because the strength of the interaction is very large [5]. For those performing theoretical calculations, this strong interaction means that the standard techniques used

[1] Though these may seem fixed by the choice we make to collide protons at a fixed energy, there is still an inherent degree of randomness that is taken into account when conducting real simulations which is explored in one of the extensions of this project in Section 7.

to describe the dynamics of systems that involve quarks and gluons cannot yet be described with the same level of precision that we can describe the interaction of electrons and photons, their electromagnetic counterparts. For experimentalists working at the LHC, the strong interaction means that the production of a single quark or gluon, which often occurs in conjunction with something more interesting like the production of a Higgs boson, does not lead to the detection of a single particle in the detector. Instead, the first quark or gluon that is produced will itself spawn the production of more quarks and gluons which themselves will spawn more quarks and gluons eventually leading to the creation of a spray of partons. This is called the *parton shower* and is represented by the final state portion of the diagram in Figure 1 until the gray ovals. This process will continue until the energy of the first parton has been subdivided so far that the large set of partons that is has spawned are low enough energy that it is energetically favorable not to spawn more partons but rather to bind together into stable composite particles, hadrons. This is called *hadronization* and is represented by the gray ovals in Figure 1 which creates stable particles, the yellow circles on the final state. These hadrons are what fly into the apparatus constructed around the collision point and are detected in what are called calorimeters[2] and are observed s a collimated spray of particles called a "jet".

It is precisely these two processes, the parton shower and hadronization, that make the study of quarks and gluons challenging at the LHC because the production of a single parton will actually appear in the detector as many particles, collimated in the same generatl direction, all arriving at once. The detection of a collimated flow of particles of this nature is what is called a *jet* [1, 2]. Moreover, because calculating the behavior of quarks and gluons in QCD is challenging, there is not even an exact prediction of what such a "jet" of particles *should* look like. Luckily, it is possible to understand the behavior of the parton shower in extreme limits of QCD on which to base a model for how additional partons are created. Likewise, it is also possible to use similar guiding principles to build a model of how these will eventual bind together into stable hadrons. In both cases, the model that is used will be simpler than the exact and parameterized in terms of an set of parameters that we choose. Finally, and most importantly for the purposes of this tutorial, the models used for these two processes both have degrees of randomness resulting from the fact that they describe a quantum process which allows them to be described by a Monte Carlo model.

2.1 The Parton Shower

In LHC collision events, single partons are produced by one process or another and emitted from the hard scatter. These partons carry what is called *color charge* and therefore interact with other *colored* particles via the strong interaction. However, unlike other fundemantal particles and interactions, single partons that can interact via the strong force cannot be isolated and observed by themselves [5]. Only composite groups of partons whose color charges cancel and form a *colorless* particle are stable. Such composite particles are called *hadrons*. Therefore, the initial parton that is emitted will gradually transform into a collimated flows of tens of stable hadrons, the ensemble of which deposits large amounts of energy into focused regions of the detector.

This evolution of a single parton to many partons, and eventually into a collection of hadrons is governed by the principles and described by the equations of QCD ... which are quite complex. In fact, they are so complex that at this point, it is impossible to fully describe with the equations of QCD. However, it can

[2] Even the behavior of the detector is somewhat random and modelled by MC simulation. How to model this is one of the topics to be explored in one of the extensions of this project in Section 7.

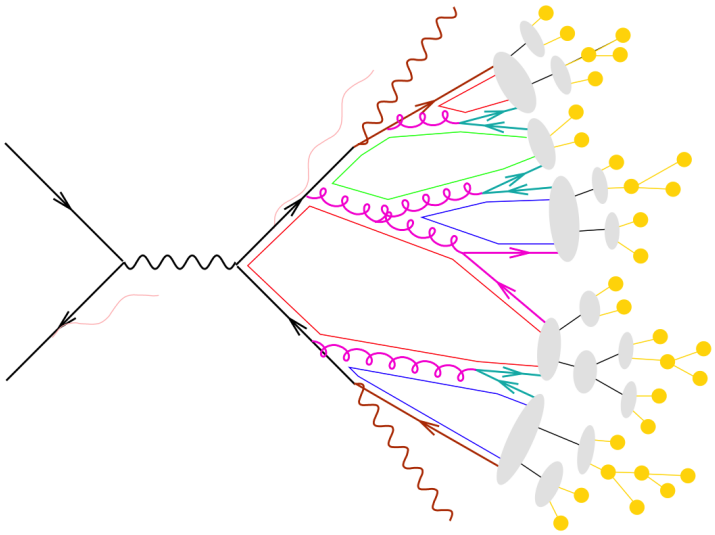


Figure 1: Illustration of the decay of a W or Z boson to a $q\bar{q}$ pair with the subsequent dynamics of the parton shower (red and purple lines) and hadronization (gray clusters and yellow circles) of the hard partons.

be shown that given an n leg Feynman diagram ($\mathcal{M}_n$), the $n+1$ leg Feynman diagram ($\mathcal{M}_{n+1}$) formed by including an additional contribution from a parton *splitting* ($a \to bc$), as shown in Figure 2, scales as:

$$|\mathcal{M}_{n+1}|^2 \sim \frac{1}{\theta^2} C_A F(z) |\mathcal{M}_n|^2 \tag{1}$$

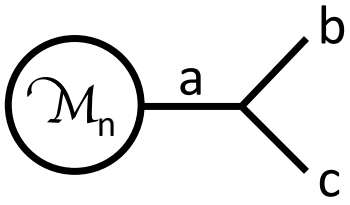


Figure 2: The Feynman diagram representing an $n+1$ leg matrix element process $\mathcal{M}_{n+1}$, build from an n leg process $\mathcal{M}_n$ and an additional parton splitting.

where $z = \frac{E_b}{E_a}$ is the relative energy of the more energetic parton after the splitting to that of the initial parton and θ is the opening angle of the splitting between parton b and parton c. In this model of a single emission, the parameters z and θ can be interpretted in a physically meaningful way and pertain to the evolution of the energy and spatial spread of the shower. However, in practice they are completely unobservable given that all of these dynamics occur within a distance scale that is less than the radius of a proton!

There are three key features that emerge from this equation that dictate the first order evolution of jets. The first concerns the function $F(z)$, related to the abc splitting vertex and depends whether the vertex is a ggg, qqg, or gqq vertex. For the gqq vertex, this function is well defined for all values of z but for the ggg and qqg vertices a singularity exists for an emission of $z = 0$ or $z = 1$, meaning that radiation coming from these splittings are preferentially low energy, "soft", and generally follow a functional form of $1/z$.

Second, the $1/\theta$ singularity common to all vertices implies that splitting are preferentially at small angles, "collinear". The third key feature is that in addition to the soft and collinear nature of the splittings, the splittings do not depend on the azimuthal angle around the axis of propagation of parton a. These three features lead to the intuition that the resulting partonic energy pattern coming from the initial parton that was produced will generally be azimuthally symmetric around the its initial direction with an energetic central core and some amount of less energetic radiation distributed further out.

2.2 Hadronization

The physics of the parton shower is modelled by a limiting case of the full model of QCD. However, when the energy scale of the parton shower becomes low enough, the same model of QCD, involving quarks and gluons interacting at vertices is no longer valid. Instead, it is necessary for the physical description to use composite particles, hadrons, to describe the physics. Therefore, it is necessary to make a transition from an ensemble of partons to an ensemble of hadrons. Understanding this transition is not well understood from first principles and describing it accurately remains an open question in particle physics. However, it is possible to model this transition using a heuristic model that is based on an intuition of the physics that is expected to be occuring. The root of this intuition is that partons are binding together to form stable particles in much the same way that electrons and protons, if placed together in close proximity, will bind together to form hydrogen. Like the parton shower, this model will have an inherent degree of randomness due to its quantum nature, as can be seen in Figure 1 where in some cases two partons enter a gray hadronization bubble and two hadrons exit whereas sometimes one enters and two exit or there is a multistep process that occurs until stable hadrons are formed. If this model is appropriately and degrees of freedom are chosen properly, then the randomness of the model can be tuned by choosing the parameters of the model such that *on average* the prediction of the hadrons from this model match that observed in data. In this way, the model of hadronization can be used to more fully describe the entire process of an LHC collision from the Feynman diagram to a complete set of stable particles.

There are two Monte Carlo models that are typically used to describe the process of hadronization. The first is called the Lund string model and is implemented in the PYTHIA [6] generator. The second is called the cluster model and is implemented in the SHERPA [7] and HERWIG [8] programs. Each focuses on different fundamental features of the physical intuition of the process of hadronization and therefore has different tuneable parameters. The details are left to one of the extensions in Section 7 with a simpler model presented in Section 4.2.

3 Monte Carlo Simulation Basics

The types of processes that contain random components are vast in terms of the exact models used to describe and study them. However, the basic principle used to construct a specific model is generic; decompose the model into discrete parts, define the behavior of each part, implement and tie together these parts. It is first necessary to decompose the process into discrete pieces and determine which components are stochastic and which are deterministic. This can be visualized as in Figure 3 in the case of population growth model [3] in which the initial value of the population P_I is propogated through 5 stages of the model to reach the final value of the population P_F. These five stages of population evolution are

[3] Note that this is not a real model of anything but may resemble a scenario from The Walking Dead and just used to demonstrate the concept of decomposing a model into pieces.

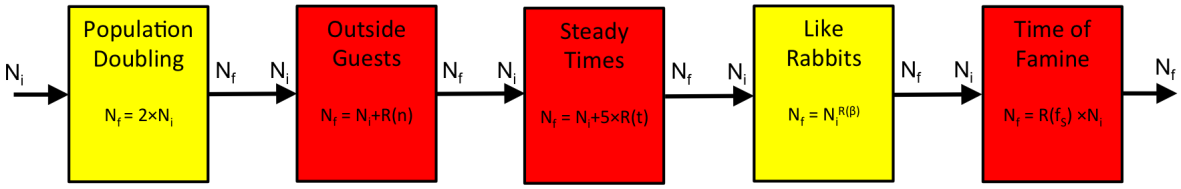


Figure 3: A pictorial representation of the layout of a population evolution model showing a number of deterministic (yellow) and stochastic (red) stages. The rule that transforms the incoming initial population N_i to the final population N_f at each stage of the process is indicated in the respective box and, as indicated, the final population from the i-th stage is used as the initial population for the $i+1$-th stage of the evolution.

1. Deterministic : Doubles the initial population because everyone in the initial group gets to choose one outside person to stay with them.
2. Stochastic : The population is augmented by adding some number of people showing up one day from some random group walking around. The number of people in this group is a random number whose probability distribution follows a Gaussian distribution of mean μ and width σ.
3. Stochastic : During stable times, the population grows at a linear rate of 5 people/day for some random amount of time t between 50 and 100 days where the probability for any given amount of time is uniformly distributed.
4. Deterministic : The population exponentially increases once from x to x^{β} because lots of people are told to have babies. The parameter β is a random number that ranges from [2,10] with equal probability of any value in that interval.
5. Stochastic : There is a famine and most of your population will not survive. The fraction of people f_S that are left after the famine is random and is some number between 0 and 1 and follows an exponentially decaying distribution ($e^{-\alpha x}$) where x is the fraction of people left and α represents the seriousness of the famine.

To proceed through each of the steps of the model of this process, the rule that is described is quantified and then applied to the existing state of the system, in this case the population. In some cases, the same thing will occur, in which case the rule is applied in the same way. In other cases, there is a stochastic element which necessitates for the application of the rule to determine an exact value for a random number. If the model being implemented calls for uniformly distributed random numbers, the manner in which these are obtained is straightforward [ADDREF] and it can always be assumed that the generation of an ensemble of numbers uniformly distributed between [0,1] can be obtained easily [4]. However, in some cases, the transformation that occurs in a stochastic stage of your model depends on random numbers that are not uniformly distributed. As such, it is necessary to be able to select random numbers distributed according to any arbitrary probability distribution function (PDF). Two common methods that can be used to achieve this are the *accept-reject method* and the *inverse transform method*. Both of these techniques are commonly used in high energy physics and described by the particle data group in Reference [9] but provided here is a short summary of them.

[4] In python, this is done via the `random.random()` and this is the only built in random generator function that should be used throughout the tutorial.

Practice writing a program in PYTHON that generates random numbers. In the first case, write a program to generate 1000 random numbers distributed uniformly between [0,1] and make a histogram of these numbers. In the second case, write a program that distributes the random numbers between [5,15] and makes a histogram. Remember, it is only allowed to use the `random.random()` functionality in PYTHON. As a hint, think about how to make the minimum and maximum of the [0,1] distribution into the minimum and maximum of the [5,15] distribution.

3.1 Intermezzo : Documentation and Version Control

You may be working through this tutorial with your main interest being in particle physics. That is great! However, much research in particle physics, in particular the kind you will learn here, involves data analysis through the use of programming. Therefore, to start off on the right foot and build good habits in your work, all of your work should be documented and version controlled on GITHUB. The reason that we do this, and use a service like GITHUB is to ensure two main things

1. "Sharing is good!" : Our code is well-documented and accessible to the outside world.
2. "Oh $*@T!" : If your hard drive crashes, you lose your laptop, or accidentally type `rm -rf /*` into the terminal, you are not screwed.

If this is a new concept to you, then do not skip this section. Instead, follow this Git tutorial and register for an account on GITHUB. Once you are more familiar with how to use GIT, another good tutorial can be found at this second Git tutorial. After that, all of the work that you share with collaborators should follow these simple guidelines :

- Do not email your code to people. If your code is worth sending to someone else, it should be accessible by sharing a link like https://github.com/smeehan12/PythiaSimulatorAnalysis in an email.
- Include a README file. This file should include (*i*) a description, (*ii*) a list of authors, (*iii*) a set of directions to follow to run the code, (*iv*) a description of the expected output from the code.
- Do not send untested code. Before sending your code to someone else to use, you should test the code as if you were them.

If you follow these rules and build habits, you will be a much more enjoyable individual with which to work.

3.2 Accept-Reject Monte Carlo

The accept-reject method, sometimes called the Von-Neumann method, can best be understood by examining a random process of making a measurement of a truly random physical system, that being a quantum mechanical particle in a box in a well-defined single Eigen state of the ground state energy as in Figure 4(a).

In this scenario, the measurement of interest which will be the randomly distributed observable is the position of the particle in the box x. In this case, the set of measurements that is produced should

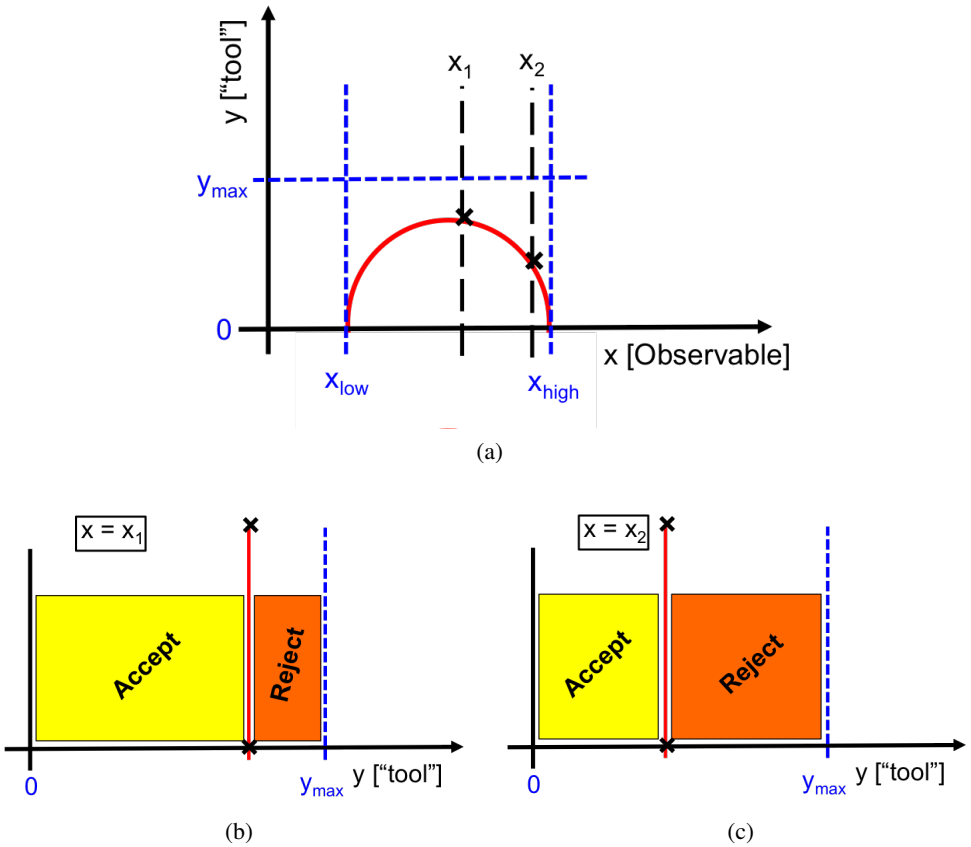


Figure 4: A cartoon representation of the implementation of the accept-reject Monte Carlo generation method to produce a single x measurement from the ground state of a particle in a box system in quantum mechanics. The ground state probability density function (left) with the observable bounds x_{low} and x_{high} and two example random x values x_1 and x_2. Also shown are the decision making tools for single generations of x_1 (middle) and x_2 (right) showing the y values for which the given x value is accepted and rejected.

be distributed according to the wave function of the system [5] which gives the probability of where the particle is, and when performing a measurement (i.e. draw a random value for x) it becomes a specific instance of that location. Finally, we wish to make many measurements and wish to obtain an ensemble of x measurement values. However, in order to generate a measurement the position of the particle, it is actually necessary to generate two numbers, both the observable of interest x and an auxiliary random number "tool" which we will call y. For the purpose of simplicity, for now you can generate them from uniformly random distributions distributed between 0 and 1. This imposes a few constraints on where the particle is by constraining x to be between [0,1] but an ensemble of such events will not be distributed according to the wave function. To obtain an ensemble of x values that are distributed according to that constraints, you need to be clever about choosing which instances of x to keep in the ensemble and which to not keep.

To achieve this, we will use y. The value y has nothing to do with physics (unlike x which represents the location of the particle in the box). Instead, y can be viewed as nothing more than a tool which allows you to either "accept" the event and collect it into part of your ensemble of events, or "reject" the event

[5] Actually, if the wave function is $\Psi(x)$ then they will be distributed according to the probability density $|\Psi(x)|^2$.

and think no more of that specific instance of the choice of x. You want to preferentially keep x values that are close to the center of the box but also sometimes keep events in the tails. To do this, think about taking a single slice in x near the tail as in Figure 4(b). Whenever you draw an x value here, you must draw a corresponding y value and compare it to the value of the function $PDF(x)$. Since its in the tail, its more likely that the y value chosen at random is larger than the function value. Therefore, if you were to throw away the x values here when the corresponding random y is greater than $PDF(x)$, you will only be left with a few x values with this value in your ensemble. However, if instead you were to be looking near the center of the distribution, as in Figure 4(c), then the situation is different. More of the time, $PDF(x)$ is greater than y so you are forced to keep a larger fraction of these events. But, the value of y that you choose must always be able to allow for the possibility of either accepting *or* rejecting a given x value. So the range of y values must be chosen to extend at least to the maximum value of the function. As such, if the functional form of the probability density function has a maximum value that is greater than 1, then the range over which the y value is chosen must be extended to encompass the full function. If it extends above the maximum value of $PDF(x)$ in the range of interest, that is fine, but then the overall fraction of throws of x that you accept will be smaller and your generation of accepted Monte Carlo events will be less efficient.

Another explanation of the accept reject method is provided in Reference [10].

Practice writing a program in Python that generates random numbers, distributed according to a Gaussian distribution with a mean μ of 5 and a width σ of 2 in the range of [0,10].

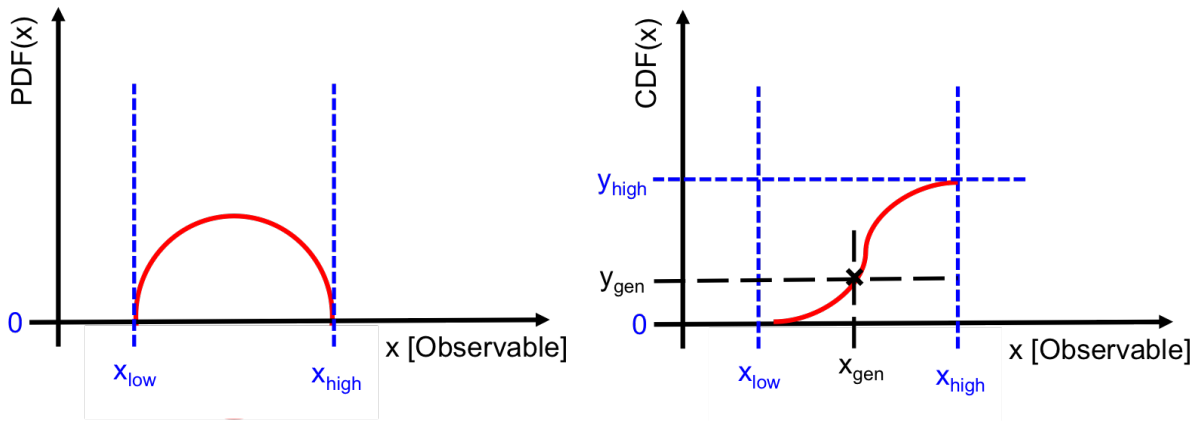
The accept-reject method is a very powerful. So much so that we can use it to numerically evaluate the value of π. To do this, imagine that you have a circle of diameter 1 inscribed within a square of side length 1 (you may want to draw a picture of this). Now, using and accept-reject method, generate 1000 random points falling anywhere within the square shaped region. Determine what fraction of these fall within the circle. These two numbers, the total generated points, and those falling within the accepted region of the circle, can directly be related to the area of the square and the circle

$$A_{\text{square}} = d \times d = N_{\text{total points}} A_{\text{circle}} = \pi * (d/2)^2 = N_{\text{accepted points}} \tag{2}$$

and if we take the ratio of these two equations then we find

$$\frac{\pi \frac{d^2}{4}}{d^2} = \frac{N_{\text{accepted points}}}{N_{\text{total points}}} \pi = 4 \times \frac{N_{\text{accepted points}}}{N_{\text{total points}}} \tag{3}$$

Write a short program that calculates π using this method.
If you are able to calculate π, the next most important thing to realize is that this is an *approximate* calculation and you should ask yourself about the precision of the answer. You probably compared the results with the value of π you know from math class. We now want to use statistics to estimate the uncertainty. You already plotted a histogram with several π-values. One method to calculate and reduce the uncertainty is to calculate π several times and determine the mean μ and standard deviation σ of the distribution of calculated π values. Increase the number of π-values you use and examine how σ evolves. Another thing that will impact the precision of the determination of π is to the number of random points used in each accept-reject calculation. Again calculate several π values and get the mean and σ. Keep the number of π-values constant and alter the number of random points. How does σ change? Does this make sense?



(a) (b)

Figure 5: A cartoon representation of the implementation of the iverse-transform Monte Carlo generation method to produce a single x measurement from the ground state of a particle in a box system in quantum mechanics. (Left) The ground state probability density function (left) with the observable bounds x_{low} and x_{high} corresponding to 0 and y_{high} on the cumulative probability density function (right). Also shown is the generation of a single observation by the generation of a random value y_{gen} and translation to the observed value x_{gen}.

3.3 Inverse Transform Method

A complementary method to the accept-reject method is the inverse transform method and aims to achieve the same goal, generating an ensemble of measurements, x values, that are distributed according to some generic probability distribution function. However, there exist a number of differences that make this method markedly different in how it achieves this goal.

To understand how the algorithm proceeds and examine the differences with respect to the accept-reject method, we will again use the example of a particle in a box in the ground state. In this case, the ground state PDF is as shown in Figure 5(a) and we desire to generate an ensemble of x values that are preferentially located near the center of the box. The first step is to *transform* the PDF to the cumulative density function (CDF) by integrating the PDF from 0 to x to obtain $CDF(x)$ as in Figure 5(b). It is this CDF which will be used in the generation of accepted x values by using it as a translation tool. To achieve this, you again start with the generation of uniformly distributed random numbers using `random.random()`. However, in this case, these values will be used as the y values to begin with, however instead of comparing these values to the PDF, these initial y values will be translated to accepted x measurement values by finding the *inverse* of the CDF by inverting the formula $CDF(x) = y$. Because the CDF is one-to-one, by construction this x value is uniquely defined and is directly taken as the observed measurement.

Reason for yourself, conceptually, how the generation of a uniformly distributed set of events on the y axis is correctly translated to a set of events distributed according to the PDF. Specifically why is the translation function chosen to be the CDF? Is it possible to choose another functional form and achieve the same result? It may help to address this question with a picture.

Finally, it is necessary to carefully tune the range of generated y values to give the desired range in x for the ensemble of observations. This is rather straightforward and is achieved by starting with the desired

x range $[x_1,x_2]$ and translating it to the necessary y range $[y_1,y_2]$=$[CDF(x_1),CDF(x_2)]$.

In this case, there is the advantage that for each accepted measurement x, only a single random number must be generated. Moreover, the fraction of generated random numbers that are kept as accepted events is 100% and therefore fully efficient. This is in contrast to the accept-reject method in which the efficiency of the algorithm, depending on the functional form of the PDF, can be very small. However, there is an additional step that is not required in accept-reject, being the evaluation of the CDF from the PDF. In this example, the PDF of the ground state system is $PDF(x)\ sin(x)^2$ and so the CDF can be evaluated analytically to give a complete formula for the CDF. In cases where the PDF is not as well defined, or perhaps not even analytically defined, this transformation can be done numerically to obtain a piecewise function.

Practice writing a program in PYTHON that generates random numbers, distributed according to a Gaussian distribution with a mean μ of 5 and a width σ of 2 in the range of [0,10].

Practice writing a program in PYTHON that generates random numbers, distributed according to a falling distribution of the form e^{-x} in the range [0,5]. After doing this, generate the same distribution using the accept-reject method and determine the efficiency of the generation, defined as the fraction of accepted x values with respect to the total generated x values. How does this efficiency compare for the accept-reject method and the inverse transform method?

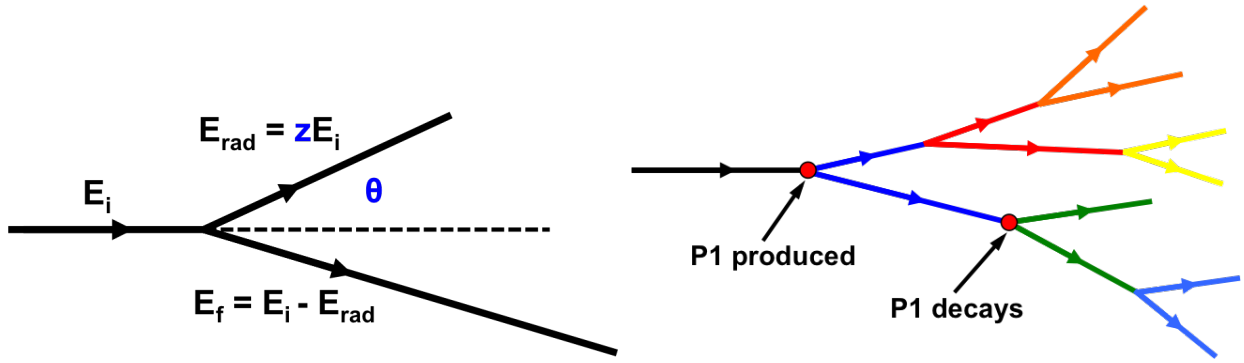
4 Monte Carlo Jets Model

As described in Section 2, one of the most commonly observed objects in an LHC collision event is a jet. A jet itself is a collection of many particles travelling in the same general direction produced from an initial high energy parton that is produced in the event. The process by which these particles are produced, as described in Sections 2.1 and 2.2 is itself a stochastic process that centers around the concept of one-to-two splittings. At each stage of splitting, exponentially more particles are created, but the energy of these particles monotonically decreases as the energy of the initial particle is divided more and more until the system "freezes out" into a collection of stable hadrons.

In this section, we will describe a simple model for this production of the building blocks of a jet which will then be clustered using the principles in Section 6. The primary goal of this section is for you to implement the model in a PYTHON or C++ program with the only external random number generator package being that which produces uniformly distributed random numbers, as described in Section 3. The specifications for this program are that it should take as *input* a single four-momentum vector, representative of an initial parton and produce as *output* a collection of final state four-momentum vectors representative of the stable final state hadrons. There are a number of free parameters that should be easily tuneable when given to a user and these are described in Sections 4.1 and 4.2.

4.1 The Parton Shower

The parton shower model itself is a recursive process built around the idea of successive one-to-two splittings, shown in Figure 6, which are chained together in a recursive tree-like sequence as in Figure 6.



(a) (b)

Figure 6: A single one-to-two splitting (left) highlighting the stochastic observables in the process and the composition of multiple splittings to form a full parton shower (right).

In this sense, the Monte Carlo model of the parton shower can be described by a sequence of only two stochastic and deterministic processes

- Deterministic : A particle produced with some four-momentum (E, p_x, p_y, p_z) travels for a finite amount of time t and traverses some distance, determined by its kinematics and the travel time[6].
- Stochastic : After this travel time, the particle decays by the emission of a radiative particle at an angle θ with respect to the initial direction of travel with a fraction z of the energy of the initial particle.

To implement this recursive model, it is necessary to understand more completely both of these base processes. The deterministic stage which, in Figure 6(b), brings particle P1 from the initial location, where it was produced by the initial radiative decay, to the final location, where it too will decay, is calculable from relativistic kinematics.

The details of the process of a decay, the one-to-two splitting, must be treated with more care, and where possible, simplified. As described in Section 2.1, the exact way this splitting occurs depends on the initial and final state particles. However, we will start by making the simplification to treat all splittings in the same way. As such, there are only two parts of this stage of the model that are stochastic, the angle of emission and the energy of the emission. As discussed in Section 2.1, the physics of QCD produces soft and collinear splittings and so the PDFs of these observables should follow the functional forms

- z (soft $\frac{1}{z}$) : This can be seen to have a singularity at $z = 0$ and therefore must be slightly modified to either (i) have a finite value at $z = 0$ by setting $PDF(z) = \frac{1}{1+z}$ and allowing the range of possible z values to span [0,1] or (ii) setting $PDF(z) = \frac{1}{z}$ and setting a lower minimum bound on the allowable z values as $[\epsilon_z,1]$. Either is fine, and is a specific choice of your model.
- θ (collinear $\frac{1}{\theta}$) : This can be seen to have a singularity at $\theta = 0$ and therefore must be modified in a similar way as for z. However, the range of θ is slightly different and should vary between $[0,\pi/2]$.

In a given splitting, after generating a random (z,θ) pair, the kinematics of the radiated particle are determined. However, to continue the shower process and allow for the model to proceed it is necessary

[6] In a real collision, no finite observable distance is traversed. This stage of the model is included for visualization purposes.

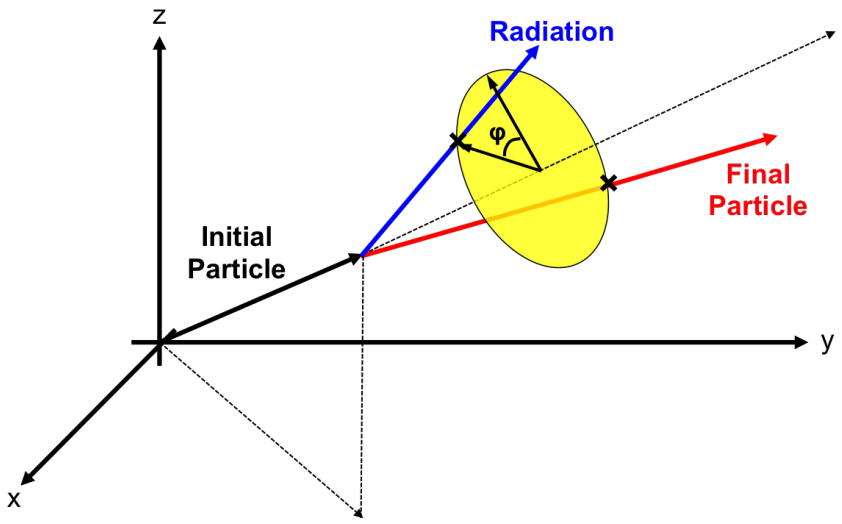


Figure 7: A illustration of the azimuthal angle ϕ of a one-to-two splitting occurring in three dimensions.

to determine the kinematics of the final state particle, from which the radiation was emitted. At this point, with one simplifying assumption, and by applying the conservation of energy and momentum, the final state kinematics of both outgoing particles.

Can you determine which simplifying assumption it is that allows for the full determination of the final state? The main point is to recognize the energy scales we are talking about here in comparison to the typical energy scale of stuff at the LHC (TeV).

This same type of splitting, but with different random (z,θ) pairs can then be linked together in a sequence to generate a pictorial representation of this two dimensional shower structure as in Figure 6(b).

This parton shower model should be implemented such that, in addition to the output set of four-momentum vectors, a graphical representation of the parton shower tree is produced from your program. This takes careful organization of the flow of information and can be achieved in multiple ways.

The entire description of the parton shower model thus far has implicitly assumed that the kinematics are constrained to two dimensions and are in a two-dimensional plane. Since the real physics we care about modelling is in three dimensions, we need to modify the stochastic portion of the process model to account for this. This is achieved by including an additional splitting angle ϕ which is the azimuthal angle of the decay around the direction of travel of the initial particle, as shown in Figure 7. However, given the structure of the QCD interaction described in Section 2.1, there is no preference for the value of this angle and it should therefore be chosen from a uniform distribution with a range of $[0,\pi]$.

It is recommended to proceed through Section 4.2 working in two dimensions. This can help to build intuition and make the validation of your process model simpler. However, it is necessary to include the full three dimensional model, as described here, before proceeding to Section 6.

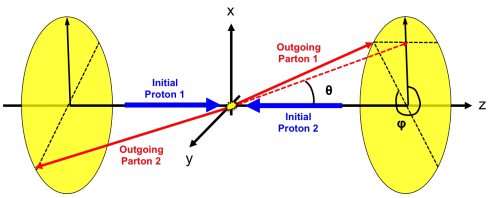


Figure 8: A pictorial representation of a proton-proton collision event similar to that occurring in the LHC. Shown here are the incoming protons and the two outgoing partons with the topology specified by the polar angle θ and azimuthal angle ϕ.

4.2 Hadronization

As described in Section 2.2, the parton shower proceeds until the energy scale of the individual particles decreases to the point where it is energetically favorable to form stable hadrons with other partons existing within the event. The manner in which this happens is not well understood from first principles. For this model, we will take a very simple approach and make a direct translation between each parton and each hadron. To achieve this, choose an energy scale threshold E_{crit} below which a parton does not split further. This should be made a parameter that is easily tunable in a single location within your program.

5 A Jetty Event

In LHC collision, it is very rare to produce a single parton in the hard collision. More often than not, two partons will be emitted. Now, because the collision was between two particles travelling opposite to each other, we will make the simplifying assumption that the initial state was completely balanced and had an initial momentum of $p_{initial} = 0$. Therefore, when the two partons are emitted, they must be in a configuration that obeys the conservation of momentum (i.e. "back-to-back"). However, there are some aspects of this configuration which are stochastic, namely :

- Energy : The actual energy of one of the emitted partons is random and distributed according to an exponentially falling distribution $e^{-\alpha E_{parton}}$ where α is a physically meaningful parameter that characterizes collisions at the LHC and E_{parton} is the randomly chosen energy of the parton.
- Azimuthal direction : The collisions that occur have no preferred direction in terms of the azimuthal angle ϕ as shown in Figure 8. Therefore, this is a randomly distributed observable with a flat distribution from [0,2π].
- Polar direction : Again, the collisions that occur have no preferred direction for the emissions and so the polar angle θ in Figure 8 is a random number distributed evenly from [0,π].

In Section 4, a single parton with an initial hard-coded four vector momentum of a parton P was sent through the parton shower and hadronization to produce a collection of particle pseudojets p_i like $P \rightarrow [j_1, j_2, ..., j_n]$. Augment this by initially producing an *event* with the properties described here and then showing both of the partons in the collision. Procedurally, you will have to produce two initial partons P^a and P^b and then shower each independently as $P^a \rightarrow [j_1^a, j_2^a, ..., j_n^a]$ and $P^b \rightarrow [j_1^b, j_2^b, ..., j_n^b]$. Finally, you should take the union of these final state particles to produce the full showered event.

6 Jet Reconstruction

The simulation that you have written in Section 4 represents the evolution of a single energetic parton to many low energy stable hadrons. Conceptually, this is the process that occurs after a proton proton collision emits a parton which produces the collimated sprays of particles that are observed in the detector. However, this is not yet a jet. Even though all particles originated from the same parton, there is a level of ambiguity due to the recursive structure of the shower. This can be seen from the first splitting in the tree. After this splitting, the system contained not one parton, but two. At this point, the intuition of "*jet=parton*" would lead us to believe that there should be two jets in this (pseudo)event. So which notion is correct? Is there one jet or two?

To resolve this ambiguity, it is necessary to concretely define what is meant by collimated spray and this is done via a *jet clustering algorithm*. A full review of these algorithms (and much more) is provided in Reference [2] but all achieve the same aim of partitioning the full set of stable final state particles into subsets of particles using a set of rules based on the intuition of what a "collimated spray" actually is. After performing this clustering, the concept of a jet is fully defined and it is possible to quantify features of this jet by designing observables to study how the jet was formed in the parton shower or to highlight defining features to classify the initial type of particle from which the jet originated.

6.1 Jet Finding

The idea that a collimated spray of hadrons in the detector is representative of an initial high energy parton is only useful if it is possible to concretely define a collimated spray. This is done by clustering algorithms that take as input a collection of four-momentum vectors from an entire event (which themselves may originate from multiple partons as described in 5) and return subsets of four vectors with each of these subsets being a "jet". The way in which these subsets are defined has evolved over time [2] and at the LHC, the most commonly used clustering algorithms that are used rely on the concept of *iterative recombination* [11][7]. The intuition that drives these algorithms can be thought of as a reversal of the model created for the parton shower. Therefore, instead of starting with one particle and performing one-to-two splittings until you have many low energy particles, these algorithms start with many low energy particles and perform pairwise two-to-one combinations until you have "sufficiently distinct" four momentum collections.

This clustering process begins with the set of final state particles. In the case of the simulation here, these are the real particles coming from the hadronization process but in an actual experiment these will be observed and reconstructed signals in the detector, such as calorimeter clusters or charged particle tracks. At this point, it is also helpful to regard the particles as pseudojets, which are four-momentum vectors at an intermediate stage of the clustering process which themselves may be composed of previous combinations of pseudojet four momentum vectors. The order of the pairwise clustering combinations the decision that a given collection (pseudojet) is sufficiently distinct from the others then proceeds by following a set of abstract distance metrics[8]. The first, d_{ij}, measures the distance between any two

[7] The primary reason that these algorithms are preferred is because they are (i) infrared and (ii) collinear (IRC) safe, meaning that they are invariant under the presence of additional infinitely soft radiation in the event or an additional infinitely collinear one-to-two splitting in the parton shower. In some sense, this can be thought of as being robust against the exact details of the parton shower, which themselves are only approximation of the true physics.

[8] In this context, distance does not mean physical distance of length measured in meters. Remember, the entire parton shower and hadronization process occurs in an unimaginably small distance.

pseudojets in the event. This distance is defined in Equation 4 and takes into consideration both the energy of the particles that may be combined by using their transverse momentum p_T and the angular distance $\Delta_{ij} = \sqrt{(\phi_i - \phi_j)^2 + (\eta_i - \eta_j)^2}$[9] separating them. The second, d_{iB}, is commonly referred to as the *beam distance* and provides a measure of the distinctness of any given pseudojet i from the rest of the clustered event and is defined in Equation 5.

$$d_{ij} = min(p_{Ti}^{2n}, p_{Tj}^{2n}) \times \frac{\Delta_{ij}}{R} \tag{4}$$

$$d_{iB} = p_{Ti}^{2p} \tag{5}$$

For a given stage of clustering, whether it be the initial stage when there are N initial single particle pseudojets or later when some of these have been bundled together into some lesser number of composite pseudojets, all possible distance measured are calculated. This will produce ${}_N\mathrm{C}_2$[10] pseudojet-pseudojet distances and N pseudojet-beam distances. From the full set of distances, the minimum distance dictates the action that is to be taken in the clustering sequence. In the case that the minimum distance is a d_{ij} distance, it can be interpreted as indicating that in the event, the next best thing to do is group those two particles, i and j, together as having originated from a common one-to-two splitting in the parton shower. Conventionally, this grouping is done by simply adding together the four-momentum vectors of these constituents to obtain some resultant four vector. If, on the other hand, the minimum distance is a d_{iB} beam distance, then the interpretation is that there exists a pseudojet in the event[11] is distinct enough that it should be interpreted as representing a hard parton originating from the underlying physics process. In this case, the pseudojet is set aside and becomes a jet.

In both of these cases, either an ij combination or the promotion of a pseudojet i to a jet, the number of pseudojets in the event decreases by one. Therefore, the algorithm proceeds in this iterative fashion until all pseudojets have been promoted to some jets and there exist no more pseudojets for clustering.

From the previous Monte Carlo model you have created, you have the ability to simulate parton showers/hadronizations that are representative of hadronic activity in LHC collisions. At this stage, implement the clustering algorithm described here to cluster a single event, including proper validation of the procedure. Note that you will have to specify the parameters n and R which are choices of the algorithm. Make a histogram[a] of the number of jets that result from this algorithm for 1000 parton showers and do this for the set of n values of [-1,0,1] and R values of [0.01,0.05,0.1,0.5,1.0]. What differences do you see?
A histogram is a great way to check if your program is doing what it is supposed to do. Think about what this histogram should look like and ask youself "Does it make sense?"

[a] If you are unfamiliar with python histograms check the python library MATPLOTLIB manual.

[9]

[10] This is the number of combinations of N things taken 2 at a time without replacement, which is calculated as the binomial.

[11] Note that this pseudojet can itself be a single particle or a pseudojet composed of multiple pairings due to previous d_{ij} combinations.

The algorithm described here is commonly referred to as *inclusive jet finding*. A complementary algorithm, called *exclusive jet finding*, exists that performs clustering in a slightly different way and truncates based on a different condition. Find a description of this algorithm and compare and contract it to inclusive jet finding. Implement this algorithm in your program such that it is possible to cluster jets in either fashion.

6.2 Jet Observables

After having performed jet clustering on the generated parton shower, there exist a set of jets that themselves are exclusive subsets of the initial set of stable particles resulting from the parton shower and hadronization. In principle, they are representative of a hard parton in the event and because they were formed by combining four-momentum vectors together, it is possible to make the association $(E, p_x, p_y, p_z)_{parton} \sim (E, p_x, p_y, p_z)_{jet}$. Focusing on the 0^{th} entry of this four vector and noting that the jet itself is composed of multiple particles from the parton shower, one can recognize that the parton energy is represented as $E_{parton} \sim E_{jet} = \sum_{i \in \text{jet constituents}} E_i$ where the *jet constituents* are the stable particles that the jet clustering initially started with. Viewing one component of the jet four vector this way is an example of a single jet *observable*.

However, it is possible to generalize this concept of a jet observable to be any function that takes as an argument the set of jet constituents and produces a single scalar quantity. Explicitly, this is $O(jet) = function([j_1, j_2, ..., j_n])$ where j_i are each of the n four vector measurements of the initial single particle pseudojets. Implementing this is challenging from the organizational standpoint as it requires you to keep track of the initial four vectors as the clustering proceeds, not just the combined four vectors. The exact form of $function()$ is chosen based on the intuition of some physical property of the parton or jet that one is interested in summarizing in a way that can be formed into a one dimensional spectrum. In the case of the jet energy, the interpretation of the feature of the jet is direct and represents simply the amount of energy that the initial parton had when produced by the physics process. However, more complicated observables can be formed that quantify and can be used to investigate features of the parton shower model that was initially implemented in Section 4. Perhaps most importantly though, the quantification of a jet feature in this way can be done both in Monte Carlo simulation and in real collision data, allowing for the measurement or tuning of the underlying parameters of the model used to generate the parton shower.

Based on the jets that you have clustered in the previous section, calculate the jet "pseudo-mass" as

$$\text{SimpleMass}(J) = E(j_1) \times E(j_2) \times \Delta R(j_1, j_2)$$

where j_1 and j_2 are the two highest p_T constituents of the jet J. In the case that a given parton shower and clustering produces more than one jet, perform this calculation for the jet of highest energy. Perform this calculation for a large number of parton shower generations and make a histogram of the pseudo-mass for all of these generated showers.

7 Conclusion and Project Extensions

That brings you to the end of this tutorial. Having worked through it, the intention is that you have learned something practical, in terms of programming, and intellectual, in terms of physics. The next step is to

start to create your own questions and define a research direction of your own. This can be done by either building on the project you have already begun, with some suggestions provided in Section 7.1 or by moving to the usage of tools more commonly used in HEP as in Section 8.

7.1 Toy Monte Carlo Extensions

Described here is a list of possible extensions for the toy Monte Carlo simulation that you have written. If you have an idea to pursue that has come up during the course of working through the tutorial, pursue that instead. However, some of these may be interesting jump off points.

- **Parton Distribution Functions** [12–15] : If a proton collides with a proton, then the collisions really occur between the underlying constituents, the quarks and gluons. Because the physics that can result from a given type of collision (*up-quark:up-quark* vs. *up-quark:gluon*) differs, understanding how to incorporate this degree of randomness into the generation of an ensemble of full events is important. In this extension you will take the jet model you built and interface it with a model of proton collisions which may produce different final states depending on the initial state.
- **Comparing Hadronization Models** : Depending on what observable is being reconstructed, the degree of sensitivity to the details of hadronization model can vary. And in a number of processes studied at the LHC and searches for new physics, this can become a limiting factor in terms of uncertainties. In this extension, you will construct two hadronization models based on the principles of PYTHIA and HERWIG and compare a number of observables.
- **Detector Simulation** : After a jet (or any observable physics object) has been formed, it is observed in the detector that has been constructed around the interaction point. This observation is made possible because of the known interactions of the incoming particle with the material of the detector. Unfortunately, aspects of this are random and the particle may interact with multiple sub-detector components. Therefore, it is important to model the detector using Monte Carlo simulation [16]. Some common simulation toolkits that are used to do this are GEANT [17] and DELPHES [18]. In this extension, you will build your own detector simulation using similar principles as those implemented in these programs.
- **Simulating W Bosons and top Quarks** [19, 20] : At the LHC, the high center of mass energy means that when massive objects are produced they can often have high kinetic energy. If these massive objects decay to partons, then it is possible that the parton showers of these particles will merge and the paradigm of a jet being a parton breaks down. Instead, it is necessary to treat a jet as a $W/Z/higgs$ boson or a top quark. The primary difference is that prior to the parton shower process, there will be one or two "hard splittings" that occur due to the decay of the massive object. In this project, you will simulate these hard splittings and then interface the results to the parton shower jet simulator you have previously written.

8 PYTHIA/FASTJET Analysis Extension

In high energy physics, the tools used to perform the simulation of collisions events, the parton shower, and hadronization as well as the clustering of jets are standardized. They are provided in packages that can be installed on your computer and built into larger analysis programs, similar to the use of the

`random.random()` in PYTHON. This helps ensure that all people working in HEP speak a common language and can share their ideas more effectively. In the case of the simulation of events, one of the most commonly used simulation tools is PYTHIA. Although it includes more refined models of the underlying physics, it functions on the same general principle of the program designed here. In the case of jet clustering, there is a single package used by all of the LHC experiments called FASTJET. In this case, although the working principle is the same as the algorithm you implemented in Section 6, the exact implementation itself allows for the execution of jet finding to occur at a much faster rate.

In this extension, you will be provided with guidance on how to use these programs in the context of a simple simulation and analysis and then, after defining a research question, use this setup to investigate it.To start this extension, follow the directions provided in the PYTHIASIMULATORANALYSIS.

References

[1] S. Ellis, *"What is a Jet?". "What is a Jet?"*, (2010), URL: https://cds.cern.ch/record/1276292.

[2] S. D. Ellis, J. Huston, K. Hatakeyama, P. Loch and M. Tonnesmann, *Jets in hadron-hadron collisions*, Prog. Part. Nucl. Phys. **60** (2008) 484, arXiv: 0712.2447 [hep-ph].

[3] L. Evans and P. Bryant, *LHC Machine*, JINST **3** (2008) S08001.

[4] J. Proudfoot, 'Hadron Calorimetry at the LHC', *Proceedings, 34th SLAC Summer Institute on Particle Physics: The Next Frontier: Exploring with the LHC (SSI 2006): Menlo Park, California, July 17-28, 2006*, 2006, URL: http://www.slac.stanford.edu/econf/C060717/proceedings.htm.

[5] F. Wilcek, *Asymptotic Freedom: From Paradox to Paradigm*, Nobel Lecture (2004), URL: http://web.mit.edu/physics/people/faculty/docs/wilczek_nobel_lecture.pdf.

[6] T. Sjostrand, S. Mrenna and P. Z. Skands, *PYTHIA 6.4 Physics and Manual*, JHEP **05** (2006) 026, arXiv: hep-ph/0603175 [hep-ph].

[7] T. Gleisberg et al., *Event generation with SHERPA 1.1*, JHEP **02** (2009) 007, arXiv: 0811.4622 [hep-ph].

[8] M. Bahr et al., *Herwig++ Physics and Manual*, Eur. Phys. J. **C58** (2008) 639, arXiv: 0803.0883 [hep-ph].

[9] G. C. et. al., *Monte Carlo Techniques*, Particle Data Group Review (2009), URL: http://pdg.lbl.gov/2009/reviews/rpp2009-rev-monte-carlo-techniques.pdf.

[10] U. Davis, *Sampling Random Variables : Accept-Reject*, YouTube (2010), URL: https://www.youtube.com/watch?v=-mkLWm2f4hg.

[11] M. Cacciari, G. P. Salam and G. Soyez, *The Anti-k(t) jet clustering algorithm*, JHEP **04** (2008) 063, arXiv: 0802.1189 [hep-ph].

[12] G. Salam, *QCD at hadron colliders Lecture 3: Parton Distribution Functions*, Lecture Notes (2010), URL: https://gsalam.web.cern.ch/gsalam/repository/talks/2010-MariaLaach-lecture3.pdf.

[13] I. Cloet, *Deep Inelastic Scattering and Parton Distribution Functions*, Lecture Notes (2013), URL: http://www.physics.adelaide.edu.au/cssm/workshops/NPQFT/lectures/cloet_lecture_3.pdf.

[14] H.-L. Lai, *Introduction to Parton Distribution Functions*, Lecture Notes (2010), URL: http://phys.cts.nthu.edu.tw/~particle/2010TOPICAL/LHC/download/workshops_PDF/1025_1026_Pedagogical_Lectures/20101026_Lai.pdf.

[15] J. Huston, *Parton Distribution Functions:their generation and use(especially at the LHC)*, Lecture Notes (2008), URL: https://conference.ippp.dur.ac.uk/event/156/contribution/31/material/slides/0.pdf.

[16] Y. Chao, *Detector Simulation*, Lecture Notes (2013), URL: http://www.phys.nthu.edu.tw/~numscc/nums-hep2013/materials/Lecture-ychao_20130123.pdf.

[17] S. Agostinelli et al., *GEANT4: A Simulation toolkit*, Nucl. Instrum. Meth. **A506** (2003) 250.

[18] S. Ovyn, X. Rouby and V. Lemaitre, *DELPHES, a framework for fast simulation of a generic collider experiment*, (2009), arXiv: 0903.2225 [hep-ph].

[19] G. Aad et al., *Identification of boosted, hadronically decaying W bosons and comparisons with ATLAS data taken at* $\sqrt{s} = 8$ *TeV*, Eur. Phys. J. **C76** (2016) 154, arXiv: 1510.05821 [hep-ex].

[20] S. Schätzel, *Boosted Top Quarks and Jet Structure*, Eur. Phys. J. **C75** (2015) 415, arXiv: 1403.5176 [hep-ex].