This report will cover several of the most fundamental methods used in studying hadronic jets in LHC, tested by Yonsi, 3rd-March-2019, according to the tutorial.

Part1: Sampling

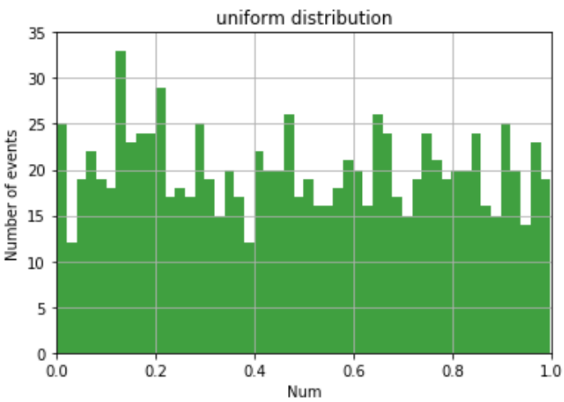
The first part is about the basic concepts of Monte Carlo method. We try to imitate the real physic process by generating random numbers and build up different particles, obeying physical laws. Monte Carlo method works well when it comes to simulating large number of random events. Now we start on the easiest.

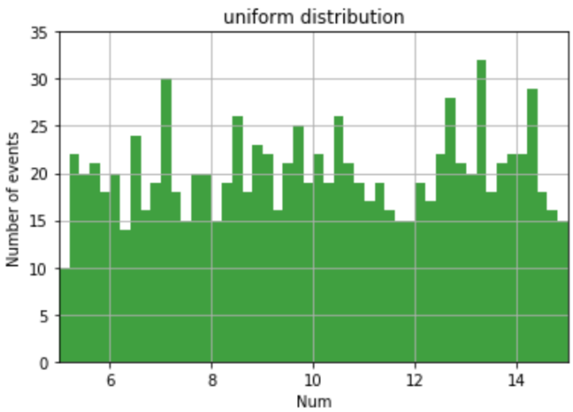
1.1 Uniform Distribution

We can easily get a uniform distribution using computational method. Here is a histogram(left) showing a uniform distribution between [0,1]. 1000 random numbers are used.

It can never be definitely 'flat' because it comes from random process. The error is within our expectation.

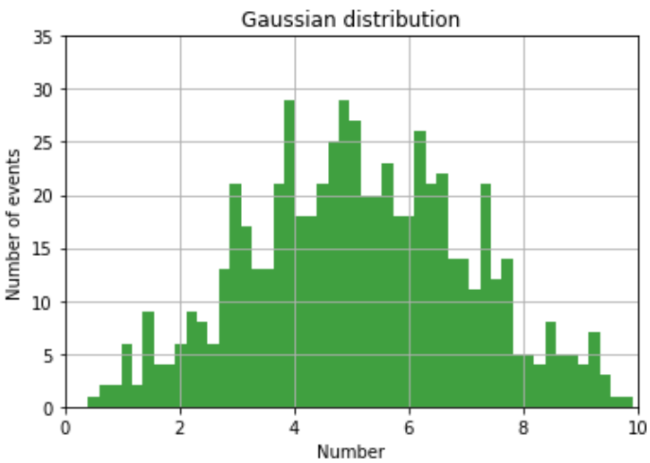
The right one also shows another uniform distribution, ranging from [5,15]. The shapes are almost the same in a large scale. And as we know, they should be the same if we can generate infinite numbers.





1.2 Accept Reject Method

Generating a uniform distribution is always the fastest for computers. So, we will carry out different ways to get various distributions. The first one is the Accept-Reject Method.

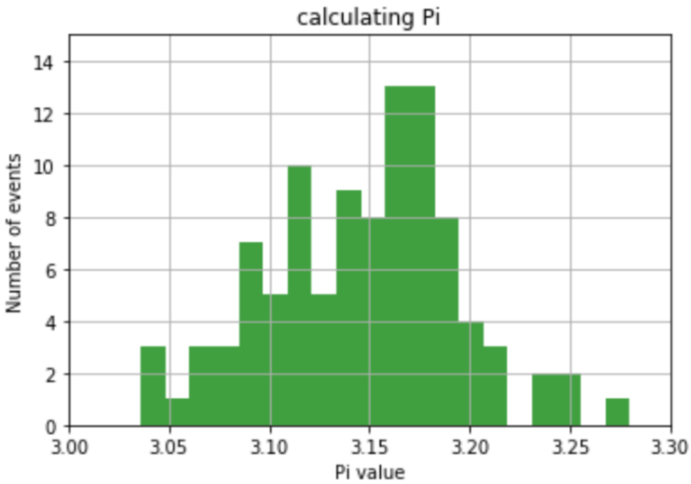


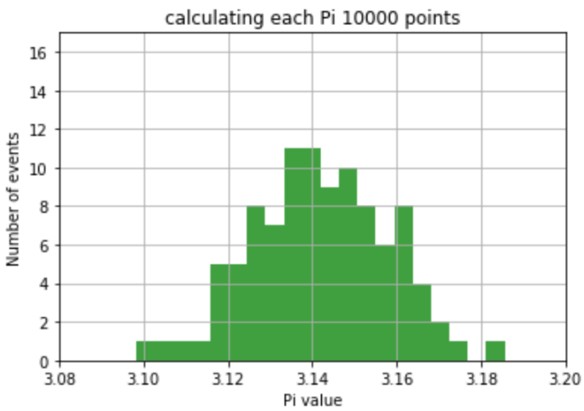
We generate two uniform-distributed random number and get one, say, Gaussian-distributed number. The first number is the value while the second is used for judge and select.

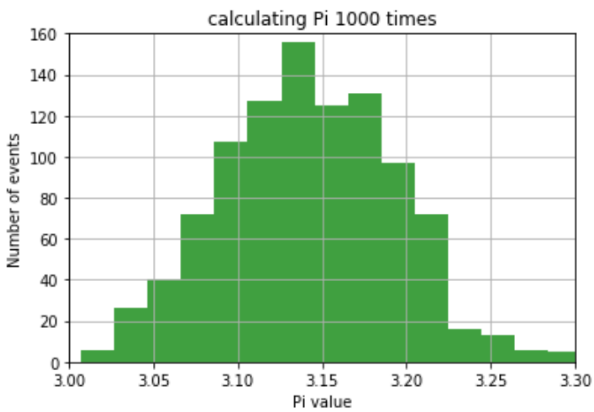
Now we get a Gaussian distribution using 1000+1000 uniform random number.

1.3 Calculating π

The accept-reject method can be used as calculating π. We can draw a square and estimated π by pin-casting. Originally, we calculate each π value using 1000 points and calculate 100 times. And then we try to reduce the uncertainty of calculation. Two methods are applied. Firstly, in each calculation, we add iteration times and cast more points (10000). Secondly, we calculate more times (1000).



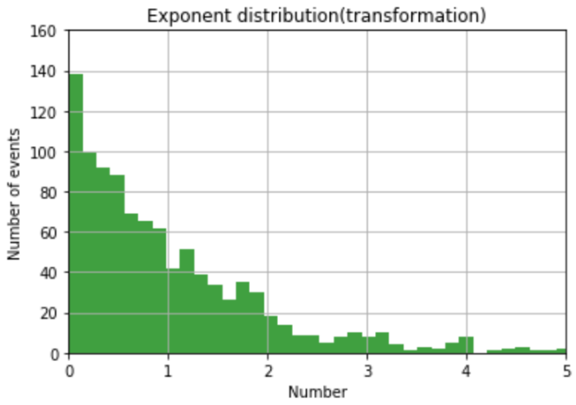


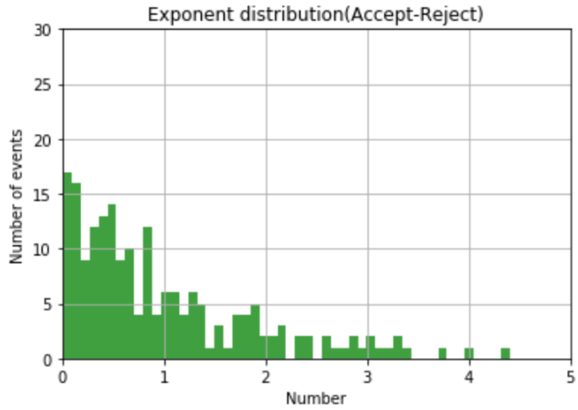


The plots are not normalized, but we still can see the scale is different. It seems that casting more point helps reduce the uncertainty of distribution while calculating more times does not. The precision of π distribution directly relates to the pin-casting process. The peak will become significantly sharper. If you calculate more times, it merely makes the shape of distribution more stable. However, there is a confidential difference between the uncertainty of distribution and the uncertainty of the mean value. The mean value of π is mathematically equivalent in both methods. So does the uncertainty of mean. So, if we only want the mean value, both methods are suitable.

1.4 Inverse Transform method

The efficiency of Accept-Reject method is relatively low. To improve it, we can use transformation method for some certain distributions. We can manually integrate the Probability Distribution Function as a Cumulative Distribution Function in some cases. As we learned in Computational Physics Courses, the efficiency of Inverse Transform method can be up to 100%. We generate 1000 uniform-distributed random numbers for inverse transform method and 2000 for accept-reject method. But much fewer events are left in the right histogram. A large amount of numbers are rejected.



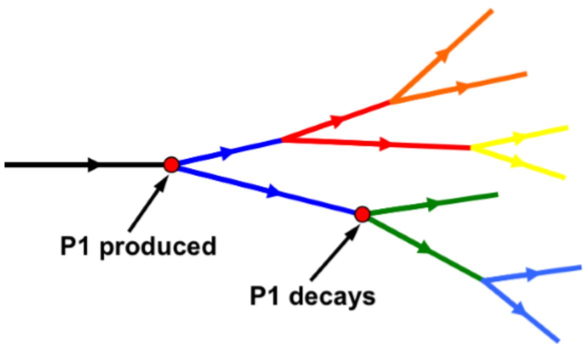


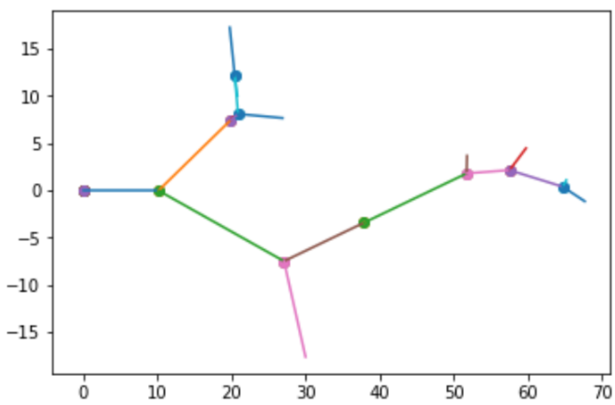
Part2: Jet Simulation

The second part relates to a preliminary method of jet events simulation and reconstruction. It succinctly shows the hadronization process from a shower. To simplify the jet model, we ignore the invariant mass of partons. It is a really rough estimation, because the conservation of momentum and energy cannot be obeyed simultaneously due to lacking of degree of freedom. And also, we assume that the energy of a parton does not attenuate in the medium. Due to these simplifications, we can only show the graphic results and it cannot be used quantitively.

❍ 2.1 Jet decay 2D

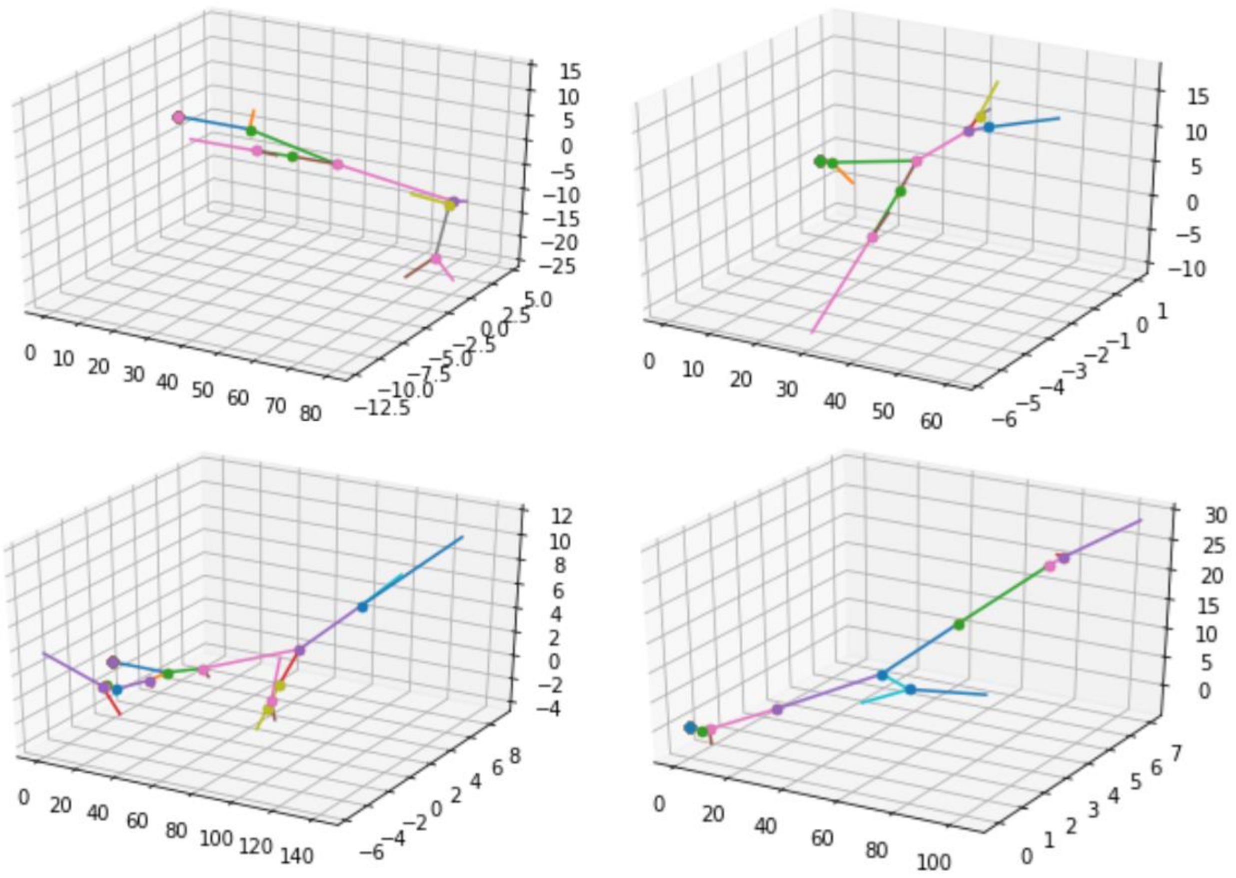
Firstly, we consider a 2D model. The expected plot is the left one and the computational simulation plot is the right one. The original Energy is 10 and the threshold value is 2. The energy proportion subjects to $\frac{1}{1+z}$, the angular coefficients subjects to $\frac{1}{1+\theta}$.





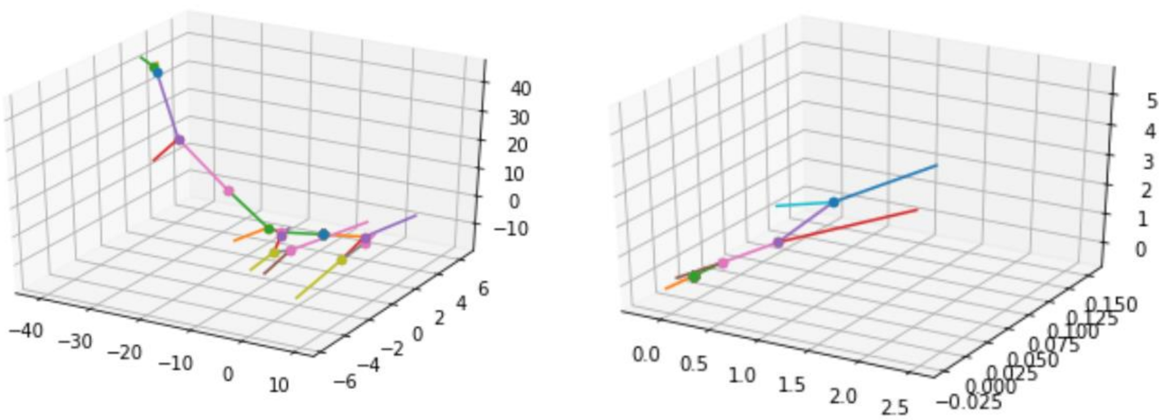
❍ 2.2 Jet decay 3D

We add a dimension φ ranging from $[0,2\pi]$ to get a 3D plot. The procedures are almost the same. We also apply 10 to the initial energy and 2 to the threshold. Here we show several different probabilities from the same shower.



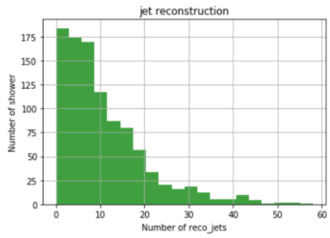
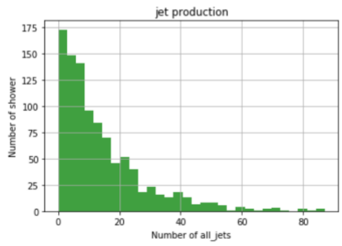
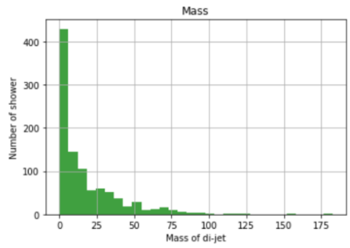
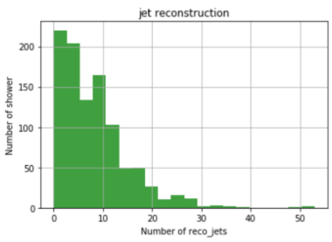
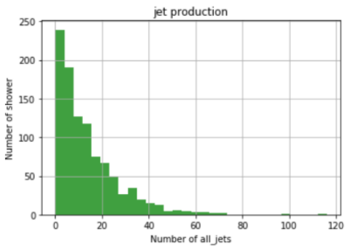
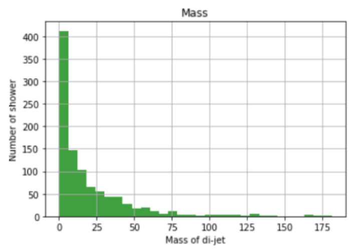
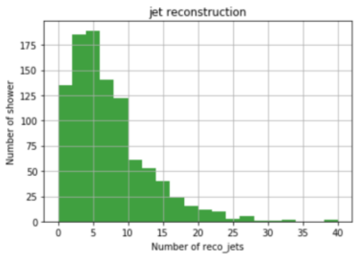
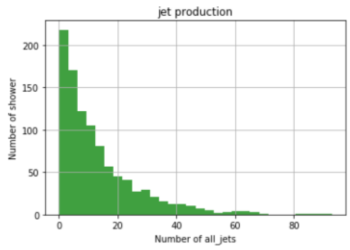
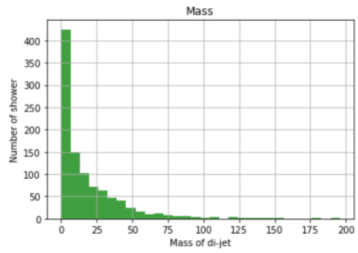
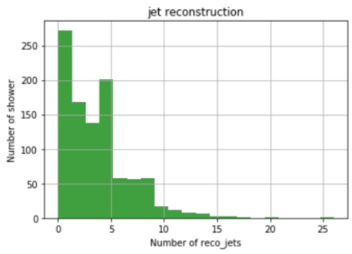
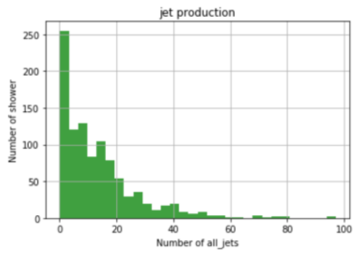
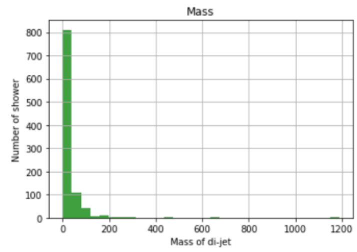
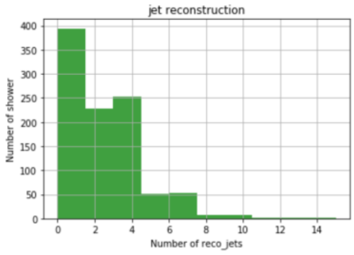
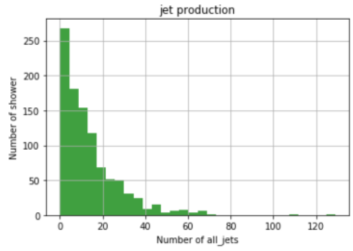
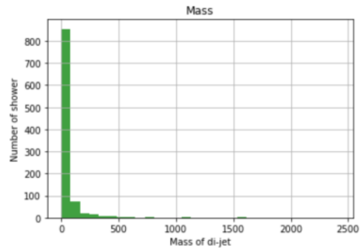
2.2 A Jetty event

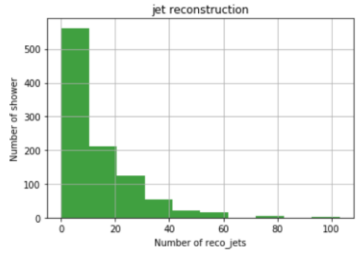
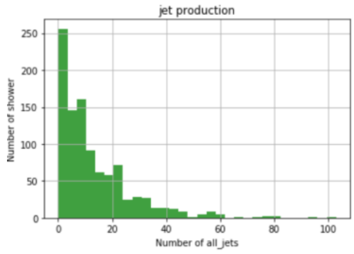
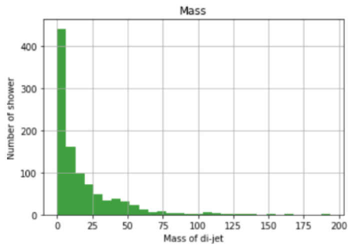
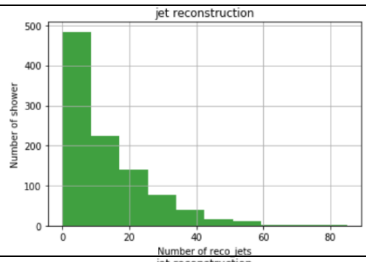
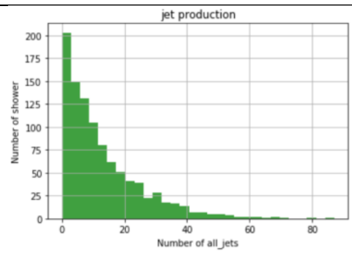
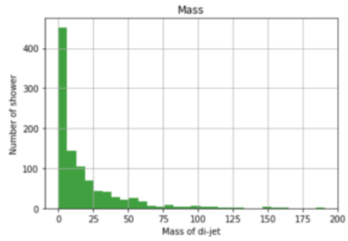
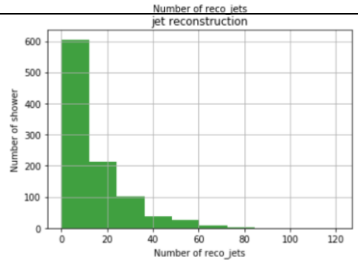
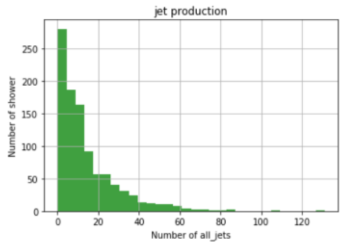
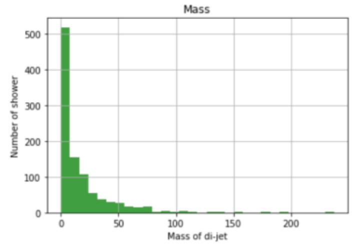
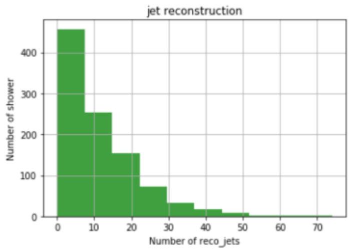
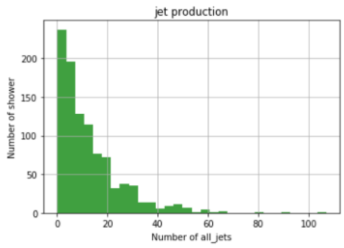
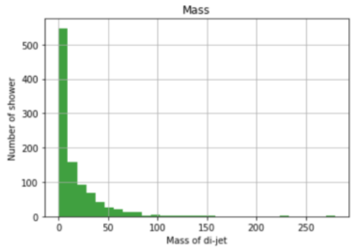
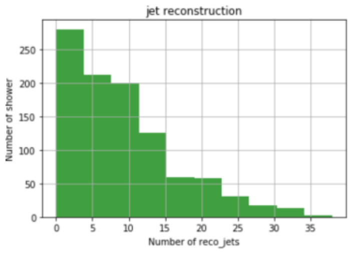
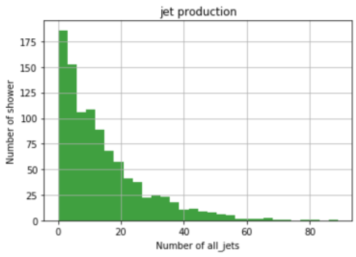
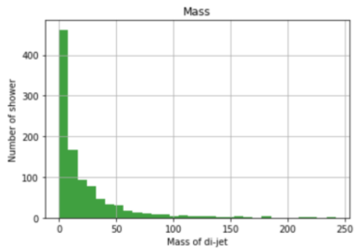
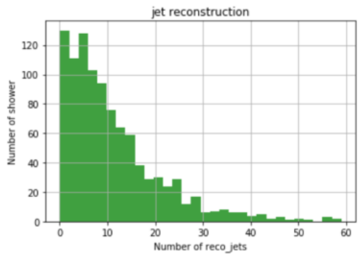
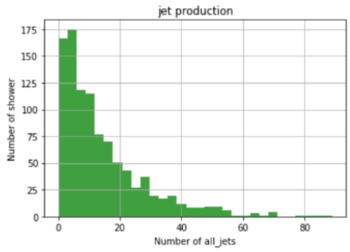
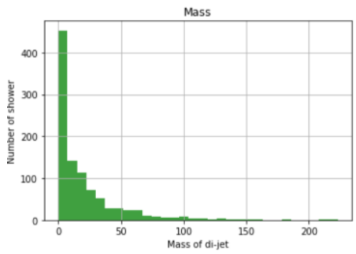
We can produce Monte Carlo simulation not only from a single shower, but also a di-shower event from a collision vertex. In real physics process, we rarely see single partons but often events. We also improved a little, letting the energy of center-of-mass frame subject to exponential distribution instead of a fixed value. The coefficient is 10 and the threshold is 2.

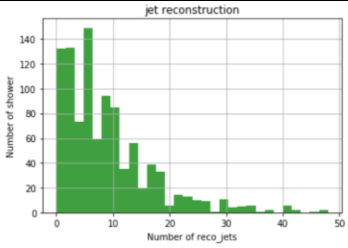
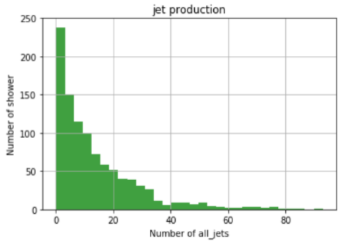
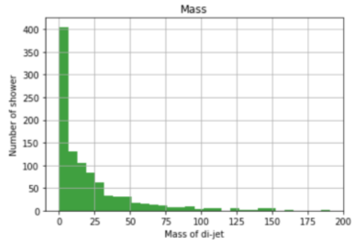
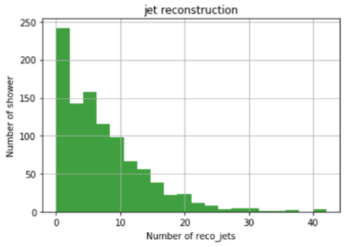
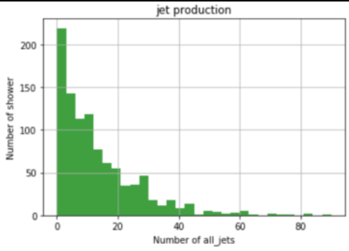
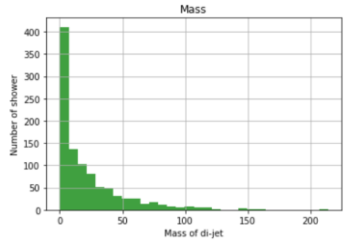
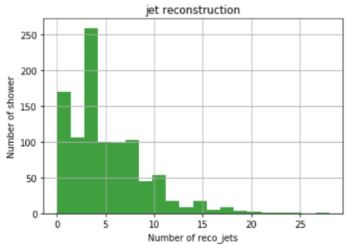
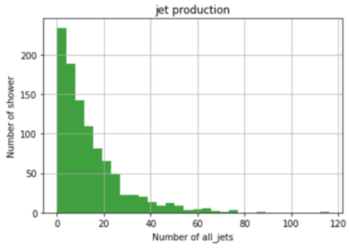
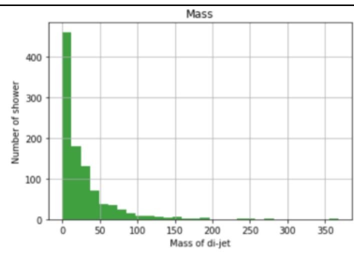
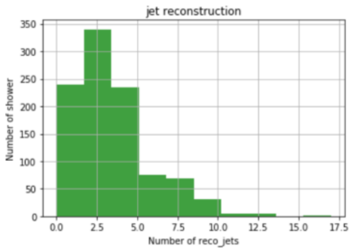
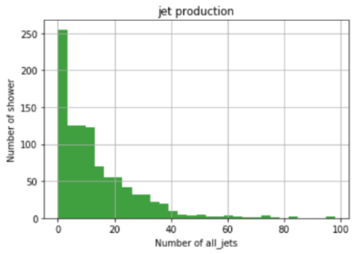
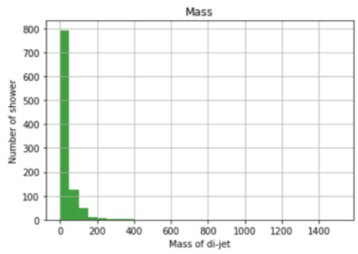


2.3 Jet Reconstruction and Mass

We use the method above to generate a Monte Carlo simulation of a jetty event. Now we consider it as pseudo data and what we need to do is to trace back from the stable hadrons. We will combine them and try to reconstruct the initial jet by adding up the four momentums of partons. If there is no more partons in its vicinity, we consider it as a single jet. We use variable n to represent different forms of formula and variable R to represent the distance between two partons. A list of histograms are shown as followed, with different n and R values. And there is also a simple mass calculation attached. The left figures are the number of hadrons, the middle ones are the number of constructed jets and the right ones are the simple mass. Initial energy subjects to exponential distribution and multiplied by the efficient 20. The energy threshold is 3. Randomly 1000 jetty events are generated.

	$n = -1$		
0.01			
0.05			
0.1			
0.5			
1			

n = 0			
0.01			
0.05			
0.1			
0.5			
1			
n = 1			
0.01			

0.05			
0.1			
0.5			
1			

I will still polish up the method for further use…