Jet reconstruction: course work for HIP 2022

FYSS4551, Spring 2022

Laura Huhta
Teacher: Dong Jo Kim

August 15, 2022

1 Introduction

In this report, I present some results from the jet analysis I worked on during the Heavy Ion Physics 2022 course.

The jet study begins with a `PYTHIA` [1] event generation where I generate about six million proton-proton (pp) collision events. From the generated output of final particles, I reconstruct jets using the `FASTJET` package [2, 3]. Finally, I plot distributions of some common jet properties and perform some basic analysis of these properties.

2 Jets

Jets are a tool widely used in collider experiments and analyses, especially in heavy ion physics, due to their capability of acting as probes for the QCD matter created in the collision. Jets are produced in QCD hard scattering processes, consisting of high transverse momentum quarks and gluons, or partons, created in the process. As a parton which was produced in the hard scattering is moving away from the interaction point, the strong coupling constant increases. As a result, the probability of the parton emitting QCD radiation, gluons, increases. The gluons are emitted in predominantly small angles and then radiate to $q\bar{q}$ pairs, which remain nearly collinear with the parent parton. This fragmentation continues until the new partons reach a region where perturbative QCD is not valid anymore, and the outgoing partons begin to hadronize.

There are multiple ways of merging the outgoing partons into jets. A jet algorithm is a set of rules that project a set of particles into a set of jets. [2] An algorithm requires a set of parameters to govern its behavior, and the combination of a jet algorithm and its set of parameters is known as a jet definition. Nowadays, sequential recombination algorithms, such as the widely-used anti-k_T algorithm [4], are the main kind of jet algorithm used at the LHC experiments.

Also in this work the anti-k_T algorithm is used. The parameters required by the algorithm are the momenta of all the particles, their rapidities and azimuthal angles, as well as a user-defined value for the so-called jet radius, or resolution parameter, R. The jet radius is chosen to be 0.4 in this work, as is often the case in ALICE jet analyses.

A couple of jet observables are studied in this work. The splitting fraction of a jet is defined as

$$z = \frac{p_{T,\text{parton}}}{p_{T,\text{jet}}}, \tag{1}$$

where the transverse momenta p_T are those of a parton in a jet and of the parent jet. The distribution of this variable is often presented via a jet fragmentation function, defined here as

$$D(z) = \frac{1}{N_{\text{jets}}} \frac{\mathrm{d}N}{\mathrm{d}z}, \tag{2}$$

where N_{jets} is the total number of jets.

Another commonly used variable is

$$\xi = \log\left(\frac{1}{z}\right), \tag{3}$$

from which another form of jet fragmentation function can be formulated as

$$D(\xi) = \frac{1}{N_{\text{jets}}} \frac{\mathrm{d}N}{\mathrm{d}\xi}. \tag{4}$$

3 Event generator and framework

3.1 PYTHIA

The general purpose Monte Carlo event generator `PYTHIA` [1] is a program for the generation of high-energy physics collision events. With `PYTHIA`, the user can choose what particles will be collided together and at which energy, how many of these collision events will be simulated, as well as a wide range of parameters defining the processes following the collision.

3.2 FASTJET

`FASTJET` [2, 3] is a package providing a wide range of tools for jet finding and analysis. All widely used $2 \to 1$ sequential recombination jet algorithms, like the anti-k_T algorithm, are implemented in the package, and all cone algorithms can also be included via plugin. `FASTJET` also offers tools for estimating pileup and underlying event noise, jet area determination as well as suppression of noise in jets.

The use of `FASTJET` in this work is quite simple. First a jet definition is defined by choosing the jet reconstruction algorithm and the jet cone radius. Then, after events have been generated, a clustering is done for the final particles according to the jet definition, and this gives out a list of jets. This list can then be looped over to analyze the jets, or further, the jet constituents.

4 Code and settings

All the code for this work, simulation and analysis, can be found from https://github.com/lmhuhta/Jets.

First I simulate a pp event with PYTHIA (version 8.307 is used in this work). Configurations for how PYTHIA simulates the event are given either in the separate PYTHIA configuration file, or as input to the code. These setting include, among others, the number of simulated events, the beam energies and particles, and whether we include initial- and final-state radiation in the simulation. In our case, the parton transverse momentum, $\hat{p}_T$, minimum and maximum values are given as input values to `jetreco.C`.

Two proton beams are simulated with PYTHIA at a center-of-mass energy of 7 TeV. The setting HardQCD is set to be on, which means that all QCD jet+jet processes are switched on. This requires the setting of a minimum $\hat{p}_T$, which is set to be 5 GeV in this work. Multiple parton interactions, initial-state and final-state radiation and hadronization are also all switched on.

In this work, we use the anti-k_T algorithm for reconstructing the jets. Also, the jet cone radius R is chosen to be 0.4, as is the case in most ALICE papers.

As for settings used during the PYTHIA event generation, we implement a cut for the minimum pT of particles, 0.15 GeV, as well as a pseudorapidity cut for jets of $|\eta| < 1.2$. Thus, the corresponding pseudorapidity cut for jet constituents is $|\eta| < 1.2 + R$, where R is the previously mentioned jet cone radius. A jet transverse momentum cut $p_T > 5$ GeV is implemented to disregard low p_T jets.

The amount of events simulated amounts to 6 million, the computing time being the major factor in the amount of statistics available. Running a million events on my laptop took around 1.5 hours, so a larger amount of statistics was not feasible for this work. Also, this limited the runs to only proton-proton runs, since a proton-lead or lead-lead run would take even longer to run.

Initially, it was tested to run the code using a card from class `AliJCard` as input, which would then allow the histograms to be created in the class `JHistos`. However, this would cause issues in the code which I was not able to solve (see example of error output in Appendix A.1.). Therefore, the histograms are created in the same code as they are filled, in `jetreco.C`.

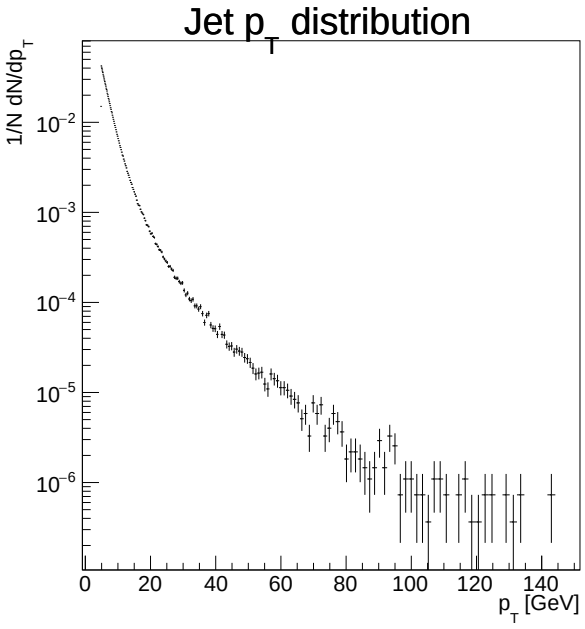


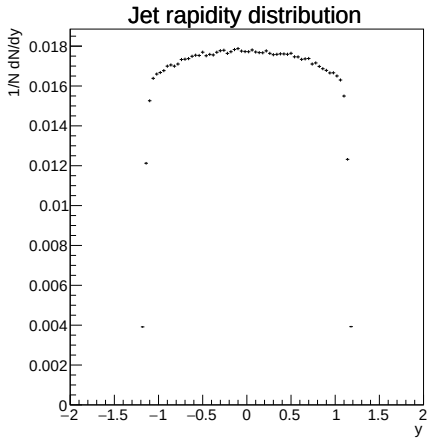
Figure 1: The transverse momentum p_T distribution of all accepted jets, normalized by the amount of jets N.

5 Results

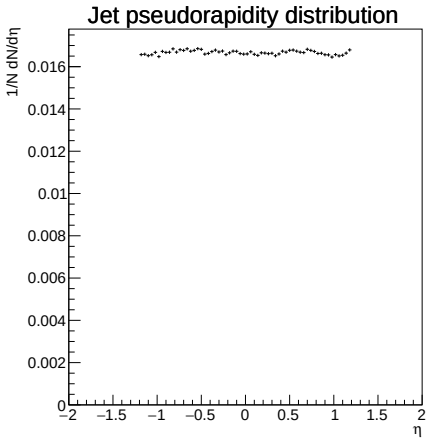
I begin by showing the jet momentum, rapidity and pseudorapidity distributions, see figures 1 and 2. These figures are of all the accepted jets in the study, normalized by the number of all jets, N. It is clearly seen, as expected, that jets of lower momenta are much more common than those of higher momenta. The limited amount of statistics available affects the large uncertainties at high momenta. The jet p_T cut of 5 GeV is visible in the p_T spectrum. The implemented rapidity cuts are clearly visible in the jet rapidity and pseudorapidity figures. The figures are as expected.

The jet fragmentation function $D(z)$ of the splitting fraction z, normalized by the amount of jets for each p_T bin, N_{jet}, is shown in figure 3. The fragmentation function is plotted for five different p_T bins, as marked in the figure. The statistics decrease significantly as the jet momentum increases, as was already seen in figure 1, resulting in only the five bins being shown.

For the two bins of smallest momentum, the distribution is shifted more towards the larger z values, while for momentum bins above 20 GeV the shape of the distribution is very similar for the different bins. This may



(a) The jet rapidity spectrum.



(b) The jet pseudorapidity spectrum.

Figure 2: The jet rapidity and pseudorapidity spectra, both normalized by the amount of jets N.

Table 1: Slope parameters and their erros given by fitting an exponential function to the jet fragmentation function distributions for given p_T bins.

p_T [GeV]	slope	error
5-10	-5.0823	0.0046
10-20	-6.4424	0.0153
20-30	-7.1768	0.0629
30-40	-7.3370	0.1180
40-60	-7.4053	0.1521

be explained by considering how improbable it is for a high momentum jet to contain a parton that is close to its total momentum, while for a small momentum jet the probability is higher, as the momentum scale is much smaller to begin with. Also, since we use the `PYTHIA` setting HardQCD, it might play a part in how the lowest-p_T jets behave.

An interesting question is whether the slope of the fragmentation function is universal or whether it depends on the p_T bin. I fitted an exponential function to the distributions, which can be seen in figure 4. The slope parameters given by the fits are presented in the figure as well as in table 1. The range of the fit was chosen separately for each distribution, leaving out the

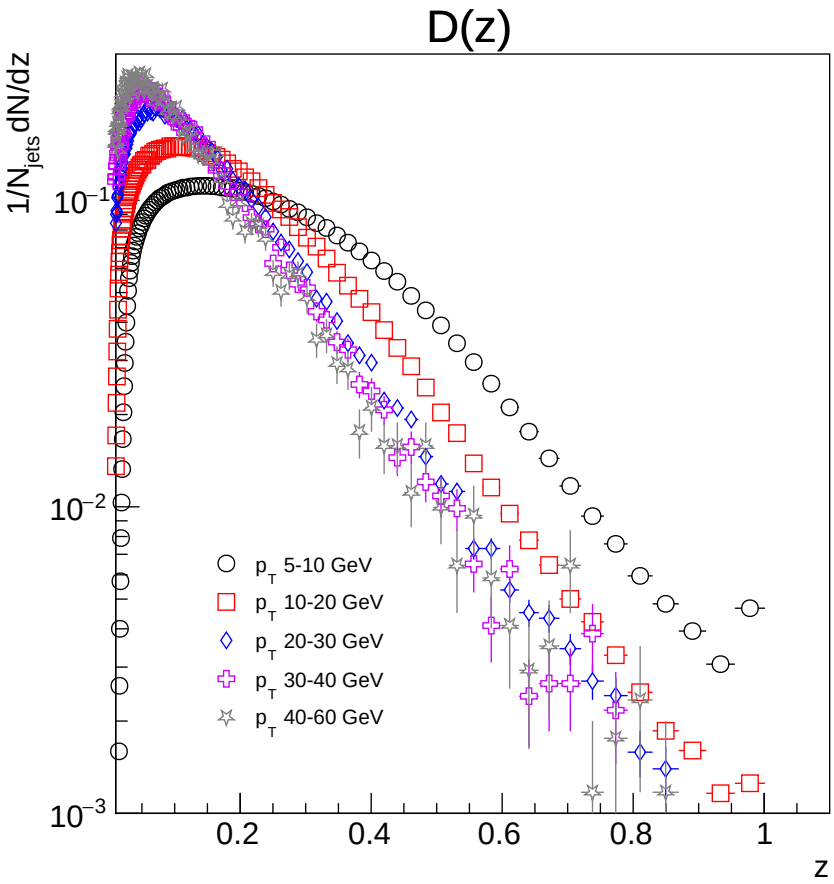


Figure 3: The jet fragmentation function $D(z)$ given by the splitting fraction z, normalized by the amount of jets for each p_T bin, N_{jet}.

final bin, which unexpectedly shows a higher value than the previous bins.

The most striking difference in the slope parameters is between those of the lowest two and the highest three p_T bins. The values of the slope for the lower p_T bins are much smaller than the rest, and this agrees with the previous observation of the shifted shape of the fragmentation function distribution. The slopes for the largest three p_T bins are very similar.

This result can be compared with some ALICE published results [5], as in figure 5. It is to be noted that the ALICE results are for charged particles only, while this work considers all particles. For the bins corresponding to the three largest p_T bins in this work, the functions seem to follow a common shape and have a very similar slope, which fits with our findings.

Notably, the ALICE results are only shown for jet momenta 20 GeV and up, so we cannot compare the small p_T bin behavior. This also seems to be the case for most other experiments, with the smallest p_T bins I found ranging down to 15 GeV at minimum. [6]

As the final figure, the fragmentation function $D(\xi)$ for the variable ξ is plotted in figure 6. This can also be compared to a figure from an ALICE paper, see figure 7. The general shape of the distribution appears similar when comparing the two figures, and the change in shape with rising p_T bins also acts similarly between the two figures. As jet p_T increases, the area of the distribution increases, which shows the rise in particle multiplicity of the jets. The maximum also shifts to higher values of ξ, which corresponds to lower values of z.

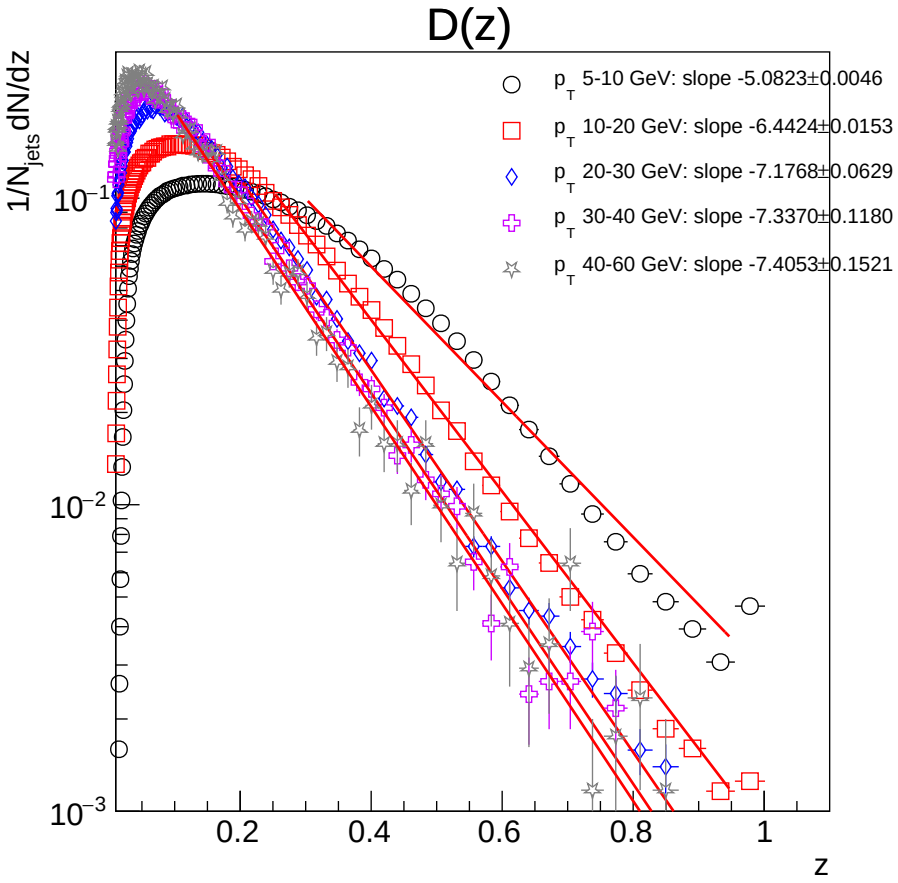


Figure 4: The jet fragmentation function $D(z)$ of the splitting fraction z, normalized by the amount of jets for each p_T bin, N_{jet}. Exponential fits have been fitted for shown sections of the distributions.

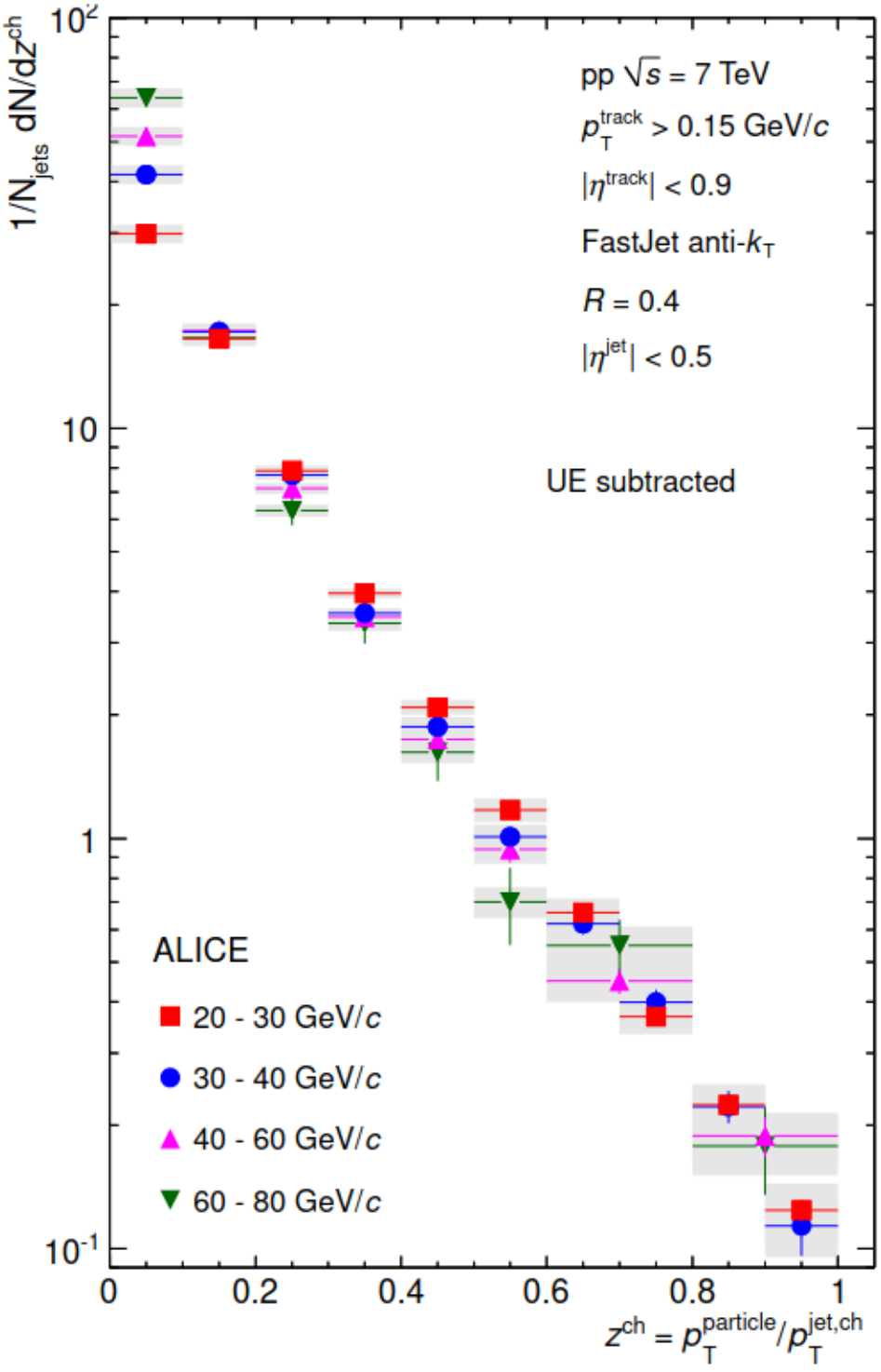


Figure 5: The jet fragmentation function $D(z)$, from [5], for charged particles.

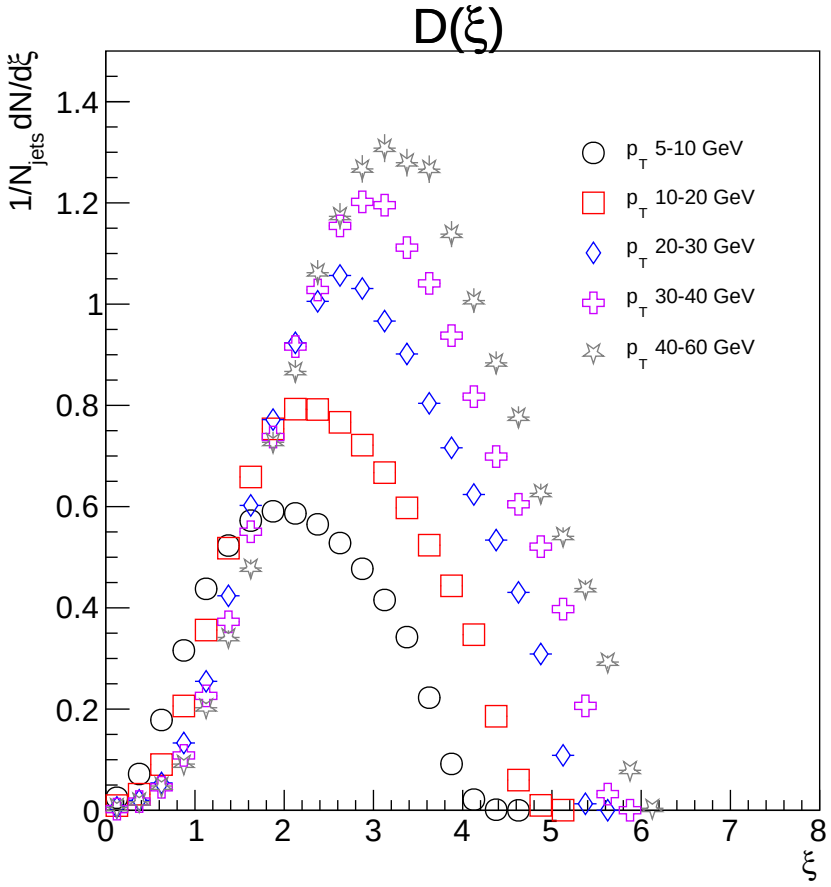


Figure 6: The jet fragmentation function $D(\xi)$, normalized by the amount of jets for each p_T bin, N_{jet}.

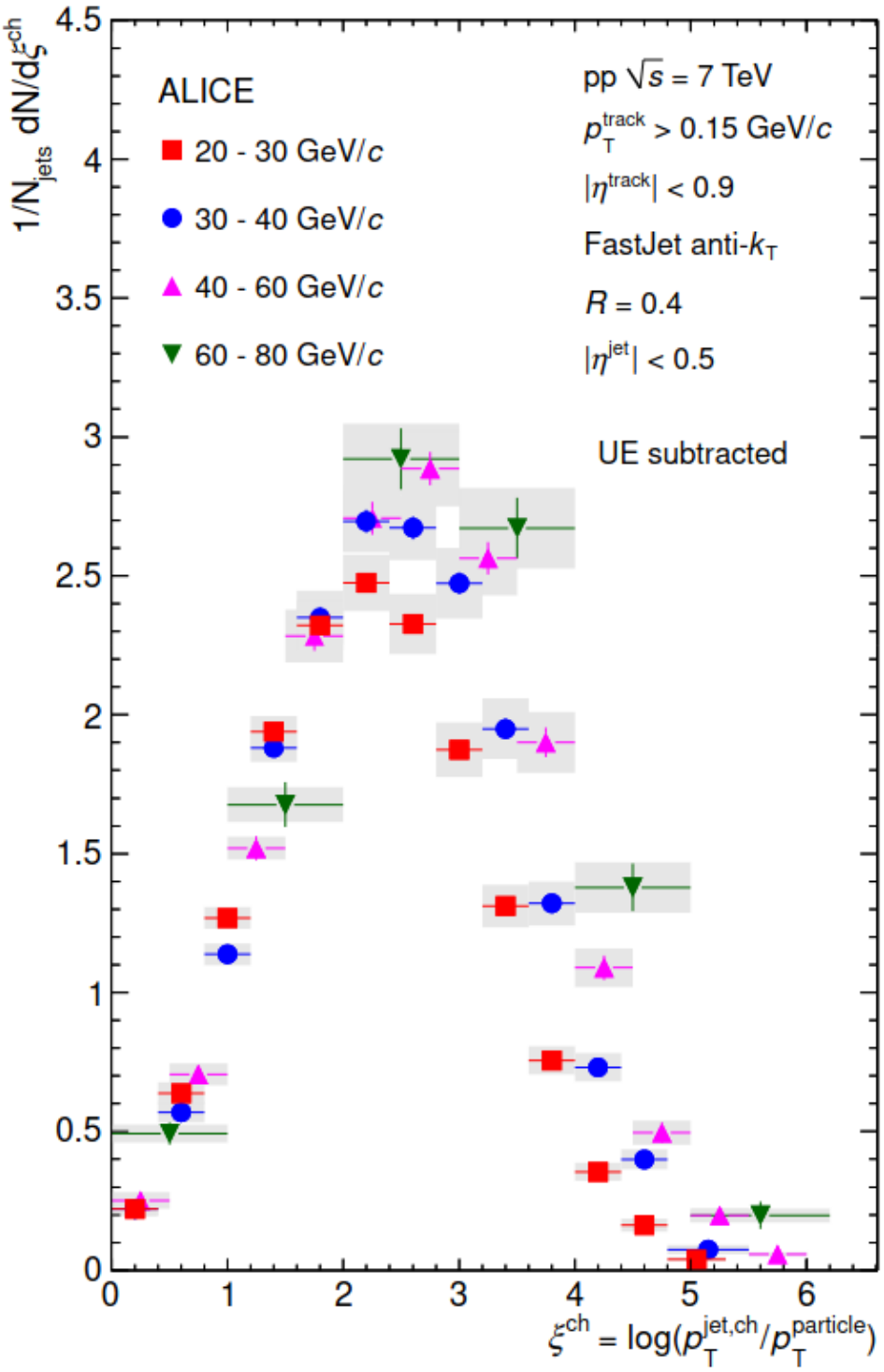


Figure 7: The jet fragmentation function $D(\xi)$, from [5], for charged particles.

6 Conclusions

In this report, I have briefly described the methods used to generate PYTHIA pp events and construct jets from the PYTHIA output with FASTJET , and then briefly analyzed these jets. Basic jet properties, like the splitting fractions and resulting fragmentation functions, were plotted.

It was found that the slope of the jet fragmentation function $D(z)$ differs between the lowest two and the highest three p_T bins, but stays near-constant between the three highest p_T bins. The shapes of the distributions $D(z)$ and $D(\xi)$ simulated in this work are similar to those seen in ALICE data for the three highest p_T bins, but the lowest bins could not be compared, as there is no data to compare to.

Other interesting topics to study using similar methods include simulating proton-lead and lead-lead collision events, for example, and performing a similar jet analysis on those events. These collisions, especially the lead-lead collisions, would cause a significant amount of background surrounding the jet under study. This analysis implements no background subtraction, so including that would be a remaining task. Also, notably, the produced medium causes jet energy loss, which could be taken into account with other MC generators.

Bibliography

[1] Torbjörn Sjöstrand, Stefan Ask, Jesper R. Christiansen, Richard Corke, Nishita Desai, Philip Ilten, Stephen Mrenna, Stefan Prestel, Christine O. Rasmussen, and Peter Z. Skands. An introduction to PYTHIA 8.2. *Comput. Phys. Commun.*, 191:159–177, 2015.

[2] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. FastJet User Manual. *Eur. Phys. J. C*, 72:1896, 2012.

[3] Matteo Cacciari and Gavin P. Salam. Dispelling the N^3 myth for the k_t jet-finder. *Phys. Lett. B*, 641:57–61, 2006.

[4] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. The anti-k_t jet clustering algorithm. *JHEP*, 04:063, 2008.

[5] Betty Bezverkhny Abelev et al. Charged jet cross sections and properties in proton-proton collisions at $\sqrt{s} = 7$ TeV. *Phys. Rev. D*, 91(11):112012, 2015.

[6] Saehanseul Oh. Jet shapes and fragmentation functions in Au+Au collisions at $\sqrt{s_{NN}} = 200$GeV in STAR. *Nucl. Phys. A*, 1005:121808, 2021.

Appendix

A.1. Error message when using AliJCard input

```
*** Break *** segmentation violation


============================================================
There was a crash.
This is the entire stack trace of all threads:
============================================================
#0  0x00007fb28ce86c3a in wait4 () from
↪  /lib/x86_64-linux-gnu/libc.so.6
#1  0x00007fb28cdf5f67 in ?? () from
↪  /lib/x86_64-linux-gnu/libc.so.6
#2  0x00007fb28f89f18c in TUnixSystem::Exec
↪  (shellcmd=<optimized out>, this=0x1ef05a0) at
↪  /mnt/mesos/sandbox/sandbox/jenkins/workspace/DailyBuilds
↪  /Daily02-ubuntu2004/daily-tags.qsRUFjGVdh/SOURCES/ROOT/v
↪  6-26-04-patches-alice1/v6-26-04-patches-alice1/core/unix
↪  /src/TUnixSystem.cxx:2108
#3  TUnixSystem::StackTrace (this=0x1ef05a0) at
↪  /mnt/mesos/sandbox/sandbox/jenkins/workspace/DailyBuilds
↪  /Daily02-ubuntu2004/daily-tags.qsRUFjGVdh/SOURCES/ROOT/v
↪  6-26-04-patches-alice1/v6-26-04-patches-alice1/core/unix
↪  /src/TUnixSystem.cxx:2399
#4  0x00007fb28f89c8f5 in TUnixSystem::DispatchSignals
↪  (this=0x1ef05a0, sig=kSigSegmentationViolation) at
↪  /mnt/mesos/sandbox/sandbox/jenkins/workspace/DailyBuilds
↪  /Daily02-ubuntu2004/daily-tags.qsRUFjGVdh/SOURCES/ROOT/v
↪  6-26-04-patches-alice1/v6-26-04-patches-alice1/core/unix
↪  /src/TUnixSystem.cxx:3619
#5  <signal handler called>
#6  TDirectoryFile::Write (this=this
```

```
entry=0x1f98ca0, opt=opt
entry=0, bufsize=bufsize
entry=0) at /mnt/mesos/sandbox/sandbox/jenkins/workspace/Daily
↪  Builds/Daily02-ubuntu2004/daily-tags.qsRUFjGVdh/SOURCES/RO
↪  OT/v6-26-04-patches-alice1/v6-26-04-patches-alice1/io/io/s
↪  rc/TDirectoryFile.cxx:1830
#7  0x00007fb28f340540 in TFile::Write (this=0x1f98ca0,
↪  opt=0, bufsiz=0) at /mnt/mesos/sandbox/sandbox/jenkins/w
↪  orkspace/DailyBuilds/Daily02-ubuntu2004/daily-tags.qsRUF
↪  jGVdh/SOURCES/ROOT/v6-26-04-patches-alice1/v6-26-04-patc
↪  hes-alice1/io/io/src/TFile.cxx:2393
#8  0x0000000000411469 in main (argc=<optimized out>,
↪  argv=<optimized out>) at jetreco.C:201
============================================================


The lines below might hint at the cause of the crash.
You may get help by asking at the ROOT forum
↪  https://root.cern/forum
Only if you are really convinced it is a bug in ROOT then
↪  please submit a
report at https://root.cern/bugs Please post the ENTIRE stack
↪  trace
from above as an attachment in addition to anything else
that might help us fixing this issue.
============================================================
#6  TDirectoryFile::Write (this=this
entry=0x1f98ca0, opt=opt
entry=0, bufsize=bufsize
```

```
entry=0) at /mnt/mesos/sandbox/sandbox/jenkins/workspace/Daily
↪  Builds/DailyO2-ubuntu2004/daily-tags.qsRUFjGVdh/SOURCES/RO
↪  OT/v6-26-04-patches-alice1/v6-26-04-patches-alice1/io/io/s
↪  rc/TDirectoryFile.cxx:1830
#7  0x00007fb28f340540 in TFile::Write (this=0x1f98ca0,
↪  opt=0, bufsiz=0) at /mnt/mesos/sandbox/sandbox/jenkins/w
↪  orkspace/DailyBuilds/DailyO2-ubuntu2004/daily-tags.qsRUF
↪  jGVdh/SOURCES/ROOT/v6-26-04-patches-alice1/v6-26-04-patc
↪  hes-alice1/io/io/src/TFile.cxx:2393
#8  0x0000000000411469 in main (argc=<optimized out>,
↪  argv=<optimized out>) at jetreco.C:201
============================================================


terminate called after throwing an instance of
↪  'std::bad_weak_ptr'
  what():  bad_weak_ptr
Aborted (core dumped)
```