b-jet shape analysis using Monte Carlo methods.

Luke Pomfrey

Physics and Astronomy Dept.,
University College London

March, 2009

Abstract

The accuracy of Monte Carlo simulations in describing the shape of b-quark jets has been investigated before. Here, this is investigated more closely, comparing several tunes of PYTHIA6 and also HERWIG and JIMMY. Jets with transverse momentum in the range 52 to 300 GeV/c are investigated, and the experimental data used is from the upgraded Collider Detector at Fermilab (CDF II) in $p\overline{p}$ collisions at a centre of mass energy $\sqrt{s} = 1.96$ TeV. The effect of the simulated quark masses in PYTHIA6 are also investigated, and a tune of PYTHIA6 attempted.

The results confirm the previous conclusion, that the jets produced in Monte Carlo simulations are too narrow. It can also be seen that the quark masses in PYTHIA6 are not a factor in determining the b-jet shape. It is shown that by increasing the amount of b-quarks PYTHIA6 produce in parton showers a better result is obtained.

Contents

1 Introduction

The Monte Carlo (MC) method[1] is important in high energy physics (HEP), both in order to make predictions of new physics and to test current physics assumptions and theories. By tuning these MC generators it is possible to not only make better predictions but also learn more about the underlying physics of the standard model.

By measuring jet shapes, studies of the processes occurring between the initial hard interaction and the collimated flows of hadrons that are experimentally observed can be performed. [2] b-jet shapes are particularly interesting as a method of studying heavy flavour production and, since the Higgs boson should couple preferentially to the heavy quarks, they are a good candidate to be used in Higgs searches. [3]

Studying jet shapes computationally, in leading order (LO) MC generators, is problematic due to high order QCD corrections that must be made. In practise various parton shower methods are implemented. It is also highly important to have an understanding of the process of hadronisation, and in the case of the heavy quarks, a good understanding of the quark decay processes. Jet shapes are sensitive to a variety of factors including whether the initial hard-scattered particle is a gluon or quark, and the initial quark flavour. [4] For heavy flavour jets, the jet shape is sensitive to the contributions from the various production mechanisms. In b-jets the b- and $\overline{\text{b}}$-quarks are expected to be in the same jet, resulting in significantly broader jets than those from flavour creation. [5] For this reason the fraction of gluon splitting events is particularly important in MC simulations. Also important in jet production, and in determining the jet shape, is the underlying event, consisting of the initial- and final-state radiations, multiple particle interactions and beam-beam remnants. [4]

Here, the jet shapes — describing the transverse momentum[2] (p_T) of the jet as a function of distance from the jet axis — simulated in the LO MC generators PYTHIA6 [6, 7] and HERWIG [8, 9] are compared to b-jet shape data collected at the Collider Detector at Fermilab (CDF II). [10] Several tunes of PYTHIA6 are compared, these being Tune A, the S0-Pro tune, and the LEP tune.The old and new, p_T-ordered, shower methods in PYTHIA6, the dependence of PYTHIA6 on quark masses, and the effect of the JIMMY [11] add on for HERWIG are investigated. An attempt to tune PYTHIA6 to better describe the CDF data is also made.

Inclusive jet shapes have been studied at CDF previously and it has been seen that MC simulations can describe the data well [12]. In inclusive jets PYTHIA6 has been shown to provide a better description of measured data than HERWIG. Heavy flavour jets, however, are not described as well as inclusive jets in LO MC generators. When c-jet shapes were studied at HERA deviations were found between the MC simulations and the data, especially in the region where gluon splitting is expected to contribute significantly to the jet shape. [13] The CDF collaboration has previously concluded that the contribution of gluon splitting in heavy flavour jet shapes is roughly double that expected from PYTHIA6. [14] This was inferred by investigating $\text{b}\overline{\text{b}}$ azimuthal correlations in $\text{p}\overline{\text{p}}$ collisions at 1.8 TeV by measuring the azimuthal angle difference, $\Delta\phi$ between two heavy-flavour jets tagged in semileptonic b-hadron decay. The minimum b-hadron p_T in this study was 14 GeV/c. When next to leading order (NLO) MC generators were used the results obtained were closer to actual data. Conclusions reached by the DØ collaboration when studying $\text{b}\overline{\text{b}}$ production cross section and azimuthal correlation were similar. [15] The b-hadron p_T range in this study was $6 < p_T < 30$ GeV/c. It was found that the NLO generator enhances the $\text{b}\overline{\text{b}}$ production in the $\Delta\phi < 1$ region compared to the LO generators, which have no gluon splitting in the matrix element. A second measurement at CDF with jet transverse energies of $E_T > 35$ GeV confirmed the enhancement, above the PYTHIA6 and HERWIG simulations, for $\Delta\phi < 1$. [16] A final investigation by CDF, using the same analysis and data set as in this investigation, again confirmed that the measured b-jet shapes are sig-

[1] A good description of the Monte Carlo method and it's origins can be found in [1].

[2] The transverse momentum describes the momentum transverse to the beam axis. The transverse energy, E_T, is the energy multiplied by the sine of the angle (with respect to the beam axis). The pseudorapidity is defined as $\eta = -\ln\left(\tan\left(\frac{\theta}{2}\right)\right)$ and the rapidity as $y = \frac{1}{2}\ln\left(\frac{E+p_z}{E-p_z}\right)$. θ is the polar angle with respect to the beam axis, ϕ is is in the plane orthogonal to beam direction, and p_z is the component of momentum along the beam direction. [4]

nificantly broader than inclusive jets and significantly broader than the b-jets simulated in PYTHIA6 and HERWIG. This was again attributed to the underestimation in LO MC simulations of the fraction of b-jets originating from gluon splitting. It was noted that measured jet shapes show, and NLO simulations predict, a higher rate of b-jets containing more than one b-quarks in the jet cone compared to LO MC simulations. By increasing the double b-quark jet fraction by 0.2 in PYTHIA6 and HERWIG a better description of the measured b-jet shapes was found. [4] An attempt to correct this here is made by tuning PYTHIA6 to produce more heavy quarks.

This paper is organised in the following way. §2 describes how the jet shape is defined. §3 describes the collection and analysis of the original CDF data sample. §4 covers the generation of events using the MC generators. §5 covers the initial analysis of the generated events using AGILe and RIVET and how the MC data were compared in a quantitative way. Finally, §6 describes the results obtained.

2 Defining the jet shape

Jet shapes are defined here as the distribution, as a function of the distance away from the jet axis, of the fractional transverse momentum, p_T, inside the jet. [4] Simply put, the jet shape provides a measure of the fraction of the total p_T of the jet inside a given radius in (y, ϕ)-space from the jet axis. If we define the radius of the jet cone as r_0. We can then introduce the integrated jet shape, $\Psi\left(\frac{r}{r_0}\right)$, as being the fraction of the total p_T of the jet carried by particles in a sub-cone with a radius r about the jet axis. In the Monte Carlo simulations we can compute this quantity at the hadron level from the final state hadrons. In the measured CDF data this is computed using calorimeter energy deposits and charged particle tracks from the tracking subsystems. [4]

We can now express, mathematically, the integrated jet shape as

$$\Psi\left(\frac{r}{r_0}\right) = \left\langle \frac{p_T(0 \to r)}{p_T(0 \to r_0)} \right\rangle \tag{1}$$

where $p_T(0 \to r)$ is the sum of the transverse momenta of all of the particles inside the sub-cone, and $p_T(0 \to r_0)$ is the sum of the transverse momenta of all of the particles in the jet cone, *i.e.* the total transverse momentum of the jet. The integrated jet shape is, by definition, normalised such that the following are true

$$\Psi\left(\frac{r}{r_0} = 1\right) = 1 \tag{2a}$$

$$\Psi\left(\frac{r}{r_0} = 0\right) = 0 \tag{2b}$$

Particles outside of the jet cone radius are not considered. [4]

For the purposes of the Monte Carlo simulations we define our b-jets to be simply jets that contain one, or more, b-quarks inside the jet cone. The b-jets in the CDF data sample were obtained as described in §3 and in [4].

3 The CDF jet shape data

The data from the CDF measured jet shapes is tabulated in Table 1. This is the data that the Monte Carlo generators will be compared to in this paper.

The collection of the jet shape data from CDF is fully described in [4]. The data was collected in $p\overline{p}$ collisions, at a centre of mass energy of $\sqrt{s} = 1.96$ TeV, at the upgraded Collider Detector at Fermilab (CDF II). Here basic details on the analysis are provided.

3.1 The CDF detector

The upgraded CDF detector, CDF II, is a general purpose detector for the study of $p\overline{p}$ collisions. A full description of the detector is not essential here and can be found elsewhere. [10] The most important subsystems of the detector for the collection of the measured jet shape data are the tracking systems and the calorimeters.

The tracking system comprises an open-cell drift chamber and silicon micro-vertex detectors, lying in a superconducting magnetic solenoid with a field strength of 1.4 T. The silicon micro-vertex detectors cover the region where $|\eta| \leqslant 2$ and has the highest resolution for the reconstruction of displaced vertices. The drift chamber covers the region $|\eta| \leqslant 1$ and is used to measure the momentum of charged particles. The calorimeters lie outside the solenoid and measure the energy flow of particles in the region $|\eta| \leqslant 3.6$. They are segmented and arranged in a projective tower geometry. The transverse momentum associated with any tower is computed by assuming that total tower

$\frac{r}{R}$	$\Psi\left(\frac{r}{R}\right) \pm \sigma_{\text{stat}} \pm \sigma_{\text{sys}}$
$\frac{0.1}{0.7} \approx 0.14$	$0.283 \pm 0.010 \pm 0.105$
$\frac{0.2}{0.7} \approx 0.28$	$0.553 \pm 0.010 \pm 0.076$
$\frac{0.3}{0.7} \approx 0.42$	$0.717 \pm 0.007 \pm 0.068$
$\frac{0.4}{0.7} \approx 0.57$	$0.825 \pm 0.005 \pm 0.037$
$\frac{0.5}{0.7} \approx 0.71$	$0.901 \pm 0.003 \pm 0.015$
$\frac{0.6}{0.7} \approx 0.86$	$0.953 \pm 0.002 \pm 0.006$
$\frac{0.7}{0.7} = 1.00$	$1.000 \pm 0.000 \pm 0.000$

(a) $52 \leqslant p_T < 80$ GeV/c

$\frac{r}{R}$	$\Psi\left(\frac{r}{R}\right) \pm \sigma_{\text{stat}} \pm \sigma_{\text{sys}}$
$\frac{0.1}{0.7} \approx 0.14$	$0.336 \pm 0.007 \pm 0.059$
$\frac{0.2}{0.7} \approx 0.28$	$0.565 \pm 0.006 \pm 0.051$
$\frac{0.3}{0.7} \approx 0.42$	$0.710 \pm 0.004 \pm 0.039$
$\frac{0.4}{0.7} \approx 0.57$	$0.817 \pm 0.003 \pm 0.024$
$\frac{0.5}{0.7} \approx 0.71$	$0.898 \pm 0.002 \pm 0.017$
$\frac{0.6}{0.7} \approx 0.86$	$0.957 \pm 0.001 \pm 0.006$
$\frac{0.7}{0.7} = 1.00$	$1.000 \pm 0.000 \pm 0.000$

(b) $80 \leqslant p_T < 104$ GeV/c

$\frac{r}{R}$	$\Psi\left(\frac{r}{R}\right) \pm \sigma_{\text{stat}} \pm \sigma_{\text{sys}}$
$\frac{0.1}{0.7} \approx 0.14$	$0.403 \pm 0.008 \pm 0.064$
$\frac{0.2}{0.7} \approx 0.28$	$0.623 \pm 0.007 \pm 0.024$
$\frac{0.3}{0.7} \approx 0.42$	$0.747 \pm 0.005 \pm 0.026$
$\frac{0.4}{0.7} \approx 0.57$	$0.837 \pm 0.003 \pm 0.019$
$\frac{0.5}{0.7} \approx 0.71$	$0.906 \pm 0.002 \pm 0.010$
$\frac{0.6}{0.7} \approx 0.86$	$0.963 \pm 0.001 \pm 0.004$
$\frac{0.7}{0.7} = 1.00$	$1.000 \pm 0.000 \pm 0.000$

(c) $104 \leqslant p_T < 142$ GeV/c

$\frac{r}{R}$	$\Psi\left(\frac{r}{R}\right) \pm \sigma_{\text{stat}} \pm \sigma_{\text{sys}}$
$\frac{0.1}{0.7} \approx 0.14$	$0.413 \pm 0.008 \pm 0.048$
$\frac{0.2}{0.7} \approx 0.28$	$0.637 \pm 0.006 \pm 0.037$
$\frac{0.3}{0.7} \approx 0.42$	$0.760 \pm 0.005 \pm 0.020$
$\frac{0.4}{0.7} \approx 0.57$	$0.849 \pm 0.003 \pm 0.013$
$\frac{0.5}{0.7} \approx 0.71$	$0.919 \pm 0.002 \pm 0.007$
$\frac{0.6}{0.7} \approx 0.86$	$0.966 \pm 0.001 \pm 0.008$
$\frac{0.7}{0.7} = 1.00$	$1.000 \pm 0.000 \pm 0.000$

(d) $142 \leqslant p_T < 300$ GeV/c

Table 1: Integrated jet shapes for b-jets from CDF measurements with statistical and systematic error values for four p_T ranges. [4] This is the data that Monte Carlo simulation results will be compared to in this paper.

momentum is given by the tower energy. The momentum vector is then assumed to be parallel to the vector linking the primary interaction and the tower, and the transverse momentum is the projection of the momentum vector projected onto the plane perpendicular to the beam. A three-level trigger system is then used to select events. [4]

3.2 Jet reconstruction, event selection and tagging

The jets were reconstructed using the Mid-Point cone algorithm using a cone size of 0.7. [4,17,18] The hadronic and electromagnetic sections of each calorimeter tower were combined into physics towers, each physics tower with $p_T > 1$GeV/c was considered as a seed around which a jet could form.

The measured data comes from four data-sets of central jets, $|y| \leqslant 0.7$. The events collected satisfy the conditions required by the inclusive jet trigger, the different data-sets each have a different lower E_T threshold. Each data-set is defined by their trigger path that has requirements at the trigger levels L1, L2, and L3. The inclusive jet triggers are only dependent on the transverse energy of the jet. The jet triggers are not fully efficient at the trigger threshold and thus, to avoid trigger bias, events are only considered when the trigger efficiency is above 99%. The trigger levels and paths are tabulated in Table 2. [4]

Inclusive jet data mostly contains light-flavour jets and gluon jets, so a secondary vertex tagger is used to increase the proportion of b-jets. [19] The vertex tagger utilises the fact that, due to their relativistic boost, the b-hadrons (and other heavy-flavour hadrons) travel a few millimetres before their decay. The tagging algorithm reconstructs the secondary/displaced vertex by reconstructing the charged particle tracks within a cone of radius 0.4 around the jet axis. This cone is smaller than the jet cone (radius 0.7) since the direction of the b-hadrons are generally close to the jet axis. A larger cone would increase the rate of false-positive secondary vertex tagging. In order to be considered a b-jet the jet must be tagged by the secondary vertex algorithm. The

Trigger Path	L1 Tower E_T / GeV	L2 Cluster E_T / GeV	L3 Jet E_T / GeV	99% trig. eff. Offline p_T / GeV/c
JET 20	5	15	20	52
JET 50	5	40	50	80
JET 70	10	60	70	104
JET 100	10	90	100	142

Table 2: Trigger paths and thresholds from the CDF trigger system. The last column shows the final offline cuts applied to corrected jets after the 90% trigger efficiency requirement has been applied. [4]

tagging algorithm is described in [4] and, in more detail, in [19].

Once the samples of inclusive and tagged jets had been compiled, the hadron level b-jet shape was extracted by processing the data to remove biases and errors due to b-quark jet content, detector level biases, purity, biases in the secondary vertex tagging, and hadron level corrections in order to give the final data. [4]

4 The Monte Carlo generators

In this paper the use of several LO MC generators and tunes for those generators is described. The main generator studied is PYTHIA6 [6] (version 6.419), and several tunes for it, these being;

- Tune A (a tune to the underlying event of CDF Run I [20], it is noted that underlying event tuning is important for a good description of inclusive jet shapes [12]),
- the LEP tune (a tune produced with the PROFESSOR tuning tool [21] for data from the LEP experiment) and,
- the S0-Pro tune (a combination of the S0 and LEP tunes).

For the LEP and S0-Pro tunes the effectiveness of PYTHIA6's new p_T-ordered shower method over it's older method is also investigated, this is done by setting the parameter `MSTJ(41) = 12` to select the p_T ordered shower. In addition to PYTHIA6 the HERWIG [8, 9] (version 6.510) generator was used, both with and without the JIMMY (version 4.31) add-on.

4.1 PYTHIA6

In PYTHIA6 the final-final state parton shower is implemented in the JETSET part of the code [22,23]. In order to compute the parton shower PYTHIA6 uses splitting functions and an ordering in the virtuality scale Q^2, which is equal to the square of the four-momentum of the branching parton. The splitting functions, $P_{a\to bc}(z)$ describe the probability that a parton, a, will split into two partons, b and c, with one of those partons carrying away a particular momentum fraction of the initial parton, z. PYTHIA6 uses the Lund string model for non-perturbative hadronisation and fragmentation. In the Lund model the long-range confinement forces distribute the energies and flavours of a parton configuration in the primary hadrons, these may then decay further. When simulating the decay of hadrons containing heavy quarks PYTHIA6 uses decay tables of the QQ program. QQ is called to handle the decay whenever PYTHIA6 produces a hadron containing heavy quarks.

4.2 HERWIG

In the parton shower process, the main difference between HERWIG and PYTHIA6 is that, instead of PYTHIA6's Q^2 ordering, HERWIG uses an angular ordering of the successive emissions to simulate the colour flow. [8, 9] HERWIG uses the cluster model to simulate fragmentation. [24] This model is independent of he initial process and of energy. In it clusters of colour neutral partons decay into the final hadrons. The decay of heavy hadrons in HERWIG is performed using an exponential law depending on the mean lifetime of the hadron. Without the JIMMY add-on HERWIG itself does not simulate

multi-parton interactions, for this reason it is expected that HERWIG will perform better with the JIMMY add-on which provides a method for simulating these interactions.

4.3 Generating the samples and the p_T cuts used

In order to generate the inclusive dijet samples required for analysis the `MSEL = 1` parameter was passed to PYTHIA6 and the `IPROC = 1500` parameter was passed to HERWIG. In both generated the minimum and maximum p_T cuts were set at 50 GeV/c and 302 GeV/c, respectively. When comparing the generators the effects of restricting the generators into only producing events that contain a b-quark were also studied.

4.4 Quark masses in PYTHIA6

In PYTHIA6 the effect of changing the quark rest masses in simulations is studied. One may intuitively expect this to affect the production of b- and inclusive-jets, however, since in a lot of the physics incorporated into the parton showering code in PYTHIA6 the quarks are considered mass-less this may not be the case. When changing the quark masses the quark mass values given by the Particle Data Group [25] are used, with values being taken from both the nominal value given along with the upper and lower error bar values. The quark rest masses from PYTHIA6 and the masses from the Particle Data Group are tabulated in Table 3.

4.5 Energy partitioning in PYTHIA6

In PYTHIA6 the energy partitioning when a hadron or resolved-photon remnant is split into two jets, the energy fraction, χ taken by one of the two jets is proportional to

$$\frac{(1-\chi)^k}{\sqrt{4\chi^2 + c_{\min}^2}} \tag{3}$$

Where $c_{\min}$ is defined as

$$c_{\min} = \frac{0.6\text{GeV}}{E_{\text{cm}}} \tag{4}$$

and k is given, in the case when a meson or resolved-photon remnant is split into two fragments, by `PARP(94)` and, in the case when a nucleon remnant is split into a diquark and a quark fragment, by `PARP(96)`, with χ giving the fraction of the energy taken by the quark jet. [6] The energy partitioning when a hadron or resolved-photon remnant is split into a hadron and a remainder-jet, the χ of the hadron is selected according to the normal, Lund model, fragmentation functions.

Here, the effect of varying the `PARP(94)` and `PARP(96)` parameters in an effort to improve the results is investigated.

4.6 Increasing the number of b-quarks in PYTHIA6

It has been suggested that by increasing the number of b-quarks within the jets produced by PYTHIA6 it may be possible to obtain a better result. [4] Here the number of b-quarks is increased by setting the parameter `MSTJ(42) = 4`. This parameter controls the branching mode, especially the coherence level, for time-like showers. [6] With this setting in a branching $a \to bg$ where m_b is non-vanishing, the decay angle is reduced by a factor of

$$\left(1 + \frac{\left(\frac{m_b^2}{m_a^2}\right)(1-z)}{z}\right)^{-1} \tag{5}$$

This allows more branchings from an angular ordering point of view. $g \to gg$ are governed by coherent branching, with angular ordering. $g \to q\overline{q}$ branchings, however, have no angular ordering requirement conditions imposed on them. Whilst this is somewhat unphysical, it is known to elicit a much higher rate of c- and b-quark production in showers. [6]

5 Data analysis

In addition to the MC generators themselves, several packages are used in order to interface with the generators, and analyse the data from them.

5.1 AGILe

AGILe (A Generator Interface Library) [26] is used in order to control and generate data from the MC generators. In provides a consistent method for interfacing with a variety of generators. Generator parameters and other settings are passed to AGILe which initialises the

Quark	PDG mass	PYTHIA6 mass
d	$5.04^{+0.96}_{-1.54}$ MeV	9.9 MeV
u	$2.55^{+0.75}_{-1.05}$ MeV	5.6 MeV
s	104^{+26}_{-34} MeV	199 MeV
c	$1.27^{+0.07}_{-0.11}$ GeV	1.23 GeV
b	$4.20^{+0.17}_{-0.07}$ GeV	4.17 GeV
t	171.2 ± 2.1 GeV	165 GeV

Table 3: Quark rest masses in PYTHIA6 [6] compared to the values given by the Particle Data Group. [25] It can be seen that in most cases the default rest masses in PYTHIA6 fall outside of the error bars given by the Particle Data Group.

required generator with the intended parameters. AGILe then outputs the generated events in the HepMC format. This output can be written to a file or to a pipe. Here output is written to a pipe as it is less prohibitive for RIVET to analyse the data in real time due to the colossal amount of disc space that writing to a file requires.

5.2 RIVET

RIVET (Robust Independent Validation of Experiment and Theory) [27,28] is a toolkit for the validation of MC generators. Here, RIVET reads events from AGILe via a pipe (it can also read from a HepMC file, but as described above, this method would have been prohibitive), it then performs a selected analysis on the generated events and produces histogram data, which is output in the form of an `aida` [29] XML file. RIVET also provides tools for converting this `aida` XML file into other formats for analysis (here, the `aida` XML is converted to ROOT [30] format for the plotting of data). In RIVET, jets are identified using the FastJet [31] library, which implements the longitudinally invariant k_t [32, 33], longitudinally invariant inclusive Cambridge/Aachen [34, 35], and anti-k_t [36] jet finders.

The analysis performed by RIVET for this paper is the `CDF_2008_S7782535` b-jets analysis, originally written for [4]. The C++ source code for this analysis can be found in the appendices and at [27]. For each event this analysis loops over any jets found in the event and checks if they contain b-quarks. If no b-jets are observed in an event then that event is discarded. If an event is found to contain a b-jet, the jets momentum and shape are are stored, ready to be included in the relevant histogram. For each of the transverse momentum ranges $50 \leqslant p_T < 80$ GeV/c, $80 \leqslant p_T < 104$ GeV/c, $104 \leqslant p_T < 142$ GeV/c, and $142 \leqslant p_T < 300$ GeV/c, a histogram is produced to plot the value of the jet shape, $\Psi\left(\frac{r}{r_0}\right)$, against $\frac{r}{r_0}$.

Another way to analyse the jet shape is by plotting $1 - \Psi\left(\frac{0.2}{r_0}\right)$ against p_T, *i.e.* the fraction of the jet's p_T outside a cone of radius 0.2 against p_T. This shows how the jet shape changes as the p_T increases. It is expected that the jets will become narrower as p_T increases due to the running of the strong coupling constant, α_s. [4]

5.3 Analysis of histogrammed data

Once the necessary histograms have been created by RIVET they are plotted in ROOT. In order to give a quantitative measurement when comparing the various generators and tune to themselves, and to the measured data, a χ^2 goodness-of-fit test is performed.

When performing the χ^2 test, the assumption made is that the measured data and the MC data are Gaussian distributed about a mean λ. Then the χ^2 value is given by

$$\chi^2 = \sum \frac{(y_{\text{CDF}} - \lambda)^2}{\sigma^2_{\text{CDF}}} + \frac{(y_{\text{MC}} - \lambda)^2}{\sigma^2_{\text{MC}}} \qquad (6)$$

Where y is the value, σ is the error on the value, and the CDF and MC subscripts represent the measured CDF data and the MC data, respectively. The sum is taken over all bins in the two

histograms. λ is given by

$$\lambda = \frac{\left(\frac{y_{\text{CDF}}}{\sigma^2_{\text{CDF}}}\right) + \left(\frac{y_{\text{MC}}}{\sigma^2_{\text{MC}}}\right)}{\left(\frac{1}{\sigma^2_{\text{CDF}}}\right) + \left(\frac{1}{\sigma^2_{\text{MC}}}\right)} \tag{7}$$

Substituting (7) into (6) gives

$$\chi^2 = \sum \frac{(y_{\text{CDF}} - y_{\text{MC}})^2}{\sigma^2_{\text{CDF}} + \sigma^2_{\text{MC}}} \tag{8}$$

In order to calculate the χ^2 values from the `aida` XML a Python script was written, it can be found in the appendices.

6 Results

6.1 Generator and tune comparisons

Figs. 1 and 2 contain plots of the integral jet shape for the p_T ranges $52 \leqslant p_T < 80$ GeV/c, $80 \leqslant p_T < 104$ GeV/c, $104 \leqslant p_T < 142$ GeV/c, and $142 \leqslant p_T < 300$ GeV/c, and a plot of $1 - \Psi\left(\frac{0.2}{r_0}\right)$ against p_T for the PYTHIA6 tunes, HERWIG, and JIMMY. Table 4 contains χ^2 values for the plots in Figs. 1 and 2.

The first things to note are that instructing the generators to ensure there are b-quarks in each event makes no difference to the results. As has been seen before the jets are, in all cases, too narrow. This was expected from earlier results, particularly those in [4]. Using the average χ^2 values (from Table. 4f) in order to define how closely the output of a generator describes the measured data, it can be seen that HERWIG with the JIMMY add-on is, overall, the most effective. As was explained in §4, it was expected that the use of JIMMY would improve the results from HERWIG and this has proven to be the case. Unlike in studies of inclusive jets, where HERWIG was seen to be less effective at describing the jet shapes, it can be seen that, for the most part HERWIG provides a better description of b-jet shapes than PYTHIA6.

In PYTHIA6 it can be seen that the LEP tune is the most effective at improving the result, the S0-Pro tune is the least effective out of the tunes and generators compared here. The new, p_T ordered, shower method slightly improves the results when using the LEP tune, although it has the opposite effect with the S0-Pro tune.

Another thing that is apparent from both the plots and the χ^2 values is that, in general, the MC generators more closely describe the measured data in the higher p_T ranges. This can also be seen in Fig. 5 which is a plot of the χ^2 values, for all of the generators/tunes, for each p_T binning range. It is also seen that the jets in all cases become narrower as the p_T increases, as expected due to the running of the strong coupling constant.

6.2 Increasing the number of b-quarks in PYTHIA6

As was described in §4, setting the parameter `MSTJ(42) = 4` is known to increase the number of b-quarks produced in showers. Using tune A as a basis, the effect of setting `MSTJ(42) = 4` is investigated and compared to the measured data. Figs. 3 and 4 contain plots of the integral jet shape for the p_T ranges $52 \leqslant p_T < 80$ GeV/c, $80 \leqslant p_T < 104$ GeV/c, $104 \leqslant p_T < 142$ GeV/c, and $142 \leqslant p_T < 300$ GeV/c, and a plot of $1 - \Psi\left(\frac{0.2}{r_0}\right)$ against p_T for tune A with and without setting `MSTJ(42) = 4`. Table. 5 contains χ^2 values for the plots in Figs. 3 and 4.

It can be seen, from both the plots and the χ^2 values, that using the `MSTJ(42) = 4` parameter to increase the number of c- and b-quarks produced in showers produces a marked improvement in the results, with the biggest improvement in the lower p_T regime. By increasing the number of b-quarks in the parton shower there should be an increase in the number of double b-quark jets, which it has been shown are not produced as much in PYTHIA6 as they are in measured data. This would seem to reinforce the conclusion in [4] that the LO MC generators are underestimating the fraction of b-jets originating from gluon splitting.

6.3 Energy partitioning parameters in PYTHIA6

In PYTHIA6 the parameters `PARP(94)` and `PARP(96)` control the energy partitioning in the case when a hadron or resolved-photon remnant is split into two jets. These parameters were used here to attempt to improve Tune A's description of the data. Whilst it was found that it is quite easy to adjust these parameters and obtain a significantly worse result than Tune A,

a set of parameter values was not found that improved on Tune A.

6.4 Quark masses in PYTHIA6

When investigating the quark rest masses in PYTHIA6 they were changed using the parameters `PARF(91)` through `PARF(96)`. The masses were changed to values on the lower end of the Particle Data Group values and to values at the upper end of those measurements (see Table. 3), Tune A was, again, used as the basis. It was seen that changing the quark masses produced no effect on the results, as was suspected due to PYTHIA6's largely mass-less treatment of quarks.

7 Conclusion

Here, b-jet shape data from several LO MC generators, and tunes for those generators, have been compared to b-jet shape data from the CDF II experiment. It has been shown that in all cases the b-jets produced in LO MC simulations are narrower than the measured b-jets. The LO MC generators describe the measured data better in the higher p_T regime. The results from the HERWIG generator have been shown to be, largely, better than those from PYTHIA6 when simulating b-jet shapes, this is opposite to the results seen when investigating inclusive jet shapes previously. Using the JIMMY add-on with HERWIG produces better results than using HERWIG alone. This is due to the ability of JIMMY to simulate multi-parton interactions which affect jet production. Quark masses in PYTHIA6 are unimportant in the production of b-jets, this is due to PYTHIA6's largely mass-less treatment of quarks in parton showers. The newer p_T ordered shower method produces, on average, very similar results to the old shower method. In some cases it improves the data, and in others makes it worse, and in either case the effect is rather small. It can be seen that by increasing the number of b-quarks produced in parton showers in PYTHIA6 the simulated jet shapes better describe the measured data. This reinforces previous conclusions that suggest that the LO MC generators underestimate the fraction of b-jets originating from gluon splitting.

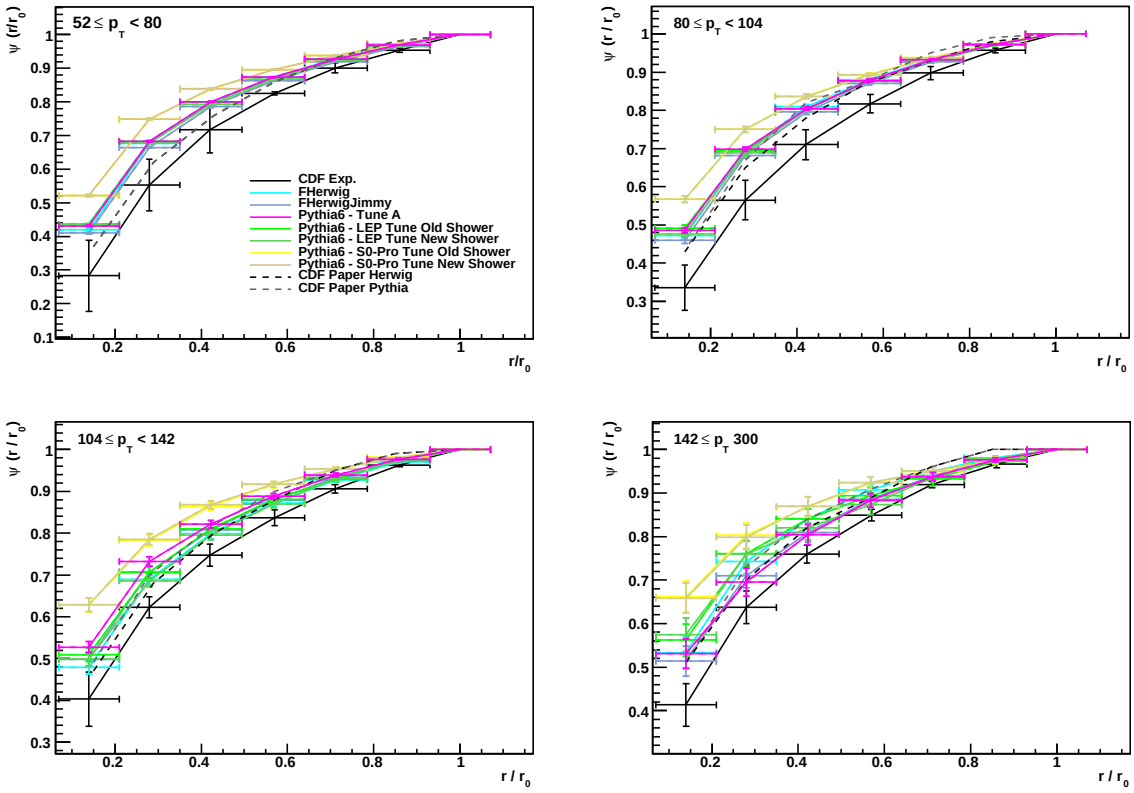

Figure 1: Integral jet shapes for the p_T ranges $52 \leqslant p_T < 80$ GeV/c, $80 \leqslant p_T < 104$ GeV/c, $104 \leqslant p_T < 142$ GeV/c, and $142 \leqslant p_T < 300$ GeV/c. The black plot is the CDF measured data from [4]. The dashed plots are the data from [4], however, they were gathered using an x/y grabber from that papers plots and are thus only included here as a guide to previous results. The coloured plots are MC generator results for the tunes and generators mentioned in §4. The dashed, coloured plots are data for when the generators were initialised to produce strictly b-quark containing events, they lie directly over — *i.e.* they are perfectly identical to — the results where the generators received no such initialisation. It can be seen that the MC generators more closely describe the measured data in the higher p_T regime.

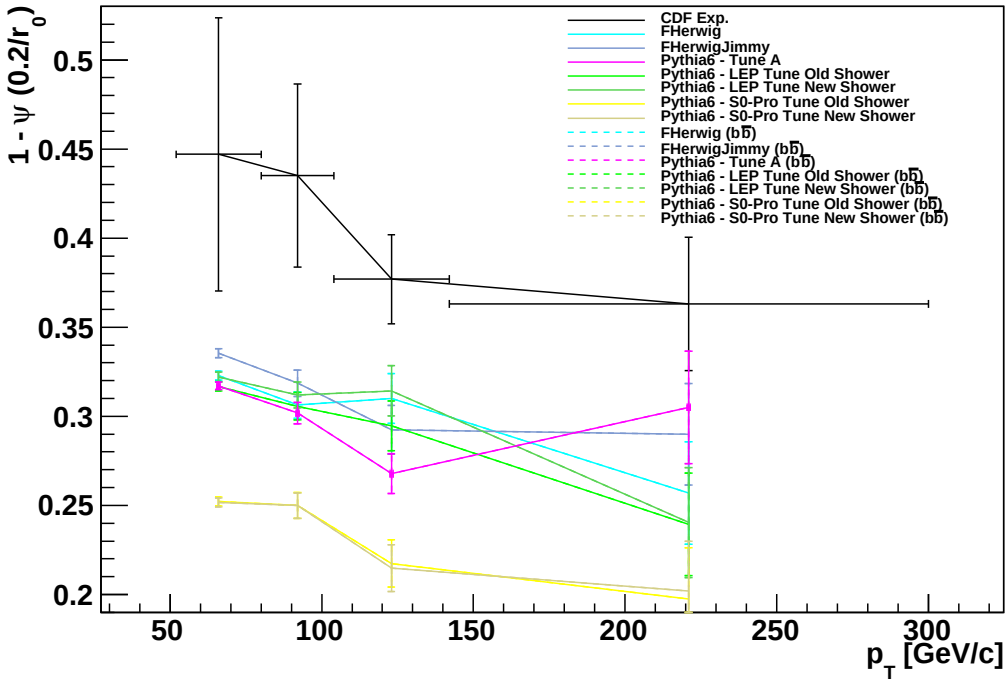

Figure 2: A plot of $1 - \Psi\left(\frac{0.2}{r_0}\right)$ against p_T. The black plot is the CDF measured data from [4]. The coloured plots are MC generator results for the tunes and generators mentioned in §4. The dashed, coloured plots are data for when the generators were initialised to produce strictly b-quark containing events, they lie directly over — *i.e.* they are perfectly identical to — the results where the generators received no such initialisation. It is seen that the jets get narrower as the p_T increases. The narrowness of the MC generated jets is more apparent in this plot.

	χ^2
HERWIG	26.55
HERWIG with JIMMY	15.29
PYTHIA6 Tune A	25.76
LEP Tune (New Shower)	19.49
LEP Tune (Old Shower)	24.99
S0-Pro (New Shower)	53.58
S0-Pro (Old Shower)	53.08

(a) Integral jet shape, $52 \leqslant p_T < 80$ GeV/c

	χ^2
HERWIG	8.77
HERWIG with JIMMY	6.64
PYTHIA6 Tune A	8.89
LEP Tune (New Shower)	8.18
LEP Tune (Old Shower)	8.29
S0-Pro (New Shower)	14.93
S0-Pro (Old Shower)	15.03

(b) Integral jet shape, $80 \leqslant p_T < 104$ GeV/c

	χ^2
HERWIG	4.75
HERWIG with JIMMY	7.22
PYTHIA6 Tune A	13.04
LEP Tune (New Shower)	5.76
LEP Tune (Old Shower)	7.30
S0-Pro (New Shower)	28.17
S0-Pro (Old Shower)	27.79

(c) Integral jet shape, $104 \leqslant p_T < 142$ GeV/c

	χ^2
HERWIG	9.04
HERWIG with JIMMY	4.11
PYTHIA6 Tune A	3.48
LEP Tune (New Shower)	5.02
LEP Tune (Old Shower)	5.99
S0-Pro (New Shower)	16.06
S0-Pro (Old Shower)	15.39

(d) Integral jet shape, $142 \leqslant p_T < 300$ GeV/c

	χ^2
HERWIG	4.83
HERWIG with JIMMY	4.59
PYTHIA6 Tune A	6.70
LEP Tune (New Shower)	4.86
LEP Tune (Old Shower)	6.06
S0-Pro (New Shower)	16.00
S0-Pro (Old Shower)	15.82

(e) $1 - \Psi\left(\frac{0.2}{r_0}\right)$ against p_T

	χ^2
HERWIG	10.79
HERWIG with JIMMY	7.57
PYTHIA6 Tune A	11.57
LEP Tune (New Shower)	8.66
LEP Tune (Old Shower)	10.53
S0-Pro (New Shower)	25.75
S0-Pro (Old Shower)	25.42

(f) Mean χ^2 values

Table 4: χ^2 goodness-of-fit values for the plots in Figs. 1 and 2. The values were calculated using the method described in §5. The MC plots are compared to the CDF measured jet shape data. It can be seen that HERWIG with the JIMMY add-on produces the best results. The S0-Pro tune for PYTHIA6 produces the worst result.

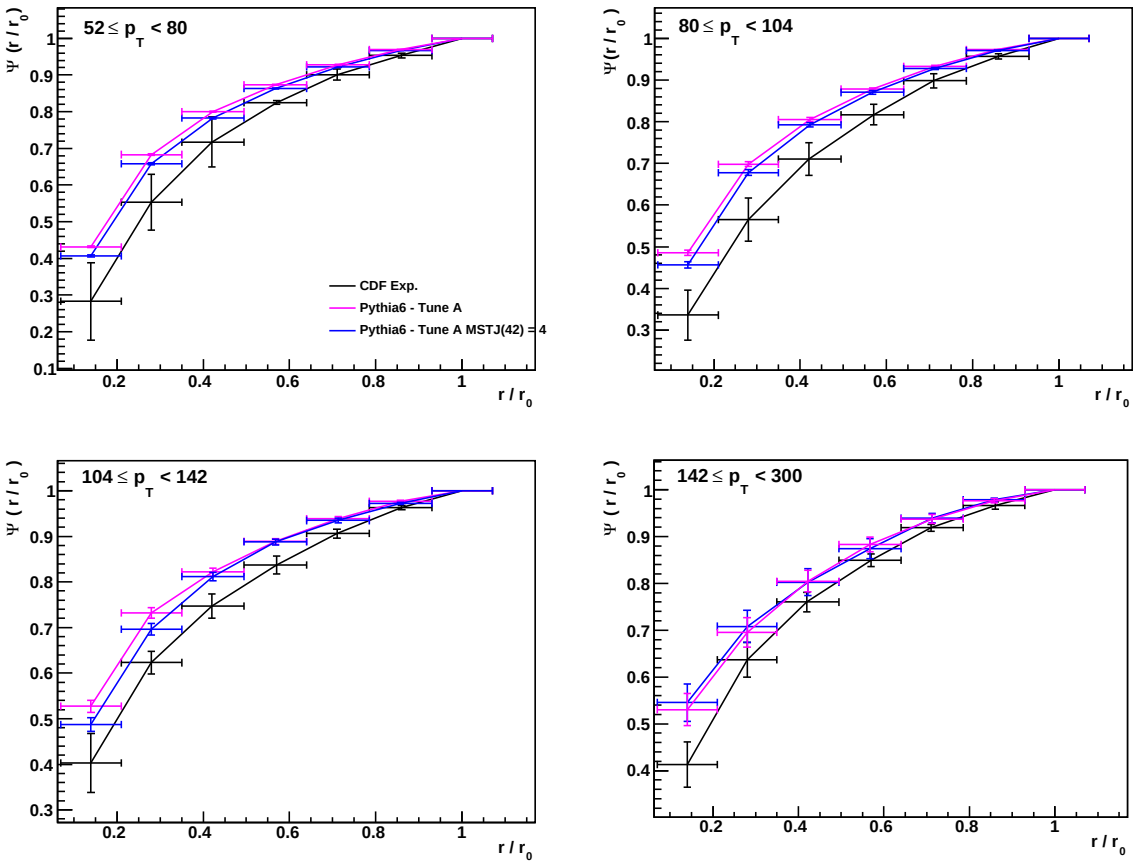

Figure 3: Integral jet shapes for the p_T ranges $52 \leqslant p_T < 80$ GeV/c, $80 \leqslant p_T < 104$ GeV/c, $104 \leqslant p_T < 142$ GeV/c, and $142 \leqslant p_T < 300$ GeV/c. The black plot is the CDF measured data from [4]. The blue plot is the result of using tune A and setting `MSTJ(42) = 4`. Setting `MSTJ(42) = 4` produces a better result, with the greatest improvement in the lower p_T regimes.

	χ^2
PYTHIA6 Tune A	25.76
Tune A `MSTJ(42) = 4`	15.24

(a) Integral jet shape, $52 \leqslant p_T < 80$ GeV/c

	χ^2
PYTHIA6 Tune A	8.89
Tune A `MSTJ(42) = 4`	5.74

(b) Integral jet shape, $80 \leqslant p_T < 104$ GeV/c

	χ^2
PYTHIA6 Tune A	13.04
Tune A `MSTJ(42) = 4`	9.89

(c) Integral jet shape, $104 \leqslant p_T < 142$ GeV/c

	χ^2
PYTHIA6 Tune A	3.48
Tune A `MSTJ(42) = 4`	2.85

(d) Integral jet shape, $142 \leqslant p_T < 300$ GeV/c

	χ^2
PYTHIA6 Tune A	6.70
Tune A `MSTJ(42) = 4`	4.41

(e) $1 - \Psi\left(\frac{0.2}{r_0}\right)$ against p_T

	χ^2
PYTHIA6 Tune A	11.57
Tune A `MSTJ(42) = 4`	7.63

(f) Mean χ^2 values

Table 5: χ^2 goodness-of-fit values for the plots in Figs. 3 and 4. The values were calculated using the method described in §5. The MC plots are compared to the CDF measured jet shape data. It is seen that setting the parameter `MSTJ(42) = 4` produces a better description of the data.

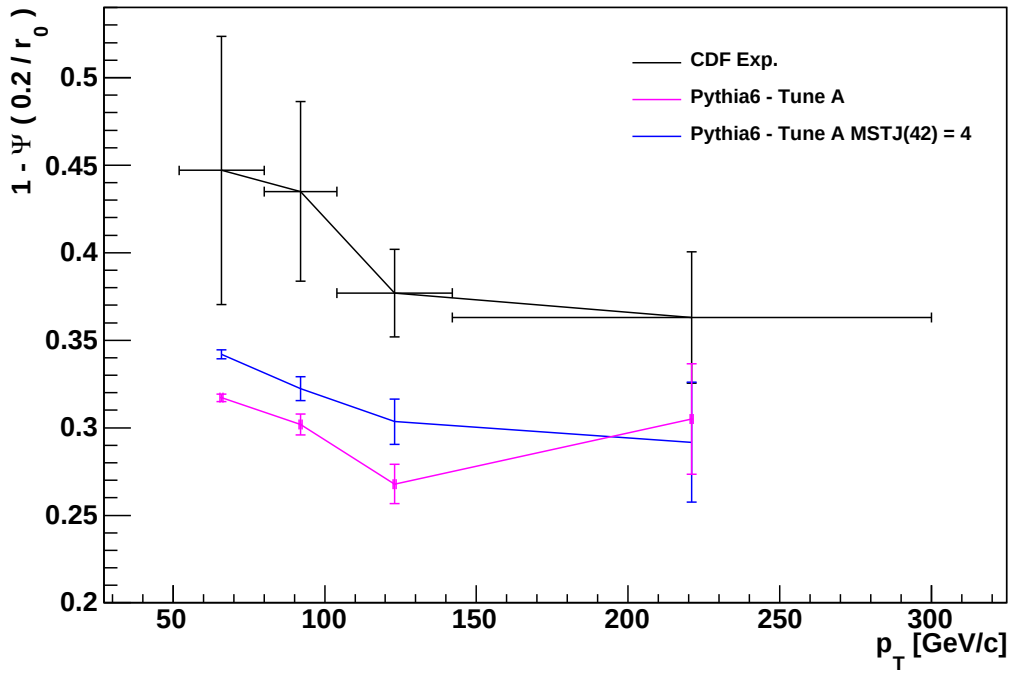

Figure 4: A plot of $1 - \Psi\left(\frac{0.2}{r_0}\right)$ against p_T. The black plot is the CDF measured data from [4]. The blue plot is the result of using tune A and setting `MSTJ(42) = 4`.

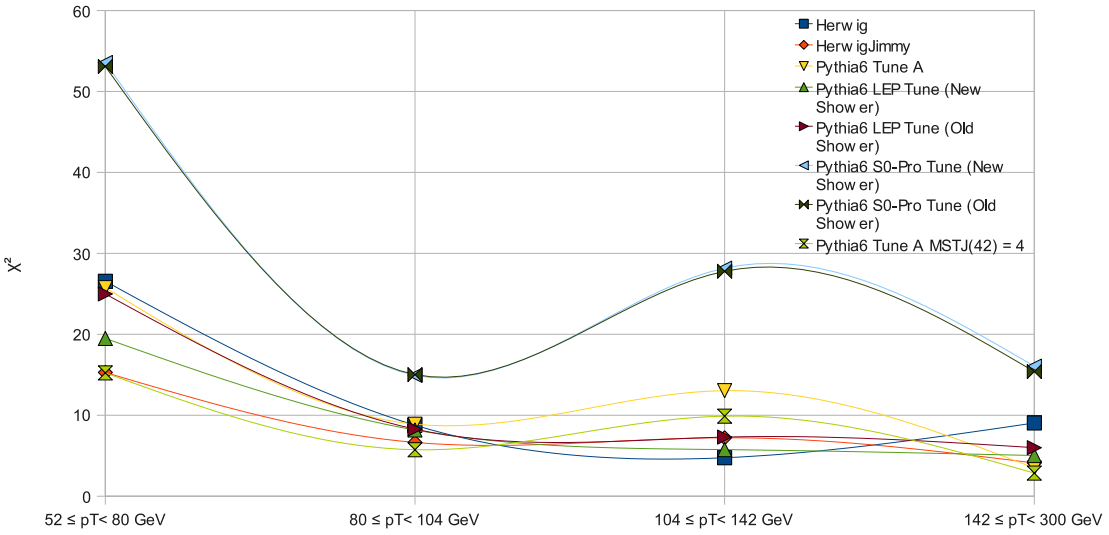

Figure 5: A plot of the χ^2 values, for each generator/tune, for each of the p_T binning ranges. It can be seen that the MC generators more closely describe the measured data in the higher p_T regime.

Appendix A: The analysis source code

Below is the source code for the `CDF_2008_S7782535` analysis performed in RIVET for this paper. [27, 28] The analysis was originally created for [4].

Listing 1: CDF_2008_S7782535.hh

```
// -*- C++ -*-
#ifndef RIVET_CDF_2008_S7782535_HH
#define RIVET_CDF_2008_S7782535_HH

#include "Rivet/Analysis.hh"
#include "Rivet/Projections/FastJets.hh"
#include "Rivet/Projections/PVertex.hh"
#include "Rivet/Projections/TotalVisibleMomentum.hh"
#include "Rivet/Projections/JetShape.hh"

namespace Rivet {


  /// Implementation of CDF RunII b-jet shape paper
  /// @todo Test with Pythia
  class CDF_2008_S7782535 : public Analysis {

  public:

    /// Constructor.

    /// jet cuts: |eta| <= 0.7

    /// Don't attempt to model the following cuts :
    ///   Missing ET significance
    ///   veto on additional vertices
    ///   Zvtx < 50

    CDF_2008_S7782535()
      : _Rjet(0.7) , _NpTbins(4)
    {
      setBeams(PROTON, ANTIPROTON);

      const FinalState fs(-3.6, 3.6);
      addProjection(fs, "FS");
      // Veto (anti)neutrinos, and muons with pT above 1.0 GeV
      VetoedFinalState vfs(fs);
      vfs
        .addVetoPairId(NU_E)
        .addVetoPairId(NU_MU)
        .addVetoPairId(NU_TAU)
        .addVetoDetail(MUON, 1.0*GeV, MAXDOUBLE);
      addProjection(vfs, "VFS");
      addProjection(FastJets(vfs, FastJets::CDFMIDPOINT, 0.7), "Jets");
      addProjection(JetShape(vfs, _jetaxes, 0.0, 0.7, 0.1, 0.3, ENERGY), "JetShape"
          );
    }

  public:

    /// Factory method
    static Analysis* create() {
      return new CDF_2008_S7782535();
    }


    /// @name Publication metadata
    //@{
```

```
58      /// Get SPIRES ID code.
59      string spiresId() const {
60        return "7782535";
61      }
62      /// Get a description of the analysis.
63      string description() const {
64        return "CDF Run II b-jet shape paper";
65      }
66      /// Experiment which performed and published this analysis.
67      string experiment() const {
68        return "CDF";
69      }
70      /// When published (preprint year according to SPIRES).
71      string year() const {
72        return "2008";
73      }
74      /// Journal, and preprint references.
75      virtual vector<string> references() const {
76        vector<string> ret;
77        ret.push_back("arXiv:0806.1699");
78        return ret;
79      }
80      //@}
81
82
83      /// @name Analysis methods
84      //@{
85      void init();
86      void analyze(const Event & event);
87      void finalize();
88      //@}
89
90
91    private:
92
93      /// @name Analysis cuts
94
95      /// @name Analysis data
96      //@{
97      vector<FourMomentum> _jetaxes;
98
99      double _Rjet;
100     vector<double> _pTbins;
101     int _NpTbins;
102     //@{
103     /// Histograms
104     AIDA::IProfile1D* _Psi_pT[4];
105     AIDA::IDataPointSet* _OneMinusPsi_vs_pT;
106     //@}
107
108
109   private:
110     /// Hide the assignment operator
111     CDF_2008_S7782535& operator=(const CDF_2008_S7782535&);
112
113   };
114
115 }
116
117 #endif
```

Listing 2: CDF_2008_S7782535.cc

```
1 // -*- C++ -*-
2 #include "Rivet/Tools/Logging.hh"
3 #include "Rivet/Analyses/CDF_2008_S7782535.hh"
4 #include "Rivet/RivetAIDA.hh"
```

```
5  #include "Rivet/Tools/ParticleIDMethods.hh"
6
7  namespace Rivet {
8
9
10   void CDF_2008_S7782535::init() {
11     _pTbins.push_back(52.);
12     _pTbins.push_back(80.);
13     _pTbins.push_back(104.);
14     _pTbins.push_back(142.);
15     _pTbins.push_back(300.);
16      // Book histograms
17     for (int i = 0; i < _NpTbins; ++i) {
18        stringstream title;
19        title << "Integral jet shape $\\Psi$ for $" << _pTbins[i] << " < p_\\perp <
              " << _pTbins[i+1] << "$";
20        _Psi_pT[i] = bookProfile1D(i+1, 2, 1, title.str());
21     }
22     _OneMinusPsi_vs_pT = bookDataPointSet(5, 1, 1, "$1 - \\Psi$ vs jet $p_\\perp$")
           ;
23   }
24
25
26
27   // Do the analysis
28   void CDF_2008_S7782535::analyze(const Event& event) {
29     // Put all b-quarks in a vector
30     ParticleVector bquarks;
31     /// @todo Provide nicer looping
32     for (GenEvent::particle_const_iterator p = event.genEvent().particles_begin();
33          p != event.genEvent().particles_end(); ++p) {
34       if ( fabs((*p)->pdg_id()) == BQUARK ) {
35         bquarks.push_back(Particle(**p));
36       }
37     }
38
39     if (bquarks.empty()) {
40       getLog() << Log::DEBUG << "No b-quarks, exiting" << endl;
41       vetoEvent(event);
42     }
43
44     // Get jets
45     const FastJets& jetpro = applyProjection<FastJets>(event, "Jets");
46     getLog() << Log::DEBUG << "Jet multiplicity before any pT cut = " << jetpro.
           size() << endl;
47
48     /// @todo Don't expose FastJet objects in Rivet analyses
49     const PseudoJets& jets = jetpro.pseudoJetsByPt();
50     getLog() << Log::DEBUG << "jetlist size = " << jets.size() << endl;
51     // Determine the central jet axes
52     FourMomentum jetaxis;
53     _jetaxes.clear();
54     for (PseudoJets::const_iterator jt = jets.begin(); jt != jets.end(); ++jt) {
55       // Only Central Calorimeter jets
56       /// @todo Declare this cut
57       //       cout<<jt->perp() << " " << jt->rapidity() << endl;
58       if (jt->perp() > _pTbins[0] && fabs(jt->rapidity()) <= 0.7) {
59         jetaxis.px(jt->px());
60         jetaxis.py(jt->py());
61         jetaxis.pz(jt->pz());
62         jetaxis.E(jt->E());
63         _jetaxes.push_back(jetaxis);
64       }
65     }
66     // Determine jet shapes
67     if (_jetaxes.empty())  {
68       getLog() << Log::DEBUG << "No jet axes" << endl;
```

```
69        vetoEvent(event);
70      }
71
72      const JetShape& jetShape = applyProjection<JetShape>(event, "JetShape");
73      for (size_t jind = 0; jind < _jetaxes.size(); ++jind) {
74
75        /// @todo Replace this with Jet::containsParticleId
76        bool bjet = false;
77        foreach (const Particle& bquark, bquarks) {
78          // double dr = deltaR(_jetaxes[jind].rapidity(), _jetaxes[jind].
              azimuthalAngle(),
79          //                    bquark->momentum().rapidity(), bquark->momentum().
              azimuthalAngle())
80          //if (dr <= _Rjet ) {
81          if (deltaR(_jetaxes[jind], bquark.momentum()) <= _Rjet ) {
82            bjet = true;
83            break;
84          }
85        }
86
87        if(bjet) {
88          // Put jet in correct pT bin
89          int jet_pt_bin = -1;
90          if      (_jetaxes[jind].pT() > _pTbins[0] && _jetaxes[jind].pT() <= _pTbins
                [1]) jet_pt_bin = 0;
91          else if (_jetaxes[jind].pT() > _pTbins[1] && _jetaxes[jind].pT() <= _pTbins
                [2]) jet_pt_bin = 1;
92          else if (_jetaxes[jind].pT() > _pTbins[2] && _jetaxes[jind].pT() <= _pTbins
                [3]) jet_pt_bin = 2;
93          else if (_jetaxes[jind].pT() > _pTbins[3] && _jetaxes[jind].pT() <= _pTbins
                [4]) jet_pt_bin = 3;
94          if (jet_pt_bin > -1) {
95            // Fill each entry in profile
96            for (size_t rbin = 0; rbin < jetShape.getNbins(); ++rbin) {
97              const double rad_Psi = jetShape.getRmin() +(rbin+1.0)*jetShape.
                    getInterval();
98              _Psi_pT[jet_pt_bin]->fill(rad_Psi/_Rjet, jetShape.getIntJetShape(jind,
                    rbin), event.weight() );
99            }
100         } // end valid jet_pt_bin
101       } // end bjet
102     } // end loop round jets
103
104   }
105
106
107
108   // Finalize
109   void CDF_2008_S7782535::finalize() {
110     std::vector<double> x, y, ex, ey;
111     for (unsigned int i = 0; i < _pTbins.size()-1; i++) {
112       x.push_back((_pTbins[i]+_pTbins[i+1])/2.);
113       ex.push_back(0.);
114       // get entry for  rad_Psi = 0.2 bin
115       y.push_back(1.0 - _Psi_pT[i]->binHeight(1));
116       ey.push_back(_Psi_pT[i]->binError(1));
117
118     }
119     _OneMinusPsi_vs_pT->setCoordinate(0,x,ex);
120     _OneMinusPsi_vs_pT->setCoordinate(1,y,ey);
121   }
122
123
124 }
```

Appendix B: χ^2 Python code

In order to calculate the χ^2 values for the `aida` histograms the following Python code was written. The `Histo` and `bin` classes come from the `aida2root` script in RIVET. [27, 28]

Listing 3: aidachisq.py

```
#! /usr/bin/env python


import sys, os
from array import array

try:
    sorted([])
except:
    def sorted(coll):
        coll.sort()
        return coll

class Histo:
    def __init__(self):
        self._bins = []
        self.path = None
        self.name = None
        self.title = None

    def __cmp__(self, other):
        """Sort by $path/$name string"""
        return self.fullPath() > other.fullPath()

    def __str__(self):
        out = "Histogram '%s' with %d bins\n" % (self.fullPath(), self.numBins())
        out += "Title: %s\n" % self.title
        out += "\n".join([str(b) for b in self.getBins()])
        return out

    def fullPath(self):
        return os.path.join(self.path, self.name)


    def numBins(self):
        return len(self._bins)

    def getBins(self):
        return sorted(self._bins)

    def setBins(self, bins):
        self._bins = bins
        return self

    def addBin(self, bin):
        self._bins.append(bin)
        return self

    def getBin(self, index):
        self._bins.sort()
        return self.getBins()[index]

    bins = property(getBins, setBins)

    def area(self):
        return sum([bin.area() for bin in self.bins])

    def __iter__(self):
```

```
59          return iter(self.getBins())
60
61      def __len__(self):
62          return len(self._bins)
63
64      def __getitem__(self, index):
65          return self.getBin(index)
66
67
68  class Bin:
69      """A simple container for a binned value with an error."""
70      def __init__(self, xval=0, xerrminus=None, xerrplus=None, yval=0, yerrminus=0,
            yerrplus=0, focus=None):
71          self.xval = xval
72          self.xerrplus = xerrplus
73          self.xerrminus= xerrminus
74          self.yval = yval
75          self.yerrplus = yerrplus
76          self.yerrminus = yerrminus
77          self.focus= focus
78
79      def __str__(self):
80          out = "%f to %f: %f +- %f" % (self.xval-self.xerrminus, self.xval+self.
                xerrplus, self._yval, self._yerr)
81          return out
82
83      def __cmp__(self, other):
84          """Sort by mean x value (yeah, I know...)"""
85          return (self.xval) > (other.xval)
86
87      def getXRange(self):
88          return (self.xval-self.xerrminus, self.xval+self.xerrplus)
89
90      def getXErrors(self):
91          return (self.xerrminus, self.xerrplus)
92
93      def setXRange(self, xlow, xhigh):
94          self.xlow = xlow
95          self.xhigh = xhigh
96          return self
97
98      def getYErrors(self):
99          return (self.yerrminus, self.yerrplus)
100
101     def getBinCenter(self):
102         """Geometric middle of the bin range."""
103         return self.xlow + .5*(self.xhigh - self.xlow)
104
105     def getFocus(self):
106         """Mean x-value of the bin."""
107         if self.focus is not None:
108             return (self.xlow + self.xhigh)/2.0
109         else:
110             return self.focus
111
112     def getXval(self):
113         return self.xval
114
115     def getYval(self):
116         return self.yval
117
118     def area(self):
119         return self.yval * (self.xerrplus - self.xerrminus)
120
121     def getYErr(self):
122         """Get mean of +ve and -ve y-errors."""
123         return (self.yerrplus + self.yerrminus)/2.0
```

```
124
125     def setYErr(self, yerr):
126         """Set both +ve and -ve y-errors simultaneously."""
127         self.yerrplus = yerr
128         self.yerrminus = yerr
129         return self
130
131
132
133 """ Try to load faster but non-standard cElementTree module """
134 try:
135     import xml.etree.cElementTree as ET
136 except ImportError:
137     try:
138         import cElementTree as ET
139     except ImportError:
140         import xml.etree.ElementTree as ET
141
142
143
144 def mkHistoFromDPS(dps):
145     """Make a mini histo representation from an AIDA dataPointSet tag."""
146     myhist = Histo()
147     myhist.name = dps.get("name")
148     myhist.title = dps.get("title")
149     myhist.path = dps.get("path")
150     points = dps.findall("dataPoint")
151     numbins = len(points)
152     for binnum, point in enumerate(points):
153         bin = Bin()
154         for d, m in enumerate(point.findall("measurement")):
155             val  = float(m.get("value"))
156             down = float(m.get("errorMinus"))
157             up = float(m.get("errorPlus"))
158             if d == 0:
159                 low  = val - down
160                 high = val + up
161                 bin.setXRange(low, high)
162                 bin.xval = val
163                 bin.xerrplus = up
164                 bin.xerrminus = down
165             elif d == 1:
166                 bin.yval = val
167                 bin.yerrplus = up
168                 bin.yerrminus = down
169         myhist.addBin(bin)
170     return myhist
171
172 def mkHistos(aidafile):
173     """ Processes all histos in a file and returns them in an array """
174     tree = ET.parse(aidafile)
175     histos = []
176     for dps in tree.findall("dataPointSet"):
177         useThisDps = True
178         if len(opts.PATHPATTERNS) > 0:
179             useThisDps = False
180             dpspath = os.path.join(dps.get("path"), dps.get("name"))
181             for regex in opts.PATHPATTERNS:
182                 if re.compile(regex).search(dpspath):
183                     useThisDps = True
184                     break
185         if useThisDps:
186             histos.append(mkHistoFromDPS(dps))
187     return histos
188
189 def getVals(histo):
190     """ Loop over each bin and add y values and y-errors to arrays """
```

```
191     y = []
192     sig = []
193     for bin in histo.getBins():
194         y.append(bin.getYval())
195         yerrneg, yerrpos = bin.getYErrors()
196         sig.append((yerrneg + yerrpos))
197     return ( y , sig )
198
199
200 from optparse import OptionParser
201 parser = OptionParser(usage="%prog aidafile(data) aidafile2(mc)")
202 parser.add_option("-s", "--smart-output", action="store_true", default=True,
203                   help="Write to output files with names based on the corresponding
                          input filename",
204                   dest="SMARTOUTPUT")
205 parser.add_option("-m", "--match", action="append",
206                   help="Only write out histograms whose $path/$name string matches
                          these regexes",
207                   dest="PATHPATTERNS")
208 parser.add_option("-d", "--debug", action="store_true", default=False,
209                   help="Debug mode",
210                   dest="DEBUG")
211 opts, args = parser.parse_args()
212 if opts.PATHPATTERNS is None:
213     opts.PATHPATTERNS = []
214
215 if len(args) <> 2:
216     sys.stderr.write("Must specify 2 (and only 2) AIDA histogram files\n")
217     sys.exit(1)
218
219 import re
220 out = sys.stdout
221 """ Make histo arrays for both inputs """
222 datahistos = mkHistos(args[0])
223 mchistos = mkHistos(args[1])
224
225 """ Simple check that we have the right files """
226 if len(mchistos) <> len(datahistos):
227     sys.stderr.write("There should be the same number of histograms in both files!\
          n")
228     sys.exit(1)
229 else:
230     i = 0
231     if opts.DEBUG:
232         print "Processing %i histograms in total." %( len(mchistos) )
233     """ Loop over all histos calculating chi^2 for each """
234     while i < len(mchistos):
235         if opts.DEBUG:
236             sys.stderr.write("Processing histogram %i.\n" %( i + 1 ))
237         """ Get y values and errors """
238         datay , datasig = getVals(datahistos[i])
239         mcy , mcsig = getVals(mchistos[i])
240         j = 0
241         chi2 = 0
242         if opts.DEBUG:
243             sys.stderr.write("Processing %i data points.\n" %( len(datay) ))
244         while j < len(datay):
245             """ Calculate chi squared from the data points """
246             if opts.DEBUG:
247                 try:
248                     sys.stderr.write("Data y \t=\t %f \t MC y \t=\t %f\n" %( datay[
                            j] , mcy[j] ))
249                     sys.stderr.write("Data \sigma_{y} \t=\t %f \t MC \sigma_{y} \t
                            =\t %f\n" %( datasig[j] , mcsig[j] ))
250                 except IndexError:
251                     sys.stderr.write("Index error occurred.\n")
252                     sys.stderr.write("No data point here.\n")
```

```
253             try:
254                 #lam = ((datay[j]/datasig[j])+(mcy[j]/mcsig[j]))/((1/datasig[j])
                        +(1/mcsig[j]))
255                 #chi2 = chi2 + ( ( ((datay[j] - lam )**2) / datasig[j] ) + ( ((mcy[
                        j] - lam )**2) / mcsig[j] ) )
256                 chi2 = chi2 + ( ((datay[j] - mcy[j])**2) )/( (datasig[j]**2) + (
                        mcsig[j]**2) )
257             except ZeroDivisionError:
258                 if opts.DEBUG:
259                     sys.stderr.write("Division by zero error. \n")
260                     sys.stderr.write("Continuing execution...\n")
261             except IndexError:
262                 if opts.DEBUG:
263                     sys.stderr.write("Index error occurred. \n")
264                     sys.stderr.write("No chi squared value will be obtained for
                            this histogram. \n")
265                     sys.stderr.write("Continuing execution...\n")
266                 chi2 = "Undefined"
267                 break
268             j = j + 1
269         print "Histo %i: \n chi^2 = %s" %( i + 1 , chi2)
270         i = i + 1
```

References

[1] N. Metropolis. The beginning of the Monte-Carlo methods. *Los Alamos Science*, pages 125–130, 1987. `http://library.lanl.gov/la-pubs/00326866.pdf`.

[2] Stephen D. Ellis, Zoltan Kunszt, and Davison E. Soper. Jets at hadron colliders at order $\alpha - s^{3:}$ A Look inside. *Phys. Rev. Lett.*, 69:3615–3618, 1992.

[3] Jonathan M. Butterworth, Adam R. Davison, Mathieu Rubin, and Gavin P. Salam. Jet substructure as a new Higgs search channel at the LHC. *Phys. Rev. Lett.*, 100:242001, 2008. arXiv:hep-ph/0802.2470.

[4] T. Aaltonen et al. Measurement of b-jet Shapes in Inclusive Jet Production in $p\bar{p}$ Collisions at $\sqrt{s} = 1.96$-TeV. *Phys. Rev.*, D78:072005, 2008. arXiv:hep-ex/0806.1699.

[5] Stefano Frixione, Michelangelo L. Mangano, Paolo Nason, and Giovanni Ridolfi. Heavy quark production. *Adv. Ser. Direct. High Energy Phys.*, 15:609–706, 1998.

[6] Torbjorn Sjostrand, Stephen Mrenna, and Peter Skands. PYTHIA 6.4 physics and manual. *JHEP*, 05:026, 2006. arXiv:hep-ph/0603175.

[7] Torbjorn Sjostrand et al. High-energy physics event generation with PYTHIA 6.1. *Comput. Phys. Commun.*, 135:238–259, 2001. arXiv:hep-ph/0010017.

[8] G. Corcella et al. HERWIG 6.5: an event generator for Hadron Emission Reactions With Interfering Gluons (including supersymmetric processes). *JHEP*, 01:010, 2001. arXiv:hep-ph/0011363.

[9] G. Corcella et al. HERWIG 6.5 release note. 2002. arXiv:hep-ph/0210213.

[10] Darin E. Acosta et al. Measurement of the J/ψ meson and $b-$hadron production cross sections in $p\bar{p}$ collisions at $\sqrt{s} = 1960$ GeV. *Phys. Rev.*, D71:032001, 2005.

[11] J. M. Butterworth, Jeffrey R. Forshaw, and M. H. Seymour. Multiparton interactions in photoproduction at HERA. *Z. Phys.*, C72:637–646, 1996. arXiv:hep-ph/9601371.

[12] Darin E. Acosta et al. Study of jet shapes in inclusive jet production in $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV. *Phys. Rev.*, D71:112002, 2005.

[13] M. Martisikova. Study of jet shapes in charm photoproduction at HERA. *AIP Conf. Proc.*, 792:819–822, 2005.

[14] Darin E. Acosta et al. Measurements of $b\bar{b}$ azimuthal production correlations in $p\bar{p}$ collisions at $\sqrt{s} = 1.8$ TeV. *Phys. Rev.*, D71:092001, 2005.

[15] B. Abbott et al. The $b\bar{b}$ production cross section and angular correlations in $p\bar{p}$ collisions at $\sqrt{s} = 1.8$ TeV. *Phys. Lett.*, B487:264–272, 2000.

[16] S. Vallecorsa. Proceedings of DIS2007. Atlantic Press.

[17] Gerald C. Blazey et al. Run II jet physics. 2000.

[18] A. Abulencia et al. Measurement of the inclusive jet cross section in $p\bar{p}$ interactions at $\sqrt{s} = 1.96$-TeV using a cone-based jet algorithm. *Phys. Rev.*, D74:071103, 2006.

[19] Darin E. Acosta et al. Measurement of the $t\bar{t}$ production cross section in $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV using lepton + jets events with secondary vertex $b-$tagging. *Phys. Rev.*, D71:052003, 2005.

[20] R. Field. FERMILAB-CONF-06-408-E. In *TeV4LHC Workshop*. FNAL, 2005.

[21] Professor (PROcedure For Estimating SyStematic errORs). Web:. `http://projects.hepforge.org/professor/`.

[22] Torbjorn Sjostrand. The Lund Monte Carlo for Jet Fragmentation and e+ e- Physics: Jetset Version 6.2. *Comput. Phys. Commun.*, 39:347–407, 1986.

[23] Bo Andersson, G. Gustafson, G. Ingelman, and T. Sjostrand. Parton Fragmentation and String Dynamics. *Phys. Rept.*, 97:31–145, 1983.

[24] B. R. Webber. A QCD Model for Jet Fragmentation Including Soft Gluon Interference. *Nucl. Phys.*, B238:492, 1984.

[25] C. Amsler et al. Review of particle physics. *Phys. Lett.*, B667:1, 2008.

[26] AGILe (A Generator Interface Library). Web:. `http://projects.hepforge.org/agile/`.

[27] Rivet (Robust Independent Validation of Experiment and Theory). Web:. `http://projects.hepforge.org/rivet/`.

[28] Andy Buckley. Tools for event generator tuning and validation. 2008. arXiv:hep-ph/0809.4638.

[29] AIDA (Abstract Interfaces for Data Analysis). Web:. `http://aida.freehep.org/`.

[30] ROOT - A Data Analysis Framework. Web:. `http://root.cern.ch/`.

[31] Matteo Cacciari and Gavin P. Salam. Dispelling the N^3 myth for the k_t jet-finder. *Phys. Lett.*, B641:57–61, 2006.

[32] S. Catani, Yuri L. Dokshitzer, M. H. Seymour, and B. R. Webber. Longitudinally invariant K_t clustering algorithms for hadron hadron collisions. *Nucl. Phys.*, B406:187–224, 1993.

[33] Stephen D. Ellis and Davison E. Soper. Successive combination jet algorithm for hadron collisions. *Phys. Rev.*, D48:3160–3166, 1993.

[34] Yuri L. Dokshitzer, G. D. Leder, S. Moretti, and B. R. Webber. Better Jet Clustering Algorithms. *JHEP*, 08:001, 1997.

[35] M. Wobisch. Measurement and QCD analysis of jet cross sections in deep-inelastic positron proton collisions at s**(1/2) = 300-GeV. DESY-THESIS-2000-049.

[36] Gavin P. Salam and Gregory Soyez. A Practical Seedless Infrared-Safe Cone jet algorithm. *JHEP*, 05:086, 2007.

List of Figures

List of Tables