

Rivet user manual

version 1.1.3

Andy Buckley

IPPP, Durham University, UK.

E-mail: andy.buckley@durham.ac.uk

Jonathan Butterworth

HEP Group, Dept. of Physics and Astronomy, UCL, London, UK.

E-mail: J.Butterworth@ucl.ac.uk

Leif Lönnblad

Theoretical Physics, Lund University, Sweden.

E-mail: lonnblad@thep.lu.se

Hendrik Hoeth

Theoretical Physics, Lund University, Sweden.

E-mail: hendrik.hoeth@cern.ch

James Monk

HEP Group, Dept. of Physics and Astronomy, UCL, London, UK.

E-mail: jmonk@hep.ucl.ac.uk

Frank Siegert

IPPP, Durham University, UK.

E-mail: frank.siegert@durham.ac.uk

Lars Sonnenschein

CERN, Genève 1206, Switzerland.

E-mail: sonne@cern.ch

ABSTRACT: This is the manual and user guide for the Rivet system for the validation and tuning of Monte Carlo event generators. As well as the core Rivet library, this manual describes the usage of the `rivet` program and the `AGILE` generator interface library. The depth and level of description is chosen for users of the system, starting with the basics of using validation code written by others, and then covering sufficient details to write new Rivet analyses and calculational components.

KEYWORDS: [Event generator](#), [simulation](#), [validation](#), [tuning](#), [QCD](#).

Contents

1. Introduction	3
1.1 Typographic conventions	4
I Getting started with Rivet	5
2. Quickstart	5
2.1 Getting generators for AGILe	7
2.2 Command completion	7
3. Running Rivet analyses	8
3.1 The FIFO idiom	8
3.2 Example <code>rivet</code> commands	8
4. Using analysis data	9
4.1 Histogram formats	9
4.2 Plotting and comparing data	10
II Standard Rivet analyses	11
5. LEP analyses	11
5.1 ALEPH_1991_S2435284	11
5.2 ALEPH_1996_S3486095	12
5.3 DELPHI_1995_S3137023	13
5.4 DELPHI_1996_S3430090	14
5.5 DELPHI_2002_069_CONF_603	15
5.6 DELPHI_2003_WUD_03_11	16
5.7 JADE_OPAL_2000_S4300807_133GEV	17
5.8 JADE_OPAL_2000_S4300807_161GEV	18
5.9 JADE_OPAL_2000_S4300807_172GEV	19
5.10 JADE_OPAL_2000_S4300807_183GEV	20
5.11 JADE_OPAL_2000_S4300807_189GEV	21
5.12 JADE_OPAL_2000_S4300807_91GEV	22
5.13 OPAL_1998_S3780481	23
6. Tevatron analyses	24
6.1 CDF_1990_S2089246	24
6.2 CDF_1994_S2952106	25
6.3 CDF_2000_S4155203	26
6.4 CDF_2001_S4751469	27

6.5	CDF_2002_S4796047	28
6.6	CDF_2004_S5839831	29
6.7	CDF_2005_S6217184	30
6.8	CDF_2006_S6653332	31
6.9	CDF_2007_S7057202	32
6.10	CDF_2008_LEADINGJETS	33
6.11	CDF_2008_NOTE_9351	34
6.12	CDF_2008_S7540469	35
6.13	CDF_2008_S7541902	36
6.14	CDF_2008_S7782535	37
6.15	CDF_2008_S7828950	38
6.16	CDF_2008_S8095620	39
6.17	CDF_2009_S8233977	40
6.18	D0_2001_S4674421	41
6.19	D0_2004_S5992206	42
6.20	D0_2006_S6438750	43
6.21	D0_2007_S7075677	44
6.22	D0_2008_S6879055	45
6.23	D0_2008_S7554427	46
6.24	D0_2008_S7662670	47
6.25	D0_2008_S7719523	48
6.26	D0_2008_S7837160	49
6.27	D0_2008_S7863608	50
6.28	D0_2009_S8202443	51
7.	HERA analyses	52
7.1	H1_1994_S2919893	52
7.2	H1_1995_S3167097	53
7.3	H1_2000_S4129130	54
7.4	ZEUS_2001_S4815815	55
8.	Monte Carlo analyses	56
8.1	MC_LHC_LEADINGJETS	56
8.2	MC_TVT1960_ZJETS	57
9.	Example analyses	58
9.1	EXAMPLE	58
9.2	EXAMPLETREE	59
10.	Misc. analyses	60
10.1	JADE_OPAL_2000_S4300807_35GEV	60
10.2	JADE_OPAL_2000_S4300807_44GEV	61
10.3	PDG_HADRON_MULTIPLICITIES	62
10.4	PDG_HADRON_MULTIPLICITIES_RATIOS	63

10.5	STAR_2006_S6870392	64
10.6	STAR_2008_S7993412	65
III	How Rivet works	66
11.	Projections	66
11.1	Projection caching	66
11.2	Using projection caching	67
12.	Analyses	68
12.1	Writing a new analysis	68
12.1.1	Analysis constructor	69
12.2	Histogramming	70
12.3	Pluggable analyses	71
IV	How Rivet <i>really</i> works	72
13.	Projection caching	72
13.1	Writing a Projection comparison operator	72
V	Appendices	73
A.	Typical <code>agile-runmc</code> commands	73
VI	Bibliography	74

1. Introduction

This manual is a users' guide to using the Rivet generator validation system. Rivet is a C++ class library, which provides the infrastructure and calculational tools for simulation-level analyses, enabling physicists to validate event generator models and tunings with minimal effort and maximum portability. Rivet is designed to scale effectively to large numbers of analyses for truly global validation, by transparent use of an automated result caching system.

The Rivet ethos, if it may be expressed succinctly, is that user analysis code should be extremely clean and easy to write — ideally it should be sufficiently self-explanatory to in itself be a reference to the experimental analysis algorithm — without sacrificing power or extensibility. The machinery to make this possible is intentionally hidden from the view of

all but the most prying users. Generator independence is explicitly required by virtue of all analyses operating on the generic “HepMC” event record.

The simplest way to use Rivet is via the `rivet` command line tool, which analyses textual HepMC event records as they are generated and produces output distributions in a structured textual format. The input events are generated using the generator’s own steering program, if one is provided; for generators which provide no default way to produce HepMC output, the AGILE generator interface library, and in particular the `agile-runmc` command which it provides, may be useful. For those who wish to embed their analyses in some larger framework, Rivet can also be run programmatically on HepMC event objects with no special executable being required.

Before we get started, a declaration of intent: this manual is intended to be a guide to using Rivet, rather than a comprehensive and painstakingly maintained reference to the application programming interface (API) of the Rivet library. For that purpose, you will hopefully find the online generated documentation at <http://projects.hepforge.org/rivet> to be sufficient. Similar API documentation is maintained for AGILE at <http://projects.hepforge.org/agile>.

1.1 Typographic conventions

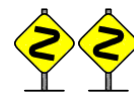
As is normal in computer user manuals, the typography in this manual is used to indicate whether we are describing source code elements, commands to be run in a terminal, the output of a command etc.

The main such clue will be the use of **typewriter-style** text: this indicates the name of a command or code element — class names, function names etc. Typewriter font is also used for commands to be run in a terminal, but in this case it will be prefixed by a dollar sign, as in `$ echo 'Hello' | cat`. The output of such a command on the terminal will be typeset in **sans-serif** font. When we are documenting a code feature in detail (which is not the main point of this manual), we will use square brackets to indicate optional arguments, and italic font between angle brackets to represent an argument name which should be replaced by a value, e.g. `Event::applyProjection(<proj>)`.

Following the example of Donald Knuth in his books on T_EX, in this document we will indicate paragraphs of particular technicality or esoteric nature with a “dangerous bend” sign. These will typically describe internals of Rivet of which most people will be fortunate enough to remain happily ignorant without adverse effects. However they may be of interest to detail obsessives, the inordinately curious and Rivet hackers. You can certainly skip them on a first reading. Similarly, you may see double bend signs — the same rules apply for these, but even more strongly.



Dangerous bend



Double bend

Part I

Getting started with Rivet

As with many things, Rivet may be meaningfully approached at several distinct levels of detail:

- The simplest, and we hope the most common, is to use the analyses which are already in the library to study events from a variety of generators and tunes: this is enormously valuable in itself and we encourage all manner of experimentalists and phenomenologists alike to use Rivet in this mode.
- A more involved level of usage is to write your own Rivet analyses — this may be done without affecting the installed standard analyses by use of a “plugin” system (although we encourage users who develop analyses to submit them to the Rivet developers for inclusion into a future release of the main package). This approach requires some understanding of programming within Rivet but you don’t *need* to know about exactly what the system is doing with the objects that you have defined.
- Finally, Rivet developers and people who want to do non-standard things with their analyses will need to know something about the messy details of what Rivet’s infrastructure is doing behind the scenes. But you’d probably rather be doing some physics!

The current part of this manual is for the first sort of user, who wants to get on with studying some observables with a generator or tune, or comparing several such models. Since everyone will fall into this category at some point, our preent interest is to get you to that all-important “physics plots” stage as quickly as possible. Analysis authors and Rivet service-mechanics will find the more detailed information that they crave in Part [III](#).

2. Quickstart

The point of this section is to get you up and running with Rivet as soon as possible. Doing this by hand may be rather frustrating, as Rivet depends on several external libraries — you’ll get bored downloading and building them by hand in the right order. Here we recommend two much simpler ways — for the full details of how to build Rivet by hand, please consult the Rivet Web page.

Ubuntu/Debian package archive A selection of HEP packages, including Rivet, are maintained as Debian/Ubuntu Linux packages on the Launchpad PPA system: <https://launchpad.net/~hep/+archive>. This is the nicest option for Debian/Ubuntu, since not only will it work more easily than anything else, but you will also automatically benefit from bug fixes and version upgrades as they appear.

The PPA packages have been built as binaries for a variety of architectures, and the package interdependencies are automatically known and used: all you need to do on a

Debian-type Linux system (Ubuntu included) is to add the Launchpad archive address to your APT sources list and then request installation of the `rivet` package in the usual way. See the Launchpad and system documentation for all the details.

Bootstrap script For those not using Debian/Ubuntu systems, we have written a bootstrapping script which will download tarballs of Rivet, AGILE and the other required libraries, expand them and build them in the right order with the correct build flags. This is generally nicer than doing it all by hand, and virtually essential if you want to use the existing versions of FastJet, HepMC, generator libraries, and so on from CERN AFS: there are issues with these versions which the script works around, which you won't find easy to do yourself.

You can get the bootstrap script from the following Web address: <http://svn.hepforge.org/rivet/bootstrap/rivet-bootstrap>

To run the script, we recommend that you choose a personal installation directory. Personally, I make a `~/local` directory for this purpose, to avoid polluting my home directory with a lot of files. If you already use a directory of the same name, you might want to use a separate one, say `~/rivetlocal`, such that if you need to delete everything in the installation area you can do so without difficulties. You'll need to add `<localdir>/bin` to your `$PATH` environment variable and `<localdir>/lib` to your `$LD_LIBRARY_PATH`.

Now, change directory to your build area (you may also want to make this, e.g. `~/build`), and download the script:

```
$ wget http://svn.hepforge.org/rivet/bootstrap/rivet-bootstrap
```

Now run it, specifying the install area as the argument:

```
$ chmod +x rivet-bootstrap
```

```
$ ./rivet-bootstrap <localdir>
```

If you are running on a system where the CERN AFS area is mounted as `/afs/cern.ch`, then the bootstrap script will attempt to use the pre-built HepMC, LHAPDF, FastJet and GSL libraries from the LCG software area. Either way, you'll see a large amount of build output, and finally a message telling you what changes to your environment variables will make the system useable.

You now have a working, installed copy of the Rivet and AGILE libraries, and the `rivet` and `agile-runmc` executables: respectively these are the command-line frontend to the Rivet analysis library, and a convenient steering command for generators which do not provide their own main program with HepMC output. To test that they work as expected, set the environment variables as instructed, if you've not already done so, run this:

```
$ rivet --help
```

This should print a quick-reference user guide for the `rivet` command to the terminal. Similarly, for `agile-runmc`,

```
$ agile-runmc --help
```

```
$ agile-runmc --list-gens
```

```
$ agile-runmc --beams=pp:14TeV FPythia:6413
```

which should respectively print the help, list the available generators and make 10 LHC-type

events using the Fortran Pythia 6.4.13 generator. You're on your way! If no generators are listed, you probably need to install a local Genser-type generator repository: see section 2.1.

In this manual, because of its convenience, we will use `agile-runmc` as our canonical way of producing a stream of HepMC event data; if your interest is in running a generator like Sherpa or Herwig++ which provides its own native way to make HepMC output, or a generator like Cascade or PHOJET which is not currently supported by AGILE, then substitute the appropriate command in what follows. We'll discuss using these commands in detail in section 3.

2.1 Getting generators for AGILE

One last thing before continuing, though: the generators themselves. Again, if you're running on a system with the CERN LCG AFS area mounted, then `rivetgun` will attempt to automatically use the generators packaged by the LCG Genser team.

Otherwise, you'll have to build your own mirror of the LCG generators. This process is not standardised at the moment (this will hopefully change), so we've provided a script, `agile-genser-bootstrap`:

```
$ wget http://svn.hepforge.org/agile/genser/agile-genser-bootstrap
```

Now make yourself a Genser installation directory, e.g. `$HOME/genser`, and `cd` into it. Then run the `agile-genser-bootstrap` script, and wait for it all to build. Finally, set the `$AGILE_GEN_PATH` path variable to contain the `<genserDir>` directory: you should now have a few generators to play with.

If you are interested in using a generator not currently supported by AGILE, which does not output HepMC events in its native state, then please contact the authors and hopefully we can help.

2.2 Command completion

A final installation point worth considering is using the supplied bash-shell programmable completion setup for the `rivet` and `agile-runmc` commands. Despite being cosmetic and semi-trivial, programmable completion makes using `rivet` positively pleasant, especially since you no longer need to remember the somewhat cryptic analysis names¹!

To use programmable completion, source the appropriate files from the install location:

```
$ . <localdir>/share/Rivet/rivet-completion
```

```
$ . <localdir>/share/AGILE/agile-completion
```

If there is already a `<localdir>/etc/bash_completion.d` directory in your install path, Rivet and AGILE's installation scripts will install extra copies into that location, since automatically sourcing all completion files in such a path is quite standard.

Apologies to `{C,k,z,...}`-shell users, but this feature is currently only available for the `bash` shell. Anyone who feels like supplying fixes or additions for their favourite shell is very welcome to get in touch with the developers.

¹Standard Rivet analyses have names which, as well as the publication date and experiment name, incorporate the 8-digit Spires ID code.

3. Running Rivet analyses

The `rivet` executable is the easiest way to use Rivet, and will be our example throughout this manual. This command reads HepMC events in the standard ASCII format, either from file or from a text stream.

3.1 The FIFO idiom

Since you rarely want to store simulated HepMC events and they are computationally cheap to produce (at least when compared to the remainder of experiment simulation chains), we recommend using a Unix *named pipe* (or “FIFO” — first-in, first-out) to stream the events. While this may seem unusual at first, it is just a nice way of “pretending” that we are writing to and reading from a file, without actually involving any slow disk access or building of huge files: a 1M event LHC run would occupy $\sim 60GB$ on disk, and typically it takes twice as long to make and analyse the events when the filesystem is involved! Here is an example:

```
$ mkfifo fifo.hepmc
$ agile-runmc Pythia:6418 -o fifo.hepmc &
$ rivet -a EXAMPLE fifo.hepmc
```

Note that the generator process (`agile-runmc` in this case) is *backgrounded* before `rivet` is run. This is absolutely necessary, since the buffer size of a pipe in Linux is only 64K — about the space required to store one LHC event in HepMC’s textual event format. The generator process will have to wait until the buffer is cleared, e.g. by being read by `rivet`, before computing or writing any more events. By running the generator and `rivet` at the same time, this flow control through the buffer is invisible to the user.

Notably, `mkfifo` will not work if applied to a directory mounted via the AFS distributed filesystem, as widely used in HEP. This is not a big problem: just make your FIFO object somewhere not mounted via AFS, e.g. `/tmp`. There is no performance penalty, as the filesystem object is not written to during the streaming process.

In the following command examples, we will assume that a generator has been set up to write to the `fifo.hepmc` FIFO, and just list the `rivet` command that reads from that location. Some typical `agile-runmc` commands are listed in [appendix A](#).

3.2 Example `rivet` commands

- **Getting help:** `rivet --help` will print a (hopefully) helpful list of options which may be used with the `rivet` command, as well as other information such as environment variables which may affect the run.
- **Choosing analyses:** `rivet --list-analyses` will list the available analyses, including both those in the Rivet distribution and any plugins which are found at runtime. `rivet --show-analysis < patt >` will show a lot of details about any analyses whose name match the `< patt >` regular expression pattern — simple bits of analysis name are a perfectly valid subset of this. For example, `rivet --show-analysis`

CDF_200 exploits the standard Rivet analysis naming scheme to show details of all available CDF experiment analyses published in the “noughties.”

- **Running particular analyses:** `rivet -a DELPHI_1996_S3430090 in.hepmc` will run the Rivet `DELPHI_1996_S3430090` [1] analysis on the events in the `in.hepmc` data file. This analysis is the one originally used for the DELPHI automated “PROFESSOR” generator tuning. If the first event in the data file does not have appropriate beams, the analysis will be disabled; since there is only one analysis in this case, the command will exit immediately with a warning.
- **Using all analyses:** `rivet -n 50000 -A -` will read up to 50k events from standard input (specified by the special “-” input filename) and analyse them with *all* the Rivet library analyses. As above, incompatible analyses (based on beam particle IDs), will be removed before the main analysis run begins.
- **Histogramming:** `rivet in.hepmc -H foo` will read all the events in the `in.hepmc` file. The `-H` switch is used to specify that the output histogram file will be named `foo.aida`. By default the output file is called `Rivet.aida`.
- **Fine-grained logging:** `rivet in.hepmc -A -l Rivet.Analysis=DEBUG \`
`-l Rivet.Projection=DEBUG -l Rivet.Projection.FinalState=TRACE \`
`-l RivetGun=WARN -l NEvt=WARN` analyse events as before, but will print different status information as the run progresses. Hierarchical logging control is possible down to the level of individual analyses and projections as shown above; this is useful for debugging without getting overloaded with debug information from *all* the components at once. The default level is “INFO”, which lies between “DEBUG” and “WARNING”; the “TRACE” level is for very low level information, and probably isn’t needed by normal users.

4. Using analysis data

In this section, we summarise how to use the data files which Rivet produces for plotting, validation and tuning.

4.1 Histogram formats

Rivet currently produces output histogram data in the AIDA XML format. Most people aren’t familiar with AIDA (and we recommend that you remain that way!), and it will disappear entirely from Rivet in version 1.2.0. You will probably wish to cast the AIDA files to a different format for plotting, and for this we supply several scripts.

Conversion to ROOT Your knee-jerk reaction is probably to want to know how to plot your Rivet histograms in ROOT. Don’t worry; you can recover from this unfortunate behaviour after only a few months of therapy. For unrepentant ROOT junkies, Rivet installs an `aida2root` script, which converts the AIDA records to a `.root` file full of ROOT `TGraph`

s. One word of warning: a bug in ROOT means that `TGraph`s do not render properly from file because the axis is not drawn by default. To display the plots correctly in ROOT you will need to pass the "AP" drawing option string to either the `TGraph::Draw()` method, or in the options box in the `TBrowser` GUI interface.

Conversion to “flat format” Most of our histogramming is based around the YODA “flat” plain text format, which can easily be read (and written) by hand. We provide a script called `aida2flat` to do this conversion. Run `aida2flat -h` to get usage instructions; in particular the `Gnuplot` and “split output” options are useful for further visualisation. Aside from anything else, this is useful for simply checking the contents of an AIDA file, with `aida2flat Rivet.aida | less`.



We get asked a lot about why we don’t use ROOT internally: aside from a general unhappiness about the design and quality of the data objects in ROOT, the monolithic nature of the system makes it a big dependency for a system as small as Rivet. While not an issue for experimentalists, most theorists and generator developers do not use ROOT and we preferred to embed the AIDA system, which in its LWH implementation requires no external package. The replacement for AIDA will be another lightweight system rather than ROOT, with an emphasis on friendly, intuitive data object design, and correct handling of sample merging statistics for all data objects.

4.2 Plotting and comparing data

Rivet comes with two commands — `compare-histos` and `make-plots` — for comparing and plotting data files. These commands produce nice comparison plots of publication quality from the YODA format text files, e.g.:

```
$ compare-plots path/to/CDF_2001_S4751469.aida py.aida:'Pythia 6.418' \
hw.aida:'Herwig++ 2.3.0'
```

This command will have compared the three named data files (ending in `.aida`), identified which plots are available in them, and combined the MC and reference plots appropriately into a set of plot data files ending with `.dat`. The strings after the “:” for the MC files are specifying ID strings to appear in the plot legends. You can also run `compare-plots` to just compare MC–MC data files. More options are described by running `compare-histos --help`.

Incidentally, the reference files for each Rivet analysis are to be found in the installed Rivet shared data directory, `<installdir>/share/Rivet`. You can find the location of this by using the `rivet-config` command:

```
$ rivet-config --datadir
```

You can now plot the created data files using the `make-plots` command:

```
$ make-plots --pdf *.dat
```

The `--pdf` flag makes the output plots in PDF format: by default the output is in PostScript (`.ps`), and flags for conversion to EPS and PNG are also available.

Part II

Standard Rivet analyses

In this section we describe the standard experimental analyses included with the Rivet library. To maintain synchronisation with the code, these descriptions are generated automatically from the metadata in the analysis objects themselves. This is currently rather sparse, hence the briefness of the descriptions shown here. Richer metadata will be added to the code soon!

5. LEP analyses

5.1 ALEPH_1991_S2435284

Hadronic Z decay charged multiplicity measurement

Experiment: ALEPH (LEP 1)

Spires ID: [2435284](#)

Status: VALIDATED

Authors:

- Andy Buckley (andy.buckley@durham.ac.uk);

References:

- Phys. Lett. B, 273, 181 (1991)

Run details:

- Hadronic Z decay events generated on the Z pole ($\sqrt{s} = 91.2$ GeV)

The charged particle multiplicity distribution of hadronic Z decays, as measured on the peak of the Z resonance using the ALEPH detector at LEP. The unfolding procedure was model independent, and the distribution was found to have a mean of 20.85 ± 0.24 . Comparison with lower energy data supports the KNO scaling hypothesis. The shape of the multiplicity distribution is well described by a log-normal distribution, as predicted from a cascading model for multi-particle production.

5.2 ALEPH_1996_S3486095

Studies of QCD with the ALEPH detector.

Experiment: ALEPH (LEP 1)

Spires ID: [3486095](#)

Status: VALIDATED

Authors:

- Holger Schulz (holger.schulz@physik.hu-berlin.de);

References:

- Phys. Rept., 294, 1–165 (1998)

Run details:

- Hadronic Z decay events generated on the Z pole ($\sqrt{s} = 91.2$ GeV)

Summary paper of QCD results as measured by ALEPH at LEP 1. The publication includes various event shape variables, multiplicities (identified particles and inclusive), and particle spectra.

5.3 DELPHI_1995_S3137023

Strange baryon production in Z hadronic decays at Delphi

Experiment: DELPHI (LEP 1)

Spires ID: [3137023](#)

Status: VALIDATED

Authors:

- Hendrik Hoeth (hendrik.hoeth@cern.ch);

References:

- Z. Phys. C, 67, 543–554 (1995)

Run details:

- Hadronic Z decay events generated on the Z pole ($\sqrt{s} = 91.2$ GeV)

Measurement of the Ξ^- and $\Sigma^+(1385)/\Sigma^-(1385)$ scaled momentum distributions by DELPHI at LEP 1. The paper also has the production cross-sections of these particles, but that's not implemented in Rivet.

5.4 DELPHI.1996.S3430090

Delphi MC tuning on event shapes and identified particles.

Experiment: DELPHI (LEP 1)

Spires ID: [3430090](#)

Status: UNVALIDATED

Authors:

- Andy Buckley [<andy.buckley@durham.ac.uk>](mailto:andy.buckley@durham.ac.uk);
- Hendrik Hoeth [<hendrik.hoeth@cern.ch>](mailto:hendrik.hoeth@cern.ch);

References:

- Z.Phys.C73:11-60,1996
- DOI: [10.1007/s002880050295](https://doi.org/10.1007/s002880050295)

Run details:

- Energy: 91.2 GeV
- Event type is $e^+ e^- Z$ production with hadronic decays only

Event shape and charged particle inclusive distributions measured using 750000 decays of Z bosons to hadrons from the DELPHI detector at LEP. This data, combined with identified particle distributions from all LEP experiments, was used for tuning of shower-hadronisation event generators by the original PROFESSOR method.

This is a critical analysis for MC event generator tuning of final state radiation and both flavour and kinematic aspects of hadronisation models.

5.5 DELPHI.2002.069.CONF.603

Study of the b-quark fragmentation function at LEP 1

Experiment: DELPHI (LEP 1)

Spires ID: [NONE](#)

Status: VALIDATED

Authors:

- Hendrik Hoeth (hendrik.hoeth@cern.ch);

References:

- DELPHI note 2002-069-CONF-603 (ICHEP 2002)

Run details:

- Hadronic Z decay events generated on the Z pole ($\sqrt{s} = 91.2$ GeV)

Measurement of the b-quark fragmentation function by DELPHI using 1994 LEP 1 data. The fragmentation function for both weakly decaying and primary b-quarks has been determined in a model independent way. Nevertheless the authors trust $f(x_{B\text{weak}})$ more than $f(x_{B\text{prim}})$.

5.6 DELPHI.2003_WUD.03.11

4-jet angular distributions at LEP

Experiment: DELPHI (LEP 1)

Spires ID: [NONE](#)

Status: UNVALIDATED

Authors:

- Hendrik Hoeth (hendrik.hoeth@cern.ch);

References:

- Diploma thesis WUD-03-11, University of Wuppertal

Run details:

- Hadronic Z decay events generated on the Z pole ($\sqrt{s} = 91.2$ GeV)

The 4-jet angular distributions (Bengtsson-Zerwas, Körner-Schierholz-Willrodt, Nachtmann-Reiter, and α_{34}) have been measured with DELPHI at LEP 1 using Jade and Durham cluster algorithms.

5.7 JADE_OPAL_2000_S4300807_133GEV

Jet rates in e+e- at OPAL [133 GeV].

Experiment: JADE_OPAL (LEP Run 2)

Spires ID: [4300807](#)

Status: VALIDATED

Authors:

- Frank Siegert (frank.siegert@durham.ac.uk);

References:

- Eur.Phys.J.C17:19-51,2000
- arXiv: [hep-ex/0001055](#)

Run details:

- e+ e- collisions:
- e+ e- \rightarrow jet jet (+ jets) at 133 GeV. * no cuts needed

Differential and integrated jet rates for Durham and JADE jet algorithms at $\sqrt{s} = 133$.

5.8 JADE_OPAL_2000_S4300807_161GEV

Jet rates in e+e- at OPAL [161 GeV].

Experiment: JADE_OPAL (LEP Run 2)

Spires ID: [4300807](#)

Status: VALIDATED

Authors:

- Frank Siegert (frank.siegert@durham.ac.uk);

References:

- Eur.Phys.J.C17:19-51,2000
- arXiv: [hep-ex/0001055](#)

Run details:

- e+ e- collisions:
- e+ e- \rightarrow jet jet (+ jets) at 161 GeV. * no cuts needed

Differential and integrated jet rates for Durham and JADE jet algorithms at $\sqrt{s} = 161$.

5.9 JADE_OPAL_2000_S4300807_172GEV

Jet rates in e+e- at OPAL [172 GeV].

Experiment: JADE_OPAL (LEP Run 2)

Spires ID: [4300807](#)

Status: VALIDATED

Authors:

- Frank Siegert (frank.siegert@durham.ac.uk);

References:

- Eur.Phys.J.C17:19-51,2000
- arXiv: [hep-ex/0001055](#)

Run details:

- e+ e- collisions:
- e+ e- \rightarrow jet jet (+ jets) at 172 GeV. * no cuts needed

Differential and integrated jet rates for Durham and JADE jet algorithms at $\sqrt{s} = 172$.

5.10 JADE_OPAL_2000_S4300807_183GEV

Jet rates in e+e- at OPAL [183 GeV].

Experiment: JADE_OPAL (LEP Run 2)

Spires ID: [4300807](#)

Status: VALIDATED

Authors:

- Frank Siegert (frank.siegert@durham.ac.uk);

References:

- Eur.Phys.J.C17:19-51,2000
- arXiv: [hep-ex/0001055](#)

Run details:

- e+ e- collisions:
- e+ e- \rightarrow jet jet (+ jets) at 183 GeV. * no cuts needed

Differential and integrated jet rates for Durham and JADE jet algorithms at $\sqrt{s} = 183$.

5.11 JADE_OPAL_2000_S4300807_189GEV

Jet rates in e+e- at OPAL [189 GeV].

Experiment: JADE_OPAL (LEP Run 2)

Spires ID: [4300807](#)

Status: VALIDATED

Authors:

- Frank Siegert (frank.siegert@durham.ac.uk);

References:

- Eur.Phys.J.C17:19-51,2000
- arXiv: [hep-ex/0001055](#)

Run details:

- e+ e- collisions:
- e+ e- \rightarrow jet jet (+ jets) at 189 GeV. * no cuts needed

Differential and integrated jet rates for Durham and JADE jet algorithms at $\sqrt{s} = 189$.

5.12 JADE_OPAL_2000_S4300807_91GEV

Jet rates in e+e- at OPAL [91 GeV].

Experiment: JADE_OPAL (LEP Run I)

Spires ID: [4300807](#)

Status: VALIDATED

Authors:

- Frank Siegert (frank.siegert@durham.ac.uk);

References:

- Eur.Phys.J.C17:19-51,2000
- arXiv: [hep-ex/0001055](#)

Run details:

- e+ e- collisions:
- e+ e- \rightarrow jet jet (+ jets) at 91.2 GeV. * no cuts needed

Differential and integrated jet rates for Durham and JADE jet algorithms at $\sqrt{s} = 91.2$.

5.13 OPAL_1998_S3780481

Measurements of flavor dependent fragmentation functions in $Z^0 \rightarrow q \text{ anti-}q$ events.

Experiment: OPAL (LEP 1)

Spires ID: [3780481](#)

Status: VALIDATED

Authors:

- Hendrik Hoeth [〈hendrik.hoeth@cern.ch〉](mailto:hendrik.hoeth@cern.ch);

References:

- Eur. Phys. J, C7, 369–381 (1999)
- hep-ex/9807004

Run details:

- Hadronic Z decay events generated on the Z pole ($\sqrt{s} = 91.2 \text{ GeV}$)

Measurement of scaled momentum distributions and total charged multiplicities in flavour tagged events at LEP 1. OPAL measured these observables in uds-, c-, and b-events separately. An inclusive measurement is also included.

6. Tevatron analyses

6.1 CDF_1990_S2089246

CDF pseudorapidity distributions at 630 and 1800 GeV

Experiment: CDF (Tevatron Run 0)

Spires ID: [2089246](#)

Status: UNVALIDATED

Authors:

- Andy Buckley [<andy.buckley@cern.ch>](mailto:andy.buckley@cern.ch);

References:

- Phys.Rev.D41:2330,1990
- DOI: [10.1103/PhysRevD.41.2330](https://doi.org/10.1103/PhysRevD.41.2330)

Run details:

- Energy: $\sqrt{s} = 630$ and 1800 GeV
- Event type: generic QCD events
- $|\eta| < 3.5$

Pseudorapidity distributions based on the CDF 630 and 1800 GeV runs from 1987. All data is detector corrected. The data confirms the UA5 measurement of a N/η rise with energy faster than $\ln \sqrt{s}$, and as such this analysis is important for constraining the energy evolution of minimum bias and underlying event characteristics in MC simulations.

6.2 CDF_1994_S2952106

CDF Run I color coherence analysis.

Experiment: CDF (Tevatron Run 1)

Spires ID: [2952106](#)

Status: UNVALIDATED

Authors:

- Lars Sonnenschein (Lars.Sonnenschein@cern.ch);

References:

- Phys.Rev.D50,5562,1994
- DOI: [10.1103/PhysRevD.50.5562](https://doi.org/10.1103/PhysRevD.50.5562)

Run details:

- Energy: $\sqrt{s} = 1800$ GeV
- Event type: generic QCD events
- Cut on primary vertex z position: $z(\text{PV}) \leq 60$ cm
- p_{\perp}^{min} : leading jet = 100 GeV; and 3rd jet = 10 GeV
- Max. pseudorapidity range of 2nd and 3rd jets: $|\eta| < 0.7$
- Azimuthal angle requirement: $\Delta\phi < \pi/18$ (transverse back-to-backness)
- MET cut requirement: $mET/\sqrt{\text{Scalar } E_T} < 6.0 \text{ GeV}$

CDF Run I color coherence analysis. Events with ≥ 3 jets are selected and E_t distributions of the three highest- p_{\perp} jets are obtained. The plotted quantities are the ΔR between the 2nd and 3rd leading jets in the p_{\perp} and pseudorapidity of the 3rd jet, and $\alpha = d\eta/d\phi$, where $d\eta$ is the pseudorapidity difference between the 2nd and 3rd jets and $d\phi$ is their azimuthal angle difference.

Since the data has not been detector-corrected, a bin by bin correction is applied, based on the distributions with ideal and CDF simulation as given in the publication.

6.3 CDF_2000_S4155203

Z p_{\perp} measurement in CDF $Z \rightarrow e^+ e^-$ events

Experiment: CDF (Tevatron Run 1)

Spires ID: [4155203](#)

Status: UNVALIDATED

Authors:

- Hendrik Hoeth (hendrik.hoeth@cern.ch);

References:

- Phys.Rev.Lett.84:845-850,2000
- arXiv: [hep-ex/0001021](#)
- DOI: [10.1103/PhysRevLett.84.845](#)

Run details:

- Tevatron Run I: p pbar collisions at 1800 GeV
- Z Drell-Yan with $e^+ e^-$ decay mode (for Z *and* γ^*) only.

Measurement of transverse momentum and total cross section of e^+e^- pairs in the Z-boson region of $66 \text{ GeV}/c^2 < m_{ee} < 116 \text{ GeV}/c^2$ from pbar-p collisions at $\sqrt{s} = 1.8 \text{ TeV}$, with the Tevatron CDF detector.

The Z p_{\perp} , in a fully-factorised picture, is generated by the momentum balance against initial state radiation (ISR) and the primordial/intrinsic p_{\perp} of the Z's parent partons in the incoming hadrons. The Z p_{\perp} is important in generator tuning to fix the interplay of ISR and multi-parton interactions (MPI) ingenerating ‘underlying event’ activity.

6.4 CDF_2001_S4751469

Field & Stuart Run I underlying event analysis.

Experiment: CDF (Tevatron Run 1)

Spires ID: [4751469](#)

Status: VALIDATED

Authors:

- Andy Buckley [⟨andy.buckley@durham.ac.uk⟩](mailto:andy.buckley@durham.ac.uk);

References:

- Phys.Rev.D65:092002,2002
- FNAL-PUB 01/211-E

Run details:

- CDF Run I conditions: ppbar QCD interactions at 1800 GeV. Leading jet bins from 0–49 GeV: usually can be filled with a single generator run without kinematic sub-samples, with $\sim 1M$ events.

The original CDF underlying event analysis, based on decomposing each event into a transverse structure with “toward”, “away” and “transverse” regions defined relative to the azimuthal direction of the leading jet in the event. Since the toward region is by definition dominated by the hard process, as is the away region by momentum balance in the matrix element, the transverse region is most sensitive to multi-parton interactions. The transverse regions occupy $|\phi| \in [60^\circ, 120^\circ]$ for $|\eta| < 1$. The p_\perp ranges for the leading jet are divided experimentally into the ‘min-bias’ sample from 0–20 GeV, and the ‘JET20’ sample from 18–49 GeV.

6.5 CDF_2002_S4796047

CDF Run 1 charged multiplicity measurement

Experiment: CDF (Tevatron Run 1)

Spires ID: [4796047](#)

Status: UNVALIDATED

Authors:

- Hendrik Hoeth (hendrik.hoeth@cern.ch);

References:

- Phys.Rev.D65:072005,2002
- DOI: [10.1103/PhysRevD.65.072005](https://doi.org/10.1103/PhysRevD.65.072005)

Run details:

- Energy: $\sqrt{s} = 630$ and 1800 GeV
- Event type: generic QCD events
- TODO: MORE?

A study of pp collisions at $\sqrt{s} = 1800$ and 630 GeV collected using a minimum bias trigger in which the data set is divided into two classes corresponding to ‘soft’ and ‘hard’ interactions. For each subsample, the analysis includes measurements of the multiplicity, transverse momentum (p_{\perp}) spectra, and the average p_{\perp} and event-by-event p_{\perp} dispersion as a function of multiplicity. A comparison of results shows distinct differences in the behavior of the two samples as a function of the center of mass energy. The properties of the soft sample are invariant as a function of c.m. energy.

6.6 CDF_2004_S5839831

Transverse cone and 'Swiss cheese' underlying event studies

Experiment: CDF (Tevatron Run 2)

Spires ID: [5839831](#)

Status: UNVALIDATED

Authors:

- Andy Buckley [⟨ andy.buckley@durham.ac.uk ⟩](mailto:andy.buckley@durham.ac.uk);

References:

- Phys. Rev. D70, 072002 (2004)
- arXiv: [hep-ex/0404004](#)

Run details:

- Two different beam energies: $\sqrt{s} = 630$ & 1800 GeV
- Event type: generic QCD events
- Several p_{\perp}^{\min} cutoffs are probably required to fill the profile histograms, e.g.
 - * 0 (min bias), $30, 90, 150$ GeV at 1800 GeV; and * 0 (min bias), $20, 90, 150$ GeV at 630 GeV

This analysis studies the underlying event via transverse cones of $R = 0.7$ at 90 degrees in ϕ relative to the leading (highest E) jet, at $\sqrt{s} = 630$ and 1800 GeV. This is similar to the 2001 CDF UE analysis, except that cones, rather than the whole central η range are used. The transverse cones are categorised as TransMIN and TransMAX on an event-by-event basis, to give greater sensitivity to the UE component.

'Swiss Cheese' distributions, where cones around the leading n jets are excluded from the distributions, are also included for $n = 2, 3$.

This analysis is useful for constraining the energy evolution of the underlying event, since it performs the same analyses at two distinct CoM energies.

WARNING: this analysis is not currently considered valid for MC tuning and validation studies due to ambiguities in the paper and non-reproducibility of the MC plots shown in the paper. The fit to data is sufficiently poor that this analysis skews the overall goodness of fit in tuning studies, and has to be excluded. If you can help to improve this analysis and make it usable for validation studies, please get in touch!

6.7 CDF_2005_S6217184

CDF Run II jet shape analysis

Experiment: CDF (Tevatron Run 2)

Spires ID: [6217184](#)

Status: UNVALIDATED

Authors:

- Lars Sonnenschein [⟨Lars.Sonnenschein@cern.ch⟩](mailto:Lars.Sonnenschein@cern.ch);
- Andy Buckley [⟨andy.buckley@cern.ch⟩](mailto:andy.buckley@cern.ch);

References:

- Phys.Rev.D71:112002,2005
- DOI: [10.1103/PhysRevD.71.112002](https://doi.org/10.1103/PhysRevD.71.112002)
- arXiv: [hep-ex/0505013](https://arxiv.org/abs/hep-ex/0505013)

Run details:

- Energy: $\sqrt{s} = 1960$ GeV
- Event type: generic QCD events.
- $\eta \in [-2, 2]$ cut used on final state.
- Jet axes must have $|y| \in [0.1, 0.7]$.
- Jet shape $r \in [0.0, 0.7]$
- Jet p_{\perp}^{\min} in plots is 37 GeV/c: choose generator min p_{\perp} somewhere well below this.

Measurement of jet shapes in inclusive jet production in p pbar collisions at center-of-mass energy $\sqrt{s} = 1.96$ TeV. The data cover jet transverse momenta from 37–380 GeV and absolute jet rapidities in the range 0.1–0.7.

6.8 CDF_2006_S6653332

p_{\perp} and eta distributions of jets in Z + jet production

Experiment: CDF (Tevatron Run 2)

Spires ID: [6653332](#)

Status: UNVALIDATED

Authors:

- Lars Sonnenschein (Lars.Sonnenschein@cern.ch);

References:

- Phys.Rev.D.74:032008,2006
- DOI: [10.1103/PhysRevD.74.032008](https://doi.org/10.1103/PhysRevD.74.032008)
- arXiv: [hep-ex/0605099v2](https://arxiv.org/abs/hep-ex/0605099v2)

Run details:

- Energy: $\sqrt{s} = 1960$ GeV
- Event type: Z + jets events
- Jets min p_{\perp} cut: $p_{\perp \text{ -jet}} > 20$ GeV
- Leptons min p_{\perp} cut: $p_{\perp \text{ -jet}} > 10$ GeV

Measurement of the b jet cross section in events with Z boson in p pbar collisions at center-of-mass energy $\sqrt{s} = 1.96$ TeV. The data cover jet transverse momenta above 20 GeV and jet pseudo-rapidities in the range -1.5 to 1.5. Z bosons are identified in their electron and muon decay modes in an invariant dilepton mass range between 66 and 116 GeV.

6.9 CDF_2007_S7057202

CDF Run II inclusive jet cross-section using the kT algorithm

Experiment: CDF (Tevatron Run 2)

Spires ID: [7057202](#)

Status: UNVALIDATED

Authors:

- David Voong
- James Monk (jmonk@hep.ucl.ac.uk);

References:

- Phys.Rev.D75:092006,2007
- Erratum-ibid.D75:119901,2007
- FNAL-PUB 07/026-E
- hep-ex/0701051

Run details:

- Standard Tevatron Run II: p-pbar collisions at 1960 GeV. Jet p_{\perp} bins from 54 GeV to 700 GeV. Jet rapidity $< |2.1|$.

CDF Run II measurement of inclusive jet cross sections at a p-pbar collision energy of 1.96 TeV. Jets are reconstructed in the central part of the detector ($|y| < 2.1$) using the kT algorithm with an R parameter of 0.7. The minimum jet p_{\perp} considered is 54 GeV, with a maximum around 700 GeV.

The inclusive jet p_{\perp} is plotted in bins of rapidity $|y| < 0.1$, $0.1 < |y| < 0.7$, $0.7 < |y| < 1.1$, $1.1 < |y| < 1.6$ and $1.6 < |y| < 2.1$.

6.10 CDF_2008_LEADINGJETS

CDF Run 2 underlying event in leading jet events

Experiment: CDF (Tevatron Run 2)

Spires ID: [NONE](#)

Status: VALIDATED

Authors:

- Hendrik Hoeth (hendrik.hoeth@cern.ch);

References:

-

Run details:

- Tevatron Run 2: ppbar QCD interactions at 1960 GeV. Particles with $c\tau > 10$ mm should be set stable. Several p_{\perp} min cutoffs are probably required to fill the profile histograms: * $p_{\perp} \text{ min} = 0$ (min bias), 10, 20, 50, 100, 150 GeV * The corresponding merging points are at $p_T = 0, 30, 50, 80, 130, 180$ GeV

Rick Field's measurement of the underlying event in leading jet events. If the leading jet of the event is within $|\eta| < 2$, the event is accepted and "toward", "away" and "transverse" regions are defined in the same way as in the original (2001) CDF underlying event analysis. The leading jet defines the ϕ direction of the toward region. The transverse regions are most sensitive to the underlying event.

6.11 CDF_2008_NOTE_9351

CDF Run 2 underlying event in Drell-Yan

Experiment: CDF (Tevatron Run 2)

Spires ID: [NONE](#)

Status: VALIDATED

Authors:

- Hendrik Hoeth (hendrik.hoeth@cern.ch);

References:

- CDF public note 9351

Run details:

- Tevatron Run 2: ppbar collisions at 1960 GeV.
- Drell-Yan events with $Z/\gamma^* \rightarrow ee$ and $Z/\gamma^* \rightarrow \mu\mu$.
- A mass cut $m_{ll} > 70$ GeV can be applied on generator level.
- Particles with $c\tau > 10$ mm should be set stable.

Deepak Kar’s and Rick Field’s measurement of the underlying event in Drell-Yan events. $Z \rightarrow ee$ and $Z \rightarrow \mu\mu$ events are selected using a Z mass window cut between 70 and 110 GeV. “Toward”, “away” and “transverse” regions are defined in the same way as in the original (2001) CDF underlying event analysis. The reconstructed Z defines the ϕ direction of the toward region. The leptons are ignored after the Z has been reconstructed. Thus the region most sensitive to the underlying event is the toward region (the recoil jet is boosted into the away region).

6.12 CDF_2008_S7540469

Measurement of differential $Z/\gamma^* + \text{jet} + X$ cross sections

Experiment: CDF (Tevatron Run 2)

Spires ID: [7540469](#)

Status: VALIDATED

Authors:

- Frank Siegert (frank.siegert@durham.ac.uk);

References:

- Phys.Rev.Lett.100:102001,2008
- arXiv: [0711.3717](#)

Run details:

- Tevatron Run 2 conditions: $p\bar{p} \rightarrow e^+ e^- + \text{jets}$ at 1960 GeV.
- Needs mass cut on lepton pair to avoid photon singularity: min. range $66 < m_{ee} < 116$

Cross sections as a function of jet transverse momentum in 1 and 2 jet events, and jet multiplicity in $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV, based on an integrated luminosity of 1.7 fb⁻¹. The measurements cover the rapidity region $|y_{\text{jet}}| < 2.1$ and the transverse momentum range $p_{\perp\text{-jet}} > 30$ GeV/ c .

6.13 CDF_2008_S7541902

Jet p_{\perp} distributions for 4 jet multiplicity bins as well as the jet multiplicity distribution in $W + \text{jets}$ events.

Experiment: CDF (Tevatron Run 2)

Spires ID: [7541902](#)

Status: UNVALIDATED

Authors:

- Ben Cooper \langle b.d.cooper@qmul.ac.uk \rangle ;
- Emily Nurse \langle nurse@hep.ucl.ac.uk \rangle ;

References:

- arXiv: [0711.4044](#)
- Phys.Rev.D77:011108,2008

Run details:

- Requires the process $p\bar{p} \rightarrow W \rightarrow e\nu$, additional hard jets will also have to be included to get a good description. The LO process in Herwig is set with IPROC=1451.

Measurement of the cross section for W boson production in association with jets in $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV. The analysis uses 320 pb $^{-1}$ of data collected with the CDF II detector. W bosons are identified in their $e\nu$ decay channel and jets are reconstructed using an $R < 0.4$ cone algorithm. For each $W + \geq n$ -jet sample (where $n = 1-4$) a measurement of $d\sigma(p\bar{p} \rightarrow W + \geq n \text{ jet})/dE_T(nth\text{-jet}) \times \text{BR}(W \rightarrow e\nu)$ is made, where $dE_T(nth\text{-jet})$ is the E_T of the n th-highest E_T jet above 20 GeV. A measurement of the total cross section, $\sigma(p\bar{p} \rightarrow W + \geq n\text{-jet}) \times \text{BR}(W \rightarrow e\nu)$ with $E_T(nth\text{-jet}) > 25$ GeV is also made. Both measurements are made for jets with $|\eta| < 2$ and for a limited region of the $W \rightarrow e\nu$ decay phase space: $|\eta_e| < 1.1$, $p_{Te} > 20$ GeV, $p_{T\nu} > 30$ GeV and $M_{Te} > 20$ GeV. The cross sections are corrected for all detector effects and can be directly compared to particle level $W + \text{jet(s)}$ predictions. These measurements can be used to test and tune QCD predictions for the number of jets in and kinematics of $W + \text{jets}$ events.

6.14 CDF_2008_S7782535

CDF Run II b-jet shape paper

Experiment: CDF (Tevatron Run 2)

Spires ID: 7782535

Status: UNVALIDATED

Authors:

- Alison Lister \langle alister@fnal.gov \rangle ;
- Emily Nurse \langle nurse@hep.ucl.ac.uk \rangle ;

References:

- arXiv: [0806.1699](https://arxiv.org/abs/0806.1699)
- Phys.Rev.D78:072005,2008

Run details:

- Requires $2 \rightarrow 2$ QCD scattering processes. The minimum jet E_t is 52 GeV, so a cut on kinematic p_{\perp}^{\min} may be required for good statistics.

A measurement of the shapes of b-jets using 300 pb $^{-1}$ of data obtained with CDF II in $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV. The measured quantity is the average integrated jet shape, which is computed over an ensemble of jets. This quantity is expressed as $\Psi(r/R) = \langle \frac{p_{\perp T}(0 \rightarrow r)}{p_{\perp T}(0 \rightarrow R)} \rangle$, where $p_{\perp}(0 \rightarrow r)$ is the scalar sum of the transverse momenta of all objects inside a sub-cone of radius r around the jet axis. The integrated shapes are by definition normalized such that $\Psi(r/R = 1) = 1$. The measurement is done in bins of jet p_{\perp} in the range 52 to 300 GeV/c. The jets have $|\eta| < 0.7$. The b-jets are expected to be broader than inclusive jets. Moreover, b-jets containing a single b-quark are expected to be narrower than those containing a $b\bar{b}$ pair from gluon splitting.

6.15 CDF_2008_S7828950

CDF Run II inclusive jet cross-section using the Midpoint algorithm

Experiment: CDF (Tevatron Run 2)

Spires ID: [7828950](#)

Status: UNVALIDATED

Authors:

- Craig Group (group@fnal.gov);

References:

- arXiv: [0807.2204](#)
- Phys.Rev.D78:052006,2008

Run details:

- Requires $2 \rightarrow 2$ QCD scattering processes. The minimum jet E_t is 62 GeV, so a cut on kinematic p_{\perp}^{\min} may be required for good statistics.

Measurement of the inclusive jet cross section in $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV as a function of jet E_t , for $E_t > 62$ GeV. The data is collected by the CDF II detector and has an integrated luminosity of 1.13 fb⁻¹. The measurement was made using the cone-based Midpoint jet clustering algorithm in rapidity bins within $|y| < 2.1$. This measurement can be used to provide increased precision in PDFs at high parton momentum fraction x .

6.16 CDF_2008_S8095620

CDF Run II Z+b-jet cross section paper, 2 fb-1

Experiment: CDF (Tevatron Run 2)

Spires ID: [8095620](#)

Status: UNVALIDATED

Authors:

- Emily Nurse (nurse@hep.ucl.ac.uk);

References:

- arXiv: [0812.4458](#)

Run details:

- Requires the process $p\bar{p} \rightarrow Z \rightarrow \ell\ell$, where ℓ is e or μ . Additional hard jets will also have to be included to get a good description.

Measurement of the b-jet production cross section for events containing a Z boson produced in $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV, using data corresponding to an integrated luminosity of 2 fb⁻¹ collected by the CDF II detector at the Tevatron. Z bosons are selected in the electron and muon decay modes. Jets are considered with transverse energy $E_T > 20$ GeV and pseudorapidity $|\eta| < 1.5$. The ratio of the integrated Z + b-jet cross section to the inclusive Z production cross section is measured differentially in jet E_T , jet η , Z -boson transverse momentum, number of jets, and number of b-jets. The first two measurements have an entry for each b-jet in the event, the last three measurements have one entry per event.

6.17 CDF_2009_S8233977

CDF Run 2 min bias cross-section analysis

Experiment: CDF (Tevatron Run 2)

Spires ID: [8233977](#)

Status: PARTIALLY VALIDATED

Authors:

- Hendrik Hoeth (hendrik.hoeth@cern.ch);

References:

- CDF public note 9337
- hep-ex/0904.1098

Run details:

- Tevatron Run 2: ppbar QCD interactions at 1960 GeV.
- Particles with $c\tau > 10$ mm should be set stable.

Niccolo Moggi's minbias analysis. Minimum bias events are used to measure the average track p_{\perp} vs charged multiplicity, a track p_{\perp} distribution and an inclusive $\sum E_T$ distribution. WARNING: Only average track p_{\perp} vs charged multiplicity is validated!

6.18 D0_2001_S4674421

Tevatron Run I differential W/Z boson cross-section analysis

Experiment: D0 (Tevatron Run 1)

Spires ID: [4674421](#)

Status: UNVALIDATED

Authors:

- Lars Sonnenschein (Lars.Sonnenschein@cern.ch);

References:

- Phys.Lett.B517:299-308,2001
- DOI: [10.1016/S0370-2693\(01\)01020-6](https://doi.org/10.1016/S0370-2693(01)01020-6)
- arXiv: [hep-ex/0107012v2](https://arxiv.org/abs/hep-ex/0107012v2)

Run details:

- Energy: $\sqrt{s} = 1800$ GeV
- Event type: W/Z events with decays to first generation leptons

Measurement of differential W/Z boson cross section and ratio in p pbar collisions at center-of-mass energy $\sqrt{s} = 1.8$ TeV. The data cover electrons and neutrinos in a pseudo-rapidity range of -2.5 to 2.5.

6.19 D0_2004_S5992206

Run II jet azimuthal decorrelation analysis

Experiment: D0 (Tevatron Run 2)

Spires ID: [5992206](#)

Status: VALIDATED

Authors:

- Lars Sonnenschein (lars.sonnenschein@cern.ch);

References:

- Phys. Rev. Lett., 94, 221801 (2005)
- arXiv: [hep-ex/0409040](#)

Run details:

- Tevatron Run 2: ppbar QCD interactions at 1960 GeV.

Correlations in the azimuthal angle between the two largest p_{\perp} jets have been measured using the D0 detector in ppbar collisions at 1960 GeV. The analysis is based on an inclusive dijet event sample in the central rapidity region. The correlations are determined for four different p_{\perp} intervals.

6.20 D0_2006_S6438750

Inclusive isolated photon cross-section, differential in p_{\perp} (gamma)

Experiment: D0 (Tevatron Run 2)

Spires ID: 6438750

Status: UNVALIDATED

Authors:

- Andy Buckley [⟨ andy.buckley@durham.ac.uk ⟩](mailto:andy.buckley@durham.ac.uk);
- Gavin Hesketh [⟨ gavin.hesketh@cern.ch ⟩](mailto:gavin.hesketh@cern.ch);

References:

- Phys.Lett.B639:151-158,2006, Erratum-ibid.B658:285-289,2008
- DOI: [10.1016/j.physletb.2006.04.048](https://doi.org/10.1016/j.physletb.2006.04.048)
- arXiv: [hep-ex/0511054](https://arxiv.org/abs/hep-ex/0511054)

Run details:

- Requires gamma + jet (q,qbar,g) hard processes
* for Pythia 6, MSEL=10 for with MSUB indices 14, 18, 29, 114, 115 enabled
- Lowest p_{\perp} bin is at 23 GeV: a p_{\perp} min cut at 10–15 GeV may be required to get good statistics.

Measurement of differential cross section for inclusive production of isolated photons in p pbar collisions at $\sqrt{s} = 1.96$ TeV with the D0 detector at the Fermilab Tevatron collider. The photons span transverse momenta 23–300 GeV and have pseudorapidity $|\eta| < 0.9$. Isolated direct photons are probes of pQCD via the annihilation ($q \bar{q} \rightarrow \text{gamma } g$) and quark-gluon Compton scattering ($q g \rightarrow \text{gamma } q$) processes, the latter of which is also sensitive to the gluon PDF. The initial state radiation / resummation formalisms are sensitive to the resulting photon p_{\perp} spectrum

6.21 D0_2007_S7075677

Z/gamma* + X cross-section shape, differential in y(Z)

Experiment: D0 (Tevatron Run 2)

Spires ID: [7075677](#)

Status: UNCLEAR: Photons in Z reconstruction?

Authors:

- Andy Buckley [⟨ andy.buckley@durham.ac.uk ⟩](mailto:andy.buckley@durham.ac.uk);
- Gavin Hesketh [⟨ ghesketh@fnal.gov ⟩](mailto:ghesketh@fnal.gov);
- Frank Siegert [⟨ frank.siegert@durham.ac.uk ⟩](mailto:frank.siegert@durham.ac.uk);

References:

- Phys.Rev.D76:012003,2007
- arXiv: [hep-ex/0702025](#)

Run details:

- Tevatron Run 2 conditions:
- $p\bar{p} \rightarrow e^+ e^- + \text{jets}$ at 1960 GeV.
- Needs mass cut on lepton pair to avoid photon singularity: min. range $71 < m_{ee} < 111$

Cross sections as a function of boson rapidity $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV, based on an integrated luminosity of 0.4 fb⁻¹.

6.22 D0_2008_S6879055

Measurement of the ratio $\sigma(Z/\gamma^* + n \text{ jets})/\sigma(Z/\gamma^*)$

Experiment: D0 (Tevatron Run 2)

Spires ID: [6879055](#)

Status: VALIDATED

Authors:

- Giulio Lenzi
- Frank Siegert (frank.siegert@durham.ac.uk);

References:

- [hep-ex/0608052](#)

Run details:

- Tevatron Run 2 conditions:
- $p\bar{p} \rightarrow e^+ e^- + \text{jets}$ at 1960 GeV
- Needs mass cut on lepton pair to avoid photon singularity: min. range $75 < m_{ee} < 105$

Cross sections as a function of p_\perp of the three leading jets and n-jet cross section ratios in $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV, based on an integrated luminosity of 0.4 fb⁻¹.

6.23 D0_2008_S7554427

Z/gamma* + X cross-section shape, differential in p_{\perp} (Z)

Experiment: D0 (Tevatron Run 2)

Spires ID: [7554427](#)

Status: VALIDATED

Authors:

- Andy Buckley [⟨ andy.buckley@durham.ac.uk ⟩](mailto:andy.buckley@durham.ac.uk);
- Frank Siegert [⟨ frank.siegert@durham.ac.uk ⟩](mailto:frank.siegert@durham.ac.uk);

References:

- arXiv: [0712.0803](#)

Run details:

- Tevatron Run 2 conditions:
- $p\bar{p} \rightarrow e^+ e^- + \text{jets}$ at 1960 GeV.
- Needs mass cut on lepton pair to avoid photon singularity: m in range $40 < m_{ee} < 200$ GeV

Cross sections as a function of p_{\perp} of the vector boson inclusive and in forward region ($|y| > 2, p_{\perp} < 30$ GeV) in $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV, based on an integrated luminosity of 0.98 fb⁻¹.

6.24 D0_2008_S7662670

Measurement of D0 Run II differential jet cross sections

Experiment: D0 (Tevatron Run 2)

Spires ID: [7662670](#)

Status: UNVALIDATED

Authors:

- Andy Buckley [⟨ andy.buckley@durham.ac.uk ⟩](mailto:andy.buckley@durham.ac.uk);
- Gavin Hesketh [⟨ gavin.hesketh@cern.ch ⟩](mailto:gavin.hesketh@cern.ch);

References:

- Phys.Rev.Lett.101:062001,2008
- DOI: [10.1103/PhysRevLett.101.062001](https://doi.org/10.1103/PhysRevLett.101.062001)
- arXiv: [0802.2400v3](https://arxiv.org/abs/0802.2400v3)

Run details:

- Energy: $\sqrt{s} = 1960$ GeV
- Event type: QCD events
- p_{\perp}^{\min} cut may be necessary: lowest jet p_{\perp} bin is at 50 GeV

Measurement of the inclusive jet cross section in p pbar collisions at center-of-mass energy $\sqrt{s} = 1.96$ TeV. The data cover jet transverse momenta from 50–600 GeV and jet rapidities in the range -2.4 to 2.4.

6.25 D0_2008_S7719523

Isolated gamma + jet cross-sections, differential in p_{\perp} (gamma) for various y -bins

Experiment: D0 (Tevatron Run 2)

Spires ID: [7719523](#)

Status: UNVALIDATED

Authors:

- Andy Buckley [⟨andy.buckley@durham.ac.uk⟩](mailto:andy.buckley@durham.ac.uk);
- Gavin Hesketh [⟨gavin.hesketh@cern.ch⟩](mailto:gavin.hesketh@cern.ch);

References:

- Phys.Lett.B666:435-445,2008
- DOI: [10.1016/j.physletb.2008.06.076](https://doi.org/10.1016/j.physletb.2008.06.076)
- arXiv: [0804.1107v2](https://arxiv.org/abs/0804.1107v2)

Run details:

- Produce only gamma + jet (q,qbar,g) hard processes
 - * for Pythia 6: MSEL=10, and MSUB indices 14, 29 & 115 enabled
- Lowest bin edge at 30 GeV: kinematic p_{\perp}^{\min} cut may be required for good statistics.

The process $p \bar{p} \rightarrow \text{photon} + \text{jet} + X$ as studied by the D0 detector at the Fermilab Tevatron collider at center-of-mass energy $\sqrt{s} = 1.96$ TeV. Photons are reconstructed in the central rapidity region $|y_{\gamma}| < 1.0$ with transverse momenta in the range 30–400 GeV, while jets are reconstructed in either the central $|y_{jet}| < 0.8$ or forward $1.5 < |y_{jet}| < 2.5$ rapidity intervals with $p_{\perp jet} > 15$ GeV. The differential cross section $d^3\sigma/dp_{\perp\gamma}dy_{\gamma}dy_{jet}$ is measured as a function of $p_{\perp\gamma}$ in four regions, differing by the relative orientations of the photon and the jet.

MC predictions have trouble with simultaneously describing the measured normalization and $p_{\perp\gamma}$ dependence of the cross section in any of the four measured regions.

6.26 D0_2008_S7837160

Measurement of W charge asymmetry from D0 Run II

Experiment: D0 (Tevatron Run 2)

Spires ID: [7837160](#)

Status: UNVALIDATED

Authors:

- Andy Buckley [⟨ andy.buckley@durham.ac.uk ⟩](mailto:andy.buckley@durham.ac.uk);
- Gavin Hesketh [⟨ gavin.hesketh@cern.ch ⟩](mailto:gavin.hesketh@cern.ch);

References:

- Phys.Rev.Lett.101:211801,2008
- DOI: [10.1103/PhysRevLett.101.211801](https://doi.org/10.1103/PhysRevLett.101.211801)
- arXiv: [0807.3367v1](https://arxiv.org/abs/0807.3367v1)

Run details:

- Event type: W production with decay to e ν_e only
* for Pythia 6: MSEL = 12, MDME(206,1) = 1
- Energy: 1.96 TeV

Measurement of the electron charge asymmetry in $p \bar{p} \rightarrow W + X \rightarrow e \nu_e + X$ events at a center of mass energy of 1.96 TeV. The asymmetry is measured as a function of the electron transverse momentum and pseudorapidity in the interval $(-3.2, 3.2)$.

This data is sensitive to proton parton distribution functions due to the valence asymmetry in the incoming quarks which produce the W. Initial state radiation should also affect the p_\perp distribution.

6.27 D0_2008_S7863608

Measurement of differential $Z/\gamma^* + \text{jet} + X$ cross sections

Experiment: D0 (Tevatron Run 2)

Spires ID: [7863608](#)

Status: VALIDATED

Authors:

- Andy Buckley [⟨ andy.buckley@durham.ac.uk ⟩](mailto:andy.buckley@durham.ac.uk);
- Gavin Hesketh [⟨ gavin.hesketh@fnal.gov ⟩](mailto:gavin.hesketh@fnal.gov);
- Frank Siegert [⟨ frank.siegert@durham.ac.uk ⟩](mailto:frank.siegert@durham.ac.uk);

References:

- arXiv: [0808.1296](#)

Run details:

- Tevatron Run 2 conditions:
- $p\bar{p} \rightarrow \mu^+ \mu^- + \text{jets}$ at 1960 GeV
- Needs mass cut on lepton pair to avoid photon singularity: min. range $65 < m_Z < 115$

Cross sections as a function of p_\perp and rapidity of the boson and p_\perp and rapidity of the leading jet in $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV, based on an integrated luminosity of 1.0 fb^{-1} .

6.28 D0_2009_S8202443

Z/ γ^* + jet + X cross sections differential in p_\perp (jet 1,2,3)

Experiment: D0 (Tevatron Run 2)

Spires ID: [8202443](#)

Status: NOT VALIDATED

Authors:

- Frank Siegert (frank.siegert@durham.ac.uk);

References:

- arXiv: [0903.1748](#)

Run details:

- Tevatron Run 2 conditions:
- $p\bar{p} \rightarrow e^+ e^- + \text{jets}$ at 1960 GeV
- Needs mass cut on lepton pair to avoid photon singularity: min. range $65 < m_Z < 115$

Cross sections as a function of p_\perp of the three leading jets in $Z/\gamma^*(\rightarrow e^+e^-) + \text{jet} + X$ production in $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV, based on an integrated luminosity of 1.0fb⁻¹.

7. HERA analyses

7.1 H1_1994_S2919893

H1 energy flow and charged particle spectra in DIS

Experiment: H1 (HERA)

Spires ID: [2919893](#)

Status: VALIDATED

Authors:

- Peter Richardson (peter.richardson@durham.ac.uk);

References:

- Z.Phys.C63:377-390,1994
- DOI: [10.1007/BF01580319](https://doi.org/10.1007/BF01580319)

Run details:

- Event type: e- p / e+ p deep inelastic scattering
- HERA beam conditions: 820 GeV protons colliding with 26.7 GeV electrons

Global properties of the hadronic final state in deep inelastic scattering events at HERA are investigated. The data are corrected for detector effects. Energy flows in both the laboratory frame and the hadronic centre of mass system, and energy-energy correlations in the laboratory frame are presented.

Historically, the Ariadne colour dipole model provided the only satisfactory description of this data, hence making it a useful 'target' analysis for MC shower models.

7.2 H1_1995_S3167097

Transverse energy and forward jet production in the low x regime at H1

Experiment: H1 (HERA Run I)

Spires ID: [3167097](#)

Status: UNVALIDATED

Authors:

- Leif Lonnblad (leif.lonnblad@thep.lu.se);

References:

- Phys.Lett.B356:118,1995
- hep-ex/9506012

Run details:

- HERA beam conditions: 820 GeV protons colliding with 26.7 GeV electrons
- DIS events with an outgoing electron energy > 12 GeV
- $5 \text{ GeV}^2 < Q^2 < 100 \text{ GeV}^2$, $10^{-4} < x < 10^{-2}$.

DIS events at low x may be sensitive to new QCD dynamics such as BFKL or CCFM radiation. In particular, BFKL is expected to produce more radiation at high transverse energy in the rapidity span between the proton remnant and the struck quark jet. Performing a transverse energy sum in bins of x and η may distinguish between DGLAP and BFKL evolution.

7.3 H1_2000_S4129130

H1 energy flow in DIS

Experiment: H1 (HERA)

Spires ID: [4129130](#)

Status: VALIDATED

Authors:

- Peter Richardson (peter.richardson@durham.ac.uk);

References:

- Eur.Phys.J.C12:595-607,2000
- DOI: [10.1007/s100520000287](https://doi.org/10.1007/s100520000287)
- arXiv: [hep-ex/9907027v1](https://arxiv.org/abs/hep-ex/9907027v1)

Run details:

- Event type: e+ p deep inelastic scattering
- Energy: p at 820 GeV, e+ at 27.5 GeV $\rightarrow \sqrt{s} = 300$ GeV

Measurements of transverse energy flow for neutral current deep-inelastic scattering events produced in positron-proton collisions at HERA. The kinematic range covers squared momentum transfers Q^2 from 3.2 to 2200 GeV²; the Bjorken scaling variable x from 8×10^{-5} to 0.11 and the hadronic mass W from 66 to 233 GeV. The transverse energy flow is measured in the hadronic centre of mass frame and is studied as a function of Q^2 , x , W and pseudorapidity. The behaviour of the mean transverse energy in the central pseudorapidity region and an interval corresponding to the photon fragmentation region are analysed as a function of Q^2 and W .

This analysis is useful for exploring the effect of photon PDFs and for tuning models of parton evolution and treatment of fragmentation and the proton remnant in DIS.

7.4 ZEUS_2001_S4815815

Dijet photoproduction analysis

Experiment: ZEUS (HERA Run I)

Spires ID: [4815815](#)

Status: UNVALIDATED

Authors:

- Jon Butterworth (jmb@hep.ucl.ac.uk);

References:

- Eur.Phys.J.C23:615,2002
- DESY 01/220
- hep-ex/0112029

Run details:

- HERA beam conditions: 820 GeV protons colliding with 27.5 GeV positrons
- Direct and resolved photoproduction of di-jets
- Leading jet $p_{\perp} \gtrsim 14$ GeV, second jet $p_{\perp} \gtrsim 11$ GeV
- Jet pseudorapidity $-1 < \eta < 2.4$

ZEUS photoproduction of jets from proton-positron collisions at beam energies of 820 GeV on 27.5 GeV. Photoproduction can either be direct, in which case the photon interacts directly with the parton, or resolved, in which case the photon acts as a source of quarks and gluons. A photon-proton centre of mass energy of between 134 GeV and 227 GeV is probed, with values of x_P , the fractional momentum of the partons inside the proton, predominantly in the region between 0.01 and 0.1. The fractional momentum of the partons from the photon, x_γ , is in the region 0.1 to 1. Jets are reconstructed in the range $-1 < |\eta| < 2.4$ using the kT algorithm with an R parameter of 1.0. The minimum p_{\perp} of the leading jet should be greater than 14 GeV, and at least one other jet must have $p_{\perp} \gtrsim 11$ GeV.

8. Monte Carlo analyses

8.1 MC_LHC_LEADINGJETS

Underlying event in leading jet events, extended to LHC

Experiment: NONE (LHC)

Spires ID: [NONE](#)

Status: UNVALIDATED

Authors:

- Andy Buckley [<andy.buckley@cern.ch>](mailto:andy.buckley@cern.ch);

References:

-

Run details:

- LHC: pp QCD interactions at 0.9, 10 or 14 TeV. Particles with $c\tau > 10$ mm should be set stable. Several p_{\perp} min cutoffs are probably required to fill the profile histograms.

Rick Field's measurement of the underlying event in leading jet events, extended to the LHC. As usual, the leading jet of the event defines an azimuthal toward/transverse/away decomposition, in this case the event is accepted within $|\eta| < 2$, as in the CDF 2008 version of the analysis. Since this isn't the Tevatron, I've chosen to use k_{\perp} rather than midpoint jets.

8.2 MC_TVT1960_ZJETS

Monte Carlo validation observables for $Z[e^+ e^-] + \text{jets}$ production at Tevatron Run II

Experiment: MC (Tevatron Run 2)

Spires ID: [NONE](#)

Status: NOT TO BE VALIDATED

Authors:

- Frank Siegert (frank.siegert@durham.ac.uk);

No references listed Run details:

- Tevatron Run 2 conditions:
- $p\bar{p} \rightarrow e^+ e^- + \text{jets}$ at 1960 GeV. * Needs mass cut on lepton pair to avoid photon singularity: min. range $66 < m_{ee} < 116$ GeV

Available observables

- * Z mass
- * p_\perp of jet 1-4
- * jet multiplicity
- * Delta eta (Z, jet1)
- * Delta R (jet2, jet3)
- * Differential jet rates $0 \rightarrow 1, 1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4$
- * Integrated 0-4 jet rates

9. Example analyses

9.1 EXAMPLE

A demo to show aspects of writing a Rivet analysis

Experiment: NONE (NONE)

Spires ID: [NONE](#)

Status: EXAMPLE

Authors:

- Andy Buckley [<andy.buckley@durham.ac.uk>](mailto:andy.buckley@durham.ac.uk);

No references listed Run details:

- All event types will be accepted.

This analysis is a demonstration of the Rivet analysis structure and functionality: booking histograms; the initialisation, analysis and finalisation phases; and a simple loop over event particles. It has no physical meaning, but can be used as a simple pedagogical template for writing real analyses.

9.2 EXAMPLETREE

Demonstrate filling a ROOT tree from a kT jets analysis

Experiment: NONE (NONE)

Spires ID: [NONE](#)

Status: EXAMPLE

Authors:

- Jon Butterworth [⟨jmb@hep.ucl.ac.uk⟩](mailto:jmb@hep.ucl.ac.uk);

No references listed Run details:

- All event types will be accepted.

This analysis is a demonstration of how Rivet can be used to produce ROOT data trees rather than Rivet’s own histograms. We don’t recommend this, since analyses written this way will not be accepted for inclusion into the Rivet library and hence will not contribute to standard MC validation and tuning studies. However, it may be useful for nascent private MC analyses if you are a ROOT fan.

Note that this example analysis does some things such as accessing parton level information, which are unphysical and also may be generator dependent. You should not use this method in your own analyses if you expect the results to be meaningfully comparable to data!

10. Misc. analyses

10.1 JADE_OPAL_2000_S4300807_35GEV

Jet rates in e+e- at JADE [35 GeV].

Experiment: JADE_OPAL (DESY PETRA)

Spires ID: [4300807](#)

Status: VALIDATED

Authors:

- Frank Siegert (frank.siegert@durham.ac.uk);

References:

- Eur.Phys.J.C17:19-51,2000
- arXiv: [hep-ex/0001055](#)

Run details:

- e+ e- collisions:
- e+ e- \rightarrow jet jet (+ jets) at 35 GeV. * no cuts needed

Differential and integrated jet rates for Durham and JADE jet algorithms at $\sqrt{s} = 35$.

10.2 JADE_OPAL_2000_S4300807_44GEV

Jet rates in e+e- at JADE [44 GeV].

Experiment: JADE_OPAL (DESY PETRA)

Spires ID: [4300807](#)

Status: VALIDATED

Authors:

- Frank Siegert (frank.siegert@durham.ac.uk);

References:

- Eur.Phys.J.C17:19-51,2000
- arXiv: [hep-ex/0001055](#)

Run details:

- e+ e- collisions:
- e+ e- \rightarrow jet jet (+ jets) at 44 GeV. * no cuts needed

Differential and integrated jet rates for Durham and JADE jet algorithms at $\sqrt{s} = 44$.

10.3 PDG_HADRON_MULTIPLICITIES

Hadron multiplicities in hadronic $e+e-$ events

Experiment: PDG (various)

Spires ID: [7857373](#)

Status: VALIDATED

Authors:

- Hendrik Hoeth (hendrik.hoeth@cern.ch);

References:

- Phys. Lett. B, 667, 1 (2008)

Run details:

- Hadronic events in $e + e-$ collisions

Hadron multiplicities in hadronic $e+e-$ events, taken from Review of Particle Properties 2008, table 40.1, page 355.

Average hadron multiplicities per hadronic $e + e-$ annihilation event at $\sqrt{s} \approx 10, 29-35, 91$, and $130-200$ GeV. The numbers are averages from various experiments. Correlations of the systematic uncertainties were considered for the calculation of the averages.

10.4 PDG_HADRON_MULTIPLICITIES_RATIOS

Ratios (w.r.t. π^+/π^-) of hadron multiplicities in hadronic e^+e^- events

Experiment: PDG (various)

Spires ID: [7857373](#)

Status: VALIDATED

Authors:

- Holger Schulz (holger.schulz@physik.hu-berlin.de);

References:

- Phys. Lett. B, 667, 1 (2008)

Run details:

- Hadronic events in e^+e^- collisions

Ratios (w.r.t. π^+/π^-) of hadron multiplicities in hadronic e^+e^- events, taken from Review of Particle Properties 2008, table 40.1, page 355.

Average hadron multiplicities per hadronic e^+e^- annihilation event at $\sqrt{s} \approx 10, 29\text{--}35, 91$, and $130\text{--}200$ GeV, normalised to the pion multiplicity. The numbers are averages from various experiments. Correlations of the systematic uncertainties were considered for the calculation of the averages.

10.5 STAR_2006_S6870392

Inclusive jet cross-section in pp at 200 GeV

Experiment: STAR (RHIC pp 200 GeV)

Spires ID: [6870392](#)

Status: VALIDATED

Authors:

- Hendrik Hoeth (hendrik.hoeth@cern.ch);

References:

- Phys. Rev. Lett. 97, 252001
- hep-ex/0608030

Run details:

- RHIC pp 200 GeV run conditions:
- pp at 200 GeV

Inclusive jet cross section as a function of p_{\perp} in pp collisions at $\sqrt{s} = 200$ GeV, measured by the STAR experiment at RHIC.

10.6 STAR_2008_S7993412

Di-hadron correlations in d-Au at 200 GeV

Experiment: STAR (RHIC d-Au 200 GeV)

Spires ID: [7993412](#)

Status: UNVALIDATED

Authors:

- Christine Nattrass (christine.nattrass@yale.edu);
- Hendrik Hoeth (hendrik.hoeth@cern.ch);

References:

- arXiv: [0809.5261](#)

Run details:

- RHIC d-Au 200 GeV run conditions:
- d-Au at 200 GeV (use pp Monte Carlo! See description.)

Correlation in η and ϕ between the charged hadron with the highest p_{\perp} (“trigger particle”) and the other charged hadrons in the event (“associated particles”). The data was collected in d-Au collisions at 200 GeV. Nevertheless, it is very proton-proton like and can therefore be compared to pp Monte Carlo (not for tuning, but for qualitative studies).

Part III

How Rivet works

Hopefully by now you’ve run Rivet a few times and got the hang of the command line interface and viewing the resulting analysis data files. Maybe you’ve got some ideas of analyses that you would like to see in Rivet’s library. If so, then you’ll need to know a little about Rivet’s internal workings before you can start coding: with any luck by the end of this section that won’t seem particularly intimidating.

The core objects in Rivet are “projections” and “analyses”. Hopefully “analyses” isn’t a surprise — that’s just the collection of routines that will make histograms to compare with reference data, and the only things that might differ there from experiences with HZTool are the new histogramming system and the fact that we’ve used some object orientation concepts to make life a bit easier. The meaning of “projections”, as applied to event analysis, will probably be less obvious. We’ll discuss them now.

11. Projections

The name “projection” is meant to evoke thoughts of projection operators, low-dimensional slices/views of high-dimensional spaces, and other things that might appeal to physicists who view the world through quantum-tinted lenses. A more mundane, but equally applicable, name would be “observable calculators”, but since that’s a long name, the things they return aren’t *necessarily* observable, and they all inherit from the `Projection` base class, we’ll stick to that name. It doesn’t take long to get used to using the name as a synonym for “calculator”, without being intimidated by ideas that they might be some sort of high-powered deep magic. 90% of them is simple and self-explanatory, as a peek under the bonnet of e.g. the all-important `FinalState` projection will reveal.

Projections can be relatively simple things like event shapes (i.e. scalar, vector or tensor quantities), or arbitrarily complex things like lossy or selective views of the event final state. Most users will see them attached to analyses by declarations in each analysis’ constructor, but they can also be recursively “nested” inside other projections² (provided there are no infinite loops in the nesting chain.) Calling a complex projection in an analysis may actually transparently execute many projections on each event.

11.1 Projection caching

Aside from semantic issues of how the class design assigns the process of analysing events, projections are important computationally because they live in a framework which automatically stores (“caches”) their results between events. This is a crucial feature for the long-term scalability of Rivet, as the previous experience with HZTool was that HERA

²Provided there are no dependency loops in the projection chains! Strictly, only acyclic graphs of projection dependencies are valid, but there is currently no code in Rivet that will attempt to verify this restriction.

validation code ran very slowly due to repeated calculation of the same k_{\perp} clustering algorithm (at that time notorious for scaling as the 3rd power of the number of particles.)

A concrete example may help in understanding how this works. Let's say we have two analyses which have the same run conditions, i.e. incoming beam types, beam energies, etc. Each also uses the thrust event shape measure to define a set of basis vectors for their analysis. For each event that gets passed to Rivet, whichever analysis gets called first will immediately (although maybe indirectly) call a **FinalState** projection to get a list of stable, physical particles (filtering out the intermediate and book-keeping entries in the HepMC event record). That FS projection is then "attached" to the event. Next, the first analysis will call a **Thrust** projection which internally uses the same final state projection to define the momentum vectors used in calculating the thrust. Once finished, the thrust projection will also be attached to the event.

So far, projections have offered no benefits. However, when the second analysis runs it will similarly try to apply its final state and thrust projections to the event. Rather than repeat the calculations, Rivet's infrastructure will detect that an equivalent calculation has already been run and will just return references to the already-run projections. Since projections can also contain and use other projections, this model allows some substantial computational savings, without the analysis author even needing to be particularly aware of what is going on.

Observant readers may have noticed a problem with all this projection caching cleverness: what if the final states aren't defined the same way? One might provide charged final state particles only, or the acceptances (defined in rapidity range and a IR p_{\perp} cutoff) might differ. Rivet handles this by making each projection provide a comparison operator which is used to decide whether the cached version is acceptable or if the calculation must be re-run with different settings. Because projections can be nested, applying a top-level projection to an event can spark off a cascade of comparisons, calculations and cache accesses, making use of existing results wherever possible.

11.2 Using projection caching

So far this is all theory — how does one actually use projections in Rivet? First, you should understand that projections, while semantically stored within each other, are actually all registered with a central **ProjectionHandler** object.³ The reason for this central registration is to ensure that all projections' lifespans are managed in a consistent way, and to protect projection and analysis authors from some technical subtleties in how C++ polymorphism works.

Inside the constructor of a **Projection** or **Analysis** class, you must call the **addProjection** function. This takes two arguments, the projection to be registered (by **const** reference), and a name. The name is local to the parent object, so you need not worry about name clashes between objects. A very important point is that the passed **Projection** is not the one that is actually centrally registered — that distinction belongs to a newly created heap

³As of version 1.1 onwards — previously, they were stored as class members inside other **Projection**s and **Analysis** classes.

object which is created within the `addProjection` method by means of the overloaded `Projection::clone()` method. Hence it is completely safe — and recommended — to use only local (stack) objects in `Projection` and `Analysis` constructors.



At this point, if you have rightly bought into C++ ideas like super-strong type-safety, this proliferation of dynamic casting may worry you: the compiler can't possibly check if a projection of the requested name has been registered, nor whether the downcast to the requested concrete type is legal. These are very legitimate concerns! In truth, we'd like to have this level of extra safety but in the past, when projections were held as members of `ProjectionApplier` classes rather than in the central `ProjectionHandler` repository, the benefits of the strong typing were outweighed by more serious and subtle bugs relating to projection lifetime and object "slicing". At least when the current approach goes wrong it will throw an unmissable runtime error every time that you run it (until it's fixed, of course!) rather than silently do the wrong thing, as was the previous behaviour. Our problems here are a microcosm of the perpetual language battle between strict and dynamic typing, runtime versus compile time errors. In practice, this manifests itself as a trade-off between the benefits of static type safety and the inconvenience of the type-system gymnastics that it engenders. We take some comfort from the number of very good programs have been and are still written in dynamically typed, interpreted languages like Python, where virtually all error checking (barring first-scan parsing errors) must be done at runtime. By pushing some checking to the domain of runtime errors, Rivet's code is (we believe) in practice safer, and certainly more clear and elegant. However, we believe that with runtime checking should come a culture of unit testing, which is not yet in place in Rivet. As a final thought, one reason for Rivet's internal complexity is that C++ is just not a very good language for this sort of thing: we are operating on the boundary between event generator codes, number crunching routines (including third party libraries like FastJet) and user routines. The former set unavoidably require native interfaces and benefit from static typing; the latter benefit from interface flexibility, fast prototyping and syntactic clarity. Maybe a future version of Rivet will break through the technical barriers to a hybrid approach and allow users to run compiled projections from interpreted analysis code. For now, however, we hope that our brand of "slightly less safe C++" will be a pleasant compromise.

12. Analyses

12.1 Writing a new analysis

This section provides a recipe that can be followed to write a new analysis using the Rivet projections.

Every analysis must inherit from `Rivet::Analysis` and, in addition to the constructor, must implement a minimum of three methods. Those methods are `init()`, `analyze(const`

`Rivet::Event&)` and `finalize()`, which are called once at the beginning of the analysis, once per event and once at the end of the analysis respectively.

The new analysis should include the header for the base analysis class plus whichever Rivet projections are to be used and should work under the `Rivet` namespace. The header for a new analysis named `UserAnalysis` that uses the `FinalState` projection might therefore start off looking like this:

```
#include "Rivet/Analysis.hh"

namespace Rivet {

    class UserAnalysis : public Analysis {
    public:
        UserAnalysis();
        void init();
        void analyze(const Event& event);
        void finalize();
    };

}
```

12.1.1 Analysis constructor

The constructor for the `UserAnalysis` class should add to the analysis all of the projections that will be used. Projections can be added to an analysis with a call to `addProjection(Projection, std::string)`, which takes as argument the projection to be added and a name by which that projection can later be referenced. For this example the `FinalState` projection is to be referenced by the string `"FS"` to provide access to all of the final state particles inside a detector pseudorapidity coverage of ± 5.0 . The syntax to create and add that projection inside the constructor for `UserAnalysis` is as follows:

```
Rivet::UserAnalysis() {
    const FinalState fs(-5.0, 5.0);
    addProjection(fs, "FS");
}
```

In addition to adding projections, the constructor may also impose certain requirements upon the events that the analysis will work with. A call to the `setBeams` method declares that the analysis may only be run on events with specific types of beam particles, for example adding the line

```
setBeams(PROTON, PROTON);
```

ensures that the analysis can only be run on events from proton-proton collisions. Other types of beam particles that may be used include `ANTIPROTON`, `ELECTRON`, `POSITRON`, `MUON`

and ALL. The later of these declares that the analysis is suitable for use with any type of collision and is the default.

Some analyses need to know the interaction cross section that was generated by the Monte Carlo generator, typically in order to normalise histograms. Depending on the Monte Carlo that is used and its interface to Rivet, the cross section may or may not be known. An analysis can therefore declare at the beginning of a run that it will need the cross section information during the finalisation stages. Such a declaration can be used to prevent what would otherwise be fruitless analyses from running. An analysis sets itself as requiring the cross section by calling inside the constructor

```
setNeedsCrossSection(true);
```

In the absence of this call the default is to assume that the analysis does not need to know the cross section.

12.2 Histogramming

Rivet's histogramming uses the AIDA interfaces, composed of abstract classes `IHistogram1D`, `IProfile1D`, `IDataPointSet` etc. which are built by a factories system. Since it's our feeling that much of the factory infrastructure constitutes an abstraction overload, we provide histogram booking functions as part of the `Analysis` class, so that in the `init` method of your analysis you should book histograms with function calls like:

```
void MyAnalysis::init() {
    _h_one = bookHistogram1D(2,1,1, "Title 2", "x label", "y label");
    _h_two = bookProfile1D(3,1,2, "Title 2", "x label", "y label");
    _h_three = bookHistogram1D("d00-x00-y00", "Title",
                              "x label", "y label", 50, 0.0, 1.0);
}
```

Here the first two bookings have a rather cryptic 3-integer sequence as the first arguments. This is the recommended scheme, as it makes use of the exported data files from HepData, in which 1D histograms are constructed from a combination of x and y axes in a dataset d , corresponding to names of the form $d\langle d \rangle - x\langle x \rangle - y\langle y \rangle$. This auto-booking of histograms saves you from having to copy out reams of bin edges and values into your code, and makes sure that any data fixes in HepData are easily propagated to Rivet. The third booking is for a histogram for which there is no such HepData entry: it uses the usual scheme of specifying the name, number of bins and the min/max x -axis limits manually.

Filling the histograms is done in the `MyAnalysis::analyse()` function. Remember to specify the event weight as you fill:

```
void MyAnalysis::analyze(const Event& e) {
    [projections, cuts, etc.]
    ...
    _h_one->fill(pT, event.weight());
    _h_two->fill(pT, Nch, event.weight());
}
```

```

    _h_three->fill(fabs(eta), event.weight());
}

```

Finally, histogram normalisations, scalings, divisions etc. are done in the `MyAnalysis::finalize()` method. For normalisations and scalings you will find appropriate convenience methods `Analysis::normalize(histo, norm)` and `Analysis::scale(histo, scalefactor)`. Many analyses need to be scaled to the generator cross-section, with the number of event weights to pass cuts being included in the normalisation factor: for this you will have to track the passed-cuts weight sum yourself via a member variable, but the analysis class provides `Analysis::crossSection()` and `Analysis::sumOfWeights()` methods to access the pre-cuts cross-section and weight sum respectively.

12.3 Pluggable analyses

Rivet's standard analyses are not actually built into the main `libRivet` library: they are loaded dynamically at runtime as an analysis *plugin library*. While you don't need to worry too much about the technicalities of this, it does mean that you can similarly write analyses of your own, compile them into a similar plugin library and run them from `rivet` without ever having to modify any of the main Rivet sources or build system. This means that you can write and run your own analyses with a system-installed copy of Rivet, and not have to re-patch the main library when a newer version comes out (although chances are you will have to recompile, since the binary interface usually change between releases.)

You will find an example plugin analysis in the `plugindemo` directory in the Rivet source directory, with a corresponding `Makefile`. To understand the plugin system better, you should check out the documentation in the wiki on the Rivet website: <http://projects.hepforge.org/rivet/trac/wiki/>

Part IV

How Rivet *really* works

In time this will be the place to look for all the nitty gritty on what Rivet is doing internally. Not very many people need to know that, and the few that do currently don't need a manual for it!

13. Projection caching

★TODO

13.1 Writing a Projection comparison operator

★TODO

Part V

Appendices

A. Typical `agile-runmc` commands

- **Simple run:** `agile-runmc Herwig:6510 -P lep1.params --beams=LEP:91.2 -n 1000` will use the Fortran Herwig 6.5.10 generator (the `-g` option switch) to generate 1000 events (the `-n` switch) in LEP1 mode, i.e. e^+e^- collisions at $\sqrt{s} = 91.2$ GeV.
- **Parameter changes:** `agile-runmc Pythia6:418 --beams=LEP:91.2 -n 1000 \ -P myrun.params -p "PARJ(82)=5.27"` will generate 1000 events using the Fortran Pythia 6.4.18 generator, again in LEP1 mode. The `-P` switch is actually the way of specifying a parameters file, with one parameter per line in the format “ $\langle key \rangle \langle value \rangle$ ”: in this case, the file `lep1.params` is loaded from the $\langle installdir \rangle / \text{share} / \text{AGILE}$ directory, if it isn’t first found in the current directory. The `-p` (lower-case) switch is used to change a named generator parameter, here Pythia’s `PARJ(82)`, which sets the parton shower cutoff scale. Being able to change parameters on the command line is useful for scanning parameter ranges from a shell loop, or rapid testing of parameter values without needing to write a parameters file for use with `-P`.
- **Writing out HepMC events:** `agile-runmc Pythia6:418 --beams=LHC:14TeV -n 50 -o out.hepmc -R` will generate 50 LHC events with Pythia. The `-o` switch is being used here to tell `agile-runmc` to write the generated events to the `out.hepmc` file. This file will be a plain text dump of the HepMC event records in the standard HepMC format. Use of filename “`-`” will result in the event stream being written to standard output (i.e. dumping to the terminal).

Part VI

Bibliography

References

- [1] DELPHI Collaboration, P. Abreu *et al.*, Z. Phys. **C73**, 11 (1996).