Rivet user manual

version 1.2.0

Andy Buckley

PPE Group, School of Physics, University of Edinburgh, UK.
E-mail: andy.buckley@ed.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

IPPP, Durham University, UK.
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

Holger Schulz

Institut für Physik, Berlin Humboldt University, Germany.
E-mail: holger.schulz@physik.hu-berlin.de

Jan Eike von Seggern

Institut für Physik, Berlin Humboldt University, Germany.
E-mail: vseggern@physik.hu-berlin.de

Frank Siegert

IPPP, Durham University, UK &
HEP Group, Dept. of Physics and Astronomy, UCL, London, UK.
E-mail: frank.siegert@durham.ac.uk

Lars Sonnenschein

RWTH Aachen University, III. Phys. Inst. A, 52056 Aachen, Germany.
E-mail: Lars.Sonnenschein@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

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 for high energy collider experiments, 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

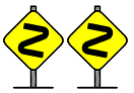
As is normal in computer user manuals, the typography in this manual is used to indicate whether we are describing source code elements, commands to be run in a terminal, the output of a command etc.

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*⟩`)`.

Dangerous bend

Following the example of Donald Knuth in his books on TEX, 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.



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 present 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.

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.

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
$ chmod +x rivet-bootstrap
```

Now run it to get some help: `$ ./rivet-bootstrap --help`

Now to actually do the install: for example, to bootstrap Rivet and AGILe to the install area specified as the prefix argument, run this:

`$ ./rivet-bootstrap --install-agile --prefix=`⟨*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[1], LHAPDF[2], FastJet[3] and GSL libraries from the LCG software area. Either way, finally the bootstrap script will write out a file containing the environment settings which will make the system useable. You can source this file, e.g. `source rivetenv.sh` to make your current shell ready-to-go for a Rivet run (use `rivetenv.csh` if you are a C shell user).

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, source the setup scripts as above, if you've not already done so, and 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[4] 6.4.13 generator. You're on your way! If no generators are

listed, you probaby 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[5] or Herwig++[6] which provides its own native way to make HepMC output, or a generator like Cascade[7] 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 `agile-runmc` 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 by Genser 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 names[1]!

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

```
$ . ⟨localdir⟩/share/Rivet/rivet-completion
$ . ⟨localdir⟩/share/AGILe/agile-completion
```

(if you are using the setup script `rivetenv.sh` this is automatically done for you). 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.

[1] Standard 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.

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 fifo.hepmc` will run the Rivet `DELPHI_1996_S3430090` [8] analysis on the events in the `fifo.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 fifo.hepmc -H foo` will read all the events in the `fifo.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 fifo.hepmc -A -l Rivet.Analysis=DEBUG \ -l Rivet.Projection=DEBUG -l Rivet.Projection.FinalState=TRACE \ -l NEvt=WARN` will 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 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[9]. Don't worry: a few months of therapy can work wonders. 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 Chopping histograms

In some cases you don't want to keep the complete histograms produced by Rivet. For generator tuning purposes, for example, you want to get rid of the bins you already know your generator is incapable of describing. You can use the script `rivet-chopbins` to specify those bin-ranges you want to keep individually for each histogram in a Rivet output-file. The bin-ranges have to be specified using the corresponding x-values of that histogram. The usage is very simple. You can specify bin ranges of histograms to keep on the command-line via the `-b` switch, which can be given multiple times, e.g.
`rivet-chopbins -b /CDF_2001_S4751469/d03-x01-y01:5:13 Rivet.aida`
will chop all bins with $x < 5$ and $x > 13$ from the histogram `/CDF_2001_S4751469/d03-x01-y01:5:13` in the file `Rivet.aida`. (In this particular case, x would be a leading jet $p_\perp$.)

4.3 Normalising histograms

Sometimes you want to use histograms normalised to, e.g., the generator cross-section or the area of a reference-data histogram. The script `rivet-rescale` was designed for these purposes. The usage is the following:
`rivet-rescale -O observables -r RIVETDATA -o normalised Rivet.aida`
By default, the normalised histograms are written to file in the AIDA-XML format. You can also give the `-f` switch on the command line to produce flat histograms.

Normalising to reference data You will need an output-file of Rivet, `Rivet.aida`, a folder that contains the reference-data histograms (e.g. `rivet-config --datadir`) and optionally, a text-file, `observables` that contains the names of the histograms you would like to normalise - those not given in the file will remain un-normalised. These are examples of how your `observables` file might look like:

```
/CDF_2000_S4155203/d01-x01-y01
```

If a histogram `/CDF_2000_S4155203/d01-x01-y01` is found in one of the reference-data files in the folder specified via the `-r` switch, then this will result in a histogram `/CDF_2000_S4155203/d01-x01-y01` being normalised to the area of the corresponding reference-data histogram. You can further specify a certain range of bins to normalise:

```
/CDF_2000_S4155203/d01-x01-y01:2:35
```

will chop off the bins with $x < 2$ and $x > 35$ of both, the histogram in your `Rivet.aida` and the reference-data histogram. The remaining MC histogram is then normalised to the remaining area of the reference-data histogram.

Normalising to arbitrary areas In the file `observables` you can further specify an arbitrary number, e.g. a generator cross-section, as follows:

```
/CDF_2000_S4155203/d01-x01-y01  1.0
```

will result in the histogram `/CDF_2000_S4155203/d01-x01-y01` being normalised to 1.0, and

```
/CDF_2000_S4155203/d01-x01-y01:2:35  1.0
```

will chop off the bins with $x < 2$ and $x > 35$ of the histogram `/CDF_2000_S4155203/d01-x01-y01` first and normalise the remaining histogram to one.

4.4 Plotting and comparing data

Rivet comes with three commands — `rivet-mkhtml`, `compare-histos` and `make-plots` — for comparing and plotting data files. These commands produce nice comparison plots of publication quality from the AIDA format text files.

The high level program `rivet-mkhtml` will automatically create a plot webpage from the given AIDA files. It searches for reference data automatically and uses the other two commands internally. Example: `$ rivet-mkhtml withUE.aida:'With UE' withoutUE:'Without UE'` The strings after the ":" are specifying ID strings to appear in the plot legends.

You can also run the other two commands separately. `compare-histos` will accept a number of AIDA files as input (ending in `.aida`), identify which plots are available in them, and combine the MC and reference plots appropriately into a set of plot data files ending with `.dat`. 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 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.

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_S3196992

Measurement of the quark to photon fragmentation function
Experiment: ALEPH (LEP Run 1)
Spires ID: 3196992
Status: VALIDATED
Authors:

- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- Z.Phys.C69:365-378,1996
- DOI: 10.1007/s002880050037

Run details:

- $e^+e^- \to$ jets with π and η decays turned off.

Earlier measurements at LEP of isolated hard photons in hadronic Z decays, attributed to radiation from primary quark pairs, have been extended in the ALEPH experiment to include hard photon production inside hadron jets. Events are selected where all particles combine democratically to form hadron jets, one of which contains a photon with a fractional energy $z > 0.7$. After statistical subtraction of non-prompt photons, the quark-to-photon fragmentation function, $D(z)$, is extracted directly from the measured 2-jet rate.

5.3 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.

5.4 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.

5.5 DELPHI_1996_S3430090

Delphi MC tuning on event shapes and identified particles.
Experiment: DELPHI (LEP 1)
Spires ID: 3430090
Status: VALIDATED
Authors:

- Andy Buckley ⟨andy.buckley@durham.ac.uk⟩;
- Hendrik Hoeth ⟨hendrik.hoeth@cern.ch⟩;

References:

- Z.Phys.C73:11-60,1996
- DOI: 10.1007/s002880050295

Run details:

- $\sqrt{s} = 91.2$ GeV, $e^+e^- \to Z^0$ 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.

5.6 DELPHI_2002_069_CONF_603

Study of the b-quark fragmentation function at LEP 1
Experiment: DELPHI (LEP 1)
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 $f(x_B^{\text{prim}})$.

5.7 JADE_OPAL_2000_S4300807

Jet rates in e+e- at JADE [35–44 GeV] and OPAL [91–189 GeV].
Experiment: JADE_OPAL (PETRA and LEP)
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:

- $e^+e^- \to$ jet jet (+ jets)

Differential and integrated jet rates for Durham and JADE jet algorithms.

5.8 OPAL_1998_S3780481

Measurements of flavor dependent fragmentation functions in $Z^0 \to q\bar{q}$ events
Experiment: OPAL (LEP 1)
Spires ID: 3780481
Status: VALIDATED
Authors:

- Hendrik Hoeth ⟨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 ($\sqrt{s} = 91.2$ 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.

5.9 OPAL_2004_S6132243

Event shape distributions and moments in e+ e- → hadrons at 91-209 GeV
Experiment: OPAL (LEP 1 & 2)
Spires ID: 6132243
Status: VALIDATED
Authors:

- Andy Buckley ⟨andy.buckley@cern.ch⟩;

References:

- Eur.Phys.J.C40:287-316,2005
- hep-ex/0503051

Run details:

- Hadronic e+ e- events at 4 representative energies (91, 133, 177, 197). Runs with $\sqrt{s}$ above the Z mass need to have ISR suppressed, since the data has been corrected to remove radiative return to the Z.

Measurement of e+ e- event shape variable distributions and their 1st to 5th moments in LEP running from the Z pole to the highest LEP 2 energy of 209 GeV.

6. Tevatron analyses

6.1 CDF_1988_S1865951

CDF transverse momentum distributions at 630 GeV and 1800 GeV.
Experiment: CDF (Tevatron Run I)
Spires ID: 1865951
Status: VALIDATED
Authors:

- Christophe Vaillant ⟨c.l.j.j.vaillant@durham.ac.uk⟩;
- Andy Buckley ⟨andy.buckley@cern.ch⟩;

References:

- Phys.Rev.Lett.61:1819,1988
- DOI: 10.1103/PhysRevLett.61.1819

Run details:

- QCD min bias events at $\sqrt{s} = 630$ GeV and 1800 GeV, $|\eta| < 1.0$.

Transverse momentum distributions at 630 GeV and 1800 GeV based on data from the CDF experiment at the Tevatron collider.

6.2 CDF_1990_S2089246

CDF pseudorapidity distributions at 630 and 1800 GeV
Experiment: CDF (Tevatron Run 0)
Spires ID: 2089246
Status: VALIDATED
Authors:

- Andy Buckley ⟨andy.buckley@cern.ch⟩;

References:

- Phys.Rev.D41:2330,1990
- DOI: 10.1103/PhysRevD.41.2330

Run details:

- QCD min bias events at $\sqrt{s} = 630$ and 1800 GeV. Particles with $c\tau > 10$mm should be set stable.

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.

6.3 CDF_1998_S3618439

Differential cross-section for events with large total transverse energy
Experiment: CDF (Tevatron Run 1)
Spires ID: 3618439
Status: VALIDATED
Authors:

- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- Phys.Rev.Lett.80:3461-3466,1998
- 10.1103/PhysRevLett.80.3461

Run details:

- QCD events at Tevatron with $\sqrt{s} = 1.8$ TeV without MPI.

Measurement of the differential cross section $\mathrm{d}\sigma/\mathrm{d}E_\perp^j$ for the production of multijet events in $p\bar{p}$ collisions where the sum is over all jets with transverse energy $E_\perp^j > E_\perp^{\min}$.

6.4 CDF_2000_S4155203

Z $p_\perp$ measurement in CDF Z $\to e^+e^-$ events
Experiment: CDF (Tevatron Run 1)
Spires ID: 4155203
Status: VALIDATED
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:

- ppbar collisions at 1800 GeV. $Z/\gamma*$ Drell-Yan events with e^+e^- decay mode only.

Measurement of transverse momentum and total cross section of e^+e^- pairs in the Z-boson region of 66 GeV/c^2 $< m_{ee} <$ 116 GeV/c^2 from pbar-p collisions at $\sqrt{s} = 1.8$ 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. This analysis is subject to ambiguities in the experimental Z $p_\perp$ definition, since the Rivet implementation reconstructs the Z momentum from the dileptonpair with finite cones for photon brem summation, rather than YFS/shower unfolding or non-portable direct use of the Z in the event record.

6.5 CDF_2000_S4266730

Differential Dijet Mass Cross Section
Experiment: CDF (Tevatron Run 1)
Spires ID: 4266730
Status: VALIDATED
Authors:

- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- Phys.Rev.D61:091101,2000
- DOI: 10.1103/PhysRevD.61.091101
- arXiv: hep-ex/9912022

Run details:

- Dijet events at Tevatron with $\sqrt{s} = 1.8$ TeV

Measurement of the cross section for production of two or more jets as a function of dijet mass in the range 180 to 1000 GeV. It is based on an integrated luminosity of 86pb^{-1}.

6.6 CDF_2001_S4517016

Two jet triply-differential cross-section
Experiment: CDF (Tevatron Run 1)
Spires ID: 4517016
Status: VALIDATED
Authors:

- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- Phys.Rev.D64:012001,2001
- DOI: 10.1103/PhysRevD.64.012001
- arXiv: hep-ex/0012013

Run details:

- Dijet events at Tevatron with $\sqrt{s} = 1.8$ TeV

A measurement of the two-jet differential cross section, $\mathrm{d}^3\sigma/\mathrm{d}E_T\,\mathrm{d}\eta_1\,\mathrm{d}\eta_2$, based on an integrated luminosity of $86\mathrm{pb}^{-1}$. The differential cross section is measured as a function of the transverse energy, $E_\perp$, of a jet in the pseudorapidity region $0.1 < |\eta_1| < 0.7$ for four different pseudorapidity bins of a second jet restricted to $0.1 < |\eta_2| < 3.0$.

6.7 CDF_2001_S4563131

Inclusive jet cross section
Experiment: CDF (Tevatron Run 1)
Spires ID: 4563131
Status: VALIDATED
Authors:

- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- Phys.Rev.D64:032001,2001
- DOI: 10.1103/PhysRevD.64.032001
- arXiv: hep-ph/0102074

Run details:

- Dijet events at Tevatron with $\sqrt{s} = 1.8$ TeV

Measurement of the inclusive jet cross section for jet transverse energies from 40 to 465 GeV in the pseudo-rapidity range $0.1 < |\eta| < 0.7$. The results are based on 87 pb^{-1} of data.

6.8 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⟩;

References:

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

Run details:

- ppbar QCD interactions at 1800 GeV. The leading jet is binned from 0–49 GeV, and histos can usually can be filled with a single generator run without kinematic sub-samples.

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 $|\phi| \in [60°, 120°]$ 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.

6.9 CDF_2002_S4796047

CDF Run 1 charged multiplicity measurement
Experiment: CDF (Tevatron Run 1)
Spires ID: 4796047
Status: VALIDATED
Authors:

- Hendrik Hoeth ⟨hendrik.hoeth@cern.ch⟩;

References:

- Phys.Rev.D65:072005,2002
- DOI: 10.1103/PhysRevD.65.072005

Run details:

- QCD events at $\sqrt{s} = 630$ and 1800 GeV.

A study of ppbar 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.

6.10 CDF_2004_S5839831

Transverse cone and 'Swiss cheese' underlying event studies
Experiment: CDF (Tevatron Run 2)
Spires ID: 5839831
Status: VALIDATED
Authors:

- Andy Buckley ⟨andy.buckley@durham.ac.uk⟩;

References:

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

Run details:

- QCD events at $\sqrt{s} = 630$ & 1800 GeV. Several $p_\perp^{\text{min}}$ 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! The $p_\perp$ plots are normalised to raw number of events. The min bias data have not been reproduced by MC, and are not recommended for tuning.

6.11 CDF_2005_S6080774

Differential cross sections for prompt diphoton production
Experiment: CDF (Tevatron Run 2)
Spires ID: 6080774
Status: VALIDATED
Authors:

- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- Phys. Rev. Lett. 95, 022003
- DOI: 10.1103/PhysRevLett.95.022003
- arXiv: hep-ex/0412050

Run details:

- $p\bar{p} \to \gamma\gamma$ [+ jets] at 1960 GeV. The analysis uses photons with $p_\perp$ larger then 13 GeV. To allow for shifts in the shower, the ME cut on the transverse photon momentum shouldn't be too hard, e.g. 5 GeV.

Measurement of the cross section of prompt diphoton production in $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV using a data sample of 207 pb^{-1} as a function of the diphoton mass, the transverse momentum of the diphoton system, and the azimuthal angle between the two photons.

6.12 CDF_2005_S6217184

CDF Run II jet shape analysis
Experiment: CDF (Tevatron Run 2)
Spires ID: 6217184
Status: VALIDATED
Authors:

- Lars Sonnenschein ⟨Lars.Sonnenschein@cern.ch⟩;
- Andy Buckley ⟨andy.buckley@cern.ch⟩;

References:

- Phys.Rev.D71:112002,2005
- DOI: 10.1103/PhysRevD.71.112002
- arXiv: hep-ex/0505013

Run details:

- QCD events at $\sqrt{s} = 1960$ GeV. Jet $p_\perp^{\text{min}}$ 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.

6.13 CDF_2006_S6450792

Inclusive jet cross section differential in $p_\perp$
Experiment: CDF (Tevatron Run 2)
Spires ID: 6450792
Status: VALIDATED
Authors:

- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- Phys.Rev.D74:071103,2006
- DOI: 10.1103/PhysRevD.74.071103
- arXiv: hep-ex/0512020

Run details:

- ppbar → jets at 1960 GeV

Measurement of the inclusive jet cross section in ppbar interactions at $\sqrt{s} = 1.96$ TeV using 385 pb^{-1} of data. The data cover the jet transverse momentum range from 61 to 620 GeV/c in $0.1 < |y| < 0.7$. This analysis has been updated with more data in more rapidity bins in CDF_2008_S7828950.

6.14 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⟩;

No references listed
Run details:

- 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. $p_\perp^{\min} = 0$ (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.

6.15 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:

- ppbar collisions at 1960 GeV. * Drell-Yan events with $Z/\gamma* \to ee$ and $Z/\gamma* \to \mu\mu$. * 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 and Rick Field's measurement of the underlying event in Drell-Yan events. Z → ee and Z → $\mu\mu$ 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).

6.16 CDF_2008_S7540469

Measurement of differential Z/γ^* + 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:

- $p\bar{p} \to e^+e^-$ + jets at 1960 GeV. Needs mass cut on lepton pair to avoid photon singularity, looser than $66 < m_{ee} < 116$

Cross sections as a function of jet transverse momentum in 1 and 2 jet events, and jet multiplicity in ppbar 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.

6.17 CDF_2008_S7828950

CDF Run II inclusive jet cross-section using the Midpoint algorithm
Experiment: CDF (Tevatron Run 2)
Spires ID: 7828950
Status: VALIDATED
Authors:

- Craig Group ⟨group@fnal.gov⟩;
- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- arXiv: 0807.2204
- Phys.Rev.D78:052006,2008

Run details:

- Requires $2 \to 2$ QCD scattering processes. The minimum jet $E_\perp$ 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 $E_\perp$, for $E_\perp > 62$ 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.

6.18 CDF_2008_S8093652

Dijet mass spectrum
Experiment: CDF (Tevatron Run 2)
Spires ID: 8093652
Status: VALIDATED
Authors:

- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- arXiv: 0812.4036

Run details:

- $p\bar{p} \to$ jets at 1960 GeV

Dijet mass spectrum from 0.2 TeV to 1.4 TeV in $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV, based on an integrated luminosity of 1.13 fb^{-1}.

6.19 CDF_2009_S8233977

CDF Run 2 min bias cross-section analysis
Experiment: CDF (Tevatron Run 2)
Spires ID: 8233977
Status: VALIDATED
Authors:

- Hendrik Hoeth ⟨hendrik.hoeth@cern.ch⟩;
- Niccolo' Moggi ⟨niccolo.moggi@bo.infn.it⟩;

References:

- Phys.Rev.D79:112005,2009
- DOI: 10.1103/PhysRevD.79.112005
- arXiv: 0904.1098

Run details:

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

Niccolo Moggi's min bias 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.

6.20 CDF_2009_S8383952

Z rapidity measurement
Experiment: CDF (Tevatron Run 2)
Spires ID: 8383952
Status: VALIDATED
Authors:

- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- arXiv: 0908.3914

Run details:

- $p\bar{p} \to e^+e^-$ + jets at 1960 GeV. Needs mass cut on lepton pair to avoid photon singularity, looser than $66 < m_{ee} < 116$ GeV

CDF measurement of the total cross section and rapidity distribution, $\mathrm{d}\sigma/\mathrm{d}y$, for $q\bar{q} \to \gamma^*/Z \to e^+e^-$ events in the Z boson mass region ($66 < M_{ee} < 116$ GeV/c^2) produced in $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV with 2.1 fb^{-1} of integrated luminosity.

6.21 CDF_2009_S8436959

Measurement of the Inclusive Isolated Prompt Photon Cross Section
Experiment: CDF (Tevatron Run 2)
Spires ID: 8436959
Status: VALIDATED
Authors:

- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- arXiv: 0910.3623

Run details:

- γ + jet processes in ppbar collisions at $\sqrt{s} = 1960$ GeV. Minimum $p_\perp$ cut on the photon in the analysis is 30 GeV.

A measurement of the cross section for the inclusive production of isolated photons. The measurement covers the pseudorapidity region $|\eta^\gamma| < 1.0$ and the transverse energy range $E_T^\gamma > 30$ GeV and is based on 2.5 fb^{-1} of integrated luminosity. The cross section is measured differential in $E_\perp(\gamma)$.

6.22 D0_2001_S4674421

Tevatron Run I differential W/Z boson cross-section analysis
Experiment: D0 (Tevatron Run 1)
Spires ID: 4674421
Status: VALIDATED
Authors:

- Lars Sonnenschein ⟨Lars.Sonnenschein@cern.ch⟩;

References:

- Phys.Lett.B517:299-308,2001
- DOI: 10.1016/S0370-2693(01)01020-6
- arXiv: hep-ex/0107012v2

Run details:

- W/Z events with decays to first generation leptons, in ppbar collisions at $\sqrt{s} = 1800$ GeV

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.

6.23 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:

- QCD events in ppbar interactions at $\sqrt{s} = 1960$ 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.

6.24 D0_2006_S6438750

Inclusive isolated photon cross-section, differential in $p_\perp$ (gamma)
Experiment: D0 (Tevatron Run 2)
Spires ID: 6438750
Status: VALIDATED
Authors:

- Andy Buckley ⟨andy.buckley@durham.ac.uk⟩;
- Gavin Hesketh ⟨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
- arXiv: hep-ex/0511054

Run details:

- ppbar collisions at $\sqrt{s} = 1960$ GeV. Requires gamma + jet (q,qbar,g) hard processes, which for Pythia 6 means MSEL=10 for with MSUB indices 14, 18, 29, 114, 115 enabled.

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 qbar → gamma g) and quark-gluon Compton scattering (q g → 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

6.25 D0_2007_S7075677

Z/gamma* + X cross-section shape, differential in y(Z)
Experiment: D0 (Tevatron Run 2)
Spires ID: 7075677
Status: VALIDATED
Authors:

- Andy Buckley ⟨andy.buckley@durham.ac.uk⟩;
- Gavin Hesketh ⟨ghesketh@fnal.gov⟩;
- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- Phys.Rev.D76:012003,2007
- arXiv: hep-ex/0702025

Run details:

- Drell-Yan $p\bar{p} \to Z/\gamma^* +$ jets events at $\sqrt{s} = 1960$ GeV. Needs mass cut on lepton pair to avoid photon singularity, looser than $71 < m_{ee} < 111$ GeV

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

6.26 D0_2008_S6879055

Measurement of the ratio sigma(Z/γ^* + n jets)/sigma(Z/γ^*)
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:

- $p\bar{p} \to e^+e^-$ + jets at 1960 GeV. Needs mass cut on lepton pair to avoid photon singularity, looser than $75 < m_{ee} < 105$ GeV.

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

6.27 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⟩;
- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- arXiv: 0712.0803

Run details:

- $p\bar{p} \to e^+e^-$ + jets at 1960 GeV. Needs mass cut on lepton pair to avoid photon singularity, looser than $40 < m_{ee} < 200$ 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}.

6.28 D0_2008_S7662670

Measurement of D0 Run II differential jet cross sections
Experiment: D0 (Tevatron Run 2)
Spires ID: 7662670
Status: VALIDATED
Authors:

- Andy Buckley ⟨andy.buckley@durham.ac.uk⟩;
- Gavin Hesketh ⟨gavin.hesketh@cern.ch⟩;

References:

- Phys.Rev.Lett.101:062001,2008
- DOI: 10.1103/PhysRevLett.101.062001
- arXiv: 0802.2400v3

Run details:

- QCD events at $\sqrt{s} = 1960$ GeV. A $p_\perp^{\min}$ cut is probably necessary since the 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.

6.29 D0_2008_S7719523

Isolated γ + jet cross-sections, differential in $p_\perp$ (γ) for various y bins
Experiment: D0 (Tevatron Run 2)
Spires ID: 7719523
Status: VALIDATED
Authors:

- Andy Buckley ⟨andy.buckley@durham.ac.uk⟩;
- Gavin Hesketh ⟨gavin.hesketh@cern.ch⟩;

References:

- Phys.Lett.B666:435-445,2008
- DOI: 10.1016/j.physletb.2008.06.076
- arXiv: 0804.1107v2

Run details:

- Produce only gamma + jet (q,qbar,g) hard processes (for Pythia 6, this means MSEL=10 and MSUB indices 14, 29 & 115 enabled). The lowest bin edge is at 30 GeV, so a kinematic $p_\perp^{\min}$ cut is probably required to fill the histograms.

The process p pbar → photon + 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 $1.5 < |y_{\text{jet}}| < 2.5$ rapidity intervals with $p_\perp^{\text{jet}} > 15$ GeV. The differential cross section $\mathrm{d}^3\sigma/\mathrm{d}p_\perp^\gamma\,\mathrm{d}y_\gamma\mathrm{d}y_{\text{jet}}$ 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.

6.30 D0_2008_S7837160

Measurement of W charge asymmetry from D0 Run II
Experiment: D0 (Tevatron Run 2)
Spires ID: 7837160
Status: VALIDATED
Authors:

- Andy Buckley ⟨andy.buckley@durham.ac.uk⟩;
- Gavin Hesketh ⟨gavin.hesketh@cern.ch⟩;

References:

- Phys.Rev.Lett.101:211801,2008
- DOI: 10.1103/PhysRevLett.101.211801
- arXiv: 0807.3367v1

Run details:

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

Measurement of the electron charge asymmetry in $p\bar{p} \to W + X \to e\nu_e + X$ 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.

6.31 D0_2008_S7863608

Measurement of differential Z/gamma* + jet + X cross sections
Experiment: D0 (Tevatron Run 2)
Spires ID: 7863608
Status: VALIDATED
Authors:

- Andy Buckley ⟨andy.buckley@durham.ac.uk⟩;
- Gavin Hesketh ⟨gavin.hesketh@fnal.gov⟩;
- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- arXiv: 0808.1296

Run details:

- $p\bar{p} \to \mu^+\mu^-$ + jets at 1960 GeV. Needs mass cut on lepton pair to avoid photon singularity, looser than $65 < m_{ee} < 115$ GeV.

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.

6.32 D0_2009_S8202443

Z/gamma* + jet + X cross sections differential in $p_\perp$ (jet 1,2,3)
Experiment: D0 (Tevatron Run 2)
Spires ID: 8202443
Status: VALIDATED
Authors:

- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- arXiv: 0903.1748

Run details:

- $p\bar{p} \to e^+e^-$ + jets at 1960 GeV. Needs mass cut on lepton pair to avoid photon singularity, looser than $65 < m_{ee} < 115$ GeV.

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

6.33 D0_2009_S8320160

Dijet angular distributions
Experiment: D0 (Tevatron Run 2)
Spires ID: 8320160
Status: VALIDATED
Authors:

- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- arXiv: 0906.4819

Run details:

- $p\bar{p} \to$ jets at 1960 GeV

Dijet angular distributions in different bins of dijet mass from 0.25 TeV to above 1.1 TeV in $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV, based on an integrated luminosity of 0.7 fb^{-1}.

6.34 D0_2009_S8349509

Z+jets angular distributions
Experiment: D0 (Tevatron Run 2)
Spires ID: 8349509
Status: VALIDATED
Authors:

- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- arXiv: 0907.4286

Run details:

- $p\bar{p} \to \mu^+\mu^-$ + jets at 1960 GeV. Needs mass cut on lepton pair to avoid photon singularity, looser than $65 < m_{ee} < 115$ GeV.

First measurements at a hadron collider of differential cross sections for Z+jet+X production in $\Delta\phi(Z,j)$, $|\Delta y(Z,j)|$ and $|y_{\text{boost}}(Z,j)|$. Vector boson production in association with jets is an excellent probe of QCD and constitutes the main background to many small cross section processes, such as associated Higgs production. These measurements are crucial tests of the predictions of perturbative QCD and current event generators, which have varied success in describing the data. Using these measurements as inputs in tuning event generators will increase the experimental sensitivity to rare signals.

6.35 D0_2010_S8566488

Dijet invariant mass
Experiment: D0 (Tevatron Run 2)
Spires ID: 8566488
Status: VALIDATED
Authors:

- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

References:

- arXiv: 1002.4594

Run details:

- $p\bar{p} \to$ jets at 1960 GeV. Analysis needs two hard jets above 40 GeV.

The inclusive dijet production double differential cross section as a function of the dijet invariant mass and of the largest absolute rapidity ($|y|_{\max}$) of the two jets with the largest transverse momentum in an event is measured using 0.7 fb^{-1} of data. The measurement is performed in six rapidity regions up to $|y|_{\max} = 2.4$.

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

Run details:

- e^-p / e^+p deep inelastic scattering, 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_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
- arXiv: hep-ex/9907027v1

Run details:

- e^+p deep inelastic scattering with 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 GeV^2; 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.

8. RHIC analyses

8.1 STAR_2006_S6500200

Identified hadron spectra in pp at 200 GeV
Experiment: STAR (RHIC pp 200 GeV)
Spires ID: 6500200
Status: VALIDATED
Authors:

- Bedanga Mohanty ⟨bedanga@rcf.bnl.gov⟩;
- Hendrik Hoeth ⟨hendrik.hoeth@cern.ch⟩;

References:

- Phys. Lett. B637, 161
- nucl-ex/0601033

Run details:

- pp at 200 GeV

$p_\perp$ distributions of charged pions and (anti)protons in pp collisions at $\sqrt{s} = 200$ GeV, measured by the STAR experiment at RHIC in non-single-diffractive minbias events.

8.2 STAR_2006_S6860818

Strange particle production in pp at 200 GeV
Experiment: STAR (RHIC pp 200 GeV)
Spires ID: 6860818
Status: VALIDATED
Authors:

- Hendrik Hoeth ⟨hendrik.hoeth@cern.ch⟩;

References:

- Phys. Rev. C75, 064901
- nucl-ex/0607033

Run details:

- pp at 200 GeV

$p_\perp$ distributions of identified strange particles in pp collisions at $\sqrt{s} = 200$ GeV, measured by the STAR experiment at RHIC in non-single-diffractive minbias events. WARNING The $\langle p_\perp \rangle$ vs. particle mass plot is not validated yet and might be wrong.

8.3 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:

- 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.

9. Monte Carlo analyses

9.1 MC_DIPHOTON

Monte Carlo validation observables for diphoton production at LHC
Status: VALIDATED
Authors:

- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

No references listed
Run details:

- LHC pp → jet+jet, photon+jet, photon+photon, all with EW+QCD shower

Different observables related to the two photons

9.2 MC_JETS

Status:
No authors listed
No references listed
No run details listed

9.3 MC_LEADINGJETS

Underlying event in leading jet events, extended to LHC
Status: VALIDATED
Authors:

- Andy Buckley ⟨andy.buckley@cern.ch⟩;

No references listed
Run details:

- LHC pp QCD interactions at 0.9, 10 or 14 TeV. Particles with $c\tau > 10$ mm should be set stable. Several $p_\perp^{\text{min}}$ 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 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.

9.4 MC_PHOTONJETS

Monte Carlo validation observables for photon + jets production
Status: VALIDATED
Authors:

- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

No references listed
Run details:

- Tevatron Run II ppbar → gamma + jets.

Different observables related to the photon and extra jets.

9.5 MC_SUSY

Validate generic SUSY events, including various lepton invariant mass
Status: VALIDATED
Authors:

- Andy Buckley ⟨andy.buckley@cern.ch⟩;

No references listed
Run details:

- SUSY events at any energy. $p_\perp$ cutoff at 10 GeV may be advised.

Analysis of generic SUSY events at the LHC, based on Atlas Herwig++ validation analysis contents. Plotted are eta, phi and $p_\perp$ observables for charged tracks, photons, isolated photons, electrons, muons, and jets, as well as various dilepton mass 'edge' plots for different event selection criteria.

9.6 MC_WJETS

Monte Carlo validation observables for $W[e\,\nu]$ + jets production
Status: VALIDATED
Authors:

- Frank Siegert ⟨frank.siegert@durham.ac.uk⟩;

No references listed
Run details:

- $e\,\nu$ + jets analysis.

Available observables are W mass, $p_\perp$ of jets 1-4, jet multiplicity, $\Delta\eta(W, \text{jet1})$, $\Delta R(\text{jet2}, \text{jet3})$, differential jet rates 0→1, 1→2, 2→3, 3→4, integrated 0–4 jet rates.

9.7 MC_ZJETS

Monte Carlo validation observables for $Z[e^+\,e^-]$ + jets production
Status: VALIDATED
Authors:

- Frank Siegert ⟨ frank.siegert@durham.ac.uk ⟩;

No references listed
Run details:

- e^+e^- + jets analysis. Needs mass cut on lepton pair to avoid photon singularity, e.g. a min range of $66 < m_{ee} < 116$ GeV

Available observables are Z mass, $p_\perp$ of jets 1-4, jet multiplicity, $\Delta\eta(Z, \text{jet1})$, $\Delta R(\text{jet2}, \text{jet3})$, differential jet rates 0→1, 1→2, 2→3, 3→4, integrated 0–4 jet rates.

10. Example analyses

10.1 EXAMPLE

A demo to show aspects of writing a Rivet analysis
Status: EXAMPLE
Authors:

- Andy Buckley ⟨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.

11. Misc. analyses

11.1 BELLE_2006_S6265367

Charm hadrons from fragmentation and B decays on the $\Upsilon(4S)$
Status: VALIDATED
Authors:

- Jan Eike von Seggern ⟨jan.eike.von.seggern@physik.hu-berlin.de⟩;

References:

- Phys.Rev.D73:032002,2006.
- arXiv: hep-ex/0506068
- DOI: 10.1103/PhysRevD.73.032002

Run details:

- e^+e^- analysis on the $\Upsilon(4S)$ resonance, with CoM boost – 8.0 GeV (e) and 3.5 GeV (e^+)

Analysis of charm quark fragmentation at 10.6 GeV, based on a data sample of 103 fb collected by the Belle detector at the KEKB accelerator. Fragmentation into charm is studied for the main charmed hadron ground states, namely D^0, D^+, D_s^+ and Λ_c^+, as well as the excited states D^{*0} and D^{*+}. This analysis can be used to constrain charm fragmentation in Monte Carlo generators. Additionally, we determine the average number of these charmed hadrons produced per B decay at the $\Upsilon(4S)$ resonance and measure the distribution of their production angle in e^+e^- annihilation events and in B decays.

11.2 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.

11.3 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 $\sqrt{s} \approx 10$, 29–35, 91, and 130–200 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.

11.4 UA1_1990_S2044935

UA1 multiplicities, transverse momenta and transverse energy distributions.
Experiment: UA1 (SPS)
Spires ID: 2044935
Status: VALIDATED
Authors:

- Andy Buckley ⟨andy.buckley@cern.ch⟩;
- Christophe Vaillant ⟨c.l.j.j.vaillant@durham.ac.uk⟩;

References:

- Nucl.Phys.B353:261,1990

Run details:

- QCD min bias events at sqrtS = 63, 200, 500 and 900 GeV.

Particle multiplicities, transverse momenta and transverse energy distributions at the UA1 experiment, at energies of 200, 500 and 900 GeV (with one plot at 63 GeV for comparison).

11.5 UA5_1982_S875503

UA5 multiplicity and pseudorapidity distributions for pp and ppbar.
Experiment: UA5 (SPS)
Spires ID: 875503
Status: VALIDATED
Authors:

- Andy Buckley ⟨andy.buckley@cern.ch⟩;
- Christophe Vaillant ⟨c.l.j.j.vaillant@durham.ac.uk⟩;

References:

- Phys.Lett.112B:183,1982

Run details:

- Min bias QCD events at sqrtS = 53 GeV. Run with both pp and ppbar beams.

Comparisons of multiplicity and pseudorapidity distributions for pp and ppbar collisions at 53 GeV, based on the UA5 53 GeV runs in 1982. Data confirms the lack of significant difference between the two beams.

11.6 UA5_1986_S1583476

Pseudorapidity distributions in ppbar (NSD, NSD+SD) events at c.m. energies of 200 and 900 GeV
Experiment: UA5 (CERN SPS)
Spires ID: 1583476
Status: VALIDATED
Authors:

- Andy Buckley ⟨andy.buckley@cern.ch⟩;
- Holger Schulz ⟨holger.schulz@physik.hu-berlin.de⟩;
- Christophe Vaillant ⟨c.l.j.j.vaillant@durham.ac.uk⟩;

References:

- Eur. Phys. J. C33, 1, 1986

Run details:

- Elastic scattering, single and double diffractive events. ppbar collider, c.m energy 200 or 900 GeV. The trigger implementations for NSD events is the same as in, e.g., the UA5_1989 analysis. No further cuts are needed.

This study comprises measurements of pseudorapidity distributions measured with the UA5 detector at 200 and 900 GeV center of momentum energy. There are distributions for non-single diffractive (NSD) events and als for the combination of single- and double-diffractive events. The NSD distributions are further studied for certain ranges of the events charged multiplicity.

11.7 UA5_1989_S1926373

UA5 charged multiplicity measurements
Experiment: UA5 (CERN SPS)
Spires ID: 1926373
Status: VALIDATED
Authors:

- Holger Schulz ⟨holger.schulz@physik.hu-berlin.de⟩;
- Christophe L. J. Vaillant ⟨c.l.j.j.vaillant@durham.ac.uk⟩;
- Andy Buckley ⟨andy.buckley@cern.ch⟩;

References:

- Z. Phys. C - Particles and Fields 43, 357-374 (1989)
- DOI: 10.1007/BF01506531

Run details:

- MinBias events at $\sqrt{s} = 200$ and 900 GeV. Enable single and double diffractive events in addition to minimum bias and non-diffractive processes.

Multiplicity distributions of charged particles produced in non single-diffractive collisions between protons and antiprotons at centre of mass energies of 200 and 900 GeV are presented. The data were recorded in the UA5 streamer chambers at the CERN Collider, which was operated in a pulsed mode between the two energies. A new method to correct for acceptance limitations and inefficiencies based on the principle of maximum entropy has been used. Multiplicity distributions in full phase space and in intervals of pseudorapidity are presented in tabular form. The violation of KNO scaling in full phase space found by the UA5 group at an energy of 546 GeV is confirmed also at 200 and 900 GeV. The shape of the 900 GeV distribution in full phase space is narrower in the peak region than at 200 GeV but exhibits a pronounced high multiplicity tail. The negative binomial distribution fits data at 200 GeV in all pseudorapidity intervals and in small intervals at 900 GeV. In large intervals at 900 GeV, however, the negative binomial distribution. Fits to the partially coherent laser distribution are also presented as well as comparisons with predictions of the Dual Parton, the Fritiof and the Pythia models.

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[10] 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 soon, but first a semi-philosophical aside on the "right way" to do physics analyses on and involving simulated data.

12. The science and art of physically valid MC analysis

The world of MC event generators is a wonderfully convenient one for experimentalists: we are provided with fully exclusive events whose most complex correlations can be explored and used to optimise analysis algorithms and some kinds of detector correction effects. It is absolutely true that the majority of data analyses and detector designs in modern collider physics would be very different without MC simulation.

But it is very important to remember that it is just simulation: event generators encode much of known physics and phenomenologically explore the non-perturbative areas of QCD, but only unadulterated experiment can really tell us about how the world behaves. The richness and convenience of MC simulation can be seductive, and it is important that experimental use of MC strives to understand and minimise systematic biases which may result from use of simulated data, and to not "unfold" imperfect models when measuring the real world. The canonical example of the latter effect is the unfolding of hadronisation (a deeply non-perturbative and imperfectly-understood process) at the Tevatron (Run I), based on MC models. Publishing "measured quarks" is not physics — much of the data thus published has proven of little use to either theory or experiment in the following years. In the future we must be alert to such temptation and avoid such gaffes — and much more subtle ones.

These concerns on how MC can be abused in treating measured data also apply to MC validation studies. A key observable in QCD tunings is the $p_\perp$ of the Z boson, which has no phase space at exactly $p_\perp = 0$ but a very sharp peak at $\mathcal{O}$(1-2 GeV). The exact location of this peak is mostly sensitive to the width parameter of a nucleon "intrinsic $p_\perp$" in MC generators, plus some soft initial state radiation and QED bremstrahlung. Unfortunately, all the published Tevatron measurements of this observable have either "unfolded" the QED effects to the "Z $p_\perp$" as attached to the object in the HepMC/HEPEVT event record with a

PDG ID code of 23, or have used MC data to fill regions of phase space where the detector could not measure. Accordingly, it is very hard to make an accurate and portable MC analysis to fit this data, without similarly delving into the event record in search of "the boson". While common practice, this approach intrinsically limits the precision of measured data to the calculational order of the generator — often not analytically well-defined. We can do better.

Away from this philosophical propaganda (which nevertheless we hope strikes some chords in influential places...), there are also excellent pragmatic reasons for MC analyses to avoid treating the MC "truth" record as genuine truth. The key argument is portability: there is no MC generator which is the ideal choice for all scenarios, and an essential tool for understanding sub-leading variability in theoretical approaches to various areas of physics is to use several generators with similar leading accuracies but different sub-leading formalisms. While the HEPEVT record as written by HERWIG and PYTHIA has become familiar to many, there are many ambiguities in how it is filled, from the allowed graph structures to the particle content. Notably, the Sherpa event generator explicitly elides Feynman diagram propagators from the event record, perhaps driven by a desire to protect us from our baser analytical instincts. The Herwig++ event generator takes the almost antipodal approach of expressing different contributing Feynman diagram topologies in different ways (*not* physically meaningful!) and seamlessly integrating shower emissions with the hard process particles. The general trend in MC simulation is to blur the practically-induced line between the sampled matrix element and the Markovian parton cascade, challenging many established assumptions about "how MC works". In short, if you want to "find" the Z to see what its $p_\perp$ or η spectrum looks like, many new generators may break your honed PYTHIA code... or silently give systematically wrong results. The unfortunate truth is that most of the event record is intended for generator debugging rather than physics interpretation.

Fortunately, the situation is not altogether negative: in practice it is usually as easy to write a highly functional MC analysis using only final state particles and their physically meaningful on-shell decay parents. These are, since the release of HepMC 2.5, standardised to have status codes of 1 and 2 respectively. Z-finding is then a matter of choosing decay lepton candidates, windowing their invariant mass around the known Z mass, and choosing the best Z candidate: effectively a simplified version of an experimental analysis of the same quantity. This is a generally good heuristic for a safe MC analysis! Note that since it's known that you will be running the analysis on signal events, and there are no detector effects to deal with, almost all the details that make a real analysis hard can be ignored. The one detail that is worth including is summing momentum from photons around the charged leptons, before mass-windowing: this physically corresponds to the indistinguishability of collinear energy deposits in trackers and calorimeters and would be the ideal published experimental measurement of Drell-Yan $p_\perp$ for MC tuning. Note that similar analyses for W bosons have the luxury over a true experiment of being able to exactly identify the decay neutrino rather than having to mess around with missing energy. Similarly, detailed unstable hadron (or tau) reconstruction is unnecessary, due to the presence of these particles in the event record with status code 2. In short, writing an effective analysis which is automatically portable between generators is no harder than trying to decipher the variable

structures and multiple particle copies of the debugging-level event objects. And of course Rivet provides lots of tools to do almost all the standard fiddly bits for you, so there's no excuse!

Good luck, and be careful!

13. 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' initialisation, but they can also be recursively "nested" inside other projections[2] (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.

13.1 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 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

[2]Provided 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.

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.

13.2 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 the `init` method of an `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 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

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

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 — until it's fixed, of course! — rather than silently do the wrong thing.*
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.

---

14. Analyses

14.1 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();
  };

}
```

The constructor for the `UserAnalysis` may 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 latter 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.

The `init()` method 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 is as follows:

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

A second task of the `init()` method is the booking of all histograms which are later to be filled in the analysis code. Information about the histogramming system can be found in Section 14.3.

14.2 Utility classes

Rivet provides quite a few object types for physics purposes, such as three- and four-vectors, matrices and Lorentz boosts, and convenience proxy objects for e.g. particles and jets. We now briefly summarise the most important features of some of these objects; more complete interface descriptions can be found in the generated Doxygen web pages on the Rivet web site, or simply by browsing the relevant header files.

14.2.1 `FourMomentum`

The `FourMomentum` class is the main physics vector that you will encounter when writing Rivet analyses. Its functionality and interface are similar to the CLHEP `HepLorentzVector` with which many users will be familiar, but without some of the historical baggage.

Vector components The `FourMomentum` `E()`, `px()`, `py()`, `pz()` & `mass()` methods are (unsurprisingly) accessors for the vector's energy, momentum components and mass.

Useful properties The `pT()` and `Et()` methods are used to calculate the transverse momentum and transverse energy. Angular variables are accessed via the `eta()`, `phi()` and `theta()` for the pseudorapidity, azimuthal angle and polar angle respectively. More explicitly named versions of these also exist, named `pseudorapidity()`, `azimuthalAngle()` and `polarAngle()`. Finally, the true rapidity is accessed via the `rapidity()` method. Many of these functions are also available as external functions, as are algebraic functions such as `cross(vec1, vec2)`, which is perhaps more palatable than `vec1.cross(vec2)`.

Distances The η–ϕ distance between any two four-vectors (and/or three-vectors) can be computed using a range of overloaded external functions of the type `deltaR(vec1, vec2)`. Angles between such vectors can be calculated via the similar `angle(vec1, vec2)` functions.

14.2.2 `Particle`

This class is a wrapper around the HepMC `GenParticle` class. `Particle` objects are usually obtained as a vector from the `particles()` method of a `FinalState` projection. Rather than having to directly use the HepMC objects, and e.g. translate HepMC four-vectors into the Rivet equivalent, several key properties are accessed directly via the `Particle` interface (and more may be added). The main methods of interest are `momentum()`, which returns a `FourMomentum`, and `pdgId()`, which returns the PDG particle ID code. The PDG code can be used to access particle properties by using functions such as `PID::isHadron()`, `PID::threeCharge()`, etc. (these are defined in `Rivet/Tools/ParticleIDMethods.hh`.)

14.2.3 `Jet`

Jets are obtained from one of the jet accessor methods of a projection that implements the `JetAlg` interface, e.g. `FastJets::jetsByPt()` (this returns the jets sorted by $p_\perp$, such that the first element in the vector is the hardest jet — usually what you want.) The most useful methods are `particles()`, `momenta()`, `momentum()` (a representative `FourMomentum`), and some checks on the jet contents such as `containsParticleId(pid)`, `containsCharm()` and `containsBottom()`.

14.2.4 Mathematical utilities

The `Rivet/Math/MathUtils.hh` header defines a variety of mathematical utility functions. These include testing functions such as `isZero(a)`, `fuzzyEquals(a, b)` and `inRange(a, low, high)`, whose purpose is hopefully self-evident, and angular range-mapping functions such as `mapAngle0To2Pi(a)`, `mapAngleMPiToPi(a)`, etc.

14.3 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 reference data files which are used for these booking methods are distributed and installed with Rivet, you can find them in the $\langle$*installdir*$\rangle$`/share/Rivet` directory of your installation. 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.

14.4 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.)

To load pluggable analyses you will need to set the `$RIVET_ANALYSIS_PATH` environment variable: this is a standard colon-separated UNIX path, specifying directories in which analysis plugin libraries may be found. If it is unspecified, the Rivet loader system will assume that the only entry is the `lib` directory in the Rivet installation area – specifying the variable will disable this standard location to allow you to override standard analyses with same-named variants of your own (provided they are loaded from different directories).

Note that the search path behaviour has changed as of Rivet 1.2.0: previously the standard library install directory was always *used, as were the current directory and `./.libs`, if found. While the new system requires a bit more setup if you are to use personal plugin analyses, it also solves many niggling problems and areas for confusion!*

You may also wish or need to use the `$RIVET_REF_PATH` and `$RIVET_INFO_PATH` variables, which respectively provide similar search paths for analysis reference data and analysis metadata (e.g. author, date, run conditions, experiment, etc.) files.

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
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 "⟨*key*⟩ ⟨*value*⟩": in this case, the file `lep1.params` is loaded from the ⟨*installdir*⟩`/share/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.

B. Acknowledgements

Rivet development has been supported by a variety of sources:

- All authors aknowledge support from the EU MCnet research network. MCnet is a Marie Curie Research Training Network funded under Framework Programme 6 contract MRTN-CT-2006-035606.

- Andy Buckley has been supported by grants from the UK Science and Technology Facilities Council (Special Project Grant) and from the Scottish Universities Physics Alliance (Advanced Research Fellowship).

- Holger Schulz acknowledges the support of the German Research Foundation (DFG).

Part V
Bibliography

References

[1] M. Dobbs and J. B. Hansen, Comput. Phys. Commun. **134**, 41 (2001).

[2] M. R. Whalley, D. Bourilkov, and R. C. Group, (2005), hep-ph/0508110.

[3] M. Cacciari and G. Salam and G.Soyez, (2006), hep-ph/0607071.

[4] T. Sjostrand, S. Mrenna, and P. Skands, JHEP **05**, 026 (2006), hep-ph/0603175.

[5] T. Gleisberg *et al.*, (2008), 0811.4622.

[6] M. Bahr *et al.*, Eur. Phys. J. **C58**, 639 (2008), 0803.0883.

[7] H. Jung and G. P. Salam, Eur. Phys. J. **C19**, 351 (2001), hep-ph/0012143.

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

[9] I. Antcheva *et al.*, Comput. Phys. Commun. **180**, 2499 (2009).

[10] J. Bromley *et al.*, (1995), ZEUS and H1 Collaborations.