Herwig++ Tutorial

1 Preparation

The Herwig++ homepage is at `http://projects.hepforge.org/herwig/`. To speed up the setup, we have pre-installed Herwig++ version 2.5.0 for the tutorials. To use it, you should copy the configuration files to the working directory (we'll explain their role in section 3 below). For the DESY tutorial,

```
HERWIGPATH=/intro-school/opt/Herwig++/
```

```
$ cd /intro-school/Herwig++
$ cp /HERWIGPATH/share/Herwig++/LHC.in .
```

For convenience, set a the path variable to include the Herwig++ binary directory:

```
$ export PATH=$PATH:/HERWIGPATH/bin
```

2 Simple LHC events

As a first step, we will generate 100 LHC Drell-Yan events in the default setup that comes with the distribution:

```
$ Herwig++ read LHC.in
$ Herwig++ run LHC.run -N100 -d1
```

We'll explain the commands in the next section. We have actually passed a debug flag, `-d1`, to explicitly print out some of the generated events. Looking at the file LHC.log, you should see the detailed record of the first 10 events of this *run*. Each *event* is made up of individual *steps* that reflect the treatment of the event as it passes through the various stages of the generator (hard subprocess, parton shower, hadronization and decays).

Every *particle* in a step has an entry like

```
16    g     21 [13] (42,43)    {+6,-5}
                -1.040    -2.805   177.756   177.783     0.750
```

The first line contains `16`, the particle's label in this event; `g 21`, the particle's name and PDG code; `[13]`, the label(s) of parent particle(s); `(42,43)` the label(s) of child particle(s); and {`+6,-5`}, the colour structure: this particle is connected via colour lines 5 and 6 to other

coloured particles. Sometimes you'll also see something like `7v` or `2^`; they signify that the current particle is a clone of particle 7 below / 2 above. The second line shows p_x, p_y, p_z, E and $\pm\sqrt{|E^2 - p^2|}$.

Note that everybody has generated the exact same events (go and compare!), with exactly the same momenta. Adding 10 of these runs together will *not* be equivalent to running 1000 events! To make statistically independent runs, you need to specify a random seed, either with `$ Herwig++ run LHC.run -N100 -seed 123456`
or, as we'll see now, in the LHC.in file.

3 Input files

Any ThePEG-based generator like Herwig++ is controlled mainly through input files (.in files). They create a *Repository* of component objects (each one is a C++ class of its own) and their parameter settings, assemble them into an event generator and run it. Herwig++ already comes with a pre-prepared default setup[1]. As a user, you will only need to write a file with a few lines (like LHC.in) for your own parameter modifications. The next few sections will go through this.

The first command we ran (`Herwig++ read LHC.in`) takes the default repository provided with the installation[2], and reads in the additional instructions from LHC.in to modify the repository accordingly. A complete setup for a generator run will now be saved to disk in a .run file, for use with a command like `Herwig++ run LHC.run -N100`. The run can also be started directly from the LHC.in file, which is especially useful for batch jobs or parameter scans.

Writing new .in files is the main way of interacting with Herwig++. Have a look at the other examples we have provided for LEP, Tevatron, or ILC (LEP.in, TVT.in and ILC.in) and see if you can understand the differences:
`$ cp -u /HERWIGPATH/share/Herwig++/???.in .`

The two most useful repository commands are create, which registers a C++ object with the repository under a chosen name, and set, which is used to modify parameters of an object. Note that all this can be done without recompiling any code!

Take your time to play with the options in the example files. Here are some suggestions for things you can try:

1. Run 100 Tevatron events.

[1] in `/HERWIGPATH/share/Herwig++/defaults`

[2] The default repository can also be re-created by the user, by copying over the files from `/HERWIGPATH/share/Herwig++/defaults`, and running `$ Herwig++ init`. This reads the file HerwigDefaults.in (which in turn includes most of the other .in files) to prepare a default *LEPGenerator* and *LHCGenerator* object. This default setup is now stored in HerwigDefaults.rpo. A regular user does not really need to run `init` at all, everything can be modified at the `read` stage.

2. Control a run directly from the .in file. Be careful with the number of events you generate, the default is 10^7, and we don't have that much time today!

3. Compare the Drell-Yan cross sections for Tevatron and LHC. The cross sections are written to TVT.out and LHC.out, respectively.

4 Analysis handlers

There is an easier way to analyse the generated events than looking at the .log file. ThePEG provides the option to attach multiple *analysis handlers* to a generator object. Every analysis handler initializes itself before a run (*e.g.* to book histograms), analyses each event in turn (fill histograms) and then runs some finalization code at the end of the run (output histograms).

As part of the default setup, one analysis handler has always been running already. The *BasicConsistency* handler does what its name promises: checking for charge and momentum conservation.

We ship various simple analysis handlers with a Herwig++ release, most of which produce output files ending in `.top` to be plotted by the `topdrawer` program,

```
$ tdps ???.top
$ gv ???.eps
```

4.1 Graphviz plot

Before we go on to a physics analysis, let's briefly look at a useful handler that allows us to visualize the internal structure of an event within Herwig++. Enable the *GraphvizPlot* analysis for LHC (the line in LHC.in which mentions /Herwig/Analysis/Plot) and run one LHC event. *Plot* should have produced a file LHC-Plot-1.dot, which contains the description of a directed graph for the generated event. The graphviz package will plot the graph for us:

```
$ dot -Tsvg LHC-Plot-1.dot > LHC-plot.svg
```

Have a look with `$ inkscape LHC-plot.svg`

1. Identify the Drell-Yan process. Has there been initial state radition?

2. Keep track of the incoming protons and proton remnants. Did only one $2 \to 2$ scattering take place?

It is important to note that these plots only reflect the internal event structure in the generator. Many internal lines do *not* have a physical significance!

5 Changing default settings

Take a look at the default settings in `/HERWIGPATH/share/Herwig++/defaults`, we have commented them extensively. Ask the tutors to explain parameters. Can you identify which four lines in HerwigDefaults.in control the hard subprocess, the parton shower, the hadronization and the decays?

5.1 Switching on or off simulation steps

So far, we did look at completely generated events including parton showers, hadronization, decays of hadrons and multiple parton interactions. The first three of these steps may be switched off by setting the corresponding *step handler* interfaces of an event handler to `NULL`. Multiple parton interactions are switched off by setting the `MPI` interface of the `ShowerHandler` to `NULL`. For the remainder of the tutorial we suggest to switch off multiple parton interactions, as this part of the simulation is very complex, taking too much time in running a reasonable number of events.

Add repository commands to your local LHC.in switching on or off successive steps and look at the effects by generating few events. The default settings are provided in HerwigDefaults.in and Shower.in. Take care of the directories, in which the different objects reside.

5.2 Changing the hard process

The default hard process for LHC is Drell-Yan vector boson production with leptonic decays. Edit LHC.in to replace the matrix element for vector boson production by the one for top-quark pair production for `LHCGenerator`'s default `SubProcessHandler`. The relevant matrix element is contained in the default repository, `/Herwig/MatrixElements/MEHeavyQuark`. Generate few events as for the default settings, using the flag `-d1` to see explicit event printouts.

Try to keep track of the top quarks in LHC.log. Can you identify, if there has been gluon radiation off the top quarks prior to decay?

For top-quark pair production you may be interested in using the `/Herwig/Analysis/TTbar` analysis handler to study the effects of changing default settings other than the hard process.

Warning: This analysis does not 'reconstruct' the top and anti-top from their decay products, but makes direct use of the event record as given by Herwig++.

5.3 Initial and final state radiation

Initial and final state QCD radiation may be switched off seperately. Work out the relevant interface settings by looking at the comments and repository commands in `Shower.in`, for

1. switching off initial state radiation,

2. switching off final state radiation,

Notice the difference between the `newdef` command, referring to a default setting, and the `set` command changing defaults.

Again, you can just add the respective `set` repository commands for setting interface values to your local LHC.in file.

If you want to look at some events in detail, it is reasonable to switch off hadronization, decays and multiple parton interactions when looking at the effect of the different settings (just to have clearer output).

5.4 Changing particle properties

The properties of a particle are contained in a `ParticleData` object. All of these objects are stored in the default repository in `/Herwig/Particles`.

The top quark's properties are contained in `/Herwig/Particles/t`, the anti-top's properties are set automatically. You can change the mass and width of the top quark using the `NominalMass` and `Width` interfaces.

5.5 Matrix element options

There are often also switches for the selection of a particular subprocess given with a matrix element. For W production, the relevant switch for e.g. the leptonic W–channel is
`set MEqq2W2ff:Process Leptons`.

5.6 MSSM

For inclusive SUSY particle production use the default file LHC-MSSM.in. Here, all possible hard processes are constructed automatically on the basis of a model file. All you need to do is to modify the list of possible incoming and outgoing particles. E.g.
`$ insert HPConstructor:Incoming 0 /Herwig/Particles/g`

That's it!

Thanks for trying Herwig++! If you have any questions later on, please email us at `herwig@projects.hepforge.org` or have a look at `http://projects.hepforge.org/herwig/`, where many how-tos can be found, and we'll add more on request. For detailed documentation refer to our manual, `arxiv:0803.0883`.