⚡ Herwig Tutorial

This tutorial teaches you the basics of the Herwig 7 Monte Carlo Event Generator. An installation of Herwig on your own device is not necessary if you use the provided Docker image (see below). If you plan to use Herwig for your own research, please follow the instructions provided under Installation for a local build.

For the purpose of this tutorial, you do not need to install Herwig.

In this tutorial, you will learn:

- How to simulate generic LHC Events;
- How to combine Herwig with NLO Tools like MadGraph and OpenLoops;
- How to include rivet analyses and compare a simulation with data;
- How to switch parts of the Event Generator on and off;
- How to modify parameters of the Parton shower and the Hadronisation model;
- How to run LHE files in Herwig with its Les Houches Event Handler.

Before you start: setting up Herwig's docker containers

All exercises within this tutorial can be done by using the provided Docker image for Herwig. This contains a working copy of Herwig's latest version, Herwig-7.3.0, with ThePEG-2.3.0. If you have not already installed Docker, please follow the instructions on the Docker website and install Docker on your device.

The required Docker containers can be pulled directly from the Docker Hub with

```
docker pull herwigcollaboration/herwig-7.3:7.3.0
```

Linux users will need to add **sudo** to the beginning of the above command.

To run Herwig, execute the following shell command,

```
docker run -i --rm -u `id -u $USER`:`id -g` -v $PWD:$PWD -w $PWD herwigcollaboration/herwig-7.3:7.3.0 Herwig --
```

This allows the container to access your local files and to write to the current working directory. To save rewriting this many times, you can create an alias for the run command

```
alias DRH='docker run -i --rm -u `id -u $USER`:`id -g` -v $PWD:$PWD -w $PWD herwigcollaboration/herwig-7.3:7.3.
```

You need to do this once when you start the terminal. To have it automated for you in the future, you can add this line to the end of the file **~/.bashrc** on your local machine.

To run **Herwig** execute

```
DRH Herwig
```

Similar commands like **rivet-mkhtml** can be run in the same fashion

```
DRH rivet-mkhtml
```

From here on, we will use the alias **DRH** to denote running Herwig-7.3.0 from its Docker image.

If this works, you are ready to start the tutorial with First run. If you encounter problems, please talk to a tutor.

Instructions:

1. Installation
2. First run: Z production at the LHC
3. Second run: NLO Z production at the LHC
4. Third run: minimum bias events and soft physics
5. The Les Houches Event Handler
6. Photon Induced Process in Herwig

Helpful literature
Herwig manual

⚡The First run

A First Herwig Run: Z Production at the LHC

In the first part of this tutorial, we will simulate the process **pp -> Z/gamma -> e+e-** at sqrt(s) = 13 TeV at the LHC. We will focus on the different components of the event generator: we will start by simulating the hard process, then add the parton shower and hadronisation components.

Please start by creating a work directory in your preferred path:

mkdir Herwig_tutorial
cd Herwig_tutorial

For this part of the tutorial, you will need to download this file:

You can do this manually or using the following terminal command:

wget https://phab-files.hepforge.org/file/download/v4xwihvtrkawxvwazfpg/PHID-FILE-iyegxl2qznfz5wyptfkq/LHC.in

Step 1: Hard Process Generation

Look inside the `LHC.in` input file. This is where we specify the details of our simulation. For our case, we need to specify the following:

This is a proton-proton collision

We want sqrt(s) = 13000 GeV

We want to generate Drell-Yan Z/gamma

We just want e+ e- (no other pairs)

This is also the place to specify any cuts on the events we want to generate. To make things efficient, we can specify cuts of the combined mass of the electrons to be around the *Z* boson mass.

Running Herwig

Simulating events in Herwig involves two steps:

DRH Herwig read LHC.in

will generate the `LHC.run` file (combining all requested settings with MEs, ready for simulation). This can then be run with

DRH Herwig run LHC.run -N 10000 -j 8

which generates 10000 events, parallelised into 8 separate jobs. The results will be 8 distinct sets of output files

LHC-1.log LHC-1.out LHC-1.tex LHC-1.yoda

to

LHC-8.log LHC-8.out LHC-8.tex LHC-8.yoda

Here, **.log** files contain details of the first few events. The cross-section calculated from these simulated events is also provided at the end. Useful analyses of the event and the electron pair are done using Rivet and stored as histograms in **.yoda** files. These files can be merged back together and combined using

DRH yodamerge LHC-*.yoda -o LHC.yoda

Now, you can plot these histograms using

DRH rivet-mkhtml --mc-errs LHC.yoda

This will create histograms in a folder called `rivet-plots`. For now, focus on the `MC_ZINC` folder, which has observables for the *Z*.

(Careful - every time this folder is overwritten, you can specify a different folder name using `-o`)

Monte Carlo and Convergence

Now try simulating 1, 10, 100, 1000, 10000 and 100000 events. What happens to the cross-section and its uncertainty as you increase the number of events? Does this agree with what we talked about in the lectures?

(You should achieve results like this GIF)

Step 2: The Parton Shower

You will notice in your plots that while the *Z* mass and rapidity show interesting distributions, the famously used *Z p_T* distribution is empty. This is because the quarks from our protons collide face-to-face. Gluon/Photon emissions from these quarks would offset their path, leading to the *Z* gaining transverse momentum. Similarly, the electron pair can emit photons, changing their trajectory.

Additionally, the emitted gluons can emit more gluons and decay into quark-antiquark pairs. This leads to a chain reaction/ cascade, giving rise to jets.

Parton Showers Choices

Let's stick to QCD for now - In `LHC.in`, you will find the options for two parton shower models:

The Angular Ordered Shower: The default shower in Herwig. This shower orders emissions by the angle generated between the emitter and the emission.

The Dipole Shower: The alternate shower. This model includes a third *spectator* parton, which is involved in maintaining colour flows and taking recoil from the emission.

Turn on one shower at a time and see what the *Z p_T* distribution looks like. You can now also have a look inside the `MC_ZJETS` folder, which does jet clustering with the quarks and gluons in the system.

You can now start renaming `LHC.yoda` and store the results of your different simulations. Multiple histograms can be plotted by the command

DRH rivet-mkhtml --mc-errs LHC-Angular.yoda LHC-Dipole.yoda ...

What immediate differences do you notice from before? Do the two showers agree with each other, or differ? If so, discuss the differences with your peers and the tutorial hosts.

Why is MPI off? We want to study the effect of the parton shower in isolation in this step. Additionally, we have a complete tutorial on Soft QCD that covers MPI in detail, which you can do after.

Step 3: Hadronisation

Due to confinement, all our quarks and gluons need to be converted into hadrons. While we don't have a full theory of this hadronisation process, we have models informed by the physical phenomena that we understand. There are two popular models of hadronisation used today, the string model and the cluster model.

In Herwig, use the cluster model. We start by forcing all gluons to split into quark-antiquark pairs, as well as the remnant of the proton into quarks and diquarks. These particles are paired up by their colour to form colourless pseudoparticles called clusters. These clusters then decay into unstable hadrons, which decay into stable ones. This model is based on the pre-confinement theory of quarks and relies on the parton shower to provide it with accurate momenta and colour.

Turning on Hadronisation

Enable Hadronisation by uncommenting the lines. What differences do you see? How do the two showers behave with the hadronisation model? Which areas of the histograms are affected most?

End of First Tutorial

Now you know how to generate an LHC Event in Herwig! We examined each event generator component's contribution to the simulated results. You can now proceed to the second tutorial.

Next Steps: There are two tutorials you may be interested in looking into

Second Run: NLO Processes, Matching and Merging

Third Run: MPI, Diffraction, Colour Reconnection

NLO Z production at the LHC

Now that we have gained an insight into the components of an event generator, we can start thinking about simulating the hard process with increased accuracy. In this section, we will simulate the 'P + P -> Z+X' process again, but at Next-to-Leading-Order (NLO) accuracy.

Download the following input file and move it to your work directory:

Alternatively, you can use the following command-line code:

```
wget https://phab-files.hepforge.org/file/download/r6x7crod6y5n2fvdy5z5/PHID-FILE-s7mrlfobrjxckrr24zsr/LHC-Mat
```

For now, all parts except for the hard matrix element are on. We will uncomment the necessary parts as we progress with the exercise.

To speed up the process, we use parallelisation for the integration stage:

```
DRH Herwig build --maxjobs=8 LHC-Matchbox.in
```

This will form an `LHC-Matchbox.run` file that contains all the initial setup, with unintegrated MEs. The process will give you a command to run upon completion:

```
for i in $(seq 0 4);do Herwig integrate --jobid=$i LHC-Matchbox.run & done
```

Since we are using a Docker container to run Herwig here, **we should modify this command** as

```
for i in $(seq 0 4);do DRH Herwig integrate --jobid=$i LHC-Matchbox.run & done
```

This will take a moment to run. Afterwards, you can use the `run` command to follow through with your simulation:

```
Herwig run LHC-Matchbox.run -N 100000 -j 8
```

And then, merge the results with

```
DRH yodamerge LHC-Matchbox-*.yoda -o LHC-noShower.yoda
```

Now plot the results from your First run and this new attempt together:

```
rivet-mkhtml LHC.yoda LHC-noShower.yoda
```

For the next task, we will simulate the NLO process. Have a look at the shower and matching section:

```
##################################################
## Matching and shower selection
##################################################

# read Matchbox/MCatNLO-DefaultShower.in
# read Matchbox/Powheg-DefaultShower.in
## use for strict LO/NLO comparisons
# read Matchbox/MCatLO-DefaultShower.in
## use for improved LO showering
read Matchbox/LO-DefaultShower.in

# read Matchbox/MCatNLO-DipoleShower.in
# read Matchbox/Powheg-DipoleShower.in
## use for strict LO/NLO comparisons
# read Matchbox/MCatLO-DipoleShower.in
## use for improved LO showering
# read Matchbox/LO-DipoleShower.in

# read Matchbox/NLO-NoShower.in
read Matchbox/LO-NoShower.in
```

In the input file, we choose the "read Matchbox/LO-NoShower.in" setting. Change that to "read Matchbox/NLO-NoShower.in" and re-run Herwig. By now, you should have created several .yoda files. You can compare the corresponding calculations with

```
rivet-mkhtml *.yoda
```

For the final task in this section, activate the rest of the event generation framework by choosing the other settings. How different are the LO and NLO results, with full event generation? Do the different matching schemes return different results?

You have finished all the main parts of the Herwig tutorial.
You can choose to dive deeper into parton shower variations, play around with different cuts and scales

or continue with Third run: minimum bias events and soft physics.

A note on matrix elements

As you can see, the process selection section has been completely deactivated. This is because for a process like `P + P -> Z + X`, Herwig needs to use the external matrix element (ME) providers. However, the use of these tools in Docker/VM would be problematic due to some linking issues. In an installed (and properly linked) version of Herwig, one can easily use these external ME providers by selecting the hard-process in the `Process selection` and choosing the ME providers from the `Matrix element library selection`.

```
##################################################
## Process selection
##################################################

## Model assumptions
# read Matchbox/StandardModelLike.in
# read Matchbox/DiagonalCKM.in

## Set the order of the couplings
# cd /Herwig/MatrixElements/Matchbox
# set Factory:OrderInAlphaS 1
# set Factory:OrderInAlphaEW 1

## Select the process
# do Factory:Process p p -> Z0 j

##################################################
## Matrix element library selection
##################################################

## Select a generic tree/loop combination or a
## specialized NLO package

# read Matchbox/MadGraph-GoSam.in
# read Matchbox/MadGraph-MadGraph.in
# read Matchbox/MadGraph-NJet.in
read Matchbox/MadGraph-OpenLoops.in
# read Matchbox/HJets.in
# read Matchbox/VBFNLO.in

## Uncomment this to use ggh effective couplings
## currently only supported by MadGraph-GoSam

# read Matchbox/HiggsEffective.in
```

minimum bias events and soft physics

```
If you are unfamiliar with non-perturbative physics or anything that is mentioned here (minimum bias, underlyi
```

In this exercise, we are going to simulate minimum bias events, and compare with minimum bias and underlying event measurements. Also, we will switch off parts of the event generator, use different parameters for the hadronization model and study the effects on the observables (playing around with the event generator in order to improve the description of some observables).

The simulation of minimum bias events heavily relies on the accurate modeling of:

- Multi-parton interactions,
- Diffraction
- Colour reconnection
- Hadronization

Minimum bias and underlying event analyses are therefore excellent assets to study these aspects of the event simulation.

Use what you have learned so far and simulate 10000 Events using the prepared LHC-MB.in file, then change the name of the created .yoda file to something else (default.yoda for example) and look at the plots.

You can download the input file via

```
wget https://phab-files.hepforge.org/file/download/3fx5ccp6qkd5bnztpya5/PHID-FILE-llgb6slr2623oxfg6cx5/LHC-MB.
```

Colour Reconnection

In the next step, switch off colour reconnection and simulate again 10000 events.

```
The switch is already included in the input file. You just need to find and remove the comment.
```

Now plot the two yoda files with the following command

```
rivet-mkhtml default.yoda LHC-MB.yoda
```

and look at the observables.

```
Which observables are heavily affected by colour reconnection?
```

Diffraction

Now we are going to switch off diffraction (remember to include colour reconnection again). Just comment out the following line in the input file

```
read snippets/Diffraction.in
```

and run LHC-MB.in again for 10000 events.

Have a look at the plots from ATLAS_2012_I1084540 and try to explain what happens.

Hadronization parameters

For a realistic simulation of minimum bias events we need to include colour reconnection and diffraction.
While the minimum bias model does a relatively good job in describing general properties of the measurements, such as rapidity distributions or pT spectra of charged particles, it becomes more difficult once the observables become more inclusive.

Have a look at the flavor observables from CMS_2011_S8978280.
In two steps we are going to change several parameters in the Hadronization model to improve the description of strangeness observables like the Kaon pT distribution.

```
1: Increase the value of PwtSquark (default is ~0.3)
```

The parameter PwtSquark is the weight to produce a strange-antistrange quark pair during cluster fission.
Increasing it will increase to probability to produce strangeness during the cluster fissioning stage.

Run Herwig again for 10000 events and compare with the default.yoda file you created in the beginning.

In the next step we allow the non-perturbative gluons, which are left after the Parton Shower evolution has terminated, to split into strange-antistrange quark pairs.
Fot this we have to make several changes to the input file.

```
2: Remove the comments from Change 3 (see input file) and assign a value between 0 and 1 to SplitPwtSquark (de
```

Run Herwig again for 10000 events and compare with the other .yoda files.

```
Do the observables favor strangeness from the cluster fissioning stage or from the gluon splitting stage?
What is the difference? (Low PwtSquark and high SplitPwtSquark value and vice versa)
```