Sherpa SM Tutorial

Freiburg GK 2016

1 Introduction

Sherpa is a complete Monte-Carlo event generator for particle production at lepton-lepton, lepton-hadron, and hadron-hadron colliders [1]. The simulation of higher-order perturbative QCD effects, including NLO corrections to hard processes and resummation as encoded in the parton shower, is emphasized in Sherpa. QED radiation, underlying events, hadronization and hadron decays can also be simulated. Alternatively, Sherpa can be used for pure parton-level NLO QCD calculations with massless or massive partons.

Many reactions at the LHC suffer from large higher-order QCD corrections. The correct simulation of Bremsstrahlung in these processes is essential. It can be tackled either in the parton-shower approach, or using fixed-order calculations. Sherpa combines both these methods using a technique known as Matrix Element + Parton Shower merging (ME+PS). Details are described in Ref. [2] and have been discussed in the lectures. This tutorial will show you how to use the method in Sherpa.

Sherpa is installed on the virtual machine in `/opt/hep`, and its documentation is found online [3]. Example setups are located in `/opt/hep/share/SHERPA-MC/Examples`. To analyse events the Rivet [4] analysis package is used.

1.1 Getting started

Start the virtual machine. Sherpa has been pre-installed, and the standard examples are located at `/opt/hep/share/SHERPA-MC/Examples`. You will find the tutorial in the directory

`~/tutorial/mc/intro/sherpa`

Change there via

```
cd ~/tutorial/mc/intro/sherpa
```

2 The Input File

Sherpa is steered using input files, which consist of several sections. A comprehensive list of all input parameters for Sherpa is given in the Sherpa manual [3]. For the purpose of this tutorial, we will focus on the most relevant ones.

In the directory `tutorials/mc/intro/sherpa` open the file `Run.dat` in an editor. Have a look at the section which is delimited by the tags `(run){` and `}(run)` (We will call this section the `(run)` section in the following). You will find the specification of the collider, i.e. its beam type and center-of-mass energy, as well as a couple of other parameters, which will be explained later.

The `(processes)` section specifies, which reactions are going to be simulated. Particles are identified by their PDG codes, i.e. 1 stands for a down-quark, -1 stands for an anti-down, 2 for an up-quark, etc. The special code 93 represents a "container", which comprises all light quarks, b-quarks, and the gluon. It is also called the "jet" container.

3 Running Sherpa

Have you found out which physics process is going to be generated? You can verify your guess by running the command line

```
Sherpa -e1 -o3
```

When run for the first time, Sherpa will produce diagram information for the calculation of hard scattering processes. It will also compute that hard scattering cross sections, which are stored, together with the diagram information, for subsequent runs.

The option `-e1` used above instructs Sherpa to produce one event, and the option `-o3` will print the result of event generation on the screen. You will see Sherpa's internal event record. Search for `Signal Process` inside the output and check incoming and outgoing particles. What happens to the unstable particles after they have been produced in the hard scattering? Is this result physically meaningful?

Have a look at the Feynman diagrams, which contribute to the simulation of the hard process:

```
plot_graphs.sh graphs/
firefox graphs/index.html
```

Are these the diagrams you expect to find? If not, which ones are missing? Can you find the setting in the runcard which restricts the set of diagrams?

4 Unstable particles

Open the input file again. Add the setting `HARD_DECAYS On;` to the `(run)` section, which instructs Sherpa to automatically decay unstable particles. Verify that the particles you wish to decay are flagged unstable by checking the screen output of Sherpa during runtime. Search for the `'List of Particle Data'`. Note that you can set particles stable individually using the switch `STABLE[<PDG ID>]=1`.

5 ME+PS merging

The current runcard lets Sherpa generate events at lowest order in the strong coupling. To improve the description of real radiative corrections, we can include higher-multiplicity tree-level contributions in the simulation. This is done by changing the process specification

```
Process 93 93 -> 6 -6;
```

to

```
Process 93 93 -> 6 -6 93{1};
```

The last entry instructs Sherpa to produce up to one additional "jet" using hard matrix elements and combine the respective process with the leading-order process. This is known as Matrix Element + Parton Shower merging (ME+PS), or the CKKW method. The essence of the method is a separation of hard from soft radiative corrections, achieved using phase-space slicing by means of a variable called the jet criterion. The slicing parameter is called the merging cut.

Let us assume we want to classify jets of more than 20 GeV transverse momentum as hard. In Sherpa, the corresponding merging cut would be specified as

```
CKKW sqr(20/E_CMS);
```

Therefore, the complete `(processes)` section for the merged event sample reads:

```
(processes){
  Process 93 93 -> 6 -6 93{1};
  CKKW sqr(20/E_CMS);
  Order (*,0);
  End process;
}(processes);
```

If you like, you can have a look at the Feynman graphs again, which contribute to the ME+PS merged event sample. In this case, you should not remove the line `Print_Graphs graphs;` from the `(processes)` section, and rerun the plot command from Sec. 3.

Run the new setup. Why does Sherpa compute another cross section?

6 Analyses

By default, Sherpa does not store events. Run Sherpa with the following command to write out event files in HepMC format, which can subsequently be analyzed

```
Sherpa EVENT_OUTPUT=HepMC_Short[events_01j]
```

Sherpa will produce a gzipped file called `events_01j.hepmc.gz`, which can be processed with Rivet using the command

```
mkfifo events
gunzip -c events_01j.hepmc.gz > events & \
rivet -a MC_TTBAR -H analysis_01j.yoda events
```

The option `-a MC_TTBAR` instructs Rivet to run a Monte-Carlo analysis of semileptonic $t\bar{t}$ events, which will provide us with a few observables that can be used for diagnostic purposes. Using a fifo pipe to extract the compressed event file avoids having to expand this file on disk, which can take substantial time and would unnecessary take up storage space. This method will prove very useful in the tutorial on boosted final states.

You can display the results of this analysis using the command

```
rivet-mkhtml --mc-errs analysis_01j.yoda
firefox plots/index.html
```

Now it is your turn: Generate a separate event file without ME+PS merging and analyze it with Rivet. Compare the results to your previous analysis using `rivet-mkhtml` (Hint: You can specify multiple yoda input files to `rivet-mkhtml`.) Do you observe any differences? Why?

7 Playing with the setup

Here are a few suggestions to try if you have time left during the tutorial.

7.1 Changing the functional form of scales

You may have noticed the following line in the runcard

```
CORE_SCALE VAR{sqr(175)};
```

It invokes Sherpa's interpreter to compute the renormalization and factorization scale for the $pp \to t\bar{t}$ production process as the mass of the top quark (Note that all scales have dimension GeV^2, hence the scale is squared using the `sqr` function).

You can change this to a different value. For example, you could use the invariant mass of the top quark pair:

```
CORE_SCALE VAR{Abs2(p[2]+p[3])};
```

Note that when you change the scale, the cross section will change and needs to be recomputed! You can instruct Sherpa not to pick up the old result and compute a new one by launching it with the commandline option `-r Result_NewScale/`. New cross section information will then be stored in the directory `Results_NewScale/`, while the old is still present in the directory `Results/`.

How do the observables change with the new scale and why? What is the effect of ME+PS merging when using this scale?

7.2 Hadronization and underlying event

So far, multiple parton interactions, which provide a model for the underlying event, have not been simulated. Also, hadronization has not been included in order to speed up event generation. You can enable both by removing or commenting the line

```
MI_HANDLER None; FRAGMENTATION Off;
```

How do the predictions for the observables change and why?

7.3 Variation of the merging cut

ME+PS merging is based on phase-space slicing, and the slicing cut is called the merging scale. Is is an unphysical parameter, and no observable should depend sizeably on its precise value. Verify this by varying the merging cut by a factor of two up and down.

Note that the initial cross sections that Sherpa computes will be different for different values of the merging cut (Why?). You should therefore instruct Sherpa to use different result directories for each run in the test. The result directory is specified on the command line with option `-r`, for example

```
Sherpa -r MyResultDirectory/
```

7.4 Other hard scattering processes

Try to generate other hard scattering processes, like the production of a Drell-Yan muon pair (Hint: The PDG id for the muon is `13`.) You will have to insert an additional section into the runcard, which reads

```
(selector){
  Mass 13 -13 66 116;
}(selector);
```

Why is this section needed?

References

[1] T. Gleisberg et al. "Event generation with SHERPA 1.1" JHEP **02** (2009) 007.

[2] A. Buckley et al. "General-purpose event generators for LHC physics" Phys. Rept. **504** (2011) 145.

[3] https://sherpa.hepforge.org/doc/SHERPA-MC-2.2.1.html.

[4] A. Buckley, J. Butterworth, L. Lönnblad, D. Grellscheid, H. Hoeth, J. Monk, H. Schulz and F. Siegert, "Rivet user manual" Comput. Phys. Commun. **184** (2013) 2803.