Tutorial on Monte-Carlo Uncertainties

CTEQ School 2014

July 15, 2014

1 Introduction

In this tutorial we will discuss some of the uncertainties related to the simulation of multi-jet final states. We will use Higgs-boson plus jets production as an example. You should work in groups of two (or multiples thereof), where each member generates predictions from a different event generator. Ideally, you will have a selection of people who already know "their" generator from using it in Monday's tutorial.

At the end of the analysis part of the tutorial, you should exchange the aida files, such that all members of your group can generate the plots on their own machine.

The easiest way to accomplish this may be DropBox (`http://www.dropbox.com/`). The DropBox Linux application is installed on the VM. You can launch it by typing `dropbox start` in a terminal window. Please note that you will have to sign in using your own account.

Alternatively, you can use a USB flash drive. You need to make it available to the VM using the "Devices" tab, see `http://tinyurl.com/69ybgp4`, steps 4 and 5. Note that it is essential to add your user to the vboxuser group, as explained in step 5! You can then open the file manager from the task bar and mount the flash drive by clicking on the "usb" icon.

2 General considerations

Different event generators are sometimes based on very different assumptions about the underlying physics in certain collider processes. In order to compare their results it is therefore important to know the uncertainty associated with a particular prediction. This tutorial will show you how to quantify perturbative QCD uncertainties associated with the predictions from Pythia and Sherpa for Higgs-boson plus multi-jet production with the Higgs decaying to a tau lepton pair (The actual type of decay is irrelevant here, because we focus on uncertainties in the production, not the decay process).

3 Generating and analyzing events

To visualize the results of this tutorial you can use the script `plotit.sh`, which is provided in the top level directory, `~/tutorials/higgs/`. Running `./plotit.sh --help` will show all available options. The plot script will be your most efficient means to put all results onto a single web page and compare the output of the different generators. To this end, you would run

```
./plotit.sh py pyme pyps sh shme shps
```

You can display the results in a browser using the command `firefox plots/index.html`.

Alternatively, if you want to plot, say, the central predictions only and store the plots in a different folder, you would run

```
./plotit.sh -o central py sh
```

Alternatively, you can use your own `rivet-mkhtml` commands.

3.1 Pythia

The most recent development in combining fixed-order calculations and parton shower resummation is next-to-leading order merging. This is different from NLO matching (e.g. POWHEG and MC@NLO) because it allows to describe different parton multiplicities simultaneously at NLO accuracy. NLO merging schemes are the successor of tree-level merging schemes.

In this tutorial, we will assess the uncertainties of the unitarised NLO+PS merging method (UNLOPS, [1]). For this, we will depart from our usual practise of simply printing histograms to the terminal. Instead, we will use a sample main program that uses HepMC output, which can then be analysed with Rivet. We will briefly describe the main program used for this study, and come back to the actual uncertainty estimates below. The Pythia 8 sources are installed in `/opt/hep/`, but you will not need to know their explicit location, as the relevant environment variables have been set for you in `~/.bashrc`.

3.1.1 main88.cc

UNLOPS merging is the direct extention of UMEPS (see last session) to NLO accuracy. For this school, we would like our simulation to describe $\mathrm{H}+0$ and $\mathrm{H}+1$ jet simultaneously with NLO accuracy, describe $\mathrm{H}+2$ jets with tree-level accuracy, and take all further jets from the PS approximation. This means we have to consider two types of input calculations: Tree-level and NLO inputs. For this school, we supply tree-level event samples produced with MadGraph, and NLO event samples produced with POWHEG-BOX. UNLOPS then processes these samples in the following way:

(1) Use tree-level matrix elements for n partons as "seeds" for higher-order corrections (of $\mathcal{O}\left(\alpha_{\mathrm{s}}^{n+2}\right)$ and beyond), making sure that no $\mathcal{O}\left(\alpha_{\mathrm{s}}^{n+1}\right)$ are produced.

(2) Add the NLO samples, making sure that no higher orders (of $\mathcal{O}\left(\alpha_{\mathrm{s}}^{n+2}\right)$ and beyond) are produced.

(3) Unitarise everything, making sure that no unwanted $\mathcal{O}\left(\alpha_{\mathrm{s}}^{n+1}\right)$ terms are produced, i.e. ensure that the inclusive cross section is given by the $\mathrm{H}+0$ NLO result.

This is done internally in `main88.cc`, the sample main program we will use to generate NLO merged results. In the event generation phase, `main88.cc` uses the tree-level input file twice (once for Step (1) and once for Step (3)), as well as using the NLO (POWHEG) inputs twice (once for Step (2) and once for Step (3)). `main88.cc` is described in detail in the online manual under **Link to other programs** $\rightarrow$ **NLO merging**[1].

`main88.cc` reads a settings file (e.g. `main88.cmnd`) for the necessary settings. It is important to set the switches

```
// Definition core process for merging
Merging:Process                = ?
Merging:mayRemoveDecayProducts  = ?
// Maximal number of additional LO jets.
Merging:nJetMax                = ?
// Maximal number of additional NLO jets.
Merging:nJetMaxNLO             = ?
// Merging scale value.
Merging:TMS                    = ?
// Values of (fixed) scales in the matrix element calculation.
Merging:muFacInME              = ?
Merging:muRenInME              = ?
// Values of (fixed) scales for the PS lowest multiplicity process.
Merging:muFac                  = ?
Merging:muRen                  = ?
```

in such an input file. The process definition, maximal number of additional tree-level jets and merging scale value have already been discussed in the CKKW-L section. In addition, you now need to set the

[1]In fact, we have slightly changed `main88.cc` to be able to read compressed LHE files.

maximal number of additional jets for which an NLO event sample is available, and the renormalisation and factorisation scales with which the event samples have been produced. Finally, it is also necessary to set te scales which would be used in default PYTHIA 8 to evaluate the core scattering (i.e. for H production, the mass m_{H}). In the case of wimpy showers, the value of `Merging:muFac` further sets the shower starting scale in UNLOPS.

In general, you also need to generate tree-level- and POWHEG event samples as input for `main88.cc`. For this school, we have included some pre-calculated samples in the directory

```
∼/tutorials/higgs/pythia .
```

The main program `main88.cc` assumes, to allow for a streamlined file parsing, a particular naming convention. All POWHEG event samples should be called

```
myLHEF_powheg_njets.lhe.gz
```

where `myLHEF` is a free process idetifier, which is assumed to be identical for all samples belonging to one particular process. `njets` should give the number of additional partons that are described at NLO accuracy (i.e. *not* counting real-emission partons). POWHEG events for $\mathrm{H} + j$@NLO could for example be called `higgs_powheg_1.lhe.gz`. Tree-level inputs are called

```
myLHEF_tree_njets.lhe.gz
```

A legitimate name for a tree-level sample with two additional jets could e.g. be `higgs_tree_2.lhe.gz`.

To use `main88.cc`, go into the `examples` directory and compile the main program

```
cd ∼/tutorials/higgs/pythia .
make main88
```

and run by issuing a command of the form

```
./main88.exe myInputFile myLHEF myHEPMCoutput
```

`main88.cc` will then, consecutively, read tree-level and POWHEG event samples, and produce HEPMC events.

To get used to the program, produce only small number of events, and run

```
./main88.exe main88.cmnd ggh_cc myHEPMCoutput
```

Can you identify the Steps (1)-(3) through the terminal output? Why is step (3) applied both to tree-level and POWHEG samples?

3.1.2 Uncertainty estimates in PYTHIA 8

In this part, we will try to estimate the scale uncertainty of the UNLOPS method. For this, we will vary μ_f and μ_r in the matrix element calculation, while keeping the scales in the parton shower resummation unchanged. In the `examples` directory, you can find event files produced with three different scale settings:

- Files whose name contains `ll` have been generated with $\mu_f = \mu_r = \frac{1}{2}m_{\mathrm{H}}$
- Files whose name contains `cc` have been generated with $\mu_f = \mu_r = m_{\mathrm{H}}$
- Files whose name contains `hh` have been generated with $\mu_f = \mu_r = 2m_{\mathrm{H}}$

Let us now investigate the scale uncertainty of the (H+0, 1@NLO)+(H+2@LO) prediction. We will use the analysis `MC_HJETS` of Rivet to do this.

For each (fixed-order) scale setting, you will need to change `Merging:muFacInME` and `Merging:muRenInME` in your input file and use the correct LHE samples.

We use HEPMC events to parse the PYTHIA 8 output to Rivet. However, HEPMC event files quickly become prohibitively large. This is why Rivet allows the usage of `fifo` pipes to pass the events at generator run time, without having to store large intermediate files. PYTHIA 8 will write a single event to the pipe, and Rivet will read and analyse this event. To make this as simple as possible, you can use the BASH script `run.sh`. This script allows to run both PYTHIA 8 and Rivet in the background of one common terminal. Simply run

```
./run.sh
```

`main88.cc` needs to be compiled before running the script. You will have to change the name of LHE files, log-files and `aida` output files in the script, and the scales in the settings file, if you want to generate histograms with different scale choices.

Now, try to do the following

- Run `main88.cc`, for the central scales, with only $(\mathrm{H}+0@\mathrm{NLO})+(\mathrm{H}+1,2@\mathrm{LO})$, and with $\mathrm{H}+0,1@\mathrm{NLO})+(\mathrm{H}+2@\mathrm{LO})$. Check the exclusive jet multiplicity of the `MC_HJETS` analysis. Can you explain the differences?
- Run `main88.cc`, with $\mathrm{H}+0,1@\mathrm{NLO})+(\mathrm{H}+2@\mathrm{LO})$ for the three scale choices
  (a) $\mu_f = \mu_r = \frac{1}{2}m_\mathrm{H} = 62.5$ GeV
  (b) $\mu_f = \mu_r = m_\mathrm{H} = 125$ GeV
  (c) $\mu_f = \mu_r = 2m_\mathrm{H} = 250$ GeV

  Display the results together, and try to understand the differences.
- Switch between wimpy and power showers by changing the settings `TimeShower:pTmaxMatch` and `SpaceShower:pTmaxMatch`.

Finally, compare the UNLOPS results to the results of the other Event Generators. This is most conveniently done if you display the variations as an uncertainty band, using the `plotit.sh` script described above. The Pythia results by themselves can be plotted using

```
cd ~/tutorials/higgs/
./plotit.sh -o MyPythiaPlots py pyme pyps
```

Don't hesitate to contact us [2] if you have further questions.

3.2 Sherpa

To assess the Monte-Carlo uncertainty in Sherpa, we will produce events with the MC@NLO technique, and with ME+PS merging at the NLO. The NLO merged simulation contains up to 2 jets computed with hard matrix elements, where the 0 and 1 jet process have NLO accuracy and the second jet is simulated at LO. We will vary renormalization and factorization scales as well as parton-shower parameters.

Change to the tutorial directory `tutorials/higgs/sherpa`. In the first two runs, Sherpa will generate process-specific source code necessary for the computation of the tree-level matrix elements and for the NLO subtraction terms (Note that this is different from what you have experienced during the previous tutorial, because we are using a different matrix element generator). Generating the source code is done one process at a time, hence you will execute the following commands

```
Sherpa
./makelibs -n
```

The individual cross sections have been pre-computed and the results are stored in the directory `Results`. You can launch Sherpa and it will start generating events immediately:

```
Sherpa
```

While Sherpa runs and generates events, have a look at the runcard. The most important part is the `(processes)` section:

```
(processes){
  Process 93 93 -> 25 93{NJET};
  Order_EW 1; CKKW sqr(QCUT/E_CMS);
  NLO_QCD_Mode MC@NLO {LJET};
  Loop_Generator Internal;
  Integration_Error 0.001 {1};
  Integration_Error 0.025 {2};
  End process;
}(processes);
```

Note that we produce a stable Higgs boson with up to two additional “jets”. This is because the tag `NJET` is defined as `NJET:=2` in the `(run)` section. Such a tag can be used anywhere in the runcard, and its definition can also be changed on the command line.

Another tag is `LJET`, which is defined as `LJET:=1,2`. When it is used in the line

```
NLO_QCD_Mode MC@NLO {LJET};
```

it steers the implementation of NLO corrections to the hard processes. All processes with final state multiplicity 1 and 2 are then computed at NLO accuracy using the MC@NLO method.

We use the command `Integration_Error` to reduce the target precision of the integration step in order to save valuable runtime. This option should be used as few as possible.

All other options in the `(processes)` section are known from yesterday’s tutorial.

The `(run)` section contains several more settings, which are new.

- We use an effective operator approach to implement the Higgs-Gluon couplings using `MODEL SM+EHC`.
- The Higgs boson mass is set to 125 GeV.
- We allow variation of factorization, renormalization and resummation scales using the `SCALES` parameter.

The last point deserves some more attention. Look at the precise specification of the scale:

```
# tags and settings for scale variations
SP_NLOCT 1; FSF:=1.0; RSF:=1.0; QSF:=1.0;
SCALES STRICT_METS{FSF*MU_F2}{RSF*MU_R2}{QSF*MU_Q2};
```

Firstly, the scale is defined using the ME+PS merging algorithm. This is indicated by the `STRICT_METS` setting (METS stands for Matrix Elements plus Truncated Showers). Secondly, the three components of the scale definition, factorization, renormalization and resummation scale, which are indicated by `MU_F2`, `MU_R2` and `MU_Q2`, respectively, can be varied independently using the tags `FSF`, `RSF` and `QSF`.

Note that all scales in Sherpa have dimension GeV^2, hence setting `FSF:=4`, for example, means increasing the factorization scale by a factor two.

Now it is your turn! Generate the following event files:

- The central prediction, defined by the settings in the runcard.
- Simultaneous variations of the renormalization and factorization scale in the range $1/2 \ldots 2$.
- Variations of the resummation scale in the range $\sqrt{1/2} \ldots \sqrt{2}$.

Note that, because of the setting `ANALYSIS_OUTPUT Analysis/NJET/QCUT/FSF-RSF/QSF;` your aida files will be put into a directory determined by the tags you specify, and it is not necessary to set file names on the command line.

Generate plots from the Sherpa output by using the plot script.

```
cd ~/tutorials/higgs/
./plotit.sh -o MySherpaPlots sh shme shps
```

Compare your results to the predictions from other generators by exchanging aida files with your peers and running the plot script as described in the introduction.

References

[1] L. Lönnblad and S. Prestel, JHEP **03** (2013) 166, `arxiv:1211.7278 [hep-ph]`

[2] For merging related questions in Pythia 8, email `stefan.prestel@thep.lu.se`
In case of general problems, contact us under `pythia8@projects.hepforge.org`