Table of Contents

Part 1: Using to generate parton-level events

In the first part of the exercise, we will use the matrix element generator MadGraph5 _amc@NLO, or in short MG5. MG5 can perform automatic matrix element predictions for many processes at leading and next-to-leading order accuracy in QCD. Because of it's ease of use for processes both in and beyond the standard model, it is one of the most widely used software tools to model the hard interaction.

Setting up CMSSW

While MG5 can be used completely standalone, we will still set up a CMSSW area just to make sure that everyone has the same libraries loaded and can profit from pre-installed software such as LHAPDF.

```
source /cvmfs/cms.cern.ch/cmsset_default.sh
scram p CMSSW_7_1_30
cd CMSSW_7_1_30/src
cmsenv
cd -
```

Task 1: Using standalone MG5

In this part of the exercise, we will learn how to use MG5 to generate W boson events in proton-proton collisions.

Input files

What type of events are produced and what settings are used is defined in the MadGraph steering files, which are called run_card.dat and proc_card.dat.

In CMS, we maintain a large archive of these steering files for all samples that are generated in the official CMS production. The cards are stored in a GitHub repository. Let's check it out:

```
git clone git@github.com:AndreasAlbert/genproductions.git -b CMSDAS2018_26x
```

The cards we are interested in are located in this subfolder:

```
ls genproductions/bin/MadGraph5_aMCatNLO/cards/examples/wplustest_4f_LO/

wplustest_4f_LO_run_card.dat
wplustest_4f_LO_proc_card.dat
```

The proc_card file defines process to be generate (hence the name proc). Look at the syntax:

```
generate p p > w+, w+ > l+ vl
```

What does this statement mean? Refer to the MG5 documentation on the syntax . What modifications of this statement can you think of? What could we replace the "w+ > l+ vl" part with?

Setting up Madgraph

We will download MG5 version 2.6.0 from a CMS-hosted mirror of the official release:

```
wget https://cms-project-generators.web.cern.ch/cms-project-generators/MG5_aMC_v2.6.0.tar.gz
tar xf MG5_aMC_v2.6.0.tar.gz
rm MG5_aMC_v2.6.0.tar.gz
cd MG5_aMC_v2_6_0
```

While all interaction with MG5 can be entirely scripted (which is useful for larger scale production, as we shall see later), in this exercise it will be instructive to use the interactive command shell provided by MG5.

```
./bin/mg5
```

You will be asked whether you want to update to a newer version of MG5. Don't.

Running Madgraph

To get the generation started, we can now paste the contents of the proc_card line by line. What does each line do? Make sure to pay attention to the console output you get in reply to your input.

Tip: If you want to execute a shell command from inside of the MG5 console, preface it with an exclamation mark, e.g."

```
!cat  ../genproductions/bin/MadGraph5_aMCatNLO/cards/examples/wplustest_4f_LO/wplustest_4f_LO_pro
```

The generation can be started by typing launch. MG5 will ask you a few more questions. When asked about the run_card, one can either use a default run card by just pressing ENTER, edit the default run_card by hand or provide a path to run card of one's choice. Please provide the path to the pre-made run_card:

```
../genproductions/bin/MadGraph5_aMCatNLO/cards/examples/wplustest_4f_LO/wplustest_4f_LO_run_card.
```

Inspecting the output

After the run has terminated, have a look at the reported cross-section, and check out the output file. (Tip: If you have trouble finding it, you can search for it:

```
find -name '*.lhe.gz'
```

To get the human-readable LHE output file, extract the gzipped file, and inspect with less:

```
gzip -d output_file.lhe.gz
less output_file.lhe
```

Task 2: Using the gridpack workflow

As mentioned previously, interactive running of madgraph is useful for exercises and smaller tests, but ultimately not practical for large-scale production. To avoid having to use the interactive mode, one can use gridpacks instead. A gridpack is simply an archive file that contains all the executable MG5 code needed to produce LHE events for a given process. It has the advantage that once it is created, it is a one-button program to generate events, no thinking required.

In this part of the exercise, we will use the same input cards as before to create a gridpack, run it, and compare the results to before.

Creating the gridpack

The script infrastructure to create gridpacks is maintained in the genproductions repository. Gridpacks are generated using the gridpack_generation script, which we will run in local mode, i.e. on the machine we are currently logged in to. Note that scripts are provided to run the gridpacks on other computing infrastructures such as the CERN batch system and CMSConnect, which is useful for more complicated processes.

To create a gridpack, we simply call gridpack_generation and pass the process name and card location to it:

```
cd genproductions/bin/Madgraph5_aMCatNLO
time ./gridpack_generation.sh wplustest_4f_LO cards/examples/wplustest_4f_LO local
```

(Note that there are naming conventions for the card. For a given process name $NAME, the input cards must be named as $NAME_run_card.dat, $NAME_proc_card.dat,etc...)

After the gridpack has been created (this should take approximately 5 minutes), we can extract and run it to produce LHE events:

```
mkdir work
cd work
tar xf ../*.xz

NEVENTS=10000
RANDOMSEED=12345
NCPU=1
./runcmsgrid.sh $NEVENTS $RANDOMSEED $NCPU
```

The output file will be cmsgrid_final.lhe.

Comparing the output LHE files

There are multiple ways of analyzing an LHE file, each of which has its own advantages and disadvantages. For the purpose of this exercise, we will use the most straightforward tool: MadAnalysis (MA). MA is a tool designed to be used by theorists to analyze parton-level LHE files, particle-level HEPMC files or even events with DELPHES detector simulation. We can install MA directly from the MG5 console:

```
cd MG5_aMC_v2_6_0
./bin/mg5
install MadAnalysis5
```

After installation finishes, exit the MG5 console and launch MA:

```
./HEPTools/madanalysis5/madanalysis5/bin/ma5
```

LHE files can be imported as different datasets:

```
import wplustest_4f_LO/Events/run_01/unweighted_events.lhe as STANDALONE
import ../genproductions/bin/MadGraph5_aMCatNLO/work/cmsgrid_final.lhe as GRIDPACK
```

Now we can user simple syntax to define the distributions we would like to look at:

```
# set STANDALONE.xsection = 1
# set GRIDPACK.xsection = 1

plot PT(mu+) 20 0 100
plot PT(mu-) 20 0 100
plot PT(l+) 20 0 100
plot PT(mu+ vm) 20 0 100
plot M(mu+ vm) 40 40 120
plot M(e+ ve) 40 40 120
plot M(ta+ vt) 40 40 120
plot M(l+ vl) 40 40 120
```

and start the analysis:

```
set main.stacking_method = superimpose
submit
```

The analysis output can be viewed as HTML:

```
firefox ANALYSIS_0/HTML/index.html
```

What observations do you make? Are the two datasets consistent? What are the shapes of the lepton pT distributions? What is the shape of the pT distribution of the W system? Are these shapes physical?

Feel free to experiment here and plot other quantities you find interesting. You can refer to the user manual to learn about the syntax.

Part 2: Generating particle-level events

Setting up CMSSW

```
# Setup
scram p CMSSW_9_3_9_patch1
cd CMSSW_9_3_9_patch1;
cmsenv;
cd -;
```

TASK 3: Parton showering

To generate physical collision events, the LHE files we have created need to be showered. Parton showering accounts for non-perturbative QCD effects with phenomenological models. The most used tool for parton showering in CMS is Pythia 8 (P8). While we could again run P8 completely on our own (you can download it from the P8 website INSERTLINKHERE, compile and run it on your laptop if you'd like), we are going to use the CMSSW GeneratorInterface in this exercise. Simply stated, CMSSW can act as a wrapper around various generators and chain their outputs together. This makes it quite easy to run large-scale production from a gridpack to the finished (Mini/Nano)AOD file.

Use cmsDriver to shower an existing LHE file

All CMSSW data and MC processing is controlled with the cmsDriver.py tool and its options. As an input to the tool, configuration fragments are used. Note that the fragments **must** be located in a CMSSW package directory structure like below and you **must** build your CMSSW release area after adding or changing the file. Many people have spent hours trying to figure out why their config fragment does not do what it should. **Always remember the folder structure. Always remember to build.**

```
# Copy the config fragment for showering
CONFIG_PATH=${CMSSW_BASE}/src/CMSDAS/GEN/python
mkdir -p ${CONFIG_PATH}
cp Hadronizer_TuneCP5_13TeV_generic_LHE_pythia8_cff.py ${CONFIG_PATH}
cd ${CMSSW_BASE}/src;
scram b;
cd -;
```

Have a look at the fragment.

We can run the fragment using this cmsDriver command:

```
cmsDriver CMSDAS/GEN/python/Hadronizer_TuneCP5_13TeV_generic_LHE_pythia8_cff.py \
--mc \
--eventcontent RAWSIM,LHE \
--datatier GEN,LHE \
--conditions 93X_upgrade2023_realistic_v5 \
--beamspot HLLHC14TeV \
--step GEN \
--geometry Extended2023D17 \
--era Phase2_timing \
--filein file:unweighted_events.lhe \
--fileout file:GEN.root \
--no_exec \
-n 1000
```

At first, the many options to cmsDriver may seem prohibitively complicated. However, for practical purposed one can look up the cmsDriver command for any given central production campaign and use that.

Since we specified the --no_exec option, cmsDriver does not start the actual computation, but merely generates a CMSSW python configuration that encodes what we asked it to do. It can simply be run with cmsRun:

```
cmsRun config.py
```

Without opening the output file, can you already discern some differences with respect to the LHE input file?

Inspecting the output

Since the output files are now in EDM format, we cannot use a simple text editor to inspect the events. As a simple alternative, CMSSW contains the ParticleListDrawer module:

```
cmsRun record_cfg.py inputFiles=file:/path/to/GEN.root
```

What differences do you notice compared to the LHE events you have seen before?

Also for distribution plots, we can not use the same method as above: MA5 does not read EDM files. Instead, we will use simple CMSSW modules to plot basic quantities. Copy the configuration input file to our configuration directory and build:

```
cp WjetsAnalysis_cfi.py ${CONFIG_PATH}
cd ${CMSSW_BASE}/src;
scram b;
cd -;
```

Now you can run the actual plotting and check out the resulting histograms in a ROOT file:

```
cmsRun WjetsComparisons_cfg.py
rootbrowse analyzed.root
```

As in the LHE-level case, look at the distributions and try to make sense of them. What are the differences compared to before?

Jet multiplicity merging

In the example case we have studied so far, parton radiation was modeled exclusively by P8. MG5 created a matrix element prediction for production of only a W boson on its own, and P8 then took care of adding radiated jets. Another approach is to generate the full matrix elements not only for the W final state, but also for W + 1 jet, W + 2 jet, ..., W + N jet final states. In this case, we will rely on P8 only to model jet multiplicities > N (and of course model the hadronization of outgoing partons).

Generating events

We have prepared an MG5 gridpack for a W+jet process, which you can find at this location:

```
PATH_TO_GRIDPACK="/afs/cern.ch/work/a/aalbert/public/2018-07-30_CMSDAS/wplustest_4f_012jet_LO_slc
```

To understand what is going on inside a gridpack, it is a good idea to unpack it and look at the InputCards directory, as well as the gridpack_generation.log file:

```
mkdir unpack
cd unpack
tar xf ${PATH_TO_GRIDPACK}
ls -l gridpack_generation.log InputCards
```

Compare the input cards to the ones we used before. What is different, what stayed the same? What can you learn from the log file? How many subprocesses are produced? What is the cross section? How does all of this compare to the previous case?

Let's generate some events. Again, we are going to use cmsDriver, but now with a different fragment. Compare the fragment to the earlier one to see what is different now. Before running it, make sure that it has the correct gridpack path in it.

```
cp Hadronizer_TuneCP5_13TeV_MLM_5f_max2j_qCut20_LHE_pythia8_cff.py ${CONFIG_PATH}
cd ${CMSSW_BASE}/src
scram b;
cd -

cmsDriver.py CMSDAS/GEN/python/Hadronizer_TuneCP5_13TeV_MLM_5f_max2j_qCut20_LHE_pythia8_cff.py \
--mc \
--eventcontent RAWSIM,LHE \
--datatier GEN-SIM,LHE \
--conditions 93X_mc2017_realistic_v3 \
--beamspot Realistic25ns13TeVEarly2017Collision \
--step LHE,GEN \
--nThreads 1 \
--geometry DB:Extended \
--era Run2_2017  \
--fileout file:step0.root \
-n 1000
```

At the end of the run, CMSSW automatically runs the GenXSecAnalyzer, which is a simple tool to calculate the sample cross-section, check the distribution of weights and give other useful insights. In the case of jet matching, it also gives the matching efficiencies. What are the matching efficiencies for each subprocess? How does the cross-section after matching compare to the cross-section before matching?

If you want to generate merged samples, it is important to check that the merging performs well. Since we have artificially split the physical process into energy regimes above and below QCUT, we need to make sure that the transition between the two regimes is smooth. Optimally, it should be impossible to tell from the merged sample what value of QCUT we used. To test this, we consider the distribution of the differential jet rates (DJRs). The DJRs correspond to the kt separation of the final clustering step for a given jet multiplicity. E.g. if we keep clustering an event until it has exaclty 2 jets left, and these 2 jets have a kt separation of 20 GeV, then DJR(1->2) = 20 GeV. It is called "1->2" because as we decrease the cutoff scale from >20 GeV to <20 GeV, the event turns from a 1-jet into a 2-jet event.

In the genproductions repository, there is a macro that plots these quantities for a given EDM file:

```
root -l -b -q /path/to/genproductions/bin/MadGraph5_aMCatNLO/macros/plotdjr.C\(\"step0.root\",\"d
```

Look at the resulting PDF file and check out the distributions. The contributions with different numbers of matrix element partons should sum up to give a smooth distribution.

To save on computing time in this exercise, we have pre-generated samples with different values of QCUT. What do the DJR distributions look like for the different values of QCUT? Which one would you choose?

-- AndreasAlbert - 2018-08-02

This topic: Main > CMSDAS2018_GEN_Draft
Topic revision: r4 - 2018-08-23 - AndreasAlbert