CMS MC Tutorial

Motivation: Background information and **tutorials** on

- accessing computing resources (lxplus)
- using the CMS software (CMSSW) for MC simulations within the ***CMS*** *collaboration*.

CMS Computing Concepts: This chapter introduces you to the CMS computing environment and to CMSSW, the software framework used to analyze data in CMS. You will learn how to connect to lxplus machines at CERN and to find out which CMSSW releases are available for MC/Detector simulations.

```
➤ ssh –X username@lxplus.cern.ch
➤ kinit username@CERN.CH
➤ scram list CMSSW
......
CMSSW   CMSSW_3_9_7   -> /afs/cern.ch/cms/slc5_ia32_gcc434/cms/cmssw/CMSSW_3_9_7
CMSSW   CMSSW_3_9_8   -> /afs/cern.ch/cms/slc5_ia32_gcc434/cms/cmssw/CMSSW_3_9_8
CMSSW   CMSSW_3_9_9   -> /afs/cern.ch/cms/slc5_ia32_gcc434/cms/cmssw/CMSSW_3_9_9
......
```

You will see a list with names like CMSSW_X_Y_Z with other suffixes. The names point to directories. For each listed directory, subsystems are under its `src` directory, i.e., under `CMSSW_3_9_7`. As an example, let's list the subsystems under this directory.

```
➤ ls /afs/cern.ch/cms/slc5_ia32_gcc434/cms/cmssw/CMSSW_3_9_7/src/
  Alignment      FastSimDataFormats    PerfTools          SimCalorimetry
  AnalysisAlgos  FastSimulation        CMS.PhysicsTools   SimDataFormats
  ..................
  AnalysisExamples GeneratorInterface RecoBTag            SimG4Core
```

Each subsystem contains several packages, for example in **/GeneratorInterface**

AlpgenInterface Configuration ExhumeInterface GenFilter HijingInterface MCatNLOInterface

PyquenInterface SherpaInterface AMPTInterface Core ExternalDecays Herwig6Interface HydjetInterface

PartonShowerVeto Pythia6Interface ThePEGInterface BeamHaloGenerator CosmicMuonGenerator

GenExtension HiGenCommon LHEInterface PomwigInterface Pythia8Interface

It shows all generator interface directories in the CMSSW chain. Next page we will create a new project area.

Altan Cakir @ DESY-CMS
altan.cakir@desy.de

- `cmsrel CMSSW_X_Y_Z`
- `cd CMSSW_X_Y_Z/src`
- `cmsenv`

A new project area created in your lxplus machine in the */afs/cern.ch/user/u/username*.

MC Simulations in the CMSSW: This section provides an entry point to the physics event generation and detector simulation in CMSSW. It should serve as a jump-start for people who will need to prepare configuration application for central production of the Monte Carlo samples, as representatives of their Physics groups. All interfaces with generators are in the "GeneratorInterface" subsystem in CMSSW. We will mainly consider Pythia-MC simulations and interface with other parton level generators, namely *MadGraph, Alpgen, SOFTSUSY* etc., in the CMSSW.

- **Pythia-MC:** Pythia is a general purpose event generator, containing theory and models for a number of physics aspects, including hard and soft interactions, parton distributions, initial and final state parton showers, multiple interactions, fragmentation ("Lund" model) and decays. It has been largely used in the CMS collaboration for event generation, as well as for hadronization of the parton-level events coming from one or another Matrix Element (ME) tool of interest to CMS.

In addition to the commands above:

- `addpkg GeneratorInterface/Pythia6Interface`
- OR `cvs co –r CMSSW_X_Y_Z GeneratorInterface/Pythia6Interface`
- `scram b`
- `cd GeneratorInterface/Pythia6Interface/test`

You can see some example files in the test directory.

The two major software components are

- Pythia6GeneratorFilter - full event generation
- Pythia6HadronizerFilter - processing of parton-level event by an ME generator

Both components deliver the same output: **HepMCProduct** and **GenEventInfoProduct** (these are common across CMSSW interfaces to all multi-purpose generators). For a Pythia-MC test run in the CMSSW, execute the following commands:

Generator Level Simulation:

- `cd CMSSW_X_Y_Z/src/`
- `cmsenv`
- `cp /afs/cern.ch/user/c/cakir/public/MC2011/testPythia.py .`
- `cmsRun –p testPythia.py &> log_Pythia6MC`

You will get 1000 events at generator level in the CMSSW. You can see the HepMCProduct in the **/tmp/username/** directory with the name of **testPythia_GEN.root.**

Example-1: ttbar simulation via Pythia-MC.

- `cp /afs/cern.ch/user/c/cakir/public/MC2011/Pythia_ttbar.py .`
- `cmsRun –p Pythia_ttbar.py &> log_Pythia6MC_ttbar.py`

Now let us revisit some specific details of the configuration card above. The very first line in the example

```
import FWCore.ParameterSet.Config as cms
```

is related to the mechanics of the python configuration language. The next line

```
source = cms.Source("EmptySource")
```

is a general-purpose Framework Source, that drives event loop and defines `edm::Event` principal, but does not add any branches to the `edm::Event`.

The following line

```
from Configuration.Generator.PythiaUESettings_cfi import *
```

is important because it brings **standard** pre-fabricated block to describe setting for the Underlying Event (UE), as currently approved by the CMS collaboration. Later in this write-up we will revisit several details of the UE settings.

Specifically, the configuration of the generator's module, Pythia6GeneratorFilter in this case, starts with the

```
generator = cms.EDFilter("Pythia6GeneratorFilter",
```

followed by the module's configuration parameters; remember that **"generator"** is the only label allowed for any event generation software component within CMSSW.

The following several parameters are of service nature, as they allow various levels of verbosity (here all configured to "None").

```
pythiaHepMCVerbosity = cms.untracked.bool(True),
maxEventsToPrint = cms.untracked.int32(0),
pythiaPylistVerbosity = cms.untracked.int32(12),
```

Obviously, `maxEventsToPrint` defines how many events one wants to display (starting from the 1st event; skipping events is NOT possible). Printing out Pythia6-specific event record is controlled by the `pythiaPylistVerbosity` parameters; if you are uncertain what setting to choose, please check the PYLIST(MLIST) settings from Pythia manual.

Finally, the two <u>**most essential**</u> parameters, as they determine the center of mass energy and event topology to be generated by Pythia6:

```
comEnergy = cms.double(7000.0),
```

Specific to the Pythia6Interface is that it allows to combine Pythia6 configuration cards into groups of character strings:

```
PythiaParameters = cms.PSet(
  pythiaUESettingsBlock,
  processParameters = cms.vstring(...),
  # This is a vector of ParameterSet names to be read, in this order
  parameterSets = cms.vstring('pythiaUESettings',
      'processParameters')
)
```

The cards provided in the block

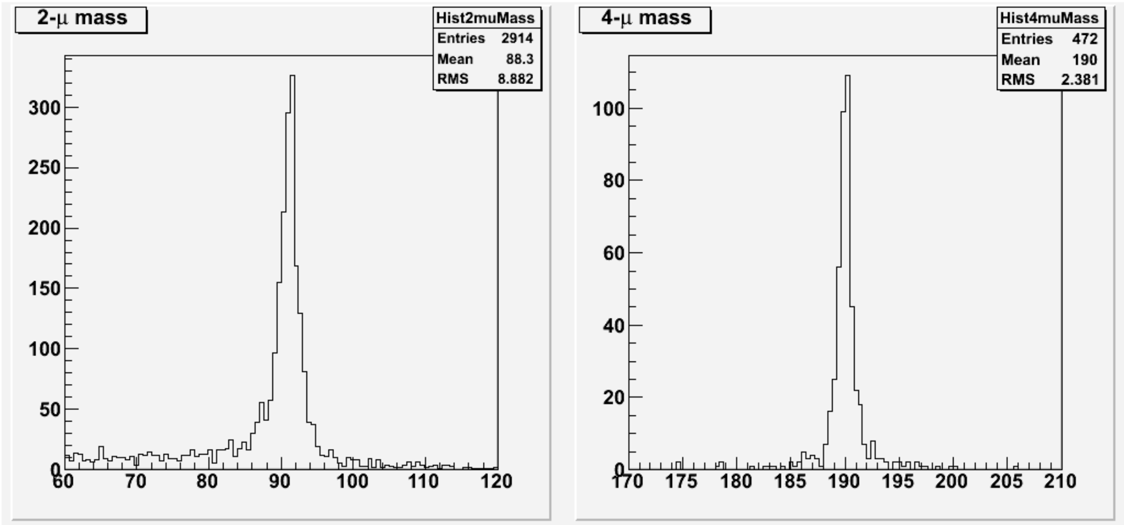
```
processParameters = cms.vstring('MSEL      = 0     ! User defined processes',
  'MSUB(81)  = 1     ! qqbar to QQbar',
  'MSUB(82)  = 1     ! gg to QQbar',
  'MSTP(7)   = 6     ! flavour = top',
  'PMAS(6,1) = 175.  ! top quark mass'),
```

are specific to the **ttbar** event generation.

Example-2: Simulation of Z/gamma*+jets via Pythia-MC and analysis of the opposite-sign same-flavor (OSSF) mass and Electron and Muon Pt distributions at the generator level in the CMSSW framework.

- **cp /afs/cern.ch/user/c/cakir/public/MC2011/Pythia_Zgammajets.py .**
- **cmsRun –p Pythia_Zgammajets.py &> log_Pythia6MC_Zgammajets.py**
- **cp /afs/cern.ch/user/c/cakir/public/MC2011/analyzeZmass.C .**
- **cp /afs/cern.ch/user/c/cakir/public/MC2011/rootlogon.C .**
- **root –L analyzeZmass.C**

Exercise: Simulate H->ZZ->4l (l=muons) events (10K) with Pythia and read the generator level opposite-sign invariant Z mass (OSSF) and the ZZ mass with the corresponding Higgs candidate.



Detector Level Simulation:

Simulating and reconstructing events with Full/Fast Simulation: the **Fast Simulation** is based on the same data formats as the output provided by the complete reconstruction of either fully-simulated or real-data events. As a consequence, high-level algorithms like ECAL clustering, particle-flow reconstruction, or b-tagging algorithms, can be used in Fast Simulation without changes. Analysis code is therefore expected to work in a transparent way with the Fast Simulation.

To create the **full configuration file** from the generation fragment we will use the same local scram area (CMSSW_3_9_7).

- `cd ~/CMSSW_3_9_7/src`
- `cmsenv`
- `cvs co Configuration/Generator`
- `scram b`
- `ls Configuration/Generator/python`

You can see all generator fragments used in the central MC production. Now we will reconstruct H->ZZ->4l (l=muons) in the CMS full detector simulation. Use the **cmsDriver.py** script to convert the generation fragment into a full configuration file, which can be run with cmsRun. We will apply the “cmsDriver” script in two steps. HLT (High Level Trigger) and RECO (Reconstruction) steps can be obtained with the following commands

- `cmsDriver.py Configuration/Generator/python/H200ZZ4L_cfi.py -s GEN,SIM,DIGI,L1,DIGI2RAW,HLT:GRun --conditions DESIGN_39_V8::All --datatier GEN-SIM-RAW --eventcontent RAWSIM -n 10 --fileout RAWSIM.root`

Altan Cakir @ DESY-CMS
altan.cakir@desy.de

This is the **HLT** step and it can be executed below:

- `cmsRun <file-name-for-hlt>.py &> log_hlt_step1`

After HLT events we should reconstruct the samples: For the **RECO** step

- `cmsDriver.py H200ZZ4L_cfi_py_GEN_SIM_DIGI_L1_DIGI2RAW_HLT -s RAW2DIGI,L1Reco,RECO VALIDATION:validation_prod --conditions DESIGN_39_V8::All --datatier GEN-SIM-RECO --eventcontent RECOSIM -n 10 --filein file:RAWSIM.root --fileout RECOSIM.root --cust_function customisePPMC`

- `cmsRun <file-name-for-reco>.py &> log_reco_step2`

Exercise: Look inside the "RECOSIM.root" file and try to find the experimental observables which are reconstructed by CMSSW_3_9_7.

FAST-SIMULATION: For the **FastSimulation** the following cmsDriver commands can be used:

- `cmsDriver.py Configuration/Generator/python/H200ZZ4L_cfi.py -s GEN,FASTSIM,HLT:GRun --pileup=NoPileUp --geometry DB -n 10 --conditions DESIGN_39_V8::All --datatier GEN-SIM-DIGI-RECO --eventcontent RECOSIM`

- `cmsRun <file-name-for-fastsim>.py &> log_fastsim_3_9_7`

Pythia Hadronization for external Matrix-Element output:

Pythia-MC has been largely used for hadronization of the parton-level events coming from one or another Matrix Element (ME) tool of interest to CMS. Here are the basic examples on how these files, so called Les Houches Events (LHE), SUSY Les Houches Accord (SLHA) etc., can be read by the Pythia interface.

Pythia Hadronization for LHE file: There is a dedicated module Pythia6HadronizerFilter, which is different from Pythia6GeneratorFilter (previous examples) used for full event generation.

- `cd GeneratorInterface/Pythia6Interface/test`
- `less Py6HadFilter_cfg.py` OR `less Py6HadFilter_mgmatching_cfg.py`
- `cmsRun Py6HadFilter_mgmatching_cfg.py &> log_LHE_MadGraph_MGMMatching`

These files are set for Pythia Hadronization and they read LHE flies from MadGraph simulation.

Pythia Hadronization for an SLHA spectrum file: For an accurate calculation of SUSY particle masses, you can use a dedicated program (ISASUGRA, SUSPECT, SOFTSUSY, ...) to calculate the spectrum and let Pythia6 use this to generate the events. The SLHA file reading option is available for both Pythia6-based generator modules, Pythia6GeneratorFilter and Pythia6HadronizerFilter. This means that, in addition to the Pythia6 SUSY event generation, one can also simulate events from external generator partons.

- `cd GeneratorInterface/Pythia6Interface/test`

- `less Py6GenFilter_SLHA_cfg.py`

You can see the examples GenFilter simulation for SLHA format in the Pythia interface. We will see how these SLHA files are produced via certain external MC generators/spectrum calculators in the next section.

How to generate SUSY samples using an external SLHA spectrum file:

For this purpose we need to install SOFTSUSY and SUSYHIT programs: For **SOFTSUSY**

- `mkdir SLHA_Production`
- `cd SLHA_Production`
- `wget http://www.hepforge.org/archive/softsusy/softsusy-3.1.6.tar.gz`
- `tar -zxvf softsusy-3.1.6.tar.gz`
- `./configure`
- `make`
- `make install`
- `ls spsSLHAfiles/`

You can see input datacards (*.in) for SOFTSUSY run and an output datacard (*.out) as SLHA format. We will use one ".out" file in the "`Py6GenFilter_SLHA_cfg.py`" Pythia simulation card in order to generate SUSY samples in the CMSSW. The simulation syntax for a specific SUSY point is the following:

- `./softpoint.x leshouches < spsSLHAfiles/<your-file>.in > spsSLHAfiles/<yourout>.out`

Exercise: Simulate 12 different m0-m1/2 points (starting from m0=50GeV and m1/2=100) with 50GeV steps in the **SOFTSUSY**. Read these files via **Pythia** simulation and reconstruct them with **FastSimulation** in the CMSSW. Finally analyze them related to their cross-section values with the corresponding m0-m1/2 values on the SUSY plane. (Details Francesco Costanza)

Appendix: Writing your own (ED)Analyzer

You will see the first steps of interacting with the CMS framework and how to write a module where you can put your analysis code.

- `cd CMSSW_3_9_7/src`
- `cmsenv`
- `mkdir DESYMC2011`
- `cd DESYMC2011`
- `mkedanlzr DemoMC2011`
- `cd DemoMC2011`
- `scram b`

The `mkedanlzr` script has generated an example python configuration file `demoMC2011_cfg.py` in the `DemoMC2011` directory. Open the file using your favorite text editor and change the data source file. In order to get informations and fill the histograms, insert the necessary commands in reference 4.

References

1. https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBook
2. https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuidePythia6Interface
3. https://twiki.cern.ch/twiki/bin/view/CMS/GeneratorProduction
4. https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookWriteFrameworkModule