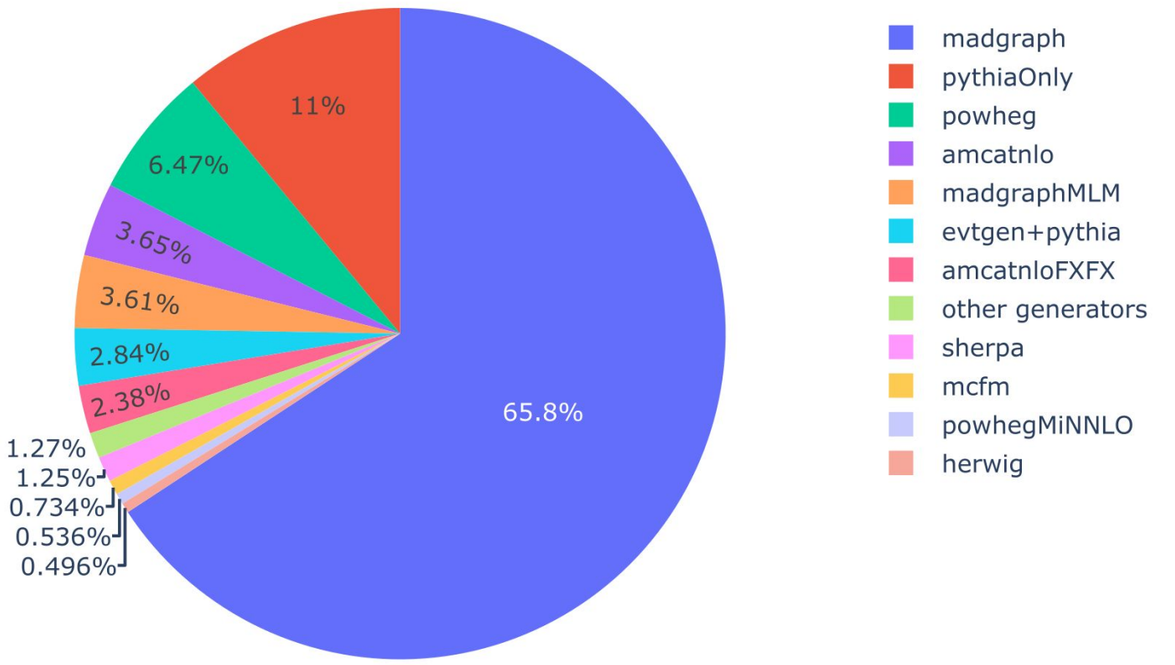
Advanced Generation + Hands-on

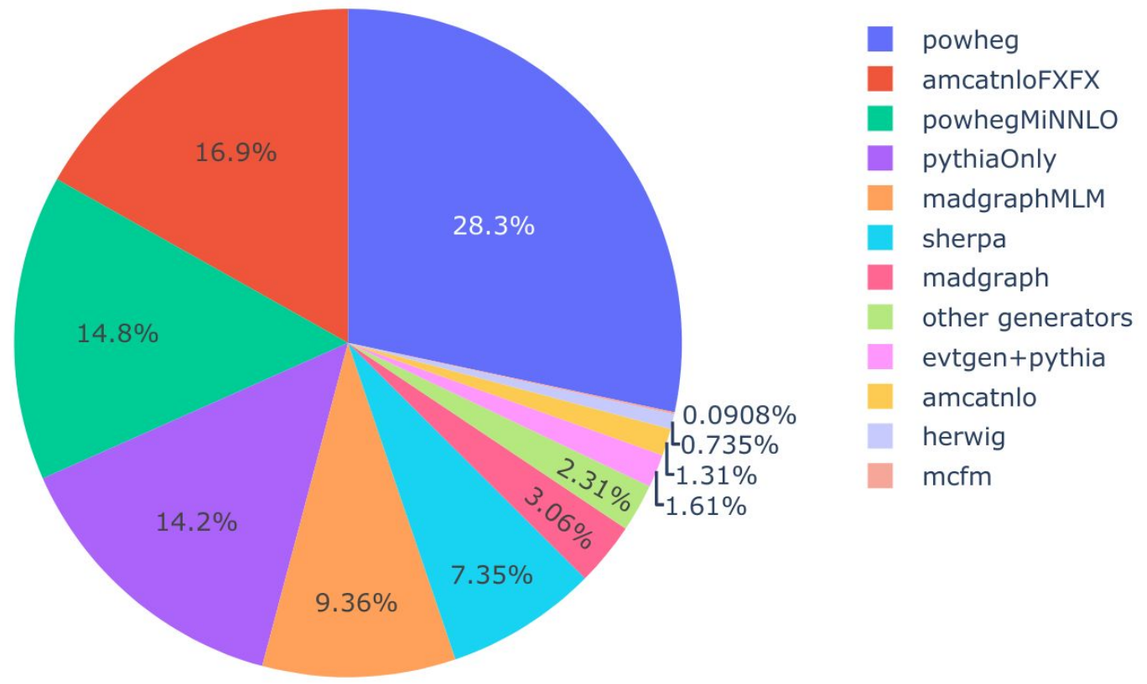
Qiang Li, Meng Lu

20 Dec 2021

Generators inside CMS



By #samples



By #events

There are many generators used inside CMS, this talk will include several of them:

- Madgraph
- Pythia
- sherpa

Madgraph

In yesterday's generator course, we generate the DY process using syntax "generate p p > lep+ lep-", this process includes contribution from on-shell Z boson, off-shell Z boson, virtual photon and their interference. Can we separate these contributions? Let's say "p p > e+ e-" (with 10k events each)

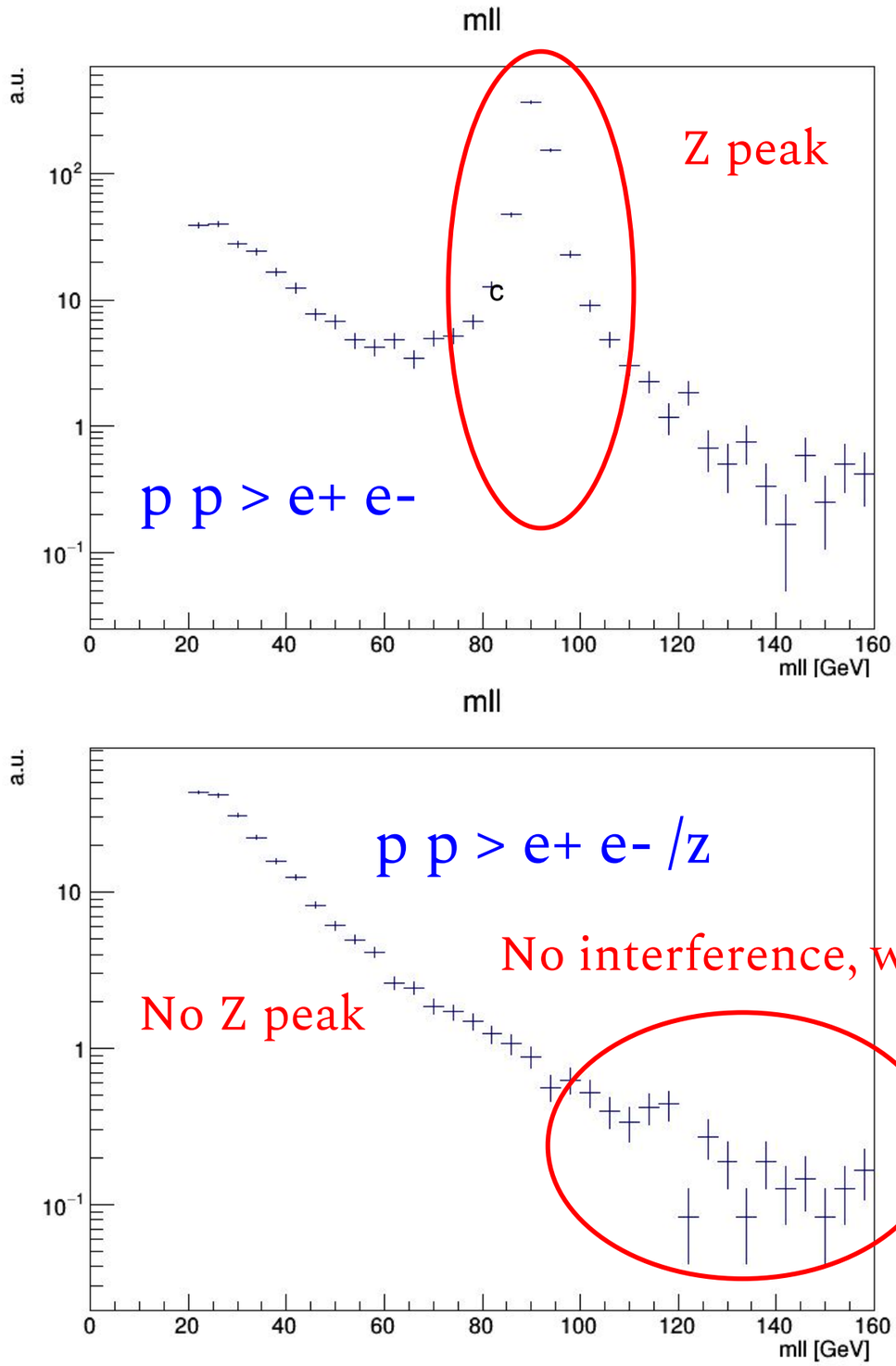
1. p p > e+ e- (all contributions)
2. p p > z, z > e+ e- (z is on-shell)
3. p p > e+ e- $z (forbids s-channel z to be on-shell)
4. p p > e+ e- /z (forbids any z)

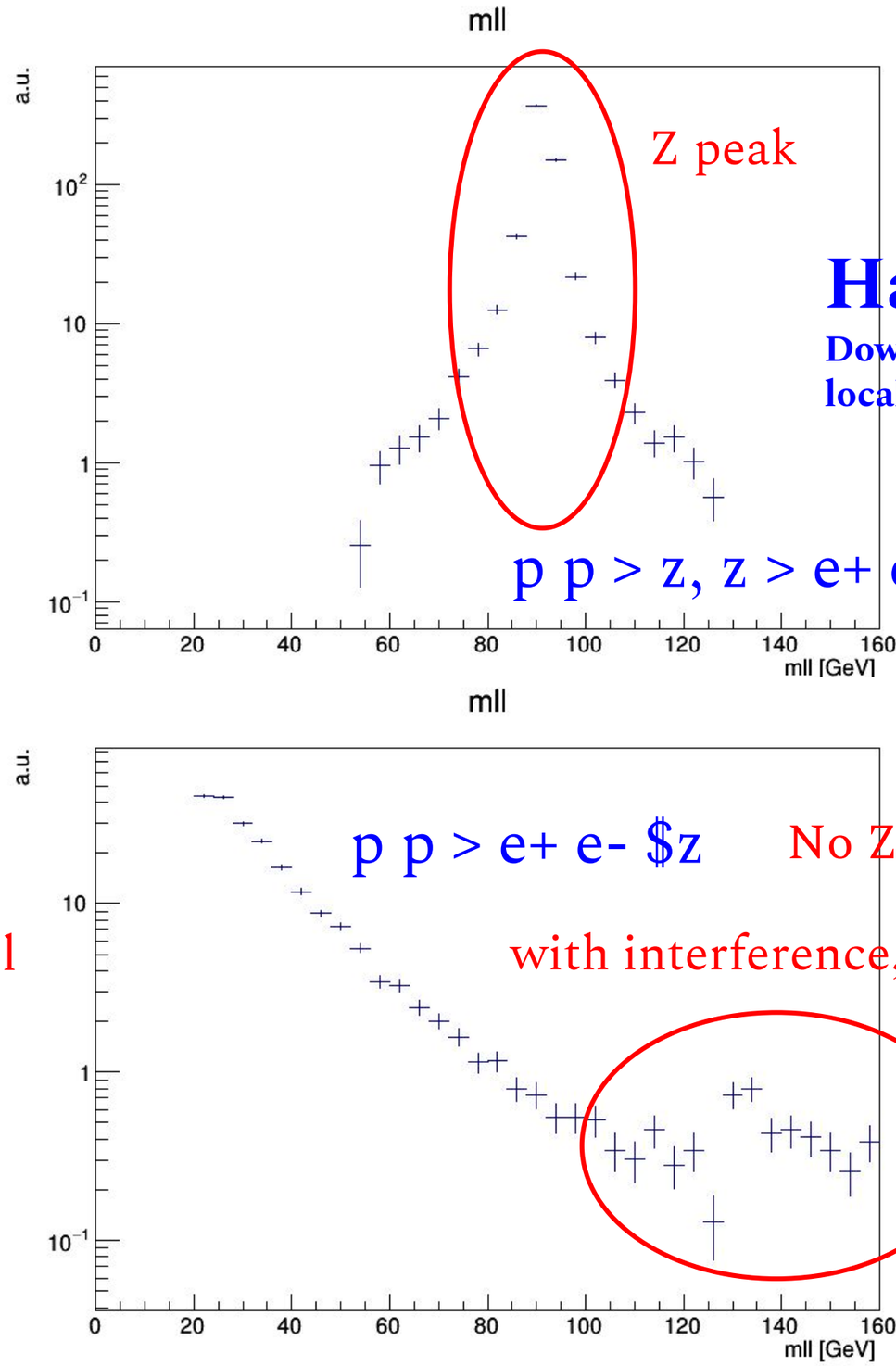
Hands-on

From the definition above: process 1 = process 2 + process 3

	Process 1	Process 2 (False = cut_decays)	Process 2 (True = cut_decays)	Process 3	Process 4
XS (with default cut)	843.0 pb	1420.7 pb	631.66 pb	213.0511 pb	207.76 pb
contributions	Onshell Z Offshell Z Virtual photon Interference	Onshell Z (but no cut on the decayed electrons)	Onshell Z (with cuts on the decayed electrons)	Offshell Z Virtual photon Interference	Virtual photon

DY: syntax

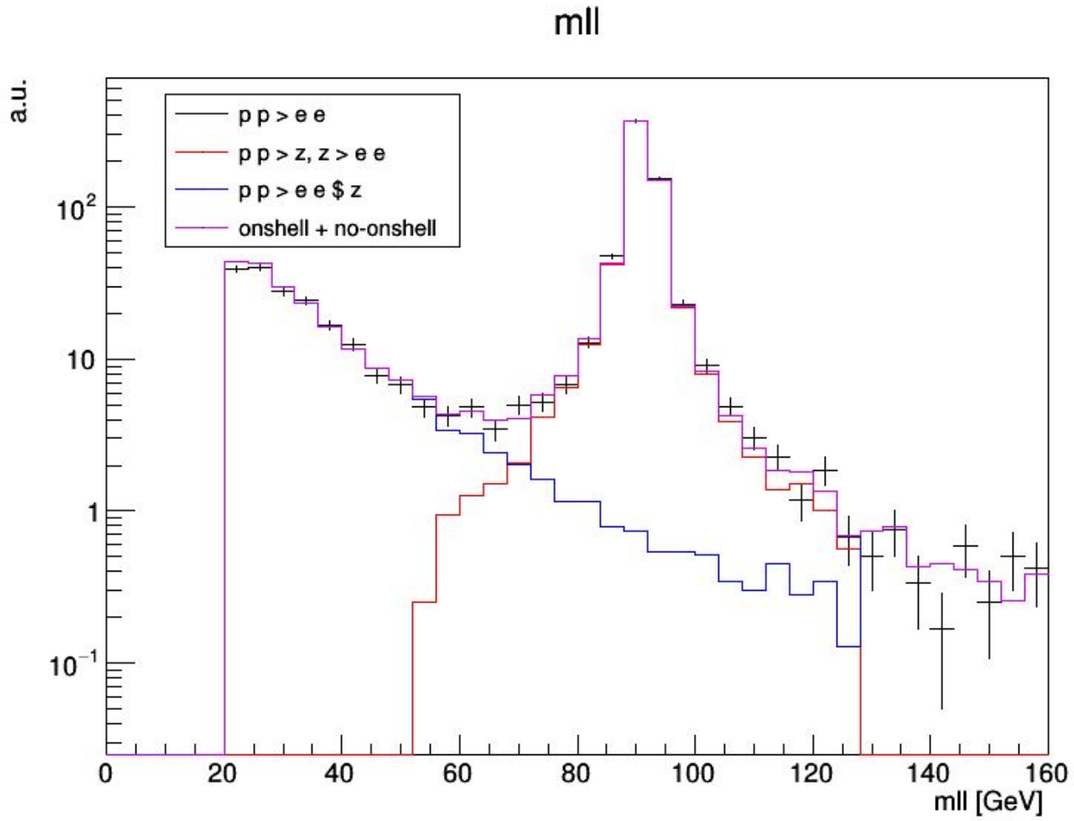




Hands-on

Download plot to your local

DY: syntax



Hands-on

Download plot to your local

On-shell contribution + no-onshell contribution is consistent with the overall contribution.

Code:
/data/pubfs/pku_visitor/public_write/generator_resource/ana.py
: modify the "libExRootPath" to your path and the corresponding root file path

⏭ cd /YOUR/MG/PATH/bin/dy_all/Events/run_01

⏭ ../../../../ExRootAnalysis/ExRootLHEFConverter unweighted_events.lhe dy.root

```
DY  ana.py    dy_all   dy_noz  mg5       mll_all.png   mll_noz.png  py.py
ZA  comp.png  dy_nosz  dy_zee  mg5_aMC   mll_nosz.png  mll_zee.png
```

DY + 1 additional parton (LO)

Currently we only look on the simplest process, how could we run some things nontrivial.

```
generate p p > e+ e- @0
add process p p > e+ e- j @1
output dy_01j
launch dy_01j
```

```
[lum@farm SubProcesses]$ ls
addmothers.f  done            lhe_event_infos.inc  P1_gq_llq             results.dat     subproc.txt
cluster.f     dummy_fct.f     makefile             P1_qq_llg             reweight.f      sudakov.inc
cluster.inc   epsterms.f      maxconfigs.inc       proc_characteristics  run_config.inc  survey.sh
coupl.inc     finiteterms.f   maxparticles.inc     procdef_mg5.dat       run.inc         symmetry.f
cuts.f        genps.f         message.inc          projection.f          setcuts.f       transform.f
cuts.inc      genps.inc       MGVersion.txt        randinit              setscales.f     transformint.f
dipole.inc    idenparts.f     myamp.f              refine.sh             shrinktops.f    unwgt.f
dipolesub.f   initcluster.f   P0_qq_ll             refine_splitted.sh    subproc.mg
```

```
#  Number of Events        :        10000
#  Integrated weight (pb)  :        1019.1613
```

Download plot to your local

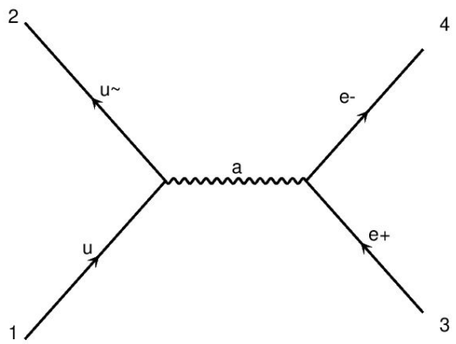


diagram 1 QCD=0, QED=2

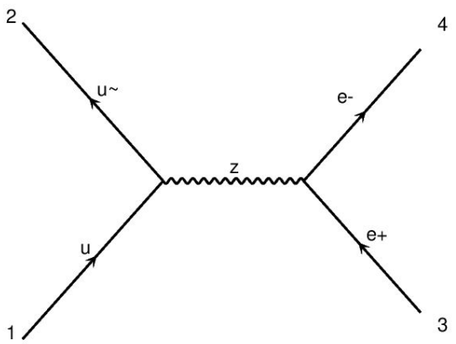


diagram 2 QCD=0, QED=2

P0_qq_ll/matrix1.ps

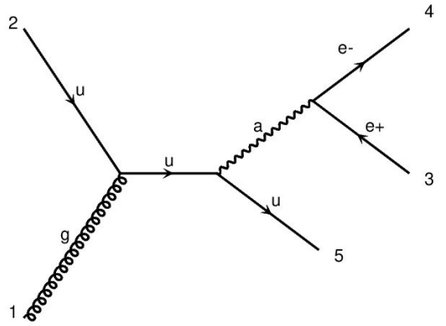


diagram 1 QCD=1, QED=2

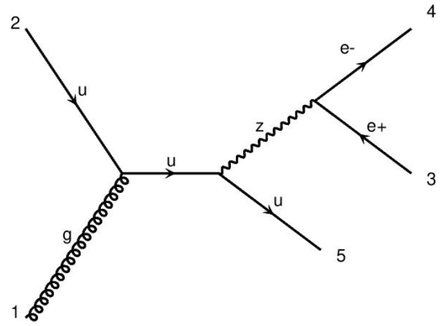


diagram 2 QCD=1, QED=2

P1_gq_llq/matrix1.ps

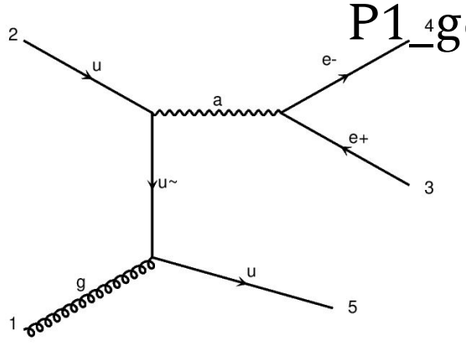


diagram 3 QCD=1, QED=2

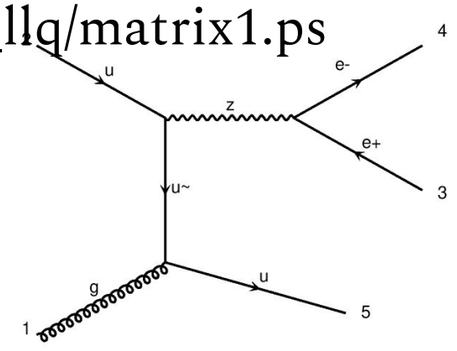


diagram 4 QCD=1, QED=2

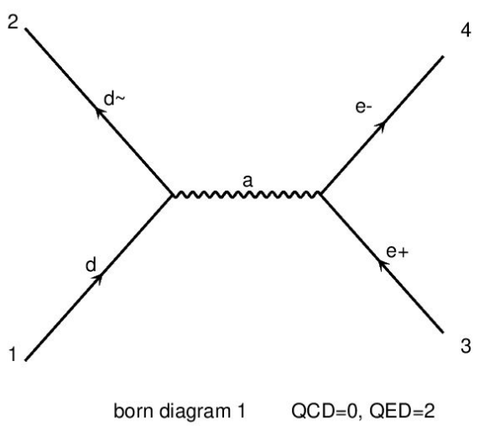
DY + 1 additional parton (NLO)

We could use another syntax to generate process with 1 additional parton

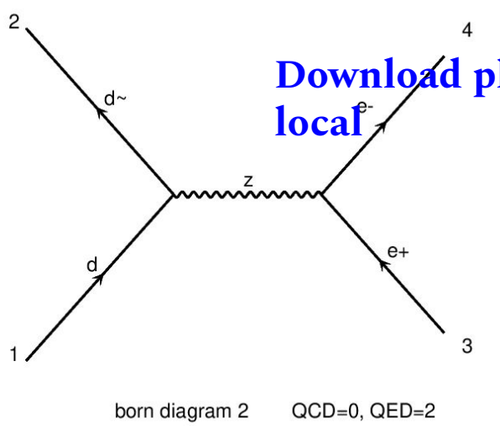
```
generate p p > e+ e- [QCD]
output dy_nlo
launch dy_nlo
```

```
[lum@farm dy_ee_nlo]$ ls SubProcesses/
add_write_info.f                FKSParamReader.f          nevents_unweighted.orig
ajob_template                   FKS_params.dat            open_output_files_dummy.f
analyse_opts                    FKSParams.inc             open_output_files.f
analysis_dummy.f                fks_powers.inc            P0_ddx_epem
analysis_lhe.f                  fks_Sij.f                 P0_dxd_epem
appl_common.inc                 fks_singular.f            P0_uux_epem
appl_interface.cc               genps_fks.f               P0_uxu_epem
```

```
[lum@farm dy_ee_nlo]$ ls SubProcesses/P0_ddx_epem/*.ps
SubProcesses/P0_ddx_epem/born.ps      SubProcesses/P0_ddx_epem/matrix_2.ps
SubProcesses/P0_ddx_epem/matrix_1.ps  SubProcesses/P0_ddx_epem/matrix_3.ps
```



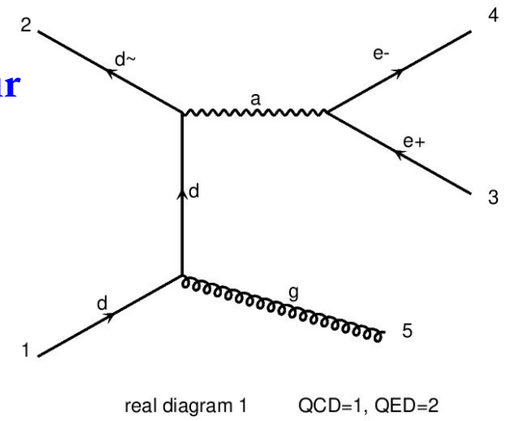
born diagram 1 QCD=0, QED=2



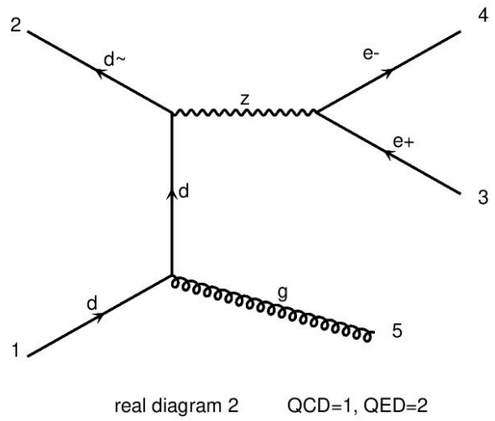
born diagram 2 QCD=0, QED=2

P0_ddx_epem/born.ps

Download plot to your local



real diagram 1 QCD=1, QED=2



real diagram 2 QCD=1, QED=2

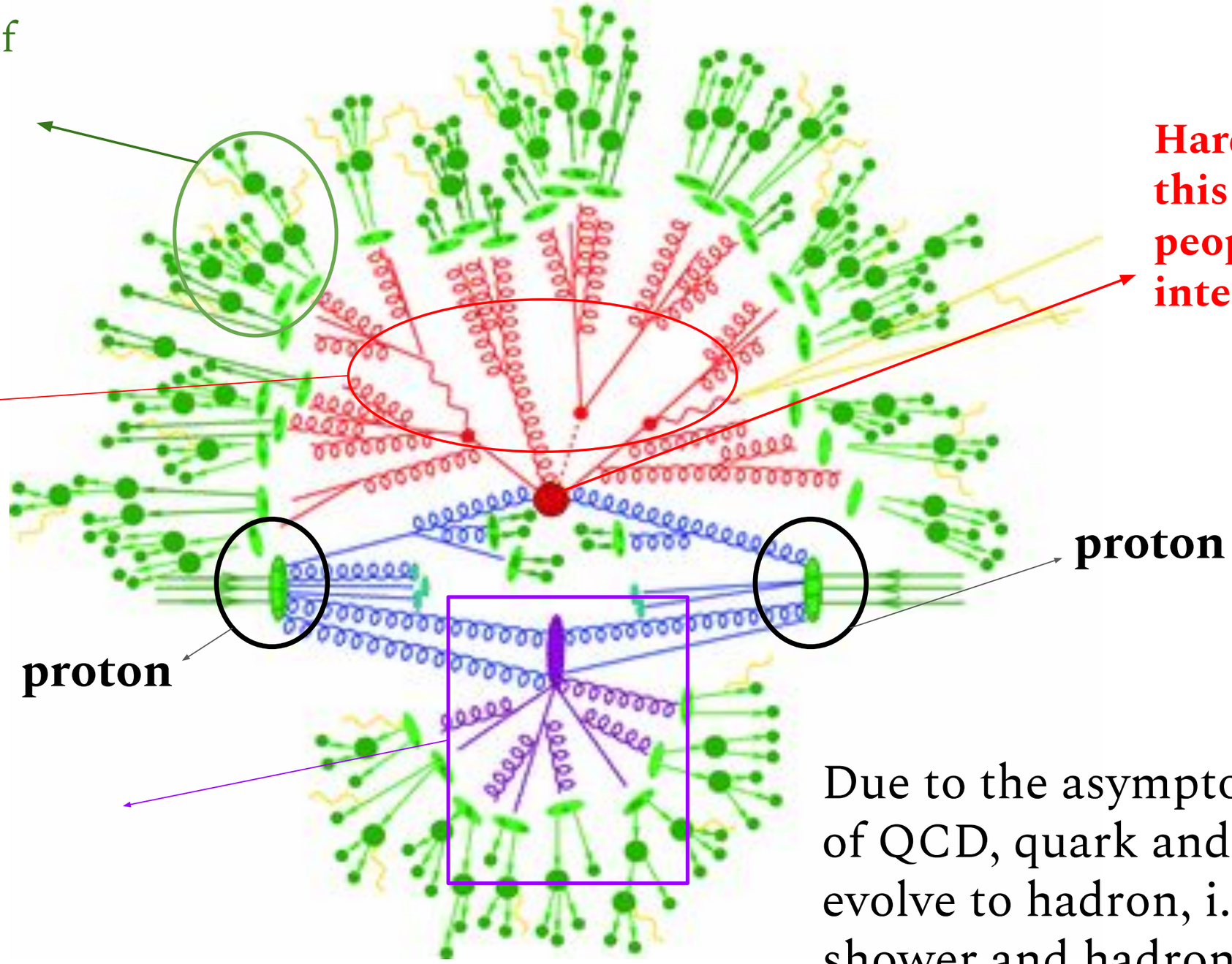
P0_ddx_epem/matrix_1.ps

The "[QCD]" in the syntax means the process will include tree-level LO, real emission, virtual correction

Go beyond of the parton level

Hadronization of parton shower particles and further decay

Parton shower: the evolution of the particles from Hard process



Hard process, this is what people usually interested in.

Double parton scattering

Due to the asymptotic freedom of QCD, quark and gluon will evolve to hadron, i.e., the parton shower and hadronization.

Embedded pythia in MG

Default env (no cmsenv)

- ./mg5_aMC
- install hepmc (output format of pythia)
- install pythia8
- generate p p > e+ e-
- output dy_shower
- launch dy_shower (change event number to 100 for test)
- shower=Pythia8

Hands-on

```
[lum@atlas bin]$ ls tes/Events/run_01/
run_01_tag_1_banner.txt  tag_1_djrs.dat  tag_1_pythia8.cmd              tag_1_pythia8.log
run_shower.sh            tag_1_pts.dat   tag_1_pythia8_events.hepmc.gz  unweighted_events.lhe.gz
```

unweighted_events.lhe is again the LHE file, tag_1_pythia8_events.hepmc is the output after parton shower.

Pythia output (HepMC format)

The detail information of HepMC format is here, the output has 207738 lines for 100 events.

The beginning of the first event

```
HepMC::Version 2.06.09
HepMC::IO_GenEvent-START_EVENT_LISTING
E 0 -1 -1.0000000000000000e+00 -1.0000000000000000e+00 -1.0000000000000000e+00 9999 0 381 1 2 0 2 8.432100
0000000015e+02 8.432100000000019e+00
N 2 "Weight" "Weight_MERGING=0.000"
U GEV MM
C 8.432100000000019e+00 8.432100000000015e+02
V -1 0 0 0 0 0 0 1 0
P 3 -2 0 0 4.2034825773730255e+01 4.2034825773730255e+01 0 21 0 0 -3 1 2 501
V -2 0 0 0 0 0 0 2 0
P 4 2 0 0 -5.0470080641030258e+01 5.0470080641030258e+01 0 21 0 0 -3 1 1 501
P 9 -2 3.4540503568452303e+00 7.3427246483307149e+00 -3.3992532555941409e+01 3.4949209414456895e+01 3.3000
000000000002e-01 43 0 0 -10 1 2 502
V -3 0 0 0 0 0 0 1 0
P 5 23 0 0 -8.4352548672999994e+00 9.2504906415167355e+01 9.2119510344999995e+01 22 0 0 -6 0
V -4 0 0 0 0 0 0 2 0
P 6 -2 3.7747582837255322e-15 7.5495165674510645e-15 4.2034825773730262e+01 4.2034825773730248e+01 0 42 0
0 -1 1 2 501
```

```
P 691 22 3.5096015001430336e-02 1.1922175095798168e-01 1.0961289665548738e+02 1.0961296711047800e+02 0 1 0
 0 0 0
P 692 22 -1.4527302433576082e-02 -1.6628097146523718e-02 4.2280898493148005e+00 4.2281475033411011e+00 0 1
 0 0 0 0
E 1 -1 -1.0000000000000000e+00 -1.0000000000000000e+00 -1.0000000000000000e+00 9999 0 1225 1 2 0 2 8.43210
0000000015e+02 8.432100000000019e+00
```

The beginning of the second event

Pythia output (HepMC format)

```
HepMC::Version 2.06.09
HepMC::IO_GenEvent-START_EVENT_LISTING
E 0 -1 -1.0000000000000000e+00 -1.0000000000000000e+00 -1.0000000000000000e+00 9999 0 381 1 2 0 2 8.432100
0000000015e+02 8.432100000000019e+00
N 2 "Weight" "Weight_MERGING=0.000"
U GEV MM
C 8.432100000000019e+00 8.432100000000015e+02
V -1 0 0 0 0 0 0 1 0
P 3 -2 0 0 4.2034825773730255e+01 4.2034825773730255e+01 0 21 0 0 -3 1 2 501
V -2 0 0 0 0 0 0 2 0
P 4 2 0 0 -5.0470080641030258e+01 5.0470080641030258e+01 0 21 0 0 -3 1 1 501
P 9 -2 3.4540503568452303e+00 7.3427246483307149e+00 -3.3992532555941409e+01 3.4949209414456895e+01 3.3000
000000000002e-01 43 0 0 -10 1 2 502
V -3 0 0 0 0 0 0 1 0
P 5 23 0 0 -8.4352548672999994e+00 9.2504906415167355e+01 9.2119510344999995e+01 22 0 0 -6 0
V -4 0 0 0 0 0 0 2 0
P 6 -2 3.7747582837255322e-15 7.5495165674510645e-15 4.2034825773730262e+01 4.2034825773730248e+01 0 42 0
0 -1 1 2 501
```

P 3 -2 0 0 4.2034825773730255e+01 4.2034825773730255e+01 0 21 0 0 -3 1 2 501

P - GenParticle information

- **int**: *barcode*
- **int**: *PDG id*
- **double**: *px*
- **double**: *py*
- **double**: *pz*
- **double**: *energy*
- **double**: *generated mass*
- **int**: *status code*
- **double**: *Polarization theta*
- **double**: *Polarization phi*
- **int**: *barcode for vertex that has this particle as an incoming particle*
- **int**: *number of entries in flow list (may be zero)*
- **int, int**: *optional code_index and code for each entry in the flow list*

Pythia output (HepMC format)

```
HepMC::Version 2.06.09
HepMC::IO_GenEvent-START_EVENT_LISTING
E 0 -1 -1.0000000000000000e+00 -1.0000000000000000e+00 -1.0000000000000000e+00 9999 0 381 1 2 0 2 8.432100
0000000015e+02 8.432100000000019e+00
N 2 "Weight" "Weight_MERGING=0.000"
U GEV MM
C 8.432100000000019e+00 8.432100000000015e+02
V -1 0 0 0 0 0 0 1 0
P 3 -2 0 0 4.2034825773730255e+01 4.2034825773730255e+01 0 21 0 0 -3 1 2 501
V -2 0 0 0 0 0 0 2 0
P 4 2 0 0 -5.0470080641030258e+01 5.0470080641030258e+01 0 21 0 0 -3 1 1 501
P 9 -2 3.4540503568452303e+00 7.3427246483307149e+00 -3.3992532555941409e+01 3.4949209414456895e+01 3.3000
000000000002e-01 43 0 0 -10 1 2 502
V -3 0 0 0 0 0 0 1 0
P 5 23 0 0 -8.4352548672999994e+00 9.2504906415167355e+01 9.2119510344999995e+01 22 0 0 -6 0
V -4 0 0 0 0 0 0 2 0
P 6 -2 3.7747582837255322e-15 7.5495165674510645e-15 4.2034825773730262e+01 4.2034825773730248e+01 0 42 0
0 -1 1 2 501
```

HepMC output

```
<event>
 5      1 +8.4321000e+02 9.21195100e+01 7.54677100e-03 1.29779900e-01
       -2 -1    0    0    0  501 -0.0000000000e+00 +0.0000000000e+00 +4.2034825774e+01 4.2034825774e+01 0.
0000000000e+00 0.0000e+00 1.0000e+00
        2 -1    0    0  501    0 +0.0000000000e+00 -0.0000000000e+00 -5.0470080641e+01 5.0470080641e+01 0.
0000000000e+00 0.0000e+00 -1.0000e+00
       23  2    1    2    0    0 +0.0000000000e+00 +0.0000000000e+00 -8.4352548673e+00 9.2504906416e+01 9.
2119510345e+01 0.0000e+00 0.0000e+00
      -11  1    3    3    0    0 -4.2789482875e+01 -1.7045743714e+01 -4.1555378681e+00 4.6246791440e+01 0.
0000000000e+00 0.0000e+00 1.0000e+00
       11  1    3    3    0    0 +4.2789482875e+01 +1.7045743714e+01 -4.2797169992e+00 4.6258114976e+01 0.
0000000000e+00 0.0000e+00 -1.0000e+00
```

LHE

Standalone Pythia

Pythia could be used for parton shower and also generate hard scattering event.
Standalone Pythia Installation: we use HepMC for the pythia output. We can use the pythia installed from MG.

Hands-on

Default env (no cmsenv)

- cd /YOUR/MGPATH/HEPTools/pythia8/share/Pythia8/examples/
- cp /data/pubfs/pku_visitor/public_write/generator_resource/main90.cc .
- cp /data/pubfs/pku_visitor/public_write/generator_resource/main94.cc .

Modify line 65 of Makefile, add “main90 main94”

```
# HEPMC2 or HEPMC3 (use HEPMC3 if both).
main41 main42 main43 main44 main45 main85 main86 main87 main88 main89 main90 main94 main280:\
```

main90.cc use dy.lhe as input, and perform parton shower

- make main90
- ./main90

Standalone Pythia

```
HepMC::Version 2.06.09
HepMC::IO_GenEvent-START_EVENT_LISTING
E 0 -1 -1.0000000000000000e+00 -1.0000000000000000e+00 -1.0000000000000000e+00 9999 0 381 1 2 0 2 8.432100
0000000015e+02 8.4321000000000019e+00
N 2 "Weight" "Weight_MERGING=0.000"
U GEV MM
C 8.4321000000000019e+00 8.4321000000000015e+02
V -1 0 0 0 0 0 0 1 0
P 3 -2 0 0 4.2034825773730255e+01 4.2034825773730255e+01 0 21 0 0 -3 1 2 501
V -2 0 0 0 0 0 0 2 0
P 4 2 0 0 -5.0470080641030258e+01 5.0470080641030258e+01 0 21 0 0 -3 1 1 501
P 9 -2 3.4540503568452303e+00 7.3427246483307149e+00 -3.3992532555941409e+01 3.4949209414456895e+01 3.3000
000000000002e-01 43 0 0 -10 1 2 502
V -3 0 0 0 0 0 0 1 0
P 5 23 0 0 -8.4352548672999994e+00 9.2504906415167355e+01 9.2119510344999995e+01 22 0 0 -6 0
V -4 0 0 0 0 0 0 2 0
P 6 -2 3.7747582837255322e-15 7.5495165674510645e-15 4.2034825773730262e+01 4.2034825773730248e+01 0 42 0
0 -1 1 2 501
```

Embedded pythia

```
HepMC::Version 2.06.09
HepMC::IO_GenEvent-START_EVENT_LISTING
E 0 -1 9.2119510000000005e+01 1.2977990000000000e-01 7.5467709999999999e-03 9999 0 428 1 2 0 1 8.432100000
0000015e+02
N 1 "Weight"
U GEV MM
C 8.4321000000000015e+02 8.4321000000000015e+02
F -2 2 6.4668962729230763e-03 7.7646277909230775e-03 9.2119510000000005e+01 0 0 0 0
V -1 0 0 0 0 0 0 2 0
P 3 -2 0 0 4.2034825773730255e+01 4.2034825773730255e+01 0 21 0 0 -3 1 2 501
P 9 21 -1.7845930638254352e+01 -1.1258677575325635e+01 2.0893087514008862e+01 2.9694379383286648e+01 0 43
0 0 -30 2 1 501 2 502
V -2 0 0 0 0 0 0 1 0
P 4 2 0 0 -5.0470080641030258e+01 5.0470080641030258e+01 0 21 0 0 -3 1 1 501
V -3 0 0 0 0 0 0 1 0
P 5 23 0 0 -8.4352548672999994e+00 9.2504906415167355e+01 9.2119510344999995e+01 22 0 0 -6 0
V -4 0 0 0 0 0 0 2 0
P 6 -2 -3.5527136788005009e-15 0 7.3759919891666755e+01 7.3759919891666755e+01 0 41 0 0 -1 1 2 502
```

Standalone pythia

Standalone Pythia: hard scattering

Pythia can not only for parton shower, but also hard scattering event.

- make main94 (this is similar with main02.cc)
- ./main94

Hands-on

root [0] 1.2200541040873152e+02+1.8065985822583283e+01
(double) 140.07140

WeakSingleBoson:ffbar2gmZ process

```
HepMC::Version 2.06.09
HepMC::IO_GenEvent-START_EVENT_LISTING
E 0 -1 9.3896709521103006e+01 1.2939916608378665e-01 7.8201047882192239e-03 221 0 233 1 2 0 1 1.0000000000
000000e+00
N 1 "Weight"
U GEV MM
C 1.1704359412386539e+04 1.1704359412386539e+04
F 1 -1 1.2449531674360359e-01 1.8434679410799269e-02 9.3896709521103006e+01 3.2631317026899648e-01 5.49104
00223817590e-01 0 0
V -1 0 0 0 0 0 0 1 0
P 3 1 0 0 1.2200541040873152e+02 1.2200541040873152e+02 0 21 0 0 -3 1 1 101
V -2 0 0 0 0 0 0 2 0
P 4 -1 0 0 -1.8065985822583283e+01 1.8065985822583283e+01 0 21 0 0 -3 1 2 101
P 9 21 1.8375055415314758e+01 1.7973774214634386e+00 -1.4514866499640334e+01 2.3485199096738690e+01 0 43 0
 0 -10 2 1 101 2 103
V -3 0 0 0 0 0 0 1 0
P 5 23 0 0 1.0393942458614823e+02 1.4007139623131479e+02 9.3896709521103006e+01 22 0 0 -6 0
V -4 0 0 0 0 0 0 1 0
P 6 1 5.3290705182007514e-15 6.6613381477509392e-16 1.2200541040873151e+02 1.2200541040873151e+02 0 42 0 0
 -1 1 1 101
```

Desired method for production: gridpack

We know that MC samples are produced according to the phase space integration of process. It's too slow to generate LHE from MG promptly, and the procedure is not easy to track.

Gridpack pre-calculate the phase space integration, tar all the related information into a single package:

- One-time calculation for phase space integration
- Could be used repeatably
- Easy to track the input cards

Gridpack: MadGraph

- git clone https://github.com/cms-sw/genproductions.git
- cd genproductions/bin/MadGraph5_aMCatNLO/

No Hands-on

```
[lumeng@wz MadGraph5_aMCatNLO]$ ls
cards                          submit_cmsconnect_gridpack_generation.sh
gridpack_generation.sh         submit_cmsconnect_gridpack_generation_singlejob.sh
macros                         submit_condor_gridpack_generation.sh
patches                        submit_gridpack_generation_local.sh
PLUGIN                         submit_gridpack_generation.sh
runcmsgrid_LO.sh               Utilities
runcmsgrid_NLO.sh
```

Input cards: /data/pubfs/pku_visitor/public_write/generator_resource/dy

- ./gridpack_generation.sh dy_5f_LO_MLM dy/

The “dy_5f_LO_MLM” is the process name, i.e., the prefix name of cards in dy/, this process is time consuming.

You can refer the CMS twiki for gridpack generation after you own CMS account,

Gridpack: MadGraph

```
[lum@atlas MadGraph5_aMCatNLO]$ ls
cards
dy
dy_5f_LO_MLM
dy_5f_LO_MLM.log
dy_5f_LO_MLM_slc7_amd64_gcc700_CMSSW_10_6_19_tarball.tar.xz
gridpack_generation.sh
macros
patches
PLUGIN
runcmsgrid_LO.sh
runcmsgrid_NLO.sh
submit_cmsconnect_gridpack_generation.sh
submit_cmsconnect_gridpack_generation_singlejob.sh
submit_condor_gridpack_generation.sh
submit_gridpack_generation_local.sh
submit_gridpack_generation.sh
Utilities
```

No Hands-on

The dy_5f_LO_MLM_slc7_amd64_gcc700_CMSSW_10_6_19_tarball.tar.xz is the so-called gridpack. Here we use script “gridpack_generation.sh” to produce the gridpack locally. Those scripts start with “submit” are for job submitted to clusters. e.g.:
./submit_condor_gridpack_generation.sh <name of process card without _proc_card.dat> <folder containing cards relative to current location>

Gridpack to sample

```
[lum@atlas temp]$ ls
dy_5f_LO_MLM_slc7_amd64_gcc700_CMSSW_10_6_19_tarball.tar.xz
[lum@atlas temp]$ tar xf dy_5f_LO_MLM_slc7_amd64_gcc700_CMSSW_10_6_19_tarball.tar.xz
[lum@atlas temp]$ ls
dy_5f_LO_MLM_slc7_amd64_gcc700_CMSSW_10_6_19_tarball.tar.xz  InputCards  mgbasedir  runcmsgrid.sh
gridpack_generation.log                                      merge.pl    process
```

- cp /data/pubfs/pku_visitor/public_write/generator_resource/dy_5f_LO_MLM_slc7_amd64_gcc700_CMSSW_10_6_19_tarball.tar.xz /YOUR/WORKING/PATH/
- tar xf dy_5f_LO_MLM_slc7_amd64_gcc700_CMSSW_10_6_19_tarball.tar.xz
- ./runcmsgrid.sh 100

But it's more useful to use the gridpack for events production.

Hands-on

- cd /YOUR/PATH
- source /cvmfs/cms.cern.ch/cmsset_default.sh
- cmsrel CMSSW_10_6_19
- cd CMSSW_10_6_19/src
- cmsenv
- mkdir -p Configuration/GenProduction/python/
- cp /data/pubfs/pku_visitor/public_write/generator_resource/dy_fragment.py Configuration/GenProduction/python/
- scram b
- cmsDriver.py Configuration/GenProduction/python/dy_fragment.py --python_filename dy_cfg.py --eventcontent RAWSIM,LHE --customise Configuration/DataProcessing/Utils.addMonitoring --datatier GEN,LHE --fileout file:dy.root --conditions 106X_mc2017_realistic_v6 --beamspot Realistic25ns13TeVEarly2017Collision --customise_commands process.source.numberEventsInLuminosityBlock="cms.untracked.uint32(250)" --step LHE,GEN --geometry DB:Extended --era Run2_2017 --no_exec --mc -n 200

Gridpack to sample

```
customising the process with addMonitoring from Configuration/DataProcessing/Utils
Config file dy_cfg.py created
[lum@atlas src]$ ls
Configuration  dy_cfg.py
```

Hands-on

⏭ cmsRun dy_cfg.py

```
[lum@atlas src]$ cmsRun dy_cfg.py

   ________________________________________
         Running Generic Tarball/Gridpack

   ________________________________________
gridpack tarball path = /home/pku/lum/genproductions/bin/MadGraph5_aMCatNLO/dy_5f_LO_MLM_slc7_amd64_gcc700
_CMSSW_10_6_19_tarball.tar.xz
%MSG-MG5 number of events requested = 200
%MSG-MG5 random seed used for the run = 234567
%MSG-MG5 thread count requested = 1
%MSG-MG5 residual/optional arguments =
%MSG-MG5 number of events requested = 200
%MSG-MG5 random seed used for the run = 234567
%MSG-MG5 number of cpus = 1
%MSG-MG5 SCRAM_ARCH version = slc7_amd64_gcc700
%MSG-MG5 CMSSW version = CMSSW_10_6_19
Running MG5_aMC for the 1 time
produced_lhe  0 nevt  200 submitting_event  200  remaining_event  200
run.sh 200 2345670
Now generating 200 events with random seed 2345670 and granularity 1
WRITE GRIDCARD /data/pku/home/lum/CMSSW_10_6_19/src/lheevent/process/madevent
No handlers could be found for logger "madevent.cards"
P0_qq_ll
P0_qq_taptam
P1_qq_ll
P1_qq_taptam
DONE
write ./events.lhe.gz
```

Gridpack to sample

```
Overall cross-section summary
----------------------------------------------------------------------------------------------------
-------------------------------------------------------------
Process         xsec_before [pb]                passed  nposw   nnegw   tried   nposw   nnegw   xsec_match
 [pb]                   accepted [%]     event_eff [%]
0               5.457e+03 +/- 3.226e+01         88      88      0       125     125     0       3.842e+03
+/- 2.240e+02           70.4 +/- 4.1    70.4 +/- 4.1
1               3.328e+03 +/- 6.022e+01         28      28      0       75      75      0       1.242e+03
+/- 1.872e+02           37.3 +/- 5.6    37.3 +/- 5.6
----------------------------------------------------------------------------------------------------
-------------------------------------------------------------
Total           8.785e+03 +/- 6.832e+01         116     116     0       200     200     0       5.095e+03
+/- 3.092e+02           58.0 +/- 3.5    58.0 +/- 3.5
----------------------------------------------------------------------------------------------------
-------------------------------------------------------------
Before matching: total cross section = 8.785e+03 +- 6.832e+01 pb
After matching: total cross section = 5.095e+03 +- 3.092e+02 pb
Matching efficiency = 0.6 +/- 0.0   [TO BE USED IN MCM]
Filter efficiency (taking into account weights)= (116) / (116) = 1.000e+00 +- 0.000e+00
Filter efficiency (event-level)= (116) / (116) = 1.000e+00 +- 0.000e+00    [TO BE USED IN MCM]

After filter: final cross section = 5.095e+03 +- 3.092e+02 pb
After filter: final fraction of events with negative weights = 0.000e+00 +- 0.000e+00
After filter: final equivalent lumi for 1M events (1/fb) = 1.963e-01 +- 1.191e-02
```

The final part of the output.
The cross section is reported as "5095 +- 309.2 pb".

The cross section obtained while produce gridpack is "8758 +- 68.32 pb" in the dy_5f_LO_MLM.log

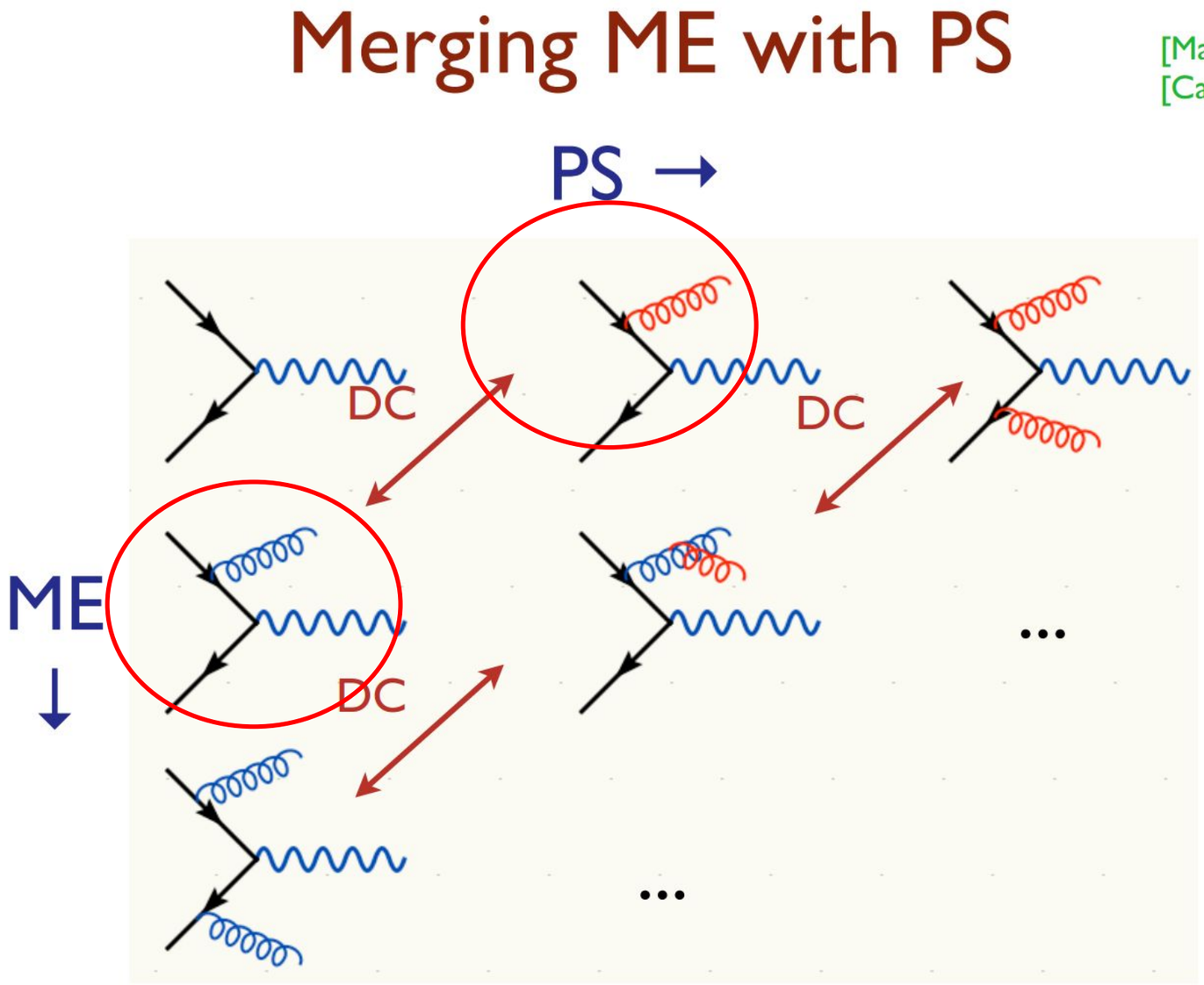
```
[lum@atlas src]$ root -l dy.root
*** DISPLAY not set, setting it to localhost:0.0
root [0]
Attaching file dy.root as _file0...
^[[A(TFile *) 0x19185d0
root [1] Events->GetEntries()
(long long) 116
```

We required 200 events, but here we just get 116 events.

What happened?

Matrix element + Parton shower

The LHE level event, i.e., quark and gluons, are not physical since they are not color-singlet. In order to get physical events, we must perform parton shower and hadronization to have hadrons in the events.



E.g., for process with one additional parton, the parton could be from either Parton shower or the matrix element.

Are there overlap between these two contributions?
Yes! So we need to do matching/merging between PS and ME.

MLM for LO process

For example, we are interested in the following process:
generate p p > e+ e- @0
add process p p > e+ e- j @1
add process p p > e+ e- j j @2
add process p p > e+ e- j j j @3

$\Delta R = \mathrm{sqrt}\{(\eta 1-\eta 2)\text{^}2 + (\phi 1-\phi 2)\text{^}2\}$

MLM Algorithm

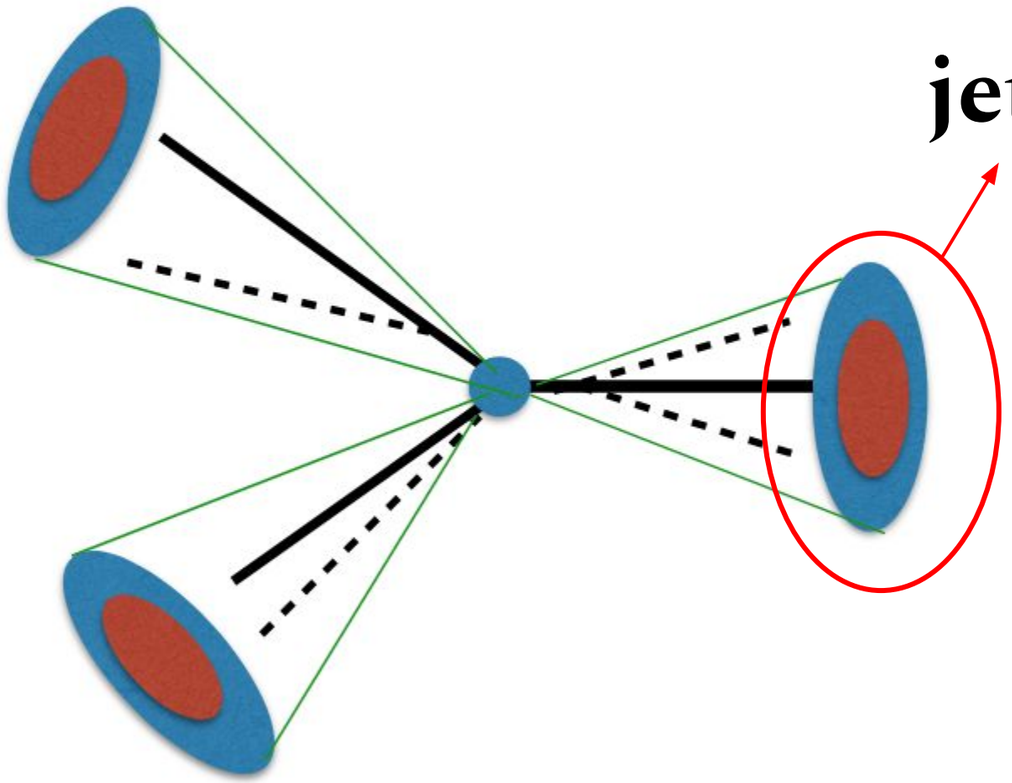
- run MadGraph to get the parton level (LHE level) events
- run other generators (e.g., pythia) for parton shower and hadronization
- the particles after parton shower will be clustered as jets (passing some minimum energy requirement)
- Set a discriminator R, perform a match between the jets list (parton shower) and the partons (LHE level quark) w.r.t their ΔR, if ΔR<R, the jet is matched to the parton. And then remove the matched jet from the jet list, and continue this step
- If there are partons that have not been matched to any jet, then this event is vetoed
- If all partons have matched to jets, but there are still extra jets, then veto this event; but keep this event if this happen to the highest jet multiplicity event. (i.e., for event @0, @1, @2, the partons and jets are one-to-one matched, no extra parton or jet is allowed; but for @3, there could be extra jets that are not matched to partons)

MLM for LO process

These instructive plots are copied from here, made by Andreas Papaefstathiou

——— = hard partons

- - - - - - - - - - - = **PS** partons



Example 1

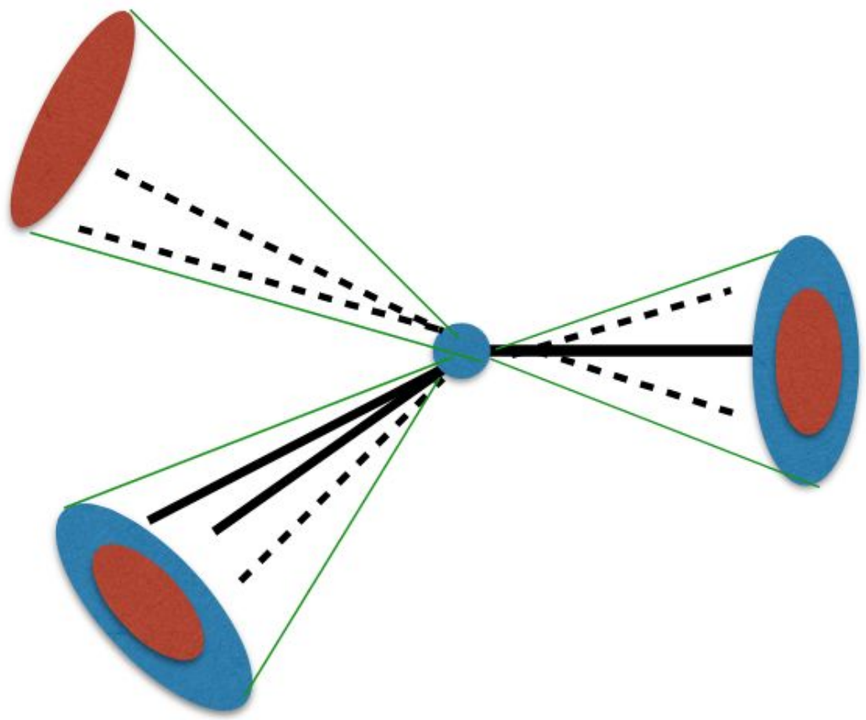
all partons matched: **keep** event.

MLM for LO process

These instructive plots are copied from here, made by Andreas Papaefstathiou

——— = hard partons

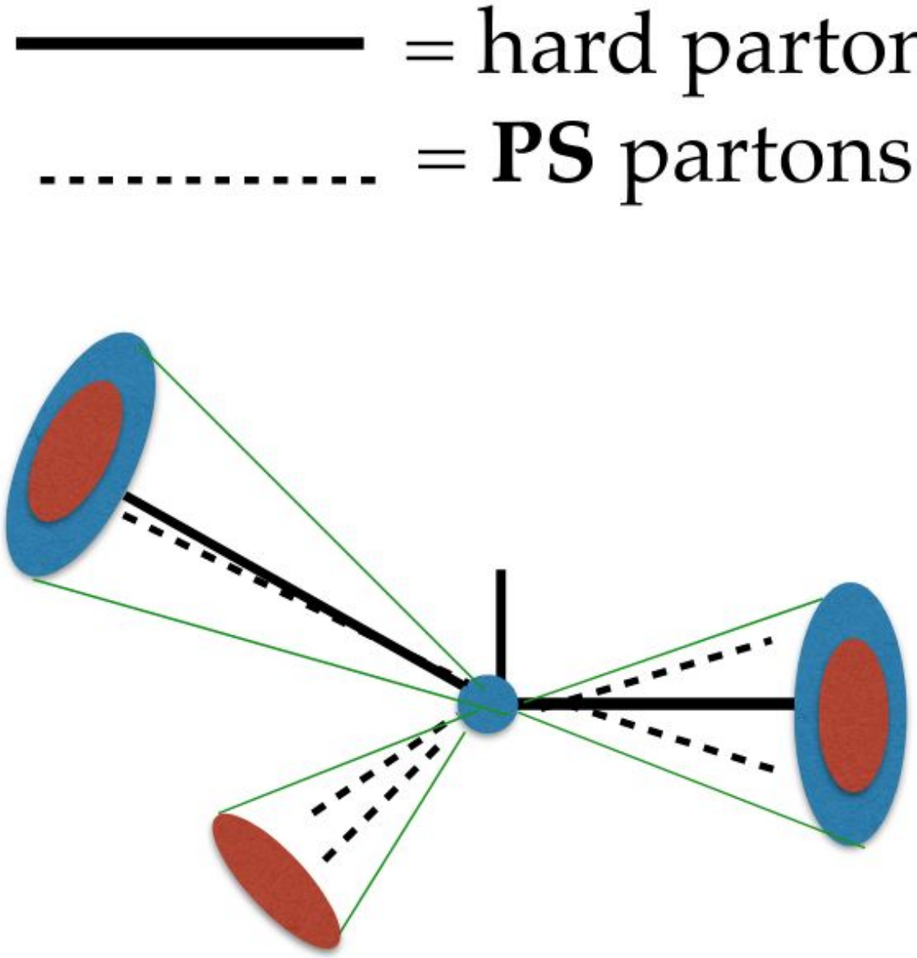
..... = **PS** partons



not all partons match: **veto** event.
(collinear double-log double-counting)

Example 2

MLM for LO process

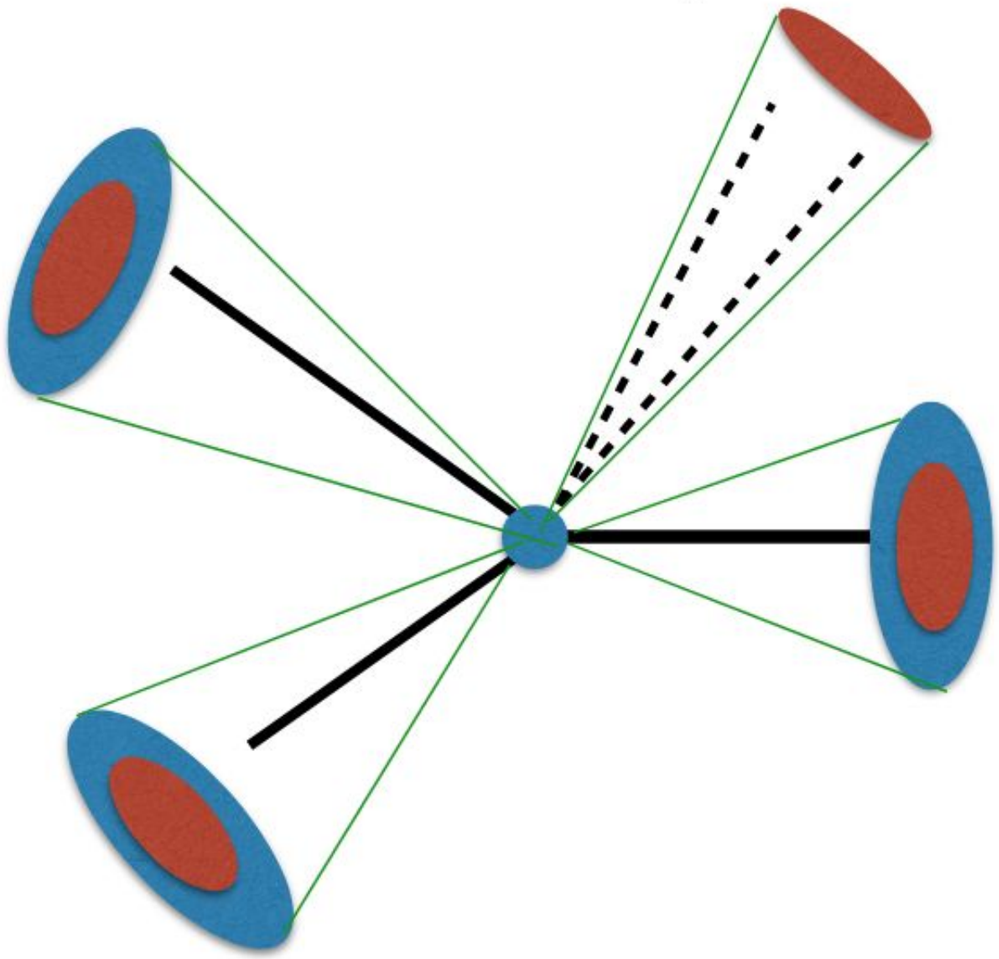


These instructive plots are copied from here, made by Andreas Papaefstathiou

Example 3

not all partons match: **veto** event.
(soft single-log double-counting)

MLM for LO process



all partons match:

These instructive plots are copied from here, made by Andreas Papaefstathiou

Example 4

generate p p > e+ e- @0
add process p p > e+ e- j @1
add process p p > e+ e- j j @2
add process p p > e+ e- j j j @3

Keep this event if the process is as above, the extra jet comes in the highest jet multiplicity event.

MLM for LO process

So we are able to answer the question below:

- The cross section is reported as “5.095e+03 +- 3.092e+02 pb”.
  The cross section obtained while produce gridpack is “8769 +- 20.12 pb” in the dy_5f_LO_MLM.log

- We required 200 events, but here we just get 116 events.

As we apply the MLM matching on our sample, some events are removed to avoid double counting between the Matrix Element and the Parton shower.

The setup is implemented in “/data/pubfs/pku_visitor/public_write/generator_resource/dy_fragment.py”

Sherpa

Navigation

About
Downloads
Sherpa Team
Publications
Theses

Manual
Issue Tracker
Git Repo
Mailing List

Quick search

Sherpa Homepage

Link

Sherpa is a Monte Carlo event generator for the **S**imulation of **H**igh-Energy **R**eactions of **PA**rticles in lepton-lepton, lepton-photon, photon-photon, lepton-hadron and hadron-hadron collisions. Simulation programs - also dubbed event generators - like Sherpa are indispensable work horses for current particle physics phenomenology and are (at) the interface between theory and experiment.

- For a brief summary on the necessity and construction principles of event generators, see About
- To download Sherpa, see Downloads
- To browse the Sherpa manual online, see the manual
- To find out more about the physics in Sherpa, see Publications and Theses.
- To get information about or contact the authors of Sherpa, see Sherpa Team
- To ask questions and browse answers about Sherpa, see the Sherpa Issue Tracker
- To be informed about patches and newer releases, subscribe to our announcement mailing list

Sherpa generator

we need sherpack (just like gridpack) if we want to generate events with CMSSW. All the following is here (require CMS account). It's possible to use standalone sherpa, it will not be introduced here.

2 is the order of EW

maximal number of extra jets

93: quark
90: lepton

```
(processes){
  Process 93 93 -> 90 90 93{NJET};
  Order (*,2); CKKW sqr(QCUT/E_CMS);
  NLO_QCD_Mode MC@NLO {LJET};
  ME_Generator Amegic {LJET};
  RS_ME_Generator Comix {LJET};
  Loop_Generator LOOPGEN {LJET};
  Integration_Error 0.02 {4};
  Integration_Error 0.02 {5};
  Integration_Error 0.05 {6};
  Integration_Error 0.08 {7};
  Integration_Error 0.10 {8};
  Scales LOOSE_METS{FSF*MU_F2}{RSF*MU_R2}{QSF*MU_Q2} {7,8};
  End process;
}(processes)

(selector){
  Mass 11 -11 66 E_CMS
  Mass 13 -13 66 E_CMS
}(selector)
```

ALL these information are in a single input card.

Example link

```
(run){
  % general setting
  EVENTS 1M; ERROR 0.99;

  % scales, tags for scale variations
  FSF:=1.; RSF:=1.; QSF:=1.;
  SCALES METS{FSF*MU_F2}{RSF*MU_R2}{QSF*MU_Q2};

  % tags for process setup
  NJET:=4; LJET:=2,3,4; QCUT:=20.;

  % me generator settings
  ME_SIGNAL_GENERATOR Comix Amegic LOOPGEN;
  EVENT_GENERATION_MODE Weighted;
  LOOPGEN:=BlackHat;

  % exclude tau from lepton container
  MASSIVE[15] 1;

  % collider setup
  BEAM_1 2212; BEAM_ENERGY_1 = 4000.;
  BEAM_2 2212; BEAM_ENERGY_2 = 4000.;
}(run)
```

Sherpa generator

- Tags (`LJET`, `NJET` and `QCUT`) have been introduced to be used in the process setup, defining the multiplicity of the MC@NLO subprocesses, the maximal number of extra jets, and the merging cut.
- The `LOOPGEN` tag is used to name the provider of the one-loop matrix elements for the respective multiplicities. For complicated processes this needs external one-loop programs like BlackHat, GoSam or OpenLoops.
- tau leptons are set massive in order to exclude them from the massless lepton container (`90`).
- As both Comix and Amegic are specified as matrix element generators to be used, Amegic has to be specified to be used for all MC@NLO multiplicities using `ME_Generator Amegic {LJET}`. Additionally, we specify `RS_ME_Generator Comix {LJET}` such that the subtracted real-emission bit of the NLO matrix elements is calculated more efficiently with Comix instead of Amegic.
- The `LOOSE_METS` scale setter, a simplified version of the `METS` scale setter, is used for the highest multiplicities (if `NJET` is set to `5` or `6`) to speed up the calculation.

```
NJET:=4; LJET:=2,3,4;
...
Process 93 93 -> 90 90 93{NJET};
Order (*,2); CKKW sqr(QCUT/E_CMS);
NLO_QCD_Mode MC@NLO {LJET};
ME_Generator Amegic {LJET};
RS_ME_Generator Comix {LJET};
Loop_Generator LOOPGEN {LJET};
```

```
93 93 -> 90 90 : order QCD NLO
93 93 -> 90 90 93: order QCD NLO
93 93 -> 90 90 93 93: order QCD NLO
93 93 -> 90 90 93 93 93: order QCD LO
93 93 -> 90 90 93 93 93 93: order QCD LO
```

Sherpack generation

No hands-on, some steps require good connection

- cmsrel CMSSW_10_6_21
- cd CMSSW_10_6_21/src
- cmsenv
- export TOPDIR=$PWD
- git cms-addpkg GeneratorInterface/SherpaInterface
- mkdir -p MY/PROJECT/test
- mkdir -p MY/PROJECT/python
- cd MY/PROJECT/test/
- cp $TOPDIR/GeneratorInterface/SherpaInterface/data/*SherpaLibs.sh .
- cp [PATH_TO_YOUR_RUNCARD]/Run.dat_[XYZ] . (e.g., the card is Run.dat_dy)
- sh MakeSherpaLibs.sh -p dy -o LBCR -v -m mpirun -M '-n 4'

```
In Event_Handler::Finish : Summarizing the run may take some time.
+---------------------------------------------+
|                                             |
|  Total XS is 6567.33 pb +- ( 643.92 pb = 9.8 % )  |
|                                             |
+---------------------------------------------+
```

```
[melu@lxplus770 test]$ ls
MakeSherpaLibs.sh     Run.dat_dy          sherpa_dy_crss.tgz  sherpa_dy_logL.tgz
PrepareSherpaLibs.sh  sherpa_dy_crdE.tgz  sherpa_dy_libs.tgz  sherpa_dy_migr.tgz
```

- ./PrepareSherpaLibs.sh -p dy

Sherpack generation: Run.dat_dy

```
(processes){
 Process 93 93 -> 90 90 93{NJET};
 Order (*,2); CKKW sqr(QCUT/E_CMS);
 NLO_QCD_Mode MC@NLO {LJET};
 ME_Generator Amegic {LJET};
 RS_ME_Generator Comix {LJET};
 Loop_Generator LOOPGEN {LJET};
 Integration_Error 0.02 {2};
 End process;
}(processes)

(selector){
 Mass 11 -11 50 E_CMS
 Mass 13 -13 50 E_CMS
 Mass 15 -15 50 E_CMS
}(selector)
```

[93 93 -> 90 90 : order QCD NLO]
[93 93 -> 90 90 93: order QCD LO]

```
(run){
 EVENTS 100K; ERROR 0.99;
 HEPMC_USE_NAMED_WEIGHTS=1
 FSF:=1.; RSF:=1.; QSF:=1.;
 SCALES
METS{FSF*MU_F2}{RSF*MU_R2}{QSF*MU_Q2};
 NJET:=1; LJET:=2; QCUT:=20.;
 ME_SIGNAL_GENERATOR Comix Amegic
LOOPGEN;
 EVENT_GENERATION_MODE P;
 LOOPGEN:=OpenLoops;
 BEAM_1 2212; BEAM_ENERGY_1 = 6500.;
 BEAM_2 2212; BEAM_ENERGY_2 = 6500.;
 PDF_LIBRARY LHAPDFSherpa;
 PDF_SET NNPDF30_nlo_nf_5_pdfas;
 PDF_VARIATIONS NNPDF30_nlo_nf_5_pdfas[all];
}(run)
```

Sherpack generation

```
<I>  for a production with CRAB add the sherpack to the list of additional files
<I>    additional_input_files = [name of the ..._MASTER.tgz sherpack]
<I>  make sure that the sherpack location is
<I>    SherpackLocation = cms.string('./')
<I>  and make sure that the SherpaInterface does not try to fetch the sherpack
<I>    FetchSherpack = cms.bool(False)
<I>
<I>
<I>
<I>  a good way to test the generated python file is to cross-check it with cmsDriver.py:
      cmsDriver.py A/B/python/sherpa_dy_MASTER_cff.py \
       -s GEN -n 100 --no_exec --conditions auto:mc --eventcontent RAWSIM
<I>
 <I>
 <I>
 <I>
[melu@lxplus770 test]$ ls
MakeSherpaLibs.sh     sherpa_dy_MASTER.md5     sherpa_dy_crdE.tgz  sherpa_dy_logL.tgz
PrepareSherpaLibs.sh  sherpa_dy_MASTER.tgz     sherpa_dy_crss.tgz  sherpa_dy_migr.tgz
Run.dat_dy            sherpa_dy_MASTER_cff.py  sherpa_dy_libs.tgz
```

The “sherpa_dy_MASTER.tgz” is the sherpack, and “sherpa_dy_MASTER_cff.py” is the fragment. The output also tell us how to construct the file we needed for cmssw event generation.

Sherpack to events

- ⏭ cd /YOURPATH/CMSSW_10_6_19/src (your previous CMSSW used for MG gridpack could be used here)
- ⏭ cp /data/pubfs/pku_visitor/public_write/generator_resource/sherpa_dy_MASTER_cff.py Configuration/GenProduction/python/
- ⏭ cmsDriver.py Configuration/GenProduction/python/sherpa_dy_MASTER_cff.py -s GEN -n 1000 --no_exec --conditions auto:mc --eventcontent RAWSIM

Then you will have sherpa_dy_MASTER_cff_py_GEN.py **Hands-on**

Change two lines in sherpa_dy_MASTER_cff_py_GEN.py:

1. FetchSherpack = cms.bool(False) -> FetchSherpack = cms.bool(True)
2. SherpackLocation = cms.string('./') -> SherpackLocation = cms.string('/data/pubfs/pku_visitor/public_write/generator_resource')

⏭ cmsRun sherpa_dy_MASTER_cff_py_GEN.py

```
GenXsecAnalyzer:
------------------------------------
Before Filter: total cross section = 5.656e+03 +- 1.854e+02 pb
Filter efficiency (taking into account weights)= (1.58741e+08) / (1.58741e+08) = 1.000e+00 +- 0.000e+00
Filter efficiency (event-level)= (1000) / (1000) = 1.000e+00 +- 0.000e+00    [TO BE USED IN MCM]

After filter: final cross section = 5.656e+03 +- 1.854e+02 pb
After filter: final fraction of events with negative weights = 2.100e-02 +- 9.522e-05
After filter: final equivalent lumi for 1M events (1/fb) = 1.623e-01 +- 5.098e-03
```

Sherpack to samples

- cd /YOURPATH/CMSSW_10_6_19/src (your previous CMSSW used for MG gridpack could be used here)
- cp /data/pubfs/pku_visitor/public_write/generator_resource/sherpa_dy_MASTER_cff.py Configuration/GenProduction/python/
- cmsDriver.py Configuration/GenProduction/python/sherpa_dy_MASTER_cff.py -s GEN -n 1000 --no_exec --conditions auto:mc --eventcontent RAWSIM

Then you will have sherpa_dy_MASTER_cff_py_GEN.py

Change two lines in sherpa_dy_MASTER_cff_py_GEN.py:

1. FetchSherpack = cms.bool(False) -> FetchSherpack = cms.bool(True)
2. SherpackLocation = cms.string('./') -> SherpackLocation = cms.string('/data/pubfs/pku_visitor/public_write/generator_resource')

- cmsRun sherpa_dy_MASTER_cff_py_GEN.py

```
[lum@atlas src]$ ls
Configuration                                  Result.db              dy_inLHE.root
MIG_P+P+_13000_LHA[NNPDF30_nlo_nf_5_pdfas]_1.db  Run.dat              sherpa_dy_MASTER.tar
MPI_Cross_Sections.dat                         Sherpa_References.tex  sherpa_dy_MASTER.tgz
Process                                        dy.root                sherpa_dy_MASTER_cff_py_GEN.py
Result                                         dy_cfg.py              sherpa_dy_MASTER_cff_py_GEN.root
```

```
[lum@atlas src]$ root -l sherpa_dy_MASTER_cff_py_GEN.root
root [0]
Attaching file sherpa_dy_MASTER_cff_py_GEN.root as _file0...
(TFile *) 0x309d940
root [1] Events->GetEntries()
(long long) 1000
```

The match is handled by sherpa itself, required number of events are generated.

Summary

There are many generators used in CMS, we have introduced several of them, i.e., MadGraph, pythia, sherpa, including how to use them to generate events.

Other generators not covered are also very interesting, e.g., powheg (Positive Weight Hardest Emission Generator,we may need login to lxplus, twiki), MCFM (Monte Carlo for FeMtobarn processes), Herwig (multi-purpose particle physics event generator, could be used for hard scattering events and also for parton shower)...

Hope you have enjoying the generator studies in the past days!

Good luck with your GENERATOR STUDY!

Additional slides

Standalone Pythia

Pythia could be used for parton shower and also generate hard scattering event.
Standalone Pythia Installation: we use HepMC for the pythia output (we don't install HepMC here, use the lib /home/pku/lum/software/HepMC)

Default env (no cmsenv)

- cp /home/pku/lum/pythia8306.tgz /YOUR/PATH
- tar xf pythia8306.tgz
- cd pythia8306
- ./configure --with-hepmc2=/home/pku/lum/software/HepMC --prefix=.
- make (~ 10mins)

```
[lum@atlas pythia8306]$ ls
AUTHORS  CODINGSTYLE  COPYING   GUIDELINES  lib      Makefile.inc  pythia8-config.inc  share  tmp
bin      configure    examples  include     Makefile  plugins      README              src
```

- cd examples
- cp /home/pku/lum/pythia8306/examples/main90.cc .
- g++ main90.cc -o main90 -I../include -O2 -std=c++11 -pedantic -W -Wall -Wshadow -fPIC -L../lib -Wl,-rpath,../lib -lpythia8 -ldl -I/data/pku/home/lum/software/HepMC/include -L/data/pku/home/lum/software/HepMC/lib -Wl,-rpath,/data/pku/home/lum/software/HepMC/lib -lHepMC -DHEPMC2
- ./main90