POWHEG Tutorial @

CMS Topical Workshop on Off-shell Higgs Boson Production at the LPC

Meng Lu (Northeastern University)

3/25/2024

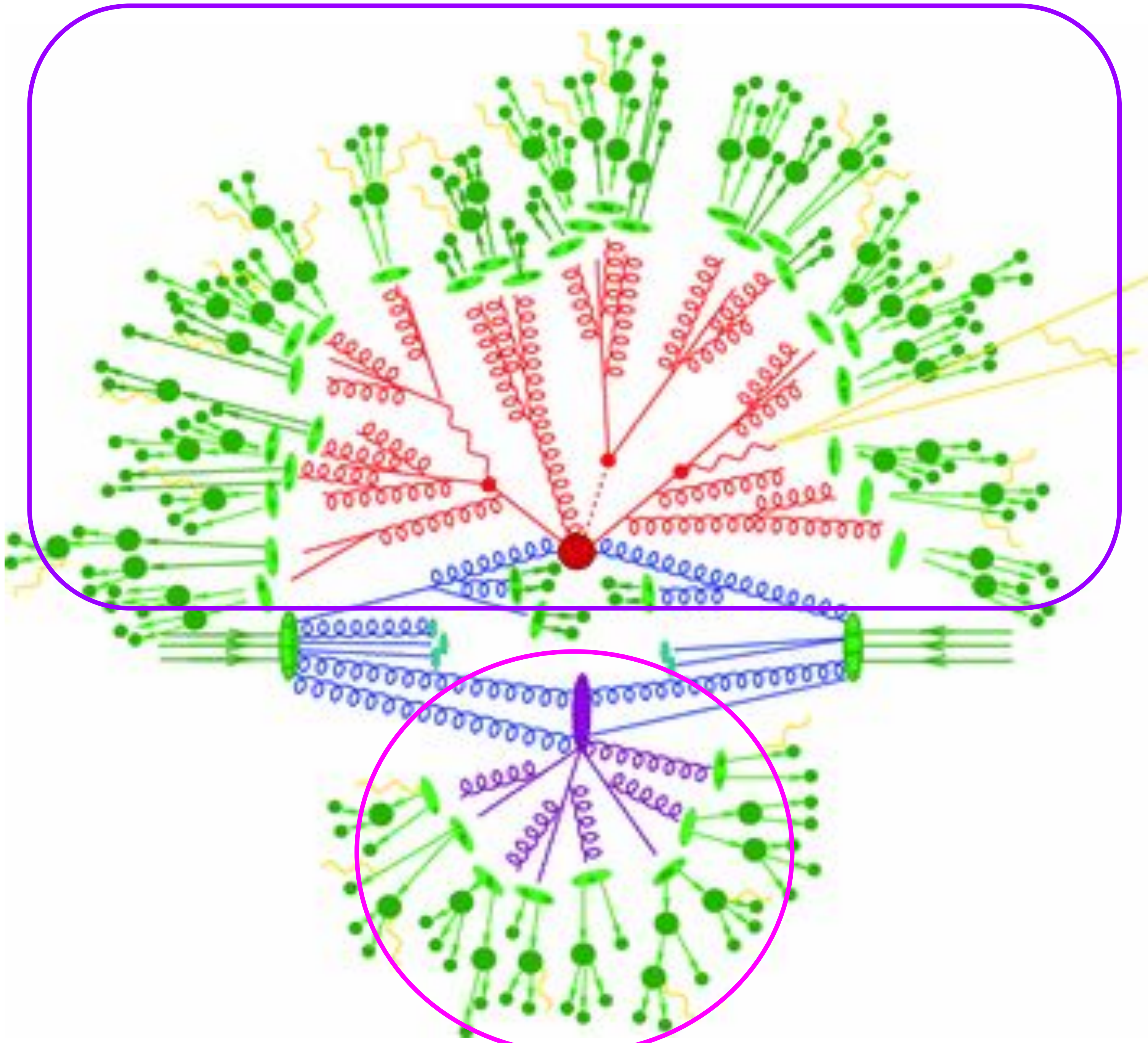
Disclaimer

- I'm NOT a theory expert, NO detail on the calculation, on the code implementation ...
- Some personal understanding (mostly conceptional) on POWHEG will be shared, maybe useful from a non-expert point of view
- Some experience on how to use POWHEG within CMS, i.e., gridpack
- More detail on POWHEG main page, useful talks can be found in recent GEN tutorial
- CREDITS: Thanks to Paolo Nason and Carlo Oleari for a lot of materials appearing in this talk

Powheg Tutorial

- Recap of MC events
- Brief introduction to Powheg
  - Shower MC
  - Sudakov form factor
  - NLOSMC and Powheg implementation
- How to run powheg within CMS environment
  - The CMS interface for POWHEG gridpack
  - Input card
  - Hands-on to a powheg gridpack
  - Check validity of gridpack
  - Other features

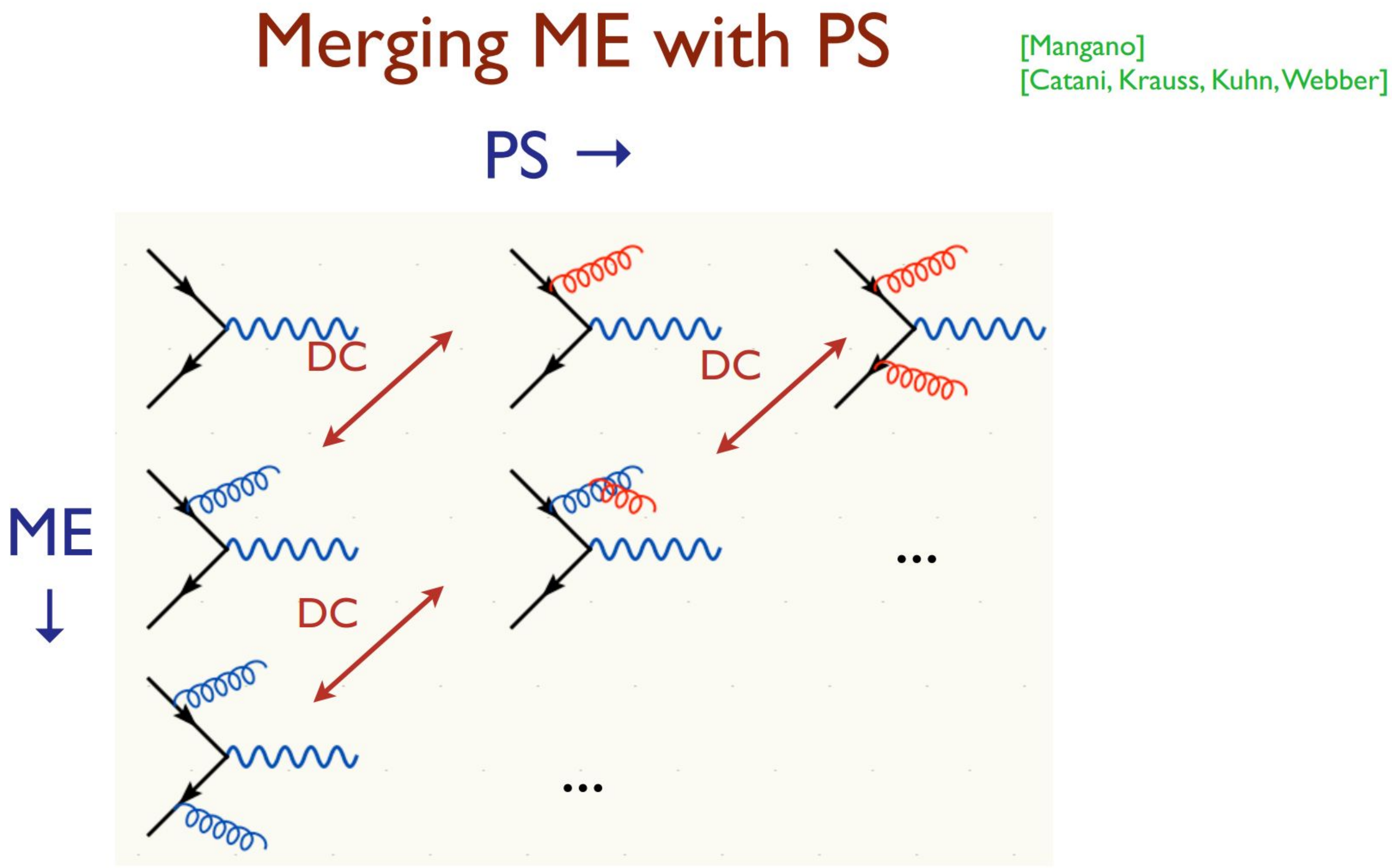
Recap of MC events in LHC



- Hard process: interaction between partons from incoming protons, the Matrix-element can be provided by: MG (HELAS, OneLoop ...), Sherpa (Amegic, Comix) ...
- Soft process: the evolution of the hard event, QCD/QED radiation (color labels are assigned to partons during parton shower), typical generators are Pythia and Herwig
- Hadronization: color singlet structures are formed out of color connected partons, i.e., hadrons, and then undergo the decays

Underlying events: Beam remnants, multiple-parton interactions (e.g., DPS)

Matrix Element + Parton Shower



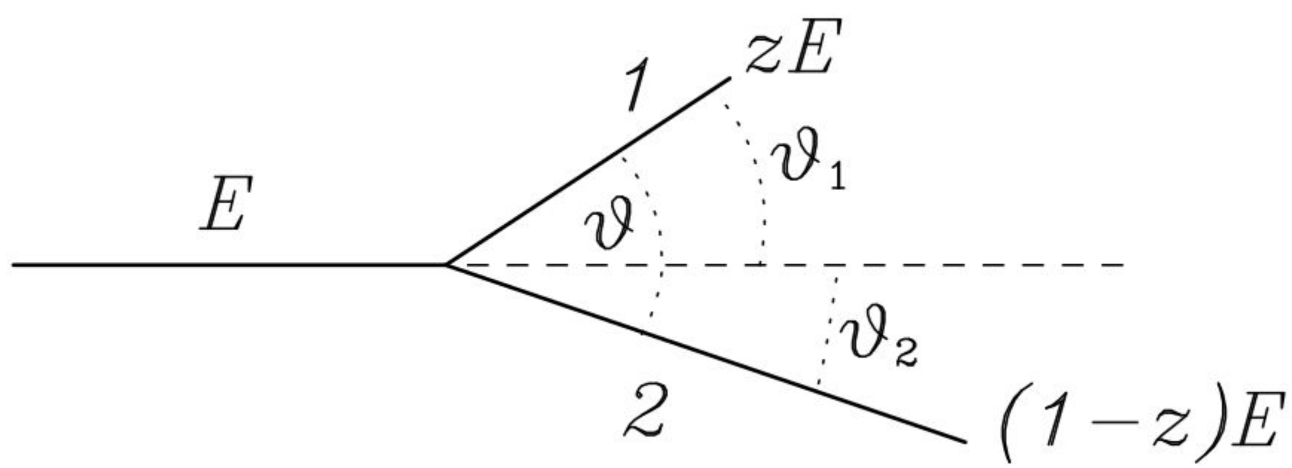
QCD emission can happen in both ME and PS, how to avoid double counting (e.g., LO, MLM).

- Generate parton level event, with requirement on minimum parton energy as well as minimum separation between partons
- Then events are showered, use jet algorithm to get jets
- Starting from the hardest parton, select the closest jet, if their separation is smaller than some pre-defined $\mathbf{R}_{\min}$, the parton is matched to the jet.
- Accept the event if all partons have its corresponding jets. Otherwise veto this event.

Shower MC

Given a partonic event, let's say only one quark or gluon with energy E for simplicity, the probability of emission in the interval dt:

$$dP_{\text{emis}}(t+dt,t) = \frac{dt}{t}\frac{\alpha_s(t)}{2\pi}\int dz\, P_{i,jk}(z)$$



Where

- Z is energy fraction
- There are many choice of t which corresponds to different shower algorithm, e.g. $t=E^2\theta^2$
- $P_{i,jk}$ is the splitting function

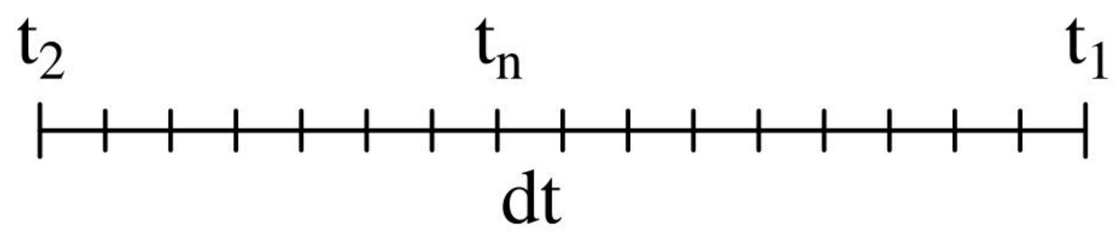
So the probability of no emission in the interval dt:

$$dP_{\text{no emis}}(t+dt,t) = 1 - dP_{\text{emis}}(t+dt,t) = 1 - \frac{dt}{t}\frac{\alpha_s(t)}{2\pi}\int dz\, P_{i,jk}(z)$$

One thing keep in mind is that such "no emission" probability contains all the virtual correction.

Shower MC: Sudakov form factor

Divide a finite interval [t2, t1] in N small intervals dt = (t1 - t2)/N



The probability of not emitting radiation between the two ordering scales t1 and t2 is given by the product

$$
\begin{aligned}
P_{\text{no emis}}(t_1, t_2) &= \lim_{N \to \infty} \prod_{n=1}^{N} \left[1 - \frac{dt}{t_n} \frac{\alpha_s(t_n)}{2\pi} \int dz\, P_{i,jk}(z) \right] \\
&= \exp\left\{ - \int_{t_2}^{t_1} \frac{dt}{t} \frac{\alpha_s(t)}{2\pi} \int dz\, P_{i,jk}(z) \right\} \\
&\equiv \Delta(t_1, t_2)
\end{aligned}
$$

The weight Δ(t1, t2) is called Sudakov form factor. It resums all the dominant virtual corrections to the tree graph

Shower MC: XS for the first emission

The cross section for first emission in Shower MC can be defined (n-body at LO):

$$\langle O \rangle = \int d\mathbf{\Phi}_n B(\mathbf{\Phi}_n) \left\{ O(\mathbf{\Phi}_n) \Delta_{t_0} + \int_{t_0} \frac{dt}{t} dz\, d\varphi\, O(\mathbf{\Phi}_n, \Phi_r) \Delta_t \frac{\alpha_s}{2\pi} P(z) \right\}$$

N-body phase space | Born-level structure | Observable | Sudakov form factor | Splitting function

- The first item in the square bracket is the no-emission probability, t0 stands for the lower cutoff
- The second item in the square bracket is the probability of the first emission, carrying energy *zE* and at separation *t*.

The expansion of above formula (basically expand Δ(t)) at order αs gives the NLO shower MC

$$\langle O \rangle = \int d\mathbf{\Phi}_n B(\mathbf{\Phi}_n) \left\{ O(\mathbf{\Phi}_n) + \int_{t_0} \frac{dt}{t} dz\, d\varphi \left[O(\mathbf{\Phi}_n, \Phi_r) - O(\mathbf{\Phi}_n) \right] \frac{\alpha_s}{2\pi} P(z) \right\}$$

NLO + Parton shower

“An MC@NLO is affected by double counting if its prediction for any observable, at the first order beyond the Born approximation in the expansion in the coupling constant, is not equal to the NLO prediction.” in PAPER

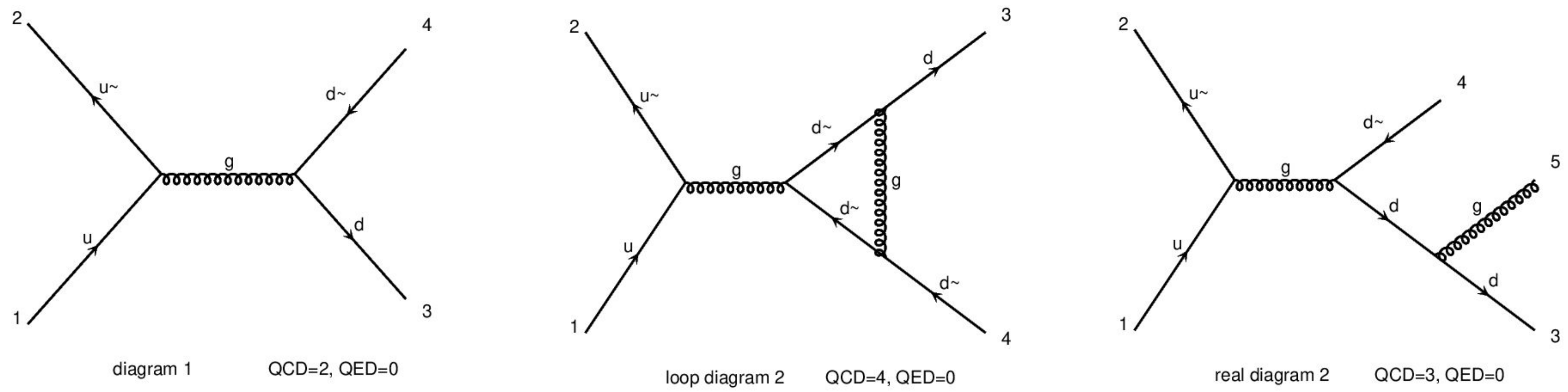
➡ combine a NLO with Shower MC need to overcome the double counting between them, do a “matching” between Shower MC and fixed order NLO calculation.

$$\langle O \rangle = \int d\mathbf{\Phi}_n \, B(\mathbf{\Phi}_n) \left\{ O(\mathbf{\Phi}_n) + \int_{t_0} \frac{dt}{t} \, dz \, d\varphi \left[O(\mathbf{\Phi}_n, \Phi_r) - O(\mathbf{\Phi}_n) \right] \frac{\alpha_s}{2\pi} \, P(z) \right\}$$

Do some tricks on the formula to make it match the NLO calculation. Two well-established and widely used methods currently:

- POWHEG (POSITIVE-WEIGHT HARDEST EMISSION GENERATOR)
- MC@NLO (Equivalent as Powheg, but may have large number of negatively weighted events for some processes)

Fixed order NLO



$$\langle O \rangle = \int O\, d\sigma = \int d\boldsymbol{\Phi}_n\, O(\boldsymbol{\Phi}_n)\, [B(\boldsymbol{\Phi}_n) + V_b(\boldsymbol{\Phi}_n)] + \int d\boldsymbol{\Phi}_n\, d\Phi_r\, O(\boldsymbol{\Phi}_n, \Phi_r)\, R(\boldsymbol{\Phi}_n, \Phi_r)$$

The virtual and real-radiation integrals are separate divergent, but their sum is finite.
Use counter term to re-organize the integrals

$$\begin{aligned}\langle O \rangle &= \int d\boldsymbol{\Phi}_n\, O(\boldsymbol{\Phi}_n) \left[B(\boldsymbol{\Phi}_n) + \underbrace{V_b(\boldsymbol{\Phi}_n) + \int d\Phi_r\, C(\boldsymbol{\Phi}_n, \Phi_r)}_{\text{Finite}}\right] \\ &+ \int d\boldsymbol{\Phi}_n\, d\Phi_r \underbrace{\left[O(\boldsymbol{\Phi}_n, \Phi_r)\, R(\boldsymbol{\Phi}_n, \Phi_r) - O(\boldsymbol{\Phi}_n)\, C(\boldsymbol{\Phi}_n, \Phi_r)\right]}_{\text{Finite}}\end{aligned}$$

$$\langle O \rangle = \int d\boldsymbol{\Phi}_n\, O(\boldsymbol{\Phi}_n)\, [B(\boldsymbol{\Phi}_n) + V(\boldsymbol{\Phi}_n)] + \int d\boldsymbol{\Phi}_n\, d\Phi_r\, [O(\boldsymbol{\Phi}_n, \Phi_r)\, R(\boldsymbol{\Phi}_n, \Phi_r) - O(\boldsymbol{\Phi}_n)\, C(\boldsymbol{\Phi}_n, \Phi_r)]$$

More on fixed order NLO

$$\begin{aligned}\langle O\rangle &= \int d\mathbf{\Phi}_n\, O(\mathbf{\Phi}_n)\,[B(\mathbf{\Phi}_n)+V(\mathbf{\Phi}_n)] \\ &+ \int d\mathbf{\Phi}_n\, d\Phi_r\,[O(\mathbf{\Phi}_n,\Phi_r)\,R(\mathbf{\Phi}_n,\Phi_r) - O(\mathbf{\Phi}_n)\,C(\mathbf{\Phi}_n,\Phi_r)] \\ &= \int d\mathbf{\Phi}_n\, O(\mathbf{\Phi}_n)\left\{B(\mathbf{\Phi}_n)+V(\mathbf{\Phi}_n)+\int d\Phi_r\,[R(\mathbf{\Phi}_n,\Phi_r)-C(\mathbf{\Phi}_n,\Phi_r)]\right\} \\ &+ \int d\mathbf{\Phi}_n\, d\Phi_r\, R(\mathbf{\Phi}_n,\Phi_r)\,[O(\mathbf{\Phi}_n,\Phi_r)-O(\mathbf{\Phi}_n)]\end{aligned}$$

Define

$$\overline{B}(\mathbf{\Phi}_n) = B(\mathbf{\Phi}_n)+V(\mathbf{\Phi}_n)+\int d\Phi_r\,[R(\mathbf{\Phi}_n,\Phi_r)-C(\mathbf{\Phi}_n,\Phi_r)]$$

$$\langle O\rangle = \int d\mathbf{\Phi}_n\, O(\mathbf{\Phi}_n)\,\overline{B}(\mathbf{\Phi}_n) + \int d\mathbf{\Phi}_n\, d\Phi_r\, R(\mathbf{\Phi}_n,\Phi_r)\,[O(\mathbf{\Phi}_n,\Phi_r)-O(\mathbf{\Phi}_n)]$$

In $\mathrm{NLO}_{\mathrm{SMC}}$, it was

$$\langle O\rangle = \int d\mathbf{\Phi}_n\, O(\mathbf{\Phi}_n)\,B(\mathbf{\Phi}_n) + \int d\mathbf{\Phi}_n\, d\Phi_r\, B(\mathbf{\Phi}_n)\,\frac{\alpha_s}{2\pi}\,P(z)\,\frac{1}{t}\,[O(\mathbf{\Phi}_n,\Phi_r)-O(\mathbf{\Phi}_n)]$$

Modify the corresponding terms in $\mathrm{NLO}_{\mathrm{SMC}}$ to fit the fixed order NLO calculation

POWHEG implementation

$$d\sigma_{\text{POWHEG}} = \overline{B}(\mathbf{\Phi}_n)\, d\mathbf{\Phi}_n \left\{ \Delta(\mathbf{\Phi}_n, p_T^{min}) + \frac{R(\mathbf{\Phi}_n, \Phi_r)}{B(\mathbf{\Phi}_n)} \Delta(\mathbf{\Phi}_n, p_T)\, d\Phi_r \right\}$$

$$\overline{B}(\mathbf{\Phi}_n) = B(\mathbf{\Phi}_n) + V(\mathbf{\Phi}_n) + \int d\Phi_r \left[R(\mathbf{\Phi}_n, \Phi_r) - C(\mathbf{\Phi}_n, \Phi_r) \right]$$

$$\Delta(\mathbf{\Phi}_n, p_T) = \exp\left[- \int d\Phi'_r \frac{R(\mathbf{\Phi}_n, \Phi'_r)}{B(\mathbf{\Phi}_n)} \theta\left(k_T(\mathbf{\Phi}_n, \Phi'_r) - p_T \right) \right] \quad \text{POWHEG Sudakov}$$

- ➢ generate a random number 0 < r < 1
- ➢ Solve the equation Δ(Φ, pT) = r for pT
- ➢ If pT < pTmin, no radiation is generated
- ➢ Otherwise, the other radiation variables are generated at fixed pT proportionally to R/B
- ➢ Then put the event in the shower MC with requirement: no emission larger than pT

Why POWHEG avoid negative weight event?

→ Following formula is negative must signal the failure of perturbation theory, since the NLO negative terms have overcome the Born term, generally it should be positive

$$\overline{B}(\mathbf{\Phi}_n) = B(\mathbf{\Phi}_n) + V(\mathbf{\Phi}_n) + \int d\Phi_r \left[R(\mathbf{\Phi}_n, \Phi_r) - C(\mathbf{\Phi}_n, \Phi_r) \right]$$

POWHEG within CMS

- As shown earlier, POWHEG needs to modify the sudakov form factor according to the process, specific code is needed for each process, all available processes can be found here.
- Most of the CMS samples are produced using gridpack mode, during which the compilation and integration are performed, which are usually very time consuming. All the detail for POWHEG gridpack production can be found here

The CMS interface for POWHEG gridpack

The codes used for gridpack production is stored in git

Name	Last commit message
..	
Templates	fix to WWJ tarball and lhe file creation
Utilities	updated cmake version - temporary solution
examples	complement decay inforamtion of VBF ZZ
lheMacros	some minor changes suggested by users
patches	Delete ttJ_minnlo_rwt_topmass_compiler.patch
production	Merge pull request #3648 from De-Cristo/bbH_mssm_run3
source	Update powheg scripts to python3
test	Automated addition hvq_el8_amd64_gcc12_CMSSW_13_3_0
check_dag_success.py	Use DAG for powheg parallel (see also run_pwg_condor changes in prev …
dag_wrapper.sh	Automatic OS for condor submission
gridpack_generation.sh	Update gridpack_generation.sh
make_rwl.py	Fixing WWJ Powheg process
run_lhe_condor.sh	fixes for nnlops
run_nnlops.sh	fixes for nnlops
run_pwg_condor.py	Fixing WWJ Powheg process
run_pwg_parallel_condor.py	Automatic OS for condor submission
runcmsgrid_powheg.sh	fix to WWJ tarball and lhe file creation
runcmsgrid_powhegjhugen.sh	Update runcmsgrid_powhegjhugen.sh
slc6wrapper.sh	Powheg EW and SLC6 singularity

The steer script for the gridpack production:

- run_pwg_condor.py
- run_pwg_parallel_condor.py: multicore mode

The CMS interface for POWHEG gridpack

```
parser.add_argument('-p', '--parstage'      , dest="parstage",      default= '0',            help='stage of the production process [0]')
parser.add_argument('-x', '--xgrid'         , dest="xgrid",         default= '1',            help='loop number for the grids production [1]')
parser.add_argument('-f', '--folderName'    , dest="folderName",    default='testProd',      help='local folder and last eos folder name[testProd]')
parser.add_argument('-e', '--eosFolder'     , dest="eosFolder",     default='NONE' ,         help='folder before the last one, on EOS')
parser.add_argument('-j', '--numJobs'       , dest="numJobs",       default= '10',           help='number of jobs to be used for multicore grid step 1,2,3')
parser.add_argument('-t', '--totEvents'     , dest="totEvents",     default= '10000',        help='total number of events to be generated [10000]')
parser.add_argument('-n', '--numEvents'     , dest="numEvents",     default= '2000',         help='number of events for a single job [2000]')
parser.add_argument('-i', '--inputTemplate' , dest="inputTemplate", default= 'powheg.input', help='input cfg file (fixed) [=powheg.input]')
parser.add_argument('-g', '--inputJHUGen' , dest="inputJHUGen", default= '', help='input JHUGen cfg file []')
parser.add_argument('-q', '--doQueue'      , dest="doQueue",      default= 'none',          help='Jobflavour: if none, running interactively [none]')
parser.add_argument('-s', '--rndSeed'       , dest="rndSeed",       default= '42',           help='Starting random number seed [42]')
parser.add_argument('-m', '--prcName'       , dest="prcName",       default= 'DMGG',           help='POWHEG process name [DMGG]')
parser.add_argument('-k', '--keepTop'       , dest="keepTop",       default= '0',           help='Keep the validation top draw plots [0]')
parser.add_argument('-d', '--noPdfCheck'    , dest="noPdfCheck",    default= '0',           help='If 1, deactivate automatic PDF check [0]')
parser.add_argument('--fordag'     , dest="fordag",     default= 0,           help='If 1, deactivate submission, expect condor DAG file to be created [0]')
parser.add_argument('--svn'     , dest="svnRev",     default= 0,           help='SVN revision. If 0, use tarball [0]')
parser.add_argument('--ion'     , dest="ion",     default= '',           help='Ion type. Options: Pb []')
```

run_pwg_condor.py:

Higgs boson production in gluon fusion with quark mass and EW effects, E. Bagnaschi, G. Degrassi, P. Slavich, A. Vicini, arXiv:1111.2854 [paper]

POWHEG-BOX/gg_H_quark-mass-effects

If you plan to generate ggH with the quark mass effect, the parameter after '-m' MUST EXACTLY BE "gg_H_quark-mass-effects"

- -p grid production stage [f] (one go)
- -i intput card name [powheg.input]
- -m process name (process defined in POWHEG: CANNOT be a sub-process, see below)
- -f working folder [my_ggH]
- -q job flavor / batch queue name (run locally if not specified)
- -n the number of events to run

The input card

Powhg is widely used within CMS, basically you should be able to find all the process cards under GEN_Git. Or if you are eager to learn more, you can download all the materials you are interested in. The POWHEG BOX is made available only via SVN, with the command, e.g., in lxplus:

- svn checkout svn://powhegbox.mib.infn.it/trunk/POWHEG-BOX-NoUserProcesses POWHEG-BOX
- cd POWHEG-BOX
- svn checkout svn://powhegbox.mib.infn.it/trunk/POWHEG-BOX/"process-of-interest", e.g., svn checkout svn://powhegbox.mib.infn.it/trunk/POWHEG-BOX/gg_H
- cd gg_H

```
[melu@lxplus805 gg_H]$ ls
Born.f                   LesHouchesreg.f      pwhg_cpHTO_reweight.f
Born_phsp.f              make_btilde.sh       pythia-6.4.23.f
Docs                     Makefile             pythia.f
herwig6520.f             make_sigremnants.sh  real.f
herwig6520.inc           mod                  setup-HERWIG-lhef.f
HERWIG65.INC             nlegborn.h           setup-PYTHIA-lhef.f
herwig.f                 obj                  testrun-lhc
init_couplings.f         PhysPars.h           testrun-tev
init_processes.f         pt2maxreg.f          virtual.f
last_tested_revision     pwhg_analysis.f
```

e.g., inside Docs, you have the fully documentation of this process, as well as the template card

```
[melu@lxplus805 gg_H]$ ls Docs/
JHEP3.cls  manual-BOX-ggH.pdf  manual-BOX-ggH.tex
```

```
[melu@lxplus805 gg_H]$ ls testrun-lhc/
cteq6m  powheg.input
```

The input card: gg_H_quark-mass-effects

Use [gg_H_quark-mass-effects_NNPDF31_13TeV_M125.input]() as the example

```
numevts NEVENTS       ! number of events to be generated
ih1   1               ! hadron 1 (1 for protons, -1 for antiprotons)
ih2   1               ! hadron 2 (1 for protons, -1 for antiprotons)
ebeam1 6500           ! energy of beam 1
ebeam2 6500           ! energy of beam 2

lhans1 325300         ! pdf set for hadron 1 (LHA numbering)
lhans2 325300         ! pdf set for hadron 2 (LHA numbering)


! Parameters to allow or not the use of stored data
use-old-grid   1 ! if 1 use old grid if file pwggrids.dat is present (<> 1 regenerate)
use-old-ubound  1 ! if 1 use norm of upper bounding function stored in pwgubound.dat, if present; <> 1 regenerate
```

Nothing special, collision setup, beam, energy, PDF

In order to re-use the existing grid files, these two settings are required in the powheg input data card

The input card: gg_H_quark-mass-effects

Use gg_H_quark-mass-effects_NNPDF31_13TeV_M125.input as the example

```
ncall1 550000   ! number of calls for initializing the integration grid
itmx1  7   ! number of iterations for initializing the integration grid
ncall2 75000 ! number of calls for computing the integral and finding upper bound
itmx2  5  ! number of iterations for computing the integral and finding upper bound
foldcsi   2       ! number of folds on csi integration
foldy       5      ! number of folds on  y  integration
foldphi   2       ! number of folds on phi integration
nubound 50000  ! number of bbarra calls to setup norm of upper bounding function
icsimax  1       ! <= 100, number of csi subdivision when computing the upper bounds
iymax      1       ! <= 100, number of y subdivision when computing the upper bounds
xupbound 2d0   ! increase upper bound for radiation generation
```

- ncall1, itmx1: Perform a total of ncall1×itmx1 calls to the function btilde to initialize the grid.
- ncall2, itmx2: At fixed grids, the integral and the determination of the upper bounding functions for btilde and the remnants is carried out.
- foldcsi, foldy, foldphi. Generally used if too many negative weights show up (the folding
- number must be a divisor of 50)

The input card: gg_H_quark-mass-effects

Use [gg_H_quark-mass-effects_NNPDF31_13TeV_M125.input]() as the example

```
! OPTIONAL PARAMETERS

renscfact  1   ! (default 1d0) ren scale factor: muren  = muref * renscfact
facscfact  1   ! (default 1d0) fac scale factor: mufact = muref * facscfact
testplots  1     ! (default 0, do not) do NLO and PWHG distributions
hfact       60.0d0   ! (default no dumping factor) dump factor for high-pt radiation: > 0 dumpfac=h**2/(pt2+h**2)
runningscale 1       ! 0 = scales equal to the Higgs pole mass; 1  = scales equal to the Higgs virtuality;
                     ! 2 = scales equal to the Higgs pole mass for Born-like configuration and to the transverse mass for real
            emission contribution

iseed       SEED       ! initialize random number sequence
```

- hfact is used to avoid very large pT emission

The input card: gg_H_quark-mass-effects

Use [gg_H_quark-mass-effects_NNPDF31_13TeV_M125.input]() as the example

! GGF_H production:
! **** Mandatory parameters for ALL models ****
massren 0 ! Mass renormalization scheme. 0 = OS, 1 = MSBAR , 2 = DRBAR
zerowidth 0 ! Control if the Higgs boson is to be produced on-shell or not: 1 = On-Shell; 0 = Off-shell with Breit-Wigner
ew 1 ! ew = 0 disable EW corrections - ew = 1 enable EW corrections
model 0 ! model: 0 = SM
gfermi 0.116637D-04 ! GF
hdecaymode -1 ! PDG code for first decay product of the higgs
masswindow 1d0 !(default 10d0) number of widths around hmass in the BW for an off-shell Higgs boson

! **** Mandatory parameters for SM or MW ****
hmass 125d0 ! Higgs boson mass
hwidth 0.00407D0 ! Higgs boson width
topmass 172.5 ! top quark mass
bottommass 4.75d0 ! bottom quark mass - if defined it enables the bottom quark

! Optional
hdecaywidth 0 ! If equals to 1 read total decay width from HDECAY sm.br2 file
withnegweights 1
pdfreweight 0 ! PDF reweighting
storeinfo_rwgt 0 ! store weight information
bwshape 3 ! complex-pole scheme according to Passarino et al.

Hands-on to a powheg gridpack

- Login to lxplus8, cd to YOUR/WORK/DIRECTORY
- cmsrel CMSSW_12_4_8
- cd CMSSW_12_4_8/src/
- cmsenv
- git clone --depth 1 https://github.com/cms-sw/genproductions.git
- cd genproductions/bin/Powheg/
- curl -O https://raw.githubusercontent.com/Offshell-Workshop-LPC/Powheg-tutorial/main/tt.input
- python3 ./run_pwg_condor.py -p f -i tt.input -m hvq -f my_tt -q longlunch -n 1000

```
[melu@lxplus806 Powheg]$ python3 ./run_pwg_condor.py -p f -i tt.input -m hvq -f my_tt -q longlunch -n 1000

RUNNING PARAMS: parstage = f , xgrid = 1 , folderName = my_tt
                Total Events = 10000
                Number of Events = 1000
                powheg input cfg file : tt.input
                powheg process name : hvq
                working folder : my_tt
                EOS folder (stages 4,7,8) : NONE/my_tt
                base folder : /afs/cern.ch/work/m/melu/work/ttt/CMSSW_12_4_8/src/genproductions/bin/Powheg
                forDAG : 0
                scram_arch (set from environment) : el8_amd64_gcc10
                SVN : 0

Getting and compiling POWHEG source...
Running single job for grid
Creating tarball distribution for my_tt_hvq.tgz

Submitting to condor queues
```

check powheg gridpack validaity

```
[melu@lxplus806 Powheg]$ du -h hvq_el8_amd64_gcc10_CMSSW_12_4_8_my_tt.tgz
459K	hvq_el8_amd64_gcc10_CMSSW_12_4_8_my_tt.tgz
```

The gridpack size is very small, it's the feature of POWHEG. (MG gridpack can be ~O(Gb))
Basically for all kinds of gridpacks, one fast way to check it's valid or not is to untar it and do some LHE event generation locally.

- ➢ mkdir tmp
- ➢ cp hvq_el8_amd64_gcc10_CMSSW_12_4_8_my_tt.tgz tmp/
- ➢ cd tmp/
- ➢ tar xf hvq_el8_amd64_gcc10_CMSSW_12_4_8_my_tt.tgz

```
[melu@lxplus806 tmp]$ ls
bornequiv                                   Makefile          pwg-rwl.dat      pwhg_checklimits         runcmsgrid_par.sh
FlavRegList                                 powheg.input      pwg-stat.dat     pwhg_main                runcmsgrid.sh
hvq_el8_amd64_gcc10_CMSSW_12_4_8_my_tt.tgz  pwgcounters.dat   pwgubound.dat    realequivregions-btl     VERSION
include                                     pwggrid.dat       pwgxgrid.dat     realequivregions-rad     virtequiv
```

Check the file "pwg-stat.dat", **N.B, the XS here doesn't include the BR**

```
btilde pos.   weights:   714.12945888608738        +-   0.26029724362610185
btilde |neg.| weights:   3.0915934343899103        +-    3.6276423968918670E-002
btilde Total (pos.-|neg.|):   711.03786545171022        +-  0.26541531505962246
 Remnant cross section in pb   51.099247645201608        +-    4.3860407728416421E-002
 total (btilde+remnants) cross section in pb   762.13711309691178        +-  0.26901491563536323
 negative weight fraction:   4.0238341151834223E-003
```

check powheg gridpack validaity

- ./runcmsgrid.sh 100 10 >& tt.logs

```
         Running Powheg

%MSG-POWHEG number of events requested = 100
%MSG-POWHEG random seed used for the run = 10
%MSG-POWHEG number of cputs for the run =
%MSG-POWHEG SCRAM_ARCH version = el8_amd64_gcc10
%MSG-POWHEG CMSSW version = CMSSW_12_4_8
WARNING: In non-interactive mode release checks e.g. deprecated releases, production architectures are disabled.
numevts 100
iseed 10
ih1 1
ih2 1
ebeam1 6800
ebeam2 6800
lhans1 325300
lhans2 325300
```

```
Thanks for using LHAPDF 6.4.0. Please make sure to cite the paper:
  Eur.Phys.J. C75 (2015) 3, 132  (http://arxiv.org/abs/1412.7420)

 finished computing weights ..

-rw-r--r--. 1 melu zh 3169475 Mar 21 18:59 cmsgrid_final.lhe
/afs/cern.ch/user/m/melu/work/ttt/CMSSW_12_4_8/src/genproductions/bin/Powheg/tmp/powhegbox_hvq
Output ready with cmsgrid_final.lhe at /afs/cern.ch/user/m/melu/work/ttt/CMSSW_12_4_8/src/genproductions/bin/Powheg/tmp
End of job on  Thu Mar 21 18:59:18 CET 2024
```

check powheg gridpack validaity

If everything is fine, you should have "cmsgrid_final.lhe" now. The tt.logs and LHE file can be accessed from git. Every Event in LHE file with format

<event>

…

</event>

```
[melu@lxplus806 tmp]$ grep "/event" cmsgrid_final.lhe | wc -l
100
```

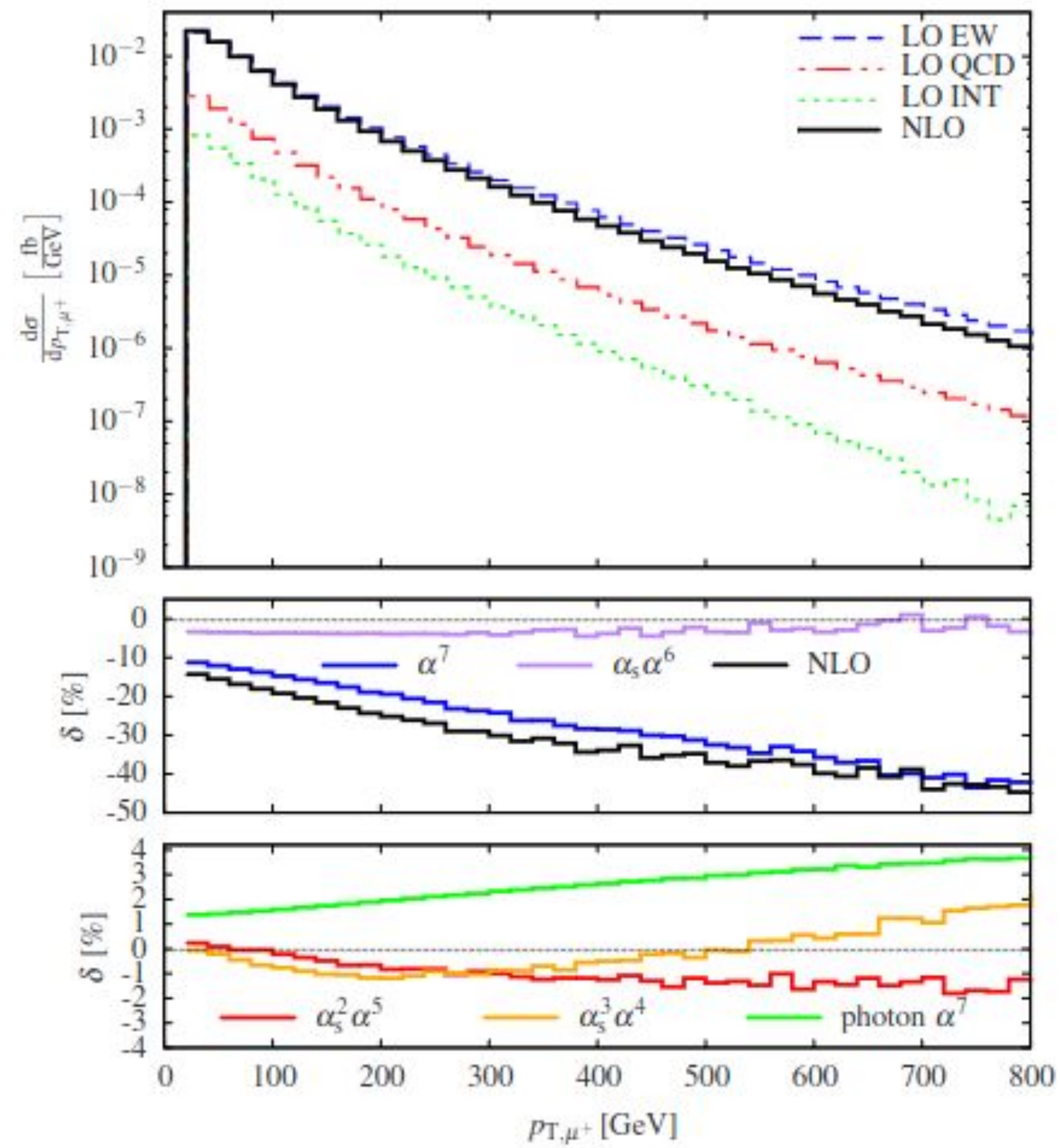
```
<event>
 13 322200  8.11039E+01  1.33255E+02 -1.00000E+00  1.17018E-01
  21    -1     0     0   504   505  0.000000000E+00  0.000000000E+00  5.995783349E+02  5.995783349E+02  0.000000000E+00  0.00000E+00  9.000E+00
  21    -1     0     0   502   503  0.000000000E+00  0.000000000E+00 -3.552168421E+02  3.552168421E+02  0.000000000E+00  0.00000E+00  9.000E+00
   6     2     1     2   502     0 -7.693434959E+01  8.228811151E+01  2.178009103E+02  2.996433512E+02  1.722169387E+02  0.00000E+00  9.000E+00
  -6     2     1     2     0   505 -1.901302547E+01 -1.747597816E+02  2.781065105E+02  3.704901627E+02  1.703445130E+02  0.00000E+00  9.000E+00
  21     1     1     2   504   503  9.594737506E+01  9.247167004E+01 -2.515459279E+02  2.846616630E+02  3.814697266E-06  0.00000E+00  9.000E+00
  24     2     3     3     0     0 -8.153409457E+01 -4.731985749E+00  1.531875167E+02  1.910424801E+02  7.975345550E+01  0.00000E+00  9.000E+00
   5     1     3     3   502     0  4.599744985E+00  8.702009726E+01  6.461339364E+01  1.086008712E+02  5.060000000E+00  0.00000E+00 -1.000E+00
 -24     2     4     4     0     0 -4.236934669E+01 -8.291277587E+01  7.170743721E+01  1.433171642E+02  8.202538067E+01  0.00000E+00  9.000E+00
  -5     1     4     4     0   505  2.335632121E+01 -9.184700568E+01  2.063990732E+02  2.271729985E+02  5.060000000E+00  0.00000E+00  1.000E+00
 -15     1     6     6     0     0  7.742395048E+00 -2.114759367E+01  2.216851182E+01  3.165068790E+01  1.777000000E+00  0.00000E+00  1.000E+00
  16     1     6     6     0     0 -8.927648962E+01  1.641560793E+01  1.310190048E+02  1.593917922E+02  0.000000000E+00  0.00000E+00 -1.000E+00
  13     1     8     8     0     0  2.495081375E+01 -2.353011206E+01  2.108830374E+01  4.026086198E+01  1.057000000E-01  0.00000E+00 -1.000E+00
 -14     1     8     8     0     0 -6.732016044E+01 -5.938266381E+01  5.061913347E+01  1.030563023E+02  0.000000000E+00  0.00000E+00  1.000E+00
```

E.g., this event is gluon-gluon induced, the ME process before decay is g g → t t~ g, then top → b W, and W undergoes the leptonic decay.

Other features

- NNLO+PS ➔ MiNNLOPS, this is the future!
- Electroweak correction
  - Sometimes the NLO EW correction is non-negligible, or even larger than the NLO QCD correction, e.g., p p > mu+ e+ ve vm j j
- Powheg interface to MG5

All the above materials can be found here



Summary

- POWHEG is an implementation of NLO+Shower MC, with the advantage of suppressing the negative weight event
- The code of POWHEG is process dependent, i.e., every process needs its own code
- There are many useful and interesting features in POWHEG, which have been widely used within CMS

Thanks!

Additional materials

$$d\sigma_{\rm NLO} = d\boldsymbol{\Phi}_n \Big\{ B(\boldsymbol{\Phi}_n) + V(\boldsymbol{\Phi}_n) + \left[R(\boldsymbol{\Phi}_n, \Phi_r) - C(\boldsymbol{\Phi}_n, \Phi_r) \right] d\Phi_r \Big\}$$

$$d\boldsymbol{\Phi}_{n+1} = d\boldsymbol{\Phi}_n \, d\Phi_r \qquad\qquad d\Phi_r \div dt\, dz\, d\varphi$$

$$V(\boldsymbol{\Phi}_n) = V_b(\boldsymbol{\Phi}_n) + \int d\Phi_r \, C(\boldsymbol{\Phi}_n, \Phi_r) \qquad \Longleftarrow \text{ finite}$$

$$d\sigma_{\rm SMC} = B(\boldsymbol{\Phi}_n) \, d\boldsymbol{\Phi}_n \left\{ \Delta_{t_0} + \frac{\alpha_s}{2\pi} P(z) \frac{1}{t} \Delta_t \, d\Phi_r \right\}$$

$$\Delta_t = \exp\left[- \int d\Phi_r' \frac{\alpha_s}{2\pi} P(z') \frac{1}{t'} \theta(t' - t) \right] \qquad \text{SMC Sudakov form factor}$$

$$d\sigma_{\rm POWHEG} = \overline{B}(\boldsymbol{\Phi}_n) \, d\boldsymbol{\Phi}_n \left\{ \Delta(\boldsymbol{\Phi}_n, p_T^{min}) + \frac{R(\boldsymbol{\Phi}_n, \Phi_r)}{B(\boldsymbol{\Phi}_n)} \Delta(\boldsymbol{\Phi}_n, p_T) \, d\Phi_r \right\}$$

$$\overline{B}(\boldsymbol{\Phi}_n) = B(\boldsymbol{\Phi}_n) + V(\boldsymbol{\Phi}_n) + \int d\Phi_r \left[R(\boldsymbol{\Phi}_n, \Phi_r) - C(\boldsymbol{\Phi}_n, \Phi_r) \right]$$

$$\Delta(\boldsymbol{\Phi}_n, p_T) = \exp\left[- \int d\Phi_r' \frac{R(\boldsymbol{\Phi}_n, \Phi_r')}{B(\boldsymbol{\Phi}_n)} \theta\left(k_T(\boldsymbol{\Phi}_n, \Phi_r') - p_T \right) \right] \qquad \text{POWHEG Sudakov}$$

MC@NLO vs POWHEG

The advantage of this method is that it does not require to modify the showering code in order to be implemented. Observe that the NLO correction in (ii) is always a hard correction, since collinear and soft singularities have been removed from the NLO formula by subtracting the Monte Carlo NLO approximation. The disadvantages of this approach are

- The NLO calculation has to be tuned to the Monte Carlo, by changing the projection $\mathbb{P}$. It becomes therefore Monte Carlo dependent.

Pythia/Herwig

- One has to determine the Monte Carlo NLO approximation, a task that is not always simple.
- The Monte Carlo implementation of the soft limit may be less than perfect. Thus, one has to make sure that the remaining IR sensitivity in the difference between the exact and the Monte Carlo NLO expression does not have sensible consequences. This last problem is in fact, in practice, a very minor problem.
- The NLO correction needs not be positive. Thus, one is forced to accept negative weighted events.