Monte Carlo Generators in CMS for Run II

Josh Bendavid (Caltech, LPC DR)
for the CMS Generators group (+ some personal annotations)

Caltech

Mar. 22, 2017
LHCTheory ERC Meeting

Introduction

- Technical integration and workflows for generator tools in CMS
  - Important for the theory community, and especially generator authors to understand **how** we are using external generator programs in CMS at both a physics and technical level
  - Better understanding of our needs and workflows $\rightarrow$ more effective use of generators in CMS, suggestions for how we can improve on what we are doing
- Most commonly used generators and standard configurations for Run II
  - Matrix element generators, parton showers, PDF's, underlying event tunes
- More detailed feedback on MG5_aMC@NLO
  - Past challenges and solved/improved technical aspects
  - Open issues and possible future improvements

CMS Software Overview

- Main CMS software application: CMSSW
- Modular C++ application which can be used for event generation, detector simulation, reconstruction, and analysis
- Configuration of CMSSW runs is steered with python-based configuration files
- Input and output with root-based EDM files, which store run-level, lumi-section-level (23s periods for real data), or event-level data products
- CMSSW links directly to many **externals**, externally maintained C, C++, fortran, or python software which is either an indirect dependency or is directly called from within CMSSW
- Externals are compiled with the same common libraries, compiler version as CMSSW and packaged together with a given release, starting from either a tarball from the author's website, from GENSER, or from a cms-managed github mirror

CMS Production Overview

- Python-based tools manage large-scale submission of CMSSW jobs to grid resources for central production of Monte Carlo, data processing, etc
- Jobs are assumed to be CMSSW jobs configured by the corresponding python-based configuration
- All input and output are assumed to be EDM files (with a few special cases)
- A similar mechanism is available to end users to submit analysis jobs
- CMSSW software and corresponding externals is made available on worker nodes through CVMFS (distributes http-based read-only filesystem)

CMS Software: Event Generation

- Basic paradigm: A C++ module with a common interface makes the needed calls to a linked external generator code in order to produce for each event a HepMC::GenEvent, which can then directly stored in the EDM output
- Configuration of the generator takes place within the CMSSW python configuration
- **Advantages:**
  - Uniform configuration and IO mechanism (production tools only have to deal with CMSSW)
  - No intermediate files needed (HepMC::GenEvent is passed along in memory to standard CMSSW/root IO mechanisms or directly to GEANT, which is also called from inside CMSSW)
- **Disadvantages:**
  - Each generator needs a dedicated interface in CMSSW and must be packaged as a CMSSW external
  - Initialization and event generation calls must be possible from within a C++ application
- In practice, Pythia, Herwig, Sherpa fit very nicely into this paradigm (with some preference for C++-based versions)

Example CMSSW GEN Configuration Fragment

```
import FWCore.ParameterSet.Config as cms

from Configuration.Generator.Pythia8CommonSettings_cfi import *
from Configuration.Generator.Pythia8CUEP8M1Settings_cfi import *

generator = cms.EDFilter("Pythia8GeneratorFilter",
  maxEventsToPrint = cms.untracked.int32(1),
  pythiaPylistVerbosity = cms.untracked.int32(1),
  filterEfficiency = cms.untracked.double(1.0),
  pythiaHepMCVerbosity = cms.untracked.bool(False),
  comEnergy = cms.double(13000.0),

  crossSection = cms.untracked.double(1.92043e+07),

  PythiaParameters = cms.PSet(
    pythia8CommonSettingsBlock,
    pythia8CUEP8M1SettingsBlock,
    processParameters = cms.vstring(
      'HardQCD:all = on',
      'PhaseSpace:pTHatMin = 50  ',
      'PhaseSpace:pTHatMax = 80  ',
    ),
    parameterSets = cms.vstring('pythia8CommonSettings',
                                'pythia8CUEP8M1Settings',
                                'processParameters',
                               )
  )
)
```

CMS Software: LHE Input

- CMS maintains its own LHE parser (based on xerces-c xml library)
- An LHE file can be read as input to a CMSSW job and is converted on the fly to C++ classes LHERunInfoProduct and LHEEventInfoProduct which store the relevant information and can be stored/read from EDM files (support for per-event weights added to CMS lhe parser and classes)
- LHE information can be passed as input to a hadronizer as part of the event generation step in CMSSW (using for example the Pythia8::LHAup mechanism to pass the needed information on the fly in memory)
- LHE parsers included with Pythia, Herwig etc are not used
- **Advantage:** Uniform hadronizer-independent storage and access to lhe information
- **Disadvantage:** We have to maintain our own LHE parser

LHE Input for Central Production

- CMS production tools do not work transparently with ascii LHE input (metadata not automatically available in data management system, skipping of events is inefficient, etc)
- It is possible to use privately produced LHE files for central production (user copies the files to eos and then a conversion step is run to produce EDM files containing the LHE products, which can then be used for further production steps for hadronization, simulation, etc)
- Disk space, file corruption, etc, are major issues when dealing with large sets of lhe files in this way

Central production of LHE events

- LHE generators like Madgraph_aMC@NLO, POWHEG, etc cannot be directly called from within CMSSW in general
- Solution is "externalLHEProducer" a C++ CMSSW module which calls an external script, then reads the resulting LHE file (with the CMS lhe parser) and produces the necessary LHERunInfoProduct/LHEEventInfoProduct which can be stored in the EDM file and/or passed along to the hadronizer
- Further issue: LHE generator code cannot easily be included with CMSSW as an external, since each process requires dedicated (and sometimes dynamically generated) libraries
- Solution: "gridpacks" with pre-generated/compiled code, and with initial phase space integration results stored in a tarball
- Gridpacks are put in CVMFS and can be accessed by jobs (gridpack location is a configuration parameter of the externalLHEProducer module)
- Minimal and compact external input, and compressed EDM output make very large scale LHE production possible.
- CMS has produced over 30 billion LHE events (before matching) through this mechanism for the initial Run 2 campaign

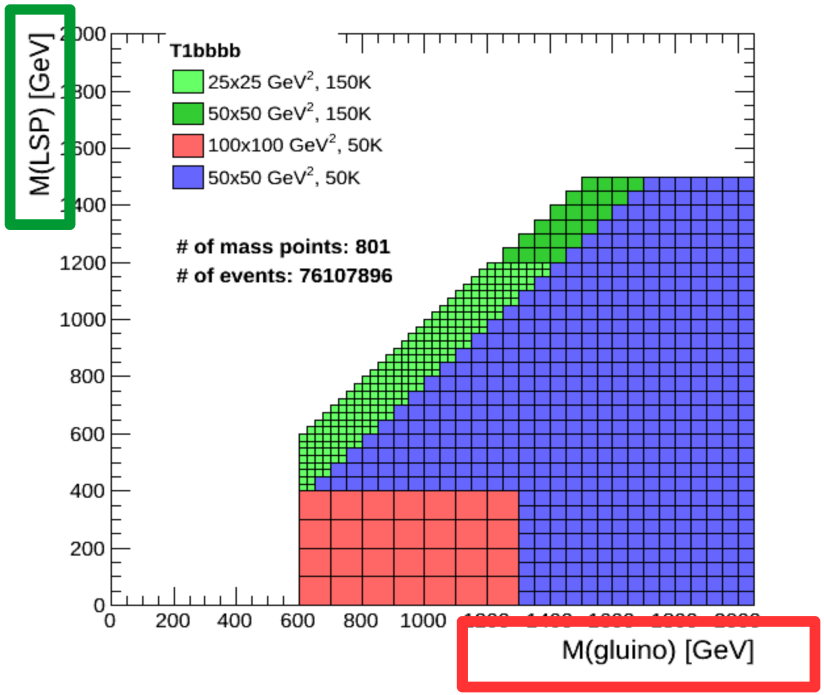
Gridpacks

- General gridpack mechanism used in CMS is modeled on the built-in functionality for LO processes in Madgraph_aMC@NLO
- We maintain scripts for Madgraph_aMC@NLO (including NLO processes), POWHEG, JHUGen to produce gridpack tarballs based on the appropriate input cards
- Important considerations:
  - Compiling code on batch workers is discouraged (should be possible to fully precompile everything)
  - Long initialization time for event generation is discouraged
  - Gridpack size is an issue (more than about 500MB for the tarball or 5GB decompressed starts to become problematic)
  - (For Madgraph_aMC@NLO we use lzma compression with very large dictionaries because of large use of space from duplicated code in statically linked executables for each subprocess)
  - Gridpack generation step needs reliability and reasonable run-time "as the physicist waits" (we can use multi-core machines and/or condor/lsf batch queues to do the phase space integration, but does no good if process is bottle-necked by single-threaded steps, or individual long-running jobs)

Parameter Scans

- Recently added some functionality in CMSSW for parameter scans (used so far for SUSY signal MC)
- SUSY signal production with MG5_aMC@NLO + Pythia 8 (LO MLM)
- Typical case: gluino/squark pair production (+0,1,2 jets LO) in MG5_aMC@NLO, decay in Pythia, steered by SLHA table
- Produce one gridpack eg. for each gluino mass (can this be improved in the future?)
- Gridpack and pythia configuration+SLHA table for decays are randomly selected for each luminosity section ($\sim$ 200 events after matching)
- Resulting sample contains a mixture of all scan points
- High granularity of randomization ensures missing events from job failures are randomly and $\sim$ evenly distributed across scan points
- For technical reasons this implementation also bypasses the CMSSW LHE parser and uses the Pythia 8 (or eventually Herwig 7) one directly (may migrate to this in general in the future, but some missing functionality in CMSSW and/or Pythia8 for general use related to storage and reuse of LHE information, book-keeping of LHE header information)

Parameter Scans



- Horizontal axis tied to madgraph LHE production (ie choice of gridpack)
- Vertical axis tied to pythia configuration/SLHA table for decays
- Example scan defined with python loops and string replacement:

`https://github.com/cms-sw/cmssw/blob/CMSSW_8_0_26/GeneratorInterface/Pythia8Interface/test/randomizedParametersSLHALHE.py`

Standard Configurations for CMS Run 2 Monte Carlo

- Pythia8 Standalone
  - Mainly for QCD, especially with additional generator level filters, where LHE-based production has additional complications (though "multiple hadronization" feature has been added to CMSSW to make this more feasible)
- POWHEG-BOX + Pythia 8 (power showers with emission veto)
  - Used mostly for Higgs signals, diboson production, and $t\bar{t}$ production
  - MINLO and NNLOPS starting to be used for available processes
- (Run 1 CMS MC used mainly Pythia 6 standalone, Madgraph5+Pythia 6 with/without MLM matching, POWHEG+Pythia 6, little bit of Sherpa 1.x and Herwig 6)

Standard Configurations for CMS Run 2 Monte Carlo

- MG5_aMC@NLO + Pythia 8
  - LO without jet matching used for many exotic signal samples with BSM models
  - LO with MLM matching (up to 4 additional partons depending on process) used for boosted/multijet phase space for search backgrounds with W/Z/γ+jets, QCD multijet, $t\bar{t}$+jets, and for SUSY signal samples
  - NLO (without merging) used for a few complex processes where extra jets are either computationally expensive (ttW/Z/γ, *ttbb*, etc), or not possible with FXFX (γ+jets, dijets, VBF, etc)
  - NLO with FXFX merging (up to 2 additional partons at NLO) used for $Z/W + jets$, dibosons, $t\bar{t}$, some Higgs signals
  - CKKW/UMEPS/UNLOPS have not been used due to difficulty of stitching together separate samples (solved now) and additional weights, both magnitude and sign (maybe solved now also at least for LO CKKW?)

Standard Configurations for CMS Run 2 Monte Carlo: PDF's

- Standardized on NNPDF30+uncertainties so far for most samples, using appropriate LO, NLO, 4fs and 5fs variations
- Additional weights for variations of central pdf+α_S included for $\sim$all POWHEG and MG5_aMC@NLO samples
- Additional weights for alternate pdfs+uncertainties included for POWHEG, and LO MG5_aMC@NLO (was not previously possible at NLO, will be included in future productions)
- (Pythia8-only samples used NNPDF23LO1 and no variations)
- Central and alternate PDF sets will be updated for 2017 production

Standard Configurations for CMS Run 2 Monte Carlo: Pythia 8 Tune

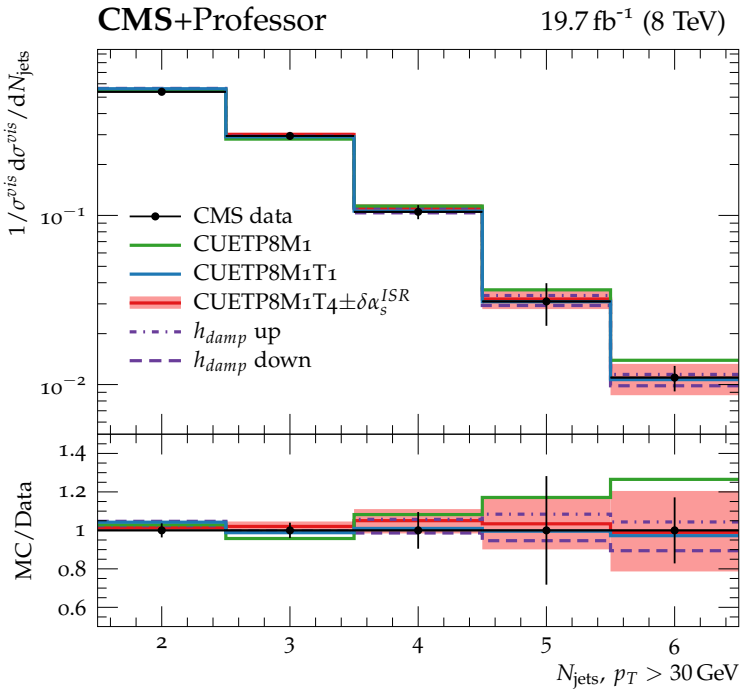
- So far have used Tune CUETP8M1 for most samples (arXiv:1512.00815)
  - Re-tuning of UE parameters on top of Monash, α_S and other shower parameters left untouched
  - In particular this means α_S=0.1365 used for both ISR and FSR in the shower, despite using 0.118 in the ME for NLO samples and 0.130 for LO samples
- Tuning of shower parameters in particular being revisited in future production

Additional Configurations for CMS Run 2 Monte Carlo

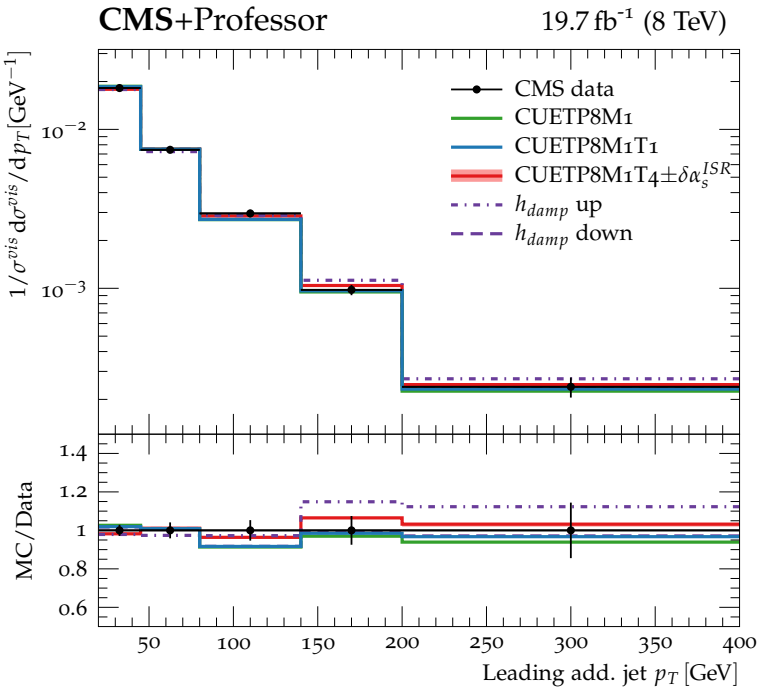
- Sherpa 2.X used for some samples (especially diphoton+jets)
- POWHEG+Herwig++ (wimpy showers) used for systematics vs Pythia 8
- MG5_aMC@NLO+Herwig++ (no merging)
- Herwig++ standalone used for QCD and MinBias
- Herwig 7 integration in progress
- JHUGen used for anomalous Higgs spin/parity studies, $H \to ZZ$ decays in Higgs signal samples
- Several other generators used for special samples

Pythia 8 Shower Tuning with $t\bar{t}$ (CMS-PAS-TOP-16-021)

- Differences observed between generators, and sensitivity to shower α_s in $t\bar{t}$ production in kinematics/multiplicity of additional jets
- Retune PS ISR α_s and POWHEG hdamp using POWHEG+Pythia $t\bar{t}$ vs dilepton+jets data, yields (much) lower value of $\alpha_s^{ISR} = 0.1108$



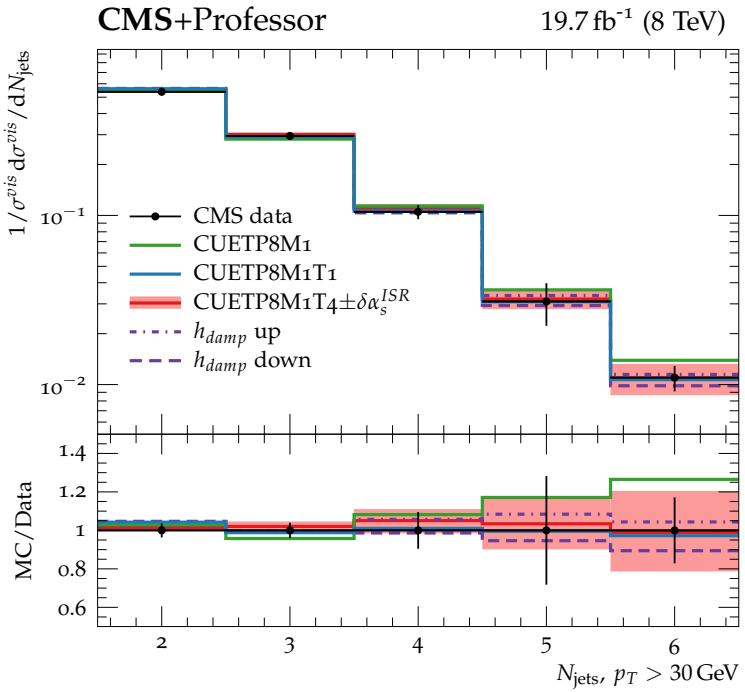
(a) Njets (p_T >30 GeV)



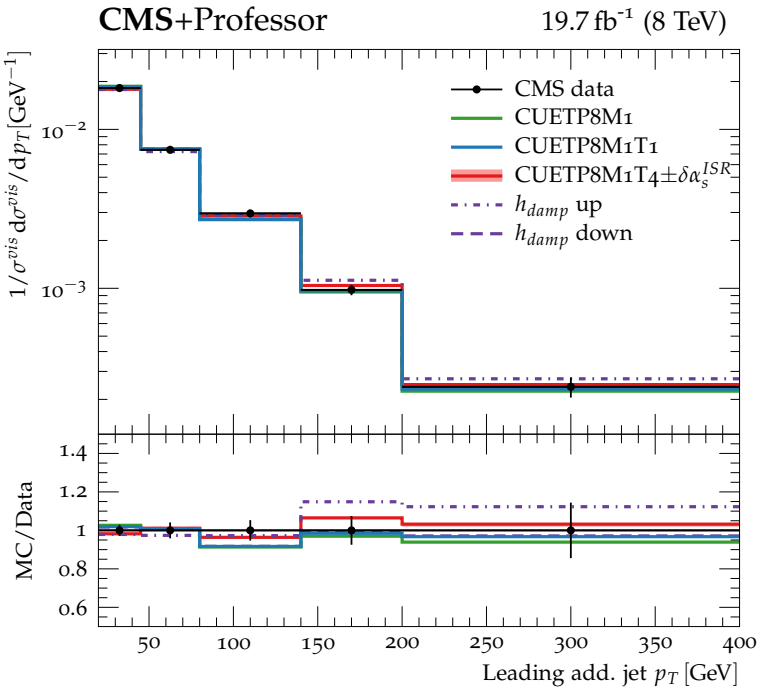
(b) Lead jet p_T

Pythia 8 Shower Tuning with $t\bar{t}$ (CMS-PAS-TOP-16-021)

- Differences observed between generators, and sensitivity to shower α_s in $t\bar{t}$ production in kinematics/multiplicity of additional jets
- Retune PS ISR α_s and POWHEG hdamp using POWHEG+Pythia $t\bar{t}$ vs dilepton+jets data, yields (much) lower value of $\alpha_s^{ISR} = 0.1108$



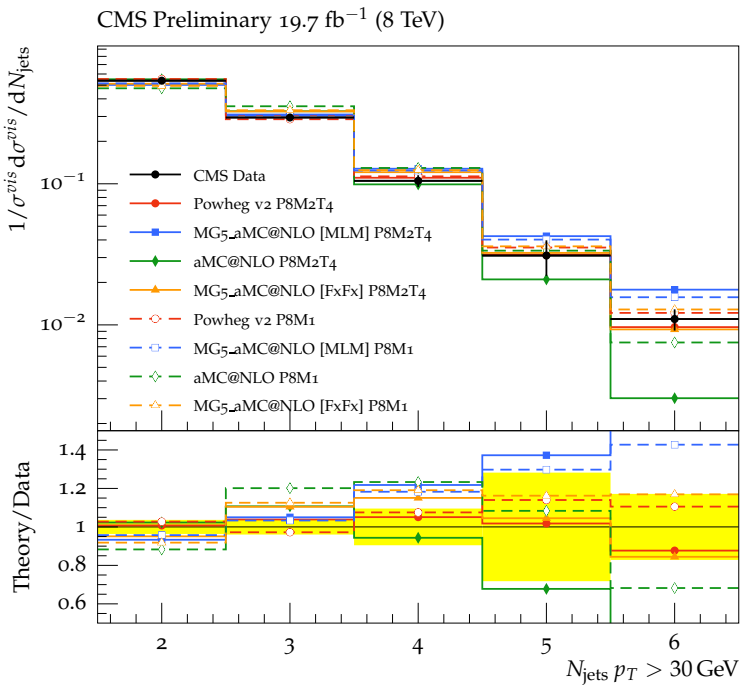
(a) Njets (p_T >30 GeV)



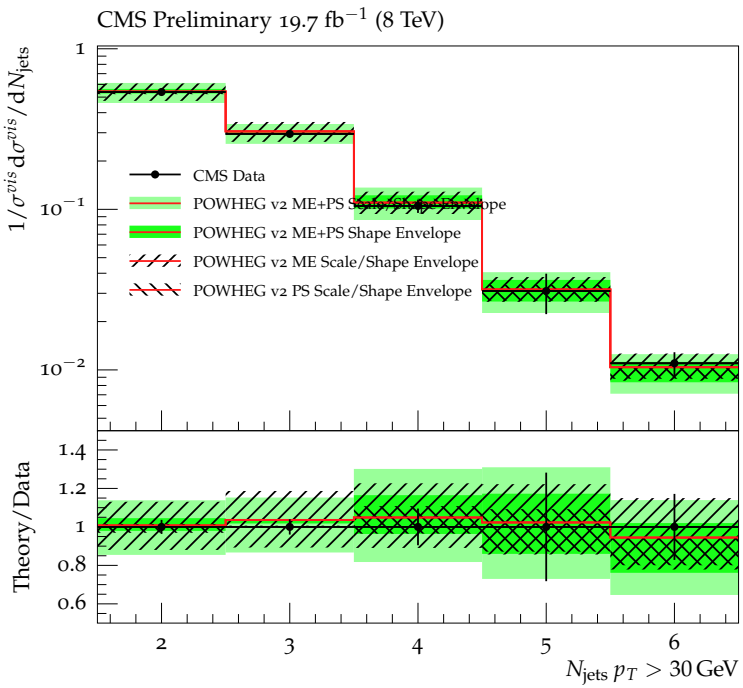
(b) Lead jet p_T

Pythia 8 Shower Tuning with $t\bar{t}$ (CMS-PAS-TOP-16-021)

- Smaller α_s^{ISR} somewhat favoured for MG5_aMC@NLO+Pythia 8 with FXFX merging, but uncertainties are large
- More systematic cross comparison of total uncertainty between generators would be useful
- Impact of shower starting scale in mc@NLO configurations?



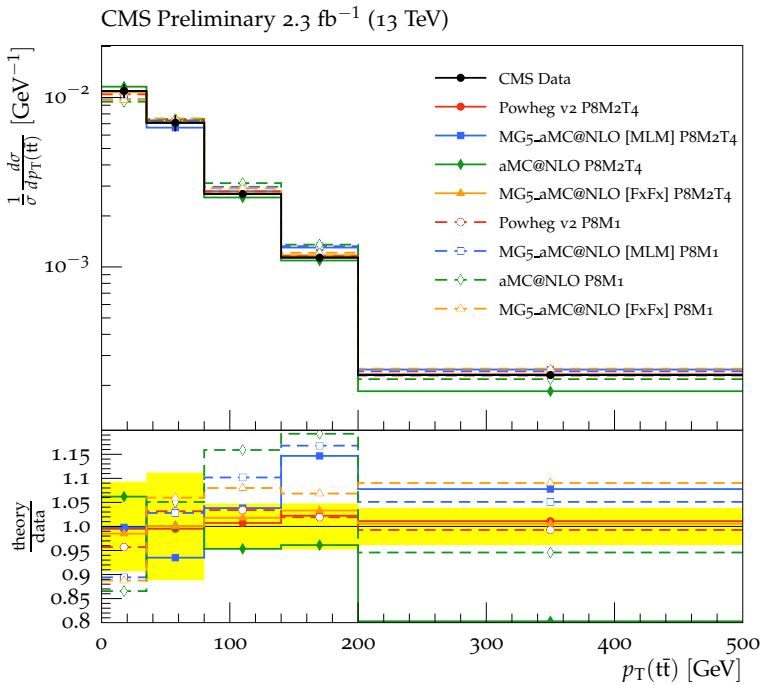
(a) Generator Comparison



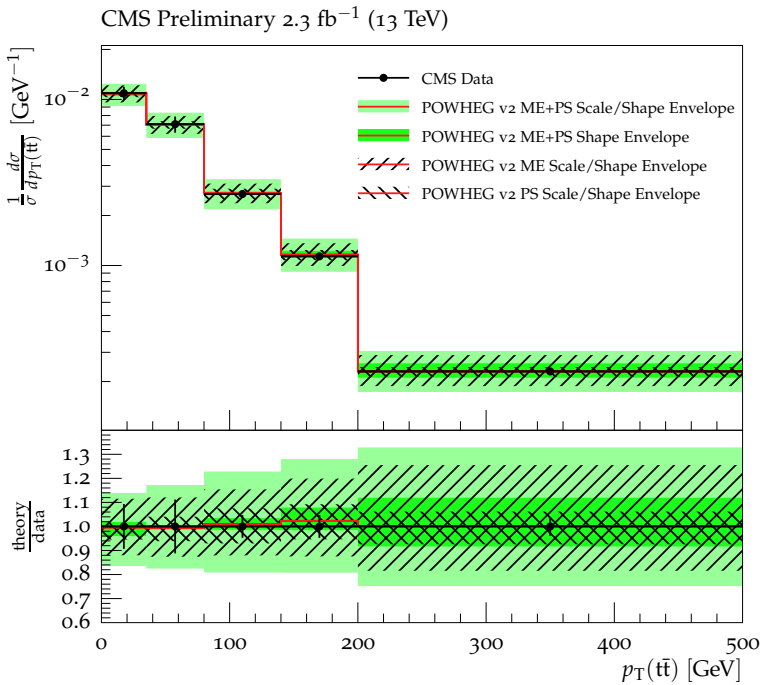
(b) POWHEG+Py8 Uncertainties

Pythia 8 Shower Tuning with $t\bar{t}$ (CMS-PAS-TOP-16-021)

- More systematic cross comparison of total uncertainty between generators would be useful
- Need to make sure we understand the level of predictivity of each configuration relative to the underlying precision
- Some additional comparisons planned for LO MLM configuration with older mg+py6 configuration to make sure changes are understood



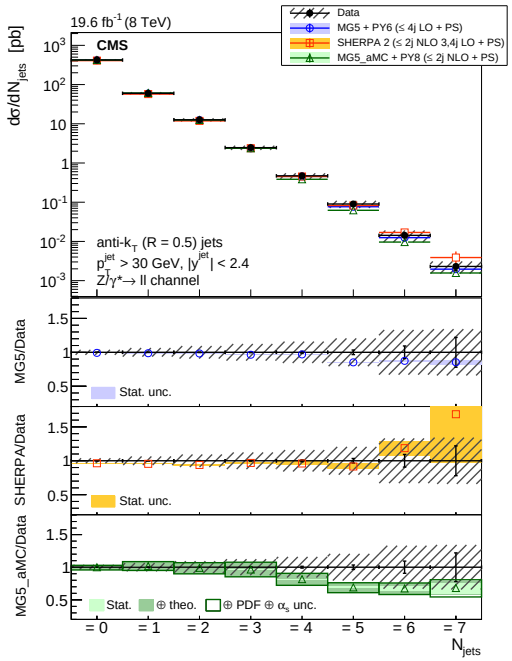
(a) Generator Comparison



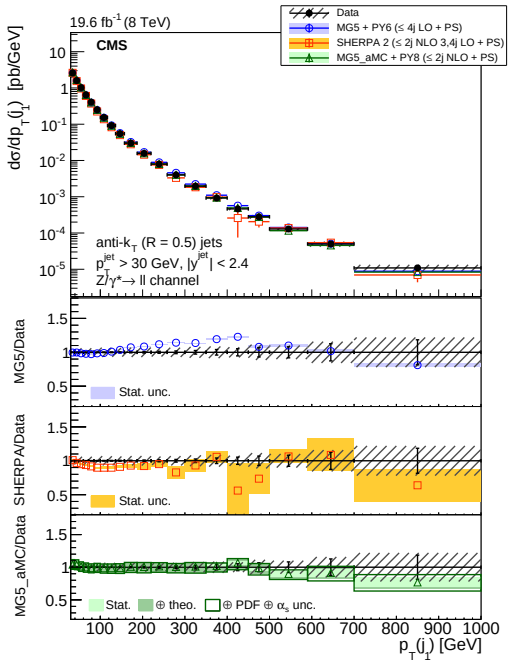
(b) POWHEG+Py8 Uncertainties

Meanwhile for V+jets

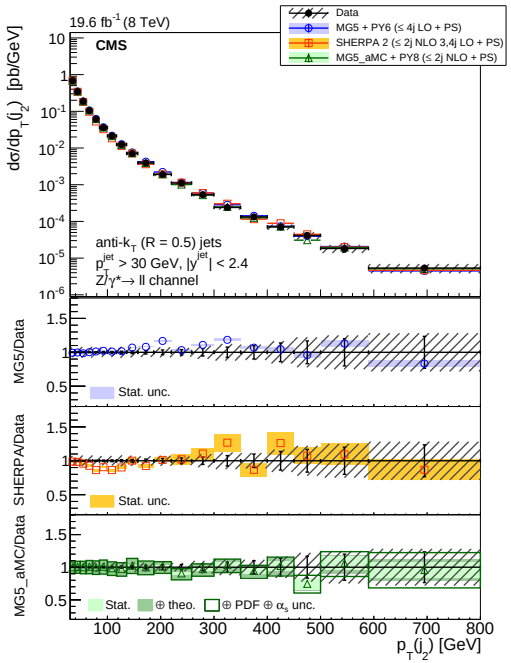
- Generally very good agreement for merged NLO predictions of V+jets, and also much less sensitivity to shower parameters
- Example here for 8 TeV Z+jets (arXiv:1611.03844)



(a) Njets



(b) lead jet p_T



(c) sub-lead jet p_T

MG5_aMC: Some technical particularities for CMS usage

- mg_aMC is configured to never artificially add masses to particles in the LHE files if massless in the physics model
- Verified that recent versions of Pythia8 used in CMSSW correctly handles massless lhe input particles for both leptons and b/c quarks (should this be made the default?)
- Nominal and alternate event weights (for scale and pdf variations) are propagated through to LHEEventProduct
- Nominal event weight is also propagated to GenEventInfoProduct::weight() (but may be further modified by matching/parton shower/hadronization in special cases)

MG5_aMC: Some physics particularities for CMS usage

- Jet matching schemes fully tested and widely used in CMS production are MLM (LO) and FXFX (NLO)
- Generally preferred to produce samples with jet matching where computationally feasible, for better description of additional jets. Generally preferred to use 5fs with massless b-quarks, b's included in proton/jet definition in order that extra b's are also modeled
- LO vs NLO is a tradeoff for accuracy at lower multiplicities, vs fewer additional jets computationally feasible, and (possibly large fraction of) negative weights for NLO
- Full set of example cards for 4fs/5fs/LO/NLO with/without jet matching in `https://github.com/cms-sw/genproductions/tree/master/bin/MadGraph5_aMCatNLO/cards/examples`

MG5_aMC: Caveats

- Gridpack production can be slow (several days)
- Any error requires starting from the beginning
- lsf can be error-prone or busy
- Condor support in principle working, but not extensively tested (more relevant now that CERN is migrating from lsf to condor)
- NLO processes (especially with extra jets) may have a large fraction of negative weights (up to 40% in extreme cases)
- Default Pythia8 parton shower parameters not very well tuned, can also spoil jet multiplicity distributions even with NLO or jet matching (also true with POWHEG+Pythia8, but less sensitive in inclusive phase space because first emission not by Pythia)

MG5_aMC: Successes

- Several key technical issues addressed as a result of feedback or code contributions from CMS experts:
  - Gridpack-functionality and single-sample FXFX merging with soup of jet multiplicities for NLO generation
  - Gridpack-functionality for additional steps like Madspin, or reweighting by alternate physics parameters
  - More robust handling of lsf batch jobs/retries (and similar work for Condor in progress)
  - Handling of weights for multiple pdf sets
  - (Single-node) parallelization and drastically reduced memory usage of NLO code generation
- Communication model with CMS (launchpad + dedicated mailing list for MC conveners and experts) largely successful
- **Large numbers (billions) of events produced at scale and widely used in CMS, including at NLO with FXFX merging**

MG5_aMC: Provocative (Personal) Comments and Speculation

- Is there low hanging fruit where a modest amount of work on the theory/technical side may yield large practical gains for the experiments?
- More efficient/even splitting of work between batch jobs, reducing maximum job length to enable more complex processes/higher jet multiplicities (Multi-threaded or MPI phase space integration? Partitioning of phase space?)
- **POWHEG-type matching** as an option in MG5_aMC@NLO?
  - More directly assess shower-matching systematics
  - Reduce negative weights with phase-space folding a la POWHEG-BOX?
  - Trivial multiplicity merging possible with shower-k_T-like algorithm? MINLO?

MG5_aMC: Provocative (Personal) Comments and Speculation

- Is there low hanging fruit where a modest amount of work on the theory/technical side may yield large practical gains for the experiments?
- Handle MC@NLO matched emissions (ie the first emission) with some kind of simple **internal parton shower implementation?** (ie all events become H-type events at LHE level)
  - Simplify configuration of external shower and reduce the possibility of mismatches which invalidate the matching
  - Might enable additional tricks to reduce negative weights?
  - Use lower multiplicity matrix elements + shower emissions to integrate higher multiplicity matrix elements a la VINCIA?

BACKUP

Source Code Availability

- CMSSW and all of its dependencies and externals (and their dependencies) are open-source
- Practical requirement: Externals (and any code included in gridpacks) need to be compiled with compatible compilers/libraries/etc to link to CMSSW and/or run within a CMSSW environment on the grid
- Further Benefit: Substantial manpower and expertise within the experimental collaborations. We are happy to help debug issues with the software we are using. Source code (and publicly accessible cvs/svn/git repository) make it much easier for us to do this
- Further consideration: We should be able to know exactly what we are putting into the Monte Carlo samples used for our papers

Patching of Generators

- We strongly prefer not to apply our own patches to generator code for obvious reasons of reproducibility and communication outside CMS
- Release schedules for generator tools don't always line up with our own production campaign schedules, so sometimes patches are necessary (but we always discuss it with the authors first at least)
- Very long gap between releases can particularly make this an issue

Schedule Reminder

- Lead-time to produce billions of fully simulated and reconstructed Monte Carlo events is long
- Pythia-based production for Run 2 started in October 2014
- LHE-based production started (late) in Feb. 2015
- Lots of inertia to change things like tunes, pdf's, etc