

Stefano Frixione
(on behalf of the authors)

Status of MG5_aMC@NLO

LHC EW WG meeting

CERN, 13/11/2018

The current public versions are:

MG5_aMC@NLO v2.6.4

MG5_aMC@NLO v3.0.0 β

released on 9/11/2018 and 7/5/2018 respectively

W.r.t. the v2.5 series:

- ▶ New functionalities and some bug fixes
- ▶ Mixed-coupling expansions (v3.0.0, fixed-order only)

The basic strategy is that of vastly expanding the scope of the code, while maintaining (and improving where necessary) its core capabilities of a “hadron collider QCD tool”

- ▶ Non-SM physics (involved models)
- ▶ Different environments (e^+e^- , non-collider DM)
- ▶ Involved perturbative structures (mixed-coupling expansions)

Lots of this through user-controlled hooks (plugins)

2.6.0:

- Bias function for fNLO and NLO+PS 
- Various improvements to the reweight module 
- Expand capabilities of `systematics.py`
- Clean up code for plugins, NLO weights, internal histogramming package (HwU)
- Bug fixes to MadSpin (potentially serious, but marginal: affects 2 → 1 processes), and `montecarlocounter.f` (subleading colours: we have found no visible effects)

2.6.1:

- LO matrix elements with $F_x F_x$ 
- Online model database (`MG5_aMC> display model_list`)
- Supports LHAPDF v6.2
- Supports PDG-number-based cuts (beware at the NLO)

2.6.2/2.6.3.X/2.6.4:

- Some changes to PDF support (added ion PDFs, removed obsolete ones)
- Several improvements and bug fixes (nothing major, but suggests upgrading)

3.0.0 β :

- Start of *official* support for mixed-coupling computations
- NLO EW+QCD most prominent
- If used, please cite [arXiv:1804.10017](#) in addition to arXiv:1405.0301.
One is very strongly advised to take at least a quick look at that paper

For a complete list of the changes, see the Update notes

Types of computations available:

- ▶ fLO Fixed order, tree level
- ▶ fNLO Fixed order, NLO
- ▶ LO+PS Hard tree-level events, showered
- ▶ NLO+PS Hard NLO events, showered (MC@NLO)
- ▶ MLM/CKKW-merged Multijet tree-level merging
- ▶ FxFx- or UNLOPS-merged Multijet NLO merging

The Lagrangian-to-events chain is automated through:

FeynRules(+NLOCT) - MG5_aMC@NLO - PSMC

The emphasis of this meeting is on V +jets

- ◆ One of the primary examples of the necessity of using merged computations, *systematically improvable* by increasing matrix-element parton multiplicities
- ◆ In MG5_aMC, merged results can be obtained both at the tree-level and at the NLO
- ◆ Both have been fully automated for a while. I'll concentrate on the NLO case, which is more compelling from a physics viewpoint
(reduced uncertainties)

At present, the vast majority of applications of NLO merging within MG5_aMC are carried out with FxFx

- ◆ The method has been presented in [1209.6215](#), and the current implementation strictly follows that paper
- ◆ The full automation with both [PY8](#) and [HW++](#) has been achieved in 2015. No significant changes since then
- ◆ However, significant extensions of scope/efficiency: [reweight](#), [bias](#), and “[LO MEs](#)”
- ◆ General ideas relevant to phenomenology applications are given in [1511.00847](#), which specifically deals with V +jets production

FxFx in a nutshell

- ▶ Introduce a function $D(\mu)$, that allows one to distinguish ME-dominated, MC-dominated, and intermediate regions. E.g.:

$$D(\mu) = \Theta(\mu_Q - \mu)$$

This also defines the merging scale $\mu_Q \leftrightarrow$ merging systematics

- ▶ Use k_T -jet rates to define a scale hierarchy

$$d_n \leq d_{n-1} \leq \dots \leq d_2 \leq d_1 \leq d_0 \equiv \sqrt{\hat{s}}$$

- ▶ The i -parton sample is basically an MC@NLO one, times a suitable combination of damping factors $D(d_k)$ so that, apart from $i = N$ (the highest multiplicity), it will dominate i -jet configurations

FxFx formulae (p Born-level particles)

$$\begin{aligned}
 d\bar{\sigma}_{p+i}^{(\mathbb{S})} &= \left[\sum_{\alpha=S,C,SC} d\sigma_{p+i}^{(\text{NLO},\alpha)} + d\sigma_{p+i}^{(\text{MC},0)} D(d_{i+1}(\mathcal{K}_{p+i}^{(\mathbb{H})})) \right] \\
 &\quad \times \left(1 - D(d_i(\mathcal{K}_{p+i}^{(\mathbb{S})})) \right) \Theta \left(d_{i-1}(\mathcal{K}_{p+i}^{(\mathbb{S})}) - \mu_Q \right) \\
 d\bar{\sigma}_{p+i}^{(\mathbb{H})} &= \left[d\sigma_{p+i}^{(\text{NLO},E)} \left(1 - D(d_i(\mathcal{K}_{p+i}^{(\mathbb{H})})) \right) \Theta \left(d_{i-1}(\mathcal{K}_{p+i}^{(\mathbb{H})}) - \mu_Q \right) \right. \\
 &\quad \left. - d\sigma_{p+i}^{(\text{MC},0)} \left(1 - D(d_i(\mathcal{K}_{p+i}^{(\mathbb{S})})) \right) \Theta \left(d_{i-1}(\mathcal{K}_{p+i}^{(\mathbb{S})}) - \mu_Q \right) \right] D(d_{i+1}(\mathcal{K}_{p+i}^{(\mathbb{H})})) \\
 d\bar{\sigma}_{p+N}^{(\mathbb{S})} &= \left[\sum_{\alpha=S,C,SC} d\sigma_{p+N}^{(\text{NLO},\alpha)} + d\sigma_{p+N}^{(\text{MC},0)} \right] \\
 &\quad \times \left(1 - D(d_N(\mathcal{K}_{p+N}^{(\mathbb{S})})) \right) \Theta \left(d_{N-1}(\mathcal{K}_{p+N}^{(\mathbb{S})}) - \mu_Q \right) \\
 d\bar{\sigma}_{p+N}^{(\mathbb{H})} &= d\sigma_{p+N}^{(\text{NLO},E)} \left(1 - D(d_N(\mathcal{K}_{p+N}^{(\mathbb{H})})) \right) \Theta \left(d_{N-1}(\mathcal{K}_{p+N}^{(\mathbb{H})}) - \mu_Q \right) \\
 &\quad - d\sigma_{p+N}^{(\text{MC},0)} \left(1 - D(d_N(\mathcal{K}_{p+N}^{(\mathbb{S})})) \right) \Theta \left(d_{N-1}(\mathcal{K}_{p+N}^{(\mathbb{S})}) - \mu_Q \right)
 \end{aligned}$$

The actual implementation also features a CKKW-like Sudakov reweighting and rejection

Some remarks:

- ▶ This is NLO, so uncertainties are supposed to be meaningful:
do vary μ_Q in an appropriate way

Some remarks:

- ▶ This is NLO, so uncertainties are supposed to be meaningful:
do vary μ_Q in an appropriate way
- ▶ In particular: when possible, check values of μ_Q both sides of the $p_T(j)$ *analysis* cut (often, smooth at the NLO and kink-ish at tree level)

Some remarks:

- ▶ This is NLO, so uncertainties are supposed to be meaningful: do vary μ_Q in an appropriate way
- ▶ In particular: when possible, check values of μ_Q both sides of the $p_T(j)$ *analysis* cut (often, smooth at the NLO and kink-ish at tree level)
- ▶ Technical: if a $p_T(j)$ *generation* cut is applied, then choose $p_T(j) < \mu_Q/2$

Some remarks:

- ▶ This is NLO, so uncertainties are supposed to be meaningful: do vary μ_Q in an appropriate way
- ▶ In particular: when possible, check values of μ_Q both sides of the $p_T(j)$ *analysis* cut (often, smooth at the NLO and kink-ish at tree level)
- ▶ Technical: if a $p_T(j)$ *generation* cut is applied, then choose $p_T(j) < \mu_Q/2$
- ▶ Having assessed the μ_Q systematics, “tuning” μ_Q is possible, although I dislike it (I dislike the tuning of any NLO-matching parameters, which I see as a sort of contradiction in terms)

Some remarks:

- ▶ This is NLO, so uncertainties are supposed to be meaningful: do vary μ_Q in an appropriate way
- ▶ In particular: when possible, check values of μ_Q both sides of the $p_T(j)$ *analysis* cut (often, smooth at the NLO and kink-ish at tree level)
- ▶ Technical: if a $p_T(j)$ *generation* cut is applied, then choose $p_T(j) < \mu_Q/2$
- ▶ Having assessed the μ_Q systematics, “tuning” μ_Q is possible, although I dislike it (I dislike the tuning of any NLO-matching parameters, which I see as a sort of contradiction in terms)
- ▶ Some heuristic evidence that, contrary to the tree-level case, y -ordering in PY8 does not do much to FxFx-merged results

Helpful recent additions to NLO merging

▶ Reweight

<https://cp3.irmp.ucl.ac.be/projects/madgraph/wiki/Reweight>

→ Generate with one model, plot cross sections relevant to another

▶ Bias

<https://cp3.irmp.ucl.ac.be/projects/madgraph/wiki/LOEventGenerationB>

→ Throw more points in corners (eg tails) of phase space

▶ LO MEs

Undocumented, trivial to use (more later)

Reweight and bias

The use of both of these features can be exemplified by considering H +jets production with mass effects in the loops (which we have studied in [arXiv:1604.03017](#) with a more primitive version of the code)

- ▶ Reweight EFT matrix elements with full-SM ones
- ▶ Bias towards high- p_T tails to increase statistics for boosted searches

Note: this is done on FxFx-ed samples, so it works in a non-trivial technical context

► Generate with EFT:

```
import model HC_NLO_X0_UFO-heft
generate p p > x0 /t [QCD] @0
add process p p > x0 j /t [QCD] @1
add process p p > x0 j j /t [QCD] @2
```

► In `run_card.dat`, set:

```
bias = event_norm
```

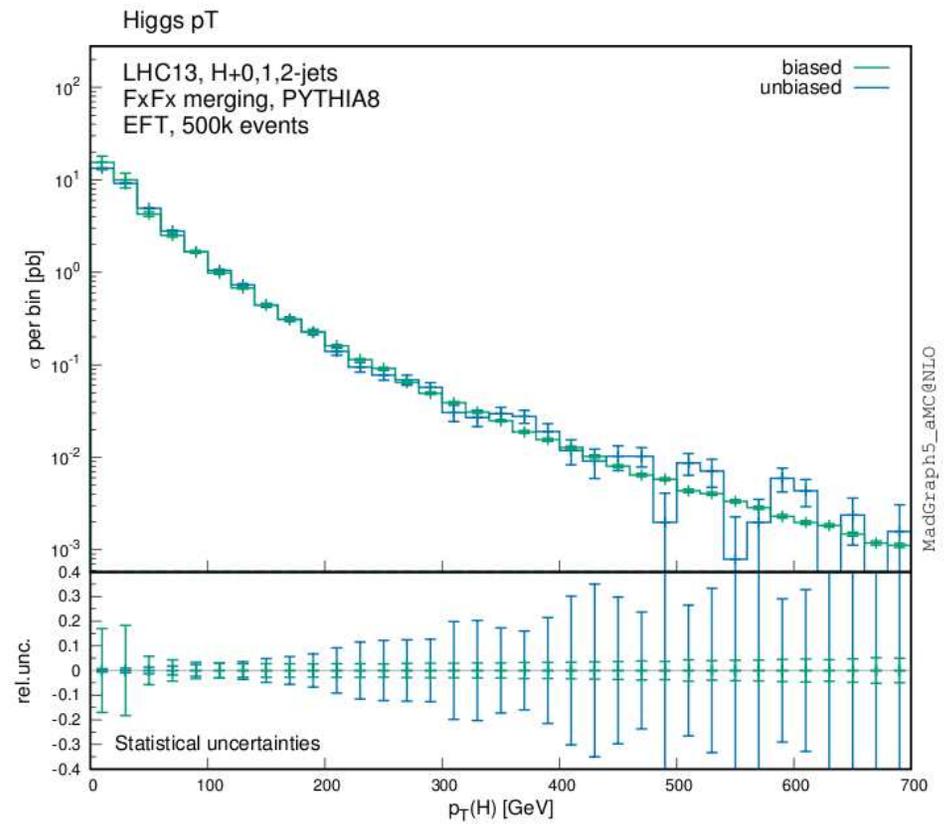
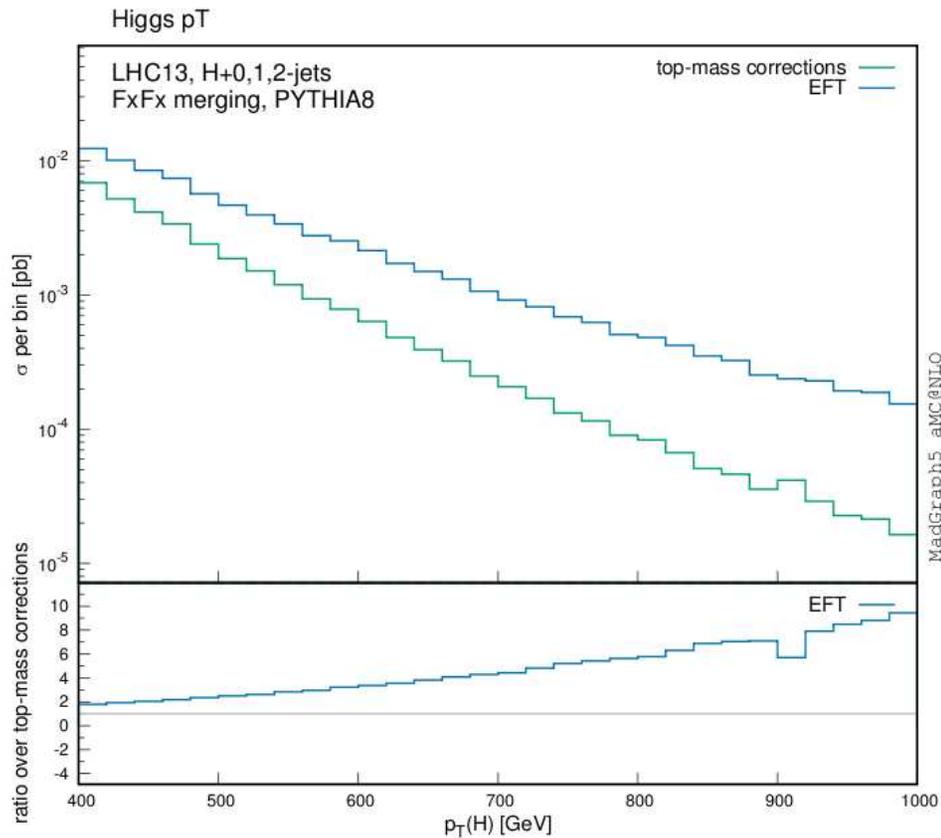
► In `reweight_card.dat`, set (eg):

```
change mode NLO_tree
change model loop_sm-no_b_mass
change process p p > h [sqrvirt=QCD]
change process p p > h j [sqrvirt=QCD] --add
change process p p > h j j QED=1 [sqrvirt=QCD] --add
change process p p > h j j j QED=1 [sqrvirt=QCD] --add
```

► In `cuts.f`, find subroutine `bias_weight_function`, and (eg):

```
do i=1,nexternal
  if (ipdg(i).eq.25) then
    bias_wgt=sqrt(p(1,i)**2+p(2,i)**2)**3
  endif
  if (bias_wgt.le.1000d0) bias_wgt=1000d0
enddo
```

whence (results by Eleni Vryonidou) 



Left: effect of reweighting
Right: effect of the bias

LO MEs

The idea is that of using only tree-level matrix elements (as opposed to the full NLO machinery) for the highest multiplicities

For this to be an improvement over pure NLO merging, it needs at least *two extra* tree-level multiplicities on top of NLO ones

W^+ example (irrelevant for physics, which can be done at NLO):

```
generate p p > w+ [QCD] @0
add process p p > w+ j [LOonly=QCD] @1
add process p p > w+ j j [LOonly=QCD] @2
```

Concluding remarks

From a recent discussion with ATLAS:

Q: Is phase space biasing also supported for merged setups, e.g. to generate otherwise unweighted events with flat statistics in a given observable like p_{TV} or HT?

A: NLO: no restrictions. LO: not possible when merged samples are constructed with a single file. It should be possible by starting from individual (fixed multiplicity) samples

Q: Are the NLO EWK corrections going to be available for ME+PS setups soon? Also, do you foresee support for NLO EWK corrections in merged samples in the near future?

A: It depends on the help we'll get from the MC authors. The main problem stems from the tagging of EW-interacting light particles. Merged sample should follow from their QCD counterparts

Concluding remarks

From a recent discussion with ATLAS:

Q: We were unable to generate complicated processes like e.g. $VV+2j@NLO$, since the initialisation/integration step requires 100s of GBs of RAM. Is there any way to circumvent this?

A: Have you tried: `set low_mem_multicore_nlo_generation?`

Concluding remarks

From Marek's message:

... we are aiming at creating benchmark cross sections and, possibly, event files, that can be shared between the experiments and do not rely on experiment specific settings/tunes. With this, we are ... trying to create a test bed towards *experiment independent event generation* which can then easily be employed by the individual experiments ... (emphasis mine)

My opinion: this is not a good idea

- ▶ Experiments must not outsource (*this, or any other core business*)
- ▶ Experiments must cross check each other
- ▶ High risk of overlooking and bypass hands-on expertise within experiments
- ▶ As far as I know, experiment-independent settings are a myth
- ▶ The level of scrutiny (*of policy, of results*) is not comparable to that within experiments