aMC@NLO tutorial

LPCC MC Workshop

CERN, 19/11/2012

Developers of core code

- Johan Alwall
- Rikkert Frederix
- Stefano Frixione
- Valentin Hirschi
- Fabio Maltoni
- Olivier Mattelaer
- Roberto Pittau
- Paolo Torrielli
- Marco Zaro

Utilities

MadSpin Pierre Artoisenet, Rikkert Frederix, Robbert Rietkerk

MadAnalysis Eric Conte, Benjamin Fuks, Guillaume Serret

What is it

A single framework for the computation of:

A. Hard events at the NLO or LO, to be subsequently showered

B. Infrared-safe observables at the NLO or LO

As in MadGraph (whose platform is used by aMC@NLO) there is no pre-defined list of processes: all is generated/computed on the fly

The code is modular: the use of third-party results for virtual matrix elements is straightforward (but not mandatory)

A.

- At the NLO, the matching formalism is the same as in MC@NLO. Currently HW6, HW++, and PY6 (Q^2) are supported.
  PY8 is under validation
- At the LO, it is the same as MadEvent (however, ME only interfaced to PY)
- In both cases, as is customary, MCs must be seen as plugins. aMC@NLO contains (lots of) utilities to shower hard events right after their generation, but this is not mandatory
- Hard event files are fully LHA compliant

B.

- This is the analogue of MCFM (no event generation, only histograms of IR-safe observables)
- Plots have to be produced on-the-fly (may change if needed)

Comparisons with MG4-based (spring 2011, not public)

- *Much* more user friendly
- All MadLoop (virtual-diagram module) limitations lifted
- MadLoop a factor of 50 faster
- Smaller number of integration channels

Requirements and dependences

- Python 2.6 or later (but earlier than 3.0)
- gcc/gfortran 4.6 or later
- Fastjet 3.0 or later
- LHAPDF is optional, but the in-house PDF library is far from being complete and up-to-date

aMC@NLO is now embedded in MadGraph5 2.0β, and works indeed in a similar manner as MG5 (eg, does not need to be pre-installed).
It can be downloaded at:

`http://amcatnlo.cern.ch`

The simplest run – setup

```
your_shell> ./bin/mg5
```

This creates the file `./input/mg5_configuration.txt`, which contains basic configuration instructions. At this point it can be edited, or accessed directly from the python shell. For example:

```
MG5> set fastjet /<PATH_TO_FASTJET>/bin/fastjet-config
```

The simplest run – setup

```
your_shell> ./bin/mg5
```

This creates the file `./input/mg5_configuration.txt`, which contains basic configuration instructions. At this point it can be edited, or accessed directly from the python shell. For example:

```
MG5> set fastjet /<PATH_TO_FASTJET>/bin/fastjet-config
```

```
MG5> install MCatNLO-utilities
```

This downloads a tarball that contains utilities for the showering of hard events, including drivers for HW6 and PY6, the sources of HW6 and PY6, and StdHEP (which is also compiled)

These operations need to be done only once (although they can be repeated if need be, for example to set different paths)

The simplest run

```
your_shell> ./bin/mg5

MG5> generate p p > t t~ W+ [QCD]

aMC@NLO> output MYDIR

aMC@NLO> launch
```

After a while, the hard-event file, and an StdHEP or HepMC file that contains the complete event record *after shower*, will be found in:

`./MYDIR/Events/run_01/`

Step by step

```
your_shell> ./bin/mg5
```

Loads (or creates if not already there) `./input/mg5_configuration.txt`, loads the *default model*, defines multiparticle tags – this is identical to MG5

Step by step

```
your_shell> ./bin/mg5
```

Loads (or creates if not already there) `./input/mg5_configuration.txt`, loads the *default model*, defines multiparticle tags – this is identical to MG5

At this point, one can modify the default model, according to the rules written in the files `models/loop_sm/restrict_XXX.dat`:

```
MG5> import model loop_sm-XXX
```

Step by step

```
your_shell> ./bin/mg5
```

Loads (or creates if not already there) `./input/mg5_configuration.txt`, loads the *default model*, defines multiparticle tags – this is identical to MG5

At this point, one can modify the default model, according to the rules written in the files `models/loop_sm/restrict_XXX.dat`:

```
MG5> import model loop_sm-XXX
```

For example, in the default model the b quark is massive. In order to set its mass to zero:

```
MG5> import model loop_sm-no_b_mass
```

This must be typically accompanied by multiparticle-tag definitions, e.g.:

```
MG5> define p = g u u~ c c~ d d~ s s~ b b~
```

Step by step

```
MG5> generate p p > t t~ W+ [QCD]
```

Generates all relevant matrix elements (Born, virtual, real-emission), and determines the singularity structure of the latter

Step by step

```
MG5> generate p p > t t~ W+ [QCD]
```

Generates all relevant matrix elements (Born, virtual, real-emission), and determines the singularity structure of the latter

The tag [QCD] stands for "NLO QCD corrections".

One may use [real=QCD] or [loop=QCD] if one wants to exclude virtual diagrams (or to use external virtuals), or to only include them. This is unphysical, and should be used only for tests

- Same syntax as in MG5, but not all options are allowed

Note: nothing is yet written on disk

Step by step

```
aMC@NLO> output MYDIR
```

This creates the directory `./MYDIR`, a copy of `./Template/NLO` which contains information common to all NLO or MC@NLO cross sections, supplemented by process-specific quantities (such as matrix elements, in `./MYDIR/SubProcesses/P0_*`)

Step by step

```
aMC@NLO> output MYDIR
```

This creates the directory `./MYDIR`, a copy of `./Template/NLO` which contains information common to all NLO or MC@NLO cross sections, supplemented by process-specific quantities (such as matrix elements, in `./MYDIR/SubProcesses/P0_*`)

At this point, one *can* edit process-specific files, eg:

- `./MYDIR/SubProcesses/cuts.f` to define generation cuts
- `./MYDIR/SubProcesses/setscales.f` to define dynamic scales
- `./MYDIR/Cards/run_card.dat, param_card.dat` for inputs

  (MC type for MC@NLO, cm energy, PDFs, fixed scales,...)

The input files can be edited directly from the python shell

Note that, if the command

```
MG5> install MCatNLO-utilities
```

had been issued any time prior to:

```
aMC@NLO> output MYDIR
```

then a directory will exist

`./MYDIR/MCatNLO`

which is a copy of `./MCatNLO-utilities/MCatNLO`, and whose files can be edited to tailor the shower-phase runs. To this end, an input file is also available

`./MYDIR/Cards/shower_card.dat`

This can be either edited directly, or accessed through the python shell

Step by step

```
aMC@NLO> launch
```

This corresponds to a complete MC@NLO run – generation of hard events and their subsequent showering. In particular, the code:

Step by step

`aMC@NLO> launch`

This corresponds to a complete MC@NLO run – generation of hard events and their subsequent showering. In particular, the code:

- Performs sanity checks: soft/collinear limits, cancellation of IR poles
- Integrates the cross sections
- Produces unweighted hard events, and stores them in `./MYDIR/Events/run_01/events.lhe.gz`
- Showers these events, and stores the results in `./MYDIR/Events/run_01/`

Some other commands allow one to have a fuller control of the code $\longrightarrow$

After having executed `aMC@NLO> output MYDIR`, one can quit the shell, edit files, and so forth. Upon re-entering the python shell, execute:

```
MG5>launch MYDIR -i
```

After having executed `aMC@NLO> output MYDIR`, one can quit the shell, edit files, and so forth. Upon re-entering the python shell, execute:

```
MG5>launch MYDIR -i
```

There are then the following options (and several more)

- `aMC@NLO_run> launch aMC@NLO`

  MC@NLO run: NLO hard events subsequently showered

After having executed `aMC@NLO> output MYDIR`, one can quit the shell, edit files, and so forth. Upon re-entering the python shell, execute:

```
MG5>launch MYDIR -i
```

There are then the following options (and several more)

- `aMC@NLO_run> launch aMC@NLO`

  MC@NLO run: NLO hard events subsequently showered

- `aMC@NLO_run> launch NLO` or

  `aMC@NLO_run> calculate_xsect NLO`

  NLO *not* matched to shower

After having executed `aMC@NLO> output MYDIR`, one can quit the shell, edit files, and so forth. Upon re-entering the python shell, execute:

```
MG5>launch MYDIR -i
```

There are then the following options (and several more)

- `aMC@NLO_run> launch aMC@NLO`

  MC@NLO run: NLO hard events subsequently showered

- `aMC@NLO_run> launch NLO` or

  `aMC@NLO_run> calculate_xsect NLO`

  NLO *not* matched to shower

- `aMC@NLO_run> launch aMC@LO`

  Same as MG5 run: LO hard events subsequently showered

After having executed `aMC@NLO> output MYDIR`, one can quit the shell, edit files, and so forth. Upon re-entering the python shell, execute:

```
MG5>launch MYDIR -i
```

There are then the following options (and several more)

- `aMC@NLO_run> launch aMC@NLO`

  MC@NLO run: NLO hard events subsequently showered

- `aMC@NLO_run> launch NLO` or

  `aMC@NLO_run> calculate_xsect NLO`

  NLO *not* matched to shower

- `aMC@NLO_run> launch aMC@LO`

  Same as MG5 run: LO hard events subsequently showered

- `aMC@NLO_run> launch LO` or

  `aMC@NLO_run> calculate_xsect LO`

  LO *not* matched to shower

Every time one performs a run which involves the integration of short-distance cross sections (i.e., generation of events or plotting of observables at fixed order) a directory is created

`./MYDIR/Events/run_nn/`

with `nn` an integer number increased by one unit at each run. With the commands listed in the previous slide, one would get:

- `nn=01` $\longrightarrow$ MC@NLO
- `nn=02` $\longrightarrow$ NLO
- `nn=03` $\longrightarrow$ MC@LO
- `nn=04` $\longrightarrow$ LO

MC runs

The command

```
aMC@NLO_run> launch aMC@NLO
```

is equivalent to:

```
aMC@NLO_run> generate_events NLO -p
aMC@NLO_run> shower run_nn
```

These two *need not* be both executed. One can simply use the hard-event file generated by the first command

```
./MYDIR/Events/run_nn/events.lhe.gz
```

and shower it outside the aMC@NLO framework

aMC@NLO_run> shower run_nn

Lots of controls are available just through the input file:

./MYDIR/Cards/shower_card.dat

For example, if one sets in that file

```
ANALYSE = myanalysis.o mydep1.o mydep2.o
EXTRALIBS = libs_needed_in_analysis
EXTRAPATHS = paths_to_those_libs
INCLUDEPATHS = paths_to_include_dirs
```

the shower phase will produce what is defined in the user-written file `myanalysis` (in place of the default StdHEP or HepMC file that collects the complete records of events), and store it in:

./MYDIR/MCatNLO/RUN_MCNAME_m/

and/or in (in the case of topdrawer files)

./MYDIR/Events/run_nn/

Fixed-order runs

The command

```
aMC@NLO_run> calculate_xsect (N)LO
```

will produce a topdrawer file:

`./MYDIR/Events/run_nn/MADatNLO.top`

whose content is defined by the (analysis) file

`./MYDIR/SubProcesses/madfks_plot.f`

For those using MCFM, this is the analogue of `nplotter.f` there

At present, modifications to this structure entail the editing of

`./MYDIR/SubProcesses/makefile_fks_dir`

This will be made as flexible as the analysis of the shower phase in the next couple of weeks

Scale and PDF uncertainties

These are done by reweighting when showers are included (MC@NLO or MC@LO), hence are essentially "free". In

./MYDIR/Cards/run_card.dat

one sets

```
.true.  = reweight_scale !  reweight to get scale dependence

.true.  = reweight_PDF ! reweight to get PDF uncertainty
```

for the relevant weights to be computed during the cross section integration, and stored in the hard-event file

./MYDIR/Events/run_nn/events.lhe.gz

In there, the bottom part of each event reads as follows $\longrightarrow$

```
<rwgt>
0.91087468D+04 0.91087478D+04 3 20
0.91087478D+04 0.94243129D+04 0.85241995D+04
0.90888221D+04 0.96416345D+04 0.82489527D+04
0.91111900D+04 0.92438738D+04 0.87238113D+04
0.91087898D+04 0.91089766D+04
...
0.90951793D+04 0.91073530D+04
</rwgt>
</event>
```

These numbers are matrix elements computed for all scale and PDF combination selected in input. Ask us for more details (the format will soon change to be made compatible with what is used elsewhere in MG5)

Concluding remarks

An "Help and FAQs" link can be visited starting from

```
http://amcatnlo.cern.ch
```

There is an online help from the python shell:

```
MG5>help
```

which lists all the available commands.

From a given environment, the relevant commands are also explained, eg:

```
MG5>launch MYDIR -i
aMC@NLO_run> help generate_events
```

Finally, the <TAB> key works from the python shell: it suggests command completion(s)