1 Installation

The NLO part in FeynRules requires version 2.1 which can be downloaded from the FeynRules web page `https://feynrules.irmp.ucl.ac.be/wiki/WikiStart`.

In order to compute NLO counterterms, FeynRules relies on the FeynArts package version 3.7 or 3.8, a Mathematica-based Feynman diagram generator which can be freely downloaded from `http://www.feynarts.de/`.

To get the plots at NLO, you should also download MadGraph5_aMC@NLO_v2.0.0 (release yesterday!) from `https://launchpad.net/mg5amcnlo`.

2 NLO precision for BSM phenomenology with FeynRules

In this section we describe how to use FeynRules to generate events at NLO accuracy with MadGraph5_aMC@NLO. Note that this feature only applies to renormalisable models.

In order to promote a model to NLO, we need to supplement the UFO produced by the UFO interface by twofold information

1. One-loop UV counterterms for all the parameters and fields that appear inside the Lagrangian.
2. A special class of tree-level vertices, called R_2, required to construct the correct rational terms in the one-loop amplitude.

A new FeynRules module, called NloCT, can be used to perform this task.

First, we need to renormalise the Lagrangian. This is achieved by the replacements

$$\begin{aligned}\phi_B &= (1+\frac{1}{2}\delta Z_\phi)\,\phi_R\,,\\ g_B &= g_R+\delta g\,,\end{aligned} \tag{1}$$

where ϕ and g represent a generic field and parameter, and the subscript B (R) refers to the bare (renormalised) quantities. These shifts can be automatically performed by FeynRules at the level of the Lagrangian via the command

```
Lren = OnShellRenormalization[ LSM + LNew, QCDOnly -> True , FlavorMixing -> False];
```

The second and third arguments are optional. The first instructs FeynRules that only parameters and field related to the strong interaction should be renormalised (which is sufficient as long as we only compute QCD corrections). The latter avoid the introduction of field renormalization constants mixing different fields as QCD corrections are flavour diagonal. Moreover, the previous command also takes of defining correctly all the counterterms for internal parameters in terms of the counterterms for external parameters. For example, the strong coupling constant α_s and the coupling $g_s=\sqrt{4\pi\alpha_s}$ are not independent, and the counterterms are related by

$$\delta g_s=\frac{\partial g_s}{\partial\alpha_s}\,\delta Z_{\alpha_s}=\sqrt{\frac{\pi}{\alpha_s}}\,\delta\alpha_s\,. \tag{2}$$

The counterterms for the external parameters on the other hand require the computation of loop diagrams, a task which cannot be directly performed by FeynRules. It is however possible to use the FeynArts package to automate this task. In particular, we can use the FeynArts interface to obtain an implementation of our model into FeynArts

```
SetDirectory["<Models directory of my FeynArts directory>"]
WriteFeynArtsOutput[Lren, Output -> "Tutorial", GenericFile -> False,
 FlavorExpand -> True]
```

For details on the options set in the interface we refer to the FeynRules manual. The interface will produce a set of files that can be use by FeynArts to generate (loop) diagrams for this model.

In order to compute the counterterms, we need to use FeynArts. Note that FeynRules and FeynArts cannot be loaded simultaneously into the same Mathematica kernel, and it is mandatory to quit the Mathematica kernel before proceeding. Next we need to load FeynArts into Mathematica,

```
SetDirectory["<your FeynArts Directory>"]
<<FeynArts`
```

In addition, we need to load the NloCT package, which is shipped together with FeynRules,

```
SetDirectory[ "<your FeynRules Directory>" ];
<<NLOCT`
```

Next, we can simply compute all the counterterms via the following simple call to NloCT,

```
SetDirectory["<your FeynRules Directory>/Models/Tutorial"];
WriteCT["Tutorial", "Lorentz", Output-> "Tutorial",
     ZeroMom -> {{aS, {F[7], V[4], -F[7]}}},
     QCDOnly -> True, Exclude4ScalarsCT -> True,
     Assumptions -> {MB >= 0, MT != 0, Muv != 0}]
```

The two first arguments in the first line refer to the name of the input file (the output of the `WriteFAOutput[ ]` function in the previous step), the generic model file to be used within FeynArts (we refer to the FeynRules and FeynArts manuals for this technical point). The name of the output file can be given by the option `Output` . By default, the FeynRulesmodel file is used. Just like before, the option `QCDOnly -> True` instructs the code that only QCD corrections should be computed. The option `Assumptions -> ...` instructs the code that certain constraints on certain variables should be taken into account when simplifying expression (e.g., when simplifying expressions involving logarithms). The option `Exclude4ScalarsCT -> True` avoid the computation of the diagrams with four external scalars as none of them are coming from QCD corrections. Finally, the renormalisation is by default done in the $\overline{MS}$ scheme for the couplings. For the strong coupling constant it is however customary to choose the the zero-momentum scheme, which imposes that the renormalized strong interaction with light quarks is equal to its tree-level value when the gluon momentum goes to zero. Running this command produced a file `Tutorial.nlo` which contains all the UV counterterms (and also the R_2 needed to compute NLO QCD corrections in this model.

The results obtained in the previous steps can be included into the UFO for the model by using the UFO interface of FeynRules. In order to do so, quit the kernel, and reload FeynRules and the model file. In addition, you need to load the `.nlo` file produced in the previous step,

```
SetDirectory["<your FeynRules Directory>/Models/Tutorial"];
Get["Tutorial.nlo"];
```

Calling the UFO with the options

```
WriteUFO[LNew + LSM, UVCounterterms -> UV$vertlist,  R2Vertices -> R2$vertlist,
    Output -> "Tutorial_NLO"]
```

produces all the files required by MadGraph5_aMC@NLO to produce events at NLO accuracy.

3 NLO cross-section computation and event generation with MadGraph5_aMC@NLO

In this section we will familiarise with the NLO output of MadGraph5_aMC@NLO, in particular with the different operations that can be performed with it.

We first need to generate our process at NLO, wchich can be done using the syntax

```
import model MODELNAME
generate PROCESS [QCD]
output myproc_folder
```

where `PROCESS` is the process one is interested in, using the usual syntax (no decay chains are allowed at NLO), and `[QCD]` means "do NLO in QCD". After these commands one should have a folder called `myproc_folder` with the relevant code inside. For the scope of this tutorial, we need to generate two processes:

```
import model Tutorial_NLO
generate p p > uv uv~ [QCD]
```

for which we will compute the NLO K-factor in Sec. 3.1. The second process is top pair production, which will be used to explore the NLO capabilities of MadGraph5_aMC@NLO

```
import model sm
generate p p > t t~ [QCD]
```

3.1 Fixed order runs

In this part of the tutorial we will compute the (LO and) NLO cross-section for `p p > uv uv~`.
After having exported the process inside `myproc_folder` with the commands

```
import model Tutorial_NLO
generate p p > uv uv~ [QCD]
output myproc_folder
```

we can start a computation by typing

```
> cd myproc_folder
>./bin/aMCatNLO
myproc_folder> launch
```

This will start the talk-to phase, where we can choose the run mode and edit the relevant cards.
If we are just interested in the computation of the total cross-section, we can avoid the generation of events, and stick to a simpler, fixed-order run. In order to do this, one can just type

```
fixed_order=ON
```

In order to compute the K-factor, we will need two subsequent runs, one with

```
order=NLO
```

(default choice), and one with

```
order=LO
```

Once we have set the run mode, we can move on to the card editing phase simply by hitting return. Once we are done, we can start the run simply by hitting return again. The run will now start: the code is compiled, then some sainity checks (poles of virtual matrix-elements, soft and collinear behaviour of the counterterms) are run, finally the cross-section is computed.
If scales and/or PDF uncertainties are asked for in the `run_card.dat` (the default setting is to compute only scale uncertainties, as for the PDF ones one needs to link LHAPDF), they are printed at the end of the run.

Using the results of the two runs, compute the K-factors and comment on the uncertainties. Results and run summary can be found inside the `Events/run_*` directories.

3.2 Event generation

After having familiarised with the fixed-order runs, we will learn how to generate a set of unweighted (up to a sign) events, to be passed to a parton shower. Please note that *NLO event samples are not physical unless they are showered with the parton-shower they have been generated for* which means that one cannot use these events to obtain parton-level results (unlike at LO), and has to generate a different sample for each parton-shower one wants to interface to. Because of some limitations in the interface to parton-showers, we have to stick to a SM process (`p p > t t~`). It can be generated with

```
generate p p > t t~ [QCD]
output myproc_folder_2
```

Again, to start the computation, we type

```
>./bin/aMCatNLO
myproc_folder> launch
```

but as this time we are not interested in fixed-order runs, we will set

```
fixed_order=OFF
```

We can either choose to run the shower right after the event file is created, or in another moment. This can be done setting

```
shower=ON / shower=OFF
```

In this case, for the sake of simplicity we will let the parton shower decay the top quarks, so we will keep

```
madspin=OFF
```

In the default running mode, the shower will produce a `.hep` (if the shower is PYTHIA6 or HERWIG6) or `.hepmc` file (if PYTHIA8 or HERWIG++ are used). For PYTHIA6 and HERWIG6 plot are created automatically. In both cases MADANALYSIS 5can be used to analyse these files. The desired shower can be set in the `run_card.dat`. Please note that in order to have PYTHIA8 or HERWIG++ working, one need to have them installed on his machine, and to set some relevant paths inside the `Cards/amcatnlo_configuration.txt` file. If the selected shower is PYTHIA6 or HERWIG6, nothing has to be done, as the source files are shipped with MADGRAPH5_AMC@NLO. Again, the talk-to phase (run mode choice and card editing) can be concluded by hitting return.

Each parton-level `.lhe` file can be found in a different `Events/run_*` directory, together with all the HEP(MC) files obtained by showering it. A `.lhe` file can be reshowered as many times as one wants, simply with the command (assuming one wants to shower the event file in the `run_XX` directory)

```
./bin/shower run_XX
```

From the showered files, obtain the LO and NLO distributions for the p_T of the top quark, and for the (cosine of the) azimuthal separation between the leptons coming from the top quark decay:

$$\cos\phi_{ll'} = \frac{\vec{p}_T(l)\cdot\vec{p}_T(l')}{|\vec{p}_T(l)||\vec{p}_T(l')|} \tag{3}$$

3.3 Spin correlated decay

The last step of this section allows us to improve the predictions obtained at the end of the previous section by including consistently the effects of spin-correlations in the decay of heavy particles (top quarks in our case). We remind the definition of spin correlations: consider the process

$$x + y \to u_1 + \dots u_p + s_1 + \dots s_q + X\,, \tag{4}$$

i.e. the inclusive production of p unstable particles u_i and q stable particles s_j. Suppose we are interested in some decay mode of the unstable particles

$$u_i \to d_{1,1} + \dots d_{1,n_1}\,. \tag{5}$$

The process is said to have *decay* spin-correlations if its amplitude depends in a non trivial manner on any $d_{k,i}\cdot d_{k,j}$ scalar product. It is said to have *production* spin-correlations if the amplitude depends on any $d_{k,i}\cdot d_{k',j}$, $d_{k,i}\cdot s_l$, $d_{k,i}\cdot x$ or $d_{k,i}\cdot y$ scalar product.

When the decay of heavy particles is performed by the parton shower, all production spin-correlation are lost, because the squared matrix-element is factorised into the product of the squared production matrix element times the squared matrix elements for the decay of unstable particles, missing all quantum interferences. Within MADGRAPH5_AMC@NLO, a package for the consistent inclusion of all spin correlations is included. This package is MADSPIN(details can be found in `arXiv:1212.3460`).

As for the shower phase, MADSPINcan be run together with the event-generation or on a pre-generated event file (as many times as one wants). In the first case one has to set

```
madspin=ON
```

syntax	example	meaning
`x, x>`	`p p > z j, z > b b~`	*s.1*
`$ x`	`p p > e+ e- $ z`	*s.2*
`/ x`	`p p > e+ e- / z`	*s.3*
`> x >`	`p p > z > e+ e-`	*s.4*
`$$ x`	`p p > e+ e- $$ z`	*s.5*

Table 1: Process-generation syntax refinements, also exemplified in the case of various processes that involve a Z boson. See the text for the explanation of the keywords *s.1*–*s.5*.

in the talk-to phase at the beginning of the run; in the second case one can decay an existing event file (say, the one in `run_XX`) with the command

```
./bin/aMCatNLO
> decay_events run_XX
```

In both cases, the decay channel one wants has to be specified inside the `madspin_card.dat`. For example, the dileptonic decay of the $t\bar{t}$ pair can be specified by writing

```
decay t > w+ b, w+ > e+ ve
decay t~ > w- b~, w- > mu- vm~
```

in the card. Multiparticle labels can be also used in the decay chains.
Note that currently MadSpinis limited to only handle decay chains which can be written in term of sequence of $1 \to 2$ decays.

The `.lhe` file with the decay can be found inside the `run_XX_decayed_YY` folder, where `XX` is the same name as the undecayed file, and `YY` is a progressive integer.

From the showered decayed file, obtain the distributions for the p_T of the top quark, and for the (cosine of the) azimuthal separation between the leptons coming from the top quark decay, and compare them with those obtained at the end of sec. 3.2.

4 Appendix: Generation Syntax

In the context of a LO-type generation, however, one can further refine the syntax above in order to include in the computation only some of the contributions that one would normally obtain. Such refinements are reported in table 1, and have the following meaning:

s.1 A production process is generated that features `x` in the final state, with `x` subsequently decaying into the list of particles that follow the "`x >`" string; more in general, there may be p primary particles that play the same role as `x`. Only p-resonant diagrams (see sect.**undefined**are included in the computation. In the example of table 1, one has the associated production of a Z and a jet, with the Z further decayed into a $b\bar{b}$ pair. Spin correlations and `x` off-shell effects are taken into account exactly, but the virtuality $m^\star_{\mathtt{x}}$ of `x` is forced to be in the following range:

$$|m^\star_{\mathtt{x}} - m_{\mathtt{x}}| \leq \mathtt{bwcutoff}\,\Gamma_{\mathtt{x}}\,, \tag{6}$$

where $m_{\mathtt{x}}$ is the pole mass of `x`, $\Gamma_{\mathtt{x}}$ its width, and `bwcutoff` is a parameter controlled by the user (through `run_card.dat`). Syntax *s.1* thus loosely imposes an on-shell condition; it is called **decay-chain syntax**, and can be iterated: any decay product can be decayed itself by using this syntax (e.g. `x > y z, y > w s`).

s.2 If `x` appears as an intermediate particle in the generated process, its virtuality is forced to be in the range:

$$|m_{\mathtt{x}}^{\star} - m_{\mathtt{x}}| > \mathtt{bwcutoff}\,\Gamma_{\mathtt{x}}\,, \tag{7}$$

which is the region complementary to that of eq. (6), and thus loosely imposes an off-shell condition. All diagrams are kept. In the example of table 1, one has Drell-Yan production with the invariant mass of the e^+e^- pair larger than or smaller than the Z mass by at least $\mathtt{bwcutoff}\,\Gamma_Z$. A consequence of the complementarity mentioned above is that, while cross sections generated with either *s.1* or *s.2* are `bwcutoff`-dependent, their sum is not (up to interference terms, which are neglected by the process of discarding non-resonant diagrams in *s.1*), and corresponds to the process generated with the simplest syntax. For example:

$$\frac{d\sigma}{dO}(\mathtt{p\ p\ >\ z}) \simeq \frac{d\sigma}{dO}(\mathtt{p\ p\ >\ z,\ z\ >\ e+\ e-}) + \frac{d\sigma}{dO}(\mathtt{p\ p\ >\ e+\ e-\ \$\ z})\,, \tag{8}$$

for any observable O.

s.3 All diagrams that feature (anywhere) the particle `x` are discarded.

s.4 The process is generated by demanding that at least one particle of type `x` be in an s-channel.

s.5 All diagrams that feature the particle `x` in an s-channel are discarded.

We stress that all syntaxes but *s.2* produce in general results which are non physical, because gauge invariance might be violated (although there are exceptions: see e.g. ref. [?]), and have therefore to be used with extreme caution.