Preprint typeset in JHEP style - HYPER VERSION

FeynRules/MadGraph_aMC@NLO/MadAnalysis5 tutorial

The FeynRules & MadGraph teams

Abstract: We present a simple example of how to make a simulation of LHC events for BSM signals and the corresponding backgrounds using a fully integrated chain of tools, including FeynRules, MadGraph5_aMC@NLO and MadAnalysis 5. A simplified model that features new heavy fermionic (triplet and singlet in color) and two neutral scalar states is first implemented in FeynRules. The output is passed to MadGraph5_aMC@NLO for process simulations, determination of the cross section and signature identification at the parton level and/or including parton shower (Pythia or Herwig). A few representative parameter benchmark points for the most promising signatures are identified. MadAnalysis 5 is used as a flexible framework to analyse events at different steps of the simulations. A search strategy is formulated based on the characteristics of the main backgrounds that are automatically simulated at LO and also at NLO through the most advanced techniques. Observables that are sensitive to the signal are identified and finally compared to two sets of pseudo LHC8 data.

Keywords: Monte Carlo simulations, LHC, Standard Model, Beyond the standard model.

Contents

---

1. Installation

1.1 Installation of FeynRules

FeynRules is a Mathematica package, and therefore requires a working Mathematica installation on the machine (version 9 or higher). The most recent version of the code (version 2) can be downloaded from `http://feynrules.irmp.ucl.ac.be/`. Note that while FeynRules itself is platform independent and can be run on any platform on which Mathematica is available, (some of) the interfaces require a Unix-based environment in order to run properly.

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

Note that the model file containing the template solution for the model to be implemented during the tutorials can be downloaded from `https://www.dropbox.com/s/vmgl52w3yflyjbf/Tutorial_Files.tar.gz?dl=0`.

1.2 Installation of MadGraph5_aMC@NLO

1.2.1 Installation on Windows

MadGraph5 and the associated programs are designed and tested on Linux and MacOs operating systems. The windows compatibility via `cygwin` is currently not supported. For Windows user, we advise they install Linux in dual boot. Virtual machines are another possibility. Note that some virtualbox packages do not include the library `readline`. This library is not mandatory but enables the auto-completion (See the paragraph associated to python installation to learn how to solve this). For this tutorial session, we have a virtual box ready to use on usb stick please ask If you need it.

1.2.2 Installation on Linux / MacOs

MadGraph5_aMC@NLO The last version of MadGraph5_aMC@NLO can be found at the following page: `https://launchpad.net/madgraph5`. This website is also the place, where you can ask question, make suggestions or report a bug. The installation is straightforward since you have only to untar it

```
tar -xzpvf MG5_aMC_v2.X.Y.tgz
```

No compilation are required for MadGraph5_aMC@NLO, you can just launch it.

```
./MG5_aMC_v2.X.Y/bin/mg5_aMC
```

If you don't have a valid python version, MadGraph5_aMC@NLO will crash directly with an explicit message. In this case, you will need to install Python2.7.

If you have admin rights on your system, you can run the following command:

```
sudo ln -s MG5_aMC_v2.X.Y/bin/mg5_aMC /usr/local/bin
```

such that MadGraph5_aMC@NLO can be launched from any directory.

Python The only requirement for MadGraph5 is to have a current version of Python (version 2.6 or 2.7). In most cases, it can be installed via your favorite repository manager. However, some of the linux repository distributes python 2.5. In that case, you will need to download python at the following link: `http://www.python.org/download/` and follows the instructions.

Note that for some linux versions (especially in virtual machine), the library `readline` is not present on the system. This will disable the auto-completion. If you care about that point, you will should first install that library (via your repository) and then to recompile python from the source code.

GCC The program requires gcc4.6 or newer. Which is available via your favourite repository manager. Previous version of gcc will not be able to compile any of the NLO code, but will work for LO processes.

Optional package Various optional packages can be linked to MadGraph5_aMC@NLO in order to customize the output, create plots, ... The installation of those packages is easy since you can install them by launching mg5 and typing

```
mg5> install NAME
```

Where NAME is one of the following package name:

- MadAnalysis: A package to draw automatically various histogram linked to the event generation.
- ExRootAnalysis : A package to convert the various output in a ROOT format.
- pythia-pgs: A package containing Pythia6 and PGS. Pythia6 is able to shower and to hadronize your events and is able to perform the matching for multi-jet production. PGS is a fast detector simultation package.
- Delphes: A package allowing to have a fast detector simulation, in replacement of PGS.
- SysCalc: A package to compute the systematic uncertainty.

Note that some of those programs might have some extra-dependencies (especially in Root).

Additional instructions for MacOs Compared to Linux the installation on MacOS might be more complex since MacOS doesn't provide various set of default programs present on Linux. Two important programs which are not present by default are gmake and gfortran4.x. We advise you to first check if those program are install via the commands.

```
    $> make --version
    $> gfortran --version
```

In order to install gmake it is easiest to install xcode (free but requiring an apple developer account)

- MacOs 10.5: https://connect.apple.com/cgi-bin/WebObjects/MemberSite.woa/wa/getSoftware?bundleID=20414
- MacOs 10.6: http://connect.apple.com/cgi-bin/WebObjects/MemberSite.woa/wa/getSoftware?bundleID=20792
- MacOs 10.7: http://itunes.apple.com/us/app/xcode/id448457090?mt=12
- MacOs 10.8 and newer: use the app store.

Note that you need to enable the command line tools (in the download section of the preference of Xcode)

Concerning gfortran you can download it (with gcc) either via "port"

```
sudo port install gcc47
```

or download a pre-compile version at one of the following address:

- MacOs 10.5: http://sourceforge.net/projects/hpc/files/hpc/gcc/gcc-leopard-intel-bin.tar.gz/download
- MacOs 10.6: http://prdownloads.sourceforge.net/hpc/gcc-snwleo-intel-bin.tar.gz?download
- MacOs 10.7: http://prdownloads.sourceforge.net/hpc/gcc-lion.tar.gz?download
- MacOs 10.8: http://prdownloads.sourceforge.net/hpc/gcc-mlion.tar.gz?download
- MacOs 10.9: http://prdownloads.sourceforge.net/hpc/gcc-4.8-bin.tar.gz?download
- MacOs 10.10: http://prdownloads.sourceforge.net/hpc/gfortran-4.9-bin.tar.gz?download

More information can be found here: http://hpc.sourceforge.net/

1.3 Testing the installation and Learning the syntax

MadGraph5_aMC@NLOincludes a build-in tutorial.

```
$> ./bin/mg5
mg5> tutorial
```

or

```
$> ./bin/mg5
mg5> tutorial aMCatNLO
```

if you want to discover the NLO functionalities.

Then just follow the instructions on the screen and you will learn the basic command/usage of MadGraph5_aMC@NLO. This takes around 15-20 minutes.

1.4 Installation of MadAnalysis 5

In order to install the latest stable version of the MadAnalysis 5 package, a tarball can be downloaded from the website
`https://launchpad.net/madanalysis5`
and be subsequently unpacked by issuing in a shell

```
mkdir MadAnalysis5
cd MadAnalysis 5
/tar -xvf ma5_xxxx.tgz
```

where `xxxx` stands for a version number. MadAnalysis 5 requires several external dependencies in order to properly run:

- Python 2.6 or a more recent version (but not the 3.X series) that can be downloaded from the website
  `http://www.python.org/`
  This requirement is common with those needed by the MadGraph 5 package.
- The `numpy` package must be properly installed to. It can be downloaded from
  `http://www.numpy.org/`
- The GNU GCC compiler. Note that MadAnalysis 5 has been validated with the versions 4.3.X and 4.4.X. The GCC compiler can be downloaded from
  `http://gcc.gnu.org/`
- ROOT v5.27 or a more recent version that can be downloaded from
  `http://root.cern.ch/`
  We remind that checking the version of ROOT installed on a system, it is sufficient to type in a shell
  `root-config --version`
  Moreover, the Python libraries generated by ROOT must be present. To install

them, it is necessary to first install the Linux package PYTHON-DEVEL on the system. Next, before generating the ROOT makefile, the ROOT configuration script has to be run as

```
./configure --enable-python
```

Let us note that the PYTHON version employed when starting MADANALYSIS 5 has to be the same as the one used for the compilation of ROOT.

To benefit from all options coming with the MADANALYSIS 5 program, we also recommend to install ZLIB headers and libraries. The latter can be downloaded from

```
http://zlib.net/
```

Once properly installed, MADANALYSIS 5 can be launched by issuing

```
bin/ma5
```

When started, MADANALYSIS 5 first checks that all the dependencies (GCC, PYTHON, ROOT, ZLIB) are present on the system and that compatibility is ensured with the installed versions. In the case of any problem, a message is printed to the screen and the code exists in the case it cannot properly run. On the first session of MADANALYSIS 5, the SAMPLEANALYZER core is compiled behind the scene as a static library stored in the directory `tools/SampleAnalyzer/Lib`. For the next sessions, the kernel is only recompiled if the configuration of the system changes (new version of the dependencies or of the main program).

2. The model

We add two real scalar fields, ϕ_1 and ϕ_2. They are singlets under all SM gauge groups. Their mass terms are[1]:

$$\mathcal{L}_{kin,scalar} = \frac{1}{2}\partial_\mu\phi_1\partial^\mu\phi_1 + \frac{1}{2}\partial_\mu\phi_2\partial^\mu\phi_2 - \frac{m_1^2}{2}\phi_1^2 - \frac{m_2^2}{2}\phi_2^2 - m_{12}^2\phi_1\phi_2\,. \tag{2.1}$$

We will call mass eigenstates Φ_1 and Φ_2, and their masses M_1 and M_2, respectively, and we will assume $M_1 < M_2$.

We add two Dirac fermion fields, U and E. Their SM quantum numbers are those of the SM u_R and e_R, respectively. These fields have mass terms

$$\mathcal{L}_{dirac,mass} = M_U\overline{U}U + M_E\overline{E}E \tag{2.2}$$

They interact with scalars via

$$\mathcal{L}_{FFS} = \lambda_{1,i}\,\phi_1\,\overline{U}P_Ru_i + \lambda_{2,i}\,\phi_2\,\overline{U}P_Ru_i + \lambda'_{1,i}\,\phi_1\,\overline{E}P_Rl_i + \lambda'_2\,\phi_2\,\overline{E}P_Rl_i + \text{h.c.}\,, \tag{2.3}$$

where u_i and l_i are the SM up-type quark and charged lepton fields. Note that there is a $\mathbb{Z}_2$ symmetry under which all fields we added $(\phi_{1,2}, U, E)$ flip sign, while all SM fields do not, so the new particles must be pair-produced and the lightest new particle (LNP) is stable. This same $\mathbb{Z}_2$ also forbids $U - u_i$ and $E - l_i$ mixing via Yukawas with the SM Higgs.

[1]All Lagrangian parameters, here and below, are assumed to be real

3. The FeynRules implementation

3.1 Preparation of the model file

As the model we are going to implement is a simple extension of the SM, it is not necessary to start from scratch, but we can use the implementation of the SM included in the folder `/Models/SM`. We therefore start by making a copy of this folder, in order to keep a clean version of the SM. To do so, change directory to the `Models` subdirectory and make a copy of the SM folder, before going into the new directory

```
cd Models
cp -r SM Tutorial
cd Tutorial
```

Even though the implementation is based on the model file `SM.fr` for the Standard Model, the SM sector of the model will be implemented into a separate file that will simply be loaded on top of `SM.fr`. We therefore start by opening a blank text file called `Tutorial.fr`[2]. You can start by personalizing the model file by including a name for you model, the name of the author, *etc.*,

```
M$ModelName = "Tutorial";

M$Information = {Authors      -> {"C. Duhr"},
                 Version      -> "1.0",
                 Date         -> "20. 11. 2015",
                 Institutions -> {"CERN"},
                 Emails       -> {"claude.duhr@cern.ch"}
             };
```

Note that his information is purely optional and could be omitted.

3.2 Implementation of the new parameters

We start by implementing the new parameters. The model we are considering depends on 9 new parameters, which we assume to be real in the following. FeynRules distinguishes between two types of parameters, the so-called *external* parameters given as numerical inputs and the *internal* parameters, related to other external and/or internal parameters via algebraic expressions. All the parameters of the model, both external and internal, are given in the FeynRules model file as a list named `M$Parameters`. Note that if an internal parameter x depends on some other parameter y, then y should appear in `M$Parameters` *before* the internal parameter x.

The new external parameters of the model are

[2]Tip: using a text editor with a Mathematica linter will make writing and identifying typos or bugs in a model file easier. On Linux, Mac OSX and Windows, the (free) ATOM editor (`https://atom.io/`) offers this syntax highlighting functionality in the inbuilt add-on manager via the 'language-mathematica' addon. This can be installed directly by searching in Edit > Preferences > Install, and enabled by selecting 'Mathematica' as the working language by clicking on the bottom right option in the window.

- 5 mass parameters: m_1, m_2, m_{12}, M_U, M_E.
- 4 vectors of coupling constants: $\lambda_{1,i}$, $\lambda_{2,i}$, $\lambda'_{1,i}$, $\lambda'_{2,i}$.

Note however that there is a difference between the mass parameters in the scalar and fermionic sectors: while the masses in fermionic sector are physical masses, the mass matrix in the scalar sector is not diagonal. For this reason, we will not discuss in the following the fermion masses M_U and M_E, as they will be defined together with the particles rather than as parameters of the model.

Let us now turn to the definition of the mass parameters in the scalar sector. The masses m_1, m_2 and m_{12} will be denoted in the FeynRules model file by `MM1`, `MM2` and `MM12`. In the following we only show how to implement `MM1`, all other cases being similar. `MM1` should be included as an entry of the form,

```
M$Parameters = {
    ...
    MM1 == {
           ParameterType -> External,
           Value         -> 200,
           Description   -> "Parameter in first entry on main diagonal of
                              mass matrix of unphysical pi scalars"
         },
    ...
}
```

The first option tags `MM1` as an external parameter, while the second option assign a value of 200GeV to m_1. We stress that this numerical value can be changed later on in the matrix element generators. The `Description` field is purely optional, but improves the readability of the model.

The masses in the scalar sector are not the physical masses, because the mass matrix is not diagonal. In order to obtain the physical masses, we need to diagonalize the mass matrix

$$\begin{pmatrix} m_1^2 & m_{12}^2 \\ m_{12}^2 & m_2^2 \end{pmatrix} . \tag{3.1}$$

In the following, we denote the eigenvalues by `MPe1` and `MPe2`. In addition, we need to introduce a mixing angle θ (`th`) relating the fields ϕ_i to the mass eigenstates Φ_i by,

$$\begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix} = \begin{pmatrix} -\sin\theta & \cos\theta \\ \cos\theta & \sin\theta \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \end{pmatrix} . \tag{3.2}$$

As in this case the mass matrix is only two-dimensional, we can compute the eigenvalues and the mixing angle analytically on paper or in a Mathematica notebook (using e.g. the Eigendecomposition function), and simply implement the analytical formulas into FeynRules. The implementation follows exactly the same lines as for the masses m_1, m_2, m_{12}, with the only differences that

1. the `ParameterType` is `Internal` (as these parameters are dependent on the external input parameters),
2. the `Value` is given by an analytical expression (in Mathematica syntax).

Next we turn to the implementation of the (vectors of) new coupling constants, which we will call `lam1`, `lam2`, `lam1p`, `lam2p`. They are all external parameters, and thus the implementation follows exactly the same lines as the implementation of the mass parameters, with two modifications: First, we have to deal with vectors in flavor space, *i.e.*, objects carrying a single flavor index. The fermion fields implemented in `SM.fr` carry an index called `Generation` which labels the flavor of a given field. We can in the same way assign an index of this type to parameters by including in the parameter definition:

```
Indices  ->  { Index[Generation] }
```

which will declare these couplings to have three components accessed by e.g. `lam1[i]`, $i = 1, 2, 3$. Consequently their numerical values are implemented as a list of rules for each value of `Index[Generation]`, such as:

```
Value -> {lam1[1]- > 0.1, lam1[2] -> 0.1, lam1[3] -> 0.1}
```

Secondly, some matrix element generators, for example MadGraph, keep track of the types of couplings that enter a process. This allows for example to generate a process by only taking into account QCD-type vertices, and to neglect all QED-type vertices. For this reason, it is mandatory to tell the matrix element generator how the new coupling constants should be counted. As in this case we are dealing with new classes of couplings which are *a priori* independent of QCD or QED interactions, we simply assign a new tag, called *interaction order*, to the coupling via the option

```
InteractionOrder ->  {NP, 1}
```

The name of the tag (`NP` for "new physics" in this case) can be chosen freely. The above option instructs the matrix element generator to count one unit of "`NP`" for each new coupling.

3.3 Implementation of the fields

In this section we discuss the implementation of the new fields. The implementation is similar to the implementation of the parameters, *i.e.*, all the fields should be defined as entries in a list called `M$ClassesDescription`. In Tab. 1 we show the names of the fields used in the implementation[3].

We illustrate the implementation of a new field on the example of the particle U (`uv`). The definition of the particle corresponds to an entry in `M$ClassesDescription` of the following form:

[3]Note that the symbol `u`, `e` and `phi` are already in use in the SM implementation. We also avoid using simply uppercase letters, as some matrix element generators are case insensitive.

U	E	ϕ_1	ϕ_2	Φ_1	Φ_2
uv	ev	pi1	pi2	p1	p2

Table 1: Symbols used for the fields in the FeynRules implementation.

```
M$ClassesDescription = {
      ...
      F[100] == {
          ClassName      -> uv,
          SelfConjugate  -> False,
          Indices        -> {Index[Colour]},
          QuantumNumbers -> {Y -> 2/3, Q -> 2/3},
          Mass           -> {Muv, 500},
          Width          -> {Wuv,1}
         },
        ...
 }
```

The meaning of this definition is as follows: each particle class has a name of the form `X[i]`, where `X` is related to the spin of the field (See Tab. 2), and `i` is an integer that labels the classes. Note that `i` can be chosen freely, as long as there is no name clash with an already existing class (in this case, there could be a name clash with the SM particles already defined in `SM.fr`). Therefore we've used `F[100]` to define the first of the new fermion species, and when defining subsequent fermions we should increment this label. Each class has a series of options:

1. `ClassName`: the symbol by which the particle will be represented in the Lagrangian.

2. `SelfConjugate`: a boolean variable, indicating whether the particle has an antiparticle (`False`) or not (`True`). If the field is not selfconjugate, a symbol for the antiparticle is automatically defined by appending "`bar`" to the name of the particle. In the above example the antiparticle associated to `uv` will be denoted by `uvbar`. Note that in the case of fermions the symbol for the antiparticle refers to the quantity $\bar{U}$ rather than $U^\dagger$.

3. `Indices`: All indices carried by the field. The available types of indices from the SM implementation are

   - `Generation`: fermion flavor index ranging from 1 to 3,
   - `Colour`: fundamental color index ranging from 1 to 3,
   - `Gluon`: adjoint color index ranging from 1 to 8,
   - `SU2W`: adjoint $SU(2)_L$ index ranging from 1 to 3.

4. `QuantumNumbers`: a list of all $U(1)$ charges carried by the field. In the SM implementation the following $U(1)$ charges are already defined
   - `Y`: weak hypercharge,
   - `Q`: electric charge.
5. `Mass`: the mass of the particle. It is a list of two elements, the first being the symbol used to represent the mass, and the second its value (in GeV). If the value of the mass is obtained from some analytic expression defined as an internal parameter with the same symbol (as is the case for example in the scalar sector of the model), the value is set to `Internal`.
6. `Width`: the width of the particle. The definition is similar to `Mass`. Note that as we do not yet know the widths of the new particles, we simply set it for now to 1GeV, and will determine its exact value later on.

The implementation of the other mass eigenstates (`ev`, `p1`, `p2`) is similar, so we do not discuss it here.

Spin	0	1/2	1	2	ghost
Symbol	`S`	`F`	`V`	`T`	`U`

Table 2: Available particle classes in FeynRules.

Let us comment on the implementation of the interaction eigenstates ϕ_i. Indeed, while the matrix element generators work exclusively at the level of the mass eigenstates, the interaction eigenstates are in general useful to write the Lagrangian in a compact form. It is therefore useful to define also the fields for the interaction eigenstates ϕ_i. The definition of these fields is similar to the mass eigenstates, *e.g.*,

```
S[100] == {
       ClassName     -> pi1,
       SelfConjugate -> True,
       Indices       -> {},
       Unphysical    -> True,
       Definitions   -> {pi1  ->  - Sin[th] p1 + Cos[th] p2}
      },
```

First, note that the `Mass` and `Width` options are omitted[4], as these fields are not mass eigenstates. This last fact is made explicit by the option

```
Unphysical    -> True,
```

which instruct FeynRules not to output this field to a matrix element generator. Finally, the expression relating the field `pi1` to the mass eigenstates `p1` and `p2` is simply given as a Mathematica replacement rule in the `Definitions` option.

[4]The `QuantumNumbers` option is also omitted, but for the simple reason that the fields ϕ_i do not carry any $U(1)$ charges.

3.4 Implementation of the Lagrangian

The definitions in the model file being complete, we now turn to the implementation of the Lagrangian. This can be done either in the model file, or directly in a Mathematica notebook. Here we use the latter approach, and we start by opening a new notebook and load the FeynRules package (see the preinstallation instructions). Next we have to load the model files, both for the SM and for the new sector,

```
LoadModel["SM.fr", "Tutorial.fr"]
```

Note that the new model file should be loaded after `SM.fr`. Furthermore, we also load two additional files, which restrict the first two fermion generations to be massless and the CKM matrix to be diagonal,

```
LoadRestriction["DiagonalCKM.rst", "Massless.rst"]
```

The new Lagrangian consists of three parts,

$$\mathcal{L} = \mathcal{L}_{scalar,kin} + \mathcal{L}_{fermion,kin} + \mathcal{L}_{Yuk}\,. \tag{3.3}$$

The kinetic terms for the new scalars can easily be implemented by using the symbols for the gauge eigenstates and the mass parameters defined in the model file, as well as the symbol for the space-time derivative ∂_μ in FeynRules, `del[ ..., mu]`. As an example, we have

$$\frac{1}{2}\partial_\mu\phi_1\partial^\mu\phi_1 - \frac{1}{2}m_1^2\phi_1^2$$

```
1/2 del[p1, mu] del[p1, mu] - 1/2 MM1^2 p1^2
```

The kinetic terms for the fermions can be implemented in a similar way. However, as the fermions are charged under the SM gauge group, we have to use the covariant derivative `DC` rather than the space-time derivative `del`. Furthermore, we have to use a "." instead of an ordinary multiplication in order to take the non-commuting nature of the fermions into account. As an example, we have

$$i\,\bar{U}\gamma^\mu D_\mu U - M_U\bar{U}U$$

```
I uvbar.Ga[mu].DC[uv, mu] - Muv uvbar.uv
```

where `Ga[mu]` is the FeynRules symbol for the Dirac matrix γ^μ. Finally, the Yukawa interactions can be implemented in the same way as the kinetic terms for the fermions, *e.g.*,

$$\lambda_{1,i}\,\phi_1\,\bar{U}P_+u_i$$

```
lam1[f] pi1 ProjP[s1,s2] uvbar[s1,i].uq[s2, f, i]
```

where `u` denotes the u quark field defined in `SM.fr` and `ProjP` denotes the right chiral projector (the left projector is denoted by `ProjM`). Note that FeynRules contains a function `HC[ ]` which allows to obtain the hermitian conjugate of an expression in an automated way. A hermitian Lagrangian `L` can then be recovered from a non-hermitian term `LnH` by defining `L := LnH + HC[LnH]`.

3.5 Computing the Feynman rules and decay rates and running the interfaces

Our model implementation is now complete, and so we can compute the Feynman rules. The Feynman rules of the new sector can be obtained by issuing the command

```
vertices = FeynmanRules[ LNew ];
```

where `LNew` is the name of the variable that contains the new Lagrangian, and the vertices are stored in the variable `vertices`.

After the vertices have been computed, it is possible to use FeynRules to compute the two-body decay widths of the new particles. In our case, the scalar particles cannot decay via two-body channels. The heavy fermion U and E, however, can have two-body decays via the Yukawa couplings in $\mathcal{L}_{FFS}$. The partial decay rates can be computed via

```
decays = ComputeWidths[ vertices ];
```

The variable `decays` now contains all the two-body partial widths that can be computed from the three-point vertices contained in `vertices`. Note that *all* possible decays for *all* possible initial states have been computed, independently whether a given channel is kinematically allowed. The reason is that, while a certain channel might be forbidden for some choices of the numerical values for the external parameters, the same channel might be allowed for other choices. The analytical formulas for the partial widths can be accessed through the function `ParticalWidth[ ]`, *e.g.*, the (analytic) partial width for the decay $U \to t\,\Phi_1$ can be accessed via

```
PartialWidth[ {uv, t, p1} ]
```

The function `TotWidth[ ]` works in a similar way. We can evaluate the total width numerically (Note that Φ_1 is stable),

```
NWuv = NumericalValue[ TotWidth[ uv ] ];
NWev = NumericalValue[ TotWidth[ uv ] ];
NWpe1 = NumericalValue[ TotWidth[ p1 ] ];
```

The numerical values used for the evaluation correspond to the numerical values for the external parameters entered into the model file. Next, we can update the numerical values of the widths in the model:

```
UpdateParameters[{Wuv -> NWuv, Wev -> NWev}]
```

Note that these values for the widths *only take two-body decays into account.* In some cases many-body decays can also be important. The contributions from these decays cannot be computed by FeynRules. We will see later how to use the matrix element generators to compute a more accurate value of the widths in cases where the contributions from many-body decays are important.

The Feynman rules can be written to file in a format suitable to various matrix element generators by using the FeynRules interfaces. In this tutorial, we will use the interface to the UFO format, and thus to MadGraph 5 (among others), which can be called via

```
WriteUFO[ LSM + LNew ];
```

where `LSM` is the SM Lagrangian implemented in `SM.fr`. Note that the SM implementation is available in both Feynman gauge and unitary gauge. A boolean variable `FeynmanGauge` allows to switch between both gauges.

3.6 Exporting the model into MadGraph 5

After successfully running the UFO interface, a directory `Tutorial_UFO` has been created in the `/Models/Tutorial/` directory. This directory contains all the UFO files needed to run the model in MadGraph 5. To import the model into MadGraph it is enough to copy the UFO directory into the `/models/` subdirectory of MadGraph 5,

```
cp -r  Tutorial_UFO <your MadGraph directory>/models/
```

The MadGraph 5 shell the new model can now be called in the same way as any other built in model,

```
mg5> import model Tutorial_UFO
```

More information on MadGraph 5 will be presented in its tutorial in the next section.

3.7 Advanced: 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

$$\begin{aligned} \phi_B &= (1 + \frac{1}{2}\delta Z_\phi)\, \phi_R\,, \\ g_B &= g_R + \delta g\,, \end{aligned} \tag{3.4}$$

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 ];
```

The second argument is optional and 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). 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

$$\delta g_s = \frac{\partial g_s}{\partial \alpha_s}\,\delta Z_{\alpha_s} = \sqrt{\frac{\pi}{\alpha_s}}\,\delta\alpha_s\,. \tag{3.5}$$

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

```
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 copy the output of the interface into the model directory of FeynArts. This can be done using the command

```
cp -r Tutorial <your FeynArts directory>/Models/
```

Next we need to load FeynArts into Mathematica,

```
<<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,

```
WriteCT["Tutorial", "Lorentz", Output-> "Tutorial",
     ZeroMom -> {{aS, {F[7], V[4], -F[7]}}},
     QCDOnly -> 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). Finally, the renormalisation is by default done in the on-shell scheme. 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, the model file and restrictions. In addition, you need to load the `.nlo` file produced in the previous step,

```
Get["Tutorial.nlo"];
```

Calling the UFO with the options

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

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

4. LO cross-section computation and event generation with MadGraph5_aMC@NLO

MadGraph5_aMC@NLOcan be run on a local computer or via the web at one of the following website:

- `http://madgraph.hep.uiuc.edu`
- `http://madgraph.phys.ucl.ac.be`

The registration is straightforward, and you can instantly creates optimized code for the computation of the cross-section of any processes. That code can be run directly on the web or downloaded and run locally. However, for security reason, generating events on the web is allowed only after that you have sent an email to one of the authors of MadGraph5. Since most of the functions are available on the internet, most users will not need to install MadGraph5.

The MadGraph5 collaboration plans to continue to improve this code both by adding new functionalities and by making the code easier to use. New tutorials and updates on this tutorial can be found at the following link:

```
https://server06.fynu.ucl.ac.be/projects/madgraph/wiki/MGTutorial
```

In this part of the tutorial, we will study a full example on how to generate events for BSM theories. We will assume that you have your own UFO model create accordingly to the previous section. If you don't, it is part of the tarball that you have downloaded from the FeynRules website.

We will split this into three sections. First we will explain in detail how to install the code, Then we will show how you can test the validity of the model. And finally we will present how you can generate BSM events, both with and without the associated decays.

4.1 Importing and checking the model

The simplest way to have access to a model in MG5 is to put it in the directory: MG5_DIR/models after that you can simply import it by typing

```
mg5> import model Tutorial_UFO
```

or

```
mg5> import model Tutorial_UFO --modelname
```

The option –modelname tells MG5 to use the name of the particles defines in the UFO model, and not the usual MG5 conventions for the particles of the SM/MSSM. For this particular model, this changes only the name associated to τ lepton (ta- and tt- respectively).

If you have developed your own model following the FeynRules tutorial, this will be the first time that you are going to use this model. It is therefore crucial to start by checking the model. MadGraph5 performs some sanity checks the first time that you load the model, but those test are quite weak. We therefore suggest to test, three properties for on a series of processes.

- The gauge invariance, by testing the Ward identities.
- The Lorentz invariance.
- The Aloha consistency, by evaluating the same square matrix element by different set of Helicity amplitudes.

For instance, we present how to check those properties for all the $2 \to 2$ BSM particles productions:

```
mg5> import model Tutorial_UFO
mg5> define new = uv uv~ ev ev~ p1 p2
mg5> check p p > new new
```

which results in the following output:

```
Gauge results:
Process          matrix           BRS              ratio            Result
g g > uv uv~     7.4113020914e-01 1.0055761722e-31 1.3568144434e-31 Passed
g u > uv p1      3.8373877873e-02 6.1629758220e-33 1.6060341471e-31 Passed
g u > uv p2      2.5726129500e-02 4.4176419953e-33 1.7171809678e-31 Passed
g u~ > uv~ p1    2.0117011717e-01 8.3456964257e-34 4.1485766093e-33 Passed
g u~ > uv~ p2    1.9705216573e-01 1.8809915790e-32 9.5456529090e-32 Passed
Summary: 5/5 passed, 0/5 failed
Lorentz invariance results:
Process          Min element      Max element      Relative diff.   Result
g g > uv uv~     4.8743514998e-01 4.8743514998e-01 1.1388417769e-16 Passed
g u > uv p1      1.3897148577e-01 1.3897148577e-01 1.1983282238e-15 Passed
g u > uv p2      5.7988729593e-02 5.7988729593e-02 5.9829676840e-16 Passed
g u~ > uv~ p1    8.8655875905e-03 8.8655875905e-03 1.7610238605e-15 Passed
g u~ > uv~ p2    1.0373598029e-01 1.0373598029e-01 8.0267932700e-16 Passed
u u > uv uv      2.1536620557e+00 2.1536620557e+00 1.8558171084e-15 Passed
u u~ > uv uv~    8.4033626775e-01 8.4033626775e-01 2.1138643036e-15 Passed
u u~ > p1 p1     7.6001259005e-03 7.6001259005e-03 2.9672410014e-15 Passed
u u~ > p1 p2     9.5102790993e-05 9.5102790993e-05 2.7645774018e-14 Passed
u u~ > p2 p2     4.6427735529e-04 4.6427735529e-04 8.6404128750e-15 Passed
u u~ > ev ev~    2.3919237366e-03 2.3919237366e-03 2.7196573767e-15 Passed
c c~ > uv uv~    8.5752329113e-01 8.5752329113e-01 2.3304340127e-15 Passed
d d~ > uv uv~    8.9275046680e-01 8.9275046680e-01 3.2333557600e-15 Passed
d d~ > ev ev~    4.8145327614e-04 4.8145327614e-04 5.6298410782e-16 Passed
s s~ > uv uv~    8.8941849408e-01 8.8941849408e-01 3.6199458330e-15 Passed
u~ u~ > uv~ uv~  2.3800428911e+00 2.3800428911e+00 1.8658874237e-15 Passed
Summary: 16/16 passed, 0/16 failed
Not checked processes: c c~ > ev ev~, s s~ > ev ev~
Process permutation results:
Process          Min element      Max element      Relative diff.   Result
g g > uv uv~     4.9184278197e-01 4.9184278197e-01 2.2572721717e-16 Passed
g u > uv p1      4.1859552985e-02 4.1859552985e-02 3.3153215498e-16 Passed
g u > uv p2      2.0129184319e-01 2.0129184319e-01 1.1030978772e-15 Passed
g u~ > uv~ p1    9.5566536137e-02 9.5566536137e-02 5.8086390357e-16 Passed
g u~ > uv~ p2    3.6165811126e-03 3.6165811126e-03 3.8372671248e-15 Passed
u u > uv uv      2.1101603787e+00 2.1101603787e+00 2.1045282356e-15 Passed
u u~ > uv uv~    1.3549964258e+00 1.3549964258e+00 1.1470969258e-15 Passed
u u~ > p1 p1     4.9555140623e-03 4.9555140623e-03 1.4002369515e-15 Passed
u u~ > p1 p2     2.0863569608e-02 2.0863569608e-02 6.6516842845e-16 Passed
u u~ > p2 p2     1.2914424155e-03 1.2914424155e-03 4.7013572521e-15 Passed
u u~ > ev ev~    1.7823584674e-03 1.7823584674e-03 0.0000000000e+00 Passed
c c~ > uv uv~    9.2608997797e-01 9.2608997797e-01 1.0789456164e-15 Passed
d d~ > uv uv~    8.3532448258e-01 8.3532448258e-01 1.3290919251e-16 Passed
```

```
d d~ > ev ev~    5.8126525280e-04 5.8126525280e-04 9.3262255680e-16 Passed
u~ u~ > uv~ uv~  2.1759606129e+00 2.1759606129e+00 2.0408880897e-16 Passed
Summary: 15/15 passed, 0/15 failed
```

More informations about these checks (like the values of the random phase-space points) can be obtained via the commands:

```
mg5> display checks
```

The display commands is very useful in order to obtained information on the particles, couplings, interactions, ... For example[5]

```
mg5>display particles
   Current model contains 21 particles:
   ve/ve~ vm/vm~ vt/vt~ e-/e+ m-/m+ tt-/tt+ u/u~ c/c~ t/t~ d/d~ s/s~ b/b~ w+/w- uv/u
   a z g h p1 p2
mg5>display particles p1
  Particle p1 has the following properties:
  {
     'name': 'p1',
     'antiname': 'p1',
     'spin': 1,
     'color': 1,
     'charge': 0.00,
     'mass': 'MPe1',
     'width': 'Wpe1',
     'pdg_code': 9000006,
     'texname': 'p1',
     'antitexname': 'p1',
     'line': 'dashed',
     'propagating': True,
     'is_part': True,
     'self_antipart': True
  }
mg5>
```

4.2 Generation of events

4.2.1 Generation of events with no decay

In this section, we start the computation of the cross-sections and the generation of events for the proposed process of interest:

$$pp \to U\bar{U}$$

First we will generate this exact process, Pythia being in charge of the decays. Note that in this way, you lose the full spin-correlations.

[5]Note that the spin is written in the $2S+1$ convention.

```
import model MODELNAME
generate p p  > uv uv~
output
launch
```

More examples, possibilities to generate the set of diagrams of interest are describe in Appendix (7).

What is prompted afterwards opens an interactive talk-to (which, again, can be scripted) which allows the user to choose among various options. Some options requires that the related package to be install via the "install" command in order to be used. This looks as follows:

```
The following switches determine which programs are run:
1 Run the pythia shower/hadronization:                pythia=OFF
2 Run PGS as detector simulator:                      pgs=OFF
3 Run Delphes as detector simulator:                  delphes=OFF
4 Decay particles with the MadSpin module:            madspin=OFF
5 Add weight to events based on coupling parameters:  reweight=OFF
Either type the switch number (1 to 5) to change its default setting,
or set any switch explicitly (e.g.  type 'madspin=ON' at the prompt)
Type '0', 'auto', 'done' or just press enter when you are done.
[0, 4, 5, auto, done, madspin=ON, madspin=OFF, madspin, reweight=ON, ...  ]
```

By entering 1, 2, 3, or 4 at the prompt one toggles between the two values of the corresponding feature (which are `ON` or `OFF`). For example, by entering 4 one is prompted again what is displayed above, except for the fact that `madspin=OFF` has now become `madspin=ON`. The various module correspond to the possibility to chain multiple type of simulation tools[6] More exactly:

`pythia=ON` $\longrightarrow$ Allows to run the Pythia6 programs for the shower and the hadronization. This program can be installed via the command `install pythia-pgs` in the MadGraph5_aMC@NLO shell command interface.

`pgs=ON` $\longrightarrow$ Allows to run the Pretty Good Simulator (PGS) programs for a basic/fast detector simulation. This program can be installed via the command `install pythia-pgs` in the MadGraph5_aMC@NLO shell command interface. Note that when `pgs=ON` then the pythia flag is set on `ON` automatically.

`delphes=ON` $\longrightarrow$ Allows to run the Delphes 3 programs [1] for a fast detector simulation. This program can be installed via the command `install Delphes` in the MadGraph5_aMC@NLOshell command interface. Note that when `pgs=ON` then the pythia flag is set on `ON` automatically, therefore in order to run Delphes, you also need to run the command `install pythia-pgs`. It is also possible to run Delphes 2 [2], the command to install that version of the code is via the command `install Delphes2`

`madspin=ON` $\longrightarrow$ Allows to include decay production with full spin correlations by means of MadSpin. Note that the decay-chain formalism is actually faster and

[6]For scripting we strongly discourage to use the number in the script file, and encourage to use the full flag identifier (`madspin=OFF`)

produce a cross-section which is more accurate (not based on the narrow-width approximation). So in most of the case, using the decay-chain formalism is advised rather than using MadSpinfor LO production. In order for MadSpinto run coherently, the generation of events needs to have unstable particles in the final state, in the opposite case, MadSpinwill crash.

`reweight=ON` $\longrightarrow$ Allows to associate to each events additional weights according to a different theoretical hypothesis. At current stage the only difference between the two theoretical hypothesis is those that correspond to a difference in the input parameter file (i.e. in the param_card.dat file). Additional reweighing for scale/pdf uncertainty are available via the `SysCalc` module .

By entering 0, or `done` or by simply hitting return, that talk-to phase ends and a second one starts for the edition of the parameter/external tools, the exact question depends of the requested modeled that are allowed to run. For example for a run with pythia module activated, the question will be:

```
Do you want to edit a card (press enter to bypass editing)?
1 / param :  param_card.dat
2 / run :  run_card.dat
3 / pythia :  pythia_card.dat
9 / plot :  plot_card.dat
you can also
- enter the path to a valid card or banner.
- use the 'set' command to modify a parameter directly.
The set option works only for param_card and run_card.
Type 'help set' for more information on this command.
- call an external program (ASperGE/MadWidth/...).
Type 'help' for the list of available command
[0, done, 1, param, 2, run, 3, pythia, 4, enter path, ...  ][60s to answer]
```

By typing one of the number and/or the associate name, you will open a text editor. [7] which allows to edit the file directly, the format of the file are most of the time self-explanatory and are quickly introduce below [8] In addition to the manual edition of the file, some special function are available to edit those cards automatically. Currently three commands are defined: `set`, `compute_widths`, `asperge`.

- `set` (syntax: set NAME VALUE) allows to edit the `param_card and run_card` without to have to open the file, this command allows easy scripting of the edition of the cards and allows easy scan over parameter.

- `compute_widths` (syntax: compute_widths PARTICLE(S) [OPTIONS]) compute the width (2 body decay and more if the code detects that two body is not the dominant

[7]By default, we use the one define in the sh variable $EDITOR, if not define when then use `vi` this setting can be modify via the configuration file of MadGraph5_aMC@NLO.

[8]Details on the meaning of the various parameter of the run_card are available at the following FAQ: https://answers.launchpad.net/mg5amcnlo/+faq/2014 .

contribution) at Tree-level and in the Narrow-width approximation via the `madwidth` module. If some of the width definition present in the param_card are set to the value `Auto` those widths will be computed in top of the list define in this command. If some width are left to Auto when the edition of the cards is completed then MadEvent will automatically called `madwidth` before starting the computation of the cross-section.

- `Asperge` (syntax AsperGe [BLOCK NAME]) allows to perform mass-matrix diagonalization for the currently define parameter sets. This is only valid for a restricted class of UFO models which include the `asperge` [3] module, this commands doesn't do anything if the module is not available.

Let's discuss here what those card are.

param_card The param_card contains all the external parameter of your model. If you enter "1" or "param" to the last question you will open that card in a text editor (vi or emacs by default) This one should looks like this (Note that I put only a part of the Card):

```
###################################
## INFORMATION FOR FRBLOCK
###################################
Block frblock
    1 2.000000e+02 # MM1
    2 3.000000e+02 # MM2
    3 5.000000e+01 # MM12
    4 1.000000e+00 # lam1
    5 1.000000e+00 # lam2
    6 1.000000e+00 # lam1p
    7 1.000000e+00 # lam2p

Block mass
    5 4.700000e+00 # MB
    6 1.720000e+02 # MT
   15 1.777000e+00 # MTA
   23 9.118760e+01 # MZ
   25 1.200000e+02 # MH
  9000008 5.000000e+02 # Muv
  9000009 2.500000e+02 # Mev
##  Not dependent paramater.
## Those values should be edited following the
## analytical expression. MG5 ignore those values
## but they are important for interfacing the output of MG5
## to external program such as Pythia.
  12 0.000000 # ve : 0.0
  14 0.000000 # vm : 0.0
  16 0.000000 # vt : 0.0
```

As it is clearly stated, some of the value of the param_card are not used by MG since those are in fact fixed by the UFO model. This information are however important for shower program (like Pythia).

Note that in general the entry for the param_card are not independent one of each other. One example is the fact that some matrix need to be unitary. A more annoying case is the the width of the particles which are correlated to the mass spectrum in a non trivial way.

A FeynRules/MadGraph collaboration solves this problem by, first, including in the UFO model a module able to compute all the two body decay partial and secondly use a dedicated tool to compute the three (and higher) body decay if it is necessary (the code determines automatically if it is the case or not). In the context of this model, the two body decay formula provided by FeynRules are actually enough. In order to use this module, you just have to replace the width value by `Auto`.

```
###################################
## INFORMATION FOR DECAY
###################################
DECAY   6 1.508336e+00 # WT
DECAY  23 2.495200e+00 # WZ
DECAY  24 2.085000e+00 # WW
DECAY  25 5.753088e-03 # WH
DECAY 9000006 Auto # Wpe1
DECAY 9000007 Auto # Wpe2
DECAY 9000008 Auto # Wuv
DECAY 9000009 Auto # Wev
```

At the time of the running of your code, you will see that your param_card would have been automatically updated and should contain the following information:

```
#      PDG        Width
DECAY  9000006   0.000000e+00
#
#      PDG        Width
DECAY  9000007   1.233920e+00
#  BR             NDA  ID1    ID2   ...
   5.000000e-01   2    9000009  -11 # 0.61696
   5.000000e-01   2    -9000009  11 # 0.61696
#
#      PDG        Width
DECAY  9000008   5.400300e+00
#  BR             NDA  ID1    ID2   ...
   5.858749e-01   2    2  9000006 # 3.1639
   4.141251e-01   2    2  9000007 # 2.2364
```

```
#
#      PDG        Width
DECAY  9000009   2.934500e-01
#  BR             NDA  ID1    ID2   ...
   1.000000e+00   2    11  9000006 # 0.29345
```

run_card The run_card contains all the parameter which are not model dependent. This include

- The parameter of the beam (energy, type, pdf, polarization)
- How to treat the scale.
- If and how to run matching (see the associate lectures)
- All the cuts that you want to perform at the **parton** level

Most of those are self-explanatory but few of them needs some explanation. Information about those parameter can be found here: https://answers.launchpad.net/madgraph5/+faq/2014

4.2.2 Generation of events with decay

In this case, the number of decay channels is quite limited, it's therefore possible to prescribe all the steps of the decay chain. The syntax for the decay chains is the following: the production first, then the decays of the final states separate by a comma. To avoid ambiguity parenthesis should be present in the case of sub-decay.

```
import model Tutorial_UFO
generate p p > uv uv~ , uv > u p1, uv~ > u~ p1
define l = e+ e-
define lv = ev ev~
add process p p > uv uv~, uv > u p1, \
                         (uv~ > u~ p2, (p2 > l lv, lv > l p1))
add process p p > uv uv~, uv~ > u~ p1,\
                         (uv > u p2, (p2 > l lv, lv > l p1))
add process p p > uv uv~, (uv > u p2, (p2 > l lv, lv > l p1)), \
                          (uv~ > u~ p2, (p2 > l lv, lv > l p1))
output
launch
```

The validity of this calculation must be considered, since specifying the decay sequence means the contribution of non-resonance Feynman diagram have been neglected. This is however valid since the interferences with those diagrams are negligible if the intermediate particles are on-shell. In order to ensure such a condition, we have associate to each of the decaying particles an additional cuts called BW_cut:

$$|m_{virt} - m_0| < BW_{cut} * \Gamma$$

This cut can be specified in the run_card.dat.

If you have installed the MadAnalysis package, you should have automatically distributions created at parton level, after Pythia and at the reconstructed level –if runned–. Those one can be used to check the sanity of the productions but also to search the potential observables of this production.

5. 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: The first is the same as in Sec. 4

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

for which we will compute the NLO K-factor in Sec. 5.1. 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]
```

5.1 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
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.

5.2 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). 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:

$$\cos\phi_{ll'} = \frac{\vec{p}_T(l)\cdot\vec{p}_T(l')}{|\vec{p}_T(l)||\vec{p}_T(l')|} \tag{5.1}$$

5.3 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

$$x + y \to u_1 + \dots u_p + s_1 + \dots s_q + X\,, \tag{5.2}$$

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

$$u_i \to d_{1,1} + \dots d_{1,n_1}\,. \tag{5.3}$$

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
```

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. 5.2.

6. Phenomenological analyses with MadGraph5_aMC@NLO and MadAnalysis 5

In this section, we present in details how to use MadGraph5_aMC@NLO and MadAnalysis 5 to investigate the properties of a Z+jets event sample. For the setup of the collisions, we choose the LHC collider running at a center-of-mass energy of 8 TeV. We focus on events where the Z-boson decays leptonically but we neglect, for the sake of the example, the $\tau^+\tau^-$ channel. We also include virtual photon exchange diagrams as the related effects cannot be neglected.

6.1 Event generation with MadGraph5_aMC@NLO

Matrix-element generator based event samples are accurate enough for describing hard and widely separated parton emissions. In contrast, it is known that this method breaks down in the soft and collinear limit. In this case, parton showering algorithms must be used for a proper description of the properties of the process under consideration. In the sequel, we therefore employ the MLM merging procedure, based on event rejection, to combine the strengths of matrix elements and the ones of parton showering in a consistent way.

As stated above, we focus on the generation of events related to a final state containing a pair of charged leptons, together with possible additional jets. In order to appropriately describe the kinematical properties of these jets, we merge matrix elements containing up to two additional hard jets to parton showering by employing the MLM merging scheme as implemented in MadGraph5_aMC@NLO.

Z+jets events can be efficiently generated in MadGraph5_aMC@NLO by means of multiparticles. Employing the two predefined symbols `l+` and `l-` that represent electrons and muons, the computation of the relevant matrix element is performed by typing in the command line interface of MadGraph5_aMC@NLO

```
generate    p p > l+ l-
add process p p > l+ l- j
add process p p > l+ l- j j
output FR_MG5aMC_MA5
launch -m
```

Let us note that writing the process in such a way allows to consistently include virtual photon contributions, in contrast to `generate p p > Z, Z > l+ l-` where the Z-boson is produced on-shell, the invariant mass of the lepton pair being always equal to the Z-mass.

MadGraph5_aMC@NLO then asks to configure the cards with the setup for event generation. The file `param_card.dat` containing the model parameters does not have to be modified, since we are using the default settings for the Standard Model. The latter can be inspected by browsing the built-in model library of MadGraph5_aMC@NLO. In a second step, the `run_card.dat` file must be edited so that the following information are provided to MadGraph5_aMC@NLO:

- For the sake of the example, we ask to generate 25000 events. The `nevents` parameter of the `run_card` must be updated accordingly.
- The beam energy has to be fixed to 4000 GeV (for each beam). The parameters `ebeam1` and `ebeam2` have to be updated.
- The MLM merging procedure must be switched on (`1 = ickkw`).
- Since the cuts on the jets are automatically handled by MadGraph5_aMC@NLO, the parameter `auto_pt_mjj` being set to true, all p_T-cuts can be safely removed.
- For the same reasons, cuts on the angular distance ΔR between jets can also be removed from the `run_card`.

- The Drell-Yan cross section explodes at low lepton-pair invariant mass. Therefore, a cut on the corresponding parameter must be added. We choose `50 = mmll`.

- The merging control parameter `xqcut` must be set. We choose to fix it to 10 GeV. The consistency of such a choice will have to be verified after event generation (by investigating the smoothness of the differential jet rate distributions). This will be performed below.

Finally, the file controlling PYTHIA, `pythia_card.dat`, contains default values (see the directory `Cards` and the file `pythia_card_default.dat`).

The event generation process leads to the creation of a collection of event files. We focus in the analysis on two of them all stored in the directory
`<path-to-MGaMC>/FR_MG5aMC_MA5/Events/run_01`:

- The hadron-level event file, including parton showering, hadronization and merging, `fermi_pythia_events.hep.gz`,

- The reconstructed-level event file, including parton showering, hadronization and merging, `fermi_pythia_events.lhe.gz`. In contrast to the `hep` file above, jet clustering has been performed by means of the program FASTJET called by the HEP2LHE routine. It consequently produces a simplified LHE file, with reconstructed objects, such as electrons, muons, photons, jets or missing energy, instead of tons of hadrons.

6.2 Analyzing reconstructed events with MADANALYSIS 5

In this section, we show how to investigate the properties of the event file `fermi_pythia_events.lhe.gz`. This file contains events where jets have been, internally to MADGRAPH5_AMC@NLO, clustered by means of a k_T-clustering algorithm.

6.2.1 Getting started

First, MADANALYSIS 5 has to be launched by issuing in a shell

```
bin/ma5
```

from the directory where MADANALYSIS 5 has been installed (see above). On start-up, two lists of labels corresponding to standard definitions of particles and multiparticles are loaded into the memory of the current session of the program. Among these, one finds labels for the charged leptons `l+` and `l-`, for the jets `j`, for the invisible particles `invisible` and for the particles taking part to the hadronic activity of the event `hadronic`.

Next, the generated event sample has to be loaded inside the current session of MADANALYSIS. This is achieved by typing in the command line interpreter

```
import <path-to-the-event-file>/fermi_pythia_events.lhe.gz as zjets
```

where `zjets` is a user-defined label for the dataset containing the sample. MADGRAPH5_AMC@NLO being used as a leading-order Monte Carlo generator, the returned value for the cross section is given at the leading-order accuracy. However, the cross section for the Drell-Yan

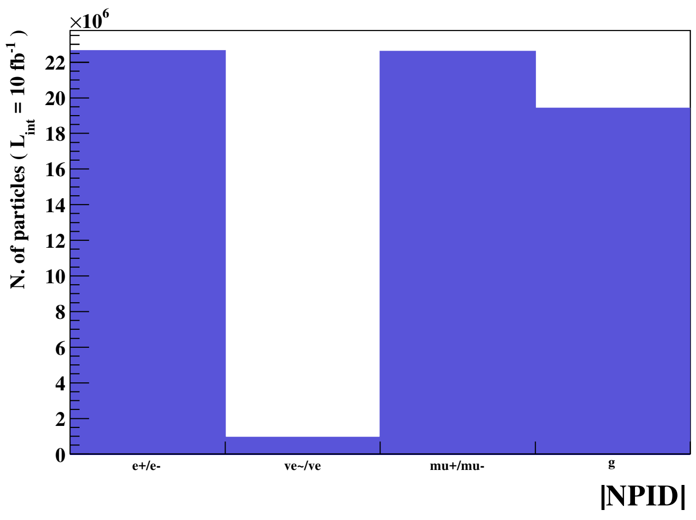

Figure 1: Particle content of the Z+jets event sample.

process is known at the next-to-next-to-leading order. Employing the Fewz program, one obtains $\sigma_{\rm DY} = 2263$ pb. This information can be passed to MadAnalysis 5 for a more precise normalization of the distributions,

```
set zjets.xsection = 2263
```

We are now ready to investigate the properties of our event sample.

As a first example, we would like to get an idea of the particle content of the events. To this aim, it is sufficient to type in the command line interpreter of MadAnalysis 5

```
plot NAPID
submit zjets
```

where we map particles and antiparticles (otherwise, the corresponding command would be `plot NPID`). The initial and intermediate particles present in the event file are automatically ignored by MadAnalysis 5, even if this default behavior can be modified by the user if necessary (we refer to the manual for more information). The result can be displayed by typing `open`. The corresponding histogram is presented on Figure 1. One observes the presence of 4 types of particles:

- electrons,
- muons,
- missing energy (which is assigned the PDG code of the electronic neutrino according to the simplified LHE format used by MadGraph5_aMC@NLO). Let us note that the computation of the missing energy is always based on the visible particles of the event and that care must be taken to the employed definition. The latter can indeed differ among the different available Monte Carlo and analysis tools.

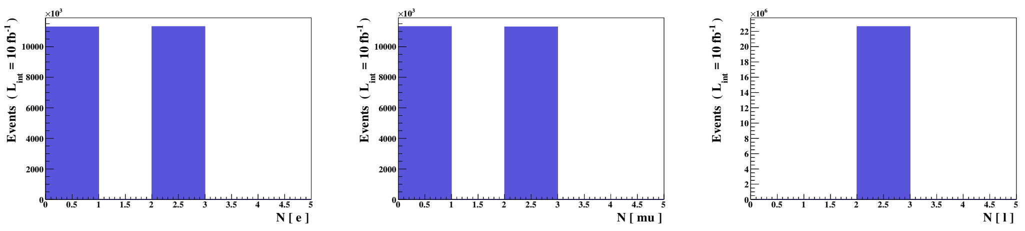
Figure 2: Lepton multiplicity of the Z+jets event sample.

- jets (which are assigned the PDG code of the gluon according to the simplified LHE format used by MadGraph5_aMC@NLO).

One can also note that MadAnalysis 5 indicates on the webpage that the event weight is too large ($\sim$ 1800) and that additional events should be generated. This weight has been computed according to the entered value for the cross section and the integrated luminosity (that can be change via `set main.lumi = ...`). By default, MadAnalysis 5 assumes an integrated luminosity of 10 fb^{-1}.

6.2.2 Lepton properties - a few examples

From Figure 1, we know that our event sample contains (among others) electrons and muons and we now study their kinematical properties. The set of commands

```
define e = e+ e-
define mu = mu+ mu-
define l = l+ l-
plot N(l) 5 0 5
plot N(e) 5 0 5
plot N(mu) 5 0 5
resubmit
```

allows, in a first stage, to define three multiparticle labels. While `e` and `mu` refer to electrons and muons, respectively, `l` refers to any type of charged leptons of the first two generations. Each of the next three commands ask for the production of one histogram. The layout of those histograms is such that they contain 5 bins on the x-axis ranging from 0 to 5. One observes the results on Figure 2. As expected, each event contains thus either two electrons or two muons, both issued from the decay of the Z-boson. This can be also illustrated by computing the dilepton invariant-mass distribution,

```
plot M(l+ l-) 50 0 200 [logY]
resubmit
```

where we ask for 50 bins ranging from 0 to 200 GeV and a logarithmic scale for the y-axis. We obtain the results of Figure 3. We have several entries in the bins related to the region $m_{ll} < 50$ GeV. Those are pure effects of the parton showering, hadronization and object reconstruction since we remind our parton-level cut enforcing $m_{ll} > 50$ GeV. To achieve our

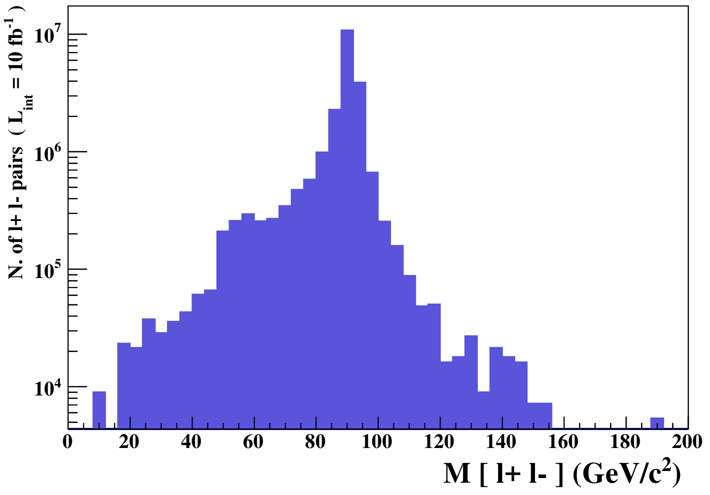

Figure 3: Dilepton invariant mass for a Z+jets event sample.

investigation of the lepton properties, let us generate a selection of additional histograms illustrating some of the possibilities of MadAnalysis 5

```
plot PT(l[1]) 30 0 250 [logY]
plot ETA(l[2]) 30 -5.5 5.5 [logY]
plot DELTAR(l[1],l[2]) 30 0 6 [logY]
resubmit
```

The first command addresses the transverse-momentum distribution of the leading lepton, the second one the pseudo-rapidity distribution of the next-to-leading lepton and the third one the angular distance in the $\eta - \varphi$ plane between the two leptons. The number of bin and the values for the lowest and highest bins are indicated for each case, and we employ a logarithmic scale for the y-axis. We leave this exercise to the reader and do not show the results in the present manuscript.

6.2.3 Jet properties - a few examples

The same exercise as in the previous section can be performed with the jets. Issuing

```
plot N(j) 10 0 10 [logY]
resubmit
```

leads to the results of Figure 4 which shows that after parton showering, one has much more than two jets in the events. For some rare events, one even has up to eight jets.

At the matrix-element level, we have accounted for up to two extra hard jets. As shown in Figure 4, parton showering yields much more (soft) jets in the final state. Jet hardness (and softness) can be checked by investigating the transverse-momentum distributions of, *e.g.*, the four leading jets. This is achieved by issuing in the interpreter the commands

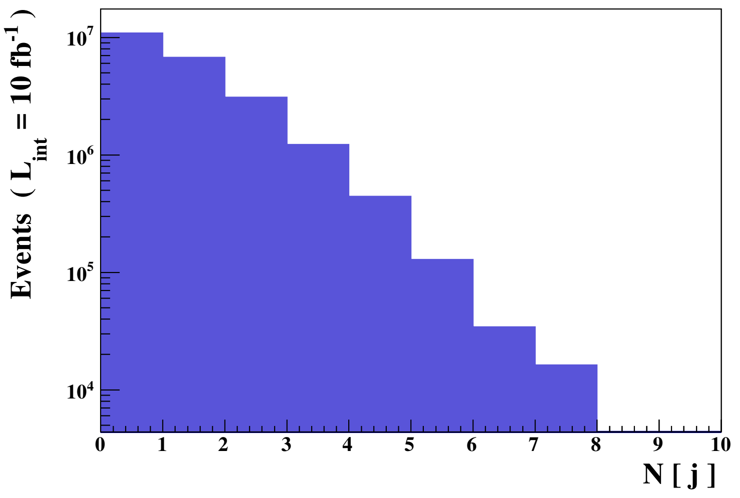

Figure 4: Jet multiplicity in a Z+jets event sample.

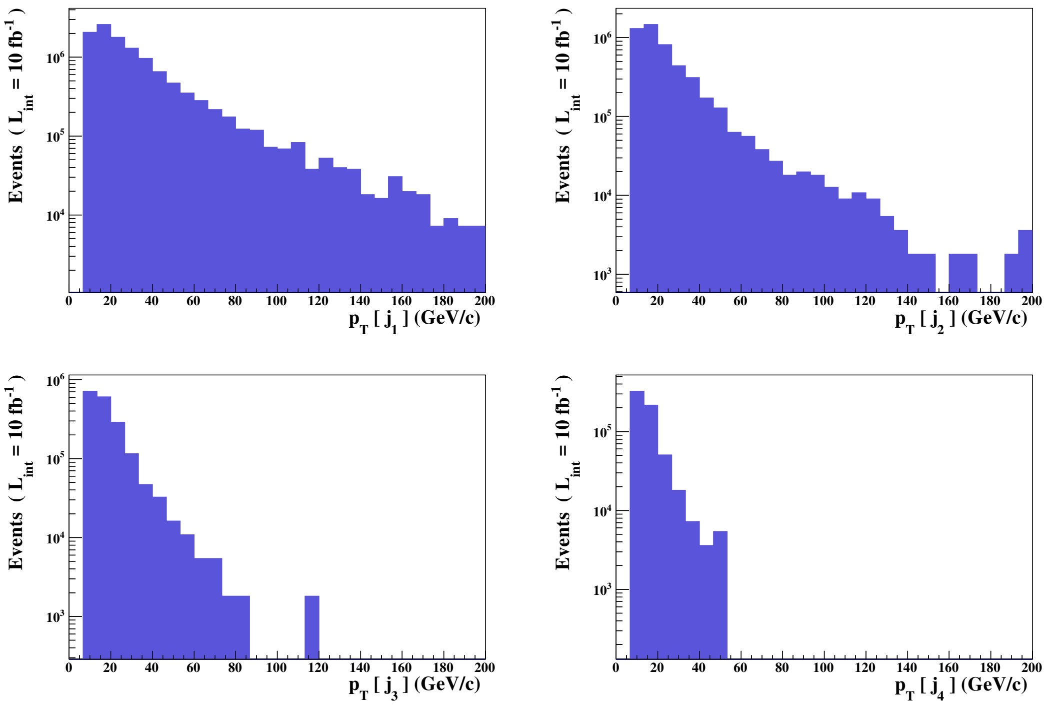

Figure 5: Transverse-momentum distributions of the four leading jets for a Z+jets event sample.

```
plot PT(j[1]) 30 0 200 [logY]
plot PT(j[2]) 30 0 200 [logY]
plot PT(j[3]) 30 0 200 [logY]
plot PT(j[4]) 30 0 200 [logY]
resubmit
```

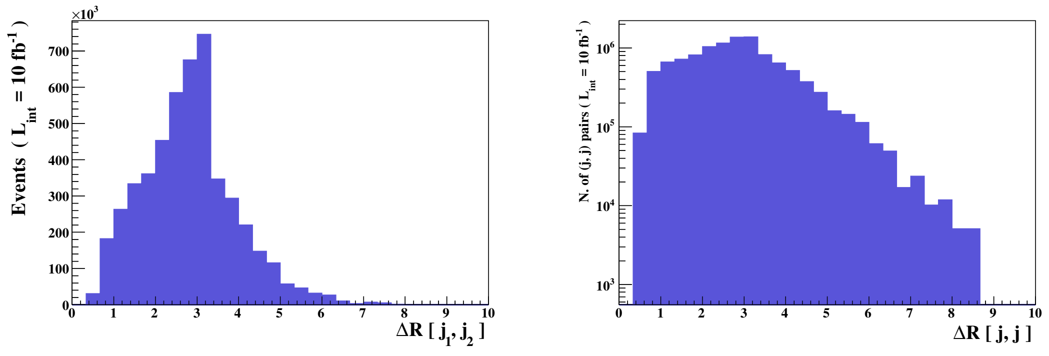

Figure 6: Distribution of the angular distance among the jets for a Z+jets event sample.

One gets the results of Figure 5. As expected, the two leading jets are indeed much harder than the others, since these two jets can be described already by the matrix-elements, in contrast to all the other jets. Inspecting the left part of the four figures, one observes a cut on jet transverse-momentum which has been automatically assigned by the merging algorithm implemented in MadGraph5_aMC@NLO.

The merging procedure also impose a cut on the angular distance among the jets. To illustrate this effect, we choose to represent by an histogram this observable,

```
plot DELTAR(j[1],j[2]) 30 0 10
plot DELTAR(j,j) 30 0 10 [logY]
resubmit
```

With the first command, we focus on the ΔR between the two leading jets, whilst while with the second command, we add one entry in the histogram for each different dijet combination that can be formed from the event particle content. The results are shown on Figure 6. The effect of the merging corresponds to the drop between zero and one, since a cut on the angular distance between jets is internally applied. Moreover, the shape of the distributions in the right panel comes mainly from the softer jets.

It could be useful (as it is the case for most phenomenological analyses), to consider as jets only jet candidates whose the transverse momentum is harder than some threshold. From Figure 5, one observes that setting this threshold to 70 GeV allows to get rid of most of the soft jets. Only the leading and possibly the next-to-leading jets remain. In some rare cases, we also have a three-jet configuration.

Such a cut and its effects can be studied by issuing

```
select (j) PT>70
plot N(j) 10 0 10 [logY]
plot DELTAR(j,j) 30 0 10 [logY]
resubmit
```

The selection cut imposed by means of the command `select` does not reject any event at all. It only removes one or several objects from the analysis. In this case, jets with a

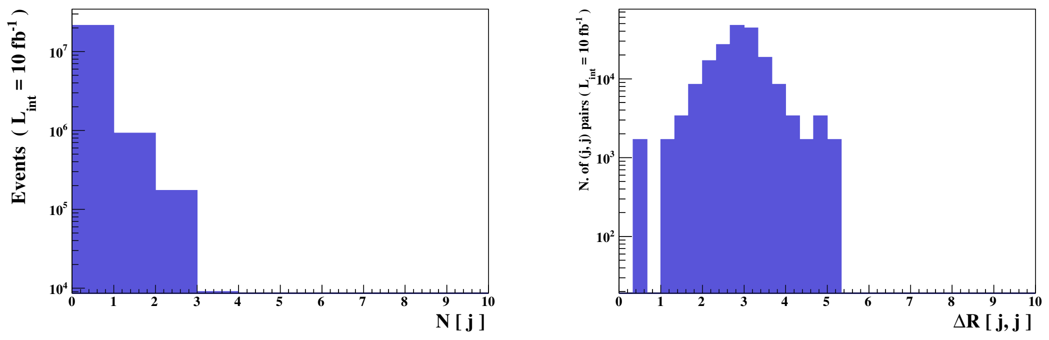

Figure 7: Distribution of the jet multiplicity and of the angular distance among the jets for a Z+jets event sample, after only considering jets with a transverse momentum harder than 70 GeV.

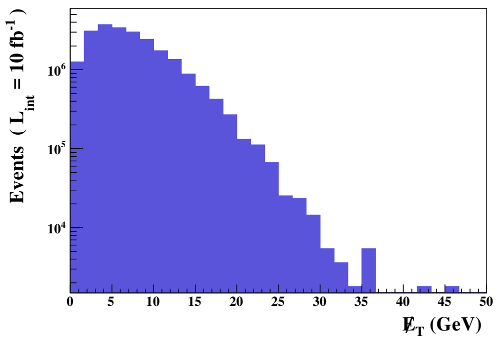

Figure 8: Missing transverse energy distribution for a Z+jets event sample.

transverse momentum smaller than 70 GeV are ignored. In order to reject or select entire events according to some criterion, we refer to the next section. The results are shown on Figure 7. One observes that the maximum jet multiplicity reaches the value of 3, whilst the considered (harder) jets are rather well separated.

As an exercise, it is left to the reader to probe various angular distances or invariant-mass distributions among jet and lepton combinations. Transverse-momentum distributions associated to the combination of different particle can also be probed.

6.2.4 The missing energy

To achieve our illustration of the properties of our event sample, we know tackle the missing energy distribution. The latter is hard-coded in MadAnalysis 5, and it is enough to type

```
plot MET 30 0 50 [logY]
resubmit
```

The results are presented in Figure 8. In the case one is only interested in events with a missing energy smaller than a given value, the selection cut

```
select MET<25
resubmit
```

allows to reject any event that the missing energy is larger than 25 GeV. MadAnalysis 5 automatically compute the cut-flow chart for the user. In the case there are several samples and in particular background and signal samples, the signal over background ratio and the related uncertainty are automatically calculated.

6.3 Controlling the merging procedure with MadAnalysis 5

In order to control that the merging procedure and in particular our choice for the merging parameters (in other words, the `xqcut` parameter of the `run_card`), the best way is to check differential jet rate distributions. These observables allow to measure how smooth the transition from a N-jet configuration to a $N+1$-jet configuration can be and are thus directly sensitive to the merging procedure. In the case of the transition $N \to N+1$, the differential jet rate variable is defined as the scale at which an event passes from a N-jet to a $N+1$-jet configuration, after clustering the showered partons into jets (but before hadronization).

MadAnalysis 5 allows to generate histograms representing differential jet rate distributions related to a given event sample. In order to test if the merging procedure has been correctly performed, it is necessary to start from an event file containing showered events that have not passed through an hadronization algorithm. Unfortunately, we do not have such a file. However, the StdHep sample that has been generated at the time of the Monte Carlo simulation contains the necessary information. MadAnalysis 5 is capable of automatically distinguishing the parton showering phase from the hadronization stage and can thus generate the relevant histograms allowing to check the merging procedure.

Jet clustering is performed with the help of FastJet, a k_T jet-algorithm being employed with a radius parameter set to $R = 1.0$. Since the merging procedure performed at the time of event generation is based on an earlier version of FastJet with respect to the one included in MadAnalysis 5, some minor issues can be expected, as it will be shown below. However, the latter are statistically negligible and do not alter the smoothness of the curves which prove the goodness of the merging.

First, as hadron-level events are about to be imported, MadAnalysis 5 must be run in the hadron-level mode,

```
bin/ma5 -H
```

Next, FastJet must be installed and linked to MadAnalysis 5. This procedure is automated and it is sufficient to type in the command line interface

```
install fastjet
```

Events can then be loaded as illustrated in the previous subsection and the cross section set according to next-to-next-to-leading order results

```
import <path-to-the-event-file>/fermi_pythia_events.hep.gz as zjets
set zjets.xsection = 2263
```

We are now ready to ask MadAnalysis 5 to generate the histograms allowing to check that the merged event sample behaves correctly. To this aim, it is enough to type in the interpreter,

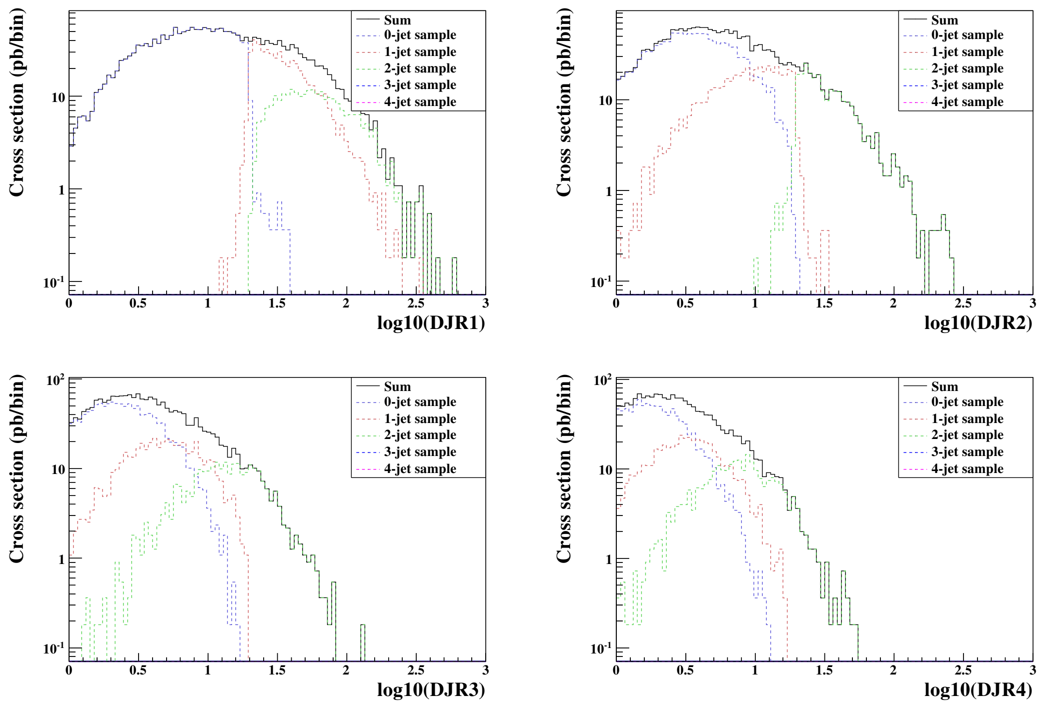

Figure 9: Differential jet rate distributions allowing to control the merging procedure.

```
set main.merging.check = true
submit
open
```

The histograms created by MadAnalysis 5 are presented on the four panels of Figure 9. They illustrate a transition from 0 to 1 jet (top, left), 1 to 2 jets (top, right), 2 to 3 jets (bottom, left) and 3 to 4 jets (bottom, right). One observes that a choice of `xqcut` equal to 10 GeV is a good choice, all the four summed curves being smooth enough. One also notes that the merging scale cannot be perfectly read from the figures ($Q^{\rm match} \approx$ 20 GeV), in contrast to the merging plots generated by the MatchChecker package of MadGraph5_aMC@NLO. This is related to the version issues of FastJet above-mentioned. One can however check that only a statistically small number of events are concerned and are irrelevant with respect to the smoothness of the summed curves.

Let us note that by default, MadAnalysis 5 generates four histograms, but histograms representing differential jet rate distributions associated to higher jet multiplicities can be created by typing in the interpreter

```
set main.merging.njets = <N>
```

where `<N>` is an integer number to be chosen by the user.

6.4 Jet clustering and hadron-level analysis with MadAnalysis 5

To achieve this section on the possibilities of MadAnalysis 5, we now focus on the analysis of the StdHep file generated by Pythia, after parton showering and hadronization. For comparison purposes, the same analysis as in Section 6.2 will be performed. However, before moving on, let us comment briefly on the expected differences between the analysis of the Lhe file and the one of the Hep file.

First of all, the Hep file contains tons of hadrons that must be clustered into jets. Since the jet clustering algorithm that we will adopt is different as the one internally employed in the Hep2Lhe routine of MadGraph5_aMC@NLO, differences in jet-related distributions can be expected. For the sake of the example, we choose to use an anti-k_T algorithm with a radius parameter set to $R = 1.0$ and a minimum transverse-momentum of 5 GeV. This is achieved by first starting MadAnalysis 5 in the reconstructed-level mode,

```
bin/ma5 -R
```

and then typing the commands

```
set main.fastsim.package = fastjet
set main.fastsim.algorithm = antikt
set main.fastsim.ptmin = 5
set main.fastsim.radius = 1
import <path-to-the-event-file>/frmg.hep.gz as zjets
set zjets.xsection = 2263
set main.normalize = lumi
```

where the two last command lines ensure a correct normalization of the histograms to be generated. We recall that by default, in the reconstructed-level mode, the histograms are not normalized at all and each event has a weight of one. In order to remove the very soft jets issued from parton showering and hadronization, we decide to only keep in the analysis jets with a transverse-momentum higher than 10 GeV,

```
reject (j) PT < 10
```

Moreover, the Hep2Lhe routine also contains a rough detector simulation, in contrast to MadAnalysis 5 which does not alter the reconstructed objects as this task is left for a fast detector simulation program possibly employed by the user. On the same lines, non-isolated leptons generated by the hadronization procedure are kept in MadAnalysis 5 while they are removed by the Hep2Lhe routine. Since those leptons are usually soft, they can be rejected from the analysis by employing a cut on their transverse momentum that can be implemented as

```
reject (l) PT < 5
```

We are now ready to perform the same study as in Section 6.2 and try to understand the differences between the results. This task is left as an exercise. In particular, attention should be paid to the missing energy distributions and the particle content of the clustered events.

Fast detector simulation by means of Delphes 3 could also be achieved. To this aim, it is sufficient to type

```
install delphes
```

to install the Delphes 3 software within MadAnalysis 5. Then, it could be used on a hadron-level event file as follows,

```
set main.fastsim.package = delphes
set main.fastsim.detector = cms
import <path-to-the-event-file>/frmg.hep.gz as zjets
submit
```

The `root` outputted by Delphes can then be subsequently analyzed or reused (it is located in the working directory, in the `Output` subdirectory).

7. The simulation session

Once the model has been exported to MadGraph a first elementary phenomenological study can be performed. Schematically this session proceeds as follows:

- Benchmark parameter setting
- Signal identification, simulation, and study
- Background identification, simulation and study
- Signal vs Background study
- Comparison with pseudo-experimental data.

To begin with let us assume that

$$M_U > M_2 > M_E > M_1 \,, \tag{7.1}$$

provides a reasonable mass hierarchy and therefore Φ_1 is the LNP. For U we consider three scenarios, $m_U = 200, 400, 800$ GeV, while we always take $M_2 = 100$ GeV and $M_E = 50 GeV$ and $M_1 = 1$ GeV.

Given that U is the only strongly interacting NP particle, this will be the one most copiously produced at the LHC, via the same subprocesses as top-anti-top are produced:

$$p\,p \to \overline{U}\,U \,. \tag{7.2}$$

Exercise 1: Generate the process at LO with MadGraph 5, and determine the cross section at the LHC 8 TeV for the three benchmark values of the U mass. Optional: generate the procecess at NLO with MadGraph 5 and find the K-factor for each of the three masses above. To this aim, use the `Tutorial_NLO_UFO` model as provided in the Wiki page.

Next we consider the possible decay chains given the hierarchy of Eq. (7.1):

$$\begin{aligned} &U \to \{u,c,t\}\,\Phi_1 \,, \\ &U \to \{u,c,t\}\,\Phi_2 \,, \quad \Phi_2 \to \ell\,E \,, \quad E \to \ell\,\Phi_1 \quad \Rightarrow \quad U \to \{u,c,t\}\,\ell^+\ell^-\,\Phi_1 \,. \end{aligned} \tag{7.3}$$

ℓ being a label that includes all flavor, $\ell = e, \mu, \tau$. Obviously having the U decaying to a light quark or a top gives very different final state signatures.

Exercise 2: First classify all possible final states in terms of the number of tops, jets ($j = u, c$) and charged leptons. Then consider the two possible decay modes for the W in the top decays, i.e. hadronically or leptonically.

For the sake of simplicity, in the following we will focus on the following simple signatures:

I. $pp \to (U \to j\Phi_1)(\bar{U} \to j\Phi_1)$, i.e., $pp \to 2$ jets + missing E_T.

II. $pp \to (U \to t\Phi_1)(\bar{U} \to \bar{t}\Phi_1)$, i.e., $pp \to t\bar{t}$ + missing E_T.

III. $pp \to (U \to j\Phi_1)(\bar{U} \to j\,\ell^+\ell'^-\,\Phi_1)$+h.c , i.e., $pp \to \ell^+\ell^-$+ 2 jets + missing E_T.

IV. $pp \to (U \to j\,\ell^+\ell'^-\,\Phi_1)(\bar{U} \to j\,\ell^+\ell^-\,\Phi_1)$ + h.c , i.e., $pp \to \ell^+\ell^-\ell^+\ell^-$+ 2 jets + missing E_T.

Exercise 3: Pick one of the processes/signatures above, allowing yourself to select a specific flavor assignment for the final state leptons. Calculate the corresponding rates with MadGraph at LO. (You can proceed in various ways). Possibly, identify the cross section corresponding to a simplified detector acceptance.

Exercise 4: Identify the dominant reducible and irreducible SM backgrounds to the signatures above. Generate them with MadGraph, calculate the corresponding rates and order them in importance. Justify the following choices for the dominant backgrounds:

I. $pp \to (Z \to \nu\bar{\nu})$+2 jets.

II. $pp \to t\bar{t}$

III. $pp \to t\bar{t} \to \ell^+\ell^-$+ 2 b-jets + missing E_T

IV. $pp \to t\bar{t}Z$

Exercise 5: Depending on the chosen final state signature create the codes and do event generation for the most relevant backgrounds:

I. $pp \to (Z \to \nu\bar{\nu})$+2 jets with the ME/PS merging of $Z + 0, 1, 2$ partons.

II. $pp \to t\bar{t}$ with aMC@NLO and the decays with the DecayPackage.

III. $pp \to t\bar{t} \to \ell^+\ell^-$+ 2 b-jets + missing E_T with MC@NLO and the decays with the DecayPackage.

IV. $pp \to t\bar{t}Z \to \ell^+\ell^-\ell^+\ell^-$ + 2 b-jets + missing E_T with MadGraph 5 and the decays with the DecayPackage.

Exercise 6: Study the distributions of the signal and the background in the acceptance region and identify simple cuts to enhance $S/\sqrt{B}$ keeping S/B as large as possible. Do this via MadAnalysis 5.

Exercise 7: Compare your predictions with two sets (A and B) of pseudo LHC data. Set limits or establish evidence of new physics in the data.

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 3: 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*.

8. 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 3, 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 3, 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 $m^\star_{\tt x}$ of `x` is forced to be in the following range:

$$|m^\star_{\tt x} - m_{\tt x}| \leq \texttt{bwcutoff}\,\Gamma_{\tt x}\,, \tag{8.1}$$

where $m_{\tt x}$ is the pole mass of `x`, $\Gamma_{\tt x}$ 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:

$$|m^\star_{\tt x} - m_{\tt x}| > \texttt{bwcutoff}\,\Gamma_{\tt x}\,, \tag{8.2}$$

which is the region complementary to that of eq. (8.1), and thus loosely imposes an off-shell condition. All diagrams are kept. In the example of table 3, 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 $\texttt{bwcutoff}\,\Gamma_Z$. 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:

$$\frac{d\sigma}{dO}(\texttt{p p > z}) \simeq \frac{d\sigma}{dO}(\texttt{p p > z, z > e+ e-}) + \frac{d\sigma}{dO}(\texttt{p p > e+ e- \$ z})\,, \tag{8.3}$$

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. [4]), and have therefore to be used with extreme caution.

References

[1] **DELPHES 3** Collaboration, J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi, *DELPHES 3, A modular framework for fast simulation of a generic collider experiment*, *JHEP* **02** (2014) 057, [`1307.6346`].

[2] S. Ovyn, X. Rouby, and V. Lemaitre, *DELPHES, a framework for fast simulation of a generic collider experiment*, `0903.2225`.

[3] A. Alloul, J. D'Hondt, K. De Causmaecker, B. Fuks, and M. Rausch de Traubenberg, *Automated mass spectrum generation for new physics*, *Eur. Phys. J.* **C73** (2013), no. 2 2325, [`1301.5932`].

[4] A. S. Papanastasiou, R. Frederix, S. Frixione, V. Hirschi, and F. Maltoni, *Single-top t-channel production with off-shell and non-resonant effects*, *Phys. Lett.* **B726** (2013) 223–227, [`1305.7088`].