Claudeth Hernandez
Douglas M. Gingrich, PhD

Starting MadGraph

UNIVERSITY OF ALBERTA

Introduction

Madgraph is an event generator based on Monte Carlo simulations. The purpose of this framework is to provide elements to study the Standard Model (SM) and the physics Beyond Standard Model (BSM). Madgraph can be used to compute cross sections, make Feynman diagrams, work whit multi-particles (just like jets), import new models, tools for event manipulation and analysis, etc. In this document, I´ll show you a basic guide to introduce, install, use, and other utilities of this program oriented to Ubuntu Linux.

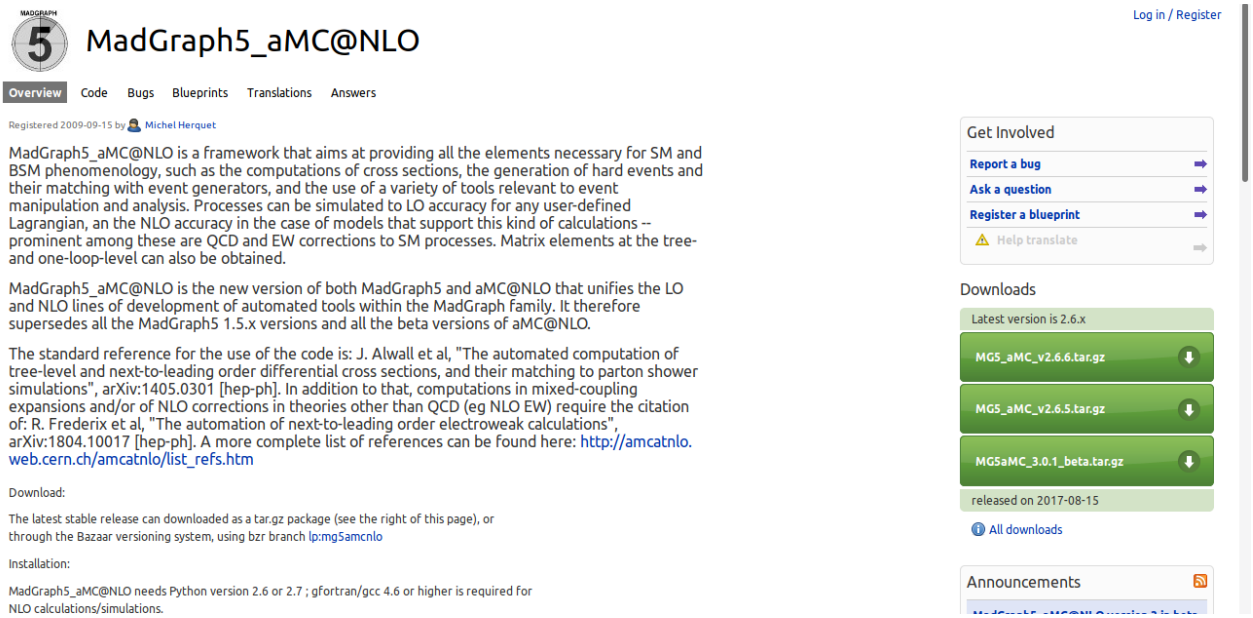
Basic concepts

All matter is made of elementary particles, that is, particles that are not constituted of other particles. Physicists study these properties using SM, a physic theory that involves particles and their interactions. In nature exists four fundamental interactions. Two of them result familiar to us, electromagnetism and gravity force. The other two are not measurable in our daily scales, their effects can be noticed when we talk about subatomic scales. These interactions are the strong and weak nuclear interactions. The strong interaction is responsible for maintaining the nucleus together, and the weak interaction is the one in charge of decays (when a particle disintegrates to multiple particles). SM study these interactions (excepting gravity) through particles, named as Bosons, which are Photons, Gluons, W+, W-, Z, and Higgs. Photons are responsible for electromagnetism, Gluons of the strong interaction, W+, W-, and Z of the weak interaction. Higgs is the boson who gives mass to all the particles. On the other hand, we have the particles that constitute matter, named as Fermions, which are divided into quarks and leptons. Quarks are the particles that form the nucleus (proton and neutrons). Exists 6 types of quarks; up, down, charm, strange, top and bottom. What happens with electrons? Here come the other type of Fermion particles, Leptons. Another example of leptons particles is neutrinos. How do we know all these? Well, to study the properties of the particles, physicists accelerate these particles to high energies and make them collide with each other, an example of this is LHC. The collisions (also known as scatterings) are usually between protons, and when they collide, it occurs a bunch of interesting things, like the production of cones of other particles (these cones are called jets). Mathematically, we can represent subatomic processes in a very easy way, the Feynman Diagrams. Feynman Diagrams have their origin on the Feynman´s Path Integral formulation (PI). Feynman Diagrams can be represented by three channels, s-channel (space-channel), t-channel (time channel) and u-channel (unitary-channel). These (s,t,u) are known as Mandelstam variables, variables that encode angles, energy, and momentum in a collision. Time later, Feynman discovered the Quantum Electrodynamics (QED), a beautiful theory that describes the electromagnetic interaction at quantum scales. Inspired on QED, born the Quantum Chromodynamics (QCD), a theory that describes the nuclear strong interaction.

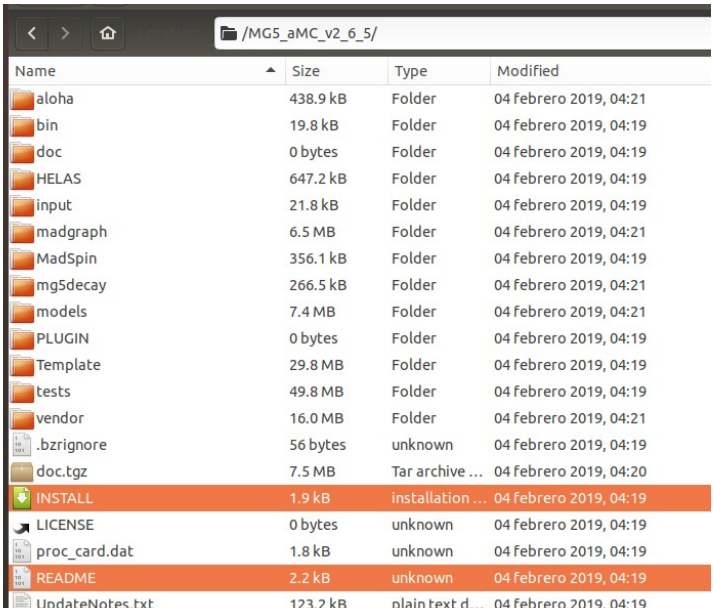
Download and Installation

We will use the following link:
https://launchpad.net/mg5amcnlo?fbclid=IwAR3XKkA-_w6pRFZYF6dqCBbiPoKAqIzghXL4dIcCPZx0Vn84fq0YljdDz7U

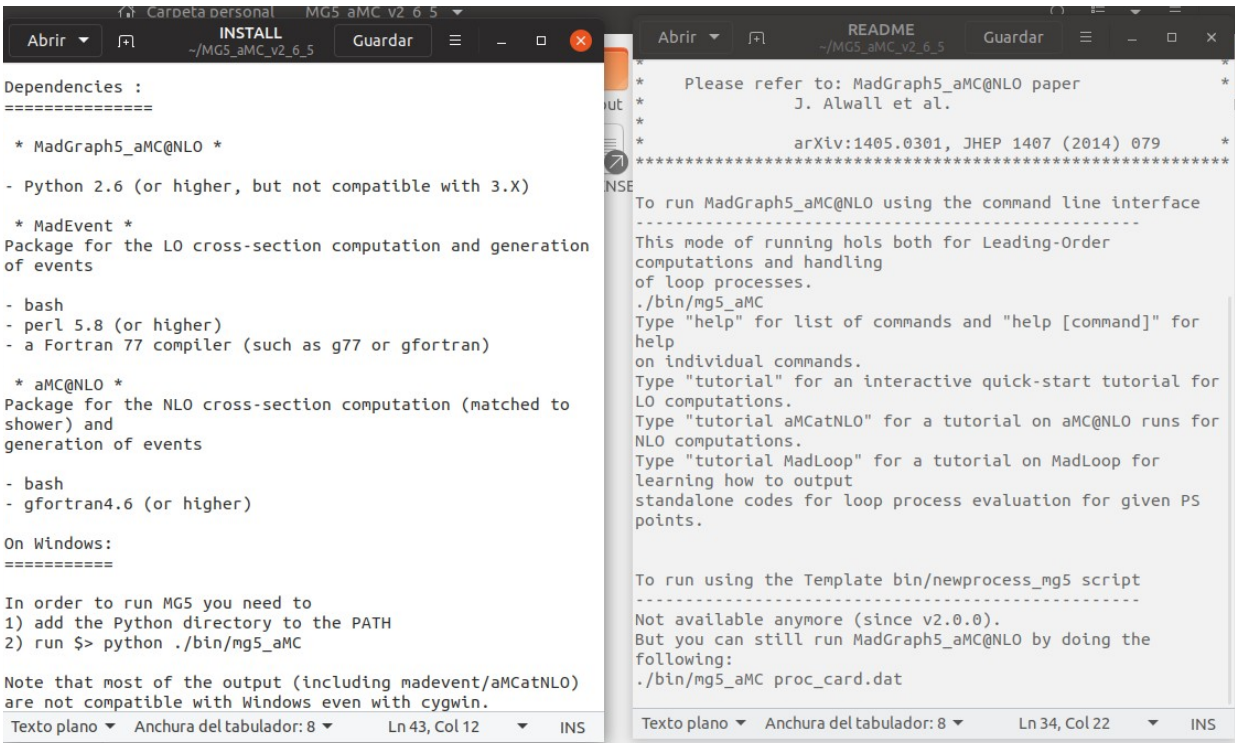
The website looks like:



We will download one of the green boxes, in my case I downloaded the first box with name: MG5_aMC_v2.6.6.tar.gz. When you have downloaded the package I recommend to change the name of the directory, to something shorter and easy to write. I named it “madgraph265”, then check two files: README and INSTALL.



When you open README and INSTALL you should see:



In the INSTALL file it said:

```
************************************************************
*                                                          *
*   W E L C O M E  to  M A D G R A P H 5 _ a M C @ N L O   *
*                                                          *
*                                                          *
*                  *                       *               *
*                     *         * *         *              *
*                        * * * * 5 * * * *                 *
*                     *         * *         *              *
*                  *                       *               *
*                                                          *
*    The MadGraph5_aMC@NLO Development Team - Find us at   *
*    https://server06.fynu.ucl.ac.be/projects/madgraph     *
*                          and                             *
*                 http://amcatnlo.cern.ch                  *
*                                                          *
*                   Code download from:                    *
*              https://launchpad.net/madgraph5             *
*                                                          *
************************************************************
```

```
Dependencies :
==============

 * MadGraph5_aMC@NLO *

- Python 2.6 (or higher, but not compatible with 3.X)

 * MadEvent *
Package for the LO cross-section computation and generation of events

- bash
- perl 5.8 (or higher)
- a Fortran 77 compiler (such as g77 or gfortran)

 * aMC@NLO *
Package for the NLO cross-section computation (matched to shower) and
generation of events

- bash
- gfortran4.6 (or higher)

On Windows:
===========

In order to run MG5 you need to
1) add the Python directory to the PATH
2) run $> python ./bin/mg5_aMC

Note that most of the output (including madevent/aMCatNLO)
are not compatible with Windows even with cygwin.
```

We should make sure that we have installed beforehand everything down “Dependencies”.

To know if you have something installed, you could write:
something --version on the Linux terminal.
For example, If we want to know if we have installed gfortran, write:
gfortran --version

If some kind of error appears it means that it is not installed yet.
To install it you could write:
sudo apt-get install gfortran
If it is already installed then you should see something like this:

```
clarissa@claudeth:~$ gfortran --version
GNU Fortran (Ubuntu 8.3.0-6ubuntu1) 8.3.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

clarissa@claudeth:~$
```

When we have verified that we have installed all, we change directory (**cd**) where we have Madgraph (In my case madgraph265), then we change of directory again but now to **bin** directory. Once we are there, we can use the **ls** command to observe what is inside the directory.

```
clarissa@claudeth:~$ cd madgraph265/
clarissa@claudeth:~/madgraph265$ cd bin
clarissa@claudeth:~/madgraph265/bin$ ls
mg5  mg5_aMC  MY_FIRST_MG5_RUN  MY_SECOND_RUN  py.py
clarissa@claudeth:~/madgraph265/bin$ python mg5_aMC
```

You can observe that there are a few python files inside there (Ignore the blue ones, we will learn how to generate them later) . If we try to run mg5_aMC and mg5, MadGraph should be open. To run the mg5_aMC file write: **python mg5_aMC** .

```
clarissa@claudeth:~/madgraph265/bin$ python mg5_aMC
************************************************************
*                                                          *
*                    W E L C O M E to                      *
*              M A D G R A P H 5 _ a M C @ N L O           *
*                                                          *
*                                                          *
*                *                       *                 *
*                  *        * *        *                  *
*                    * * * * 5 * * * *                     *
*                  *        * *        *                  *
*                *                       *                 *
*                                                          *
*         VERSION 2.6.5                2018-02-03          *
*                                                          *
*    The MadGraph5_aMC@NLO Development Team - Find us at   *
*    https://server06.fynu.ucl.ac.be/projects/madgraph     *
*                          and                             *
*            http://amcatnlo.web.cern.ch/amcatnlo/         *
*                                                          *
*               Type 'help' for in-line help.              *
*           Type 'tutorial' to learn how MG5 works         *
*    Type 'tutorial aMCatNLO' to learn how aMC@NLO works   *
*    Type 'tutorial MadLoop' to learn how MadLoop works    *
*                                                          *
************************************************************
load MG5 configuration from ../input/mg5_configuration.txt
fastjet-config does not seem to correspond to a valid fastjet-config executable (v3+). We will use fjcore instead.
 Please set the 'fastjet'variable to the full (absolute) /PATH/TO/fastjet-config (including fastjet-config).
 MG5_aMC> set fastjet /PATH/TO/fastjet-config

lhapdf-config does not seem to correspond to a valid lhapdf-config executable.
Please set the 'lhapdf' variable to the (absolute) /PATH/TO/lhapdf-config (including lhapdf-config).
Note that you can still compile and run aMC@NLO with the built-in PDFs
 MG5_aMC> set lhapdf /PATH/TO/lhapdf-config
Using default text editor "vi". Set another one in ./input/mg5_configuration.txt
Using default eps viewer "evince". Set another one in ./input/mg5_configuration.txt
Using default web browser "firefox". Set another one in ./input/mg5_configuration.txt
Checking if MG5 is up-to-date... (takes up to 2s)
failed to connect server
Loading default model: sm
INFO: Restrict model sm with file ../models/sm/restrict_default.dat .
INFO: Run "set stdout_level DEBUG" before import for more information.
INFO: Change particles name to pass to MG5 convention
Defined multiparticle p = g u c d s u~ c~ d~ s~
Defined multiparticle j = g u c d s u~ c~ d~ s~
Defined multiparticle l+ = e+ mu+
Defined multiparticle l- = e- mu-
Defined multiparticle vl = ve vm vt
Defined multiparticle vl~ = ve~ vm~ vt~
Defined multiparticle all = g u c d s u~ c~ d~ s~ a ve vm vt e- mu- ve~ vm~ vt~ e+ mu+ t b t~ b~ z w+ h w- ta- ta+
MG5_aMC>
```

But, Why we choose the mg5_aMC file instead of the mg5 file? Well, let's prove with mg5 file, write **python mg5** to run it.

```
clarissa@claudeth:~/madgraph265/bin$ ls
mg5  mg5_aMC  MY_FIRST_MG5_RUN  MY_SECOND_RUN  py.py
clarissa@claudeth:~/madgraph265/bin$ python mg5
Warning: The script ./bin/mg5 will be removed in future versions; use ./bin/mg5_aMC instead.
************************************************************
*                                                          *
*                    W E L C O M E to                      *
*              M A D G R A P H 5 _ a M C @ N L O           *
*                                                          *
*                                                          *
*                 *                       *                *
*                   *        * *        *                  *
*                     * * * * 5 * * * *                    *
*                   *        * *        *                  *
*                 *                       *                *
*                                                          *
*         VERSION 2.6.5                 2018-02-03         *
*                                                          *
*    The MadGraph5_aMC@NLO Development Team - Find us at   *
*    https://server06.fynu.ucl.ac.be/projects/madgraph     *
*                           and                            *
*            http://amcatnlo.web.cern.ch/amcatnlo/         *
*                                                          *
*               Type 'help' for in-line help.              *
*           Type 'tutorial' to learn how MG5 works         *
*    Type 'tutorial aMCatNLO' to learn how aMC@NLO works   *
*    Type 'tutorial MadLoop' to learn how MadLoop works    *
*                                                          *
************************************************************
load MG5 configuration from ../input/mg5_configuration.txt
fastjet-config does not seem to correspond to a valid fastjet-config executable (v3+). We will use fjcore instead.
 Please set the 'fastjet'variable to the full (absolute) /PATH/TO/fastjet-config (including fastjet-config).
 MG5_aMC> set fastjet /PATH/TO/fastjet-config

lhapdf-config does not seem to correspond to a valid lhapdf-config executable.
Please set the 'lhapdf' variable to the (absolute) /PATH/TO/lhapdf-config (including lhapdf-config).
Note that you can still compile and run aMC@NLO with the built-in PDFs
 MG5_aMC> set lhapdf /PATH/TO/lhapdf-config
```

How we can observe MadGraph is warning us and recommend us that we should use the mg5_aMC version, so until now we will use that version instead.

Well, we are done! We can start learning MadGraph now!

Particle notation

Standard Model notation

Fermions				Bosons	
type	1st gen	2nd gen	3rd gen	name	symbol
quarks	u (u~)	c (c~)	t (t~)	gluon	g
	d (d~)	s (s~)	b (b~)	photon	a
leptons	ve (ve~)	vm (vm~)	vt (vt~)	weak bosons	w+ (w-), z
	e- (e+)	mu- (mu+)	ta- (ta+)	higgs	h

Minimal Super Symmetric Standard Model (MSSM)

Scalars				Fermions	
type	1st gen	2nd gen	3rd gen	name	symbol
squarks	ul (ul~), ur (ur~)	cl (cl~), cr (cr~)	t1 (t1~),t2 (t2~)	gluino	go
	dl (dl~), dr (dr~)	sl (sl~), sr (sr~)	b1 (b1~),b2 (b2~)	neutralinos	n1,n2,n3,n4
sleptons	sve (sve~)	svm (svm~)	svt (svt~)	charginos	x1- (x1+), x2- (x2+)
	el- (el+), er- (er+)	mul- (mul+), mur- (mur+)	ta1- (ta1+),ta2- (ta2+)		
higgses	h1,h2,h3,h- (h+)				

General two-Higgs-doublet Model (2HDM)

Fermions				Bosons	
type	1st gen	2nd gen	3rd gen	name	symbol
quarks	u (u~)	c (c~)	t (t~)	gluon	g
	d (d~)	s (s~)	b (b~)	photon	a
leptons	ve (ve~)	vm (vm~)	vt (vt~)	weak bosons	w+ (w-), z
	e- (e+)	mu- (mu+)	ta- (ta+)	higgses	h1,h2,h3,h+ (h-

Exploring MadGraph

These are some basic MadGraph commands that you can prove to familiarize with the program:

display particles

```
MG5_aMC>display particles
Current model contains 17 particles:
w+/w- ve/ve~ vm/vm~ vt/vt~ u/u~ c/c~ t/t~ d/d~ s/s~ b/b~ e-/e+ mu-/mu+ ta-/ta+
a z g h
MG5_aMC>
```

display multiparticles

```
MG5_aMC>display multiparticles
Multiparticle labels:
all = g u c d s u~ c~ d~ s~ a ve vm vt e- mu- ve~ vm~ vt~ e+ mu+ t b t~ b~ z w+
h w- ta- ta+
l- = e- mu-
j = g u c d s u~ c~ d~ s~
vl = ve vm vt
l+ = e+ mu+
p = g u c d s u~ c~ d~ s~
vl~ = ve~ vm~ vt~
MG5_aMC>
```

display interactions

```
Current model contains 56 interactions
1:h h h h QED=2
2:h h h QED=1
3:g g g QCD=1
4:g g g g QCD=2
5:a w- w+ QED=1
6:w- w+ h h QED=2
7:w- w+ h QED=1
8:a a w- w+ QED=2
9:w- w+ z QED=1
10:w- w- w+ w+ QED=2
11:a w- w+ z QED=2
12:z z h h QED=2
13:z z h QED=1
14:w- w+ z z QED=2
15:d~ d a QED=1
16:s~ s a QED=1
17:b~ b a QED=1
18:d~ d g QCD=1
19:s~ s g QCD=1
20:b~ b g QCD=1
21:b~ b h QED=1
22:d~ d z QED=1
23:s~ s z QED=1
24:b~ b z QED=1
25:u~ d w+ QED=1
26:c~ s w+ QED=1
27:t~ b w+ QED=1
28:e+ e- a QED=1
29:mu+ mu- a QED=1
30:ta+ ta- a QED=1
31:ta+ ta- h QED=1
32:e+ e- z QED=1
33:mu+ mu- z QED=1
34:ta+ ta- z QED=1
35:ve~ e- w+ QED=1
36:vm~ mu- w+ QED=1
37:vt~ ta- w+ QED=1
38:d~ u w- QED=1
39:s~ c w- QED=1
40:b~ t w- QED=1
41:u~ u a QED=1
42:c~ c a QED=1
43:t~ t a QED=1
44:u~ u g QCD=1
45:c~ c g QCD=1
46:t~ t g QCD=1
47:t~ t h QED=1
48:u~ u z QED=1
49:c~ c z QED=1
50:t~ t z QED=1
51:e+ ve w- QED=1
52:mu+ vm w- QED=1
53:ta+ vt w- QED=1
54:ve~ ve z QED=1
55:vm~ vm z QED=1
56:vt~ vt z QED=1
(END)
```

(To get out of here write .q)

You can see the properties of a particle by writing **display particles e-**

```
MG5_aMC>display particles e-
Particle e- has the following properties:
{
    'name': 'e-',
    'antiname': 'e+',
    'spin': 2,
    'color': 1,
    'charge': -1.00,
    'mass': 'ZERO',
    'width': 'ZERO',
    'pdg_code': 11,
    'line': 'straight',
    'propagator': '',
    'is_part': True,
    'self_antipart': False,
    'type': '',
    'counterterm': {}
}
MG5_aMC>
```

Also, try to see the e- interaction, by **display interactions e-** : `MG5_aMC>display interactions e-`

```
Interactions 28 : e+ e- a  has the following property:
{
    'id': 34,
    'particles': [-11,11,22],
    'color': [1 ],
    'lorentz': ['FFV1'],
    'couplings': {(0, 0): 'GC_3'},
    'orders': {'QED': 1},
    'loop_particles': None,
    'type': 'base',
    'perturbation_type': None
}

Interactions 32 : e+ e- z  has the following property:
{
    'id': 40,
    'particles': [-11,11,23],
    'color': [1 ],
    'lorentz': ['FFV2', 'FFV4'],
    'couplings': {(0, 1): 'GC_59', (0, 0): '-GC_51'},
    'orders': {'QED': 1},
    'loop_particles': None,
    'type': 'base',
    'perturbation_type': None
}

Interactions 35 : ve~ e- w+  has the following property:
{
    'id': 43,
    'particles': [-12,11,24],
    'color': [1 ],
    'lorentz': ['FFV2'],
    'couplings': {(0, 0): 'GC_100'},
    'orders': {'QED': 1},
    'loop_particles': None,
    'type': 'base',
:
```

Default configuration

```
Using default text editor "vi". Set another one in ./input/mg5_configuration.txt
Using default eps viewer "evince". Set another one in ./input/mg5_configuration.txt
Using default web browser "firefox". Set another one in ./input/mg5_configuration.txt
Loading default model: sm
INFO: Restrict model sm with file ../models/sm/restrict_default.dat .
INFO: Run "set stdout_level DEBUG" before import for more information.
INFO: Change particles name to pass to MG5 convention
Defined multiparticle p = g u c d s u~ c~ d~ s~
Defined multiparticle j = g u c d s u~ c~ d~ s~
Defined multiparticle l+ = e+ mu+
Defined multiparticle l- = e- mu-
Defined multiparticle vl = ve vm vt
Defined multiparticle vl~ = ve~ vm~ vt~
Defined multiparticle all = g u c d s u~ c~ d~ s~ a ve vm vt e- mu- ve~ vm~ vt~ e+ mu+ t b t~ b~ z w+ h w- ta- ta+
MG5_aMC>
```

We can observe that, in the MadGraph default configuration, the notation means the following:

l+ = positive leptons	vl = leptonic neutrinos
l- = negative leptons	vl~= leptonic antineutrinos
p = parton	j = jet

all = All the particles of the corresponding model, in this case, the default model is SM.

Just as the multi-particle jet “j” means to equally sum over partons, gluons, quarks, and antiquarks, but in its final state. Similarly the “l-” means to sum over electrons and muons and “l+” over their antiparticles, “vl” and “vl~” are for all the neutrinos and antineutrinos.
We can modify the default configuration if we want. We can define new particles at your convenience, delete and/or add particles in the definitions showed above, for example:

define k = g u u~ c c~ d d~ s s~ b b~

k would define a multi-particle similar to j, but that also include the bottom and antibottom quarks.

```
MG5_aMC>define k = g u u~ c c~ d d~ s s~ b b~
Defined multiparticle k = g u c d s u~ c~ d~ s~ b b~
```

Example number 2: To define a multiparticle for the electro-weak bosons:

define v = w+ w- z a

```
MG5_aMC>define v = w+ w- z a
Defined multiparticle v = a w+ z w-
```

The multi-particle definitions can be used recursively, for example if we want to inlcude the quarks b in the jet definition we should write: **define j = j b b~**

```
MG5_aMC>define j = j b b~
Defined multiparticle j = g u c d s u~ c~ d~ s~ b b~
```

We can also change the default model (SM) for another one (BSM) . How can we do this?

Importing models

One of the principal utilities of the MadGraph program is to simulate BSM. MadGraph allows you to study hypothetical particles and their interactions of alternative models than SM, just like Super-Symmetries models. We will show this by importing MSSM.

Before running MadGraph we will copy the model's file which contains all about particle masses and coupling information. To do this, we have to be on the **madgraph265** directory and then change to the **models** directory.

```
clarissa@claudeth:~/madgraph265$ cd models
clarissa@claudeth:~/madgraph265/models$ ls
build_restriction_lib.py   check_param_card.pyo  import_ufo.pyo  loop_sm          model_reader.pyo  taudecay_UFO    write_param_card.py
build_restriction_lib.pyo  hgg_plugin            __init__.py     madgraph265      MSSM_SLHA2        template_files  write_param_card.pyo
check_param_card.py        import_ufo.py         __init__.pyo    model_reader.py  sm                usermod.py
```

If we check inside this directory we should have an MSSM_SLHA2 file (In some cases the SLHA2 termination could variate), this is the MSSM file that we have to copy. To make it possible we have to write the command **cp** to copy followed by **-rfvp,** the file´s name that we want to copy, space, and the file´s name where you want to leave the copy.

In my case I wrote: **cp -rfvp MSSM_SLHA2 madgraph265**

```
clarissa@claudeth:~/madgraph265/models$ cp -rfvp MSSM_SLHA2 madgraph265
'MSSM_SLHA2' -> 'madgraph265/MSSM_SLHA2'
'MSSM_SLHA2/write_param_card.py' -> 'madgraph265/MSSM_SLHA2/write_param_card.py'
'MSSM_SLHA2/MSSM_UFO.log' -> 'madgraph265/MSSM_SLHA2/MSSM_UFO.log'
'MSSM_SLHA2/couplings.py' -> 'madgraph265/MSSM_SLHA2/couplings.py'
'MSSM_SLHA2/lorentz.py' -> 'madgraph265/MSSM_SLHA2/lorentz.py'
'MSSM_SLHA2/__init__.py' -> 'madgraph265/MSSM_SLHA2/__init__.py'
'MSSM_SLHA2/particles.py' -> 'madgraph265/MSSM_SLHA2/particles.py'
'MSSM_SLHA2/object_library.py' -> 'madgraph265/MSSM_SLHA2/object_library.py'
'MSSM_SLHA2/decays.py' -> 'madgraph265/MSSM_SLHA2/decays.py'
'MSSM_SLHA2/restrict_default.dat' -> 'madgraph265/MSSM_SLHA2/restrict_default.dat'
'MSSM_SLHA2/restrict_no_b_mass.dat' -> 'madgraph265/MSSM_SLHA2/restrict_no_b_mass.dat'
'MSSM_SLHA2/function_library.py' -> 'madgraph265/MSSM_SLHA2/function_library.py'
'MSSM_SLHA2/parameters.py' -> 'madgraph265/MSSM_SLHA2/parameters.py'
'MSSM_SLHA2/coupling_orders.py' -> 'madgraph265/MSSM_SLHA2/coupling_orders.py'
'MSSM_SLHA2/restrict_no_tau_mass.dat' -> 'madgraph265/MSSM_SLHA2/restrict_no_tau_mass.dat'
'MSSM_SLHA2/vertices.py' -> 'madgraph265/MSSM_SLHA2/vertices.py'
'MSSM_SLHA2/restrict_no_masses.dat' -> 'madgraph265/MSSM_SLHA2/restrict_no_masses.dat'
'MSSM_SLHA2/build_restrict.py' -> 'madgraph265/MSSM_SLHA2/build_restrict.py'
'MSSM_SLHA2/__init__.pyo' -> 'madgraph265/MSSM_SLHA2/__init__.pyo'
'MSSM_SLHA2/particles.pyo' -> 'madgraph265/MSSM_SLHA2/particles.pyo'
'MSSM_SLHA2/object_library.pyo' -> 'madgraph265/MSSM_SLHA2/object_library.pyo'
'MSSM_SLHA2/parameters.pyo' -> 'madgraph265/MSSM_SLHA2/parameters.pyo'
'MSSM_SLHA2/function_library.pyo' -> 'madgraph265/MSSM_SLHA2/function_library.pyo'
'MSSM_SLHA2/couplings.pyo' -> 'madgraph265/MSSM_SLHA2/couplings.pyo'
'MSSM_SLHA2/lorentz.pyo' -> 'madgraph265/MSSM_SLHA2/lorentz.pyo'
'MSSM_SLHA2/vertices.pyo' -> 'madgraph265/MSSM_SLHA2/vertices.pyo'
'MSSM_SLHA2/coupling_orders.pyo' -> 'madgraph265/MSSM_SLHA2/coupling_orders.pyo'
'MSSM_SLHA2/write_param_card.pyo' -> 'madgraph265/MSSM_SLHA2/write_param_card.pyo'
'MSSM_SLHA2/decays.pyo' -> 'madgraph265/MSSM_SLHA2/decays.pyo'
'MSSM_SLHA2/build_restrict.pyo' -> 'madgraph265/MSSM_SLHA2/build_restrict.pyo'
'MSSM_SLHA2/model.pkl' -> 'madgraph265/MSSM_SLHA2/model.pkl'
```

Now we come back to the **madgraph265** directory (**cd ..**) and open MadGraph.
Once we have open it, write: **import model MSSM_SLHA2**

```
MG5_aMC>import model MSSM_SLHA2
INFO: load particles
INFO: load vertices
INFO: Restrict model MSSM_SLHA2 with file ../models/MSSM_SLHA2/restrict_default.dat .
INFO: Run "set stdout_level DEBUG" before import for more information.
INFO: Detect SLHA2 format. keeping restricted parameter in the param_card
INFO: Change particles name to pass to MG5 convention
Kept definitions of multiparticles l- / j / vl / l+ / p / vl~ unchanged
Defined multiparticle all = g u c d s u~ c~ d~ s~ a ve vm vt e- mu- ve~ vm~ vt~ e+ mu+ go ul cl t1 ur cr t2 dl sl b1 dr sr b2 ul~ cl~ t1~ ur~
cr~ t2~ dl~ sl~ b1~ dr~ sr~ b2~ t b t~ b~ z w+ h01 h2 h3 h+ sve svm svt el- mul- ta1- er- mur- ta2- w- h- sve~ svm~ svt~ el+ mul+ ta1+ er+ mur
+ ta2+ n1 n2 n3 n4 x1+ x2+ ta- x1- x2- ta+
MG5_aMC>
```

The above would replace the default SM (wrote as sm by MadGraph) .

There are many particles by MSSM. It is of common interest to change the super-symmetric particle masses. For example, the default gluino mass is of 607 GeV. The problem is that, by the moment, it seems like that we need to make an account on the Madgraph website to produce a “paramer_card.dat” (we will explain this card later) for this model. We can also modify the “paramer_card.dat” by default… but this is not recommended.

Generating processes

Naturally to generate any process on MadGraph we should write **generate** followed by the particles initial state , the greater than symbol (>), the particles final state. The process could be a decay or a scatter. For example if we write

generate p p > t t~

we generate a process where two partons collide to form a top quark and a antitop quark.

When we digit the process on the screen, it appears:

```
MG5_aMC>generate p p > t t~
INFO: Checking for minimal orders which gives processes.
INFO: Please specify coupling orders to bypass this step.
INFO: Trying coupling order WEIGHTED<=2: WEIGTHED IS 2*QED+QCD
INFO: Trying process: g g > t t~ WEIGHTED<=2 @1
INFO: Process has 3 diagrams
INFO: Trying process: u u~ > t t~ WEIGHTED<=2 @1
INFO: Process has 1 diagrams
INFO: Trying process: u c~ > t t~ WEIGHTED<=2 @1
INFO: Trying process: c u~ > t t~ WEIGHTED<=2 @1
INFO: Trying process: c c~ > t t~ WEIGHTED<=2 @1
INFO: Process has 1 diagrams
INFO: Trying process: d d~ > t t~ WEIGHTED<=2 @1
INFO: Process has 1 diagrams
INFO: Trying process: d s~ > t t~ WEIGHTED<=2 @1
INFO: Trying process: s d~ > t t~ WEIGHTED<=2 @1
INFO: Trying process: s s~ > t t~ WEIGHTED<=2 @1
INFO: Process has 1 diagrams
INFO: Process u~ u > t t~ added to mirror process u u~ > t t~
INFO: Process c~ c > t t~ added to mirror process c c~ > t t~
INFO: Process d~ d > t t~ added to mirror process d d~ > t t~
INFO: Process s~ s > t t~ added to mirror process s s~ > t t~
5 processes with 7 diagrams generated in 0.074 s
Total: 5 processes with 7 diagrams
MG5_aMC>
```

Hence the parton multi-particle definition, MadGraph product us the different types of particles on the parton definition (g u d c s g~ u~ d~ c~ s~), therefore it realize all these possible processes. In each process, we note that it is written “WEIGHTED”, which is related to the probability weight of the corresponding process.

At the bottom of the screen, we can observe that we obtain 5 processes with 7 diagrams. How can we see those diagrams?

Feynman Diagrams

To see the corresponding diagrams, we just have to write: **display diagrams**. In the above process the diagrams that we obtain are:

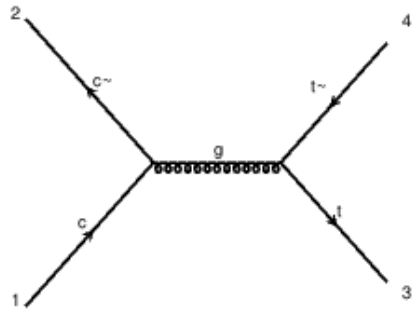


diagram 1 QCD=2, QED=0

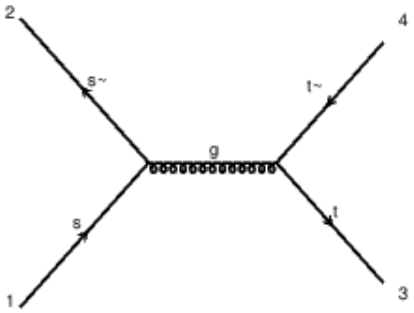


diagram 1 QCD=2, QED=0

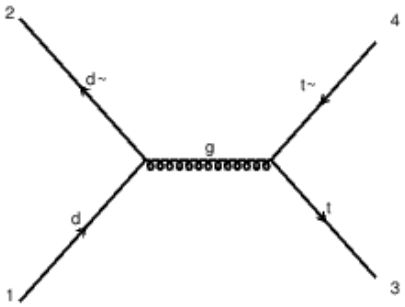


diagram 1 QCD=2, QED=0

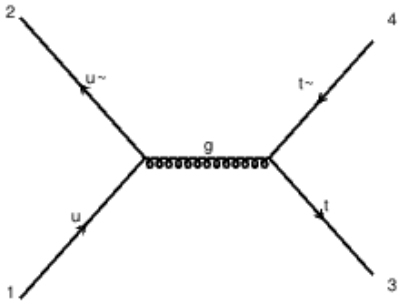


diagram 1 QCD=2, QED=0

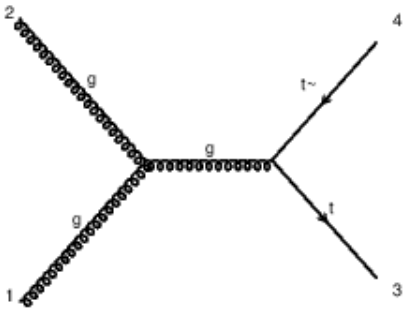


diagram 1 QCD=2, QED=0

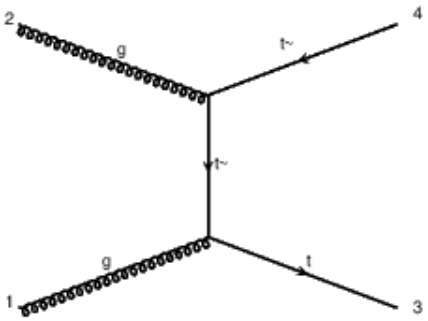


diagram 2 QCD=2, QED=0

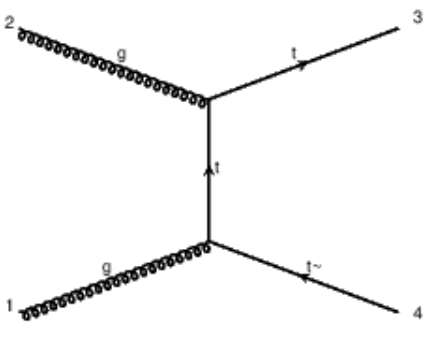
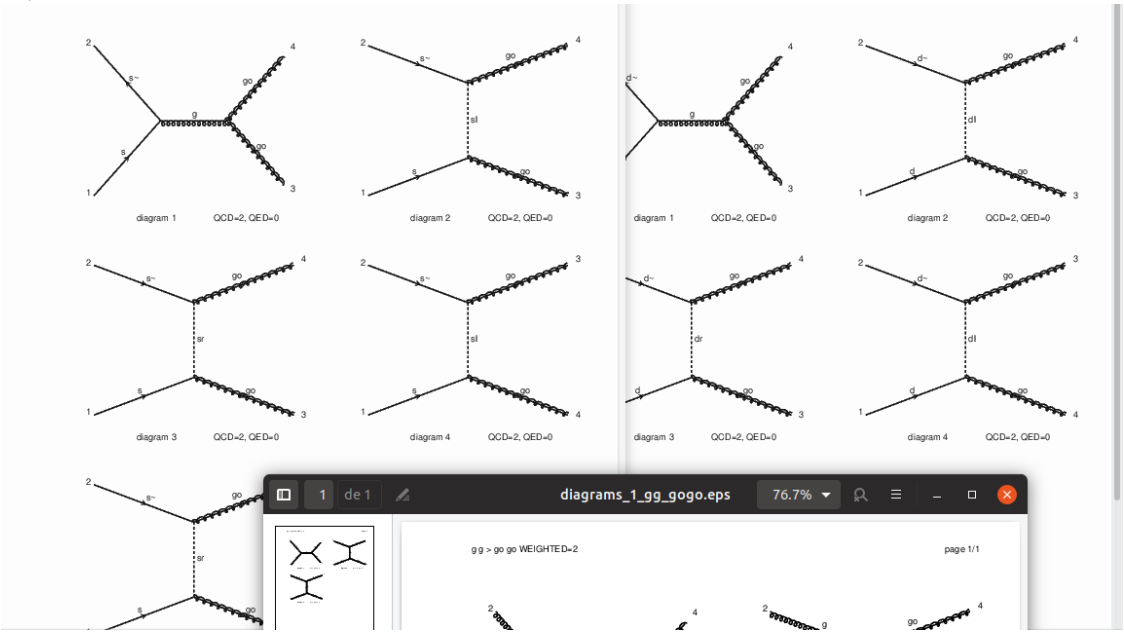


diagram 3 QCD=2, QED=0

We can make another example related to MSSM. In super-symmetric theory exist a particle named gluino, denotated as “go” by Madgraph. Thus, for example, if we work on MSSM to simulate a gluino pair production, we should write: **generate p p > go go**.

```
MG5_aMC>generate p p > go go
INFO: Checking for minimal orders which gives processes.
INFO: Please specify coupling orders to bypass this step.
INFO: Trying coupling order WEIGHTED<=2: WEIGTHED IS 2*QED+QCD
INFO: Trying process: g g > go go WEIGHTED<=2 @1
INFO: Process has 3 diagrams
INFO: Trying process: u u~ > go go WEIGHTED<=2 @1
INFO: Process has 5 diagrams
INFO: Trying process: u c~ > go go WEIGHTED<=2 @1
INFO: Trying process: c u~ > go go WEIGHTED<=2 @1
INFO: Trying process: c c~ > go go WEIGHTED<=2 @1
INFO: Process has 5 diagrams
INFO: Trying process: d d~ > go go WEIGHTED<=2 @1
INFO: Process has 5 diagrams
INFO: Trying process: d s~ > go go WEIGHTED<=2 @1
INFO: Trying process: s d~ > go go WEIGHTED<=2 @1
INFO: Trying process: s s~ > go go WEIGHTED<=2 @1
INFO: Process has 5 diagrams
INFO: Process u~ u > go go added to mirror process u u~ > go go
INFO: Process c~ c > go go added to mirror process c c~ > go go
INFO: Process d~ d > go go added to mirror process d d~ > go go
INFO: Process s~ s > go go added to mirror process s s~ > go go
5 processes with 23 diagrams generated in 0.221 s
Total: 5 processes with 23 diagrams
```

Followed by **display diagrams**. **Some** of the obtained diagrams (because there are 23 diagrams!) are:



One last example. Suppose that we want to generate an electron-positron pair production.

```
MG5_aMC>generate  p p > e+ e-
INFO: Checking for minimal orders which gives processes.
INFO: Please specify coupling orders to bypass this step.
INFO: Trying process: g g > e+ e- WEIGHTED<=4 @1
INFO: Trying process: u u~ > e+ e- WEIGHTED<=4 @1
INFO: Process has 2 diagrams
INFO: Trying process: u c~ > e+ e- WEIGHTED<=4 @1
INFO: Trying process: c u~ > e+ e- WEIGHTED<=4 @1
INFO: Trying process: c c~ > e+ e- WEIGHTED<=4 @1
INFO: Process has 2 diagrams
INFO: Trying process: d d~ > e+ e- WEIGHTED<=4 @1
INFO: Process has 2 diagrams
INFO: Trying process: d s~ > e+ e- WEIGHTED<=4 @1
INFO: Trying process: s d~ > e+ e- WEIGHTED<=4 @1
INFO: Trying process: s s~ > e+ e- WEIGHTED<=4 @1
INFO: Process has 2 diagrams
INFO: Process u~ u > e+ e- added to mirror process u u~ > e+ e-
INFO: Process c~ c > e+ e- added to mirror process c c~ > e+ e-
INFO: Process d~ d > e+ e- added to mirror process d d~ > e+ e-
INFO: Process s~ s > e+ e- added to mirror process s s~ > e+ e-
4 processes with 8 diagrams generated in 0.062 s
Total: 4 processes with 8 diagrams
```

As we can observe, it is common to obtain various Feynman diagrams on the processes, but, What happens if we want to exclude some diagrams that contain a specific particle?

For example, in the process above, suppose that for any reason we would like to exclude all the diagrams that contain Z boson. How should we resolve the problem?

Including and Excluding Feynman Diagrams

To resolve the problem above, we should write:

generate p p > e+ e- / Z

The **/Z** command means to exclude the Feynman diagrams that contain Z bosons.

```
MG5_aMC>generate  p p > e+ e- /Z
INFO: Checking for minimal orders which gives processes.
INFO: Please specify coupling orders to bypass this step.
INFO: Trying process: g g > e+ e- WEIGHTED<=4 / z @1
INFO: Trying process: u u~ > e+ e- WEIGHTED<=4 / z @1
INFO: Process has 1 diagrams
INFO: Trying process: u c~ > e+ e- WEIGHTED<=4 / z @1
INFO: Trying process: c u~ > e+ e- WEIGHTED<=4 / z @1
INFO: Trying process: c c~ > e+ e- WEIGHTED<=4 / z @1
INFO: Process has 1 diagrams
INFO: Trying process: d d~ > e+ e- WEIGHTED<=4 / z @1
INFO: Process has 1 diagrams
INFO: Trying process: d s~ > e+ e- WEIGHTED<=4 / z @1
INFO: Trying process: s d~ > e+ e- WEIGHTED<=4 / z @1
INFO: Trying process: s s~ > e+ e- WEIGHTED<=4 / z @1
INFO: Process has 1 diagrams
INFO: Process u~ u > e+ e- added to mirror process u u~ > e+ e-
INFO: Process c~ c > e+ e- added to mirror process c c~ > e+ e-
INFO: Process d~ d > e+ e- added to mirror process d d~ > e+ e-
INFO: Process s~ s > e+ e- added to mirror process s s~ > e+ e-
4 processes with 4 diagrams generated in 0.066 s
Total: 4 processes with 4 diagrams
MG5_aMC>
```

As we can observe, now our result is 4 processes with 4 diagrams, without **/Z** command we have obtained 4 processes with 8 diagrams.

We can also exclude a particle on the s-channel, writting:

generate p p > e+ e- $ Z

Finally, we can include only diagrams where appears a certain particle on the s-channel, for example, writting:

generate p p > Z > e+ e-

Each one of these should be carefully used because the Gauge invariance and unitary might get violated if we are careless.

Adding and jet process

If we write generate two times, then, Madgraph would overwrite the second generate with the first. Therefore, if we want to combinate processes in the same event sample, we should write:

add process

before the next process you want to write.

For example:

generate p p > u u~

```
5 processes with 8 diagrams generated in 0.038 s
Total: 5 processes with 8 diagrams
MG5_aMC>
```

add process p p > d d~

```
5 processes with 8 diagrams generated in 0.064 s
Total: 10 processes with 16 diagrams
MG5_aMC>
```

add process p p > s s~

```
5 processes with 8 diagrams generated in 0.077 s
Total: 15 processes with 24 diagrams
MG5_aMC>
```

add process p p > c c~

```
5 processes with 8 diagrams generated in 0.097 s
Total: 20 processes with 32 diagrams
MG5_aMC>
```

add process p p > g g

```
5 processes with 16 diagrams generated in 0.066 s
Total: 25 processes with 48 diagrams
MG5_aMC>
```

Is possible that you need to do things like:

p p > j j

```
65 processes with 112 diagrams generated in 0.449 s
Total: 65 processes with 112 diagrams
```

Also, you can generate all the 3 jets events:

generate p p > j j j

```
97 processes with 781 diagrams generated in 1.910 s
Total: 97 processes with 781 diagrams
MG5_aMC>
```

etc. Try not to generate 4, 5 or more events, MadGraph will take a time to run,

How to specify decays

Maybe we are interested in the process where the final product has some specific properties. For example, suppose that we want to produce one W boson in a proton-proton collision.

generate p p > W+

```
MG5_aMC>generate p p > W+
INFO: Checking for minimal orders which gives processes.
INFO: Please specify coupling orders to bypass this step.
INFO: Trying process: u d~ > w+ WEIGHTED<=2 @1
INFO: Process has 1 diagrams
INFO: Trying process: u s~ > w+ WEIGHTED<=2 @1
INFO: Trying process: c d~ > w+ WEIGHTED<=2 @1
INFO: Trying process: c s~ > w+ WEIGHTED<=2 @1
INFO: Process has 1 diagrams
INFO: Process d~ u > w+ added to mirror process u d~ > w+
INFO: Process s~ c > w+ added to mirror process c s~ > w+
2 processes with 2 diagrams generated in 0.039 s
Total: 2 processes with 2 diagrams
```

But we are interested in the events where the W boson decays on a muon, an antimuon, and a muonic neutrino. The command should be: **generate p p > W+, W+ > mu+ vm**

```
MG5_aMC>generate p p > W+, W+ > mu+ vm
INFO: Checking for minimal orders which gives processes.
INFO: Please specify coupling orders to bypass this step.
INFO: Trying process: u d~ > w+ WEIGHTED<=2 @1
INFO: Process has 1 diagrams
INFO: Trying process: u s~ > w+ WEIGHTED<=2 @1
INFO: Trying process: c d~ > w+ WEIGHTED<=2 @1
INFO: Trying process: c s~ > w+ WEIGHTED<=2 @1
INFO: Process has 1 diagrams
INFO: Process d~ u > w+ added to mirror process u d~ > w+
INFO: Process s~ c > w+ added to mirror process c s~ > w+
INFO: Checking for minimal orders which gives processes.
INFO: Please specify coupling orders to bypass this step.
INFO: Trying process: w+ > mu+ vm WEIGHTED<=2
INFO: Process has 1 diagrams
1 processes with 3 diagrams generated in 0.039 s
Total: 1 processes with 3 diagrams
MG5_aMC>
```

The above tells MadGraph to produce only the cross-section of the specific W decaying on múons events. If you prefer to do these things using multiparticles, then, note that it has to be only to the final particle of the decay)

Hence, for example, we can generate an event that produces a jet and a W boson, then we want to specify that the jet decay on leptons and a leptonic neutrino.

generate p p > W+ j, j > l+ vl

Usually, jets produce many processes, and we don´t want to waste time producing events that finally we won´t use, so the above is very convenient.

What happens if I want to concentrate on two specific decays? Well, we can solve this by using parenthesis. For example:

generate p p > t t~, (t > W+ b, W+ > j j), (t~ > W- b~, W- > l- vl~)

The above process, is a chain decay process, that involves a top-antitop pair production, keeping the events where the top quark decay to a W boson and a bottom quark, and where the antitop decay yo a W and bottom, where W have a leptonic decay.

Finally, we´ll present the last example of this topic: Higgs production in association with a Z boson decaying to neutrinos, with the Higgs decaying to four leptons is:

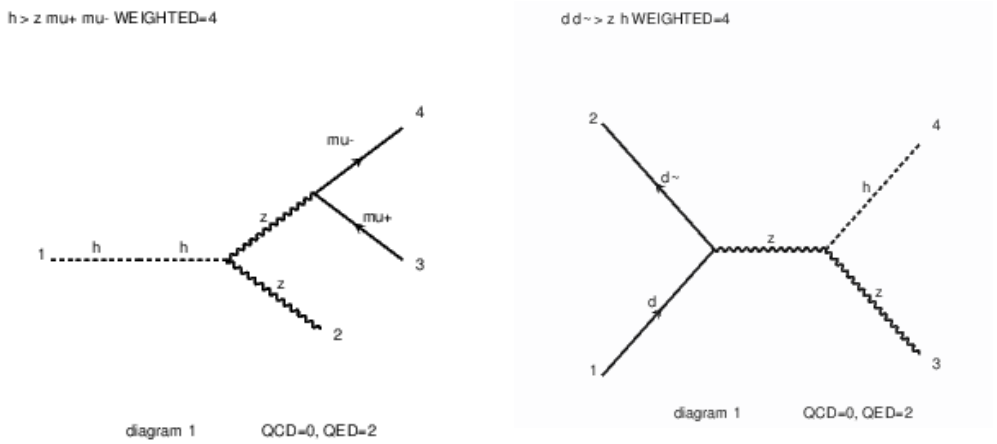
generate p p > z h, z > vl vl~, (h > z l+ l-, z > l+ l-)

For more fun you could add:

add process p p > w+ h, w+ > l+ vl, (h > z l+ l-, z > l+ l-)
add process p p > w- h, w- > l- vl~, (h > z l+ l-, z > l+ l-)

```
1 processes with 8 diagrams generated in 0.038 s
Total: 3 processes with 27 diagrams
MG5_aMC>
```

Here we have two of the 27 diagrams:



But wait! We don´t want to just throw up all these complex processes. How we could save it?

Output and Launch

Suppose that we generate the following process:

generate p p > t t~
add process p p > b b~

```
MG5_aMC>add process p p > b b~
INFO: Checking for minimal orders which gives processes.
INFO: Please specify coupling orders to bypass this step.
INFO: Trying coupling order WEIGHTED<=2: WEIGTHED IS 2*QED+QCD
INFO: Trying process: g g > b b~ WEIGHTED<=2 @2
INFO: Process has 3 diagrams
INFO: Trying process: u u~ > b b~ WEIGHTED<=2 @2
INFO: Process has 1 diagrams
INFO: Trying process: u c~ > b b~ WEIGHTED<=2 @2
INFO: Trying process: c u~ > b b~ WEIGHTED<=2 @2
INFO: Trying process: c c~ > b b~ WEIGHTED<=2 @2
INFO: Process has 1 diagrams
INFO: Trying process: d d~ > b b~ WEIGHTED<=2 @2
INFO: Process has 1 diagrams
INFO: Trying process: d s~ > b b~ WEIGHTED<=2 @2
INFO: Trying process: s d~ > b b~ WEIGHTED<=2 @2
INFO: Trying process: s s~ > b b~ WEIGHTED<=2 @2
INFO: Process has 1 diagrams
INFO: Process u~ u > b b~ added to mirror process u u~ > b b~
INFO: Process c~ c > b b~ added to mirror process c c~ > b b~
INFO: Process d~ d > b b~ added to mirror process d d~ > b b~
INFO: Process s~ s > b b~ added to mirror process s s~ > b b~
5 processes with 7 diagrams generated in 0.042 s
Total: 10 processes with 14 diagrams
MG5_aMC>
```

To output the process we write:

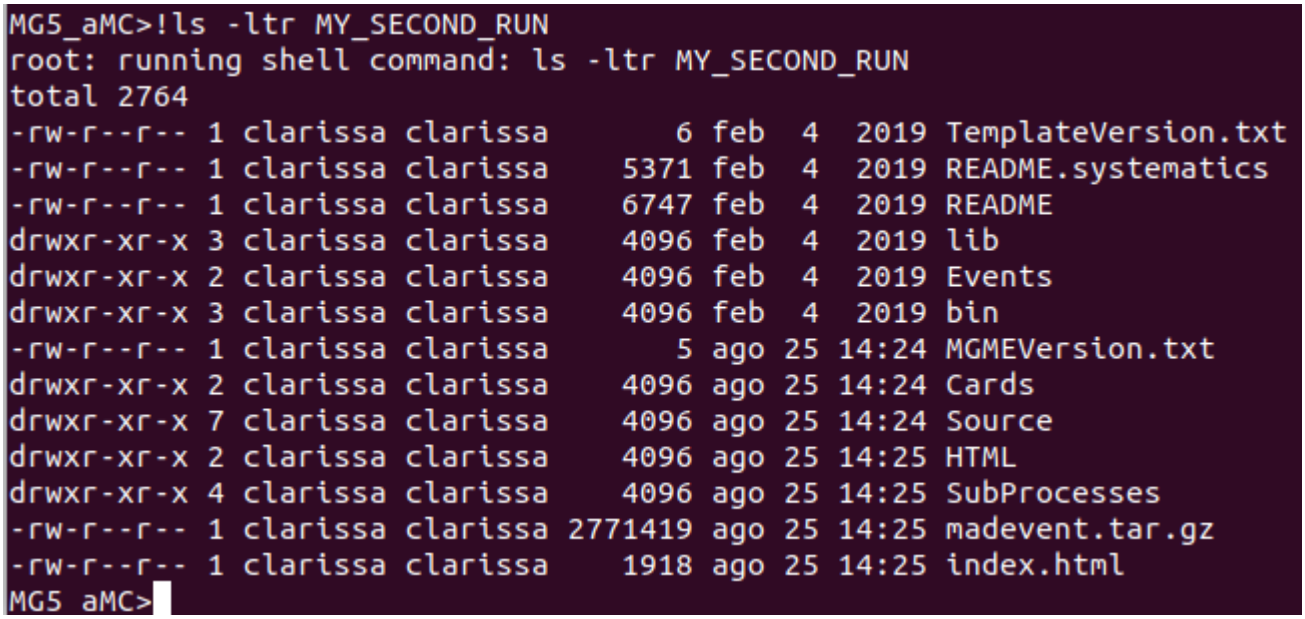
output MY_SECOND_RUN

```
MG5_aMC>output MY_SECOND_RUN
INFO: initialize a new directory: MY_SECOND_RUN
INFO: remove old information in MY_SECOND_RUN
INFO: Organizing processes into subprocess groups
INFO: Generating Helas calls for process: g g > t t~ WEIGHTED<=2
 @1
INFO: Processing color information for process: g g > t t~ @1
INFO: Generating Helas calls for process: g g > b b~ WEIGHTED<=2
 @2
INFO: Processing color information for process: g g > b b~ @2
INFO: Generating Helas calls for process: u u~ > t t~ WEIGHTED<=
2 @1
INFO: Processing color information for process: u u~ > t t~ @1
INFO: Combined process c c~ > t t~ WEIGHTED<=2 @1 with process u
 u~ > t t~ WEIGHTED<=2 @1
INFO: Combined process d d~ > t t~ WEIGHTED<=2 @1 with process u
 u~ > t t~ WEIGHTED<=2 @1
INFO: Combined process s s~ > t t~ WEIGHTED<=2 @1 with process u
 u~ > t t~ WEIGHTED<=2 @1
INFO: Generating Helas calls for process: u u~ > b b~ WEIGHTED<=
2 @2
INFO: Processing color information for process: u u~ > b b~ @2
INFO: Combined process c c~ > b b~ WEIGHTED<=2 @2 with process u
 u~ > b b~ WEIGHTED<=2 @2
INFO: Combined process d d~ > b b~ WEIGHTED<=2 @2 with process u
 u~ > b b~ WEIGHTED<=2 @2
INFO: Combined process s s~ > b b~ WEIGHTED<=2 @2 with process u
 u~ > b b~ WEIGHTED<=2 @2
INFO: Creating files in directory P1_gg_ttx
INFO: Generating Feynman diagrams for Process: g g > t t~ WEIGHT
ED<=2 @1
INFO: Finding symmetric diagrams for subprocess group gg_ttx
INFO: Creating files in directory P2_gg_bbx
INFO: Generating Feynman diagrams for Process: g g > b b~ WEIGHT
ED<=2 @2
INFO: Finding symmetric diagrams for subprocess group gg_bbx
INFO: Creating files in directory P1_qq_ttx
INFO: Generating Feynman diagrams for Process: u u~ > t t~ WEIGH
TED<=2 @1
INFO: Finding symmetric diagrams for subprocess group qq_ttx
INFO: Creating files in directory P2_qq_bbx
INFO: Generating Feynman diagrams for Process: u u~ > b b~ WEIGH
TED<=2 @2
INFO: Finding symmetric diagrams for subprocess group qq_bbx
Generated helas calls for 4 subprocesses (8 diagrams) in 0.025 s
Wrote files for 32 helas calls in 0.113 s
ALOHA: aloha creates FFV1 routines
ALOHA: aloha creates VVV1 set of routines with options: P0
save configuration file to /home/clarissa/madgraph265/bin/MY_SEC
OND_RUN/Cards/me5_configuration.txt
INFO: Use Fortran compiler gfortran
INFO: Use c++ compiler g++
INFO: Generate jpeg diagrams
INFO: Generate web pages
Output to directory /home/clarissa/madgraph265/bin/MY_SECOND_RUN
 done.
Type "launch" to generate events from this process, or see
/home/clarissa/madgraph265/bin/MY_SECOND_RUN/README
Run "open index.html" to see more information about this process
.
MG5_aMC>
```

You can check it by typing:

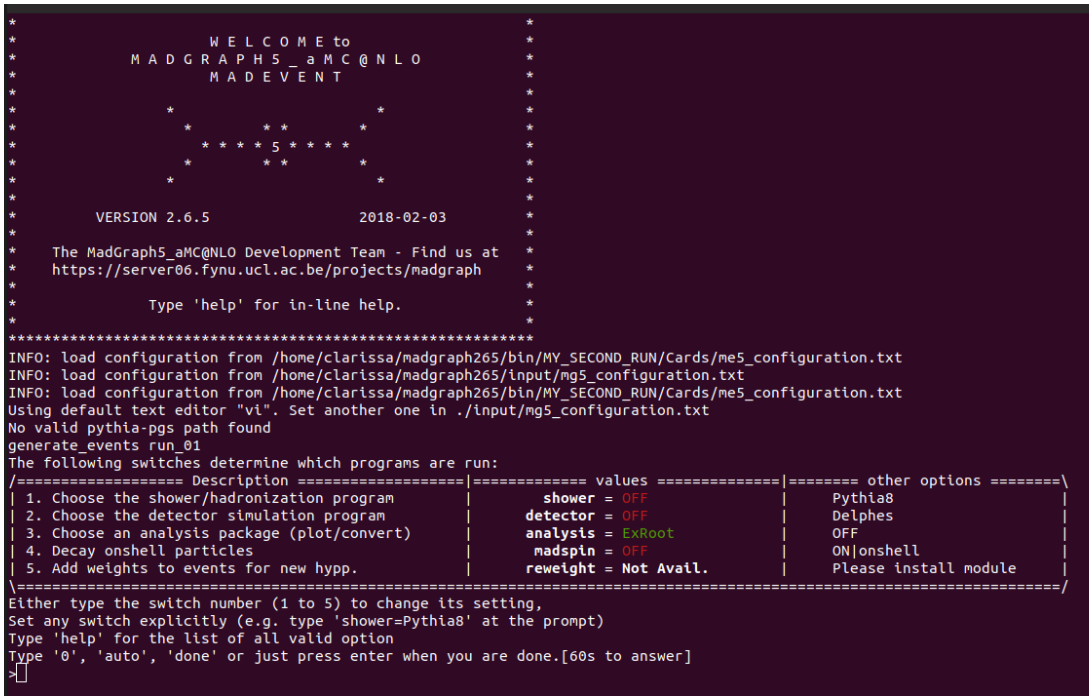
!ls -ltr MY_SECOND_RUN

The command **!** allows you to run commands outside of Madgraph.



To launch it:

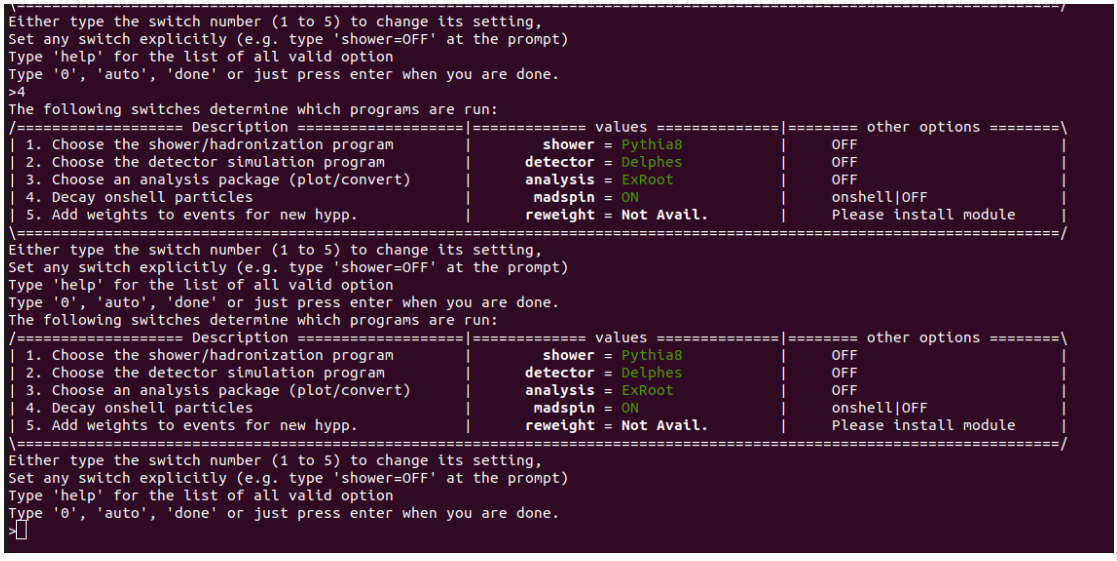
launch MY_SECOND_RUN



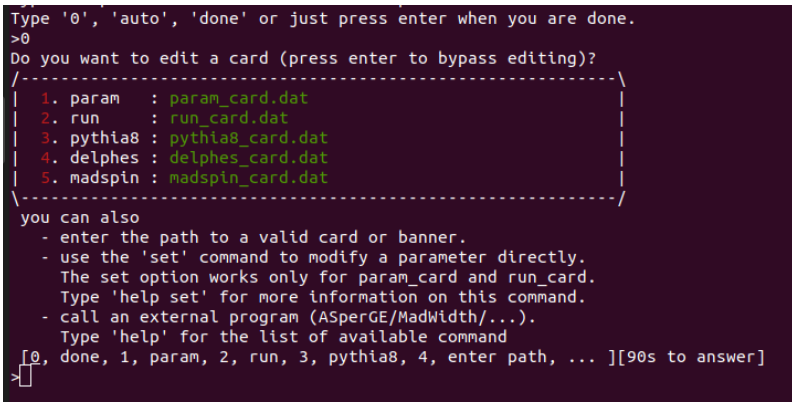
We'll choose to run Pythia, PGS, and Delphes. When we're done turning on our choice of switches 1, 2, 4, (we won´t choose 3 because it is already on, as you see on thee image above) then we choose 0 to say that we're ready to proceed. So we type:

1
2
4
0

The resulting events should be correctly weighted according to their cross sections.

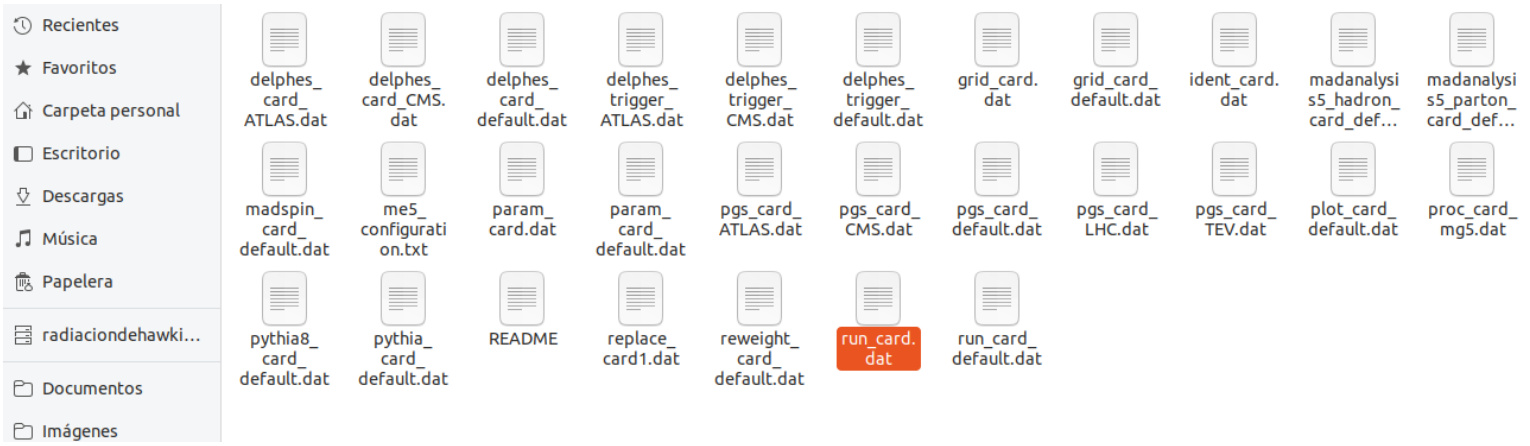


Now, MadGraph will give you the option of, either, edit, run or plot, the parameter card.



We are just concentrating on the cross-section, hence we need a quick run. So we would edit the parameter card just to ask for 1000 events.

To edit it, we should go to:



We open:

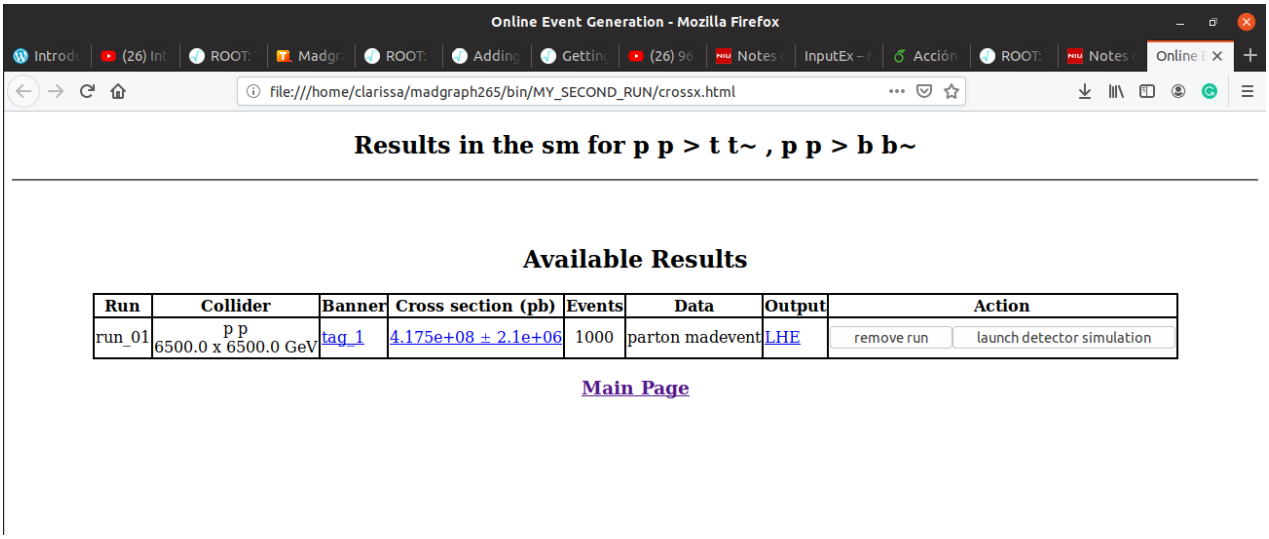
```
#*********************************************************************
#                      MadGraph5_aMC@NLO                             *
#                                                                    *
#                    run_card.dat MadEvent                           *
#                                                                    *
#  This file is used to set the parameters of the run.               *
#                                                                    *
#  Some notation/conventions:                                        *
#                                                                    *
#   Lines starting with a '# ' are info or comments                  *
#                                                                    *
#   mind the format:   value    = variable     ! comment             *
#                                                                    *
#   To display more options, you can type the command:               *
#      update full_run_card                                          *
#*********************************************************************
#
#*******************
# Running parameters
#*******************
#
#*********************************************************************
# Tag name for the run (one word)                                    *
#*********************************************************************
  tag_1     = run_tag ! name of the run
#*********************************************************************
# Number of events and rnd seed                                      *
# Warning: Do not generate more than 1M events in a single run       *
# If you want to run Pythia, avoid more than 50k events in a run.    *
#*********************************************************************
  10000 = nevents ! Number of unweighted events requested
  0   = iseed   ! rnd seed (0=assigned automatically=default))
#*********************************************************************
# Collider type and energy                                           *
# lpp: 0=No PDF, 1=proton, -1=antiproton, 2=photon from proton,      *
#                                         3=photon from electron     *
#*********************************************************************
```

Then change “10000” to “1000” on line 31.

```
#*********************************************************************
# Number of events and rnd seed                                      *
# Warning: Do not generate more than 1M events in a single run       *
# If you want to run Pythia, avoid more than 50k events in a run.    *
#*********************************************************************
  1000 = nevents ! Number of unweighted events requested
  0   = iseed   ! rnd seed (0=assigned automatically=default))
#*********************************************************************
# Collider type and energy                                           *
# lpp: 0=No PDF, 1=proton, -1=antiproton, 2=photon from proton,      *
#                                         3=photon from electron     *
```

Save the file and write **0** on the terminal to start the run.

When it runs, it shows you the following:




```
>0
INFO: Update the dependent parameter of the param_card.dat
Generating 1000 events with run name run_01
survey  run_01
INFO: compile directory
compile Source Directory
Using random number seed offset = 21
INFO: Running Survey
Creating Jobs
Working on SubProcesses
INFO:     P1_gg_ttx
INFO:     P1_qq_ttx
INFO:  Idle: 1,  Running: 0,  Completed: 1 [ current time: 14h33 ]
INFO:  Idle: 0,  Running: 1,  Completed: 1 [ current time: 14h33 ]
INFO:  Idle: 0,  Running: 0,  Completed: 2 [  0.29s  ]
INFO: End survey
refine 1000
Creating Jobs
```

```
INFO: Refine results to 1000
INFO: Generating 1000.0 unweigthed events.
INFO: Effective Luminosity 2.38408385618 pb^-1
INFO: need to improve 1 channels
Current estimate of cross-section: 503.338 +- 4.22480848987
    P1_gg_ttx
    P1_qq_ttx
INFO:  Idle: 0,  Running: 1,  Completed: 0 [ current time: 14h34 ]
INFO:  Idle: 0,  Running: 0,  Completed: 1 [  0.6s  ]
INFO: Combining runs
INFO: finish refine
refine 1000
Creating Jobs
INFO: Refine results to 1000
INFO: Generating 1000.0 unweigthed events.
INFO: Effective Luminosity 2.38285298988 pb^-1
INFO: need to improve 0 channels
Current estimate of cross-section: 503.598 +- 2.15408589255
    P1_gg_ttx
    P1_qq_ttx
INFO:  Idle: 0,  Running: 0,  Completed: 0 [ current time: 14h34 ]
```

```
INFO:  Idle: 0,  Running: 0,  Completed: 0 [ current time: 14h34 ]
INFO: Combining runs
INFO: finish refine
INFO: Combining Events
  === Results Summary for run: run_01 tag: tag_1 ===

     Cross-section :   503.6 +- 2.154 pb
     Nb of events :  1000

fail
Failed to access python version of LHAPDF: If the python interface to LHAPDF is available on your system, try adding its location to the PYTHO
NPATH environment variable and theLHAPDF library location to LD_LIBRARY_PATH (linux) or DYLD_LIBRARY_PATH (mac os x).
INFO: can not run systematics since can not link python to lhapdf
store_events
INFO: Storing parton level results
INFO: End Parton
reweight -from_cards
decay_events -from_cards
INFO: Running MadSpin
INFO: This functionality allows for the decay of resonances
INFO: in a .lhe file, keeping track of the spin correlation effets.
INFO: BE AWARE OF THE CURRENT LIMITATIONS:
INFO:   (1) Only a succession of 2 body decay are currently allowed
```

```
************************************************************
*                                                          *
*           W E L C O M E  to  M A D S P I N               *
*                                                          *
************************************************************
INFO: Extracting the banner ...
INFO: process: p p > t t~
INFO: options:
INFO: detected model: sm. Loading...
Set group_subprocesses to Auto
Note that you need to regenerate all processes
save options collier
save configuration file to /home/clarissa/madgraph265/input/mg5_configuration.txt
save options auto_update
save configuration file to /home/clarissa/madgraph265/input/mg5_configuration.txt
save options pythia8_path
save configuration file to /home/clarissa/madgraph265/input/mg5_configuration.txt
save options delphes_path
save configuration file to /home/clarissa/madgraph265/input/mg5_configuration.txt
save options cluster_type
save configuration file to /home/clarissa/madgraph265/input/mg5_configuration.txt
save options exrootanalysis_path
save configuration file to /home/clarissa/madgraph265/input/mg5_configuration.txt
save options OLP
save configuration file to /home/clarissa/madgraph265/input/mg5_configuration.txt
save options applgrid
save configuration file to /home/clarissa/madgraph265/input/mg5_configuration.txt
set lhapdf to /home/clarissa/madgraph265/HEPTools/lhapdf6/bin/lhapdf-config
save options lhapdf
save configuration file to /home/clarissa/madgraph265/input/mg5_configuration.txt
save options ninja
save configuration file to /home/clarissa/madgraph265/input/mg5_configuration.txt
save options amcfast
save configuration file to /home/clarissa/madgraph265/input/mg5_configuration.txt
save options output_dependencies
save configuration file to /home/clarissa/madgraph265/input/mg5_configuration.txt
save options mg5amc_py8_interface_path
save configuration file to /home/clarissa/madgraph265/input/mg5_configuration.txt
set max_weight_ps_point 400  # number of PS to estimate the maximum for each event
```

```
decay t > w+ b, w+ > all all
decay t~ > w- b~, w- > all all
decay w+ > all all
decay w- > all all
decay z > all all
launch
INFO: Will use seed 9095761
INFO: We need to recalculate the branching fractions for t~,w-,z,w+,t
INFO: using the FeynRules formula present in the model (arXiv:1402.1178)
INFO: Restrict model /home/clarissa/madgraph265/models/sm with file ../models/sm/restrict_default.dat
INFO: Run "set stdout_level DEBUG" before import for more information.
INFO:
INFO: decay channels for t~ : ( width = 1.491472 GeV )
INFO:        BR                d1  d2
INFO:    1.000000e+00           b~  w-
INFO:
INFO:
INFO: decay channels for w- : ( width = 2.04791 GeV )
INFO:        BR                d1  d2
INFO:    3.333605e-01           s  c~
INFO:    3.333605e-01           d  u~
INFO:    1.111202e-01           mu-  vm~
INFO:    1.111202e-01           e-  ve~
INFO:    1.110388e-01           ta-  vt~
INFO:
INFO:
INFO: decay channels for z : ( width = 2.441755 GeV )
INFO:        BR                d1  d2
INFO:    1.523651e-01           s~  s
INFO:    1.523651e-01           d~  d
INFO:    1.507430e-01           b~  b
INFO:    1.188151e-01           c~  c
INFO:    1.188151e-01           u~  u
INFO:    6.793735e-02           vt~  vt
INFO:    6.793735e-02           vm~  vm
INFO:    6.793735e-02           ve~  ve
INFO:    3.438731e-02           mu+  mu-
INFO:    3.438731e-02           e+  e-
INFO:    3.430994e-02           ta+  ta-
```

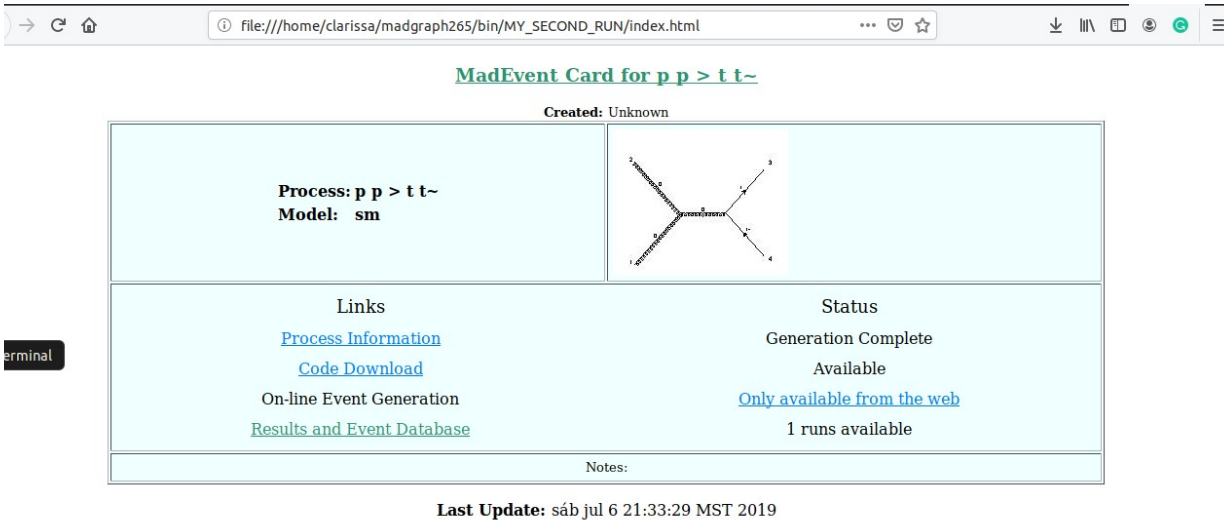
```
INFO: decay channels for w+ : ( width = 2.04791 GeV )
INFO:        BR                d1  d2
INFO:    3.333605e-01           s~  c
INFO:    3.333605e-01           d~  u
INFO:    1.111202e-01           mu+  vm
INFO:    1.111202e-01           e+  ve
INFO:    1.110388e-01           ta+  vt
INFO:
INFO:
INFO: decay channels for t : ( width = 1.491472 GeV )
INFO:        BR                d1  d2
INFO:    1.000000e+00           b  w+
INFO:
INFO: generating the production square matrix element
INFO: generate p p > t t~;
INFO: Done 2.415
INFO: generating the full matrix element squared (with decay)
INFO: generate p p > t t~, (t~ > b~ w- , w- > all all  QCD=99), (t > b w+ , w+ > all all  QCD=99)  --no_warning=duplicate;
INFO: Done 5.341
INFO: generate matrix element for decay only (1 - > N).
INFO: output standalone_msF /home/clarissa/madgraph265/bin/MY_SECOND_RUN/decay_me
INFO: Done 2.249
INFO: Compiling code
INFO: detect independant decays
INFO: Done in 0.146538972855s
INFO:
INFO:    Estimating the maximum weight
INFO:    ****************************
INFO:      Probing the first 75 events
INFO:      with 400 phase space points
INFO:
INFO: Event 1/75 :  0.23s
INFO: Event 6/75 :  1.3s
INFO: Event 11/75 :  2.2s
INFO: Event 16/75 :  3.1s
INFO: Event 21/75 :  4s
INFO: Event 26/75 :  4.8s
INFO: Event 31/75 :  5.8s
INFO: Event 36/75 :  6.6s
```

```
INFO: Event 41/75 :  7.5s
INFO: Event 46/75 :  8.2s
INFO: Event 51/75 :  9s
INFO: Event 56/75 :  9.8s
INFO: Event 61/75 :  10.6s
INFO: Event 66/75 :  11.6s
INFO: Event 71/75 :  12.5s
INFO:
INFO: Decaying the events...
INFO: Total number of events written: 1000/1000
INFO: Average number of trial points per production event: 5.757
INFO: Branching ratio to allowed decays: 1
INFO: Number of events with weights larger than max_weight: 0
INFO: Number of subprocesses 7
INFO: Number of failures when restoring the Monte Carlo masses: 0
INFO: Decayed events have been written in /home/clarissa/madgraph265/bin/MY_SECOND_RUN/Events/run_01/unweighted_events_decayed.lhe.gz
INFO: The decayed event file has been moved to the following location:
INFO: /home/clarissa/madgraph265/bin/MY_SECOND_RUN/Events/run_01_decayed_1/unweighted_events.lhe.gz
INFO: MadSpin Done
INFO: Running Pythia8 [arXiv:1410.3012]
Splitting .lhe event file for PY8 parallelization...
Submitting Pythia8 jobs...
Pythia8 shower jobs: 1 Idle, 3 Running, 0 Done [3 seconds]
Pythia8 shower jobs: 0 Idle, 3 Running, 1 Done [1m02s]
Pythia8 shower jobs: 0 Idle, 2 Running, 2 Done [1m03s]
Pythia8 shower jobs: 0 Idle, 1 Running, 3 Done [1m04s]
Pythia8 shower jobs: 0 Idle, 0 Running, 4 Done [1m04s]
Merging results from the split PY8 runs...
INFO: Pythia8 shower finished after 1m10s.
INFO: prepare delphes run
INFO: Running Delphes
INFO: If you are interested in lhco output. please run root2lhco converter.
INFO:  or edit bin/internal/run_delphes3 to run the converter automatically.
INFO: delphes done
  === Results Summary for run: run_01_decayed_1 tag: tag_1 ===
```

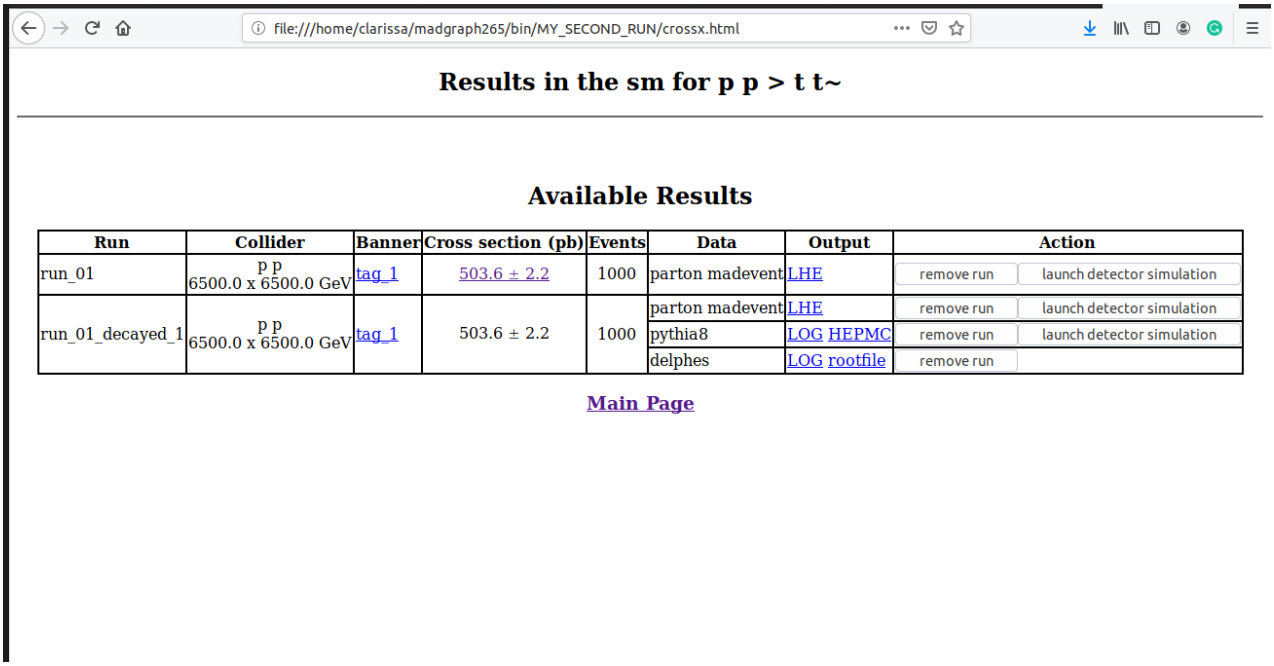
```
     Cross-section :   503.6 +- 2.154 pb
     Nb of events :  1000

INFO: storing files of previous run
INFO: Storing Pythia8 files of previous run
INFO: Done
quit
INFO:
more information in /home/clarissa/madgraph265/bin/MY_SECOND_RUN/index.html
MG5_aMC>
```

When MadGraph finishes, you should see that the cross section is in the order of 4.17e+08 pb, which is big! To see the individual contributions you should write:

open index.html



and click on **Results and Event Database.**



If you click on the cross section link number, you will see another website that enlists all the individual partonic subprocesses and their weights. We could see that mostly all the cross-section they are due to the gluón+gluón > bottom + antibottom process.

s= 4.1751e+08 ± 2.11e+06 (pb)

Graph	Cross-Section ↓	Error	Events (K)	Unwgt	Luminosity
/home/clarissa/madgraph265/bin/MY_SECOND_RUN/SubProcesses/P2_gg_bbx	4.115e+08	2.1e+06	28.232	1570.0	0
/home/clarissa/madgraph265/bin/MY_SECOND_RUN/SubProcesses/P2_qq_bbx	6.041e+06	6.19e+04	7.039	357.0	0
/home/clarissa/madgraph265/bin/MY_SECOND_RUN/SubProcesses/P1_gg_ttx	439.8	3.77	14.026	754.0	0
/home/clarissa/madgraph265/bin/MY_SECOND_RUN/SubProcesses/P1_qq_ttx	65.72	0.675	7.032	373.0	0

s= 439.78 ± 3.77 (pb)

Graph	Cross-Section ↓	Error	Events (K)	Unwgt	Luminosity
G2	393.2	3.74	7.01	382.0	0.972
G1	46.58	0.497	7.016	372.0	7.99

s= 4.1147e+08 ± 2.1e+06 (pb)

Graph	Cross-Section ↓	Error	Events (K)	Unwgt	Luminosity
G2	4.018e+08	2.1e+06	21.216	1168.0	2.9e-06
G1	9.649e+06	1.53e+05	7.016	402.0	4.17e-05

s= 65.717 ± 0.675 (pb)

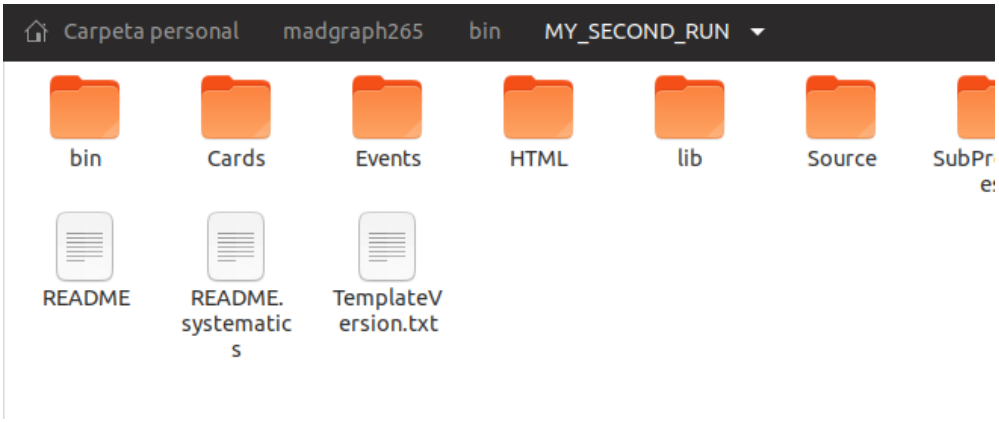
Graph	Cross-Section ↓	Error	Events (K)	Unwgt	Luminosity
G1	65.72	0.675	7.032	373.0	5.68

s= 6.0407e+06 ± 6.19e+04 (pb)

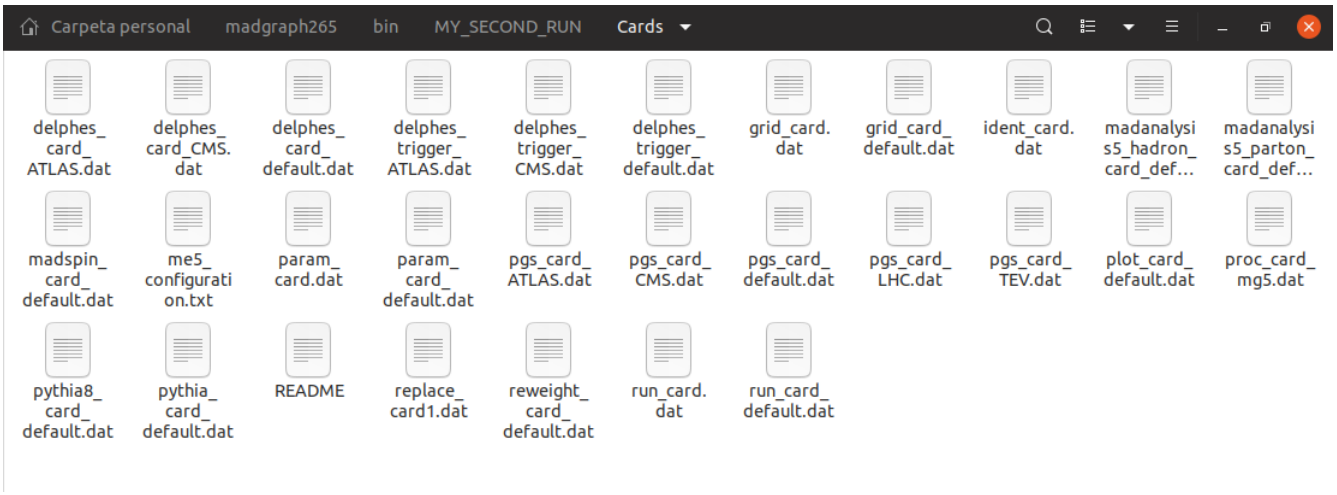
Graph	Cross-Section ↓	Error	Events (K)	Unwgt	Luminosity
G1	6.041e+06	6.19e+04	7.039	357.0	5.91e-05

Cards

In our run directory we have some files inside a directory called “Cards”



If we open it



We will center our attention on the following cards:

- Parameter card
- Run card
- Pythia card
- Delphes and/or pgs card
- Plot card
- Process card

- Parameter card (param_card.dat): It contains the information of the model that we are working. Here we can change, the masses of the particles for default, for example, the Higgs and quarks masses.

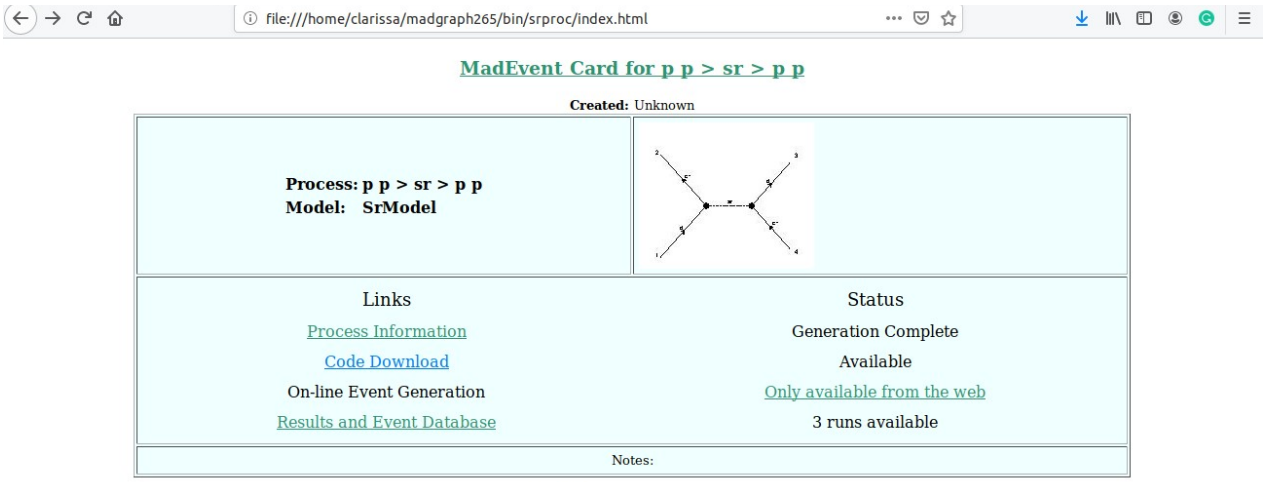
```
###################################################################
## PARAM_CARD AUTOMATICALY GENERATED BY MG5 FOLLOWING UFO MODEL   ####
###################################################################
##                                                                ##
##  Width set on Auto will be computed following the information    ##
##        present in the decay.py files of the model.             ##
##        See  arXiv:1402.1178 for more details.                  ##
##                                                                ##
###################################################################

#####################################
## INFORMATION FOR MASS
#####################################
Block mass
     5 4.700000e+00 # MB
     6 1.730000e+02 # MT
    15 1.777000e+00 # MTA
    23 9.118800e+01 # MZ
    25 1.250000e+02 # MH
## Dependent parameters, given by model restrictions.
## Those values should be edited following the
## analytical expression. MG5 ignores those values
## but they are important for interfacing the output of MG5
## to external program such as Pythia.
  1 0.000000 # d : 0.0
  2 0.000000 # u : 0.0
  3 0.000000 # s : 0.0
  4 0.000000 # c : 0.0
  11 0.000000 # e- : 0.0
#####################################
## INFORMATION FOR YUKAWA
#####################################
Block yukawa
     5 4.700000e+00 # ymb
     6 1.730000e+02 # ymt
    15 1.777000e+00 # ymtau

## INFORMATION FOR SMINPUTS
#####################################
Block sminputs
    1 1.325070e+02 # aEWM1
 #####################################
 ## INFORMATION FOR DECAY
 #####################################
 DECAY   6 1.491500e+00 # WT
 DECAY  23 2.441404e+00 # WZ
 DECAY  24 2.047600e+00 # WW
 DECAY  25 6.382339e-03 # WH
 ## Dependent parameters, given by model restrictions.
 ## Those values should be edited following the
 ## analytical expression. MG5 ignores those values
 ## but they are important for interfacing the output of MG5
 ## to external program such as Pythia.
 DECAY  1 0.000000 # d : 0.0
 DECAY  2 0.000000 # u : 0.0
 DECAY  3 0.000000 # s : 0.0
 DECAY  4 0.000000 # c : 0.0
 DECAY  5 0.000000 # b : 0.0
 DECAY  11 0.000000 # e- : 0.0
 DECAY  12 0.000000 # ve : 0.0
 DECAY  13 0.000000 # mu- : 0.0
 DECAY  14 0.000000 # vm : 0.0
 DECAY  15 0.000000 # ta- : 0.0
 DECAY  16 0.000000 # vt : 0.0
 DECAY  21 0.000000 # g : 0.0
 DECAY  22 0.000000 # a : 0.0
```

```
2. +
_exp__2)/(Gf*sqrt__2)))
```

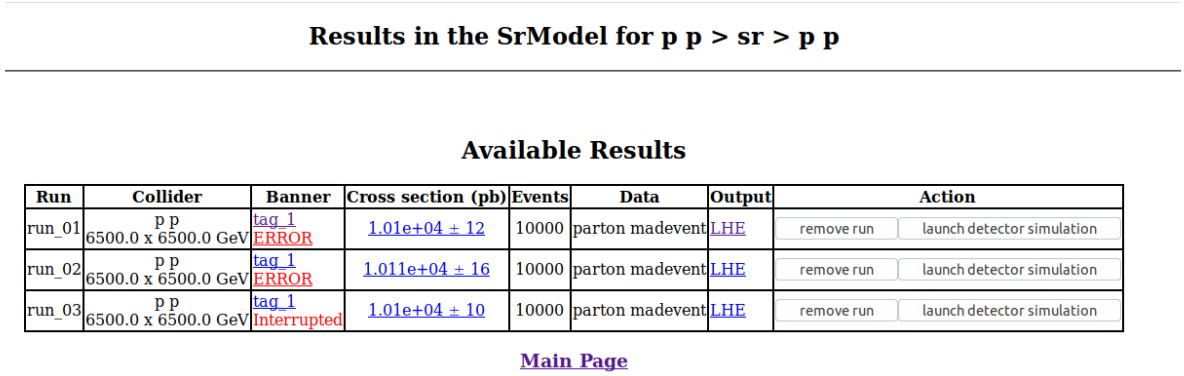
- Run card (run_card.dat): It contains information of the running. Here we can manage information of how many events we are asking (10000 for default) or the collide particles (usually proton-antiproton of electron-positron) and the beam particles. For experts, you can put kinematic restrictions.
- Pythia card (pythia_card.dat): It controls the pythia´s run, which follow to the partons final stage and it transform it to hadrons. The default it fine almost for everyone. Edit it is for experts.
- Delphes and/or pgs card (delphes_card.dat , pgs_card.dat): controls the simulation detectors. The dault are usually right and changing them is almost for experts. You can change things here, just like the energy and momentum resolutions for the detector simulation responsible of some particles. Another parameter that can be changed either directly or indirectly are: How are the magnetic fields inside the detector? Which is the granularity of the calorimeters? How often a Jet contains a Quark-b like a b-Jet? How often a lepton Tau is detected like a lepton Tau? Etc..
- Plot card (plot_card.dat): Controls which plots you want to do when the events generation finished. The default can produce almost every possible plot.
- Process Card (proc_card.dat): MadGraph does not ask you to edit this card. This card have the process information that you just generated, so it is there most like a reference in case that you forgot what process you asked.

Results Events and Data Base

We have the results of any process:



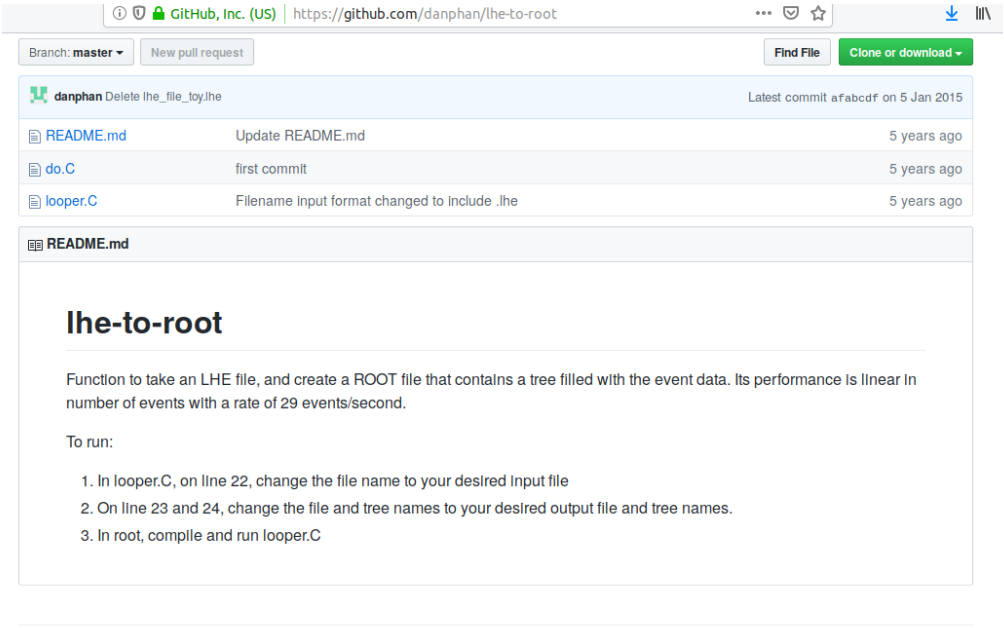
Press on Results events and Data Base:



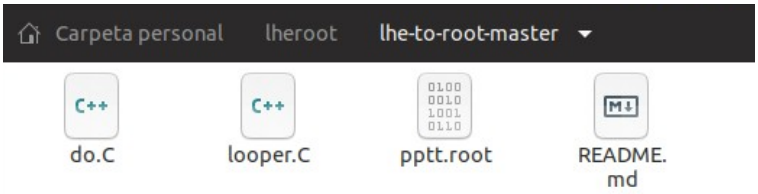
Press on one of the LHE hyperlinks, when you download the file, you can rename it

In this case, we named it **lhe1.lhe**.

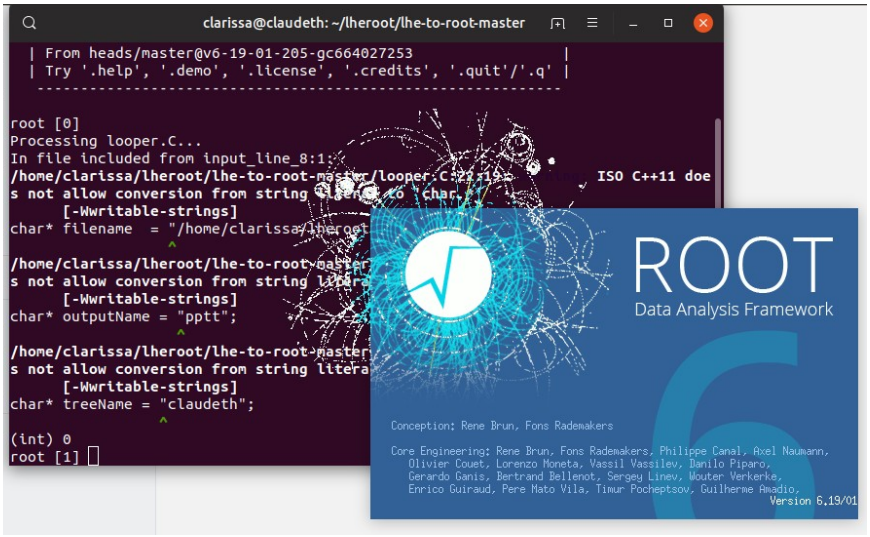
We have our results on an LHE format, we can transform it into a ROOT file format to make the graphs. To transform, I based on the website: https://github.com/danphan/lhe-to-root



Following the instructions and downloading the following files:



Opening a terminal , you will write **root looper.C**.



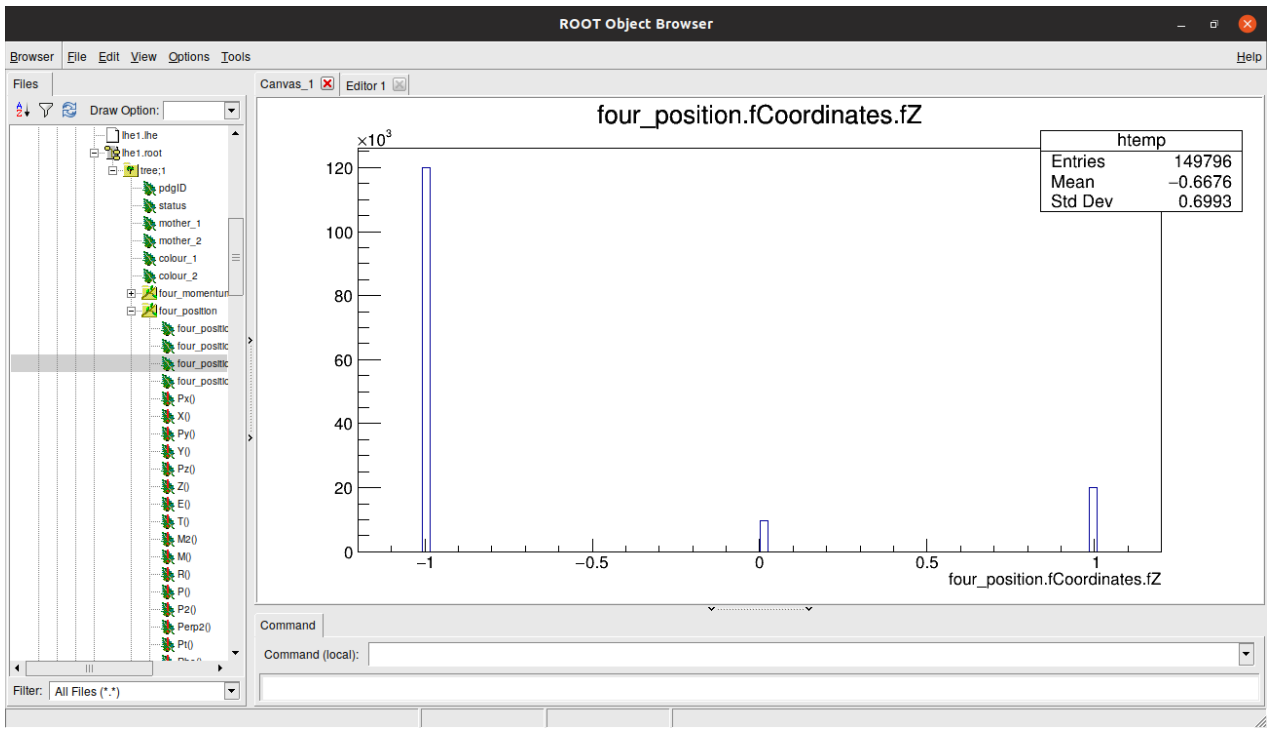
Now, exit from ROOT:

```
clarissa@claudeth:~/Verano Documentos/LHE1$ root looper.C
   ------------------------------------------------------------
  | Welcome to ROOT 6.19/01                  https://root.cern |
  |                               (c) 1995-2019, The ROOT Team |
  | Built for linuxx8664gcc on Jun 16 2019, 06:39:28           |
  | From heads/master@v6-19-01-205-gc664027253                 |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q' |
   ------------------------------------------------------------

root [0]
Processing looper.C...
In file included from input_line_8:1:
/home/clarissa/Verano Documentos/LHE1/looper.C:22:19: warning: ISO C++11 does not allow conversion from string literal to 'char *'
      [-Wwritable-strings]
char* filename  = "/home/clarissa/Verano Documentos/LHE1/lhe1.lhe";
                  ^
/home/clarissa/Verano Documentos/LHE1/looper.C:23:20: warning: ISO C++11 does not allow conversion from string literal to 'char *'
      [-Wwritable-strings]
char* outputName = "lhe1";
                   ^
/home/clarissa/Verano Documentos/LHE1/looper.C:24:18: warning: ISO C++11 does not allow conversion from string literal to 'char *'
      [-Wwritable-strings]
char* treeName = "sr1t";
                 ^
(int) 0
root [1] .q
clarissa@claudeth:~/Verano Documentos/LHE1$ ls
do.C  lhe1.lhe  lhe1.root  looper.C  looper.C~  unweighted_events.lhe.gz
```

Write **new TBrowser** On ROOT.

```
clarissa@claudeth:~/Verano Documentos/LHE1$ root
   ------------------------------------------------------------
  | Welcome to ROOT 6.19/01                  https://root.cern |
  |                               (c) 1995-2019, The ROOT Team |
  | Built for linuxx8664gcc on Jun 16 2019, 06:39:28           |
  | From heads/master@v6-19-01-205-gc664027253                 |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q' |
   ------------------------------------------------------------

root [0] new TBrowser
(TBrowser *) 0x55715ac7dcb0
root [1] (TFile *) 0x55715b499370
Warning in <TBufferFile::WriteObjectAny>: since TMethodBrowsable has no public constructor
        which can be called without argument, objects of this class
        can not be read with the current library. You will need to
        add a default constructor before attempting to read it.
Error in <TClass::New>: cannot create object of class TMethodBrowsable
Error in <TBufferFile::ReadObject>: could not create object of class TMethodBrowsable
.q
clarissa@claudeth:~/Verano Documentos/LHE1$
```

Hence, ROOT open you something like:



Conclusion

Using Madgraph allows you to simulate any collision from the SM or, from the BSM. You generate some events and when you output and launch the process, you can obtain a lot of data and results of your simulation. The results were given on an LHE program, which one can be converted into a Root file, obtaining finally the final plots of the simulation.

References

- *Introduction to elementary particles,* Second Edition. David Griffiths, Copyright © 2010 WILEY-VCH Verlag GmbH & Co. KgaA, Weinheim

  ISBM: 978-3-527-40601-2

- *Elementary Particle Physics for Enthusiasts*, Yoshiki Teramoto, Copyright © 2015

  ISBN: 1512142948 , ISBN-13: 978-1512142945

- Stephen P. Martin . *Notes on Madgraph syntax and examples for NIU PHYS 474/790,* by Northern Illinois University, Web site: https://www.niu.edu/spmartin/madgraph/madsyntax.html

- ZhangqierWang. (2016). *Madgraph and Delphes Tutorial* , by CERN Web site: https://twiki.cern.ch/twiki/bin/view/CMSPublic/MadgraphTutorial