Computer tools in particle physics

A. Vicente[a,b]

[a] Instituto de Física Corpuscular (CSIC-Universitat de València),
Apdo. 22085, E-46071 Valencia, Spain

[b] IFPA, Dep. AGO, Université de Liège,
Bat B5, Sart-Tilman B-4000 Liège 1, Belgium

Abstract

The field of particle physics is living very exciting times with a plethora of experiments looking for new physics in complementary ways. This has made increasingly necessary to obtain precise predictions in new physics models in order to be ready for a discovery that might be just around the corner. However, analyzing new models and studying their phenomenology can be really challenging. Computing mass matrices, interaction vertices and decay rates is already a tremendous task. In addition, obtaining predictions for the dark matter relic density and its detection prospects, computing flavor observables or estimating the LHC reach in certain collider signals constitutes quite a technical work due to the precision level that is currently required. For this reason, computer tools such as `SARAH`, `MicrOmegas`, `MadGraph`, `SPheno` or `FlavorKit` have become quite popular, and many physicists use them on a daily basis. In this course we will learn how to use these computer tools to explore new physics models and get robust numerical predictions to probe them in current and future experiments.

Notes of the mini-course given at CINVESTAV, Mexico
Cátedra Augusto García González, June 22nd-26th, 2015.

Contents

1 Introduction

When Wolfgang Pauli addressed his famous letter to the *Radioactive Ladies and Gentlemen* who met in Tübingen to discuss beta decay and other hot topics of the moment, inventing new particles was not a popular solution for a fundamental problem. In fact, a feeling of unease was clearly present in the text. In 1930, only three types of particles were already discovered: electrons, protons and photons. Adding a new particle, especially one with so weak interactions, was seen as a *desperate remedy*. However, these particles, later known as *neutrinos*, were indeed the solution for the problem of the continuous beta decay spectrum.

The situation is completely different nowadays. New particles are proposed every day in order to solve current problems in particle physics. Bosons or fermions, light or heavy, weakly or strongly interacting, these new hypothetical particles appear on arXiv on a daily basis. They are part of new models, some of them quite involved, proposed to address some of the issues that our current theoretical ideas, materialized in the Standard Model (SM), cannot explain.

With such an explosion of new models and particles, a systematic study of new physics scenarios has become necessary. In order to achieve this goal, many computer tools have been developed. They replace the *dirty work* traditionally given to PhD students, who spent long and tedious periods performing lengthy calculations or typing numerical codes. Instead, many of these tasks can nowadays be automatized, allowing physicists to concentrate on new ideas, rather than on technical details of complex computations. Furthermore, this also makes possible to run precise calculations, as currently required in order to test our theoretical expections at the accuracy level delivered by the experiments.

What is this course about?

In this course we will learn how to use several computer tools widely employed nowadays for phenomenological studies: `SARAH`, `SPheno`, `MicrOmegas` and `MadGraph`. We do not aim at a complete review of these tools, but just intend to provide a basic introduction. Nevertheless, after the course has been finished, we will be able to explore new physics models, study their properties analytically, obtain numerical results for many physical observables and run dark matter and collider studies.

Versions used for this course

These notes were elaborated using the following versions of the computer codes presented in the course:

- `SARAH-4.5.8`
- `SPheno-3.3.6`
- `micromegas_4.1.8`
- `MG5_aMC_v2_2_3`
- `MadAnalysis5_v1.1.11`

Two messages

Before we get started, let me tell you two important messages:

- **It is not so hard!**

One of the main goals of this course is to convince you that most of the current computer tools in particle physics are easy to use. Of course, it takes a lot of training to become a master, but getting started with them is actually quite easy. We will see that in some cases it suffices with a basic introduction to be able to produce valuable results.

- **Do not trust (too much) in codes!**

We must be careful when working with complicated computer codes. There is no such thing as a bug free software, and this also applies to particle physics tools. Therefore, we must analyze the results we obtain with a critical eye, trying to make sense of them using our physical intuition. Otherwise, we might make terrible mistakes simply by relying on buggy codes.

And now let us get started!

2 Lecture 1: Exploring new models with SARAH

In the first lecture we will learn how to use SARAH to explore a new model, study its properties and obtain input files for other computer tools.

2.1 What is SARAH?

SARAH is a Mathematica package for building and analyzing particle physics models. Although originally SARAH was designed to work only with supersymmetric models, after version 3 non-supersymmetric ones can be implemented as well. Once a model is defined in SARAH, the user can get all sorts of details about it: all vertices, mass matrices, tadpoles equations, 1-loop corrections for tadpoles and self-energies, and 2-loop renormalization group equations (RGEs). All this information about the model is derived by SARAH analytically and the user can simply handle it in Mathematica and use it for his own purposes. Furthermore, SARAH can export these analytical expressions into LaTeX files which, after the usual LaTeX compilation, result in pdf files containing all the details of the model.

> **TIP:** The LaTeX output given by SARAH is quite handy when preparing a paper. One can simply copy the relevant information, thus avoiding the tedious task of typing analytical formulas into LaTeX.

SARAH would already be a very useful tool just with the features described so far. However, there is more. SARAH writes input model files for FeynArts [1], CalcHep/CompHep [2, 3] (which can also be used as input for MicrOmegas), the UFO format which is supported by MadGraph, as well as for WHIZARD [4] and O'Mega [5]. As we will see in Sec. 2.5, this will save us a lot of time when we want to implement our favourite model in MicrOmegas or MadGraph, since SARAH can produce the required input files without us having to write them by hand.

Finally, let us briefly comment on some other interesting possibilities SARAH offers. It is well-known that Mathematica is not very efficient when dealing with heavy numerical calculations. For this reason, SARAH creates source code for SPheno, a code written in Fortran that allows for an efficient numerical evaluation of all the analytical expressions derived with SARAH. Other interesting features include the FlavorKit functionality for the calculation of flavor observables, the output for Vevacious, which can be used to check for the global minimum of the scalar potential of a given model, and the link to Susyno for the handling of group theoretical functions. For a complete and detailed descrition of SARAH and its many functionalities we refer to the manual and the recent pedagogical review [6].

2.2 SARAH: Technical details, installation and load

- **Name of the tool:** SARAH
- **Author:** Florian Staub (florian.staub@cern.ch)
- **Type of code:** Mathematica package
- **Website:** http://sarah.hepforge.org/
- **Manual:** [7–11]. For related functionalities see [12, 13], whereas for a pedagogical overview we recommend [6].

SARAH does not require any compilation. After downloading the package, one can simply copy the tar.gz file to the directory $PATH, where it can be extracted:

```
$ cp Download-Directory/SARAH_X.Y.Z.tar.gz $PATH/
$ cd $PATH
$ tar -xf SARAH_X.Y.Z.tar.gz
```

Here X.Y.Z must be replaced by the SARAH version which has been downloaded. SARAH can be used with any Mathematica version between 7 and 10.

In order to load SARAH one has to run in Mathematica the following command:

```
<<$PATH/SARAH-X.Y.Z/SARAH.m;
```

And now we are ready to use SARAH. For example, in order to initialize a model we just have to use the command

```
Start[model];
```

Here model is the name of the specific particle physics model we want to explore. However, before we do that, let us see how to define a new model in SARAH.

2.3 Defining a model in SARAH

There are already many models fully implemented in SARAH. A complete list is provided in Appendix A. If you want to study one of those, you can skip this section and go directly to Sec. 2.4.

We will now learn how to define a new model in SARAH. For this purpose, we have to visit the directory $PATH/SARAH-X.Y.Z/Models, where each folder contains the model files for a specific model. For example, the folder SM contains the model files for the Standard Model, the folder THDM-II those for the Two Higgs Doublet Model of type-II and the folder MSSM those for the Minimal Supersymmetric Standard Model. In each of these directories one finds four files:

- model.m (where model must be replaced by the name of the specific model): This file contains the basic definitions of the model.
- parameters.m: In this file we provide additional information about the parameters of the model.
- particles.m: This one is devoted to the particles in the model, with some details not present in model.m.
- SPheno.m: This file is only required if we want to create a SPheno module for our model, as shown in Sec. 2.5.

We will now show how to prepare these files for an example model: the *scotogenic model* [14]. This popular particle physics model is described in detail in Appendix C. For other physics conventions and basic definitions we also refer to the SM description in Appendix B.

Before we move on with the scotogenic model let us make a comment on Supersymmetry. In this course we will concentrate on non-supersymmetric models. The implementation of supersymmetric models is not very different, but it has a few details which would make the explanation a little more involved. In case you are interested in supersymmetric models, we recommend the guide [6].

Let us then implement the scotogenic model in SARAH. This model does not have too many ingredients beyond the SM, and thus this will be an easy task. First of all, we must create a new folder in $PATH/SARAH-X.Y.Z/Models called Scotogenic. Then we can copy the files from the SM implementation (located in the directory $PATH/SARAH-X.Y.Z/Models/SM), rename SM.m to Scotogenic.m, and then add the new elements (particles and interactions) to the four model files. We will now show the result of doing this for the scotogenic model.

> **TIP:** It is convenient not to create a new model from scratch. Instead, it is highly recommended to use a model that is already implemented in SARAH as basis for the implementation and simply add the new ingredients (particles and parameters). This way we avoid making unnecessary typos. Moreover, most of the new fields and interaction terms that we may consider for our own models are already introduced in the models distributed with SARAH. In most cases we can simply copy these elements to our model.

> **TIP:** All models files are of the type *.m. We can edit them using Mathematica, but I personally prefer to use a conventional text editor (like emacs or gedit).

All SARAH model files for the scotogenic model can be found in Appendix F.

Scotogenic.m

This is the central model file. Here we define the particles of the model, the Lagrangian, the gauge and global symmetries and the way they get broken. If this model file is properly written, SARAH can already make lots of useful computations for us, making the other files optional.

The first lines of the file contain some general Mathematica commands that might be useful. In our case we have

Scotogenic.m

```
Off[General::spell]
```

which simply switches off Mathematica warnings when the name of a new symbol is similar to the name of existing internal symbols. This is of course optional, but it is usually useful to get rid of these unwanted messages. It follows some general information about the model: its name, the authors of the implementation and the date of the implementation:

Scotogenic.m

```
Model`Name      = "Scotogenic";
Model`NameLaTeX = "Scotogenic Model";
Model`Authors   = "N. Rojas, A. Vicente";
Model`Date      = "2015-04-28";

(* "28-04-2015 (first implementation)" *)
(* "25-05-2015 (removed mixings in scalar sector)" *)
(* "10-06-2015 (fixed conventions)" *)
```

The first name (Model`Name) is the internal name that will be used in SARAH and should not contain any special character. The second name (Model`NameLaTeX) is the complete name of the model in LaTeX syntax. Notice that we have added three comments (not relevant for SARAH) just to keep track of the last modification in the model files. Comments are of course accepted in the model files and they can be introduced as usual in Mathematica by using (* comment *). As for any code, they are welcome, since they clarify the different parts of the model files and help us when we try to understand the code. After these basic details of the model we must define the symmetries of the model: global and gauge. SARAH supports $\mathbb{Z}_N$ as well as $U(1)$ global symmetries. These are defined by means of the array Global. The first element of the array is the type of symmetry, whereas the second element is the name. In the case of the scotogenic model we have a $\mathbb{Z}_2$ parity. Therefore,

Scotogenic.m

```
Global[[1]] = {Z[2], Z2};
```

It is the turn for the gauge symmetry of the model. In the scotogenic model this is just the SM gauge symmetry, defined in SARAH as

Scotogenic.m

```
Gauge[[1]]={B,   U[1], hypercharge, g1, False, 1};
Gauge[[2]]={WB, SU[2], left,        g2, True , 1};
Gauge[[3]]={G,  SU[3], color,       g3, False, 1};
```

Each gauge group is given with an array called Gauge. The first element of the array is the name of the gauge boson, the second element is the gauge group, the third element is the name of the group and the fourth one is the name of the gauge coupling. The fifth entry in the array can be either True or False, and sets whether the gauge indices for this group must be expanded in the analytical expressions. For $U(1)_Y$ this is not relevant, and we just set it to False. For $SU(2)_L$ it is convenient to expand the analytical expressions in terms of the elements of the $SU(2)_L$ multiplets, and thus we use True. Since $SU(3)_c$ will not get broken, the elements in the multiplets will always appear together, and thus it is preferable to use False. Finally, the last entry of the array sets the global charge of the gauge bosons. In this case all gauge bosons are positively charged under $\mathbb{Z}_2$.

The next step is the definition of the particle content of the model. First, the fermion fields:

Scotogenic.m

```
FermionFields[[1]] = {q , 3, {uL, dL},     1/6, 2,  3, 1};
```

```
FermionFields[[2]] = {l , 3, {vL, eL},     -1/2, 2,  1, 1};
FermionFields[[3]] = {d , 3, conj[dR],      1/3, 1, -3, 1};
FermionFields[[4]] = {u , 3, conj[uR],     -2/3, 1, -3, 1};
FermionFields[[5]] = {e , 3, conj[eR],        1, 1,  1, 1};
FermionFields[[6]] = {n , 3, conj[nR],        0, 1,  1,-1};
```

Each `FermionFields` array corresponds to a fermionic gauge multiplet. The first entry is the name of the fermion, the second the number of generations and the third the name of the $SU(2)_L$ components. The rest of entries are the charges under the gauge and global symmetries. For example, the first fermion multiplet, `FermionFields[[1]]` is the SM quark doublet q, with three generations and decomposed in $SU(2)_L$ components as

$$q = \begin{pmatrix} u_L \\ d_L \end{pmatrix} . \tag{1}$$

The charges under $U(1)_Y \times SU(2)_L \times SU(3)_c$ are $(\frac{1}{6}, \mathbf{2}, \mathbf{3})$, and the charge under the global $\mathbb{Z}_2$ is $+1$. Note that the only fermion negatively charged under $\mathbb{Z}_2$ is the right-handed neutrino, n. It is also important to notice that all fermions have to be defined as left-handed. For example, the $SU(2)_L$ singlets are identified as $d \equiv d_R^*$, $u \equiv u_R^*$, $e \equiv e_R^*$ and $n \equiv \nu_R^*$.

We now introduce the scalar fields of the model,

Scotogenic.m

```
ScalarFields[[1]] =  {H,  1, {Hp, H0},     1/2, 2,  1,  1};
ScalarFields[[2]] =  {Et, 1, {etp,et0},    1/2, 2,  1, -1};
```

which follow exactly the same conventions as for the fermions. With these two lines we have defined the SM Higgs doublet H and the inert doublet η.

After the matter content of the model is introduced, we must define two sets of states: `GaugeES` and `EWSB`.

Scotogenic.m

```
NameOfStates={GaugeES, EWSB};
```

The first set is composed by the gauge eigenstates, whereas the second is composed by the mass eigenstates after electroweak symmetry breaking (EWSB). For the scotogenic model these two sets are sufficient, but in some models we may consider some intermediate basis. Therefore, the array `NameOfStates` can be longer if necessary.

The time has come to define the Lagrangian. In `SARAH`, all kinetic terms are supposed to be canonical and thus there is no need to define them. For the rest, the mass and interaction terms, there is. In the scotogenic model this can be done as follows:

Scotogenic.m

```
DEFINITION[GaugeES][LagrangianInput]=
{
  {LagFer    ,       {AddHC->True}},
  {LagNV     ,       {AddHC->True}},
  {LagH      ,       {AddHC->False}},
  {LagEt     ,       {AddHC->False}},
  {LagHEt    ,       {AddHC->False}},
  {LagHEtHC  ,       {AddHC->True}}
};

LagFer   = Yd conj[H].d.q + Ye conj[H].e.l + Yu H.u.q + Yn Et.n.l;
LagNV    = Mn/2 n.n;
LagH     = -(- mH2 conj[H].H     + 1/2 lambda1 conj[H].H.conj[H].H );
LagEt    = -(+ mEt2 conj[Et].Et  + 1/2 lambda2 conj[Et].Et.conj[Et].Et );
LagHEt   = -(+ lambda3 conj[H].H.conj[Et].Et + lambda4 ↪
    conj[H].Et.conj[Et].H );
LagHEtHC = -(+ 1/2 lambda5 conj[H].Et.conj[H].Et );
```

First, we have split the Lagrangian in different pieces: `LagFer`, `LagNV`, `LagH`, `LagEt`, `LagHEt` and `LagHEtHC`. This is done for convenience. For each piece, we must set the option `AddHC` to either `True` or `False`. This option is use to decide whether the Hermitian conjugate of the Lagrangian piece must be added as well or not.

In our case, we have used `False` for the self-conjugated terms and `True` for the rest. Then the different terms are defined as well. For this purpose we can use `conj[X]` in order to denote the Hermitian conjugate of the `X` multiplet. For fermions, *barred spinors* are automatically introduced. For example, `LagFer` is the Yukawa Lagrangian for the fermions,

$$\texttt{LagFer} \equiv \mathcal{L}_Y = Y_d H^\dagger \,\bar{d}\, q + Y_e H^\dagger \,\bar{e}\, \ell + Y_u H \,\bar{u}\, q + Y_N \,\eta\, \overline{N}\, \ell\,, \tag{2}$$

which requires the addition of the Hermitian conjugate. Notice that all scalar terms are defined with a global sign. This is simply convenient to better identify the scalar potential of the model ($\mathcal{L} \supset -\mathcal{V}$).

Now we find several definitions related to the breaking of the gauge symmetry and the resulting mass eigenstates. First, for the gauge sector, we have

Scotogenic.m

```
59 DEFINITION[EWSB][GaugeSector] =
60 {
61   {{VB,VWB[3]},{VP,VZ},ZZ},
62   {{VWB[1],VWB[2]},{VWp,conj[VWp]},ZW}
63 };
```

In these lines we are defining the mixing of the gauge bosons. In line 60 we do it for the neutral gauge bosons, $\{B, W_3\} \to \{\gamma, Z\}$, using as name for the mixing matrix `ZZ` (the unitary matrix Z^Z in Appendix B), whereas in line 61 we do it for the charged ones, with $\{W_1, W_2\} \to \left\{W^+, (W^+)^*\right\}$, using as name for the mixing matrix `ZW` (the unitary matrix Z^W in Appendix B). Next, we define the decomposition of the scalar fields into their CP-even and CP-odd components, including also the possibility of having VEVs. This is simply done with

Scotogenic.m

```
67 DEFINITION[EWSB][VEVs]=
68 {
69   {H0,  {v, 1/Sqrt[2]}, {Ah, \[ImaginaryI]/Sqrt[2]}, {hh, 1/Sqrt[2]}},
70   {et0, {0, 0}, {etI, \[ImaginaryI]/Sqrt[2]}, {etR, 1/Sqrt[2]}}
71 };
```

where we take the neutral components in H and η, H^0 and η^0, and split them into several pieces,

$$H^0 = \frac{1}{\sqrt{2}}\,(v + h + iA)\,, \tag{3}$$

$$\eta^0 = \frac{1}{\sqrt{2}}\,(\eta_R + i\eta_I)\,. \tag{4}$$

Here $v/\sqrt{2}$ is the Higgs VEV, h (`hh` in the code) and η_R (`etR` in the code) are the CP-even components and A (`Ah` in the code) and η_I (`etI` in the code) the CP-odd ones. It is worth noticing that we have set the η^0 VEV to zero.

We are almost done. The next step is the definition of the mass eigenstates in terms of the gauge eigenstates. This is completely equivalent to the definition of the mixings in the model. In the scotogenic model this is accomplished by means of the following lines of code:

Scotogenic.m

```
73 DEFINITION[EWSB][MatterSector]=
74 {
75   {{conj[nR]},{X0, ZX}},
76   {{vL}, {VL, Vv}},
77   {{{dL}, {conj[dR]}}, {{DL,Vd}, {DR,Ud}}},
78   {{{uL}, {conj[uR]}}, {{UL,Vu}, {UR,Uu}}},
79   {{{eL}, {conj[eR]}}, {{EL,Ve}, {ER,Ue}}}
80 };
```

Here we have, on line 74 the right-handed neutrinos (which do not mix with any other field), on line 75 the left-handed neutrinos (which do not have other mixings either), on line 76 the down-type quarks, on line 77 the up-type quarks and on line 78 the charged leptons. As can be seen from the previous lines, there are several ways to make this definition, depending on the type of states:

- **For scalars and Majorana fermions:**

{{gauge eigenstate 1, gauge eigenstate 2, ...}, {mass matrix, mixing matrix}}

- **For Dirac fermions:**

{{{gauge eigenstate left 1, gauge eigenstate left 2, ...}, {mass matrix left, mixing matrix left},{gauge eigenstate right 1, gauge eigenstate right 2, ...}, {mass matrix right, mixing matrix right}}}

For example, the singlet neutrinos are Majorana fermions and can mix among themselves. We denote the mass eigenstates as `X0` and the mixing matrix as `ZX`. The charged leptons, on the other hand, are Dirac fermions: the left-handed states are transformed with a matrix called `Ve` leading to the states `EL` and the right-handed ones are transformed with the matrix `Ue` leading to the states `ER`. All these transformations are unitary matrices, and are the usual *rotations* that connect the gauge and mass basis.

We have not included in this list the mass eigenstates that do not mix (and thus are equal to the gauge eigenstates). This is the case of h, A, H^+, η_R, η_I and η^+, whose mixings are forbidden by the $\mathbb{Z}_2$ parity of the scotogenic model. These mass eigenstates have to be properly defined in the file `particles.m`, but should not be included in `DEFINITION[EWSB][MatterSector]`.

> **TIP:** We have to be careful when defining the mixings. We may forget about some of them or introduce mixing among particles which do not really mix. One way to realize about these potential mistakes is to run the command `CheckModel[model]` after loading the model in `SARAH`. This `SARAH` command checks the model files trying to find inconsistencies or missing definitions. In some cases it might be able to detect undefined mixings.

Finally, the last part of the `Scotogenic.m` file is used to define Dirac spinors. This is because so far all the fermions we have considered are 2-components Weyl spinors, since this is the way they are internally handled by `SARAH`. Therefore, we must tell `SARAH` how to combine them to form 4-component Dirac fermions, more common in particle physics calculations. This is done for the mass eigenstates,

Scotogenic.m

```
86 DEFINITION[EWSB][DiracSpinors]=
87 {
88   Fd  -> {  DL, conj[DR]},
89   Fe  -> {  EL, conj[ER]},
90   Fu  -> {  UL, conj[UR]},
91   Fv  -> {  VL, conj[VL]},
92   Chi -> {  X0, conj[X0] }
93 };
```

as well as for the gauge eigenstates,

Scotogenic.m

```
95 DEFINITION[EWSB][GaugeES]=
96 {
97   Fd1 ->{  FdL, 0},
98   Fd2 ->{  0, FdR},
99   Fu1 ->{  Fu1, 0},
100  Fu2 ->{  0, Fu2},
101  Fe1 ->{  Fe1, 0},
102  Fe2 ->{  0, Fe2}
103 };
```

For the gauge eigenstates there is no need to be exhaustive, since these Dirac fermions are not used for practical calculations (always performed in the mass basis), but for the mass eigenstates we must include in the list all the possible Dirac fermions after EWSB. Notice that in this case we have used the definitions of the mass eigenstates previously done in `DEFINITION[EWSB][MatterSector]` and that `Fv` and `Chi` are Majorana fermions, since the 2-component spinors that form them are conjugate of each other.

This concludes our review of the file `Scotogenic.m`.

parameters.m

In this file we provide additional information about the parameters of the model. This includes the original Lagrangian parameters as well as mixing matrices and angles. The proper preparation of this file is actually optional, since it will only required when some special SARAH outputs are obtained.

> **TIP:** Again, it is convenient to use one of the existing models and adapt the corresponding `parameters.m` file. Since most of the parameters are common to all models (gauge couplings, SM Yukawa matrices, ...), this will save a lot of time and avoid typos.

Before we explain the content of this file, please note that there is a file placed in `$PATH/SARAH-X.Y.Z/Models` also called `parameters.m`. This is a general file with the most common parameters already defined. For example, in this file one can find the definition of the SM gauge couplings (g_1, g_2 and g_3), some of the usual mixing matrices (like the one for the left-handed neutrinos, called `Neutrino-Mixing-Matrix` in the code) and some derived parameters like the weak mixing angle θ_W (called `ThetaW` in the code). These definitions have been made to simplify our life when defining a new model that shares some of them. In this case, although they have to be defined in our new `parameters.m` file, it suffices to point to one of these definitions for SARAH to know all the details of the parameter. We will see how to do this when we discuss the option `Description` below.

The structure of the file (of both `parameters.m` files, actually) is as follows:

parameters.m

```
ParameterDefinitions = {

{Parameter,   {Option 1 -> "value option 1",
               Option 2 -> "value option 2",
               ... }},


...

};
```

For each parameter several options can be defined. Most of them are optional and rarely necessary. For the implementation of the scotogenic model we will only need the following options:

- **Description:** this is a string that identifies the parameter if this has been previously defined in the general file `$PATH/SARAH-X.Y.Z/Models/parameters.m`. As explained above, we do not need to redefine the usual parameters each time we implement a new model. We just have to write in `Description` the same string name that is given to the parameter in the general `$PATH/SARAH-X.Y.Z/Models/parameters.m`. Furthermore, even if this parameter is not defined in the general file, this option can be used as a way to give a human readable name to the parameter, so that we can easily identify it when we open the `parameters.m` long after the implementation.

- **Real:** this option can be either `True` or `False`. If the option is set to `True`, SARAH assumes that the parameter is real. By default, all parameters are considered complex.

- **OutputName:** this is a string to be used to identify the parameters in several SARAH outputs (like `MicrOmegas`). In order to be compatible with all computer languages, this string should not contain any special characters.

- **LaTeX:** this option defines the way the parameter is shown in LaTeX. As we will see below (Sec. 2.4), SARAH can export all the derived model properties in LaTeX format. Properly defining this option for each parameter guarantees a nice and readable text. In doing this one should take into account that '\' is interpreted by `Mathematica` as escape sequence. Therefore, '\' has to be replaced by '\\' in all LaTeX commands.

- **LesHouches:** this option defines the position of the parameter in a LesHouches spectrum file. This will be important when we run numerical studies, since most input and output files follow this standard. If the parameter is a matrix we have to give the name of the block, whereas the name of the block and the entry have to be provided for parameters which are numbers.

Since the list of parameters is quite long, we will not review here all the definitions for the scotogenic model. Nevertheless, a few examples will be useful to see how it works. First, the neutrino Yukawa couplings Y_N are defined with the lines

parameters.m

```
{Yn,   {LaTeX -> "Y_N",
        LesHouches -> YN,
        OutputName->Yn }},
```

No additional information is required. For the neutrino mixing matrix we have something even simpler

parameters.m

```
{Vv, {Description ->"Neutrino-Mixing-Matrix"}}
```

Since this mixing matrix is common to other models with non-zero neutrino mixings, it is already defined in the general file `$PATH/SARAH-X.Y.Z/Models/parameters.m`. Therefore, including the option `Description` suffices for all the options to be properly defined. For example, this way we set `OutputName` to `UV`, `LaTeX` to U^V and `LesHouches` to `UVMIX`. Finally, the λ_5 coupling is defined by the lines

parameters.m

```
{lambda5,   {Real -> True,
             LaTeX -> "\\lambda_5",
             LesHouches -> {HDM,6},
             OutputName-> lam5 }},
```

Note that in the `LesHouches` option we have provided a block name (`HDM`) as well as an entry number. In addition, the parameter has been defined as real. This justifies the splitting of the scalar fields into CP-even and CP-odd states. In the presence of a complex λ_5 parameter this would not be possible since both states would mix.

Just in case you find some additional requirements when implementing another model, here you have two other useful options:

- **Dependence:** this option should be used when we want `SARAH` to replace the parameter by a particular expression in all analytical calculations. For example, in the SM the neutral gauge boson (γ, Z) mixing matrix is parameterized in terms of one angle, the so-called weak or Weinberg angle θ_W, see Eq. (35). With this option we would tell `SARAH` to use this parameterization in all analytical computations.
- **DependenceNum:** this option is similar to `Dependence`, with the only exception that the replacement is only used in numerical calculations. For example, we probably want to obtain all analytical results in terms of the SM gauge couplings g_i, but replace them in the numerical calculations by their expressions in terms of α_s (in case of the strong coupling constant), θ_W and e (the electron charge).

Finally, it is worth clarifying what happens in case of conflicts. Whenever an option is defined in the general file `$PATH/SARAH-X.Y.Z/Models/parameters.m` and later included explicitly in the specific `parameters.m` file for our model, `SARAH` will take the value of the option given in the specific file.

particles.m

This file is devoted to the particle content of the model. Although the basic information is already given in `Scotogenic.m`, there are some additional details that must be given in `particles.m`. As for `parameters.m`, this is an optional file that will only be required when producing some special `SARAH` outputs.

The particles (or more precisely, *states*) of the model are distributed into three categories: gauge and mass eigenstates and intermediate states. We have already mentioned the first two categories. The third one is composed by states which are neither gauge eigenstates nor mass ones, but appear in intermediate calculations. For instance, the 2-component Weyl fermions `X0` belong to this class, since the gauge eigenstates are `nR` and the mass eigenstates (used in practical calculations) are the 4-component Dirac fermions `Chi`.

As for the parameters, there is a general file where the definitions of the most common particles are already given. This file is located in `$PATH/SARAH-X.Y.Z/Models/particles.m` and its practical use is again similar: we can simply point to one of the existing definitions in case our model has a particle that is already in the general file.

The structure of the file (again, of both `particles.m` files) is as follows:

particles.m

```
ParticleDefinitions[states] = {

```

```
{Particle,   {Option 1 -> "value option 1",
              Option 2 -> "value option 2",
               ... }},

...

};
```

In practice, it is not necessary to provide definitions for gauge eigenstates and intermediate states since these do not participate in the calculation of physical observables. The only option that should be defined for these states is `LaTeX`, which will be helpful to get a readable LaTeX output. In contrast, the properties of the mass eigenstates are crucial, since they must be read by other tools such as `MicrOmegas` or `MadGraph`.

Let us show a couple of illustrative examples in the scotogenic models. The definitions of the η scalars (mass eigenstates) is as follows

particles.m

```
      {etR,   {  Description -> "CP-even eta scalar",
                 PDG -> {1001},
                 Mass -> LesHouches,
                 ElectricCharge -> 0,
                 LaTeX -> "\\eta_R",
                 OutputName -> "etR" }},
      {etI,   {  Description -> "CP-odd eta scalar",
                 PDG -> {1002},
                 Mass -> LesHouches,
                 ElectricCharge -> 0,
                 LaTeX -> "\\eta_I",
                 OutputName -> "etI" }},
      {etp,   {  Description -> "Charged eta scalar",
                 PDG -> {1003},
                 Mass -> LesHouches,
                 ElectricCharge -> 1,
                 LaTeX -> "\\eta^+",
                 OutputName -> "etp" }},
```

We have used the option `Description` to give simple names to the three mass eigenstates. Although they are not used (since these descriptions are not present in the general file `$PATH/SARAH-X.Y.Z/Models/particles.m`), they are helpful for future reference. We have also given `PDG` codes to the three states, using high numbers not reserved for other particles. This is necessary for `MadGraph`, which uses these codes to identify the particles. Moreover, we have defined the `ElectricCharge` and the way the particles should be shown in LaTeX format. The option `OutputName` is completely analogous to the same option in the case of parameters. Finally, we have set the option `Mass` to `LesHouches`. This option defines the way in which `MadGraph` should obtain the value of this mass. With `LesHouches`, we are telling `SARAH` that we would like `MadGraph` to take this value from a LesHouches input file (probably obtained with a spectrum generator like `SPheno`, see Sec. 2.5).

When there are several generations of a given mass eigenstate, some of the options have to be given as arrays. An example can be found in the definition of the fermion singlets

particles.m

```
{Chi,  { Description -> "Singlet Fermions",
                PDG -> {1012,1014,1016},
                Mass -> LesHouches,
                ElectricCharge -> 0,
                LaTeX -> "N",
                OutputName -> "N" }}
```

In contrast to the definitions of the η scalars, in this case the option `PDG` must be an array of three elements, since there are three generations of singlet fermions.

Finally, an option that is crucial for the proper implementation of the model is `Goldstone`. This option should be included in the definition of every massive gauge boson. It tells `SARAH` where to find the corresponding Goldstone boson that becomes its longitudinal component. For example, for the Z boson one has

```
80 {VZ,   { Description -> "Z-Boson", Goldstone -> Ah }},
```

Note that the only option that we must add is `Goldstone`. The rest of options for the Z boson are given in the general file `$PATH/SARAH-X.Y.Z/Models/particles.m`.

SPheno.m

Finally, the file `SPheno.m` is only necessary if we plan to create a `SPheno` module for our model. This is explained in more detail in Sec. 2.5, and thus we postpone the description of this file until we reach that point of the course.

2.4 Exploring a model

The model is implemented and the time has come to see what `SARAH` can do for us.

First, we have to load `SARAH` and the scotogenic model. As shown already, we can do that with these `Mathematica` commands:

```
<<$PATH/SARAH-X.Y.Z/SARAH.m;
Start["Scotogenic"];
```

After a few seconds all the initial `SARAH` computations will be finished and we will be ready to execute all kinds of commands to get analytical information about the model.

Tadpole equations

The minimization of the scalar potential proceeds via the *tadpole equations* [1],

$$\frac{\partial \mathcal{V}}{\partial v_i} = 0\,, \tag{5}$$

where v_i are the VEVs of the scalar fields. One has as many equations as VEVs. In the scotogenic model there is only one non-zero VEV, the VEV of the SM Higgs doublet. Therefore, we just need to solve one equation to minimize the scalar potential. This is

$$\frac{\partial \mathcal{V}}{\partial v} = 0\,. \tag{6}$$

`SARAH` can provide the analytical form of this equation. This is obtained with the command

```
TadpoleEquation[v]
```

We find the result

$$\frac{1}{2}\lambda_1 v^3 - m_H^2 v = 0\,, \tag{7}$$

which, using the `Mathematica` command

```
Solve[TadpoleEquation[v], mH2]
```

gives the well-known minimization condition

$$m_H^2 = \frac{1}{2}\lambda_1 v^2\,. \tag{8}$$

Finally, we point out that the command

```
TadpoleEquations[EWSB]
```

can be used to obtain the complete list of tadpole equations of a model.

[1] Note that we are assuming here CP conservation in the scalar sector.

Masses

Next, we can print some masses. There are two ways to do this, depending on whether the mass eigenstate we are interested in is a mixture of gauge eigenstates or not. When it is, we must print the mass matrix of the complete set of mass eigenstates. This is done with

```
MassMatrix[state]
```

where `state` must be replaced by the name of the mass eigenstate. For example, we can run the command

```
MassMatrix[Fe]
```

and `SARAH` would return the well-known form of the charged lepton mass matrix,

$$\begin{pmatrix} -\frac{v(Y_e)_{11}}{\sqrt{2}} & -\frac{v(Y_e)_{21}}{\sqrt{2}} & -\frac{v(Y_e)_{31}}{\sqrt{2}} \\ -\frac{v(Y_e)_{12}}{\sqrt{2}} & -\frac{v(Y_e)_{22}}{\sqrt{2}} & -\frac{v(Y_e)_{32}}{\sqrt{2}} \\ -\frac{v(Y_e)_{13}}{\sqrt{2}} & -\frac{v(Y_e)_{23}}{\sqrt{2}} & -\frac{v(Y_e)_{33}}{\sqrt{2}} \end{pmatrix}. \tag{9}$$

We can also print the mass matrix for the singlet fermions. By executing the command

```
MassMatrix[Chi]
```

we obtain

$$\begin{pmatrix} -(M_N)_{11} & -\frac{1}{2}(M_N)_{12} - \frac{1}{2}(M_N)_{21} & -\frac{1}{2}(M_N)_{13} - \frac{1}{2}(M_N)_{31} \\ -\frac{1}{2}(M_N)_{12} - \frac{1}{2}(M_N)_{21} & -(M_N)_{22} & -\frac{1}{2}(M_N)_{23} - \frac{1}{2}(M_N)_{32} \\ -\frac{1}{2}(M_N)_{13} - \frac{1}{2}(M_N)_{31} & -\frac{1}{2}(M_N)_{23} - \frac{1}{2}(M_N)_{32} & -(M_N)_{33} \end{pmatrix}. \tag{10}$$

Notice that `SARAH` does not know that the matrix M_N is symmetric. If required, we can simplify the expression with the command

```
MassMatrix[Chi] /. Mn[i_, j_] :> If[i > j, Mn[j, i], Mn[i, j]]
```

obtaining the more standard form

$$\begin{pmatrix} -(M_N)_{11} & -(M_N)_{12} & -(M_N)_{13} \\ -(M_N)_{12} & -(M_N)_{22} & -(M_N)_{23} \\ -(M_N)_{13} & -(M_N)_{23} & -(M_N)_{33} \end{pmatrix}. \tag{11}$$

In case we want to print the mass of a state that does not mix with other fields we must use the command

```
Mass[state] /. Masses[EWSB]
```

where, again, `state` must be replaced by the name of the specific mass eigenstate. As a prime example, let us consider the Higgs boson. Its mass can be printed with the command

```
Mass[hh] /. Masses[EWSB]
```

leading to

$$\frac{3}{2}\lambda_1 v^2 - m_H^2 . \tag{12}$$

Two things should be noted: (1) we actually got the Higgs boson squared mass, and (2) the minimization condition in Eq. (8) has not been applied. To obtain the resulting Higgs boson mass after applying the tadpole equations, we can simply run

```
solTadpole = Solve[TadpoleEquation[v], mH2];
Mass[hh] /. Masses[EWSB] /. solTadpole
```

obtaining the final expression for the Higgs boson mass

$$\lambda_1 v^2 \,. \tag{13}$$

Vertices

One of the most powerful features of SARAH is the calculation of interaction vertices. In order to obtain the a vertex one must execute the command

```
Vertex[{state 1, state 2, state 3}]
```

or

```
Vertex[{state 1, state 2, state 3, state 4}]
```

depending on the number of particles involved in the vertex. Here `state 1`, `state 2`, `state 3` and `state 4` are mass eigenstates. The result of this command is an array that includes all possible Lorentz structures appearing in the interaction vertex and the corresponding coefficients. For example, the $\ell_i^+ - \ell_j^- - h$ vertex, where $i, j = 1, 2, 3$ are flavor indices, is obtained with

```
Vertex[{bar[Fe], Fe, hh}]
```

and gives,

$$\frac{i}{\sqrt{2}} \sum_{m,n=1}^{3} (V_e)^*_{jn} \, (Y_e)_{mn} \, (U_e)^*_{im} \; P_L + \frac{i}{\sqrt{2}} \sum_{m,n=1}^{3} (V_e)_{in} \, (Y_e)^*_{mn} \, (U_e)_{jm} \; P_R \,, \tag{14}$$

where $P_{L,R} = \frac{1}{2}\,(1 \mp \gamma_5)$ are the usual chirality projectors. The unitary matrices V_e and U_e are defined in the model file (`Scotogenic.m`) as the matrices that transform between the gauge and mass basis for the left- and right-handed charged leptons, respectively.

We can now consider the $\ell_i^+ - \nu_j - W_\mu^-$ vertex, where again $i, j = 1, 2, 3$ are flavor indices. This is obtained with

```
Vertex[{bar[Fe], Fv, conj[VWp]}]
```

and we find,

$$-\, i \frac{g_2}{\sqrt{2}} \sum_{m=1}^{3} (V_e)_{im} \, (V_\nu)^*_{jm} \; \gamma_\mu P_L \,, \tag{15}$$

where V_ν is the unitary matrix that transforms the left-handed neutrinos from the gauge to the mass basis. Eq. (15) is nothing but the standard charged current interaction in the lepton sector. It is commonly written in terms of the so-called Pontecorvo-Maki-Nakagawa-Sakata (PMNS) matrix, K, defined as [2]

$$K = V_e V_\nu^\dagger \,, \tag{16}$$

thus leading to the vertex

$$i \frac{g_2}{\sqrt{2}} \sum_{m=1}^{3} K_{ij} \, \gamma_\mu P_L \,. \tag{17}$$

[2] A technical note for the experts that might be surprised by Eq. (16). The PMNS matrix, also known as the leptonic mixing matrix, is usually defined as $K = U_\ell^\dagger U_\nu$, where the matrices U_ℓ and U_ν connect gauge (e_L,ν_L) and mass eigenstates (E_L,ν) as $e_L = U_\ell E_L$ and $\nu_L = U_\nu \nu$. Therefore, according to SARAH's conventions, $U_\ell = V_e^\dagger$ and $U_\nu = V_\nu^\dagger$ (see Appendix B), and this is how we get Eq. (16).

Notice that SARAH also identifies when a vertex does not exist. One can see this by computing, for example, the $\nu_i - \chi_j - h$ vertex, with $i, j = 1, 2, 3$ are flavor indices, with the command

```
Vertex[{Fv, Chi, hh}]
```

which simply returns zero due to $\mathbb{Z}_2$ conservation. Instead, if we compute the $\nu_i - \chi_j - \eta_R$ vertex with

```
Vertex[{Fv, Chi, etR}]
```

we find

$$-\frac{i}{\sqrt{2}}\sum_{m,n=1}^{3}(V_\nu)^*_{in}\,(Y_N)_{mn}\,(Z_X)^*_{jm}\;P_L - \frac{i}{\sqrt{2}}\sum_{m,n=1}^{3}(V_\nu)_{in}\,(Y_N)^*_{mn}\,(Z_X)_{jm}\;P_R\,. \tag{18}$$

Renormalization group equations

SARAH also obtains the renormalization group equations (RGEs) for all the parameters of the model. More precisely, the β functions of all parameters are computed in R_ξ gauge at the 1- and 2-loop level. The 1- and 2-loop β functions of the parameter c are defined as

$$\frac{dc}{dt} = \beta_c = \frac{1}{16\pi^2}\beta_c^{(1)} + \frac{1}{(16\pi^2)^2}\beta_c^{(2)}\,, \tag{19}$$

where $t = \log\mu$, μ being the energy scale, and $\beta_c^{(1)}$ and $\beta_c^{(2)}$ are the 1- and 2-loop β functions, respectively.

> **TIP:** All calculations in SARAH are performed in R_ξ gauge. This is useful to check that all physical observables are gauge independent.

The full 2-loop RGEs are computed with the command

```
CalcRGEs[]
```

For non-supersymmetric models this command might take quite a long time to finish. For this reason, and in case one is interested only in the 1-loop β functions, the option TwoLoop turns out to be useful. By setting the option to the value False

```
CalcRGEs[TwoLoop -> False]
```

the calculation becomes much faster. The analytical results for the RGEs are saved in several arrays. In case of non-supersymmetric models (like the one we are studying), these are

- Gij: Anomalous dimensions for all fermions and scalars
- BetaGauge: Gauge couplings
- BetaMuij: Bilinear fermion terms
- BetaBij: Bilinear scalar terms
- BetaTijk: Cubic scalar couplings
- BetaLijkl: Quartic scalar couplings
- BetaYijk: Yukawa couplings
- BetaVEVs: VEVs

Each entry in these arrays contain three elements: the name of the parameter and the 1- and 2-loops β functions. For example, in the array BetaGauge the RGEs for the gauge couplings are saved. In the scotogenic model these are the same as in the SM. Simply by running

```
BetaGauge
```

we find

$$\beta^{(1)}_{g_i} = \left(\frac{21}{5}g_1^3, -3g_2^3, -7g_3^3\right) , \tag{20}$$

with $i = 1, 2, 3$. Two comments are in order: (1) the running g_1 coupling already includes the usual *GUT normalization factor* $\sqrt{5/3}$, and (2) the 2-loop RGEs are zero simply because we decided not to compute them. We note that the GUT normalization factor is not hardcoded, but can be changed by the user when g_1 is defined in the `parameters.m` file. The RGEs for the scalar squared masses are saved in the array `BetaBij`. Therefore, we can execute

```
BetaBij
```

to find, for example, that the 1-loop running of m_η^2 is given by the β function

$$\beta^{(1)}_{m_\eta^2} = -\frac{9}{2}\left(\frac{1}{5}g_1^2 + g_2^2\right) m_\eta^2 + 6\lambda_2 m_\eta^2 - 2\left(2\lambda_3 + \lambda_4\right) m_H^2 + 2\, m_\eta^2 \mathrm{Tr}\left(Y_N Y_N^\dagger\right) - 4\mathrm{Tr}\left(M_N M_N^* Y_N Y_N^\dagger\right) . \tag{21}$$

Here Tr denotes the conventional matrix trace.

These arrays are also saved in external files, so that they can be loaded in other `Mathematica` sessions without the need to compute them again. These files are placed in the directory `$PATH/SARAH-X.Y.Z/Output/Scotogenic/RGEs` and one can easily load them with commands such as

```
BetaGauge = <<$PATH/SARAH-X.Y.Z/Output/Scotogenic/RGEs/BetaGauge.m;
```

Writing all information in LaTeX format

Finally, we can export all this information to LaTeX format so that, after the usual compilation, we obtain a pdf file with all the analytical results derived by `SARAH`. The LaTeX output of `SARAH` is very useful. In addition to being visual and easy to read, we can copy the LaTeX code to our own publications, saving time and getting rid of the tedious task of typing analytical formulas.

We can generate the LaTeX output for our model with the commands

```
ModelOutput[EWSB]
CalcRGEs[TwoLoop -> False]
MakeTeX[]
```

The first line tells `SARAH` to run a long list of computations, saving the results in several directories in `$PATH/SARAH-X.Y.Z/Output/Scotogenic/EWSB`. The second line (`CalcRGEs[TwoLoop -> False]`) is optional. If we do not include it, the resulting LaTeX files will not contain the RGEs. Moreover, if this line has been executed already in the `Mathematica` session we are in, there is no need to execute it again since the RGEs are already computed.

The results of these commands are put in the directory `$PATH/SARAH-X.Y.Z/Output/Scotogenic/EWSB/TeX`. In order to generate the pdf file with all the information, we just have to go to that directory and execute a shell script that is already provided by `SARAH`. This is done with

```
$ cd $PATH/SARAH-X.Y.Z/Output/Scotogenic/EWSB/TeX
$ sh MakePDF.sh
```

Two problems might be encountered when running these commands:

1. In some cases (this is computer-dependent) it might be necessary to make the script executable by assigning the required permissions before we can run it. This is done with the terminal command

```
$ chmod 755 MakePDF.sh
```

2. By default, the pdf file will contain all vertices in the model. These are shown graphically with a Feynman diagram for each vertex. For this purpose, SARAH makes use of the LaTeX package feynmf. This package must be installed in case it is not. For instance, in Debian based systems, this is done with

```
$ sudo apt-get install feynmf
```

As a result of these commands, a pdf file with the name Scotogenic-EWSB.pdf is created in the folder $PATH/SARAH-X.Y.Z/Output/Scotogenic/EWSB/TeX. Now you can simply open it with your favourite pdf viewer.

2.5 Creating input for other computer tools

We have finished our overview of the possibilities offered by SARAH concerning analytical calculations and thus we will start discussing how to obtain reliable numerical results for observables of interest in our model.

Being a Mathematica package, SARAH is not suited for heavy numerical studies. However, it can be used to generate the required input files for other popular computer tools, which can then be used for that purpose. In this course we will focus on three tools:

- SPheno: To compute the mass spectrum, decay rates and flavor observables
- MicrOmegas: To compute the dark matter relic density and other related observables
- MadGraph: To run Monte Carlo simulations for collider studies

> **TIP:** Of course, the preparation of the input files for MicrOmegas and MadGraph does not require SARAH. There are other ways, including the direct writing *by hand*, to create these files. However, I recommend SARAH for three reasons: (1) it saves a lot of time, and (2) it is reliable (since it is automatized), and (3) it is a good idea to have a *central* code to generate all the input files, since this guarantees that all definitions and conventions will be consistent.

SPheno

SPheno is a spectrum calculator: it takes the values of the input parameters and computes, numerically, all masses and mixing matrices in the model. Besides, with this information it also computes the vertices, decay rates and many flavor observables. Although it was originally designed to cover just a few specific supersymmetric models, now it has become available for many other models (including non-supersymmetric ones) thanks to SARAH. The code is written in Fortran. See Sec. 2.6 for details.

The strategy is simple. SPheno has many model-independent numerical routines to perform standard numerical operations such as matrix diagonalization or resolution of differential equations. In order to study our own model, we just have to provide some additional routines with the details of the model. And this is what SARAH can do for us. It creates a group of Fortran files (what we call a SPheno *module*) with all the information of our model. We just have to add these files to the ones already in SPheno, and the resulting code will become a numerical spectrum calculator for our model.

In order to create a SPheno module with SARAH we have to do two things: (1) create a SPheno.m file (already mentioned in Sec. 2.3), and (2) run a command. Let us first show how to prepare the SPheno.m file.

The user can create two SPheno versions:

- **'GUT' version:** the user defines input values for the parameters of the model at some high-energy scale and then they are evolved using the RGEs down to the electroweak scale where the spectrum is computed. This is common practice in supersymmetric models, with boundary conditions at the grand unification (GUT) scale, $m_{\rm GUT} \simeq 2 \cdot 10^{16}$ GeV.
- **Low scale version:** the user defines all input values at the electroweak scale (or SUSY scale in case of supersymmetric models). There is no need for RGE running.

In this course we will show how to create a low scale version of SPheno, since this is the most common practice with non-supersymmetric models. As you will see, this is actually simpler. For instructions about how to create a GUT version we refer to the SARAH manual or to [6].

Since we are interested in a low scale SPheno, the first line of the SPheno.m file must be

SPheno.m

```
1 OnlyLowEnergySPheno = True;
```

When the SPheno code is ready, it will require an input file in LesHouches format to run. The LesHouches format [15,16] distributes the parameters in *blocks*. Some blocks are devoted to lists of parameters, all of them numbers, whereas some blocks are devoted to arrays. In particular, there must be a block called MINPAR where the *minimal parameters* are listed. This may include parameters of the scalar potential, some specific terms or even derived parameters not present in the Lagrangian. Usually, the parameters in the scalar potential are listed here.

When preparing the SPheno.m file we must tell SARAH the list of parameters in the MINPAR block. In the case of the scotogenic model this is done with the following lines

SPheno.m

```
3 MINPAR={
4   {1,lambda1Input},
5   {2,lambda2Input},
6   {3,lambda3Input},
7   {4,lambda4Input},
8   {5,lambda5Input},
9   {6,mEt2Input}
10 };
```

Note that the only parameter of the scalar potential that we have not included in this list is m_H^2. We will see the reason in the next step. We also point out that matrices (like the Yukawa coupling Y_N or the right-handed neutrino Majorana mass M_N) are not listed in the MINPAR block.

Now we must tell SARAH what parameters should be used to solve the tadpole equations. These parameters will not have input values, but instead they will be initialized to the value resulting from the solution of the minimum conditions. In principle, several parameters can be used for this purpose, but it is important to select one that will lead to a simple solution of the tadpole equations. In the scotogenic model the best choice is m_H^2, and this is indicated in the SPheno.m file as

SPheno.m

```
12 ParametersToSolveTadpoles = {mH2};
```

TIP: SARAH makes use of the Mathematica command Solve to solve analytically the tadpole equations. If no solution is found, all the subsequent steps in SPheno will contain errors. For this reason, it is crucial to choose the parameters that will be used to solve the tadpole equations properly. Usually, the best choice is to select bilinear scalar terms, like m_H^2 in the scotogenic model. The reason is that the tadpole equations are linear in them.

In the next step we link the model parameters to the input parameters introduced in the first lines of the file. This is done with

SPheno.m

```
14 BoundaryLowScaleInput={
15   {v, vSM},
16   {Ye, YeSM},
17   {Yd, YdSM},
18   {Yu, YuSM},
19   {g1, g1SM},
20   {g2, g2SM},
21   {g3, g3SM},
22   {lambda1,lambda1Input},
23   {lambda2,lambda2Input},
```

```
  {lambda3,lambda3Input},
  {lambda4,lambda4Input},
  {lambda5,lambda5Input},
  {mEt2,mEt2Input},
  {Yn, LHInput[Yn]},
  {Mn, LHInput[Mn]}
};
```

The first seven entries (from v to g_3) are required for non-supersymmetric models due to technical reasons in the code. We notice that here we also indicated that the matrices Y_N and M_N should have input values. However, the syntaxis is different with respect to the rest of inputs since they are matrices.

Finally, we have to define two lists. These contain the mass eigenstates for which SPheno will compute decay widths and branching ratios. The first list (ListDecayParticles) is for 2-body decays, whereas the second list (ListDecayParticles3B) is for 3-body decays. This is indicated in the SPheno.m file as

SPheno.m

```
ListDecayParticles = {Fu,Fe,Fd,Fv,VZ,VWp,hh,etR,etI,etp,Chi};
ListDecayParticles3B = {{Fu,"Fu.f90"},{Fe,"Fe.f90"},{Fd,"Fd.f90"}};
```

And with this we conclude the SPheno.m file. Now we just have to run, after loading SARAH and initializing our model, the following command in Mathematica

```
MakeSPheno[]
```

After some minutes (depending on the model this can be quite a lengthy process) the SPheno module will be created. This will be located in the directory $PATH/SARAH-X.Y.Z/Output/Scotogenic/EWSB/SPheno.

The installation and usage of SPheno, as well as what to do with the generated SPheno module, will be explained in Sec. 2.6.

MicrOmegas

The most popular public code for dark matter studies is MicrOmegas. With MicrOmegas one can compute several dark matter properties, including the relic density, as well as direct and indirect detection rates. This can be done in many particle physics models. For this purpose, the user just needs to provide model files in CalcHep format [17]. This can be obtained with SARAH via the command

```
MakeCHep[]
```

The folder $PATH/SARAH-X.Y.Z/Output/Scotogenic/EWSB/CHep contains all the files generated with this command. We will have an introduction to MicrOmegas in the third lecture, see Sec. 3, where we will learn how to use them.

MadGraph

MadGraph is a well-known computer tool to run Monte Carlo simulations for collider studies. This code can handle the computation of tree-level and next-to-leading order cross-sections and their matching to parton shower simulations.

The input for MadGraph must use the Universal FeynRules Output (UFO) format [18]. This format can be generated with FeynRules [19], a Mathematica package for the calculation of Feynman rules in models defined by the user. However, we will again use SARAH, which can also export the model information into the required UFO format via the command

```
MakeUFO[]
```

The resulting UFO files are located in $PATH/SARAH-X.Y.Z/Output/Scotogenic/EWSB/UFO. We will learn how to use them with MadGraph in the third lecture, see Sec. 4.

2.6 Brief introduction to SPheno

In this Section we will have a brief introduction to SPheno. A complete overview of all possibilities with this code is beyond the scope of this course, where we will only learn how to use it to obtain the spectrum and compute some flavor observables. For a complete review we refer to the manual [20, 21]. For the calculation of flavor observables we recommend having a look at the FlavorKit manual [13].

SPheno: Technical details, installation and load

- **Name of the tool:** SPheno
- **Author:** Werner Porod (porod@physik.uni-wuerzburg.de) and Florian Staub (florian.staub@cern.ch)
- **Type of code:** Fortran
- **Website:** http://spheno.hepforge.org/
- **Manual:** The original manual can be found in [20]. For a newer version see [21].

Installing SPheno is easy. First, we download the code from the indicated url. Then we copy the tar.gz file to the directory $PATH, where it can be extracted:

```
$ cp Download-Directory/SPheno-X.Y.Z.tar.gz $PATH/
$ cd $PATH
$ tar -xf SPheno-X.Y.Z.tar.gz
```

Here X.Y.Z must be replaced by the SPheno version which has been downloaded. Once this is done, we must copy the SPheno module created with SARAH (see Sec. 2.5). First, we create a directory in $PATH/SPheno-X.Y.Z with the name of the model, Scotogenic in this case,

```
$ cd $PATH/SPheno-X.Y.Z
$ mkdir Scotogenic
```

Then, we copy the SPheno module to this directory

```
$ cp $PATH/SARAH-X.Y.Z/Output/Scotogenic/EWSB/SPheno/* ↪
    $PATH/SPheno-X.Y.Z/Scotogenic
```

Finally, in order to compile the Fortran code we run

```
$ make Model=Scotogenic
```

and wait until the compilation has finished. Once this happens, we are ready to use our SPheno code for the scotogenic model.

Using SPheno

As explained above, SPheno reads an input file in LesHouches format. When the SPheno module is created, SARAH produces two templates in $PATH/SARAH-X.Y.Z/Output/Scotogenic/EWSB/SPheno. Since we have copied them to the SPheno directory, they will also be located in $PATH/SPheno-X.Y.Z/Scotogenic. The files are named LesHouches.in.Scotogenic and LesHouches.in.Scotogenic_low. For non-supersymmetric models there is no difference between them. Therefore, we will just take the first one for our numerical studies in the scotogenic model.

It is convenient to place ourselves in the root directory of SPheno and copy the selected input file to this directory. This is done with the terminal commands

```
$ cd $PATH/SPheno-X.Y.Z
$ cp Scotogenic/LesHouches.in.Scotogenic .
```

If we open the input file we will see that the information (model parameters and `SPheno` flags) is distributed in blocks. The first block that is relevant to us is `MINPAR`. We defined this block in the `SPheno.m` file we created for `SARAH`, and now we can find here the input values for the parameters λ_i, with $i = 1, \ldots, 5$, and m_η^2. All dimensionful parameters (like m_η^2) are assumed to be given in GeV (or powers of GeV).

The templates automatically provided by `SARAH` have something missing. They lack the blocks for the M_N and Y_N matrices. Although technically speaking this is not a *bug* of the code, since they are properly defined in the `SPheno` code with names `MNIN` and `YNIN`, it would be desirable to have these blocks automatically created in the LesHouches templates created by `SARAH`. Nevertheless, the solution is simple: we must add them by hand.

> **TIP:** We must not be suprised by the existence of bugs in the codes we are using. Sometimes, they are well known, although not fixed yet.

The block called `SPhenoInput` includes many `SPheno` options. Each option is given in terms of a flag and value. Comments are also included to help the user identify the meaning of each option. For example, flag number `11` determines whether `SPheno` should calculate decay rates (if we choose the value `1`) or not (if we choose the value `0`). For reasons that will become clear in lecture 2 (devoted to `MicrOmegas`), we will modify the value of flag number `50` to `0`:

LesHouches.in.Scotogenic

```
33 50 0                  # Majorana phases: use only positive masses
```

The rest of the options will be left as given by default for this course, but we encourage to take a look at the `SPheno` manual to learn about their meanings. For simplicity, we will not discuss the rest of blocks. The input values for all the parameters of the model are introduced in the three blocks we have mentioned, `MINPAR`, `MNIN` and `YNIN`, whereas the `SPheno` options are given in the block `SPhenoInput`.

We are ready to introduce input values and run the code. Let us consider the benchmark point **BS1** (Benchmark Scotogenic 1) defined by the input parameters

$$\lambda_1 = 0.25 \qquad \lambda_2 = 0.5 \qquad \lambda_3 = 0.5$$
$$\lambda_4 = -0.5 \qquad \lambda_5 = 8 \cdot 10^{-11} \qquad m_\eta^2 = 1.85 \cdot 10^5\,\text{GeV}^2$$

$$M_N = \begin{pmatrix} 345\,\text{GeV} & 0 & 0 \\ 0 & 4800\,\text{GeV} & 0 \\ 0 & 0 & 6800\,\text{GeV} \end{pmatrix}$$

$$Y_N = \begin{pmatrix} 0.0172495 & 0.300325 & 0.558132 \\ -0.891595 & 1.00089 & 0.744033 \\ -1.39359 & 0.207173 & 0.253824 \end{pmatrix}$$

As we will see later, this benchmark point is experimentally excluded, but it will serve to illustrate how to use the computer tools we are interested in. We must introduce these input values in the corresponding entries in the `LesHouches.in.Scotogenic` file. This results in

LesHouches.in.Scotogenic

```
12 Block MINPAR       # Input parameters
13 1    0.250000E+00     # lambda1Input
14 2    0.500000E+00     # lambda2Input
15 3    0.500000E+00     # lambda3Input
16 4   -0.500000E+00     # lambda4Input
17 5    8.000000E-11     # lambda5Input
18 6    1.850000E+05     # mEt2Input
```

and

LesHouches.in.Scotogenic

```
Block MNIN    #
1 1   3.450000E+02          # Mn(1,1)
1 2   0.000000E+00          # Mn(1,2)
1 3   0.000000E+00          # Mn(1,3)
2 1   0.000000E+00          # Mn(2,1)
2 2   4.800000E+03          # Mn(2,2)
2 3   0.000000E+00          # Mn(2,3)
3 1   0.000000E+00          # Mn(3,1)
3 2   0.000000E+00          # Mn(3,2)
3 3   6.800000E+03          # Mn(3,3)
Block YNIN    #
1 1   1.724950E-02          # Yn(1,1)
1 2   3.003250E-01          # Yn(1,2)
1 3   5.581320E-01          # Yn(1,3)
2 1  -8.915950E-01          # Yn(2,1)
2 2   1.000890E-00          # Yn(2,2)
2 3   7.440330E-01          # Yn(2,3)
3 1  -1.393590E-00          # Yn(3,1)
3 2   2.071730E-01          # Yn(3,2)
3 3   2.538240E-01          # Yn(3,3)
```

The syntaxis is clear. For example, the first line in the block `YNIN` is the value of the real part of $(Y_N)_{11}$. If only these two blocks are present, `SPheno` will assume that M_N and Y_N are real. In principle one could introduce additional blocks for the imaginary parts, but we will not do so in this course. Notice also that all dimensionful parameters in the LesHouches input file are given in GeV or GeV^2.

And now we can execute the code. This is done with

```
$ bin/SPhenoScotogenic
```

And `SPheno` is running! After a few seconds it will be finished and a few output files will be generated in the `SPheno` root directory. We are only interested in the file called `SPheno.spc.Scotogenic`, where all the output information is saved. This type of file is usually called *spectrum file*, since it contains all the details of the mass spectrum of the model.

Again, we can simply open the output file with a text editor and read it. The output values are distributed in blocks, following the LesHouches format. The name of the blocks and the comments added in the output are quite intuitive, making unnecessary a long explanation about what we have in this file. However, let us comment on some particularly important blocks.

After some blocks with the values of all parameters in the model (scalar couplings, Yukawa matrices, ...), we find the block `MASS`.

SPheno.spc.Scotogenic

```
Block MASS  # Mass spectrum
#   PDG code      mass          particle
         25     1.24205442E+02  # hh
       1001     4.30116263E+02  # etR
       1002     4.30116263E+02  # etI
       1003     4.47690732E+02  # etp
         23     9.11887000E+01  # VZ
         24     8.03497269E+01  # VWp
          1     5.00000000E-03  # Fd_1
          3     9.50000000E-02  # Fd_2
```

```
           5      4.18000000E+00   # Fd_3
           2      2.50000000E-03   # Fu_1
           4      1.27000000E+00   # Fu_2
           6      1.73500000E+02   # Fu_3
          11      5.10998930E-04   # Fe_1
          13      1.05658372E-01   # Fe_2
          15      1.77669000E+00   # Fe_3
          12     -8.99592902E-13   # Fv_1
          14     -1.00226011E-11   # Fv_2
          16     -4.79508271E-11   # Fv_3
        1012     -3.45000000E+02   # Chi_1
        1014     -4.80000000E+03   # Chi_2
        1016     -6.80000000E+03   # Chi_3
```

We can read in this block the masses for all mass eigenstates in the model. For example, we can see that in the benchmark point **BS1** the Higgs boson mass is found to be $m_h = 124.2$ GeV. Another detail that is remarkable about these results is the presence of non-zero masses for the neutrinos. Even though we have computed the rest of masses at tree-level, `SPheno` has also included 1-loop corrections to neutrino masses. We see that the lightest neutrino is actually massless (note the vanishing row in Y_N for **BS1**), although `SPheno` returns a tiny (non-zero) numerical value. Finally, we find that some of the Majorana fermion masses are negative. This is just a convention that will be explained in Sec. 3.4.

Next, we find several blocks with mixing matrices. For example:

SPheno.spc.Scotogenic

```
Block UVMIX Q=  1.00000000E+00   # ()
  1  1      6.17693708E-02    # Real(UV(1,1),dp)
  1  2      7.12917774E-01    # Real(UV(1,2),dp)
  1  3     -6.98521863E-01    # Real(UV(1,3),dp)
  2  1      6.66619773E-01    # Real(UV(2,1),dp)
  2  2      4.91405262E-01    # Real(UV(2,2),dp)
  2  3      5.60480996E-01    # Real(UV(2,3),dp)
  3  1      7.42834183E-01    # Real(UV(3,1),dp)
  3  2     -5.00269044E-01    # Real(UV(3,2),dp)
  3  3     -4.44891291E-01    # Real(UV(3,3),dp)
```

This is the neutrino mixing matrix. Note that the resulting values for the mixing angles are in good agreement with current oscillation data. `SPheno` can also compute many quark and lepton flavor observables thanks to the `FlavorKit` extension. The numerical results can be found in the blocks `FlavorKitQFV` and `FlavorKitLFV`. For example, the branching ratios for the radiative lepton decays $\ell_i \to \ell_j \gamma$ are

SPheno.spc.Scotogenic

```
Block FlavorKitLFV # lepton flavor violating observables
     701    1.02041308E-10   # BR(mu->e gamma)
     702    5.89179996E-12   # BR(tau->e gamma)
     703    1.27947536E-09   # BR(tau->mu gamma)
```

Therefore, this point is actually excluded, since it predicts a branching ratio for the radiative decay $\mu \to e\gamma$ above the current experimental limit ($5.7 \cdot 10^{-13}$) established by the MEG experiment. Finally, `SPheno` also computes decay rates. The results are located by the end of the output file. For instance, the Higgs boson branching ratios are found to be

SPheno.spc.Scotogenic

```
DECAY        25     1.62748056E-02   # hh
#     BR                NDA      ID1      ID2
```

```
466       5.94828580E-04    2          22          22    # BR(hh -> VP VP )
467       6.74940484E-02    2          21          21    # BR(hh -> VG VG )
468       4.49830263E-03    2          23          23    # BR(hh -> VZ VZ )
469       4.25758397E-02    2         -24          24    # BR(hh -> VWp^* ↪
              VWp_virt )
470       3.07099640E-04    2          -3           3    # BR(hh -> Fd_2^* Fd_2 )
471       8.28750986E-01    2          -5           5    # BR(hh -> Fd_3^* Fd_3 )
472       1.54859432E-02    2         -15          15    # BR(hh -> Fe_3^* Fe_3 )
473       4.02382779E-02    2          -4           4    # BR(hh -> Fu_2^* Fu_2 )
```

Therefore, we find that the dominant Higgs boson decay in the **BS1** benchmark point is to $b\bar{b}$, as expected for a Higgs mass in the 125 GeV ballpark.

We conclude our brief review of `SPheno` emphasizing that many other things can be done with this code. In combination with `SARAH`, one can create `SPheno` modules for many models and use them to run all kinds of numerical computations.

2.7 Summary of the lecture

In this lecture we learned how to use `SARAH` to study the properties of our favourite model and produce input files for other computer tools. We also had a brief overview of `SPheno`, the numerical tool that complements `SARAH` perfectly. These two codes will be central in the rest of the course and we will make use of the input files produced in this lecture as basis to run `MicrOmegas` and `MadGraph`. Furthermore, we used the scotogenic model as a simple example that allows for an easy introduction to the basics. In lectures 2 and 3 we will continue applying computer tools to this model.

3 Lecture 2: Computing dark matter properties with MicrOmegas

3.1 What is MicrOmegas?

MicrOmegas is probably the most popular computer tool for the study of dark matter. First developed to compute the relic density of a stable massive particle, the code also computes the rates for direct and indirect detection rates of dark matter. Nowadays, many phenomenologists and dark matter model builders use it on a daily basis.

3.2 MicrOmegas: Technical details, installation and load

- **Name of the tool:** MicrOmegas
- **Author:** Geneviève Bélanger, Fawzi Boudjema, Alexander Pukhov and Andrei Semenov. They can be contacted via micromegas@lapth.cnrs.fr.
- **Type of code:** C and Fortran
- **Website:** https://lapth.cnrs.fr/micromegas/
- **Manual:** The manual for the latest version of MicrOmegas can be found in [22]. For previous versions see [23–26].

After downloading the package, one should copy the tar.gz file into the $PATH folder and extract its contents,

```
$ cp Download-Directory/micromegas_X.Y.Z.tar.gz $PATH/
$ cd $PATH
$ tar -xf micromegas_X.Y.Z.tar.gz
```

Here X.Y.Z must be replaced by the MicrOmegas version which has been downloaded. The next step is the compilation of the code, performed with the commands

```
$ cd micromegas_X.Y.Z
$ make
```

And MicrOmegas will be ready to run our own dark matter studies.

3.3 General usage and description of the input files

Before we describe the input files generated by SARAH, it is convenient to explain the general usage of MicrOmegas. This will clarify the role of each file.

Strictly speaking, MicrOmegas is not a code, but a collection of routines for the evaluation of dark matter properties. It contains many model-independent functions and routines, which can be used for the specific models we are studying. The way this is done is quite simple. The user must write a short *steering file*, or main program, that (i) defines options for the DM calculations and output, and (ii) calls the built-in routines in MicrOmegas that run the desired DM calculations. Hence, the user does not need to enter into the details of the MicrOmegas routines, but just call them with the proper options. MicrOmegas will then read the details of the input model (contained in external mdl files, to be provided for each model) and execute the routines, returning the DM properties (such as relic density and detection rates) required by the user in the steering file.

Let us now describe the input files. As already explained in Sec. 2.5, SARAH can produce model files for MicrOmegas. Thanks to this feature, the user gets rid of the most tedious task when working with MicrOmegas. Once generated, these files are located in the directory $PATH/SARAH-X.Y.Z/Output/Scotogenic/EWSB/CHep. One finds the following files:

- CalcOmega.cpp
- CalcOmega_with_DDetection.cpp
- CalcOmega_with_DDetection_old.cpp
- func1.mdl

- `lgrng1.mdl`
- `prtcls1.mdl`
- `vars1.mdl`

As explained above, the `mdl` files define the input model, with details such as particle content, interactions and parameters. On the other hand, the `cpp` files are steering files that tell `MicrOmegas` what dark matter properties we are interested in. Although we will not have to get into the details of these files (since `SARAH` did it for us), let us briefly review their content.

The `mdl` files contain information about the model. In the file `vars1.mdl` one finds the definition of several decay widths of the particles in the model (including the SM ones) and standard parameters like the Fermi constant G_F. The file `func1.mdl` is devoted to the *constrained variables* of the model, this is, to all masses and vertices. The Feynman rules are given in the file `lgrng1.mdl`. Notice that each interaction vertex is given in terms of the interacting states, the Lorentz structure and the value of the vertex itself, using for the former the list of vertices defined in `func1.mdl`. For example, we see that the $\eta_I - \eta_I - h$ Feynman rule is given by the `v0002` vertex, which is defined to be equal to $-(\lambda_3 + \lambda_4 - \lambda_5)\, v$ in `func1.mdl`. Also note that the parameters of the model use the names we introduced in the `parameters.m` file using the `OutputName` option. Finally, the file `prtcls1.mdl` contains information about the particles in the model.

The `cpp` files are the main programs, in this case `C` programs, containing all the calculations we want `MicrOmegas` to perform. We can forget about `CalcOmega_with_DDetection_old.cpp`, which is an old version, no longer required. The difference between the other two `cpp` files is whether direction detection rates should be computed or not.

The file `CalcOmega_with_DDetection.cpp` includes the calculation of direct detection rates. We will talk a little about this possibility in Sec. 3.5. However, for our example we will use the file `CalcOmega.cpp`, which only computes the dark matter relic density, $\Omega_{\rm DM}h^2$.

3.4 Running `MicrOmegas`

In order to implement our model in `MicrOmegas`, we must create a new project and copy the files to the corresponding folder. This is done with

```
$ cd $PATH/micromegas_X.Y.Z
$ ./newProject Scotogenic
$ cd Scotogenic
$ cp $PATH/SARAH-X.Y.Z/Output/Scotogenic/EWSB/CHep/* work/models
```

The next step is the compilation of the selected `cpp` file. However, before we do that, let us notice one thing. In the directory `$PATH/micromegas_X.Y.Z/Scotogenic` one can find two files, `main.c` and `main.F`, with example programs for `MicrOmegas`. They are equivalent to the `cpp` files generated by `SARAH` and contain examples of calculations one can perform with `MicrOmegas`. Therefore, although we will not use them in this course, it might be helpful to take a look at them in order to see the different options and how to turn on specific calculations and outputs in `MicrOmegas`.

In order to compile our own `MicrOmegas` code for the scotogenic model we execute

```
$ mv work/models/CalcOmega.cpp .
$ make main=CalcOmega.cpp
```

This will create the binary file `CalcOmega` in the `Scotogenic` folder. In order to run it and get our results there is only one thing missing: input parameters. For this purpose we will make use of a very convenient feature of `MicrOmegas`: it can read a spectrum file in LesHouches format. Therefore, we can use `SPheno`, run with the input values of our choice, and pass the resulting output file to `MicrOmegas`, which can then read it and compute the DM observables.

> **TIP:** It has become common nowadays to combine different codes. This makes the study of a model a more efficient task, since using the output of a code as input for another code solves many conversion and formatting issues. For this reason, it is convenient to choose computer tools which can be easily combined.

We already learned how to run SPheno in Sec. 2.6. There is only one detail that we must take into account in case we want to use the SPheno output file as input for MicrOmegas. MicrOmegas cannot handle rotation matrices with complex entries. Since these may appear in some calculations with Majorana fermions (like the neutrinos in the scotogenic model), we must tell SPheno that we want numerical results without them. Indeed, it is common to find purely imaginary rotation matrices, or rows of them, in models with Majorana fermions even in the absence of CP violating phases. This type of complex phases can be absorbed by adding a negative sign to the mass of the Majorana fermion. This can be easily understood by looking at the transformation between the mass matrix in the gauge basis (M) and the diagonal mass matrix in the mass basis ($\hat{M}$). For a Majorana fermion, this transformation is of the form

$$V\, M\, V^T = \hat{M}\,, \tag{22}$$

where V is a unitary matrix. It is clear that multiplying a row of the V matrix by the imaginary unit i is equivalent to a change of sign in one eigenvalue of $\hat{M}$. Therefore, it is just a matter of convention whether we present the results with complex rotation matrices and positive masses or with real rotation matrices and negative masses. MicrOmegas can only understand the input if we take the second option, and thus we must tell SPheno to produce an output file with this choice. This is done by setting the flag 50 in the LesHouches input file of SPheno to the value 0, as we already did in Sec. 2.6.

After this comment, we can proceed to run MicrOmegas in the benchmark point **BS1** of the scotogenic model. In order to do this, we must copy the SPheno spectrum file to the Scotogenic folder in MicrOmegas and execute the binary file we just created

```
$ cp $PATH/SPheno-X.Y.Z/SPheno.spc.Scotogenic .
$ ./CalcOmega
```

The first time we run the binary it can take some time, even up to several hours depending on the computer power, since MicrOmegas has to compile all necessary annihilation channels of the DM candidate for that particular parameter point. All further evaluations of similar points are done in a second or less.

When the run is finished, we get the results on the screen:

```
Masses of odd sector Particles:
~N1 : MN1 =  345.0 || ~etI : MetI =  430.1 || ~etR : MetR =  430.1
~etp : Metp =  447.7 || ~N2 : MN2 =  4800.0 || ~N3 : MN3 =  6800.0

Xf=2.41e+01 Omega h^2=1.08e+00

# Channels which contribute to 1/(omega) more than 1%.
# Relative contributions in % are displayed
   28% ~N1 ~N1 ->e3 E3
   21% ~N1 ~N1 ->nu2 nu3
   15% ~N1 ~N1 ->nu2 nu2
    8% ~N1 ~N1 ->e2 E3
    8% ~N1 ~N1 ->E2 e3
    7% ~N1 ~N1 ->nu3 nu3
    4% ~N1 ~N1 ->nu1 nu2
    3% ~N1 ~N1 ->nu1 nu3
    2% ~N1 ~N1 ->e2 E2
```

First, MicrOmegas writes the masses (in GeV) of all particles charged under the $\mathbb{Z}_2$ parity. Note that their names are written including a tilde (~), in contrast to the names of the $\mathbb{Z}_2$-even particles, which do not have it. Since the lightest $\mathbb{Z}_2$-odd particle in the **BS1** point is the lightest right-handed neutrino, N_1, it is stable and constitutes the dark matter of the universe.

Next, MicrOmegas gives us two quantities. $x_f = m_{N_1}/T_f$ characterizes the freeze-out temperature, T_f, and $\Omega_{\rm DM}h^2$ is the dark matter relic density. We see that in the benchmark point **BS1** we obtain $\Omega_{\rm DM}h^2 = 1.08$. This relic density is too high, since the Planck observations prefer a value in the $\Omega_{\rm DM}h^2 \sim 0.11$ ballpark. Therefore, this parameter point is also excluded due to DM constraints.

MicrOmegas also gives a list with the annihilation channels that give the most relevant contributions to the DM relic density. In the **BS1** point, the most important one is

$$N_1 N_1 \to \tau^+ \tau^- \tag{23}$$

which constitutes 28% of the total annihilation cross-section. Note also that flavor violating channels are present as well in the list. For example, we find that the channels

$$N_1 N_1 \to \mu^{\pm} \tau^{\mp} \tag{24}$$

contribute with 16% of the annihilation cross-section.

Finally, note that this information is exported to the external files `omg.out` and `channels.out`. These two files are written following the LesHouches format: each entry is defined by a flag (or a few of them) and a numerical value.

3.5 Other computations in `MicrOmegas`

In the previous Section we learned how to use the `CalcOmega.cpp` file which is automatically provided by `SARAH`. With the aid of this file we can easily compute the DM relic density. However, it is easy to modify these files to (i) change some details of these calculations, (ii) change the amount of information that is shown as output, and (iii) compute additional observables.

We already saw that we can compute direct detection rates with the file `CalcOmega_with_DDetection.cpp`. This file is the main file for a `C` program that uses `MicrOmegas` to calculate the DM relic density $\Omega_{\rm DM} h^2$ as well as some direct detection rates: (i) spin independent cross-section with proton and neutron in pb, (ii) spin dependent cross-section with proton and neutron in pb, (iii) recoil events in the 10 - 50 keV region at ^{73}Ge, ^{131}Xe, ^{23}Na and ^{127}I nuclei. We decided not to use this file in benchmark point **BS1** because, for this parameter point, the DM scattering cross-sections with nucleons is zero at tree-level. Therefore, we would have obtained vanishing direct detection rates.

Just to see how direct detection rates are obtained, let us consider a slight modification of the **BS1** benchmark point. The reason why the benchmark point **BS1** leads to vanishing direct detection rates at tree-level is because the DM particle in this point is the lightest right-handed neutrino and this state does not couple directly to the nucleons. Instead, in a parameter point with scalar DM (η_I), the tree-level scattering cross-section with the nucleons does not vanish. Therefore, let us define a new benchmark point, **BS2** (Benchmark Scotogenic 2), with a lighter η_I state. The only change with respect to the **BS1** point is:

$$m_\eta^2 = 5 \cdot 10^4 \,\text{GeV}^2$$

Using this parameter point is straightforward. We just have to modify a single line in the `MINPAR` block of the `LesHouches.in.Scotogenic` input file:

LesHouches.in.Scotogenic

```
18 6    5.000000E+04     # mEt2Input
```

After running `SPheno` with this modification in the input file, we generate a new `SPheno.spc.Scotogenic` output file that we can use with `MicrOmegas`. We can easily check that this parameter indeed leads to a much lighter η_I state:

SPheno.spc.Scotogenic

```
121 1002      2.23606798E+02   # etI
```

Now we can create the `CalcOmega_with_DDetection` binary. This is completely analogous to what we did for the `CalcOmega` binary:

```
$ mv work/models/CalcOmega_with_DDetection.cpp .
$ make main=CalcOmega_with_DDetection.cpp
```

Once compiled, we copy our new `SPheno.spc.Scotogenic` file and run the `CalcOmega_with_DDetection` binary file,

```
$ cp $PATH/SPheno-X.Y.Z/SPheno.spc.Scotogenic .
$ ./CalcOmega_with_DDetection
```

This is the result that is printed on the screen:

```
Masses of odd sector Particles:
~etI : MetI =   223.6 || ~etR : MetR =   223.6 || ~etp : Metp =   255.8
~N1  : MN1  =   345.0 || ~N2  : MN2  =  4800.0 || ~N3  : MN3  =  6800.0

Xf=2.93e+01 Omega h^2=2.69e-03

# Channels which contribute to 1/(omega) more than 1%.
# Relative contributions in % are displayed
   23% ~etI ~etR ->nu3 nu3
   13% ~etR ~etR ->Wp Wm
   13% ~etI ~etI ->Wp Wm
   11% ~etR ~etR ->nu3 nu3
   11% ~etI ~etI ->nu3 nu3
    5% ~etI ~etR ->nu2 nu2
    4% ~etI ~etR ->nu2 nu3
    3% ~etR ~etR ->Z Z
    3% ~etI ~etI ->Z Z
    3% ~etR ~etR ->nu2 nu2
    3% ~etI ~etI ->nu2 nu2
    2% ~etR ~etR ->nu2 nu3
    2% ~etI ~etI ->nu2 nu3

===== Calculation of CDM-nucleons amplitudes =====
         TREE LEVEL
CDM-nucleon micrOMEGAs amplitudes:
proton:  SI  -1.254E-18  SD  0.000E+00
neutron: SI  -1.273E-18  SD  0.000E+00
         BOX DIAGRAMS
CDM-nucleon micrOMEGAs amplitudes:
proton:  SI  -1.254E-18  SD  0.000E+00
neutron: SI  -1.273E-18  SD  0.000E+00
CDM-nucleon cross sections[pb]:
 proton  SI 6.819E-28  SD 0.000E+00
 neutron SI 7.020E-28  SD 0.000E+00

======== Direct Detection ========
73Ge: Total number of events=1.45E-22 /day/kg
Number of events in 10 - 50 KeV region=7.85E-23 /day/kg
131Xe: Total number of events=2.42E-22 /day/kg
Number of events in 10 - 50 KeV region=1.22E-22 /day/kg
23Na: Total number of events=1.45E-23 /day/kg
Number of events in 10 - 50 KeV region=7.90E-24 /day/kg
I127: Total number of events=2.37E-22 /day/kg
Number of events in 10 - 50 KeV region=1.21E-22 /day/kg
```

We note that this scenario leads to a tiny dark matter relic density, of the order of $2.69 \cdot 10^{-3}$, due to the large annihilation cross-sections into pairs of neutrinos and gauge bosons (W^+W^- and ZZ). In fact, these are not only annihilations, but also co-annihilations with the η_R state, which is almost degenerate in mass. Regarding the calculation of the direct detection cross-sections, the most relevant information is given after `CDM-nucleon cross sections[pb]`. These are the (spin independent and dependent) cross-sections with proton and neutron in pb. It is usually convenient to multiply these values by a factor 10^{-36} to get the cross-sections in cm^2, the units commonly employed by the experimental collaborations. We find that in the **BS2** point these cross-sections are tiny.

Finally, `MicrOmegas` also computes the number of recoil events per day in the 10 - 50 keV region for a kg of ^{73}Ge, ^{131}Xe, ^{23}Na and ^{127}I. Again, and due to the small direct detection cross-sections, these numbers are tiny in the **BS2** point. The largest number of events would be obtained in ^{131}Xe, but even in this case we would expect only $\sim 10^{-22}\,\text{events}\,\text{kg}^{-1}\,\text{day}^{-1}$.

Before concluding the lecture, we emphasize that many other dark matter related observables can be computed using `MicrOmegas`. For a detailed list see the `MicrOmegas` manual [22].

3.6 Summary of the lecture

In this lecture we learned how to use `MicrOmegas` to compute observables related to dark matter physics. Since we had produced the input files with `SARAH`, we did not have to worry about how to write them. Instead, we focused on their practical use to obtain reliable predictions for the DM relic density and direct and indirect detection rates.

4 Lecture 3: LHC physics with MadGraph

4.1 What is MadGraph?

MadGraph is a Monte Carlo event generator for collider studies, nowadays widely used to simulate events at the LHC. Before the LHC era, this tool was used to obtain future predictions in new physics models. Currently, one can also recast the results of the searches published by the LHC collaborations and interpret these analysis in specific models.

The MadGraph software can be extended to incorporate several programs: a random event generator, the code Pythia, used for parton showering and hadronization, and two detector simulators (PGS and Delphes). This suit allows for a complete simulation at the LHC, from events at the parton level to detector response.

Depending on the type of simulation we are interested in, some of these additional pieces might be unnecessary. For example, in case we just want to compute a cross-section at the parton level, it suffices to use the basic MadGraph software. However, if we want to go beyond and include hadronization or detector simulation we will have to use Pythia and PGS or Delphes as well. This may sound a little bit complicated, but in practice the combination of these tools is straightforward, and in fact the MadGraph suit is prepared to do it automatically.

4.2 MadGraph: Technical details, installation and load

- **Name of the tool:** MadGraph (more precisely, in this course we will use MadGraph5_aMC@NLO)
- **Author:** *The MadTeam*, composed by Johan Alwall, Rikkert Frederix, Stefano Frixione, Michel Herquet, Valentin Hirschi, Fabio Maltoni, Olivier Mattelaer, Hua-Sheng Shao, Timothy J. Stelzer, Paolo Torrielli and Marco Zaro. They can be contacted through the MadGraph website.
- **Type of code:** Python
- **Website:** http://madgraph.hep.uiuc.edu/
- **Manual:** See Refs. [27, 28].

The version of MadGraph (MadGraph5_aMC@NLO) we are going to use needs Python version 2.6 or 2.7 and gfortran/gcc 4.6 or higher (in case of NLO calculations). Besides those two requirements, which will be fulfilled in most computers, there is no need for an installation of MadGraph. The only *special* requirement is to register in the MadGraph website before being able to download the latest version of the tool. Once this registration is done and the file is downloaded, we just untar it as usual

```
$ cp Download-Directory/MG5_aMC_vX_Y_Z.tar.gz $PATH/
$ cd $PATH
$ tar -xf MG5_aMC_vX_Y_Z.tar.gz
```

Here X_Y_Z is the version that has been downloaded. And then, in order to load MadGraph we just get into the untarred folder and run the binary file

```
$ cd $PATH/MG5_aMC_vX_Y_Z
$ bin/mg5_aMC
```

This opens MadGraph. In principle, we would be ready to use it. However, before we do so let us configure some details and install additional tools that can be added to the MadGraph suit.

Configuration and installation of additional tools

Let us first comment on how to configure some options in MadGraph. We can see that in the folder input there is a file called mg5_configuration.txt. This file contains the configuration details of MadGraph, including options such as the prefered text editor (which can be user to edit some input files, the so-called *cards*, before starting the simulation), the prefered web browser (some of the results obtained in our MadGraph runs are shown in a user-friendly way with a web browser) or the time given to the user to answer questions by the code (some optoinal calculations can be switched on or off at some intermediate steps in the runs). By default, these are vi, Firefox and 60 seconds. In case you do not like these choices, you can simply modify them by removing

the # symbol in front of `text_editor`, `web_browser` and `timeout`. For example, many people will prefer `emacs` instead of `vi`.

mg5_configuration.txt

```
42 text_editor = emacs
```

Once configured, we can consider adding further pieces to the MadGraph puzzle. With the MadGraph code that we just downloaded, untarred and configured we can run simulations at the parton level. This means that we are interested in *core* processes, such as $e^+e^- \to q\bar{q}$. This is already sufficient for many collider studies, for example those aiming at the determination of an approximate number of expected events for a given process in a given new physics model. However, reality is way more complicated. The products of the process will suffer many complex processes after the collision, such as hadronization and showering, and they must be taken into account in realistic simulations. Furthermore, detectors are not perfect, and their simulation must also take into account many factors, such as inefficiencies.

> **TIP:** Although we will not use some of these additional codes for the moment, it is convenient to install them as soon as possible. This way we will be able to detect incompatibility issues which later, when we have already configured and run MadGraph many times, might be problematic.

For this reason, we may want to install Pythia and PGS. These two additional codes are necessary if you are interested in hadronization and detector response. otherwise, you do not need to install them. However, before we install these two codes we must install a prerequisite tool: ROOT. This popular code is an object oriented framework for large scale data analysis, and has been developed by CERN. In order to take advantage of the rest of the tools, we should install ROOT before. The way this is done is explained in Appendix E.

Once ROOT is already installed, the installation of Pythia and PGS is trivial. We just have to open MadGraph and execute a command:

```
$ bin/mg5_aMC
$ install pythia-pgs
```

This way, MadGraph makes sure that these two codes are installed in the correct locations and makes the required links to combine them in future simulations. However, note that you must be connected to the internet for this command to work.

Last but not least, there is another useful code: MadAnalysis [29]. This independent code, which combines nicely with the MadGraph suit, is devoted to the analysis of our simulations and is usually employed for plotting purposes. The output of our simulations is given in several formats and we must use a tool to *extract* the relevant information (this can be done, for example, with ROOT). MadAnalysis simplifies our life. One can use it to read the MadGraph output and present the results in a graphical way. Since we will interested in such graphical representations, we are going to incorporate it to our MadGraph suit.

Like MadGraph, MadAnalysis does not require any special installation. It only requires to have ROOT properly installed.

```
$ cp Download-Directory/MadAnalysis5_vX.tar.gz $PATH/
$ cd $PATH
$ tar -xf MadAnalysis5_vX.tar.gz
```

Here X is the MadAnalysis version that we have downloaded. Once this is done, MadGraph will be absolutely ready to run our own collider studies.

4.3 General usage and description of the input files

As already explained, the MadGraph suit allows for different levels of sofistication in the simulation:

Events at the parton level (MadEvent) $\Rightarrow$ Showering and hadronization (Pythia) $\Rightarrow$ Detector response (PGS or Delphes)

A complete review of all the possibilities is clearly beyond this course. Fortunately, there are many resources to learn about `MadGraph` in the internet: courses, guides and presentations. The first thing we can do is to run the tutorial provided with `MadGraph`:

```
$ cd $PATH/MG5_aMC_vX_Y_Z
$ bin/mg5_aMC
MG5_aMC> tutorial
```

This tutorial shows the basic steps to simulate $t\bar{t}$ production in the SM. In fact, we should note that when `MadGraph` loads the model that is loaded by default is the SM: we can read on the screen `Loading default model: sm`. The first command of the tutorial is the generation of a new process

```
MG5_aMC> generate p p > t t~
```

This tells `MadGraph` that we want to simulate

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

Next, we must export the process to several formats. This is obtained with the `MadGraph` command

```
MG5_aMC> output MY_FIRST_MG5_RUN
```

This command will do all sorts of things. Among them, it will create a new folder called `MY_FIRST_MG5_RUN`. This is where the information about this process the output of all future runs are saved. There is a sub-folder we should visit. It is the one named `Cards`. It is full of `dat` files with information about the process and about the simulation we are about to begin. There are two files of special relevance to us: `param_card.dat` and `run_card.dat`. The first one contains the input values for all parameters of the model, whereas the second one sets the parameters of the run itself. For example, the user can determine in the `run_card.dat` file the beam type, the energy, the renormalization scale or the number of random events to launch in the Monte Carlo.

The next step in the tutorial is to launch the event generator itself. For this we must execute the command

```
MG5_aMC> launch MY_FIRST_MG5_RUN
```

After we push enter, a question will appear on the screen. `MadGraph` needs to know the type of run. We will have 60 seconds to answer (unless we changed this option in `mg5_configuration.txt`. You can see that the options `pythia` and `pgs` are `OFF`. In order tu run a complete simulation including hadronization and showering (with `Pythia`) and detector response (with `PGS`) we must press `2`. Since `PGS` requires `Pythia` to run before, `MadGraph` will ask for confirmation. We can simply push enter. Then, a new question will appear. `MadGraph` wants to know if we want to modify any of the cards or just prefer to use the ones by default. For the moment we will simply stick to the ones by default in the `Cards` sub-folder. Therefore, we just go ahead by pushing enter. And then the web browser will open. This is a very user-friendly feature of `MadGraph`. As soon as the simulation begins, all information is nicely displayed using a web browser. After some time (not too long) `MadGraph` will be done. We will be able to read the results on the terminal or on the web page shown by the web browser. Either way, we find that the cross-section for $t\bar{t}$ production is about 504.9 ± 0.8 pb. Of course, the exact number might be different (it has been obtained by Monte Carlo methods), but it should agree within the error.

We can also explore these results using the information on the generated web page. The first thing we can see is that `MadGraph` shows the Feynman diagrams used for the computation of the events. These can be found in `Process information`. For $t\bar{t}$ production in the SM these can be classified into two types: gluon-induced and quark-induced diagrams. Moreover, in the first case there are three sub-diagrams. If we click on `Results and event database` we can see the numerical results obtained with the simulation. By clicking on the cross-section result, we can even read the individual contributions given by the different Feynman diagrams. In our case, the dominant production channel is the gluon-induced one.

This concludes the tutorial, where we already run an interesting calculation.

4.4 Computing a cross-section

As an example of what we can do with `MadGraph`, we are going to compute a production cross-section in the scotogenic model. For more fancy simulations we refer to the next lecture, where we will make use of all the tools installed in Sec. 4.2.

Since the scotogenic model is not among the models provided with `MadGraph`, we must add it. In the `MadGraph` folder there is a sub-folder named `models`, where all the model definitions are saved in UFO format. In order to implement the scotogenic model, we just have to copy the files that we produced with `SARAH` in the first lecture. These are located in `$PATH/SARAH-X.Y.Z/Output/Scotogenic/EWSB/UFO`. Therefore, we must execute the following terminal commands:

```
$ cd $PATH/MG5_aMC_vX_Y_Z/models
$ mkdir Scotogenic
$ cp $PATH/SARAH-X.Y.Z/Output/Scotogenic/EWSB/UFO/* Scotogenic
```

Next, we go back to the main `MadGraph` folder, open `MadGraph` and load the model:

```
$ cd ..
$ bin/mg5_aMC
MG5_aMC> import model Scotogenic -modelname
MG5_aMC> define p d1 d1bar d2 d2bar u1 u1bar u2 u2bar g
```

Note that we added the option `-modelname`. This is used to keep the names of the particles as given in the `SARAH` model files. Although sometimes this will not be necessary, it is convenient to use this option when loading models created with `SARAH`. Next, we defined the multiparticle `p`, including the gluon (`g`) and all the SM quarks. This is required because we are not using `MadGraph`'s naming conventions, and thus the multiparticle states must be redefined.

Now we can run a simple simultation where we compute the cross-section for the production of a pair of η scalars, a CP-even neutral one and a charged one,

$$p\,p \to \eta_R\,\eta^+$$

We do this with the command

```
MG5_aMC> generate p p > etr etp
```

Next, we can create the output folder, which we will call `SimScotogenic`, and launch the simulation

```
MG5_aMC> output SimScotogenic
MG5_aMC> launch SimScotogenic
```

To the first question we will simply press enter (equivalent to option `0`), since we just want to compute the cross-section at partonic level. Then we will get the second question, asking about whether we want to modify any cards. And in this case we cannot simply press enter. The `param_card.dat` by default does not correspond to our **BS1** benchmark point. In fact, all the scotogenic model parameters are zero in the file by default. Therefore, we must replace the file by one with the correct input values for the model parameters. Since the `param_card.dat` file uses the standard LesHouches format, we can simply use the `SPheno.spc.Scotogenic` file that we generated with `SPheno` as input for `MadGraph`. This is as simple as typing in the terminal

```
$ cp $PATH/SPheno-X.Y.Z/SPheno.spc.Scotogenic ↪
    $PATH/MG5_aMC_vX/SimScotogenic/Cards/param_card.dat
```

Here, remember, `X.Y.Z` and `X` are the versions of the two codes. This way, we have replaced the `param_card.dat` file by default by one of our own, with the correct parameter values in the **BS1** benchmark point. Once

this is done, we can press enter and continue with the simulation. MadGraph will print some warning messages. These are due to some format issues in the SPheno.spc.Scotogenic file. However, they are not relevant at all. After a few minutes, we will get a result for the cross-section. This is found to be $\sigma = 0.7943 \pm 0.0008$ fb (or something very similar, since this is a random simulation).

We actually expected to get a small cross-section. The $\eta_R\,\eta^+$ states are produced at the LHC by electroweak interactions. This is shown in the Feynman diagrams produced by MadGraph, see Process Information in the web page, where we see that $p\,p \to \eta_R\,\eta^+$ is induced by W^+ s-channel exchange. Therefore, it is not surprising that the resulting cross-section turns out to be much lower than the usual QCD induced cross-sections at the LHC. With an integrated luminosity of 300 fb^{-1}, as expected by the end of the LHC Phase I, we would obtain about 240 events of this type.

MadGraph can be also used to generate all possible chains (Feynman diagrams) leading to a specific final state. For example, in the case we just studied, the η_R and η^+ scalars are not stable, but they decay to final states including the lightest right-handed neutrino N_1. For example, one can have the decays

$$\begin{aligned}\eta_R &\to N_1\,\nu\\ \eta^+ &\to N_1\,\mu^+\end{aligned}$$

where ν is any of the light neutrinos (undistinguishable at the LHC). Therefore, we eventually get the process

$$p\,p \to N_1\,N_1\,\mu^+\,\nu \tag{25}$$

wich would be seen as an antimuon plus missing energy at the LHC. However, the final state $N_1\,N_1\,\mu^+\,\nu$ can be reached by other production mechanisms (not only through intermediate $\eta_R\,\eta^+$). How can we obtain all possible topologies in the scotogenic model leading to $N_1\,N_1\,\mu^+\,\nu$? MadGraph can do it for us.

In fact, in order to see all possible diagrams leading to a specific final state we do not even need to launch the simultation. We just have to generate it. In this case we just need open MadGraph and run a few commands

```
$ bin/mg5_aMC
MG5_aMC> import model Scotogenic -modelname
MG5_aMC> define p d1 d1bar d2 d2bar u1 u1bar u2 u2bar g
MG5_aMC> generate p p > n1 n1 e2bar vl
MG5_aMC> output SimScotogenic2
```

Note that in the generation of the process we have used the multiparticle state vl, containing all light neutrinos. This simulation would take quite a long time before is finished. However, we can already see the Feynman diagrams in the web page generated by MadGraph. We can load it with

```
MG5_aMC> open index.html
```

Then, we just have to browse via Process Information until we find all the Feynman diagrams that participate in this process. The Feynman diagram with intermediate $\eta_R\,\eta^+$ scalars is diagram 5, and it is just one among many. We also find diagrams where only one of them, η_R or η^+, participates.

> **TIP:** By default, MadGraph uses 10^4 events for a simulation. This number will be sufficient in many cases, but not for rare LHC events. We must then use a larger number of events (10^5 or 10^6) when we simulate events with small cross-sections. Otherwise, the resulting simulation will contain large statistical errors.

4.5 Summary of the lecture

MadGraph is one of the most popular computer tools in particle physics, due to its high level of sofistication. In this lecture we learned how to use it to run simple LHC simulations. Since this code offers many more possibilities, clearly beyond the scope of this introductory course, we strongly encourage to explore its potential capabilities. The internet is full of resources to learn how to use MadGraph, from slides to tutorials, including many detailed guides.

5 Lecture 4: Final exercise

5.1 What is this lecture about?

In the final lecture we are going to review our previous lectures by going through the whole process for a new model. For this purpose we will choose the model introduced in [30], described in detail in Sec. D. As we will see, this model has a few additional complications that will be helpful to learn a few more features and possibilities of the computer tools presented in this course.

5.2 Implementing the model in SARAH

First of all, we must implement the model in SARAH. We know already that, in addition to useful analytical results, SARAH can also produce input files for the rest of the codes. Therefore, implementing the model in SARAH is always a practical approach.

The SARAH name of the model will be DarkBS. Since most of the definitions are analogous to the ones in lecture 1, we will only highlight those that require further refinements. All SARAH model files for the DarkBS model can be found in Appendix G.

DarkBS.m

The model is based on the extended gauge symmetry $U(1)_Y \times SU(2)_L \times SU(3)_c \times U(1)_X$. Although the $\mathbb{Z}_2$ parity obtained after symmetry breaking is automatic, we must tell SARAH so that identifies the dark matter candidate,

DarkBS.m

```
14 Global[[1]] = {Z[2], Z2};
```

The definition of the gauge groups must contain an additional piece:

DarkBS.m

```
18 Gauge[[1]]={B,    U[1], hypercharge, g1,False,1};
19 Gauge[[2]]={WB, SU[2], left,         g2,True,1};
20 Gauge[[3]]={G,  SU[3], color,        g3,False,1};
21 Gauge[[4]]={Bp,   U[1], Uchi,        gX,False,1};
```

The additional gauge group must also appear in the definition of the particles in the model. For example, the vector-like fermions are defined as

DarkBS.m

```
32 FermionFields[[6]] = {lL, 1, {v4, e4},    -1/2, 2,  1,  2, 1};
33 FermionFields[[7]] = {lR, 1, {e5, v5},     1/2, 2,  1, -2, 1};
34 FermionFields[[8]] = {qL, 1, {u4, d4},     1/6, 2,  3,  2, 1};
35 FermionFields[[9]] = {qR, 1, {d5, u5},    -1/6, 2, -3, -2, 1};
```

Notice that all right-handed fields have opposite gauge charges to those for the left-handed ones. This is equivalent to identifying, for example, lR with $\overline{L_R}$. As a consequence of this, we must write the components of the right-handed doublets as in a -2 of $U(1)_X$. This practical choice simplifies the writing of the Lagrangian, which takes a more transparent form. For instance, for the Yukawa terms one has

DarkBS.m

```
58 LagHC =  -(Yd conj[H].d.q + Ye conj[H].e.l + Yu H.u.q + mQ qL.qR + mL ↪
       lL.lR + lamQ Phi.qR.q + lamL Phi.lR.l );
```

The additional $U(1)_X$ factor implies the existence of an additional neutral gauge boson. Since this vector will get a mass after the spontaneous breaking of the gauge symmetry, we can identify it with the Z' boson. This is important when defining the gauge sector:

DarkBS.m

```
63 DEFINITION[EWSB][GaugeSector] =
64 {
65   {{VB,VWB[3],VBp},{VP,VZ,VZp},ZZ},
66   {{VWB[1],VWB[2]},{VWp,conj[VWp]},ZW}
67 };
```

The rest of the `DarkBS.m` file goes along the same lines of the `Scotogenic.m` file described in the first lecture. The only detail that should be pointed out is the resulting fermion mixings,

DarkBS.m

```
{{{dL,d4}, {conj[dR],d5}}, {{DL,Vd}, {DR,Ud}}},
{{{uL,u4}, {conj[uR],u5}}, {{UL,Vu}, {UR,Uu}}},
{{{eL,e4}, {conj[eR],e5}}, {{EL,Ve}, {ER,Ue}}},
```

It is important to note that the field that should be written in the basis definition for the charged leptons is `e5`, and not `conj[e5]`. One can easily understand this fact by having a look at the way `eR` and `e5` are defined in the Matter Fields section.

parameters.m

$U(1)$ mixing is a general feature in models with several $U(1)$ factors. `SARAH` can perfectly handle this property, but we must define the mixed gauge couplings in the `parameters.m` file

parameters.m

```
{g1X,       {LaTeX -> "\\tilde{g}",
             LesHouches -> {GAUGE,10},
             OutputName -> g1X}},
{gX1,       {LaTeX -> "\\bar{g}",
             LesHouches -> {GAUGE,11},
             OutputName -> gX1}},
```

For the mixing matrices of the CP-even and CP-odd neutral scalars we have to add the lines

parameters.m

```
{ZH, { Description->"Scalar-Mixing-Matrix",
       LaTeX -> "Z^H",
       Real -> True,
       DependenceOptional ->   {{-Sin[\[Alpha]],Cos[\[Alpha]]},
                                {Cos[\[Alpha]],Sin[\[Alpha]]}},
       Value -> None,
       LesHouches -> SCALARMIX,
       OutputName-> ZH     }},

{ZA, { Description->"Pseudo-Scalar-Mixing-Matrix",
       LaTeX -> "Z^A",
       Real -> True,
       DependenceOptional -> {{-Cos[\[Beta]],Sin[\[Beta]]},
                              {Sin[\[Beta]],Cos[\[Beta]]}},
       Value -> None,
       LesHouches -> PSEUDOSCALARMIX,
       OutputName-> ZA      }},
```

The inclusion of the `Description` options is crucial. This is because it is necessary to properly identify these two matrices since they play a role in some specific calculations (for example, the calculation of the Higgs boson flavor violating decay rate to a pair of leptons, $h \to \ell_i^+ \ell_j^-$). Without this, `SARAH` would not know how to identify these matrices among all the mixing matrices in the model. Notice also that these two mixing matrices have been expressed in terms of the angles α and β.

Finally, the mixing matrix in the neutral gauge sector is also defined in terms of two angles: θ_W and θ'_W:

parameters.m

```
{ZZ, { Description -> "Photon-Z Mixing Matrix",
       Dependence->  {{Cos[ThetaW],-Sin[ThetaW] Cos[ThetaWp], Sin[ThetaW] Sin[ThetaWp]},
                      {Sin[ThetaW],Cos[ThetaW] Cos[ThetaWp],-Cos[ThetaW] Sin[ThetaWp]},
                      {0, Sin[ThetaWp], Cos[ThetaWp]}} }},
```

particles.m

There are just a few details worth pointing out in the `particles.m` file. They all have to do with the same feature in this model: many existing sets of mass eigenstates are now extended to included additional particles. For example, the model has two CP-even neutral scalars,

particles.m

```
{hh    ,  {  Description -> "Higgs",
             PDG -> {25,35},
             PDG.IX -> {101000001,101000002} }},
```

and, for example, four charged leptons,

particles.m

```
{Fe,   { Description -> "Leptons",
         PDG -> {11,13,15,17},
         PDG.IX -> {-110000601,-110000602,-110000603,-110000604} }},
```

It is also very important to define the new Z' boson. This is done with the lines

particles.m

```
  {VZp,   { Description -> "Z'-Boson",
            Goldstone -> Ah[{2}] }},
```

Notice that the option `Description` has been used to take advantage of the general definition of Z' bosons in the file `$PATH/SARAH-X.Y.Z/Models/particles.m`. Moreover, we must indicate the Goldstone boson that constitutes the longitudinal part of the massive Z'. In this case this is given by the second CP-odd neutral scalar, the first one being the Goldstone boson of the SM Z boson.

SPheno.m

Finally, the last model file is `SPheno.m`. We have decided to use again a low scale version of `SPheno`. There are only two details which differ slightly from the `SPheno.m` file we prepared for the scotogenic model. Let us comment on them.

The first comment is about the tadpole equations. In this model there are two scalar fields acquiring a VEV. Therefore, we must solve two tadpole equations and hence select two parameters to solve them:

SPheno.m

```
  ParametersToSolveTadpoles = {mH2, mPhi2};
```

The second comment is about a feature that we can exploit to make our life simpler when we are targeting a specific value of a derived parameter. `SPheno` must have input values for all the parameters of the model in order to run properly. However, we can choose between giving this input *directly* or *indirectly*. In the first case, we introduce the values for the fundamental parameters in the Lagrangian. In the second, we introduce the values for some derived parameters which do not appear directly in the Lagrangian, like a gauge boson mass, and tell `SPheno` (and `SARAH`) how to obtain the fundamental parameters from them. This is useful when we are interested in a parameter point with a specific value for a derived parameter.

In the model under consideration, the Z' mass is an important derived parameter, since the phenomenology strongly depends on its precise value. It depends on two quantities, the new gauge coupling g_X and the $SU(2)_X$ breaking VEV, v_ϕ, via

$$m_{Z'} = 2\, g_X\, v_\phi\,. \tag{26}$$

Given the relevance of the Z' mass, it is useful to replace v_ϕ by $m_{Z'}$ as input parameter. We begin by introducing the Z' mass as one of the input parameters in the `MINPAR` block,

SPheno.m

```
  {20, gXInput},
  {21, MZpMass}
```

And then, among the definitions in `BoundaryLowScaleInput` we establish the relation with v_ϕ,

SPheno.m

```
38   {gX,   gXInput},
39   {g1X,  0},
40   {gX1,  0},
41   {vP,   MZpMass/(2*gX)}
```

This way SPheno will have input values for all the relevant parameters of the model and we will make sure that $m_{Z'}$ has exactly the value we are interested in. Note also that we have considered a scenario with vanishing $U(1)$ mixing by setting the mixed gauge couplings to zero.

5.3 Generating input files for the other tools

Once the model is implemented in SARAH we can generate input files for the other tools. Instead of generating input for the different tools one by one, we can make use of the MakeAll[] command to generate input files for all the tools at once. This will generate automatically the SPheno module as well as input files for MicrOmegas and MadGraph. Therefore, we just have to execute the following three lines in Mathematica:

```
<<$PATH/SARAH-X.Y.Z/SARAH.m;
Start["DarkBS"];
MakeAll[]
```

The results will be saved in different sub-folders of the $PATH/SARAH-X.Y.Z/Output/DarkBS/EWSB folder. Notice that MakeAll[] also includes the generation of the LaTeX files with all the model details.

5.4 Benchmark point and numerical results

The first thing we can do after executing MakeAll[] is to compile our new SPheno code. This step was explained in Sec. 2.6 and the process in this case is completely analogous:

```
$ cd $PATH/SPheno-X.Y.Z
$ mkdir DarkBS
$ cp $PATH/SARAH-X.Y.Z/Output/DarkBS/EWSB/SPheno/* ↪
    $PATH/SPheno-X.Y.Z/DarkBS
$ make Model=DarkBS
```

Next, let us consider a benchmark point for the DarkBS model. We will call it benchmark point **BDarkBS1** (Benchmark DarkBS 1), and it is defined by the input parameters

$$\begin{array}{lll} \lambda = 0.25 & \lambda_\phi = 0.1 & \lambda_\chi = 10^{-5} \\ \lambda_{\phi\chi} = 10^{-5} & \lambda_{H\phi} = 0 & \lambda_{H\chi} = 0 \\ m_\chi^2 = 3\cdot 10^6\,\text{GeV}^2 & m_Q = 1\,\text{TeV} & m_L = 1\,\text{TeV} \\ g_X = 1 & m_{Z'} = 4\,\text{TeV} & \end{array}$$

$$\lambda_Q = \begin{pmatrix} 0 \\ 3\cdot 10^{-3} \\ 3\cdot 10^{-3} \end{pmatrix} \qquad \lambda_L = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

In order to use this parameter point we must include the following lines in the LesHouches.in.DarkBS input file:

LesHouches.in.DarkBS

```
12 Block MINPAR        # Input parameters
13 1    0.250000E+00    # LambdaInput
14 2    0.100000E+00     # LPInput
15 3    0.000010E+00     # LCInput
```

```
4    0.000010E+00     # LCPInput
5    0.000000E+00     # LHPInput
6    0.000000E+00     # LHCInput
10    3.000000E+06    # mChi2Input
11    1.000000E+03    # mQInput
12    1.000000E+03    # mLInput
20    1.000000E+00    # gXInput
21    4.000000E+03    # MZpMass
Block LAMQIN #
1 0.0 #
2 3.0E-3 #
3 3.0E-3 #
Block LAMLIN #
1 0.0 #
2 1.0 #
3 0.0 #
```

By running `SPheno` we can see that the **BDarkBS1** benchmark point has a Higgs mass consistent with the observed value by ATLAS and CMS. Moreover, the `MASS` block also shows that the light neutrinos are massless in this model, whereas the two heavy neutral leptons form a Dirac pair,

`SPheno.spc.DarkBS`

```
         12      0.00000000E+00   # Fv_1
         14      0.00000000E+00   # Fv_2
         16      0.00000000E+00   # Fv_3
         18     -1.73205081E+03   # Fv_4
         20      1.73205081E+03   # Fv_5
```

5.5 Calculating the dark matter relic density

As the next step in our phenomenological study, we can compute the dark matter relic density in the **BDarkBS1** benchmark point using `MicrOmegas`. As explained in Appendix D, the spontaneous breaking of the continuous $U(1)_X$ gauge symmetry leaves a remnant $\mathbb{Z}_2$ that stabilizes the χ scalar. This is therefore the dark matter particle in this model.

The calculation of the dark matter relic density is straightforward and follows the same procedure as for the scotogenic model. Using the files in the `$PATH/SARAH-X.Y.Z/Output/DarkBS/EWSB/CHep` folder, we can proceed in exactly the same way. We find $\Omega_{\rm DM}h^2 = 0.132$, in reasonable agreement with the observed value. The most important annihilation channels are $\chi\,\chi \to d_4\,\bar{d}_4$ and $\chi\,\chi \to u_4\,\bar{u}_4$, this is, to final states including heavy vector-like quarks.

5.6 Signatures at the LHC

Finally, we can use `MadGraph` for some simple (but illustrative) LHC simulations. This model has a Z' boson with a relatively large branching ratio into a pair of muons, $\mu^+\mu^-$. Therefore, let us consider

$$p\,p \to \mu^+\,\mu^-$$

at the LHC. This process will receive many different contributions. Among them, the one induced by s-channel Z' exchange, $p\,p \to Z' \to \mu^+\,\mu^-$. However, note that in the benchmark point **BDarkBS1** the Z' production at the LHC is strongly suppressed, since we have taken $\lambda^1_Q = 0$. Given that protons have very little content of second and third family quarks, this leads to tiny production cross-sections for the Z'. Moreover, we had a Z' mass of 4 TeV, which again suppresses its production. Therefore, let us consider a new benchmark point, called **BDarkBS2** (Benchmark DarkBS 2), where these are changed. The only changes with respect to the **BDarkBS1** point are:

$$m_{Z'} = 300\,\text{GeV} \qquad \lambda_Q = \begin{pmatrix} 1 \\ 3\cdot 10^{-3} \\ 3\cdot 10^{-3} \end{pmatrix}$$

This parameter point is of course experimentally excluded, since such a light and strongly coupled Z' boson would have been observed already at the LHC. However, it serves as an academic illustration. In order to use this parameter point we must modify some lines in the `MINPAR` and `LAMQIN` blocks of the `LesHouches.in.DarkBS` input file:

`LesHouches.in.DarkBS`

```
Block MINPAR       # Input parameters
1    0.250000E+00    # LambdaInput
2    0.100000E+00     # LPInput
3    0.000010E+00     # LCInput
4    0.000010E+00     # LCPInput
5    0.000000E+00     # LHPInput
6    0.000000E+00     # LHCInput
10    3.000000E+06    # mChi2Input
11    1.000000E+03    # mQInput
12    1.000000E+03    # mLInput
20    1.000000E+00    # gXInput
21    3.000000E+02    # MZpMass
Block LAMQIN #
1 1.0 #
2 3.0E-3 #
3 3.0E-3 #
```

After running `SPheno` with this point we generate a new `SPheno.spc.DarkBS` file that we can now use with `MadGraph`. We will follow the same procedure as for the scotogenic model:

```
$ cd ..
$ bin/mg5_aMC
MG5_aMC > import model DarkBS -modelname
MG5_aMC > define p d1 d1bar d2 d2bar u1 u1bar u2 u2bar g
MG5_aMC > generate p p > e2 e2bar
MG5_aMC > output SimDBS
MG5_aMC > launch SimDBS
```

In this simulation we will also include hadronization, showering and detector response effects. This requires running `Pythia` and `PGS`. Therefore, we will answer `2` to the first question, which will require further confirmation by pressing enter. To the second question we will answer `2` in order to modify the `run_card.dat` file. Instead of 10^4 events, we want to generate 10^5. This will imply a more precise simulation. In order to increase the number of events we just have to modify the option `nevents`, which now will read

`run_card.dat`

```
  100000 = nevents ! Number of unweighted events requested
```

We should also modify the `param_card.dat` file before we save and close the `run_card.dat` file. Again, we will simply copy the file we generated with `SPheno` for the **BDarkBS2** benchmark point,

```
$ cp $PATH/SPheno-X.Y.Z/SPheno.spc.DarkBS ↪
    $PATH/MG5_aMC_vX/SimDBS/Cards/param_card.dat
```

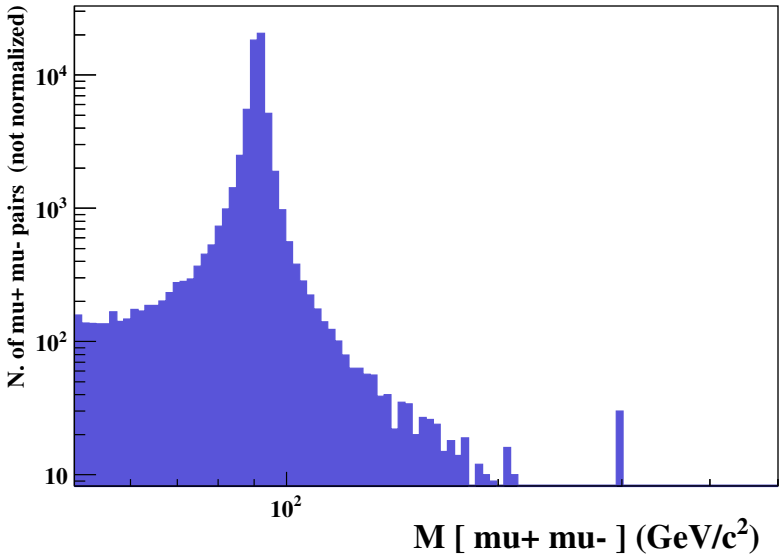

Figure 1: Histogram generated by `MadAnalysis`.

Then we are ready to save and close the `run_card.dat` file and press enter in `MadGraph`. The simulation starts, running the standard simulation at parton level, hadronization and detector response. After some minutes (remember, we are using 10^5 random events) it will be finished, showing a cross-section of about 831 pb. We will now use `MadAnalysis` to plot the results. This powerful tool has many features for the analysis of the `MadGraph` results. One can in principle use it in a similar way as `MadGraph`, opening the code and executing some commands one by one in the therminal. Instead, in this case we will use an external file with teh collection of commands we want `MadAnalysis` to execute. This file can be placed in the `MadAnalysis` folder:

plotDarkBS.txt

```
1 import $PATH/MG5_aMC_vX/SimDBS/Events/run_01/tag_1_pgs_events.lhco.gz
2 plot M(mu+ mu-) 100 50 500 [logX logY]
3 submit DarkBSPlot
```

As usual, `X` must be replaced in the first command by the specific `MadGraph` version we are using. With these commands, we are just telling `MadAnalysis` to import the results of our simulation and to an histogram with the number of $\mu^+\,\mu^-$ events as a function of the invariant mass of the muon pair, $m_{\mu\mu}$. We have also decided to use logarithmic scales in both axes, and show the invariant mass between 50 GeV and 500 GeV distributed in 100 bins. Finally, the result of the analysis should be saved in a folder called `DarkBSPlot`.

Therefore, we just have to call `MadAnalysis` with this input file

```
$ cd $PATH/madanalysis5
$ bin/ma5 -R plotDarkBS.txt
```

The flag `-R` is used because we are going to read a dataset in lhco format, the one used by `PGS`. If we wanted to use the dataset without hadronization and detector response, instead of importing the file `tag_1_pgs_events.lhco.gz` we would have to import `unweighted_events.lhe.gz`, in lhe format, which would not require the `-R` flag.

`MadAnalysis` will ask us the number of cores we want to use for the analysis. After answering the question it will proceed to the compilation of some parts of the code and the analysis of our results. Eventually, it will be finished and the folder `$PATH/madanalysis5/DarkBSPlot` will be created. In this folder we can find the results in several formats. We can open a pdf file where these are nicely presented using the `MadAnalysis` command

```
ma5 > open DarkBSPlot/PDF
```

In the last page of this pdf file (see Fig. 1) we can see the histogram generated by `MadAnalysis`. There are two visible peaks, at $m_{\mu\mu} = 91$ GeV and $m_{\mu\mu} = 300$ GeV. We have *discovered* two particles: the Z and Z' bosons.

5.7 Summary of the lecture

We conclude the course here. In the last lecture we have reviewed our previous lectures by applying what we have learned to a new model with a few additional complications. After implementing the model in `SARAH`, we have used `SPheno` to obtain numerical results for a specific benchmark point, `MicrOmegas` to compute the relic density and `MadGraph` to run a simple but interesting collider study.

6 Summary

In this course we have focused on three computer tools, nowadays widely used in particle physics:

- `SARAH` (and `SPheno`) : analytical and numerical exploration of a new physics model
- `MicrOmegas` : dark matter studies
- `MadGraph` (and `Pythia`, `PGS` and `MadAnalysis`) : collider simulations

For each of these tools, we have learned the basics and trained with some practical applications. For this purpose we have used the scotogenic model as workbench: we computed mass matrices and vertices, obtained numerical results for the particle spectrum, calculated flavor observables, decay rates and the dark matter relic density, and simulated $\eta_R \, \eta^+$ production at the LHC.

In the final lecture we reviewed the whole process with a slightly more complicated model with an additional $U(1)_X$ gauge symmetry. We paid attention to the particularities of the model and how they must be implemented in `SARAH` and used the resulting output to run the model in `MicrOmegas` and `MadGraph`. This way we also computed the dark matter relic density and performed a simple but illustrative LHC simulation.

Before concluding, let us repeat two important messages: (i) using these computers tools is not so hard, and (ii) do not trust blindly in a computer code. I hope I convinced you about the first point, whereas for the second you will get convinced as soon as you find a weird result caused by a bug in a code.

Finally, let me emphasize once more that the computer tools described in this course offer many additional possibilities worth exploring. I could only cover the most basic features of `SARAH`, `MicrOmegas` and `MadGraph`, but there are many more. Some are straightforward and some require to get into technical details. Although getting started is easy, only with frequent practice one can really master all of these computer tools.

Acknowledgements

I thank Cinvestav for giving me the opportunity to give these lectures and all the students and attendants to the course for their presence and fruitful questions. I am very grateful to Omar Miranda, for his kind invitation, and to Diego Aristizábal Sierra, Florian Staub and Nicolás Rojas for their collaboration on the implementation of the models discussed in this course. I also acknowledge fruitful discussions with Diego Restrepo, Alexander Merle and Takashi Toma on some aspects of the scotogenic model relevant for this course. Finally, I feel indebted to Florian Staub for his continuous technical assistance with `SARAH` and related issues.

A Models already implemented in SARAH

The following models are included in the public version of SARAH.

A.1 Supersymmetric Models

- Minimal supersymmetric standard model
  - With general flavor and CP structure (MSSM)
  - Without flavor violation (MSSM/NoFV)
  - With explicit CP violation in the Higgs sector (MSSM/CPV)
  - In SCKM basis (MSSM/CKM)
  - With non-holomorphic soft terms (NHSSM)
- Singlet extensions:
  - Next-to-minimal supersymmetric standard model (NMSSM, NMSSM/NoFV, NMSSM/CPV, NMSSM/CKM)
  - near-to-minimal supersymmetric standard model (near-MSSM)
  - General singlet extended, supersymmetric standard model (SMSSM)
  - DiracNMSSM (DiracNMSSM)
- Triplet extensions
  - Triplet extended MSSM (TMSSM)
  - Triplet extended NMSSM (TNMSSM)
- Models with R-parity violation
  - bilinear RpV (MSSM-RpV/Bi)
  - Lepton number violation (MSSM-RpV/LnV)
  - Only trilinear lepton number violation (MSSM-RpV/TriLnV)
  - Baryon number violation (MSSM-RpV/BnV)
  - $\mu\nu$SSM (munuSSM)
- Additional $U(1)'s$
  - $U(1)$-extended MSSM (UMSSM)
  - secluded MSSM (secluded-MSSM)
  - minimal $B-L$ model (B-L-SSM)
  - minimal singlet-extended $B-L$ model (N-B-L-SSM)
  - $U(1)_L \times U(1)_R$ supersymmetric standard model (BxL-SSM)
- SUSY-scale seesaw extensions
  - inverse seesaw (inverse-Seesaw)
  - linear seesaw (LinSeesaw)
  - singlet extended inverse seesaw (inverse-Seesaw-NMSSM)
  - inverse seesaw with $B-L$ gauge group (B-L-SSM-IS)
  - minimal $U(1)_R \times U(1)_{B-L}$ model with inverse seesaw (BLRinvSeesaw)
- Models with Dirac Gauginos
  - MSSM/NMSSM with Dirac Gauginos (DiracGauginos)
  - minimal R-Symmetric SSM (MRSSM)
  - Minimal Dirac Gaugino supersymmetric standard model (MDGSSM)
- High-scale extensions
  - Seesaw 1 - 3 ($SU(5)$ version) , (Seesaw1,Seesaw2,Seesaw3)
  - Left/right model (ΩLR) (Omega)
  - Quiver model (QEW12, QEWmld2L3)

Field	Group	Coupling
B	$U(1)_Y$	g_1
W	$SU(2)_L$	g_2
g	$SU(3)_c$	g_3

Figure 2: Gauge sector of the Standard Model.

Field	Spin	Generations	$(U(1)_Y \times SU(2)_L \times SU(3)_c)$
H	0	1	$(\frac{1}{2}, \mathbf{2}, \mathbf{1})$
q	$\frac{1}{2}$	3	$(\frac{1}{6}, \mathbf{2}, \mathbf{3})$
ℓ	$\frac{1}{2}$	3	$(-\frac{1}{2}, \mathbf{2}, \mathbf{1})$
d	$\frac{1}{2}$	3	$(\frac{1}{3}, \mathbf{1}, \overline{\mathbf{3}})$
u	$\frac{1}{2}$	3	$(-\frac{2}{3}, \mathbf{1}, \overline{\mathbf{3}})$
e	$\frac{1}{2}$	3	$(1, \mathbf{1}, \mathbf{1})$

Figure 3: Matter content in the Standard Model.

A.2 Non-Supersymmetric Models

- Standard Model (SM) (`SM`), Standard model in CKM basis (`SM/CKM`)
- Two Higgs doublet model (`THDM`), Flipped Two Higgs doublet model (`THDM-Flipped`), Type-II and -III Two Higgs doublet model (`THDM-II` and `THDM-III`), Lepton specific Two Higgs doublet model (`THDM-LS`)
- inert Higgs doublet model (`Inert`)
- B-L extended SM (`B-L-SM`)
- B-L extended SM with inverse seesaw (`B-L-SM-IS`)
- Triplet Extended Standard Model (`TSM`)
- SM extended by a scalar color octet (`SM-8C`)
- Singlet extended SM (`SSM`)
- Singlet Scalar DM (`SSDM`)

B The Standard Model

In order to set the notation and conventions used throughout this course, let us introduce the SM.

The SM gauge symmetry is $SU(3)_c \times SU(2)_L \times U(1)_Y$ and the gauge fields with the corresponding gauge couplings are given in Table 2. The matter content with the corresponding gauge charges are shown in Table 3. Note that our definition of hypercharge is $Q = T_{3L} + Y$. The $SU(2)_L$ doublets can be decomposed as

$$H = \begin{pmatrix} H^+ \\ H^0 \end{pmatrix} \quad , \quad q = \begin{pmatrix} u_L \\ d_L \end{pmatrix} \quad , \quad \ell = \begin{pmatrix} \nu_L \\ e_L \end{pmatrix} , \tag{27}$$

whereas the $SU(2)_L$ singlets can be identified as $d \equiv d_R^*$, $u \equiv u_R^*$ and $e \equiv e_R^*$.

The Yukawa terms in the Lagrangian are

$$\mathcal{L}_Y = Y_d H^\dagger \, \bar{d} \, q + Y_e H^\dagger \, \bar{e} \, \ell + Y_u H \, \bar{u} \, q + \text{h.c.} , \tag{28}$$

where we omit flavor indices for the sake of clarity. $Y_{d,e,u}$ are three 3×3 general complex matrices. The scalar potential of the model takes the form

$$\mathcal{V} = -m_H^2 H^\dagger H + \frac{\lambda}{2} \left(H^\dagger H \right)^2 , \tag{29}$$

where we have defined the m_H^2 with a negative sign for practical reasons. We assume that this scalar potential is such that the neutral component of the Higgs doublet takes a non-zero VEV. In this case one can decompose H^0 as

$$H^0 = \frac{1}{\sqrt{2}} \left(v + h + iA \right) . \tag{30}$$

Here $\langle H^0 \rangle = v/\sqrt{2} = 174$ GeV is the usual Higgs VEV, h is the CP-even state, the physical Higgs boson discovered at the LHC, and A is the CP-odd state that is absorbed by the Z^0 and becomes its longitudinal component.

After electroweak symmetry breaking (EWSB), the remnant symmetry is $SU(3)_c \times U(1)_Q$ and mixing among the gauge bosons is induced

$$\{B, W_3\} \rightarrow \{\gamma, Z\} , \tag{31}$$
$$\{W_1, W_2\} \rightarrow \left\{W^+, W^-\right\} , \tag{32}$$

where $W^\pm$, γ and Z are the mass eigenstates (with $m_\gamma = 0$). The relation between the gauge and mass eigenstates is given by the unitary transformations Z^Z and Z^W, defined as

$$\begin{pmatrix} B \\ W_3 \end{pmatrix} = Z^Z \begin{pmatrix} \gamma \\ Z \end{pmatrix} \tag{33}$$
$$\begin{pmatrix} W_1 \\ W_2 \end{pmatrix} = Z^W \begin{pmatrix} W^+ \\ (W^-)^* \end{pmatrix} , \tag{34}$$

and the mixing matrices are given by

$$Z^Z = \begin{pmatrix} \cos\Theta_W & -\sin\Theta_W \\ \sin\Theta_W & \cos\Theta_W \end{pmatrix} \tag{35}$$
$$Z^W = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} & -\frac{i}{\sqrt{2}} \end{pmatrix} . \tag{36}$$

Regarding the fermions, they also get masses after EWSB. In the basis $(f_L), (f_R^*)$, with $f = d, u, e$, the resulting Dirac mass term is given by

$$m_f = -\frac{1}{\sqrt{2}} v\, Y_f^T , \tag{37}$$

and the gauge and mass eigenstates are related by the unitary transformations $U_{L,R}^f$,

$$U_L = V_u\, u_L , \tag{38}$$
$$U_R = U_u\, u_R , \tag{39}$$
$$D_L = V_d\, d_L , \tag{40}$$
$$D_R = U_d\, d_R , \tag{41}$$
$$E_L = V_e\, e_L , \tag{42}$$
$$E_R = U_e\, e_R , \tag{43}$$

where the fields in capital letters are the mass eigenstates.

C The scotogenic model

The *scotogenic model* [14] is a popular extension of the SM proposed by Ernest Ma in 2006 that includes non-zero neutrino masses and dark matter. One of the main reasons for its popularity is the simplicity of the model which, with just a few ingredients, addresses these two major issues in particle physics and cosmology.

> **TIP:** The other reason for its popularity is its fancy name. The Greek word *skotos* (σκοτος) means *darkness* and refers to the fact that neutrino masses are induced in this model thanks to the presence of the dark matter particles. So, remember, if you want your model to become popular, give it a good name!

In the scotogenic model, the SM particle content is extended with three singlet fermions, N_i (i = 1-3), and one $SU(2)_L$ doublet, η. In addition, a $\mathbb{Z}_2$ parity is imposed, under which the new particles are odd and the SM ones are even. This symmetry not only prevents flavor changing neutral currents but it also renders stable the lightest odd particle in the spectrum, which becomes a dark matter candidate. In this model, two particles can play the role of dark matter: the neutral scalar (an inert Higgs) or the lightest singlet fermion.

The new Lagrangian terms involving the right-handed neutrinos can be written as

$$\mathcal{L}_N = \frac{M_N}{2} \overline{N^c}\, N + Y_N\, \eta\, \overline{N}\, \ell + \text{h.c.} . \tag{44}$$

Field	Spin	Generations	$(U(1)_Y \times SU(2)_L \times SU(3)_c)$	$\mathbb{Z}_2$
η	0	1	$(\frac{1}{2}, \mathbf{2}, \mathbf{1})$	$-$
N	$\frac{1}{2}$	3	$(0, \mathbf{1}, \mathbf{1})$	$-$

Figure 4: New particle content in the scotogenic model.

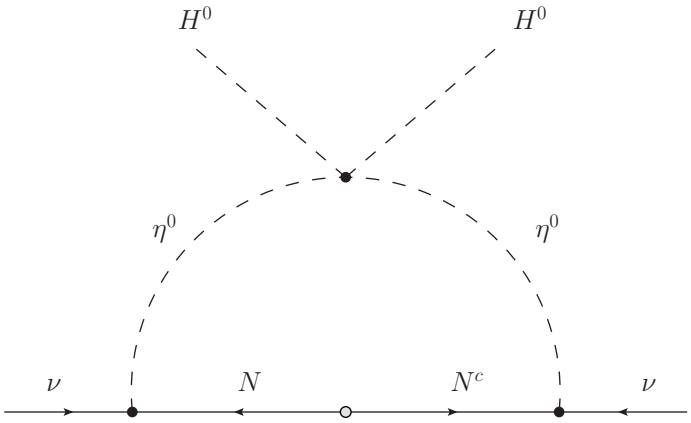

Figure 5: 1-loop neutrino masses in the scotogenic model.

The right-handed neutrino mass matrix M_N can be taken to be diagonal without loss of generality and we will do so in the following discussion. We do not write the kinetic term for the right-handed neutrinos since it takes the canonical form. The matrix of Yukawa couplings, Y_N, is an arbitrary 3×3 complex matrix. We notice that the usual neutrino Yukawa couplings with the SM Higgs doublet are not allowed due to the $\mathbb{Z}_2$ symmetry. The scalar potential of the model is given by

$$\begin{aligned} \mathcal{V} = & -m_H^2 H^\dagger H + m_\eta^2 \eta^\dagger \eta + \frac{\lambda_1}{2}\left(H^\dagger H\right)^2 + \frac{\lambda_2}{2}\left(\eta^\dagger \eta\right)^2 + \lambda_3 \left(H^\dagger H\right)\left(\eta^\dagger \eta\right) \\ & + \lambda_4 \left(H^\dagger \eta\right)\left(\eta^\dagger H\right) + \frac{\lambda_5}{2}\left[\left(H^\dagger \eta\right)^2 + \left(\eta^\dagger H\right)^2\right] . \end{aligned} \tag{45}$$

In the scotogenic model, the $\mathbb{Z}_2$ parity is assumed to be preserved after electroweak symmetry breaking. This is guaranteed by choosing a set of parameters that leads to a vacuum with $\langle \eta \rangle = 0$. Therefore, the only non-zero VEV of the model is the standard SM Higgs VEV,

$$\langle H^0 \rangle = \frac{v}{\sqrt{2}} . \tag{46}$$

After electroweak symmetry breaking, the masses of the charged component η^+ and neutral component $\eta^0 = (\eta_R + i\eta_I)/\sqrt{2}$ are split to

$$m_{\eta^+}^2 = m_\eta^2 + \lambda_3 \langle H^0 \rangle^2 , \tag{47}$$

$$m_R^2 = m_\eta^2 + (\lambda_3 + \lambda_4 + \lambda_5) \langle H^0 \rangle^2 , \tag{48}$$

$$m_I^2 = m_\eta^2 + (\lambda_3 + \lambda_4 - \lambda_5) \langle H^0 \rangle^2 . \tag{49}$$

The mass difference between η_R and η_I (the CP-even and CP-odd components of η^0, respectively) is $m_R^2 - m_I^2 = 2\lambda_5 \langle H^0 \rangle^2$.

Inspecting the new terms in $\mathcal{L}_N$ and $\mathcal{V}$ one finds that the presence of $\lambda_5 \neq 0$ breaks lepton number in two units. Although the usual tree-level contribution to neutrino masses is forbidden by the $\mathbb{Z}_2$ symmetry, these are induced at the 1-loop level as shown in figure 5. This loop is calculable and leads to the neutrino mass matrix [3]

$$\begin{aligned} (m_\nu)_{\alpha\beta} &= \sum_{i=1}^{3} \frac{(Y_N)_{i\alpha}(Y_N)_{i\beta}}{2(4\pi)^2} M_{N_i} \left[\frac{m_R^2}{m_R^2 - M_{N_i}^2} \log\left(\frac{m_R^2}{M_{N_i}^2}\right) - \frac{m_I^2}{m_I^2 - M_{N_i}^2} \log\left(\frac{m_I^2}{M_{N_i}^2}\right)\right] \\ &\equiv \left(Y_N^T \Lambda Y_N\right)_{\alpha\beta} , \end{aligned} \tag{50}$$

[3] We correct this expression by including a factor of 1/2 missing in all references on the scotogenic model. I thank Takashi Toma for pointing out this error in the literature. Notice also that the correct expression was shown in version 1 of [31].

Field	Group	Coupling
B	$U(1)_Y$	g_1
W	$SU(2)_L$	g_2
g	$SU(3)_c$	g_3
B_X	$U(1)_X$	g_X

Figure 6: Gauge sector of the model introduced in [30].

Field	Spin	Generations	$(U(1)_Y \times SU(2)_L \times SU(3)_c \times U(1)_X)$
ϕ	0	1	$(0, \mathbf{1}, \mathbf{1}, 2)$
χ	0	1	$(0, \mathbf{1}, \mathbf{1}, -1)$
$Q_{L,R}$	$\frac{1}{2}$	3	$(\frac{1}{6}, \mathbf{2}, \mathbf{3}, 2)$
$L_{L,R}$	$\frac{1}{2}$	3	$(-\frac{1}{2}, \mathbf{2}, \mathbf{1}, 2)$

Figure 7: New particle content in the model of [30].

where the Λ matrix is defined as $\Lambda = \mathrm{diag}\,(\Lambda_1, \Lambda_2, \Lambda_3)$, with

$$\Lambda_i = \frac{M_{N_i}}{2(4\pi)^2}\left[\frac{m_R^2}{m_R^2 - M_{N_i}^2}\log\left(\frac{m_R^2}{M_{N_i}^2}\right) - \frac{m_I^2}{m_I^2 - M_{N_i}^2}\log\left(\frac{m_I^2}{M_{N_i}^2}\right)\right] . \tag{51}$$

Simplified expressions can be obtained when $m_R^2 \approx m_I^2 \equiv m_0^2$ ($\lambda_5 \ll 1$). In this case the mass matrix in equation (50) can be written as

$$(m_\nu)_{\alpha\beta} \approx \sum_{i=1}^{3} \frac{\lambda_5 \,(Y_N)_{i\alpha}\,(Y_N)_{i\beta}\,\langle H^0\rangle^2}{(4\pi)^2 M_{N_i}}\left[\frac{M_{N_i}^2}{m_0^2 - M_{N_i}^2} + \frac{M_{N_i}^4}{\left(m_0^2 - M_{N_i}^2\right)^2}\log\left(\frac{M_{N_i}^2}{m_0^2}\right)\right] . \tag{52}$$

Compared to the standard seesaw formula, neutrino masses get an additional suppression by roughly the factor $\sim \lambda_5/16\pi^2$. Choosing $\lambda_5 \ll 1$, one can get the correct size for neutrino masses, compatible with singlet fermions at the TeV scale (or below) and sizable Yukawa couplings.

The conservation of $\mathbb{Z}_2$ leads to the existence of a stable particle: the lightest particle charged under $\mathbb{Z}_2$. If neutral, it will constitute a good dark matter candidate. There are, therefore, two dark matter candidates in the scotogenic model: the lightest singlet fermion N_1 and the lightest neutral η scalar (η_R or η_I).

D A model with a dark sector

The model introduced in [30] was motivated by some anomalies in B meson decays recently found by the LHCb collaboration (see [32, 33] for some recent references on the subject). These are not relevant for our course and we will not discuss the details. However, the model will be used to go a step beyond in complexity with respect to the scotogenic model presented in Appendix C.

We extend the SM gauge group with a new dark $U(1)_X$ factor (see Table 6 for details), under which all the SM particles are assumed to be singlets. The only particles charged under the $U(1)_X$ group are two pairs of vector-like fermions, Q and L, as well as the complex scalar fields, ϕ and χ, as shown in Table 7. Q and L are vector-like copies of the SM doublets q and ℓ, and they can be decomposed as

$$Q_{L,R} = \begin{pmatrix} U \\ D \end{pmatrix}_{L,R} \quad , \quad L_{L,R} = \begin{pmatrix} N \\ E \end{pmatrix}_{L,R} . \tag{53}$$

Besides canonical kinetic terms, the new vector-like fermions have Dirac mass terms,

$$\mathcal{L}_m = m_Q \overline{Q} Q + m_L \overline{L} L \,, \tag{54}$$

as well as Yukawa couplings with the SM fermions

$$\mathcal{L}_Y = \lambda_Q \,\overline{Q_R}\,\phi\, q_L + \lambda_L \,\overline{L_R}\,\phi\, \ell_L + \mathrm{h.c.}\,, \tag{55}$$

where λ_Q and λ_L are 3 component vectors. The scalar potential takes the form

$$\mathcal{V} = \mathcal{V}_{\mathrm{SM}} + \mathcal{V}\,(H, \phi, \chi) + \mathcal{V}\,(\phi, \chi) \,. \tag{56}$$

Here $\mathcal{V}_{\mathrm{SM}}$ is the SM scalar potential. The pieces involving the $U(1)_X$ charged scalars are

$$\mathcal{V}\,(H, \phi, \chi) = \lambda_{H\phi}\,|H|^2|\phi|^2 + \lambda_{H\chi}\,|H|^2|\chi|^2 \tag{57}$$

and

$$\begin{aligned}\mathcal{V}(\phi,\chi) &= m_\phi^2|\phi|^2 + m_\chi^2|\chi|^2 + \frac{\lambda_\phi}{2}|\phi|^4 + \frac{\lambda_\chi}{2}|\chi|^4 \\ &\quad + \lambda_{\phi\chi}\,|\phi|^2|\chi|^2 + \left(\mu\,\phi\chi^2 + \text{h.c.}\right) . \end{aligned} \tag{58}$$

We will assume that the scalar potential is such that only the standard Higgs boson and the ϕ field acquire non-zero vacuum expectation values,

$$\langle H^0\rangle = \frac{v}{\sqrt{2}}\,, \qquad \langle\phi\rangle = \frac{v_\phi}{\sqrt{2}}\,. \tag{59}$$

Therefore, the ϕ field will be responsible for the spontaneous breaking of $U(1)_X$, which in turn results into a new massive gauge boson, the Z' boson, with $m_{Z'} = 2g_X v_\phi$, a mixed state of the neutral gauge bosons B, W_3 and B_X. Moreover, the breaking of $U(1)_X$ also induces mixings between the vector-like fermions and their SM counterparts thanks to the Yukawa interactions in Eq. (55). And finally, after spontaneous symmetry breaking, the resulting Lagrangian contains a remnant $\mathbb{Z}_2$ symmetry, under which χ is odd and all the other fields are even. Therefore, χ is a stable neutral scalar, and thus a potentially valid DM candidate. It is worth noting that the mechanism to stabilize the DM particle does not introduce additional ad-hoc symmetries, but simply makes use of the original $U(1)_X$ symmetry of the model. This goal has been achieved by breaking the continuous $U(1)_X$ symmetry to a remnant $\mathbb{Z}_2$, something that can be easily accomplished with a proper choice of $U(1)_X$ charges.

Before concluding our review of the model we must comment on $U(1)$ mixing. It is well known that nothing prevents $U(1)$ factors from mixing. In the model under consideration, this would be given by the Lagrangian term

$$\mathcal{L} \supset \varepsilon\, F^Y_{\mu\nu} F_X^{\mu\nu}\,, \tag{60}$$

where $F^{X,Y}_{\mu\nu}$ are the usual field strength tensors for the $U(1)_{X,Y}$ groups. In the presence of a non-zero ε coupling, kinetic mixing between the $U(1)_X$ and $U(1)_Y$ gauge bosons is induced.

E Installing ROOT

In this Appendix we explain how to install ROOT. The source code can be downloaded from

https://root.cern.ch/drupal/content/downloading-root

Once downloaded, we must untar the file and compile the code:

```
$ cp Download-Directory/root_vX.source.tar.gz $PATH/
$ cd $PATH
$ gzip -dc root_vX.source.tar.gz | tar -xf -
$ cd root-X
$ ./configure
$ make
```

Here X is the ROOT version we have downloaded. In case your computer has more than one core you must replace the last command by

```
$ make -j n
```

where n is the number of cores in your computer. The compilation of ROOT can take a while, so we better find something to do while waiting. Finally, when the compilation finished, we must tell our system how to find ROOT and its libraries. Depending on the system, this must be done differently. For example, in systems with Bash shell (such as most Linux distributions), this is done by adding the lines

bashrc

```
export ROOTSYS=$PATH/root-X
export PATH=$PATH:$ROOTSYS:$ROOTSYS/bin
export LD_LIBRARY_PATH=$ROOTSYS/lib:$LD_LIBRARY_PATH
```

at the end of the `~/.bashrc` file. Here `X` must be replaced by the `ROOT` version, so that `root-X` is the name of the folder where `ROOT` is located.

For more information on how to install `ROOT`:

https://root.cern.ch/drupal/content/installing-root-source

F SARAH model files for the scotogenic model

Scotogenic.m

```
Off[General::spell]

Model`Name      = "Scotogenic";
Model`NameLaTeX = "Scotogenic Model";
Model`Authors   = "N. Rojas, A. Vicente";
Model`Date      = "2015-04-28";

(* "28-04-2015 (first implementation)" *)
(* "25-05-2015 (removed mixings in scalar sector)" *)
(* "10-06-2015 (fixed conventions)" *)


(*------------Particle Content---------------*)

(* Global Symmetries *)
Global[[1]] = {Z[2], Z2};

(*--------------Gauge Groups-----------------*)
Gauge[[1]]={B,   U[1], hypercharge, g1, False, 1};
Gauge[[2]]={WB, SU[2], left,        g2, True , 1};
Gauge[[3]]={G,  SU[3], color,       g3, False, 1};

(*--------------Matter Fields----------------*)
FermionFields[[1]] = {q , 3, {uL, dL},     1/6, 2,  3, 1};
FermionFields[[2]] = {l , 3, {vL, eL},    -1/2, 2,  1, 1};
FermionFields[[3]] = {d , 3, conj[dR],     1/3, 1, -3, 1};
FermionFields[[4]] = {u , 3, conj[uR],    -2/3, 1, -3, 1};
FermionFields[[5]] = {e , 3, conj[eR],       1, 1,  1, 1};
FermionFields[[6]] = {n , 3, conj[nR],       0, 1,  1,-1};

ScalarFields[[1]] =  {H,  1, {Hp, H0},     1/2, 2,  1,  1};
ScalarFields[[2]] =  {Et, 1, {etp,et0},    1/2, 2,  1, -1};

(*---------------DEFINITION------------------*)

NameOfStates={GaugeES, EWSB};

(* ----- Before EWSB ----- *)

DEFINITION[GaugeES][LagrangianInput]=
{
  {LagFer    ,      {AddHC->True}},
  {LagNV     ,      {AddHC->True}},
  {LagH      ,      {AddHC->False}},
  {LagEt     ,      {AddHC->False}},
  {LagHEt    ,      {AddHC->False}},
  {LagHEtHC  ,      {AddHC->True}}
};
```

```
49
50  LagFer   = Yd conj[H].d.q + Ye conj[H].e.l + Yu H.u.q + Yn Et.n.l;
51  LagNV    = Mn/2 n.n;
52  LagH     = -(- mH2 conj[H].H     + 1/2 lambda1 conj[H].H.conj[H].H );
53  LagEt    = -(+ mEt2 conj[Et].Et  + 1/2 lambda2 conj[Et].Et.conj[Et].Et );
54  LagHEt   = -(+ lambda3 conj[H].H.conj[Et].Et + lambda4 ↪
        conj[H].Et.conj[Et].H );
55  LagHEtHC = -(+ 1/2 lambda5 conj[H].Et.conj[H].Et );
56
57  (* Gauge Sector *)
58
59  DEFINITION[EWSB][GaugeSector] =
60  {
61    {{VB,VWB[3]},{VP,VZ},ZZ},
62    {{VWB[1],VWB[2]},{VWp,conj[VWp]},ZW}
63  };
64
65  (* ----- VEVs ---- *)
66
67  DEFINITION[EWSB][VEVs]=
68  {
69    {H0,  {v, 1/Sqrt[2]}, {Ah, \[ImaginaryI]/Sqrt[2]}, {hh, 1/Sqrt[2]}},
70    {et0, {0, 0}, {etI, \[ImaginaryI]/Sqrt[2]}, {etR, 1/Sqrt[2]}}
71  };
72
73  DEFINITION[EWSB][MatterSector]=
74  {
75    {{conj[nR]},{X0, ZX}},
76    {{vL}, {VL, Vv}},
77    {{{dL}, {conj[dR]}}, {{DL,Vd}, {DR,Ud}}},
78    {{{uL}, {conj[uR]}}, {{UL,Vu}, {UR,Uu}}},
79    {{{eL}, {conj[eR]}}, {{EL,Ve}, {ER,Ue}}}
80  };
81
82  (*-------------------------------------------------------*)
83  (* Dirac-Spinors *)
84  (*-------------------------------------------------------*)
85
86  DEFINITION[EWSB][DiracSpinors]=
87  {
88    Fd  -> {  DL, conj[DR]},
89    Fe  -> {  EL, conj[ER]},
90    Fu  -> {  UL, conj[UR]},
91    Fv  -> {  VL, conj[VL]},
92    Chi -> {  X0, conj[X0] }
93  };
94
95  DEFINITION[EWSB][GaugeES]=
96  {
97    Fd1 ->{  FdL, 0},
98    Fd2 ->{  0, FdR},
99    Fu1 ->{  Fu1, 0},
100   Fu2 ->{  0, Fu2},
101   Fe1 ->{  Fe1, 0},
102   Fe2 ->{  0, Fe2}
103 };
```

parameters.m

```
1  (* ::Package:: *)
2
```

```
ParameterDefinitions = {

{g1,        { Description -> "Hypercharge-Coupling"}},
{g2,        { Description -> "Left-Coupling"}},
{g3,        { Description -> "Strong-Coupling"}},

{AlphaS,    {Description -> "Alpha Strong"}},
{e,         { Description -> "electric charge"}},
{Gf,        { Description -> "Fermi's constant"}},
{aEWinv,    { Description -> "inverse weak coupling constant at mZ"}},

{Yu,        { Description -> "Up-Yukawa-Coupling",
              DependenceNum ->  Sqrt[2]/v*{ {Mass[Fu,1],0,0},
                                            {0,Mass[Fu,2],0},
                                            {0,0,Mass[Fu,3]}}}},
{Yd,        { Description -> "Down-Yukawa-Coupling",
              DependenceNum ->  Sqrt[2]/v* {{Mass[Fd,1],0,0},
                                            {0, Mass[Fd,2],0},
                                            {0, 0, Mass[Fd,3]}}}},
{Ye,        { Description -> "Lepton-Yukawa-Coupling",
                           DependenceNum ->  Sqrt[2]/v* {{Mass[Fe,1],0,0},
                                                        {0, Mass[Fe,2],0},
                                                        {0, 0, ↪
                                                            Mass[Fe,3]}}}},

{ThetaW,    { Description -> "Weinberg-Angle",
              DependenceNum -> ArcSin[Sqrt[1 - Mass[VWp]^2/Mass[VZ]^2]]}},

{ZZ, {Description -> "Photon-Z Mixing Matrix"}},
{ZW, {Description -> "W Mixing Matrix", Dependence -> 1/Sqrt[2] {{1, ↪
    1},{I,-I}} }},

{Vu,        {Description ->"Left-Up-Mixing-Matrix"}},
{Vd,        {Description ->"Left-Down-Mixing-Matrix"}},
{Uu,        {Description ->"Right-Up-Mixing-Matrix"}},
{Ud,        {Description ->"Right-Down-Mixing-Matrix"}},
{Ve,        {Description ->"Left-Lepton-Mixing-Matrix"}},
{Ue,        {Description ->"Right-Lepton-Mixing-Matrix"}},

(* Scalar sector *)

{v,          { Description -> "EW-VEV",
               DependenceNum -> Sqrt[4*Mass[VWp]^2/(g2^2)],
               DependenceSPheno -> None  }},

{mH2,        { Description -> "SM Higgs Mass Parameter"}},

{mEt2, {LaTeX -> "m_\\eta^2",
        LesHouches -> {HDM,1},
        OutputName-> mEt2 }},

{lambda1,   {LaTeX -> "\\lambda_1",
             LesHouches -> {HDM,2},
             OutputName-> lam1 }},

{lambda2,   {LaTeX -> "\\lambda_2",
             LesHouches -> {HDM,3},
             OutputName-> lam2 }},

{lambda3,   {LaTeX -> "\\lambda_3",
             LesHouches -> {HDM,4},
```

```
              OutputName-> lam3 }},

{lambda4,   {LaTeX -> "\\lambda_4",
             LesHouches -> {HDM,5},
             OutputName-> lam4 }},

{lambda5,   {Real -> True,
             LaTeX -> "\\lambda_5",
             LesHouches -> {HDM,6},
             OutputName-> lam5 }},

(* Fermion sector *)

{Yn,   {LaTeX -> "Y_N",
        LesHouches -> YN,
        OutputName->Yn }},

{Mn,   {LaTeX -> "M_N",
        LesHouches -> MN,
        OutputName->Mn }},

{ZX, {LaTeX -> "Z^{\\chi^0}",
      LesHouches -> ZXMIX,
      OutputName -> ZX }},

{Vv, {Description ->"Neutrino-Mixing-Matrix"}}

};
```

particles.m

```
(* ::Package:: *)
ParticleDefinitions[GaugeES] = {

      {H0,  {    PDG -> {0},
                 Width -> 0,
                 Mass -> Automatic,
                 FeynArtsNr -> 1,
                 LaTeX -> "H^0",
                 OutputName -> "H0" }},

      {Hp,  {    PDG -> {0},
                 Width -> 0,
                 Mass -> Automatic,
                 FeynArtsNr -> 2,
                 LaTeX -> "H^+",
                 OutputName -> "Hp" }},

      {et0, {    PDG -> {0},
                 Width -> 0,
                 Mass -> Automatic,
                 LaTeX -> "\\eta^0",
                 OutputName -> "et0" }},

      {etp, {    PDG -> {0},
                 Width -> 0,
                 Mass -> Automatic,
                 LaTeX -> "\\eta^+",
                 OutputName -> "etp" }},

      {VB,   { Description -> "B-Boson"}},
```

```
    {VG,   { Description -> "Gluon"}},
    {VWB,  { Description -> "W-Bosons"}},
    {gB,   { Description -> "B-Boson Ghost"}},
    {gG,   { Description -> "Gluon Ghost" }},
    {gWB,  { Description -> "W-Boson Ghost"}}

    };



ParticleDefinitions[EWSB] = {

    {hh   ,  {  Description -> "Higgs",
                PDG -> {25},
                PDG.IX -> {1010000001},
                Mass -> Automatic }},
    {Ah   ,  {  Description -> "Pseudo-Scalar Higgs",
                PDG -> {0},
                PDG.IX ->{0},
                Mass -> {0},
                Width -> {0} }},
    {Hp,     {  Description -> "Charged Higgs",
                PDG -> {0},
                PDG.IX ->{0},
                Width -> {0},
                Mass -> {0},
                LaTeX -> {"H^+","H^-"},
                OutputName -> {"Hp","Hm"} }},

    {etR,   {  Description -> "CP-even eta scalar",
               PDG -> {1001},
               Mass -> LesHouches,
               ElectricCharge -> 0,
               LaTeX -> "\\eta_R",
               OutputName -> "etR" }},
    {etI,   {  Description -> "CP-odd eta scalar",
               PDG -> {1002},
               Mass -> LesHouches,
               ElectricCharge -> 0,
               LaTeX -> "\\eta_I",
               OutputName -> "etI" }},
    {etp,   {  Description -> "Charged eta scalar",
               PDG -> {1003},
               Mass -> LesHouches,
               ElectricCharge -> 1,
               LaTeX -> "\\eta^+",
               OutputName -> "etp" }},

    {VP,   { Description -> "Photon"}},
    {VZ,   { Description -> "Z-Boson", Goldstone -> Ah }},
    {VWp,  { Description -> "W+ - Boson", Goldstone -> Hp}},
    {VG,   { Description -> "Gluon" }},

    {gP,   { Description -> "Photon Ghost"}},
    {gWp,  { Description -> "Positive W+ - Boson Ghost"}},
    {gWpC, { Description -> "Negative W+ - Boson Ghost" }},
    {gZ,   { Description -> "Z-Boson Ghost" }},
    {gG,   { Description -> "Gluon Ghost" }},

    {Fd,   { Description -> "Down-Quarks"}},
    {Fu,   { Description -> "Up-Quarks"}},
```

```
      {Fe,   { Description -> "Leptons" }},
      {Fv,   { Description -> "Neutrinos" }},
      {Chi,  { Description -> "Singlet Fermions",
               PDG -> {1012,1014,1016},
               Mass -> LesHouches,
               ElectricCharge -> 0,
               LaTeX -> "N",
               OutputName -> "N" }}

};

WeylFermionAndIndermediate =
{
   {H,       {LaTeX -> "H"}},
   {Et,      {LaTeX -> "\\eta"}},
   {dR,      {LaTeX -> "d_R" }},
   {eR,      {LaTeX -> "e_R" }},
   {lep,     {LaTeX -> "l" }},
   {uR,      {LaTeX -> "u_R" }},
   {q,       {LaTeX -> "q" }},
   {eL,      {LaTeX -> "e_L" }},
   {dL,      {LaTeX -> "d_L" }},
   {uL,      {LaTeX -> "u_L" }},
   {vL,      {LaTeX -> "\\nu_L" }},
   {DR,      {LaTeX -> "D_R" }},
   {ER,      {LaTeX -> "E_R" }},
   {UR,      {LaTeX -> "U_R" }},
   {EL,      {LaTeX -> "E_L" }},
   {DL,      {LaTeX -> "D_L" }},
   {UL,      {LaTeX -> "U_L" }},
   {X0,      {LaTeX -> "X^0"}},
   {VL,      {LaTeX -> "V_L" }},
   {n,       {LaTeX -> "N" }},
   {nR,      {LaTeX -> "\\nu_R" }}
};
```

SPheno.m

```
OnlyLowEnergySPheno = True;

MINPAR={
  {1,lambda1Input},
  {2,lambda2Input},
  {3,lambda3Input},
  {4,lambda4Input},
  {5,lambda5Input},
  {6,mEt2Input}
};

ParametersToSolveTadpoles = {mH2};

BoundaryLowScaleInput={
  {v, vSM},
  {Ye, YeSM},
  {Yd, YdSM},
  {Yu, YuSM},
  {g1, g1SM},
  {g2, g2SM},
  {g3, g3SM},
  {lambda1,lambda1Input},
  {lambda2,lambda2Input},
```

```
  {lambda3,lambda3Input},
  {lambda4,lambda4Input},
  {lambda5,lambda5Input},
  {mEt2,mEt2Input},
  {Yn, LHInput[Yn]},
  {Mn, LHInput[Mn]}
};

ListDecayParticles = {Fu,Fe,Fd,Fv,VZ,VWp,hh,etR,etI,etp,Chi};
ListDecayParticles3B = {{Fu,"Fu.f90"},{Fe,"Fe.f90"},{Fd,"Fd.f90"}};
```

G SARAH model files for the DarkBS model

DarkBS.m

```
Off[General::spell]

Model`Name = "DarkBS";
Model`NameLaTeX ="DarkBS";
Model`Authors = "D. Sierra, F.Staub, A.Vicente";
Model`Date = "2015-03-11";


(*-----------------------------------------*)
(*   Particle Content*)
(*-----------------------------------------*)

(* Global Symmetries *)
Global[[1]] = {Z[2], Z2};

(* Gauge Groups *)

Gauge[[1]]={B,   U[1], hypercharge, g1,False,1};
Gauge[[2]]={WB, SU[2], left,        g2,True,1};
Gauge[[3]]={G,  SU[3], color,       g3,False,1};
Gauge[[4]]={Bp,  U[1], Uchi,        gX,False,1};


(* Matter Fields *)

FermionFields[[1]] = {q, 3, {uL, dL},     1/6, 2,  3, 0, 1};
FermionFields[[2]] = {l, 3, {vL, eL},    -1/2, 2,  1, 0, 1};
FermionFields[[3]] = {d, 3, conj[dR],     1/3, 1, -3, 0, 1};
FermionFields[[4]] = {u, 3, conj[uR],    -2/3, 1, -3, 0, 1};
FermionFields[[5]] = {e, 3, conj[eR],       1, 1,  1, 0, 1};

FermionFields[[6]] = {lL, 1, {v4, e4},    -1/2, 2,  1,  2, 1};
FermionFields[[7]] = {lR, 1, {e5, v5},     1/2, 2,  1, -2, 1};
FermionFields[[8]] = {qL, 1, {u4, d4},     1/6, 2,  3,  2, 1};
FermionFields[[9]] = {qR, 1, {d5, u5},    -1/6, 2, -3, -2, 1};

ScalarFields[[1]] =  {H, 1, {Hp, H0},   1/2, 2,  1,  0,  1};
ScalarFields[[2]] =  {Phi, 1, phi,        0, 1,  1,  2,  1};
ScalarFields[[3]] =  {Chi, 1, chi,        0, 1,  1, -1, -1};


(*-----------------------------------------------*)
(*   DEFINITION                                  *)
(*-----------------------------------------------*)

```

```
NameOfStates={GaugeES, EWSB};

(* ----- Before EWSB ----- *)

DEFINITION[GaugeES][LagrangianInput]= {
        {LagHC, {AddHC->True}},
        {LagNoHC,{AddHC->False}}
};

LagNoHC = -mH2 conj[H].H - mPhi2 Phi.conj[Phi] - mChi2 Chi.conj[Chi] - 1/2 \[Lambda] conj[H].H.conj[H].H \
-1/2 LamP Phi.conj[Phi].Phi.conj[Phi] -1/2 LamC Chi.conj[Chi].Chi.conj[Chi] \
- LamCP conj[Phi].Phi.conj[Chi].Chi - LamHP conj[H].H.conj[Phi].Phi - LamHC conj[H].H.conj[Chi].Chi;
LagHC =  -(Yd conj[H].d.q + Ye conj[H].e.l + Yu H.u.q + mQ qL.qR + mL lL.lR + lamQ Phi.qR.q + lamL Phi.lR.l );


(* Gauge Sector *)

DEFINITION[EWSB][GaugeSector] =
{
  {{VB,VWB[3],VBp},{VP,VZ,VZp},ZZ},
  {{VWB[1],VWB[2]},{VWp,conj[VWp]},ZW}
};


(* ----- VEVs ---- *)

DEFINITION[EWSB][VEVs]=
{    {H0, {v, 1/Sqrt[2]}, {sigmaH, \[ImaginaryI]/Sqrt[2]},{phiH, 1/Sqrt[2]}},
     {phi, {vP, 1/Sqrt[2]}, {sigmaP, \[ImaginaryI]/Sqrt[2]},{phiP, 1/Sqrt[2]}}      };


DEFINITION[EWSB][MatterSector]=   {
    {{phiH,phiP},{hh,ZH}},
    {{sigmaH,sigmaP},{Ah,ZA}},
    {{{dL,d4}, {conj[dR],d5}}, {{DL,Vd}, {DR,Ud}}},
    {{{uL,u4}, {conj[uR],u5}}, {{UL,Vu}, {UR,Uu}}},
    {{{eL,e4}, {conj[eR],e5}}, {{EL,Ve}, {ER,Ue}}},
    {{vL,v4,v5},{VL,UV}}
};


(*-------------------------------------------------------*)
(* Dirac-Spinors *)
(*-------------------------------------------------------*)

DEFINITION[EWSB][DiracSpinors]={
 Fd ->{  DL, conj[DR]},
 Fe ->{  EL, conj[ER]},
 Fu ->{  UL, conj[UR]},
 Fv ->{  VL, conj[VL]}};

DEFINITION[EWSB][GaugeES]={
 Fd1 ->{  FdL, 0},
 Fd2 ->{  0, FdR},
 Fu1 ->{  Fu1, 0},
```

```
 Fu2 ->{  0, Fu2},
 Fe1 ->{  Fe1, 0},
 Fe2 ->{  0, Fe2}};
```

parameters.m

```
ParameterDefinitions = {

{g1,         { Description -> "Hypercharge-Coupling"}},
{g2,         { Description -> "Left-Coupling"}},
{g3,         { Description -> "Strong-Coupling"}},


{gX,        {LaTeX -> "g_X",
              LesHouches -> {GAUGE,4},
              OutputName -> gX}},

{g1X,        {LaTeX -> "\\tilde{g}",
              LesHouches -> {GAUGE,10},
              OutputName -> g1X}},
{gX1,        {LaTeX -> "\\bar{g}",
              LesHouches -> {GAUGE,11},
              OutputName -> gX1}},


{AlphaS,     {Description -> "Alpha Strong"}},
{e,          { Description -> "electric charge"}},

{Gf,         { Description -> "Fermi's constant"}},
{aEWinv,     { Description -> "inverse weak coupling constant at mZ"}},

{Yu,         { Description -> "Up-Yukawa-Coupling",
               DependenceNum ->  Sqrt[2]/v* {{Mass[Fu,1],0,0},
                                            {0, Mass[Fu,2],0},
                                            {0, 0, Mass[Fu,3]}}}},

{Yd,         { Description -> "Down-Yukawa-Coupling",
               DependenceNum ->  Sqrt[2]/v* {{Mass[Fd,1],0,0},
                                            {0, Mass[Fd,2],0},
                                            {0, 0, Mass[Fd,3]}}}},

{Ye,         { Description -> "Lepton-Yukawa-Coupling",
               DependenceNum ->  Sqrt[2]/v* {{Mass[Fe,1],0,0},
                                            {0, Mass[Fe,2],0},
                                            {0, 0, Mass[Fe,3]}}}},

{\[Lambda],  { Description -> "SM Higgs Selfcouplings",
               DependenceNum -> Mass[hh]^2/(2 v^2)}},
{v,           { Description -> "EW-VEV",
               DependenceNum -> Sqrt[4*Mass[VWp]^2/(g2^2)],
               DependenceSPheno -> None  }},

{vP,    {LaTeX ->"v_\\phi",
         OutputName -> vP,
         LesHouches -> {DBS,20}}},

{mPhi2, {LaTeX -> "m_{\\phi}^2",
          OutputName->mPhi2,
          LesHouches ->  {DBS,1}}},

{mChi2, {LaTeX -> "m_{\\chi}^2",
```

```
          OutputName->mX2,
          LesHouches ->  {DBS,2}}},

{mQ, {LaTeX -> "m_Q",
      OutputName->mQ,
      LesHouches ->  {DBS,3}}},

{mL, {LaTeX -> "m_L",
      OutputName->mL,
      LesHouches ->  {DBS,4}}},


{LamP, {LaTeX -> "\\lambda_{\\phi}",
        OutputName->LamP,
        LesHouches ->  {DBS,10}}},

{LamC, {LaTeX -> "\\lambda_{\\chi}",
        OutputName->LamC,
        LesHouches ->  {DBS,11}}},

{LamCP, {LaTeX -> "\\lambda_{\\phi\\chi}",
         OutputName->LamCP,
         LesHouches ->  {DBS,12}}},

{LamHP, {LaTeX -> "\\lambda_{H\\phi}",
         OutputName->LamHP,
         LesHouches ->  {DBS,13}}},

{LamHC, {LaTeX -> "\\lambda_{H\\chi}",
         OutputName->LamHC,
         LesHouches ->  {DBS,14}}},

{lamQ, {LaTeX -> "\\lambda_Q",
        OutputName->lamQ,
        LesHouches ->  LAMQ}},

{lamL, {LaTeX -> "\\lambda_L",
        OutputName->lamL,
        LesHouches ->  LAML}},


{mH2,       { Description -> "SM Higgs Mass Parameter"}},

{ThetaW,    { Description -> "Weinberg-Angle",
              DependenceNum -> ArcSin[Sqrt[1 - Mass[VWp]^2/Mass[VZ]^2]]  }},

{ThetaWp,  { LaTeX -> "{\\Theta'}_W",
             Real ->True,
             DependenceSPheno -> ArcCos[Abs[ZZ[3,3]]],
             OutputName-> TWp,
             LesHouches -> {ANGLES,10}      }},


{ZH, { Description->"Scalar-Mixing-Matrix",
       LaTeX -> "Z^H",
       Real -> True,
       DependenceOptional ->   {{-Sin[\[Alpha]],Cos[\[Alpha]]},
                                {Cos[\[Alpha]],Sin[\[Alpha]]}},
       Value -> None,
       LesHouches -> SCALARMIX,
       OutputName-> ZH     }},
```

```
117
118 {ZA, { Description->"Pseudo-Scalar-Mixing-Matrix",
119        LaTeX -> "Z^A",
120        Real -> True,
121        DependenceOptional -> {{-Cos[\[Beta]],Sin[\[Beta]]},
122                               {Sin[\[Beta]],Cos[\[Beta]]}},
123        Value -> None,
124        LesHouches -> PSEUDOSCALARMIX,
125        OutputName-> ZA      }},
126
127 {\[Beta],   { Description -> "Pseudo Scalar mixing angle",
128               DependenceSPheno -> ArcSin[Abs[ZH[1,2]]]  }},
129
130 {\[Alpha],  { Description -> "Scalar mixing angle" }},
131
132 {UV, { Description ->"Neutrino-Mixing-Matrix",
133        LaTeX -> "U^V",
134        Dependence ->  None,
135        Value -> None,
136        LesHouches -> UVMIX,
137        OutputName-> UV      }},
138
139 {ZZ, { Description -> "Photon-Z Mixing Matrix",
140        Dependence->  {{Cos[ThetaW],-Sin[ThetaW] Cos[ThetaWp], Sin[ThetaW] ↪
               Sin[ThetaWp]},
141                      {Sin[ThetaW],Cos[ThetaW] Cos[ThetaWp],-Cos[ThetaW] ↪
                          Sin[ThetaWp]},
142                      {0, Sin[ThetaWp], Cos[ThetaWp]}} }},
143
144 {ZW, { Description -> "W Mixing Matrix",
145        Dependence ->   1/Sqrt[2] {{1, 1},
146                                   {\[ImaginaryI],-\[ImaginaryI]}} }},
147
148 {Vu,         {Description ->"Left-Up-Mixing-Matrix"}},
149 {Vd,         {Description ->"Left-Down-Mixing-Matrix"}},
150 {Uu,         {Description ->"Right-Up-Mixing-Matrix"}},
151 {Ud,         {Description ->"Right-Down-Mixing-Matrix"}},
152 {Ve,         {Description ->"Left-Lepton-Mixing-Matrix"}},
153 {Ue,         {Description ->"Right-Lepton-Mixing-Matrix"}}
154
155  };
```

particles.m

```
1  ParticleDefinitions[GaugeES] = {
2        {H0,  {    PDG -> {0},
3                   Width -> 0,
4                   Mass -> Automatic,
5                   FeynArtsNr -> 1,
6                   LaTeX -> "H^0",
7                   OutputName -> "H0" }},
8
9
10       {Hp,  {    PDG -> {0},
11                  Width -> 0,
12                  Mass -> Automatic,
13                  FeynArtsNr -> 2,
14                  LaTeX -> "H^+",
15                  OutputName -> "Hp" }},
16
17
```

```
18
19        {VB,    { Description -> "B-Boson"}},
20        {VG,    { Description -> "Gluon"}},
21        {VWB,   { Description -> "W-Bosons"}},
22        {gB,    { Description -> "B-Boson Ghost"}},
23        {gG,    { Description -> "Gluon Ghost" }},
24        {gWB,   { Description -> "W-Boson Ghost"}}
25
26        };
27
28  ParticleDefinitions[EWSB] = {
29
30
31      {hh    ,  {  Description -> "Higgs",
32                   PDG -> {25,35},
33                   PDG.IX -> {101000001,101000002} }},
34
35      {Ah    ,  {  Description -> "Pseudo-Scalar Higgs",
36                   PDG -> {0,0},
37                   PDG.IX ->{0,0},
38                   Mass -> {0,0},
39                   Width -> {0,0} }},
40
41
42       {Hp,      { Description -> "Charged Higgs",
43                   PDG -> {0},
44                   PDG.IX ->{0},
45                   Width -> {0},
46                   Mass -> {0},
47                   LaTeX -> {"H^+","H^-"},
48                   OutputName -> {"Hp","Hm"} }},
49
50        {VP,    { Description -> "Photon"}},
51        {VZ,    { Description -> "Z-Boson",
52                  Goldstone -> Ah[{1}] }},
53        {VZp,    { Description -> "Z'-Boson",
54                   Goldstone -> Ah[{2}] }},
55        {VG,    { Description -> "Gluon" }},
56        {VWp,   { Description -> "W+ - Boson",
57                  Goldstone -> Hp }},
58        {gP,    { Description -> "Photon Ghost"}},
59        {gWp,   { Description -> "Positive W+ - Boson Ghost"}},
60        {gWpC,  { Description -> "Negative W+ - Boson Ghost" }},
61        {gZ,    { Description -> "Z-Boson Ghost" }},
62        {gZp,   { Description -> "Z'-Ghost" }},
63        {gG,    { Description -> "Gluon Ghost" }},
64
65        {chi, { LaTeX -> "\\chi",
66                PDG -> {50},
67                OutputName -> "chi",
68                ElectricCharge->0}},
69
70        {Fd,    { Description -> "Down-Quarks",
71                  PDG -> {1,3,5,7},
72                  PDG.IX->{-110890201,-110890202,-110890203,-110890204} }},
73        {Fu,    { Description -> "Up-Quarks",
74                  PDG -> {2,4,6,8},
75                  PDG.IX->{110100401,110100402,110100403,110100404} }},
76        {Fe,    { Description -> "Leptons",
77                  PDG -> {11,13,15,17},
78                  PDG.IX -> {-110000601,-110000602,-110000603,-110000604} }},
```

```
      {Fv,   { Description -> "Neutrinos",
               PDG -> {12,14,16,18,20},
               PDG.IX ↪
                    ->{-110000001,-110000002,-110000003,-110000004,-110000005} ↪
                    }}

};


WeylFermionAndIndermediate = {

    {H,       {   PDG -> {0},
                  Width -> 0,
                  Mass -> Automatic,
                  LaTeX -> "H",
                  OutputName -> "" }},

   {dR,     {LaTeX -> "d_R" }},
   {eR,     {LaTeX -> "e_R" }},
   {lep,    {LaTeX -> "l" }},
   {uR,     {LaTeX -> "u_R" }},
   {q,      {LaTeX -> "q" }},
   {eL,     {LaTeX -> "e_L" }},
   {dL,     {LaTeX -> "d_L" }},
   {uL,     {LaTeX -> "u_L" }},
   {vL,     {LaTeX -> "\\nu_L" }},

   {DR,     {LaTeX -> "D_R" }},
   {ER,     {LaTeX -> "E_R" }},
   {UR,     {LaTeX -> "U_R" }},
   {EL,     {LaTeX -> "E_L" }},
   {DL,     {LaTeX -> "D_L" }},
   {UL,     {LaTeX -> "U_L" }}
};
```

SPheno.m

```
OnlyLowEnergySPheno = True;

MINPAR={
  {1,LambdaInput},
  {2,LPInput},
  {3,LCInput},
  {4,LCPInput},
  {5,LHPInput},
  {6,LHCInput},
  {10, mChi2Input},
  {11, mQInput},
  {12, mLInput},
  {20, gXInput},
  {21, MZpMass}
};

ParametersToSolveTadpoles = {mH2, mPhi2};

BoundaryLowScaleInput={
  {v, vSM},
  {Ye, YeSM},
  {Yd, YdSM},
  {Yu, YuSM},
  {g1, g1SM},
```

```
  {g2, g2SM},
  {g3, g3SM},
  {lamQ,       LHInput[lamQ]},
  {lamL,       LHInput[lamL]},
  {\[Lambda], LambdaInput},
  {LamP,    LPInput},
  {LamC,    LCInput},
  {LamCP,   LCPInput},
  {LamHP,   LHPInput},
  {LamHC,   LHCInput},
  {mChi2,   mChi2Input},
  {mQ,      mQInput},
  {mL,      mLInput},
  {gX,      gXInput},
  {g1X, 0},
  {gX1, 0},
  {vP, MZpMass/(2*gX)}
};

ListDecayParticles = {Fu, Fe, Fd, hh, VZp};
ListDecayParticles3B = {{Fu,"Fu.f90"},{Fe,"Fe.f90"},{Fd,"Fd.f90"}};
```

References

[1] T. Hahn, "Generating Feynman diagrams and amplitudes with FeynArts 3," *Comput.Phys.Commun.* **140** (2001) 418–431, arXiv:hep-ph/0012260 [hep-ph].

[2] A. Pukhov, "CalcHEP 2.3: MSSM, structure functions, event generation, batchs, and generation of matrix elements for other packages," arXiv:hep-ph/0412191 [hep-ph].

[3] E. Boos, M. Dubinin, V. Ilyin, A. Pukhov, and V. Savrin, "CompHEP: Specialized package for automatic calculations of elementary particle decays and collisions," arXiv:hep-ph/9503280 [hep-ph].

[4] W. Kilian, T. Ohl, and J. Reuter, "WHIZARD: Simulating Multi-Particle Processes at LHC and ILC," *Eur.Phys.J.* **C71** (2011) 1742, arXiv:0708.4233 [hep-ph].

[5] M. Moretti, T. Ohl, and J. Reuter, "O'Mega: An Optimizing matrix element generator," arXiv:hep-ph/0102195 [hep-ph].

[6] F. Staub, "Exploring new models in all detail with SARAH," arXiv:1503.04200 [hep-ph].

[7] F. Staub, "SARAH," arXiv:0806.0538 [hep-ph].

[8] F. Staub, "From Superpotential to Model Files for FeynArts and CalcHep/CompHep," *Comput.Phys.Commun.* **181** (2010) 1077–1086, arXiv:0909.2863 [hep-ph].

[9] F. Staub, "Automatic Calculation of supersymmetric Renormalization Group Equations and Self Energies," *Comput.Phys.Commun.* **182** (2011) 808–833, arXiv:1002.0840 [hep-ph].

[10] F. Staub, "SARAH 3.2: Dirac Gauginos, UFO output, and more," *Computer Physics Communications* **184** (2013) pp. 1792–1809, arXiv:1207.0906 [hep-ph].

[11] F. Staub, "SARAH 4: A tool for (not only SUSY) model builders," *Comput.Phys.Commun.* **185** (2014) 1773–1790, arXiv:1309.7223 [hep-ph].

[12] F. Staub, T. Ohl, W. Porod, and C. Speckner, "A Tool Box for Implementing Supersymmetric Models," *Comput.Phys.Commun.* **183** (2012) 2165–2206, arXiv:1109.5147 [hep-ph].

[13] W. Porod, F. Staub, and A. Vicente, "A Flavor Kit for BSM models," *Eur.Phys.J.* **C74** no. 8, (2014) 2992, arXiv:1405.1434 [hep-ph].

[14] E. Ma, "Verifiable radiative seesaw mechanism of neutrino mass and dark matter," *Phys.Rev.* **D73** (2006) 077301, arXiv:hep-ph/0601225 [hep-ph].

[15] P. Z. Skands, B. Allanach, H. Baer, C. Balazs, G. Belanger, *et al.*, "SUSY Les Houches accord: Interfacing SUSY spectrum calculators, decay packages, and event generators," *JHEP* **0407** (2004) 036, arXiv:hep-ph/0311123 [hep-ph].

[16] B. Allanach, C. Balazs, G. Belanger, M. Bernhardt, F. Boudjema, *et al.*, "SUSY Les Houches Accord 2," *Comput.Phys.Commun.* **180** (2009) 8–25, arXiv:0801.0045 [hep-ph].

[17] A. Belyaev, N. D. Christensen, and A. Pukhov, "CalcHEP 3.4 for collider physics within and beyond the Standard Model," *Comput.Phys.Commun.* **184** (2013) 1729–1769, arXiv:1207.6082 [hep-ph].

[18] C. Degrande, C. Duhr, B. Fuks, D. Grellscheid, O. Mattelaer, *et al.*, "UFO - The Universal FeynRules Output," *Comput.Phys.Commun.* **183** (2012) 1201–1214, arXiv:1108.2040 [hep-ph].

[19] A. Alloul, N. D. Christensen, C. Degrande, C. Duhr, and B. Fuks, "FeynRules 2.0 - A complete toolbox for tree-level phenomenology," *Comput.Phys.Commun.* **185** (2014) 2250–2300, arXiv:1310.1921 [hep-ph].

[20] W. Porod, "SPheno, a program for calculating supersymmetric spectra, SUSY particle decays and SUSY particle production at e+ e- colliders," *Comput.Phys.Commun.* **153** (2003) 275–315, arXiv:hep-ph/0301101 [hep-ph].

[21] W. Porod and F. Staub, "SPheno 3.1: Extensions including flavour, CP-phases and models beyond the MSSM," *Comput.Phys.Commun.* **183** (2012) 2458–2469, arXiv:1104.1573 [hep-ph].

[22] G. Belanger, F. Boudjema, A. Pukhov, and A. Semenov, "micrOMEGAs4.1: two dark matter candidates," *Comput.Phys.Commun.* **192** (2015) 322–329, arXiv:1407.6129 [hep-ph].

[23] G. Belanger, F. Boudjema, A. Pukhov, and A. Semenov, "MicrOMEGAs 2.0: A Program to calculate the relic density of dark matter in a generic model," *Comput.Phys.Commun.* **176** (2007) 367–382, arXiv:hep-ph/0607059 [hep-ph].

[24] G. Belanger, F. Boudjema, A. Pukhov, and A. Semenov, "Dark matter direct detection rate in a generic model with micrOMEGAs 2.2," *Comput.Phys.Commun.* **180** (2009) 747–767, arXiv:0803.2360 [hep-ph].

[25] G. Belanger, F. Boudjema, P. Brun, A. Pukhov, S. Rosier-Lees, *et al.*, "Indirect search for dark matter with micrOMEGAs2.4," *Comput.Phys.Commun.* **182** (2011) 842–856, arXiv:1004.1092 [hep-ph].

[26] G. Belanger, F. Boudjema, A. Pukhov, and A. Semenov, "micrOMEGAs_3: A program for calculating dark matter observables," *Comput.Phys.Commun.* **185** (2014) 960–985, arXiv:1305.0237 [hep-ph].

[27] J. Alwall, M. Herquet, F. Maltoni, O. Mattelaer, and T. Stelzer, "MadGraph 5 : Going Beyond," *JHEP* **1106** (2011) 128, arXiv:1106.0522 [hep-ph].

[28] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, *et al.*, "The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations," *JHEP* **1407** (2014) 079, arXiv:1405.0301 [hep-ph].

[29] E. Conte, B. Fuks and G. Serret, "MadAnalysis 5, A User-Friendly Framework for Collider Phenomenology," *Comput. Phys. Commun.* **184** (2013) 222, arXiv:1206.1599 [hep-ph].

[30] S. D. Aristizabal, F. Staub, and A. Vicente, "Shedding light on the $b \to s$ anomalies with a dark sector," arXiv:1503.06077 [hep-ph].

[31] A. Merle and M. Platscher, "Parity Problem of the Scotogenic Neutrino Model," arXiv:1502.03098 [hep-ph].

[32] **Christoph Langenbruch on behalf of the LHCb** Collaboration, "Latest results on rare decays from LHCb," arXiv:1505.04160 [hep-ex].

[33] W. Altmannshofer and D. M. Straub, "Implications of $b \to s$ measurements," arXiv:1503.06199 [hep-ph].