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, CalcHep/CompHep (which can also be used as input for MicrOmegas), the UFO format which is supported by MadGraph, as well as for WHIZARD and O'Mega. 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 [1].

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:** [2–6]. For related functionalities see [7,8], whereas for a pedagogical overview we recommend [1].

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* [9]. 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 [1].

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

```
1 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

```
3 Model`Name       = "Scotogenic";
4 Model`NameLaTeX  = "Scotogenic Model";
5 Model`Authors    = "N. Rojas, A. Vicente";
6 Model`Date       = "2015-04-28";
7
8 (* "28-04-2015 (first implementation)" *)
9 (* "25-05-2015 (removed mixings in scalar sector)" *)
```

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 a couple of 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 set

Scotogenic.m

```
11 SupersymmetricModel=False;
```

which just tells SARAH that the model we are going to define is non-supersymmetric.

Now 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

```
16 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

```
19 Gauge[[1]]={B,   U[1], hypercharge, g1, False, 1};
20 Gauge[[2]]={WB, SU[2], left,        g2, True , 1};
21 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*,

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

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.

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 [1]

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

thus leading to the vertex

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

[1] 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. 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`.

> **TIP:** 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 these three codes 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 some input values for the parameters of the model at the grand unification (GUT) scale, $m_{\rm GUT} \simeq 2 \cdot 10^{16}$ GeV, and then they are evolved using the RGEs down to the electroweak scale where the spectrum is computed. This is the usual way to proceed in supersymmetric models with boundary conditions at the GUT scale.
- **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 scale version we refer to the SARAH manual or to [1].

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

SPheno.m

```
OnlyLowEnergySPheno = True;
```

When the SPheno code is ready, it will require an input file in LesHouches format to run. The LesHouches format [10, 11] 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

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

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

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

TIP: SARAH makes use of the Mathematica command Solve to solve analytically the tadpole equations. If no solution is found, or if there are multiple solutions, 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

```
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]}
};
```

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 [12]. 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 [13]. This format can be generated with FeynRules [14], 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 [15, 16]. For the calculation of flavor observables we recommend having a look at the `FlavorKit` manual [8].

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 [15]. For a newer version see [16].

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

```
Block MINPAR       # Input parameters
1    0.250000E+00     # lambda1Input
2    0.500000E+00     # lambda2Input
3    0.500000E+00     # lambda3Input
4   -0.500000E+00     # lambda4Input
5    8.000000E-11     # lambda5Input
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.

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
      5.94828580E-04    2           22          22    # BR(hh -> VP VP )
      6.74940484E-02    2           21          21    # BR(hh -> VG VG )
      4.49830263E-03    2           23          23    # BR(hh -> VZ VZ )
      4.25758397E-02    2          -24          24    # BR(hh -> VWp^* ↪
          VWp_virt )
      3.07099640E-04    2           -3           3    # BR(hh -> Fd_2^* Fd_2 )
      8.28750986E-01    2           -5           5    # BR(hh -> Fd_3^* Fd_3 )
      1.54859432E-02    2          -15          15    # BR(hh -> Fe_3^* Fe_3 )
      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.