SARAH (part II): Working in Mathematica

Florian Staub | Dartmouth-TRIUMF HEP Tools Bootcamp, 23rd October 2017

KARLSRUHE INSTITUTE OF TECHNOLOGY, ITP & IKP

Outline

Karlsruher Institut für Technologie

- In this session we want to obtain **analytical information about a given model** using `SARAH` and `Mathematica`, i.e.
  - Tree-level masses and tadpoles
  - Tree-level vertices
  - RGEs

Outline

- In this session we want to obtain **analytical information about a given model** using `SARAH` and `Mathematica`, i.e.
  - Tree-level masses and tadpoles
  - Tree-level vertices
  - RGEs
- It is shown how the information is exported into LaTeX

Outline

- In this session we want to obtain **analytical information about a given model** using `SARAH` and `Mathematica`, i.e.
  - Tree-level masses and tadpoles
  - Tree-level vertices
  - RGEs
- It is shown how the information is exported into LaTeX

We use as showcase the SMSSM which we have 'implemented' in the previous session.

Loading and running SARAH

SARAH can easily be loaded in Mathematica

Mathematica
```
<<"SARAH-4.12.2/SARAH.m";
```

Loading and running SARAH

SARAH can easily be loaded in Mathematica

Mathematica
```
<<"SARAH-4.12.2/SARAH.m";
```

```
In[2]:= << "SARAH-4.12.2/SARAH.m"
SARAH 4.12.2
by Florian Staub, 2017
contributions by M. D. Goodsell, K. Nickel

References:
  Comput.Phys.Commun.181 (2010) 1077-1086. (arXiv:0909.2863[hep-ph])
  Comput.Phys.Commun.182 (2011) 808-833. (arXiv:1002.0840[hep-ph])
  Comput.Phys.Commun.184 (2013) 1792-1809. (arXiv:1207.0906[hep-ph])
  Comput.Phys.Commun.185 (2014) 1773-1790. (arXiv:1309.7223[hep-ph])
Download and Documentation:
  http://sarah.hepforge.org

Start evaluation of a model with:
 Start["Name of Model"]
e.g. Start["MSSM"] or Start["NMSSM","CKM"]
To get a list with all installed models, use ShowModels
```

Loading and running SARAH

A model is initialised afterwards via `Start["MODEL"]`

Mathematica
```
Start["SMSSM"];
```

Loading and running SARAH

A model is initialised afterwards via `Start["MODEL"]`

Mathematica
```
Start["SMSSM"];
```

```
In[3]:= Start["SMSSM"]
Preparing arrays
 ... checking Directory: /home/fnstaub/Desktop/HEP-Tools/SARAH-4.12.2/Models/
Model files loaded
  Model     : SMSSM
  Author(s) : F.Staub
  Date      : 2012-09-01
*******************************************************
Loading Susyno functions for the handling of Lie Groups
Based on Susyno v.2.0 by Renato Fonseca (1106.5016)
webpage: web.ist.utl.pt/renato.fonseca/susyno.html
*******************************************************
Initialization
Checking model files: All files okay

...

All Done. SMSSM is ready!
(Model initialized in 25.052s)


Are you unfamiliar with SARAH? Use SARAH`FirstSteps
```

Loading and running SARAH

During the initialisation, the following happens **automatically**:

- The model is checked for anomalies, charge conservation, etc.
- All gauge interactions are derived
- The Lagrangian for component fields is derived from superpotential
- The soft-breaking terms are added
- All field rotations are performed
- The gauge fixing terms are derived, ghost interactions are calculated
- The mass matrices and tadpole equations are derived

Loading and running SARAH

During the initialisation, the following happens **automatically**:

- The model is checked for anomalies, charge conservation, etc.
- All gauge interactions are derived
- The Lagrangian for component fields is derived from superpotential
- The soft-breaking terms are added
- All field rotations are performed
- The gauge fixing terms are derived, ghost interactions are calculated
- The mass matrices and tadpole equations are derived

It takes less than a minute before the model is ready and we can start playing

Particles

In order to see all particles of the current model for a given set of eigenstates, use

Mathematica
```
Particles[ STATES ]
```

STATES is the name of the eigenstates.

For each fields the given information is

```
{Field, First Generation, Last Generation, Indices }
```

Particles

In order to see all particles of the current model for a given set of eigenstates, use

Mathematica
```
Particles[ STATES ]
```

STATES is the name of the eigenstates.

For each fields the given information is

```
{Field, First Generation, Last Generation, Indices }
```

Example:

```
In[4]:= Particles[EWSB]

Out[4]= {{aB, 1, 1, A, {}}, {aWB, 1, 3, A, {{generation, 3}}}, {VG, 1, 1, V, {{color, 8}, {lorentz, 4}}},
  {gG, 1, 1, G, {{color, 8}}}, {fG, 1, 1, F, {{color, 8}}}, {aG, 1, 1, A, {{color, 8}}},
  {AdL, 1, 3, A, {{generation, 3}, {color, 3}}}, {AuL, 1, 3, A, {{generation, 3}, {color, 3}}},
  {AeL, 1, 3, A, {{generation, 3}}}, {FvL, 1, 3, F, {{generation, 3}}}, {AvL, 1, 3, A, {{generation, 3}}},
  {AHd0, 1, 1, A, {}}, {AHdm, 1, 1, A, {}}, {AHu0, 1, 1, A, {}}, {AHup, 1, 1, A, {}},
  {AdR, 1, 3, A, {{generation, 3}, {color, 3}}}, {AuR, 1, 3, A, {{generation, 3}, {color, 3}}},
  {AeR, 1, 3, A, {{generation, 3}}}, {AsR, 1, 1, A, {}}, {vd, 1, 1, VEV, {}}, {vu, 1, 1, VEV, {}},
  {vS, 1, 1, VEV, {}}, {VP, 1, 1, V, {{lorentz, 4}}}, {VZ, 1, 1, V, {{lorentz, 4}}},
  {gP, 1, 1, G, {}}, {gZ, 1, 1, G, {}}, {VWm, 1, 1, V, {{lorentz, 4}}}, {gWm, 1, 1, G, {}},
  {gWmC, 1, 1, G, {}}, {Sd, 1, 6, S, {{generation, 6}, {color, 3}}}, {Sv, 1, 3, S, {{generation, 3}}},
...
```

Parameters

In order to see all parameter of the current model, use

Mathematica
```
parameters
```

For each parameter the given information is

```
{Parameter, Indices, Index Ranges }
```

Parameters

In order to see all parameter of the current model, use

Mathematica

```
parameters
```

For each parameter the given information is

```
{Parameter, Indices, Index Ranges }
```

Example:

```
In[5]:= parameters

Out[5]= {{g1, {}, {}}, {g2, {}, {}}, {g3, {}, {}}, {L1, {}, {}}, {L[L1], {}, {}}, {μ, {}, {}}, {B[μ], {}, {}}, {MS, {}, {}},
  {B[MS], {}, {}}, {Yd, {generation, generation}, {3, 3}}, {T[Yd], {generation, generation}, {3, 3}},
  {Ye, {generation, generation}, {3, 3}}, {T[Ye], {generation, generation}, {3, 3}}, {λ, {}, {}}, {T[λ], {}, {}},
  {κ, {}, {}}, {T[κ], {}, {}}, {Yu, {generation, generation}, {3, 3}}, {T[Yu], {generation, generation}, {3, 3}},
  {mq2, {generation, generation}, {3, 3}}, {ml2, {generation, generation}, {3, 3}},
  {mHd2, {}, {}}, {mHu2, {}, {}}, {md2, {generation, generation}, {3, 3}},
  {mu2, {generation, generation}, {3, 3}}, {me2, {generation, generation}, {3, 3}},
  {ms2, {}, {}}, {MassB, {}, {}}, {MassWB, {}, {}}, {MassG, {}, {}}, {vd, {}, {1}}, {vu, {}, {1}},
  {vS, {}, {1}}, {ZZ, {generation, generation}, {2, 2}}, {ZW, {generation, generation}, {2, 2}},
  {ZfW, {generation, generation}, {3, 3}}, {PhaseGlu, {}, {}}, {ZD, {generation, generation}, {6, 6}},
  {ZV, {generation, generation}, {3, 3}}, {ZU, {generation, generation}, {6, 6}},
  {ZE, {generation, generation}, {6, 6}}, {ZH, {generation, generation}, {3, 3}},
  {ZA, {generation, generation}, {3, 3}}, {ZP, {generation, generation}, {2, 2}},
...
```

Mass Matrices and Tadpoles

Mass Matrix

The tree-level mass matrix is given by

Mathematica
```
MassMatrix[ FIELD ];
```

with FIELD is the name of the mass eigenstates.

Mass Matrix

The tree-level mass matrix is given by

Mathematica

```
MassMatrix[ FIELD ];
```

with FIELD is the name of the mass eigenstates.

Examples:

The mass matrix of the CP-even Higgs states is given by:

In[34]:= `MassMatrix[hh] // FullSimplify`

Out[34]=

$$\Big\{\Big\{\frac{1}{8}\Big(8\,\text{mHd2} + (\text{g1}^2+\text{g2}^2)(3\,\text{vd}^2-\text{vu}^2) + 4\big((\text{vS}^2+\text{vu}^2)\lambda + \sqrt{2}\,\text{vS}\,\mu\big)\,\text{conj}[\lambda] + 4\big(\sqrt{2}\,\text{vS}\,\lambda + 2\mu\big)\,\text{conj}[\mu]\Big),$$

$$\frac{1}{4}\Big(-(\text{g1}^2+\text{g2}^2)\,\text{vd}\,\text{vu} - 2\,\text{B}[\mu] - 2\lambda\,\text{conj}[\text{L1}] - 2\,\text{L1}\,\text{conj}[\lambda] + 4\,\text{vd}\,\text{vu}\,\lambda\,\text{conj}[\lambda] - \text{vS}^2(\lambda\,\text{conj}[\kappa] + \kappa\,\text{conj}[\lambda]) -$$

$$2\,\text{conj}[\text{B}[\mu]] - \sqrt{2}\,\text{vS}\,(\lambda\,\text{conj}[\text{MS}] + \text{MS}\,\text{conj}[\lambda] + \text{conj}[\text{T}[\lambda]] + \text{T}[\lambda])\Big),\ \text{vd}\Big(\text{vS}\,\lambda + \frac{\mu}{\sqrt{2}}\Big)\text{conj}[\lambda] + \frac{\text{vd}\,\lambda\,\text{conj}[\mu]}{\sqrt{2}} -$$

$$\frac{1}{4}\,\text{vu}\Big(\sqrt{2}\,\lambda\,\text{conj}[\text{MS}] + 2\,\text{vS}\,(\lambda\,\text{conj}[\kappa] + \kappa\,\text{conj}[\lambda]) + \sqrt{2}\,(\text{MS}\,\text{conj}[\lambda] + \text{conj}[\text{T}[\lambda]] + \text{T}[\lambda])\Big)\Big\},$$

...

Mass Matrix

The tree-level mass matrix is given by

Mathematica

```
MassMatrix[ FIELD ];
```

with FIELD is the name of the mass eigenstates.

Examples:

The down-quark mass matrix is given by:

```
In[7]:= MassMatrix[Fd] // MatrixForm
```

Out[7]//MatrixForm=

$$\begin{pmatrix} \frac{\text{vd Yd}[1,1]}{\sqrt{2}} & \frac{\text{vd Yd}[2,1]}{\sqrt{2}} & \frac{\text{vd Yd}[3,1]}{\sqrt{2}} \\ \frac{\text{vd Yd}[1,2]}{\sqrt{2}} & \frac{\text{vd Yd}[2,2]}{\sqrt{2}} & \frac{\text{vd Yd}[3,2]}{\sqrt{2}} \\ \frac{\text{vd Yd}[1,3]}{\sqrt{2}} & \frac{\text{vd Yd}[2,3]}{\sqrt{2}} & \frac{\text{vd Yd}[3,3]}{\sqrt{2}} \end{pmatrix}$$

For fermions names of Weyl or Dirac spinors can be used, i.e. `MassMatrix[FDL]` or `MassMatrix[FDR]` give the same output.

Mass Matrix

The tree-level mass matrix is given by

Mathematica

```
MassMatrix[ FIELD ];
```

with FIELD is the name of the mass eigenstates.

Examples:

The neutralino and chargino mass matrices are given by:

```
In[9]:= MatrixForm /@ MassMatrix /@ {Cha, Chi}
```

Out[9]=

$$\left\{ \begin{pmatrix} \text{MassWB} & \frac{g2\, vu}{\sqrt{2}} \\ \frac{g2\, vd}{\sqrt{2}} & \frac{vS\, \lambda}{\sqrt{2}} + \mu \end{pmatrix}, \begin{pmatrix} \text{MassB} & 0 & -\frac{g1\, vd}{2} & \frac{g1\, vu}{2} & 0 \\ 0 & \text{MassWB} & \frac{g2\, vd}{2} & -\frac{g2\, vu}{2} & 0 \\ -\frac{g1\, vd}{2} & \frac{g2\, vd}{2} & 0 & -\frac{vS\, \lambda}{\sqrt{2}} - \mu & -\frac{vu\, \lambda}{\sqrt{2}} \\ \frac{g1\, vu}{2} & -\frac{g2\, vu}{2} & -\frac{vS\, \lambda}{\sqrt{2}} - \mu & 0 & -\frac{vd\, \lambda}{\sqrt{2}} \\ 0 & 0 & -\frac{vu\, \lambda}{\sqrt{2}} & -\frac{vd\, \lambda}{\sqrt{2}} & MS + \sqrt{2}\, vS\, \kappa \end{pmatrix} \right\}$$

Mass Matrix

The tree-level mass matrix is given by

Mathematica

```
MassMatrix[ FIELD ];
```

with FIELD is the name of the mass eigenstates.

Examples:

Mass matrices involving Goldstone bosons come with `RXi` what denotes the gauge dependent part from the gauge fixing Lagrangian.

- `RXi[_] -> 0` corresponds to Landau gauge
- `RXi[_] -> 1` corresponds to Feynman gauge

Mass Matrix

The tree-level mass matrix is given by

Mathematica
```
MassMatrix[ FIELD ];
```

with FIELD is the name of the mass eigenstates.

Examples:

Mass matrices involving Goldstone bosons come with `RXi` what denotes the gauge dependent part from the gauge fixing Lagrangian.

- `RXi[_] -> 0` corresponds to Landau gauge
- `RXi[_] -> 1` corresponds to Feynman gauge

For instance, the CP-odd Higgs mass matrix reads

In[13]:= **MassMatrix[Ah]**

Out[13]=

$$\Big\{\Big\{mHd2 + \frac{g1^2\, vd^2}{8} + \frac{g2^2\, vd^2}{8} - \frac{g1^2\, vu^2}{8} - \frac{g2^2\, vu^2}{8} + \frac{1}{2} vS^2\, \lambda\, conj[\lambda] + \frac{1}{2} vu^2\, \lambda\, conj[\lambda] + \frac{vS\, \mu\, conj[\lambda]}{\sqrt{2}} + \frac{vS\, \lambda\, conj[\mu]}{\sqrt{2}} + \mu\, conj[\mu] + \frac{1}{4} g2^2\, vd^2\, Cos[ThetaW]^2\, RXi[Z] +$$

...

Mass Matrix

The tree-level mass matrix is given by

Mathematica
```
MassMatrix[ FIELD ];
```

with FIELD is the name of the mass eigenstates.

Examples:

For vector bosons the name of one of the mass eigenstate can be used, i.e. `MassMatrix[VP]` and `MassMatrix[VZ]` give the same result:

```
In[10]:= MassMatrix[VP] // MatrixForm
Out[10]//MatrixForm=
```

$$\begin{pmatrix} \frac{g1^2\ vd^2}{4} + \frac{g1^2\ vu^2}{4} & -\frac{1}{4}\ g1\ g2\ vd^2 - \frac{1}{4}\ g1\ g2\ vu^2 \\ -\frac{1}{4}\ g1\ g2\ vd^2 - \frac{1}{4}\ g1\ g2\ vu^2 & \frac{g2^2\ vd^2}{4} + \frac{g2^2\ vu^2}{4} \end{pmatrix}$$

Mass Matrix

The tree-level mass matrix is given by

Mathematica

```
MassMatrix[ FIELD ];
```

with FIELD is the name of the mass eigenstates.

Examples:

For vector bosons the name of one of the mass eigenstate can be used, i.e. `MassMatrix[VP]` and `MassMatrix[VZ]` give the same result:

```
In[10]:= MassMatrix[VP] // MatrixForm
```

Out[10]//MatrixForm=

$$\begin{pmatrix} \frac{g1^2\,vd^2}{4}+\frac{g1^2\,vu^2}{4} & -\frac{1}{4}\,g1\,g2\,vd^2-\frac{1}{4}\,g1\,g2\,vu^2 \\ -\frac{1}{4}\,g1\,g2\,vd^2-\frac{1}{4}\,g1\,g2\,vu^2 & \frac{g2^2\,vd^2}{4}+\frac{g2^2\,vu^2}{4} \end{pmatrix}$$

one can check immediately that one state is massless

```
In[35]:= MassMatrix[VP];
         Eigenvalues[%]
```

Out[36]= $\left\{0,\ \left(g1^2+g2^2\right)\left(\frac{vd^2}{4}+\frac{vu^2}{4}\right)\right\}$

Mass Matrix

The tree-level mass matrix is given by

Mathematica
```
MassMatrix[ FIELD ];
```

with FIELD is the name of the mass eigenstates.

Examples:

The sfermion mass matrices are general 6×6 matrices which are a bit lengthy, i.e. the down squark matrix reads

```
In[12]:= MassMatrix[Sd]
```

Out[12]= $\{\{-\frac{1}{24} g1^2\, vd^2 - \frac{g2^2\, vd^2}{8} + \frac{g1^2\, vu^2}{24} + \frac{g2^2\, vu^2}{8} + mq2[1,1] + \frac{1}{2} vd^2\, sum[j1, 1, 3, conj[Yd[j1,1]]\, Yd[j1,1]],$

$mq2[1,2] + \frac{1}{2} vd^2\, sum[j1, 1, 3, conj[Yd[j1,1]]\, Yd[j1,2]],$

$mq2[1,3] + \frac{1}{2} vd^2\, sum[j1, 1, 3, conj[Yd[j1,1]]\, Yd[j1,3]],$

$-\frac{1}{2} vS\, vu\, \lambda\, conj[Yd[1,1]] - \frac{vu\, \mu\, conj[Yd[1,1]]}{\sqrt{2}} + \frac{vd\, conj[T[Yd][1,1]]}{\sqrt{2}},$

...

Tadpole Equations I

The tadpole equations corresponding to a scalar or VEV is returned by

Mathematica

```
TadpoleEquation[ X ];
```

with X is the name of VEV (or the corresponding field).

Tadpole Equations I

The tadpole equations corresponding to a scalar or VEV is returned by

Mathematica
```
TadpoleEquation[ X ];
```

with X is the name of VEV (or the corresponding field).

Examples:

The three tadpole equations in the SMSSM read:

In[38]:= `TadpoleEquation[vd]`

Out[38]=
$$\text{mHd2 vd} + \frac{\text{g1}^2\,\text{vd}^3}{8} + \frac{\text{g2}^2\,\text{vd}^3}{8} - \frac{1}{8}\,\text{g1}^2\,\text{vd}\,\text{vu}^2 - \frac{1}{8}\,\text{g2}^2\,\text{vd}\,\text{vu}^2 - \frac{1}{2}\,\text{vu}\,B[\mu] - \frac{1}{2}\,\text{vu}\,\lambda\,\text{conj}[\text{L1}] - \frac{\text{vS}\,\text{vu}\,\lambda\,\text{conj}[\text{MS}]}{2\sqrt{2}} - \frac{1}{4}\,\text{vS}^2\,\text{vu}\,\lambda\,\text{conj}[\kappa] - \frac{1}{2}\,\text{L1}\,\text{vu}\,\text{conj}[\lambda] - \frac{\text{MS}\,\text{vS}\,\text{vu}\,\text{conj}[\lambda]}{2\sqrt{2}} - \frac{1}{4}\,\text{vS}^2\,\text{vu}\,\kappa\,\text{conj}[\lambda] + \frac{1}{2}\,\text{vd}\,\text{vS}^2\,\lambda\,\text{conj}[\lambda] + \frac{1}{2}\,\text{vd}\,\text{vu}^2\,\lambda\,\text{conj}[\lambda] + \frac{\text{vd}\,\text{vS}\,\mu\,\text{conj}[\lambda]}{\sqrt{2}} + \frac{\text{vd}\,\text{vS}\,\lambda\,\text{conj}[\mu]}{\sqrt{2}} + \text{vd}\,\mu\,\text{conj}[\mu] - \frac{1}{2}\,\text{vu}\,\text{conj}[B[\mu]] - \frac{\text{vS}\,\text{vu}\,\text{conj}[T[\lambda]]}{2\sqrt{2}} - \frac{\text{vS}\,\text{vu}\,T[\lambda]}{2\sqrt{2}} == 0$$

Tadpole Equations I

The tadpole equations corresponding to a scalar or VEV is returned by

Mathematica
```
TadpoleEquation[ X ];
```

with X is the name of VEV (or the corresponding field).

Examples:

The three tadpole equations in the SMSSM read:

In[15]:= **TadpoleEquation[vu]**

Out[15]=
$$\text{mHu2 vu} - \frac{1}{8}\text{g1}^2\text{vd}^2\text{vu} - \frac{1}{8}\text{g2}^2\text{vd}^2\text{vu} + \frac{\text{g1}^2\text{vu}^3}{8} + \frac{\text{g2}^2\text{vu}^3}{8} - \frac{1}{2}\text{vd B}[\mu] - \frac{1}{2}\text{vd}\,\lambda\,\text{conj}[\text{L1}] - \frac{\text{vd vS}\,\lambda\,\text{conj}[\text{MS}]}{2\sqrt{2}} - \frac{1}{4}\text{vd vS}^2\lambda\,\text{conj}[\kappa] - \frac{1}{2}\text{L1 vd conj}[\lambda] - \frac{\text{MS vd vS conj}[\lambda]}{2\sqrt{2}} - \frac{1}{4}\text{vd vS}^2\kappa\,\text{conj}[\lambda] + \frac{1}{2}\text{vd}^2\text{vu}\,\lambda\,\text{conj}[\lambda] + \frac{1}{2}\text{vS}^2\text{vu}\,\lambda\,\text{conj}[\lambda] + \frac{\text{vS vu}\,\mu\,\text{conj}[\lambda]}{\sqrt{2}} + \frac{\text{vS vu}\,\lambda\,\text{conj}[\mu]}{\sqrt{2}} + \text{vu}\,\mu\,\text{conj}[\mu] - \frac{1}{2}\text{vd conj}[\text{B}[\mu]] - \frac{\text{vd vS conj}[\text{T}[\lambda]]}{2\sqrt{2}} - \frac{\text{vd vS T}[\lambda]}{2\sqrt{2}} == 0$$

Tadpole Equations I

Karlsruher Institut für Technologie

The tadpole equations corresponding to a scalar or VEV is returned by

```
Mathematica
TadpoleEquation[ X ];
```

with X is the name of VEV (or the corresponding field).

Examples:

The three tadpole equations in the SMSSM read:

```
In[16]:= TadpoleEquation[vS]
```

Out[16]=

$$
\begin{aligned}
&\text{ms2 vS} + \frac{1}{2}\,\text{vS B[MS]} + \frac{\text{MS conj[L1]}}{\sqrt{2}} + \text{vS}\,\kappa\,\text{conj[L1]} + \frac{\text{L1 conj[MS]}}{\sqrt{2}} + \text{MS vS conj[MS]} + \\
&\frac{3\,\text{vS}^2\,\kappa\,\text{conj[MS]}}{2\sqrt{2}} - \frac{\text{vd vu}\,\lambda\,\text{conj[MS]}}{2\sqrt{2}} + \text{L1 vS conj}[\kappa] + \frac{3\,\text{MS vS}^2\,\text{conj}[\kappa]}{2\sqrt{2}} + \text{vS}^3\,\kappa\,\text{conj}[\kappa] - \\
&\frac{1}{2}\,\text{vd vS vu}\,\lambda\,\text{conj}[\kappa] - \frac{\text{MS vd vu conj}[\lambda]}{2\sqrt{2}} - \frac{1}{2}\,\text{vd vS vu}\,\kappa\,\text{conj}[\lambda] + \frac{1}{2}\,\text{vd}^2\,\text{vS}\,\lambda\,\text{conj}[\lambda] + \\
&\frac{1}{2}\,\text{vS vu}^2\,\lambda\,\text{conj}[\lambda] + \frac{\text{vd}^2\,\mu\,\text{conj}[\lambda]}{2\sqrt{2}} + \frac{\text{vu}^2\,\mu\,\text{conj}[\lambda]}{2\sqrt{2}} + \frac{\text{vd}^2\,\lambda\,\text{conj}[\mu]}{2\sqrt{2}} + \frac{\text{vu}^2\,\lambda\,\text{conj}[\mu]}{2\sqrt{2}} + \frac{1}{2}\,\text{vS conj[B[MS]]} + \\
&\frac{\text{conj[L[L1]]}}{\sqrt{2}} + \frac{\text{vS}^2\,\text{conj}[\text{T}[\kappa]]}{2\sqrt{2}} - \frac{\text{vd vu conj}[\text{T}[\lambda]]}{2\sqrt{2}} + \frac{\text{L[L1]}}{\sqrt{2}} + \frac{\text{vS}^2\,\text{T}[\kappa]}{2\sqrt{2}} - \frac{\text{vd vu T}[\lambda]}{2\sqrt{2}} == 0
\end{aligned}
$$

Tadpole Equations I

The tadpole equations corresponding to a scalar or VEV is returned by

Mathematica
```
TadpoleEquation[ X ];
```

with X is the name of VEV (or the corresponding field).

Examples:

The three tadpole equations in the SMSSM read:

The same results are obtained from

- `TadpoleEquation[phid]`
- `TadpoleEquation[phiu]`
- `TadpoleEquation[phiS]`

Tadpole Equations II

A list with all tadpole equations for a given set of eigenstates is stored in

Mathematica
```
TadpoleEquations[ STATES ];
```

with STATES is the name of eigenstates.

Tadpole Equations II

A list with all tadpole equations for a given set of eigenstates is stored in

Mathematica
```
TadpoleEquations[ STATES ];
```

with STATES is the name of eigenstates.

Example: The three tadpole equations after EWSB are stored in:

In[17]:= `TadpoleEquations[EWSB]`

Out[17]=

$$\Big\{ \text{mHd2 vd} + \frac{\text{g1}^2\,\text{vd}^3}{8} + \frac{\text{g2}^2\,\text{vd}^3}{8} - \frac{1}{8}\,\text{g1}^2\,\text{vd vu}^2 - \frac{1}{8}\,\text{g2}^2\,\text{vd vu}^2 - \frac{1}{2}\,\text{vu B}[\mu] - \ldots.$$

$$\frac{\text{vd vS}\,\lambda\,\text{conj}[\mu]}{\sqrt{2}} + \text{vd}\,\mu\,\text{conj}[\mu] - \frac{1}{2}\,\text{vu conj}[\text{B}[\mu]] - \frac{\text{vS vu conj}[\text{T}[\lambda]]}{2\sqrt{2}} - \frac{\text{vS vu T}[\lambda]}{2\sqrt{2}},$$

$$\text{mHu2 vu} - \frac{1}{8}\,\text{g1}^2\,\text{vd}^2\,\text{vu} - \frac{1}{8}\,\text{g2}^2\,\text{vd}^2\,\text{vu} + \frac{\text{g1}^2\,\text{vu}^3}{8} + \frac{\text{g2}^2\,\text{vu}^3}{8} - \frac{1}{2}\,\text{vd B}[\mu] - \frac{1}{2}\,\text{vd}\,\lambda\,\text{conj}[\text{L1}] - \ldots$$

$$\frac{\text{vS vu}\,\lambda\,\text{conj}[\mu]}{\sqrt{2}} + \text{vu}\,\mu\,\text{conj}[\mu] - \frac{1}{2}\,\text{vd conj}[\text{B}[\mu]] - \frac{\text{vd vS conj}[\text{T}[\lambda]]}{2\sqrt{2}} - \frac{\text{vd vS T}[\lambda]}{2\sqrt{2}},$$

$$\text{ms2 vS} + \frac{1}{2}\,\text{vS B}[\text{MS}] + \frac{\text{MS conj}[\text{L1}]}{\sqrt{2}} + \text{vS}\,\kappa\,\text{conj}[\text{L1}] + \frac{\text{L1 conj}[\text{MS}]}{\sqrt{2}} + \text{MS vS conj}[\text{MS}] + \frac{3\,\text{vS}^2\,\kappa\,\text{conj}[\text{MS}]}{2\sqrt{2}} - \ldots.$$

$$\frac{\text{conj}[\text{L}[\text{L1}]]}{\sqrt{2}} + \frac{\text{vS}^2\,\text{conj}[\text{T}[\kappa]]}{2\sqrt{2}} - \frac{\text{vd vu conj}[\text{T}[\lambda]]}{2\sqrt{2}} + \frac{\text{L}[\text{L1}]}{\sqrt{2}} + \frac{\text{vS}^2\,\text{T}[\kappa]}{2\sqrt{2}} - \frac{\text{vd vu T}[\lambda]}{2\sqrt{2}} \Big\}$$

Application: Pseudo-Scalar Mass

One can combine the information of the mass matrix with the tadpole equations to check the masses of the Goldstones:

Application: Pseudo-Scalar Mass

One can combine the information of the mass matrix with the tadpole equations to check the masses of the Goldstones:

Mathematica
```
solTadpoles =  Solve[TadpoleEquations[EWSB] == 0,
Simplify[ Eigenvalues[MassMatrix[Ah] /. solTadpoles]]
```

1. We use the `Solve` command of `Mathematica` to get the solutions of the tadpole equations for $m_{H_d}^2$, $m_{H_u}^2$, m_S^2
2. We insert the solutions in the pseudo-scalar mass matrix
3. We use `Eigenvalues` to get CP odd masses

Application: Pseudo-Scalar Mass

One can combine the information of the mass matrix with the tadpole equations to check the masses of the Goldstones:

```
In[39]:= solTadpoles = Solve[TadpoleEquations[EWSB] == 0, {mHd2, mHu2, ms2}][[1]];
         Simplify[Eigenvalues[MassMatrix[Ah] /. solTadpoles]]

Out[40]= {1/4 (vd^2 + vu^2) RXi[Z] (g2 Cos[ThetaW] + g1 Sin[ThetaW])^2, 1/(8 vd vS vu)
  (-4 vd vS vu B[MS] + 2 vd^2 vS B[μ] + 2 vS vu^2 B[μ] - 2 Sqrt[2] MS vd vu conj[L1] - 8 vd vS vu κ conj[L1] +
   2 vd^2 vS λ conj[L1] + 2 vS vu^2 λ conj[L1] - 2 Sqrt[2] L1 vd vu conj[MS] - Sqrt[2] vd vS^2 vu κ conj[MS] +
   Sqrt[2] vd^2 vS^2 λ conj[MS] + Sqrt[2] vd^2 vu^2 λ conj[MS] + Sqrt[2] vS^2 vu^2 λ conj[MS] - 8 L1 vd vS vu conj[κ] -
   Sqrt[2] MS vd vS^2 vu conj[κ] + vd^2 vS^3 λ conj[κ] + 4 vd^2 vS vu^2 λ conj[κ] + vS^3 vu^2 λ conj[κ] +
   2 L1 vd^2 vS conj[λ] + Sqrt[2] MS vd^2 vS^2 conj[λ] + Sqrt[2] MS vd^2 vu^2 conj[λ] + 2 L1 vS vu^2 conj[λ] +
   Sqrt[2] MS vS^2 vu^2 conj[λ] + vd^2 vS^3 κ conj[λ] + 4 vd^2 vS vu^2 κ conj[λ] + vS^3 vu^2 κ conj[λ] -
   Sqrt[2] vd^3 vu μ conj[λ] - Sqrt[2] vd vu^3 μ conj[λ] - Sqrt[2] vd^3 vu λ conj[μ] - Sqrt[2] vd vu^3 λ conj[μ] -
   4 vd vS vu conj[B[MS]] + 2 vd^2 vS conj[B[μ]] + 2 vS vu^2 conj[B[μ]] - 2 Sqrt[2] vd vu conj[L[L1]] -
   3 Sqrt[2] vd vS^2 vu conj[T[κ]] + Sqrt[2] vd^2 vS^2 conj[T[λ]] + Sqrt[2] vd^2 vu^2 conj[T[λ]] + Sqrt[2] vS^2 vu^2 conj[T[λ]] -
   2 Sqrt[2] vd vu L[L1] - 3 Sqrt[2] vd vS^2 vu T[κ] + Sqrt[2] vd^2 vS^2 T[λ] + Sqrt[2] vd^2 vu^2 T[λ] + Sqrt[2] vS^2 vu^2 T[λ] -
   Sqrt[((-4 vd vS vu B[MS] + 2 vS (vd^2 + vu^2) B[μ] - 2 Sqrt[2] MS vd vu conj[L1] - 8 vd vS vu κ conj[L1] + 2 vd^2 vS λ conj[
        L1] + 2 vS vu^2 λ conj[L1] - 2 Sqrt[2] L1 vd vu conj[MS] - Sqrt[2] vd vS^2 vu κ conj[MS] + Sqrt[2] vd^2 vS^2 λ conj[MS] +
      Sqrt[2] vd^2 vu^2 λ conj[MS] + Sqrt[2] vS^2 vu^2 λ conj[MS] - 8 L1 vd vS vu conj[κ] - Sqrt[2] MS vd vS^2 vu conj[κ] +
      vd^2 vS^3 λ conj[κ] + 4 vd^2 vS vu^2 λ conj[κ] + vS^3 vu^2 λ conj[κ] + 2 L1 vd^2 vS conj[λ] +
      Sqrt[2] MS vd^2 vS^2 conj[λ] + Sqrt[2] MS vd^2 vu^2 conj[λ] + 2 L1 vS vu^2 conj[λ] + Sqrt[2] MS vS^2 vu^2 conj[λ] +
```

Application: Pseudo-Scalar Mass

One can combine the information of the mass matrix with the tadpole equations to check the masses of the Goldstones:

```
In[39]:= solTadpoles = Solve[TadpoleEquations[EWSB] == 0, {mHd2, mHu2, ms2}][[1]];
         Simplify[Eigenvalues[MassMatrix[Ah] /. solTadpoles]]

Out[40]= {1/4 (vd^2 + vu^2) RXi[Z] (g2 Cos[ThetaW] + g1 Sin[ThetaW])^2, 1/(8 vd vS vu)
  (-4 vd vS vu B[MS] - 2 vd^2 vS B[μ] - 2 vS vu^2 B[μ] - 2 √2 MS vd vu conj[L1] - 8 vd vS vu κ conj[L1] +
  2 vd^2 vS λ conj[L1] + 2 vS vu^2 λ conj[L1] - 2 √2 L1 vd vu conj[MS] - √2 vd vS^2 vu κ conj[MS] +
  √2 vd^2 vS^2 λ conj[MS] + √2 vd^2 vu^2 λ conj[MS] + √2 vS^2 vu^2 λ conj[MS] - 8 L1 vd vS vu conj[κ] -
  √2 MS vd vS^2 vu conj[κ] + vd^2 vS^3 λ conj[κ] + 4 vd^2 vS vu^2 λ conj[κ] + vS^3 vu^2 λ conj[κ] +
  2 L1 vd^2 vS conj[λ] + √2 MS vd^2 vS^2 conj[λ] + √2 MS vd^2 vu^2 conj[λ] + 2 L1 vS vu^2 conj[λ] +
  √2 MS vS^2 vu^2 conj[λ] + vd^2 vS^3 κ conj[λ] + 4 vd^2 vS vu^2 κ conj[λ] + vS^3 vu^2 κ conj[λ] -
  √2 vd^3 vu μ conj[λ] - √2 vd vu^3 μ conj[λ] - √2 vd^3 vu λ conj[μ] - √2 vd vu^3 λ conj[μ] -
  4 vd vS vu conj[B[MS]] + 2 vd^2 vS conj[B[μ]] + 2 vS vu^2 conj[B[μ]] - 2 √2 vd vu conj[L[L1]] -
  3 √2 vd vS^2 vu conj[T[κ]] + √2 vd^2 vS^2 conj[T[λ]] + √2 vd^2 vu^2 conj[T[λ]] + √2 vS^2 vu^2 conj[T[λ]] -
  2 √2 vd vu L[L1] - 3 √2 vd vS^2 vu T[κ] + √2 vd^2 vS^2 T[λ] + √2 vd^2 vu^2 T[λ] + √2 vS^2 vu^2 T[λ] -
  √((-4 vd ... onj[
  ... MS] +
    √2 vd^2 vu^2 λ conj[MS] + √2 vS^2 vu^2 λ conj[MS] - 8 L1 vd vS vu conj[κ] - √2 MS vd vS^2 vu conj[κ] +
    vd^2 vS^3 λ conj[κ] + 4 vd^2 vS vu^2 λ conj[κ] + vS^3 vu^2 λ conj[κ] + 2 L1 vd^2 vS conj[λ] +
    √2 MS vd^2 vS^2 conj[λ] + √2 MS vd^2 vu^2 conj[λ] + 2 L1 vS vu^2 conj[λ] + √2 MS vS^2 vu^2 conj[λ] +
    ...
```

→ The Goldstone mass squared is $R_\xi M_Z^2$

Masses and Tadpoles

Vertices

Vertex command

A vertex for a given set of particles is calculated via

```
Mathematica
Vertex[{Field 1, Field 2, ...}, Options]
```

Vertex command

A vertex for a given set of particles is calculated via

```
Mathematica
Vertex[{Field 1, Field 2, ...}, Options]
```

Possible options are

- `Eigenstates -> STATES`: for which set of states the vertices shall be calculated
- `UseDependences -> True/False`: shall dependencies defined in `parameters.m` be applied

Vertex command

A vertex for a given set of particles is calculated via

Mathematica

```
Vertex[{Field 1, Field 2, ...}, Options]
```

Examples:

The down-quark Higgs vertex is given by

```
In[20]:= Vertex[{bar[Fd], Fd, hh}]

Out[20]= {{bar[Fd[{gt1, ct1}]], Fd[{gt2, ct2}], hh[{gt3}]}, {-1/Sqrt[2] i conj[ZH[gt3, 1]] Delta[ct1, ct2]
    sum[j2, 1, 3, conj[ZDL[gt2, j2]] sum[j1, 1, 3, conj[ZDR[gt1, j1]] Yd[j1, j2]]], PL},
  {-(i conj[ZH[gt3, 1]] Delta[ct1, ct2] sum[j2, 1, 3, sum[j1, 1, 3, conj[Yd[j1, j2]] ZDR[gt2, j1]] ZDL[gt1, j2]])/Sqrt[2],
  PR}}
```

- The indices to external states are added
- For each vertex the Lorentz dependent and independent parts are separated (PL, PR are projection operators)
- Delta[i,j] is the Kronecker Delta

Vertex command

A vertex for a given set of particles is calculated via

Mathematica
```
Vertex[{Field 1, Field 2, ...}, Options]
```

Examples:

The down-quark Higgs vertex **for third generation quarks and without flavour violation** is given by

```
In[21]:= Vertex[{bar[Fd], Fd, hh}] /. {gt1 → 3, gt2 → 3} /. ZDL → Delta /. ZDR → Delta
```

$$\text{Out[21]= } \{\{\text{bar}[\text{Fd}[\{3, \text{ct1}\}]], \text{Fd}[\{3, \text{ct2}\}], \text{hh}[\{\text{gt3}\}]\}, \{-\frac{i\, \text{conj}[\text{ZH}[\text{gt3}, 1]]\, \text{Delta}[\text{ct1}, \text{ct2}]\, \text{Yd}[3, 3]}{\sqrt{2}}, \text{PL}\}, \{-\frac{i\, \text{conj}[\text{Yd}[3, 3]]\, \text{conj}[\text{ZH}[\text{gt3}, 1]]\, \text{Delta}[\text{ct1}, \text{ct2}]}{\sqrt{2}}, \text{PR}\}\}$$

Vertex command

A vertex for a given set of particles is calculated via

Mathematica
```
Vertex[{Field 1, Field 2, ...}, Options]
```

Examples:

The squark-quark gluon vertex is calculated via

```
In[28]:= Vertex[{bar[Fd], Glu, Sd}]
Out[28]= {{bar[Fd[{gt1, ct1}]], Glu[{ct2}], Sd[{gt3, ct3}]},
  {(i g3 PhaseGlu Lam[ct2, ct1, ct3] sum[j1, 1, 3, conj[ZD[gt3, 3 + j1]] conj[ZDR[gt1, j1]]])/Sqrt[2], PL},
  {-(i g3 conj[PhaseGlu] Lam[ct2, ct1, ct3] sum[j1, 1, 3, conj[ZD[gt3, j1]] ZDL[gt1, j1]])/Sqrt[2], PR}}
```

- `Lam` are the Gell-Mann matrices (`fSU3` would be the structure constants)

Vertex command

A vertex for a given set of particles is calculated via

```
Mathematica
Vertex[{Field 1, Field 2, ...}, Options]
```

Examples:

The quark-gluon vertex is calculated via

```
In[22]:= Vertex[{bar[Fd], Fd, VG}]
Out[22]= {{bar[Fd[{gt1, ct1}]], Fd[{gt2, ct2}], VG[{ct3, lt3}]},
          {-1/2 i g3 Delta[gt1, gt2] Lam[ct3, ct1, ct2], LorentzProduct[gamma[lt3], PL]},
          {-1/2 i g3 Delta[gt1, gt2] Lam[ct3, ct1, ct2], LorentzProduct[gamma[lt3], PR]}}
```

- `gamma[x]` is γ_x
- `LorentzProduct` defines a non-commutative product

Dependencies I

- SARAH usually expresses the vertices in fundamental quantities.
- Relations can be defined in `parameters.m` and used via UseDependences->True

Dependencies I

- SARAH usually expresses the vertices in fundamental quantities.
- Relations can be defined in `parameters.m` and used via UseDependences->True

Example:

The standard form of the quark-photon vertex is

```
In[23]:= Vertex[{bar[Fd], Fd, VP}]

Out[23]= {{bar[Fd[{gt1, ct1}]], Fd[{gt2, ct2}], VP[{lt3}]},
  {-1/6 i Delta[ct1, ct2] Delta[gt1, gt2] (g1 Cos[ThetaW] - 3 g2 Sin[ThetaW]), LorentzProduct[gamma[lt3], PL]},
  {1/3 i g1 Cos[ThetaW] Delta[ct1, ct2] Delta[gt1, gt2], LorentzProduct[gamma[lt3], PR]}}
```

Dependencies I

- SARAH usually expresses the vertices in fundamental quantities.
- Relations can be defined in `parameters.m` and used via UseDependences->True

Example:

One can use e, Θ_W instead of g_1, g_2 via

```
In[24]:= Vertex[{bar[Fd], Fd, VP}, UseDependences → True]
Out[24]= {{bar[Fd[{gt1, ct1}]], Fd[{gt2, ct2}], VP[{lt3}]},
  {1/3 i e Delta[ct1, ct2] Delta[gt1, gt2], LorentzProduct[gamma[lt3], PL]},
  {1/3 i e Delta[ct1, ct2] Delta[gt1, gt2], LorentzProduct[gamma[lt3], PR]}}
```

Dependencies II

- All dependencies are stored in the list `subDependences`
- This list can be applied also after calculating vertices

Dependencies II

- All dependencies are stored in the list `subDependences`
- This list can be applied also after calculating vertices

Example:

For the SMSSM the defined dependencies are

```
In[25]:= subDependences
Out[25]= {g1 → e Sec[ThetaW], g2 → e Csc[ThetaW],
  ZP[index1_Integer, index2_Integer] :→ {{-Cos[β], Sin[β]}, {Sin[β], Cos[β]}}[[index1, index2]]}
```

Dependencies II

- All dependencies are stored in the list `subDependences`
- This list can be applied also after calculating vertices

Example:

The standard form of the four charged Higgs vertex is

```
In[41]:= Vertex[{Hpm, conj[Hpm], Hpm, conj[Hpm]}]

Out[41]= {{Hpm[{gt1}], conj[Hpm[{gt2}]], Hpm[{gt3}], conj[Hpm[{gt4}]]},
  {1/4 i (ZP[gt1, 2] ((g1^2 + g2^2 - 4 λ conj[λ]) ZP[gt2, 1] ZP[gt3, 1] ZP[gt4, 2] +
        ZP[gt2, 2] ((g1^2 + g2^2 - 4 λ conj[λ]) ZP[gt3, 1] ZP[gt4, 1] - 2 (g1^2 + g2^2) ZP[gt3, 2] ZP[gt4, 2])) +
      ZP[gt1, 1] ((g1^2 + g2^2 - 4 λ conj[λ]) ZP[gt2, 2] ZP[gt3, 2] ZP[gt4, 1] +
        ZP[gt2, 1] (-2 (g1^2 + g2^2) ZP[gt3, 1] ZP[gt4, 1] + (g1^2 + g2^2 - 4 λ conj[λ]) ZP[gt3, 2] ZP[gt4, 2]))), 1}}
```

Dependencies II

- All dependencies are stored in the list `subDependences`
- This list can be applied also after calculating vertices

Example:

One can fix the generation indices and replace afterwards the rotation matrix by the defined angle

```
In[42]:= Vertex[{Hpm, conj[Hpm], Hpm, conj[Hpm]}] /. {gt1 → 1, gt2 → 1, gt3 → 1, gt4 → 1} /. subDependences //
          FullSimplify

Out[42]= {{Hpm[{1}], conj[Hpm[{1}]], Hpm[{1}], conj[Hpm[{1}]]},
          {-1/4 i Csc[ThetaW]^2 Sec[ThetaW]^2 (e^2 (1 + Cos[4 β]) + λ conj[λ] Sin[2 ThetaW]^2 Sin[2 β]^2), 1}}
```

All vertices at once

It is possible to calculate all vertices at once using

Mathematica
```
MakeVertexList[ STATES,  Options]
```

with

- STATES: the eigenstates for which all vertices shall be calculated
- One can define a subset of generic classes which should be considered, e.g. `GenericClasses->{FFS,FFV}`

All vertices at once

It is possible to calculate all vertices at once using

```
Mathematica
MakeVertexList[ STATES,  Options]
```

with

- STATES: the eigenstates for which all vertices shall be calculated
- One can define a subset of generic classes which should be considered, e.g. GenericClasses->{FFS,FFV}

Output:

- The results are lists SA‘VertexList[TYPE] for each generic class: {FFS, FFV, SSS, SSV, SVV, SSVV, VVV, VVVV, GGS, GGV}
- The results are stored in the output directory of the model
- The content of these lists are the information which gets exported into FeynArts, CalcHep or UFO files

Example:

```
In[29]:= MakeVertexList[EWSB]
Generate Directories
Building Particle List
Calculate all vertices
  Three Scalar – Interactions
    Found 26 potential vertices. Calculating 26/26 . (All done in 17.468s; 17 are non–vanishing)
  Four Scalar – Interactions
    Found 65 potential vertices. Calculating 65/65 . (All done in 30.724s; 44 are non–vanishing)
  Two Scalar – One Vector Boson – Interactions
    Found 66 potential vertices. Calculating 66/66 . (All done in 14.8s; 20 are non–vanishing)
  One Scalar – Two Vector Boson – Interactions
    Found 36 potential vertices. Calculating 36/36 . (All done in 8.856s; 6 are non–vanishing)
  Two Scalar – Two Vector Boson – Interactions
    Found 178 potential vertices. Calculating 178/178 . (All done in 46.32s; 46 are non–vanishing)
  Three Vector Boson – Interactions
    Found 21 potential vertices. Calculating 21/21 . (All done in 3.716s; 3 are non–vanishing)
  Two Fermion – One Scalar – Interactions
    Found 84 potential vertices. Calculating 84/84 . (All done in 31.076s; 36 are non–vanishing)
  Two Fermion – One Vector Boson – Interactions
    Found 59 potential vertices. Calculating 59/59 . (All done in 24.212s; 19 are non–vanishing)
  Four Vector Boson – Interactions
    Found 36 potential vertices. Calculating 36/36 . (All done in 12.14s; 5 are non–vanishing)
  Two Ghost – One Vector Boson – Interactions
    Found 65 potential vertices. Calculating 65/65 . (All done in 4.78s; 13 are non–vanishing)
  Two Ghost – One Scalar – Interactions
    Found 48 potential vertices. Calculating 48/48 . (All done in 3.588s; 12 are non–vanishing)
  Two Scalar – One Auxiliary – Interactions
    Found 102 potential vertices. Calculating 102/102 . (All done in 14.492s; 100 are non–vanishing)

Simplify Vertices
Writing vertices to files
All vertices calculated. (Time needed: 212.776s)
The vertices are saved in /home/fnstaub/Desktop/HEP–Tools/SARAH–4.12.2/Output/SMSSM/EWSB/Vertices/
```

Vertices

Example:

the FFS vertices are stored in SA‘VertexList[FFS]:

```
In[43]:= SA`VertexList[FFS]
Out[43]= {{{bar[Cha[{gt1}]], Cha[{gt2}], Ah[{gt3}]},
  {-1/Sqrt[2] (g2 conj[UM[gt2, 1]] conj[UP[gt1, 2]] conj[ZA[gt3, 2]] + conj[UM[gt2, 2]]
      (g2 conj[UP[gt1, 1]] conj[ZA[gt3, 1]] - λ conj[UP[gt1, 2]] conj[ZA[gt3, 3]])), PL},
  {1/Sqrt[2] (g2 conj[ZA[gt3, 1]] UM[gt1, 2] UP[gt2, 1] + (g2 conj[ZA[gt3, 2]] UM[gt1, 1] -
       conj[λ] conj[ZA[gt3, 3]] UM[gt1, 2]) UP[gt2, 2]), PR}}, {{Chi[{gt1}], Chi[{gt2}], Ah[{gt3}]},
  {1/2 (-conj[ZA[gt3, 2]] (conj[ZN[gt1, 4]] (g1 conj[ZN[gt2, 1]] - g2 conj[ZN[gt2, 2]]) +
        Sqrt[2] λ conj[ZN[gt1, 5]] conj[ZN[gt2, 3]] + g1 conj[ZN[gt1, 1]] conj[ZN[gt2, 4]] -
        g2 conj[ZN[gt1, 2]] conj[ZN[gt2, 4]] + Sqrt[2] λ conj[ZN[gt1, 3]] conj[ZN[gt2, 5]]) -
     conj[ZA[gt3, 1]] (conj[ZN[gt1, 3]] (-g1 conj[ZN[gt2, 1]] + g2 conj[ZN[gt2, 2]]) -
        g1 conj[ZN[gt1, 1]] conj[ZN[gt2, 3]] + g2 conj[ZN[gt1, 2]] conj[ZN[gt2, 3]] +
        Sqrt[2] λ conj[ZN[gt1, 5]] conj[ZN[gt2, 4]] + Sqrt[2] λ conj[ZN[gt1, 4]] conj[ZN[gt2, 5]]) -
...
```

Example:

Mathematica commands can be used to filter & select subgroups of vertices, e.g. all FFS vertices with a pseudo-scalar are returned by

```
In[44]:= Select[SA`VertexList[FFS], FreeQ[#, Ah] == False &]

Out[44]= {{{bar[Cha[{gt1}]], Cha[{gt2}], Ah[{gt3}]},
  {-1/√2 (g2 conj[UM[gt2, 1]] conj[UP[gt1, 2]] conj[ZA[gt3, 2]] + conj[UM[gt2, 2]]
      (g2 conj[UP[gt1, 1]] conj[ZA[gt3, 1]] - λ conj[UP[gt1, 2]] conj[ZA[gt3, 3]])), PL},
  {1/√2 (g2 conj[ZA[gt3, 1]] UM[gt1, 2] UP[gt2, 1] + (g2 conj[ZA[gt3, 2]] UM[gt1, 1] -
      conj[λ] conj[ZA[gt3, 3]] UM[gt1, 2]) UP[gt2, 2]), PR}}, {{Chi[{gt1}], Chi[{gt2}], Ah[{gt3}]},
  {1/2 (-conj[ZA[gt3, 2]] (conj[ZN[gt1, 4]] (g1 conj[ZN[gt2, 1]] - g2 conj[ZN[gt2, 2]]) +
        √2 λ conj[ZN[gt1, 5]] conj[ZN[gt2, 3]] + g1 conj[ZN[gt1, 1]] conj[ZN[gt2, 4]] -
        g2 conj[ZN[gt1, 2]] conj[ZN[gt2, 4]] + √2 λ conj[ZN[gt1, 3]] conj[ZN[gt2, 5]]) -
     conj[ZA[gt3, 1]] (conj[ZN[gt1, 3]] (-g1 conj[ZN[gt2, 1]] + g2 conj[ZN[gt2, 2]]) -
        g1 conj[ZN[gt1, 1]] conj[ZN[gt2, 3]] + g2 conj[ZN[gt1, 2]] conj[ZN[gt2, 3]] +
        √2 λ conj[ZN[gt1, 5]] conj[ZN[gt2, 4]] + √2 λ conj[ZN[gt1, 4]] conj[ZN[gt2, 5]]) -
     √2 conj[ZA[gt3, 3]] (λ conj[ZN[gt1, 4]] conj[ZN[gt2, 3]] + λ conj[ZN[gt1, 3]] conj[ZN[gt2, 4]] -
        2 κ conj[ZN[gt1, 5]] conj[ZN[gt2, 5]])), PL},
  {1/2 (-conj[ZA[gt3, 1]] (ZN[gt1, 3] (g1 ZN[gt2, 1] - g2 ZN[gt2, 2]) + g1 ZN[gt1, 1] ZN[gt2, 3] - g2 ZN[gt1, 2]
...
```

Renormalisation Group Equations

RGE calculation

The one- and two-loop RGEs for a model are calculated via

Mathematica
```
CalcRGES[ Options]
```

The possible options are

- `TwoLoop -> True/False`
- `ReadLists -> True/False` (reading previous results)
- `VariableGenerations -> FIELDS` (consider number of generations as free variable)
- `NoMatrixMultiplication -> True/False` (use explicit sums instead of matrix multiplication)
- `IgnoreAt2Loop -> PARAMETERS` (ignore some parameters at two-loop)
- `WriteFunctionsToRun->True/False` (write a file to evaluate the RGEs numerically in `Mathematica`)

```
In[45]:= CalcRGEs[]
Calculate supersymmetric RGEs
Making Lists of Particles and Couplings
Calculating anomalous Dimensions
Calculate Beta Functions for trilinear Superpotential parameters
   Calculating 5/5.(All done in 1.004s)
Calculate Beta Functions for bilinear Superpotential parameters
   Calculating 2/2.(All done in 0.208s)

...
Calculate Beta Functions for VEVs
   Calculating 3/3.(All done in 0.396s)
Writing Mathematica code to evaluate RGEs

Finished with the calculation of the RGEs. Time needed: 43.664s
The results are saved in /home/fnstaub/Desktop/HEP-Tools/SARAH-4.12.2/Output/SMSSM/RGEs/
```

RGEs

Results

The results are stored in three dimensional arrays containing

```
{Parameter, 1-loop Beta-Fkt., 2-loop Beta-Fkt}
```

Results

The results are stored in three dimensional arrays containing

```
{Parameter, 1-loop Beta-Fkt., 2-loop Beta-Fkt}
```

The names of the arrays **for SUSY models** are

- `Gij`: Anomalous dimensions of all chiral superfields
- `BetaWijkl`: Quartic superpotential parameters
- `BetaYijk`: Trilinear superpotential parameters
- `BetaMuij`: Bilinear superpotential parameters
- `BetaLi`: Linear superpotential parameters
- `BetaQijkl`: Quartic soft-breaking parameters
- `BetaTijk`: Trilinear soft-breaking parameters
- `BetaBij`: Bilinear soft-breaking parameters
- `BetaSLi`: Linear soft-breaking parameters
- `Betam2ij`: Scalar squared masses
- `BetaMi`: Majorana Gaugino masses
- `BetaGauge`: Gauge couplings
- `BetaVEVs`: VEVs
- `BetaDGi`: Dirac gaugino mass terms

Results

The results are stored in three dimensional arrays containing

```
{Parameter, 1-loop Beta-Fkt., 2-loop Beta-Fkt}
```

The names of the arrays **for non-SUSY models** are

- `Gij`: Anomalous dimensions of all fermions and scalars
- `BetaGauge`: Gauge couplings
- `BetaLijkl`: Quartic scalar couplings
- `BetaYijk`: Interactions between two fermions and one scalar
- `BetaTijk`: Cubic scalar interactions
- `BetaMuij`: Bilinear fermion term
- `BetaBij`: Bilinear scalar term
- `BetaVEVs`: Vacuum expectation values

Results

Karlsruher Institut für Technologie

The results are stored in three dimensional arrays containing

```
{Parameter, 1-loop Beta-Fkt., 2-loop Beta-Fkt}
```

Example:

The β-functions of the gauge couplings are

```
In[46]:= BetaGauge
```

```
Out[46]= {{g1, 33 g1^3/5, 1/25 g1^3 (199 g1^2 + 135 g2^2 + 440 g3^2 -
      30 λ conj[λ] - 70 trace[Yd, Adj[Yd]] - 90 trace[Ye, Adj[Ye]] - 130 trace[Yu, Adj[Yu]])},
  {g2, g2^3, 1/5 g2^3 (9 g1^2 + 125 g2^2 + 120 g3^2 - 10 λ conj[λ] - 30 trace[Yd, Adj[Yd]] -
      10 trace[Ye, Adj[Ye]] - 30 trace[Yu, Adj[Yu]])},
  {g3, -3 g3^3, 1/5 g3^3 (11 g1^2 + 45 g2^2 + 70 g3^2 - 20 trace[Yd, Adj[Yd]] - 20 trace[Yu, Adj[Yu]])}}
```

- Coefficients $1/16\pi^2$, $1/(16\pi^2)^2$ are dropped

Results

The results are stored in three dimensional arrays containing

```
{Parameter, 1-loop Beta-Fkt., 2-loop Beta-Fkt}
```

Example:

The β-functions of the gauge couplings are

```
In[46]:= BetaGauge

Out[46]= {{g1, 33 g1^3/5, 1/25 g1^3 (199 g1^2 + 135 g2^2 + 440 g3^2 -
        30 λ conj[λ] - 70 trace[Yd, Adj[Yd]] - 90 trace[Ye, Adj[Ye]] - 130 trace[Yu, Adj[Yu]])},
     {g2, g2^3, 1/5 g2^3 (9 g1^2 + 125 g2^2 + 120 g3^2 - 10 λ conj[λ] - 30 trace[Yd, Adj[Yd]] -
        10 trace[Ye, Adj[Ye]] - 30 trace[Yu, Adj[Yu]])},
     {g3, -3 g3^3, 1/5 g3^3 (11 g1^2 + 45 g2^2 + 70 g3^2 - 20 trace[Yd, Adj[Yd]] - 20 trace[Yu, Adj[Yu]])}}
```

- Coefficients $1/16\pi^2$, $1/(16\pi^2)^2$ are dropped
- 1-loop, 2-loop

Results

The results are stored in three dimensional arrays containing

```
{Parameter, 1-loop Beta-Fkt., 2-loop Beta-Fkt}
```

Example:

The β-functions of T_κ are

```
In[49]:= BetaTijk[[4]]
Out[49]= {T[κ], 6 (3 κ conj[κ] T[κ] + conj[λ] (λ T[κ] + 2 κ T[λ])),
  -6/5 (100 κ^2 conj[κ]^2 T[κ] + 10 λ conj[λ]^2 (λ T[κ] + 4 κ T[λ]) +
    conj[λ] (λ T[κ] (-3 g1^2 - 15 g2^2 + 60 κ conj[κ] + 15 trace[Yd, Adj[Yd]] + 5 trace[Ye, Adj[Ye]] +
        15 trace[Yu, Adj[Yu]]) + 2 κ (T[λ] (-3 g1^2 - 15 g2^2 + 20 κ conj[κ] + 15 trace[Yd, Adj[Yd]] +
          5 trace[Ye, Adj[Ye]] + 15 trace[Yu, Adj[Yu]]) + λ (3 g1^2 MassB + 15 g2^2 MassWB +
          15 trace[Adj[Yd], T[Yd]] + 5 trace[Adj[Ye], T[Ye]] + 15 trace[Adj[Yu], T[Yu]]))))}
```

Results

The results are stored in three dimensional arrays containing

```
{Parameter, 1-loop Beta-Fkt., 2-loop Beta-Fkt}
```

Example:

For soft masses often traces `TrX[Y]` appear, e.g. for $m_{H_u}^2$

```
In[48]:= Betam2ij[[4]]

Out[48]= {mHu2, -6/5 g1^2 MassB conj[MassB] - 6 g2^2 MassWB conj[MassWB] + 2 mHd2 λ conj[λ] +
  2 mHu2 λ conj[λ] + 2 ms2 λ conj[λ] + 2 conj[T[λ]] T[λ] + Sqrt[3/5] g1 Tr1[1] + 6 mHu2 trace[Yu, Adj[Yu]] +
  6 trace[conj[T[Yu]], Tp[T[Yu]]] + 6 trace[mq2, Adj[Yu], Yu] + 6 trace[mu2, Yu, Adj[Yu]],
  1/25 (g1^2 conj[MassB] (621 g1^2 MassB + 90 g2^2 MassB + 45 g2^2 MassWB + 80 MassB trace[Yu, Adj[Yu]] -
      40 trace[Adj[Yu], T[Yu]]) + 5 (3 g2^2 (55 g2^2 MassWB + 3 g1^2 (MassB + 2 MassWB)) conj[MassWB] -
      2 (30 (mHd2 + mHu2 + ms2) λ^2 conj[λ]^2 + 10 conj[κ]
        ((mHd2 + mHu2 + 4 ms2) κ λ conj[λ] + conj[T[λ]] (λ T[κ] + κ T[λ])) - 15 g2^4 Tr2[2] - 3 g1^2 Tr2U1[1, 1] -
      2 Sqrt[15] g1 Tr3[1] + 15 conj[T[λ]] T[λ] trace[Yd, Adj[Yd]] + 5 conj[T[λ]] T[λ] trace[Ye, Adj[Ye]] -
...
```

Results

The results are stored in three dimensional arrays containing

`{Parameter, 1-loop Beta-Fkt., 2-loop Beta-Fkt}`

Example:

The expressions for the traces are given in `TraceAbbr`

In[78]:= **TraceAbbr**

Out[78]=

$$\Big\{\Big\{\Big\{\mathtt{Tr1[1]},\ \sqrt{\tfrac{3}{5}}\ \mathtt{g1}\ (-\mathtt{mHd2}+\mathtt{mHu2}+\mathtt{trace[md2]}+\mathtt{trace[me2]}-\mathtt{trace[ml2]}+\mathtt{trace[mq2]}-2\ \mathtt{trace[mu2]})\Big\}\Big\},$$

$$\Big\{\Big\{\mathtt{Tr2U1[1,\,1]},\ \frac{1}{10}\ \mathtt{g1}^2\ (3\ \mathtt{mHd2}+3\ \mathtt{mHu2}+2\ \mathtt{trace[md2]}+6\ \mathtt{trace[me2]}+3\ \mathtt{trace[ml2]}+\mathtt{trace[mq2]}+8\ \mathtt{trace[mu2]})\Big\},$$

$$\Big\{\mathtt{Tr3[1]},\ \frac{1}{20\sqrt{15}}\mathtt{g1}\ \Big(-9\ \mathtt{g1}^2\ \mathtt{mHd2}-45\ \mathtt{g2}^2\ \mathtt{mHd2}+9\ \mathtt{g1}^2\ \mathtt{mHu2}+45\ \mathtt{g2}^2\ \mathtt{mHu2}+30\ (\mathtt{mHd2}-\mathtt{mHu2})\ \lambda\ \mathtt{conj}[\lambda]+$$
$$4\ (\mathtt{g1}^2+20\ \mathtt{g3}^2)\ \mathtt{trace[md2]}+36\ \mathtt{g1}^2\ \mathtt{trace[me2]}-9\ \mathtt{g1}^2\ \mathtt{trace[ml2]}-45\ \mathtt{g2}^2\ \mathtt{trace[ml2]}+$$
$$\mathtt{g1}^2\ \mathtt{trace[mq2]}+45\ \mathtt{g2}^2\ \mathtt{trace[mq2]}+80\ \mathtt{g3}^2\ \mathtt{trace[mq2]}-32\ \mathtt{g1}^2\ \mathtt{trace[mu2]}-160\ \mathtt{g3}^2\ \mathtt{trace[mu2]}+$$
$$90\ \mathtt{mHd2\ trace[Yd,\,Adj[Yd]]}+30\ \mathtt{mHd2\ trace[Ye,\,Adj[Ye]]}-90\ \mathtt{mHu2\ trace[Yu,\,Adj[Yu]]}-$$
$$60\ \mathtt{trace[Yd,\,Adj[Yd],\,conj[md2]]}-30\ \mathtt{trace[Yd,\,conj[mq2],\,Adj[Yd]]}-60\ \mathtt{trace[Ye,\,Adj[Ye],\,conj[me2]]}+$$
$$30\ \mathtt{trace[Ye,\,conj[ml2],\,Adj[Ye]]}+120\ \mathtt{trace[Yu,\,Adj[Yu],\,conj[mu2]]}-30\ \mathtt{trace[Yu,\,conj[mq2],\,Adj[Yu]]}\Big)\Big\},$$

$$\Big\{\mathtt{Tr2[2]},\ \frac{1}{2}\ (\mathtt{mHd2}+\mathtt{mHu2}+\mathtt{trace[ml2]}+3\ \mathtt{trace[mq2]})\Big\},\ \Big\{\mathtt{Tr2[3]},\ \frac{1}{2}\ (\mathtt{trace[md2]}+2\ \mathtt{trace[mq2]}+\mathtt{trace[mu2]})\Big\}\Big\}\Big\}$$

Running in Mathematica

- SARAH writes the RGEs also in a **format which can be used directly with Mathematica**
- This format is saved in the file RunRGEs.m
- Also a **function RunRGEs to run the RGEs** is provided in this file. The syntax is

Mathematica
```
RunRGEs[ input, scale1, scale2,  Options]
```

Running in Mathematica

- SARAH writes the RGEs also in a **format which can be used directly with Mathematica**
- This format is saved in the file RunRGEs.m
- Also a **function RunRGEs to run the RGEs** is provided in this file. The syntax is

Mathematica
```
RunRGEs[ input, scale1, scale2,  Options]
```

- non-vanishing boundary conditions at the scale where the running starts

Running in Mathematica

- SARAH writes the RGEs also in a **format which can be used directly with Mathematica**
- This format is saved in the file RunRGEs.m
- Also a **function RunRGEs to run the RGEs** is provided in this file. The syntax is

Mathematica

```
RunRGEs[ input, scale1, scale2,  Options]
```

- non-vanishing boundary conditions at the scale where the running starts
- log of the scale where the running starts

Running in Mathematica

- SARAH writes the RGEs also in a **format which can be used directly with Mathematica**
- This format is saved in the file RunRGEs.m
- Also a **function RunRGEs to run the RGEs** is provided in this file. The syntax is

Mathematica
```
RunRGEs[ input, scale1, scale2,  Options]
```

- non-vanishing boundary conditions at the scale where the running starts
- log of the scale where the running starts
- log of the scale where the running stops

Running in Mathematica

- SARAH writes the RGEs also in a **format which can be used directly with Mathematica**
- This format is saved in the file RunRGEs.m
- Also a **function RunRGEs to run the RGEs** is provided in this file. The syntax is

Mathematica
```
RunRGEs[ input, scale1, scale2,  Options]
```

- non-vanishing boundary conditions at the scale where the running starts
- log of the scale where the running starts
- log of the scale where the running stops
- option to turn off two-loop running (TwoLoop->False)

Examples:

Examples:

- Loading `RunRGEs.m`
- Running the gauge couplings as well as Y_t & λ from 10^3 to 10^{17} GeV

```
In[52]:= << "SARAH-4.12.2/Output/SMSSM/RGEs/RunRGEs.m"

In[54]:= runRGEs = RunRGEs[{g1 → 0.47, g2 → 0.65, g3 → 1.1, Yu[3, 3] → 0.9, T[λ] → 0, λ → 0.6}, 3, 17][[1]]

Out[54]= {L[L1] → InterpolatingFunction[ Domain: {{3., 17.}} Output: scalar ],

         B[μ] → InterpolatingFunction[ Domain: {{3., 17.}} Output: scalar ],
```

→ **the results are stored as interpolating function**

These functions are used as

```
Mathematica
PARAMTER[SCALE] /. InterpolatingFunction
```

to get the value of a parameter at any scale

Examples:

One can use the function to get values of the running gauge couplings at the GUT scale:

```
In[55]:= {g1[16], g2[16], g3[16], Yu[3, 3][16], λ[16]} /. runRGEs
Out[55]= {0.709092, 0.720759, 0.719522, 0.644797, 0.956999}
```

Examples:

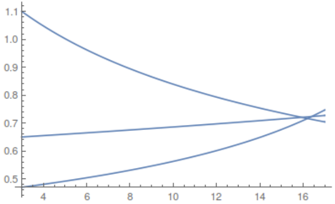
One can use the function to get values of the running gauge couplings at the GUT scale:

```
In[55]:= {g1[16], g2[16], g3[16], Yu[3, 3][16], λ[16]} /. runRGEs
Out[55]= {0.709092, 0.720759, 0.719522, 0.644797, 0.956999}
```

One can also make plots to show the running:

```
In[56]:= Plot[{g1[t], g2[t], g3[t]} /. runRGEs, {t, 3, 17}]
```

Out[56]=



Examples:

SUSY boundary conditions at the GUT scale can be set:

```
In[66]:= runRGEsSUSY =
  RunRGEs[{g1 → (g1[16] /. runRGEs), g2 → (g2[16] /. runRGEs), g3 → (g3[16] /. runRGEs), T[λ] → 0,
     λ → (λ[16] /. runRGEs), Yu[3, 3] → (Yu[3, 3][16] /. runRGEs), mq2[a__] → Delta[a] m0 ^ 2,
     mu2[a__] → Delta[a] m0 ^ 2, md2[a__] → Delta[a] m0 ^ 2, me2[a__] → Delta[a] m0 ^ 2,
     ml2[a__] → Delta[a] m0 ^ 2, ms2 → m0 ^ 2, mHd2 → m0 ^ 2, mHu2 → m0 ^ 2, MassB → M12, MassWB → M12,
     MassG → M12} /. {m0 → 200, M12 → 300}, 16, 3][[1]];
```

Examples:

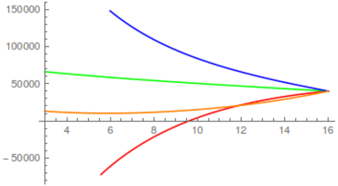
SUSY boundary conditions at the GUT scale can be set:

```
In[66]:= runRGEsSUSY =
  RunRGEs[{g1 → (g1[16] /. runRGEs), g2 → (g2[16] /. runRGEs), g3 → (g3[16] /. runRGEs), T[λ] → 0,
     λ → (λ[16] /. runRGEs), Yu[3, 3] → (Yu[3, 3][16] /. runRGEs), mq2[a__] → Delta[a] m0 ^ 2,
     mu2[a__] → Delta[a] m0 ^ 2, md2[a__] → Delta[a] m0 ^ 2, me2[a__] → Delta[a] m0 ^ 2,
     ml2[a__] → Delta[a] m0 ^ 2, ms2 → m0 ^ 2, mHd2 → m0 ^ 2, mHu2 → m0 ^ 2, MassB → M12, MassWB → M12,
     MassG → M12} /. {m0 → 200, M12 → 300}, 16, 3][[1]];
```

. . . and the running of the masses can be plotted:

```
In[76]:= Plot[{mHu2[t] /. runRGEsSUSY, mHd2[t] /. runRGEsSUSY, mu2[3, 3][t] /. runRGEsSUSY, ms2[t] /. runRGEsSUSY},
  {t, 16, 3}, PlotStyle → {Red, Green, Blue, Orange}]
```

Out[76]=

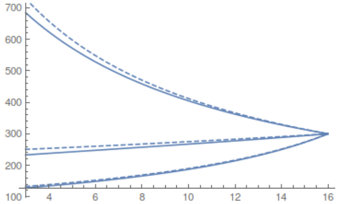


Examples:

Also one- and two-loop running can be compared, e.g. for the gauginos:

```
In[60]:= runRGEsSUSY1L =
  RunRGEs[{g1 → (g1[16] /. runRGEs), g2 → (g2[16] /. runRGEs), g3 → (g3[16] /. runRGEs),
      Yu[3, 3] → (Yu[3, 3][16] /. runRGEs), mq2[a__] → Delta[a] m0 ^ 2, mu2[a__] → Delta[a] m0 ^ 2,
      md2[a__] → Delta[a] m0 ^ 2, me2[a__] → Delta[a] m0 ^ 2, ml2[a__] → Delta[a] m0 ^ 2, mHd2 → m0 ^ 2,
      mHu2 → m0 ^ 2, MassB → M12, MassWB → M12, MassG → M12} /. {m0 → 200, M12 → 300}, 16, 3, TwoLoop → False][[
    1]];

In[61]:= Show[Plot[{MassB[t], MassWB[t], MassG[t]} /. runRGEsSUSY, {t, 16, 3}],
  Plot[{MassB[t], MassWB[t], MassG[t]} /. runRGEsSUSY1L, {t, 16, 3}, PlotStyle → Dashed]]
```

Out[61]=



LaTeX Output

LaTeX Output

All information which we have obtained so far can be exported into LaTeXusing

Mathematica
```
MakeTeX[ Options]
```

The possible options are

- `FeynmanDiagrams -> True/False` (feynman diagrams for all vertices?)
- `ShortForm -> True/False` (write vertices in a more compact form)
- `WriteSARAH -> True/False` (write information about the model implementation in `SARAH`)

LaTeX Output

All information which we have obtained so far can be exported into LaTeXusing

Mathematica

```
MakeTeX[ Options]
```

The possible options are

- `FeynmanDiagrams -> True/False` (feynman diagrams for all vertices?)
- `ShortForm -> True/False` (write vertices in a more compact form)
- `WriteSARAH -> True/False` (write information about the model implementation in `SARAH`)

About Feynman diagrams:

- To draw Feynman diagrams, the package `FeynMF` must be installed
- A batch script is provided to compile all diagrams automatically

Example

Karlsruher Institut für Technologie

Generate the LaTeX files

```
In[77]:= MakeTeX[]
```

Generate LaTeX files

```
Writing Superfields and Superpotential to TeX-File
Writing Particle Content to TeX-File
Write VEVs to TeX-File
Write Flavor Decomposition to TeX-File
Writing Mass Matrices to TeX-File
Writing Tadpole Equations to TeX-File
Writing RGEs to TeX-File
```

TeXOutput :
Loop corrections not calculated so far. Skipping this parts.

Use CalcLoopCorrections[States] to calculated RGEs and start MakeTeX again to include them in the output.

```
Write Clebsch-Gordan Coefficients
Writing Vertices to TeX-File

Done. Output is in /home/fnstaub/Desktop/HEP-Tools/SARAH-4.12.2/Output/SMSSM/EWSB/TeX/
Use Script MakePDF.sh (Linux) or MakePDF.bat (Windows) to generate pdf file.
```

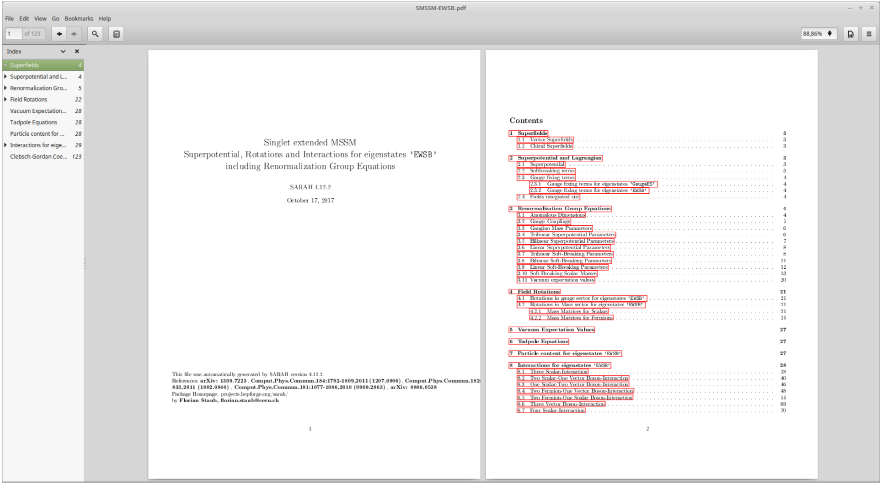
Example

make the script executable and run it

```
fnstaub@fnstaub-ThinkPad-T450s ~/Desktop/HEP-Tools $ cd SARAH-4.12.2/Output/SMSS
M/EWSB/TeX/
fnstaub@fnstaub-ThinkPad-T450s ~/Desktop/HEP-Tools/SARAH-4.12.2/Output/SMSSM/EWS
B/TeX $ chmod 775 MakePDF.sh
fnstaub@fnstaub-ThinkPad-T450s ~/Desktop/HEP-Tools/SARAH-4.12.2/Output/SMSSM/EWS
B/TeX $ ./MakePDF.sh

...

Output written on SMSSM-EWSB.pdf (123 pages, 525889 bytes).
Transcript written on SMSSM-EWSB.log.

PDF for Model finished
Thanks for using SARAH

fnstaub@fnstaub-ThinkPad-T450s ~/Desktop/HEP-Tools/SARAH-4.12.2/Output/SMSSM/EWS
B/TeX $
```

Example

Take a look . . .



Singlet extended MSSM
Superpotential, Rotations and Interactions for eigenstates 'EWSB'
including Renormalization Group Equations

SARAH 4.12.2

October 17, 2017

This file was automatically generated by SARAH version 4.12.2.
References: **arXiv: 1309.7223 , Comput.Phys.Commun.184:1792-1809,2011 (1207.0906) , Comput.Phys.Commun.182:**
833,2011 (1002.0840) , Comput.Phys.Commun.181:1077-1086,2010 (0909.2863) , arXiv: 0806.0538
Package Homepage: projects.hepforge.org/sarah/
by **Florian Staub, florian.staub@cern.ch**

1

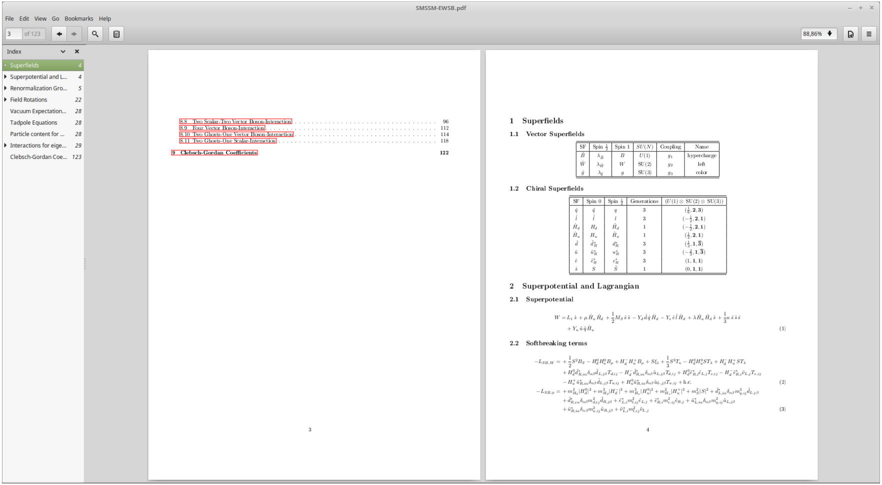
Contents

2

LaTeX Output

Example

Fields and superpotential . . .



3

1 Superfields

1.1 Vector Superfields

SF	Spin $\frac{1}{2}$	Spin 1	$SU(N)$	Coupling	Name
$\hat{B}$	$\lambda_{\tilde{B}}$	B	$U(1)$	g_1	hypercharge
$\hat{W}$	$\lambda_{\tilde{W}}$	W	SU(2)	g_2	left
$\hat{g}$	$\lambda_{\tilde{g}}$	g	SU(3)	g_3	color

1.2 Chiral Superfields

SF	Spin 0	Spin $\frac{1}{2}$	Generations	$(U(1)\otimes \mathrm{SU}(2)\otimes \mathrm{SU}(3))$
$\hat{q}$	$\tilde{q}$	q	3	$(\frac{1}{6},\mathbf{2},\mathbf{3})$
$\hat{l}$	$\tilde{l}$	l	3	$(-\frac{1}{2},\mathbf{2},\mathbf{1})$
$\hat{H}_d$	H_d	$\tilde{H}_d$	1	$(-\frac{1}{2},\mathbf{2},\mathbf{1})$
$\hat{H}_u$	H_u	$\tilde{H}_u$	1	$(\frac{1}{2},\mathbf{2},\mathbf{1})$
$\hat{d}$	$\tilde{d}_R^*$	d_R^*	3	$(\frac{1}{3},\mathbf{1},\overline{\mathbf{3}})$
$\hat{u}$	$\tilde{u}_R^*$	u_R^*	3	$(-\frac{2}{3},\mathbf{1},\overline{\mathbf{3}})$
$\hat{e}$	$\tilde{e}_R^*$	e_R^*	3	$(1,\mathbf{1},\mathbf{1})$
$\hat{s}$	S	$\tilde{S}$	1	$(0,\mathbf{1},\mathbf{1})$

2 Superpotential and Lagrangian

2.1 Superpotential

$$\begin{aligned}W = {} & L_1\,\hat{s} + \mu\,\hat{H}_u\,\hat{H}_d + \frac{1}{2}M_S\,\hat{s}\,\hat{s} - Y_d\,\hat{d}\,\hat{q}\,\hat{H}_d - Y_e\,\hat{e}\,\hat{l}\,\hat{H}_d + \lambda\,\hat{H}_u\,\hat{H}_d\,\hat{s} + \frac{1}{3}\kappa\,\hat{s}\,\hat{s}\,\hat{s} \\ & + Y_u\,\hat{u}\,\hat{q}\,\hat{H}_u \end{aligned} \quad (1)$$

2.2 Softbreaking terms

$$\begin{aligned} -L_{SB,W} = {} & + \frac{1}{2}S^2 B_S - H_d^0 H_u^0 B_\mu + H_d^- H_u^+ B_\mu + S\xi_1 + \frac{1}{3}S^3 T_\kappa - H_d^0 H_u^0 S T_\lambda + H_d^- H_u^+ S T_\lambda \\ & + H_d^0 \tilde{d}^*_{R,i\alpha}\delta_{\alpha\beta}\tilde{d}_{L,j\beta}T_{d,ij} - H_d^- \tilde{d}^*_{R,i\alpha}\delta_{\alpha\beta}\tilde{u}_{L,j\beta}T_{d,ij} + H_d^0 \tilde{e}^*_{R,i}\tilde{e}_{L,j}T_{e,ij} - H_d^- \tilde{e}^*_{R,i}\tilde{\nu}_{L,j}T_{e,ij} \\ & - H_u^+ \tilde{u}^*_{R,i\alpha}\delta_{\alpha\beta}\tilde{d}_{L,j\beta}T_{u,ij} + H_u^0 \tilde{u}^*_{R,i\alpha}\delta_{\alpha\beta}\tilde{u}_{L,j\beta}T_{u,ij} + \text{h.c.} \end{aligned} \quad (2)$$

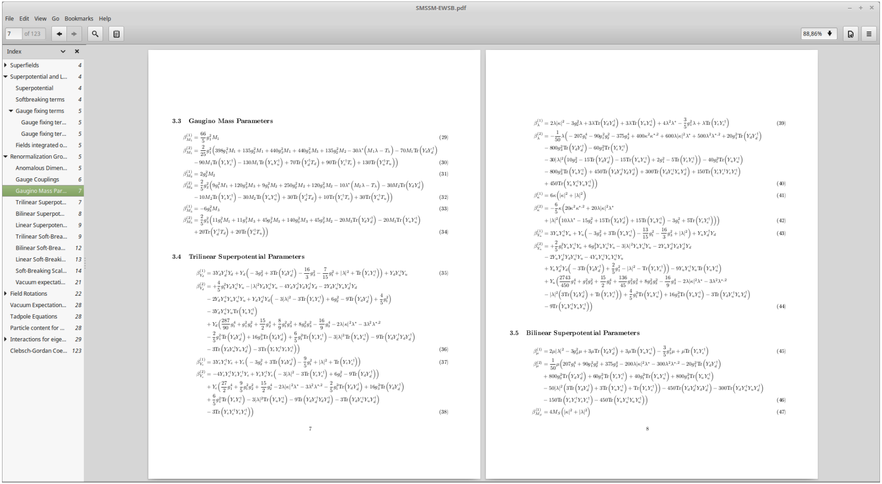
$$\begin{aligned} -L_{SB,\phi} = {} & + m^2_{H_d}|H_d^0|^2 + m^2_{H_d}|H_d^-|^2 + m^2_{H_u}|H_u^0|^2 + m^2_{H_u}|H_u^+|^2 + m_S^2|S|^2 + \tilde{d}^*_{L,i\alpha}\delta_{\alpha\beta}m^2_{q,ij}\tilde{d}_{L,j\beta} \\ & + \tilde{d}^*_{R,i\alpha}\delta_{\alpha\beta}m^2_{d,ij}\tilde{d}_{R,j\beta} + \tilde{e}^*_{L,i}m^2_{l,ij}\tilde{e}_{L,j} + \tilde{e}^*_{R,i}m^2_{e,ij}\tilde{e}_{R,j} + \tilde{u}^*_{L,i\alpha}\delta_{\alpha\beta}m^2_{q,ij}\tilde{u}_{L,j\beta} \\ & + \tilde{u}^*_{R,i\alpha}\delta_{\alpha\beta}m^2_{u,ij}\tilde{u}_{R,j\beta} + \tilde{\nu}^*_{L,i}m^2_{l,ij}\tilde{\nu}_{L,j} \end{aligned} \quad (3)$$

4

LaTeX Output

Example

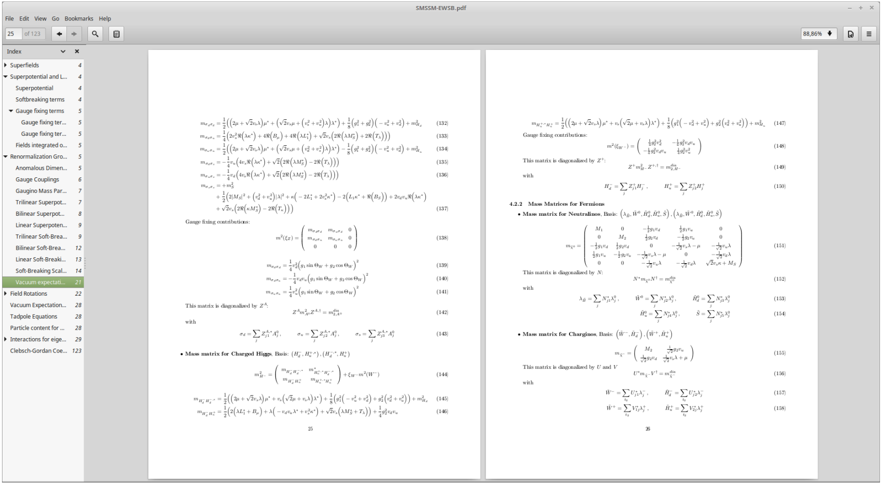
RGEs . . .



LaTeX Output

Example

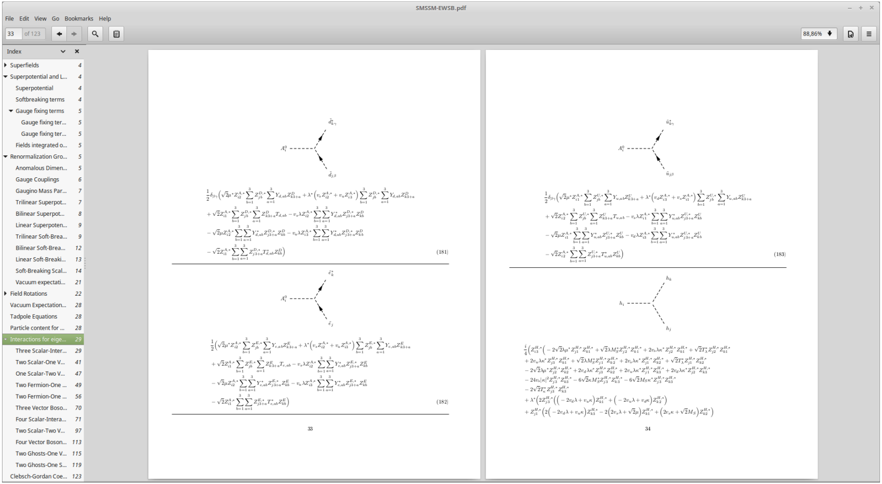
Mass matrices . . .



LaTeX Output

Example

Vertices . . .



LaTeX Output

Summary

- SARAH makes it easy to get analytical information about a given model
- RGE running can easily be performed within Mathematica at the two-loop level
- The LaTeX output provides many information in a human readible form