Introducing RGBeta

v0.4.2, June 10, 2020

A Mathematica package for evaluating Renormalization Group β-functions

Anders Eller Thomsen[1]

Physik-Institut, Universität Zürich, CH-8057 Zürich, Switzerland

1 Package overview

1.1 RGBeta in a nutshell

`RGBeta` is a Mathematica[2] package aimed at allowing theorists to easily extract the $\overline{\text{MS}}$ β-functions of their favorite model, be it for the purposes of BSM physics or for field theoretic applications. The package implements the state-of-the-art β-functions for general four-dimensional renormalizable theories, which are known up to loop order 3–2–2, for gauge, Yukawa, and quartic couplings respectively.

The aim of the package is to provide Mathematica users with an easy-to-use one-stop implementation of the β-functions directly in Mathematica, which is already already heavily used in community for many computer-algebra tasks.[3] Specifying a model to the package can be done in a rather compact manner and can easily be done directly in a Mathematica notebook in which the user can proceed to extract and manipulate the various β-functions. The β-functions can be obtained modularly for the users who wish to experiment with them in the Mathematica notebook environment.

The core of `RGBeta` is set up to evaluate the tensor structures of the general β-function formulas with Einstein summation conventions and various group identities. This approach is similar to what a human might do and is effective because many of the index contractions are in fact Kronecker delta functions or group generators, which can be contracted with the *pattern matching* functionality of Mathematica. The performance of this strategy is generally very good, evaluating e.g. the full set of SM β-functions to highest loop order in a minute on a decent laptop. On top of that, it allows the user to keep e.g. the number of colors or generations arbitrary during the evaluation. The price of the chosen implementation is that it does require the hard-coding of various representation-specific identities. For this reason, only a selection of common representations have presently been implemented, cf. Section 2 for a full list.

[1] aellerthomsen@gmail.com

[2] Mathematica is a product from Wolfram Research, Inc.

[3] Whether we succeeded in this task, we will leave for the users to judge.

1.2 Definition of the β-functions

Given that the `RGBeta` is exclusively given to evaluate the β-functions of various models, it pays to be explicit about their definition, particularly given the many different normalizations used throughout the literature. To reiterate the package exclusively provides the $\overline{\text{MS}}$ β-functions. In all cases we associate a loop expansion to any of the β-functions, writing

$$\beta_g = \sum_\ell \frac{\beta_g^{(\ell)}}{(4\pi)^{2\ell}}. \tag{1}$$

For all non-gauge couplings, both relevant and marginal, the β-functions are defined as the logarithmic derivative of the coupling wrt. the renormalization scale:

$$\beta_g = \frac{\mathrm{d}}{\mathrm{d}\ln\mu} \begin{cases} y_{s_1 f_1 f_2} & \text{for } g = y \in \{\text{Yukawas}\} \\ \lambda_{s_1 s_2 s_3 s_4} & \text{for } g = \lambda \in \{\text{quartics}\} \end{cases} \tag{2}$$

and

$$\beta_g = \frac{\mathrm{d}}{\mathrm{d}\ln\mu} \begin{cases} M_{f_1 f_2} & \text{for } g = M \in \{\text{fermion masses}\} \\ m^2_{s_1 s_2} & \text{for } g = m^2 \in \{\text{scalar masses}\} \\ h_{s_1 s_2 s_3} & \text{for } g = h \in \{\text{trilinears}\} \end{cases}, \tag{3}$$

where f_i and s_i are the flavor indices of the i'th fermion or scalar field of the coupling. In all cases the coupling will be the reference name of the β-function used in the package.

The only deviation from this pattern of β-function definitions is the gauge couplings. We will assume that the kinetic terms of the gauge fields are normalized as

$$\mathscr{L} \supset \sum_n -\frac{1}{4g_n^2}(F_n^{A,\mu\nu})^2, \qquad \beta_{g_n} = \frac{\mathrm{d}g_n^2}{\mathrm{d}\ln\mu} \tag{4}$$

for the fields of non-Abelian gauge groups or the Abelian gauge group in a model with *at most one* Abelian group. For a model with multiple Abelian gauge groups, which therefore features kinetic mixing, the kinetic term for all the Abelian gauge fields are written jointly as

$$\mathscr{L} \supset -\frac{1}{4} F_{i,\mu\nu} h^{-1}_{ij} F_j^{\mu\nu}, \qquad \beta_h = \frac{\mathrm{d}h_{ij}}{\mathrm{d}\ln\mu} \tag{5}$$

where h_{ij} is the symmetric coupling matrix. With the normalization of the gauge fields employed in this notation, the covariant derivative of a matter field can be written as e.g.

$$D_\mu\psi = \partial_\mu\psi - i\sum_n A^A_{n,\mu} T^A_n \psi - i\sum_i q_i V_{i,\mu}\psi, \tag{6}$$

where T_n^A are the Hermitian generators of the non-Abelian representations and q_i are the Abelian charges. In particular, using a coupling matrix for the Abelian kinetic term allows for keeping the coupling of the Abelian vector fields to the matter fields 'diagonal'.

1.3 Installation

At present there is no support for a permanent installation of the package. It is simply plug-and-play: put it in any directory in your system, point your notebook in the right direction, and load the package.

Most simply RGBeta can be run from a Mathematica notebook in the base directory of the package (the one with `RGBeta.m`) with the following lines:

```
SetDirectory @ NotebookDirectory[ ];
<< RGBeta`
```

Alternatively, one can load it from anywhere by providing the path to the directory of the package with

```
AppendTo[$Path, <Directory>];
<< RGBeta`
```

using the path to the `RGBeta` directory in place of <**Directory**>. In either case `RGBeta` ought to be loaded into a freshly initialized Mathematica kernel to avoid clashing symbol definitions.

2 Group Theory

`RGBeta` is set up to use analytic rules for the Einstein summation convention for repeated indices and perform the entire evaluation without using explicit matrix (or array) constructions for the couplings and group tensors. Any one of the matter fields can have multiple indices from various gauge representation or belonging to flavor groups. A single letter, a unique symbol in Mathematica, is used to denote all these many indices by always specifying what representation each index belongs to.

Using implicit summation, `RGBeta` is set up with a number of symbols, for which internal summation rules have been provided using Mathematica **UpValues**, c.f. Tab. 1 for a list. The most fundamental symbol used in the package is the Kronecker delta, δ_{ab}, which is represented by **del**[*rep*, *a*, *b*]. Note in particular the first argument specifying the type of the indices a, b. The summation conventions assigned to the Kronecker delta, for instance, allows it to evaluate as

```
In[1] :=   del[rep, a, b] del[rep, c, b] del[rep, c, a]
Out[1] :=  Dim[rep]
```

for any representation rep. As mentioned, identically named indices of different types do not sum, so e.g.

```
In[2] :=   del[rep1, a, b] del[rep2, a, b] del[rep2, c, b] del[rep1, b, a]
Out[2] :=  del[rep2,a,c] Dim[rep1]
```

The up-value approach is extended to the treatment of gauge generators, **tGen**[*rep*, *A*, *a*, *b*]. These are set to evaluate to group constants so that with, e.g., an irreducible representation rep, `RGBeta` gives

Symbol	Interpretation
Bar[*x*]	Represents complex conjugation, x^*. It is used general purpose for both fields, couplings, and group representations.
Trans[*x*]	Represents the transposed of a coupling with 2 indices.
del[*rep*, *a*, *b*]	Represents the Kronecker delta δ_{ab}, with indices running in the group representation or flavor group specified by *rep*.
eps[*rep*, *a*, *b*]	Represents the 2-index antisymmetric invariant ϵ_{ab} of a pseudoreal representation, *rep*, such as the fundamental of an Sp(N) group.
lcSymb[*rep*, *a*, *b*,...]	Represents a Levi-Civita symbol with indices in the given representation
tGen[*rep*, *A*, *a*, *b*].	Represents a Hermitian group generator $T^{Aa}{}_b$ for the corresponding representation. The first index, A, is always taken to be in the adjoint representation, while the other two belong to rep.
delIndex[*rep*, *a*, *b*]	Represents the Kronecker delta δ_{ab} where one of a, b is an integer pointing to a definite index value, e.g. δ_{a2}.
Dim[*rep*]	Specifies the dimension of a representation or flavor index. The gauge representations are mostly predefined, but the dimension of flavor indices can be set by the user.
Matrix[*A*, *B*,...][*a*, *b*]	Represents the matrix product of multiple couplings with 2 or fewer indices, e.g. $(AB)_{ab} = A_{ac}B_{cb}$. Couplings with 1 index are interpreted as column vectors.
Tensor[*A*][*a*,...]	Represents a tensor coupling, A, with three or more indices. No coupling contractions involving tensors are supported.

Table 1: A list of various symbols and their meaning in RGBeta.

```
In[3] :=   tGen[rep, A, a, b] tGen[rep, A, b, c]
In[4] :=   tGen[rep, A, a, b] tGen[rep, B, b, a]
Out[3] :=  Casimir2[rep] del[rep, a, c]
Out[4] :=  TraceNormalization[rep] del[Head[rep][adj], A, B]
```

Here **Casimir2**[rep] is the value of the quadratic Casimir of the representation and **TraceNormalization**[rep] is the trace normalization/Dynkin index. Once a Lie Group has been initialized, the values of the Casimir operators and Dynkin indices will be assigned specific values for the implemented representations.

The tensor basis occurring of the 3–2–2 β-functions has been chosen such as too minimize the occurrence of non-trivial group structures stemming from the gauge groups. Despite this, there remains a couple of them that cannot be evaluated by identifying factors of quadratic Casimir operators or Dynkin indices. These tensors are dealt with by implementing the Fierz identities of the fundamental representations of the groups, cf. Table 3. In the SM, these are necessary in the 3-loop gauge β-functions and in the quartic β-function already from 1-loop order.

The need for Fierz identities, amongst other reasons, is an obstacle to implementing

Representation		$d(R)$	$S_2(R)$	$C_2(R)$	RGBeta reference
SU(N)	**N**	N	$\frac{1}{2}$	$\frac{N^2-1}{2N}$	Gr[fund]
	G	N^2-1	N	N	Gr[adj]
	$\mathbf{S}_2$	$\frac{N(N+1)}{2}$	$\frac{N+2}{2}$	$\frac{(N-1)(N+2)}{N}$	Gr[S2]
	$\mathbf{A}_2$	$\frac{(N-1)N}{2}$	$\frac{N-2}{2}$	$\frac{(N-2)(N+1)}{N}$	Gr[A2]
SO(N)	**N**	N	$\frac{1}{2}$	$\frac{(N-1)}{4}$	Gr[fund]
	G	$\frac{(N-1)N}{2}$	$\frac{N-2}{2}$	$\frac{N-2}{2}$	Gr[adj]
	$\mathbf{S}_2$	$\frac{(N-1)(N+2)}{2}$	$\frac{N+2}{2}$	$\frac{N}{2}$	Gr[S2]
Sp(N)	**N**	N	$\frac{1}{2}$	$\frac{(N+1)}{4}$	Gr[fund]
	G	$\frac{N(N+1)}{2}$	$\frac{N+2}{2}$	$\frac{N+2}{2}$	Gr[adj]
	$\mathbf{A}_2$	$\frac{(N-2)(N+1)}{2}$	$\frac{N-2}{2}$	$\frac{N}{2}$	Gr[A2]
U(1)	q	1	q^2	q^2	Gr[q]
U(1)^n	$(q_1,\ldots,q_n)$	1	—	—	Gr[{$q1$, $q2$,...}]

Table 2: Group constants for the representations of the Lie Groups implemented in RGBeta. The group representations are all referred to by Gr[rep] in RGBeta, where Gr is the name chosen for the specific group. Conjugate representations are called with **Bar**[Gr[rep]].

Group	Fierz identity
SU(N)	$T^{Ai}{}_j T^{Ak}{}_\ell = \frac{1}{2}\delta^i{}_\ell \delta^k{}_j - \frac{1}{2N}\delta^i{}_j \delta^k{}_\ell$
SO(N)	$T^A_{ij}\, T^A_{k\ell} = \frac{1}{4}\left(\delta_{i\ell}\delta_{jk} - \delta_{ik}\delta_{j\ell}\right)$
Sp(N)	$T^{Ai}{}_j T^{Ak}{}_\ell = \frac{1}{4}\left(\epsilon^{ik}\epsilon_{j\ell} + \delta^i{}_\ell \delta^k{}_j\right)$

Table 3: Fierz identities for the fundamental representations of the ordinary, compact Lie groups.

arbitrary representations of the gauge groups. At present `RGBeta` is thus restricted to the representations and groups listed in Table 2. In addition to fundamental and adjoint representations, **N** and **G**, of the ordinary groups, it includes the two-index symmetric and antisymmetric, $\mathbf{S}_2$ and $\mathbf{A}_2$, of SU(N); the traceless two-index symmetric, $\mathbf{S}_2$, of SO(N); and the two-index antisymmetric that vanishes upon contraction with ϵ_{ab}, $\mathbf{A}_2$, of Sp(N). For all representations, the generators can be written in terms of the generator of the fundamental representation after which the Fierz identities can be employed. The $U(1)^n$ groups support representations with any charge assignments. The limit to these few, if frequently used, representations of the Lie groups is the main limitation of `RGBeta`. None of the exceptional Lie groups or their representations are supported at present.

3 Using RGBeta

This section describes how to use `RGBeta` in some detail. For further reference please see the Appendix for a more detailed documentation for the various functions. We heartily recommend prospective users to have a look at the tutorial notebook in `Documentation/Tutorial.nb`, which provides some practical uses cases with comments.

3.1 Routines

To use the `RGBeta` package one should start by specifying a model. This is done by defining gauge groups, matter fields, and couplings using the following functions:

- **AddFermion**[*field*, **Options**] defines a Weyl spinor field.
- **AddFermionMass**[*coupling*, {*psi1*, *psi2*}, **Options**] defines a mass term between two fermion fields, *psi1* and psi2.
- **AddGaugeGroup**[*coupling*, *groupName*, *lieGroup*[*n*], **Options**] defines a gauge group of type *lieGroup*[*n*] with reference name *groupName* and associated coupling *coupling*.
- **AddQuartic**[*coupling*, {*phi1*, *phi2*, *phi3*, *phi4*}, **Options**] defines a quartic coupling between four scalar fields.
- **AddScalar**[*field*, **Options**] defines a scalar field.
- **AddScalarMass**[*coupling*, {psi1, psi2}, **Options**] defines a scalar mass term, between two scalar fields.
- **AddTrilinear**[*coupling*, {*phi1*, *phi2*, *phi3*}, **Options**] defines a trilinear coupling between three scalar fields.
- **AddYukawa**[*coupling*, {*phi*, *psi1*, *psi2*}, **Options**] defines a Yukawa coupling between a scalar field, *phi*, and two fermion fields, *psi1* and psi2.
- **DefineLieGroup**[*groupName*, *lieGroup*[*n*], **Options**] defines a global symmetry group of type *lieGroup*[*n*] with reference name *groupName*.
- **CheckProjection**[*coupling*] returns the specific coupling projector applied to the generic vertex of the appropriate type.
- **ResetModel**[] clears the kernel of all model definitions.

Once the model has been loaded into the kernel the β-functions can be extracted and refined using the functions

- **BetaFunction**[*coupling*, *loop*, **Options**] returns the β-function of the *coupling* evaluated at loop order *loop*.
- **BetaTerm**[*coupling*, *loop*] returns the *loop*-loop term of the *coupling* β-function.
- **Finalize**[*expr*, **Options**] returns a refined version of emph, which can be any expression such as is given by the β-functions, $\beta_g^{(\ell)}$.
- **QuarticBetaFunctions**[*loop*, **Options**] returns all the quartic β-functions up to the given loop order, fully diagonalizing the coupling projectors.
- **SetReal**[*symbol*,...] defines a symbol (e.g. a coupling) to be treated as real.

We seek to explain the basic use of the package in the next subsection.

3.2 Setting up a model

This section details the core usage of `RGBeta`, using the SM as a concrete example due to its familiarity for most/all users. The package is set to work with Weyl spinors to be able to treat all fermions in the same manner. In this notation the matter fields have charge assignments

$$\begin{aligned} &q \in (\mathbf{3},\, \mathbf{2},\, 1/6), && \bar{u} \in (\overline{\mathbf{3}},\, \mathbf{1},\, 2/3), && \bar{d} \in (\overline{\mathbf{3}},\, \mathbf{1},\, -1/3), \\ &\ell \in (\mathbf{1},\, \mathbf{2},\, 1/2), && \bar{e} \in (\mathbf{1},\, \mathbf{1},\, -1), && H \in (\mathbf{1},\, \mathbf{2},\, 1/2) \end{aligned} \tag{7}$$

under the gauge group $G_{\rm SM} = \mathrm{SU}(3)_c \times \mathrm{SU}(2)_{\rm L} \times \mathrm{U}(1)_Y$. There are three Yukawa couplings given by

$$\mathscr{L}_{\rm yuk} = -y_u^{ij} H_\alpha^* \epsilon^{\alpha\beta} q^{\dagger i}{}_{c\beta} \bar{u}^{\dagger jc} - y_d^{ij} H^\alpha q^{\dagger i}{}_{c\alpha} \bar{d}^{jc} - y_e^{ij} H^\alpha \ell^{\dagger i}{}_\alpha \bar{e}^{\dagger j} \;+\; \text{H.c.}\,, \tag{8}$$

where c is a color index, α, β are $\mathrm{SU}(2)_{\rm L}$ indices, and i, j are generation indices. The reason for specifying the Yukawa couplings with right-handed (Hermitian conjugated) fermions is to match the conventional definition of the couplings in the Dirac notation.[4] The SM also contains a scalar potential with a quartic Higgs interaction and a Higgs mass term:

$$\mathscr{L}_V = -M^2 H_\alpha^* H^\alpha - \tfrac{1}{2}\lambda (H_\alpha^* H^\alpha)^2. \tag{9}$$

The `RGBeta` package has automated most of the process of getting the β-functions. It is, however, unavoidable that the user will have to specify the model for themselves. Specifying gauge groups and scalar and fermion fields content is fairly straight-forward. Arguably the most difficult part of using `RGBeta` is to specify the Yukawa and quartic interaction. Here the user must manually specify how the flavor and gauge indices are contracted, which is a potential source of errors.

[4] With Dirac fields, the SM Yukawa couplings are typically written as

$$\mathscr{L}_{\rm yuk} = -y_u^{ij} H_\alpha^* \epsilon^{\alpha\beta} \overline{Q}_{\rm L}{}^{i}{}_{c\beta} U_{\rm R}{}^{jc} - y_d^{ij} H^\alpha \overline{Q}_{\rm L}{}^{i}{}_{c\alpha} D_{\rm R}{}^{jc} - y_e^{ij} H^\alpha \overline{L}_{\rm L}{}^{i}{}_\alpha E_{\rm R}{}^{j} \;+\; \text{H.c.}\,,$$

The first thing one must do when defining a model is to specify the gauge groups to `RGBeta`. This is done using the **AddGaugeGroup** function. For each product group, one needs simply to specify the coupling and a Lie group. The choice of Lie group will then set the group invariants and generator properties of the supported representations. Each group must also be given a unique name, which is used for referencing the group representations.

For the SM, the $\mathrm{SU}(3)_c \times \mathrm{SU}(2)_\mathrm{L} \times \mathrm{U}(1)_Y$ gauge group is added with

```
AddGaugeGroup[g1, U1Y, U1]
AddGaugeGroup[g2, SU2L, SU[2]]
AddGaugeGroup[g3, SU3c, SU[3]]
```

Having set up the gauge group, the next step is adding the matter content. This is specified with the **AddFermion** and **AddScalar** functions. It is sufficient to know the charges and flavor indices to do so. As discussed previously, all fermions must be added as left-handed chiral fields. If any of the fields in the model are given as right-handed fermions, simply add it as a left-handed spinor in the *conjugate* representation under the gauge groups.

The fermions are specified, with a field name and all its non-trivial gauge charges are given as a list with the GaugeRep option, cf. Tab. 2. Given that the fields come in three generation they are also given a single flavor index with the FlavorIndices option.[5] To add the fermions to the model, we load the commands

```
AddFermion[q, GaugeRep → {U1Y[1/6], SU2L[fund], SU3c[fund]}, FlavorIndices → {gen}]
AddFermion[u, GaugeRep → {U1Y[−2/3], Bar @ SU3c[fund]}, FlavorIndices → {gen}]
AddFermion[d, GaugeRep → {U1Y[1/3], Bar @ SU3c[fund]}, FlavorIndices → {gen}]
AddFermion[l, GaugeRep → {U1Y[−1/2], SU2L[fund]}, FlavorIndices → {gen}]
AddFermion[e, GaugeRep → {U1Y[1]}, FlavorIndices → {gen}]
Dim[gen] = ng; (* 3 *)
```

The last line may seem a bit curious. It is simply there to specify the dimension of the flavor index 'gen'. In the SM there are, of course, three generations, but it is often kept as a free parameter in β-function computations. Either can be used in `RGBeta`.

We proceed to add the Higgs field in a similar manner. It has just the one generation, so there is no need to give it any flavor indices:

```
AddScalar[H, GaugeRep → {U1Y[1/2], SU2L[fund]}]
```

The package assumes scalar fields to be complex valued by default, as is the Higgs field. If we wished to add a real scalar field it is sufficient to specify the option SelfConjugate→ **True**.

The most involved part of defining the model, is setting up the couplings. In particular, one must take care when defining the contraction of all field indices and specify the indices of the couplings. The particulars of this should be passed to the package in terms of pure functions. It is *important* to always remember to surround pure functions given as arguments with parentheses. Otherwise the function call will not properly evaluate.

[5] The flavor group of the SM is really $\mathrm{SU}(3)^5$, so one can argue that there really are five distinct generation indices. This does not matter for our purposes here.

Let us begin by listing the code needed to specify the SM Yukawa couplings before we dissect it:

```
AddYukawa[yu, {H, q, u},
    GroupInvariant → (del[SU3c @ fund, #2, #3] eps[SU2L @ fund, #1, #2] &),
    CouplingIndices → ({gen[#2], gen[#3]} &),
    Chirality → Right]
AddYukawa[yd, {Bar @ H, q, d},
    GroupInvariant → (del[SU3c @ fund, #2, #3] del[SU2L @ fund, #1, #2] &),
    CouplingIndices → ({gen[#2], gen[#3]} &),
    Chirality → Right]
AddYukawa[ye, {Bar @ H, l, e},
    GroupInvariant → (del[SU2L @ fund, #1, #2] &),
    CouplingIndices → ({gen[#2], gen[#3]} &),
    Chirality → Right]
```

The first **AddYukawa** call specifies the up-type Yukawa coupling. The first two arguments gives the coupling name, yu, and the fields entering the interaction, {H, q, u}. The first field is always taken to be the scalar. Do bear in mind that the order of the fields matter for the construction of the index contractions, which is the next part.

The GroupInvariant option should be given a pure function with three arguments that specifies the contraction between the indices of the fields. As the name suggests, this should be a gauge invariant object. When the coupling is deployed the group invariant will be passed the three index labels of the fields as its three arguments.

For the up-type Yukawa, the **del**[SU3c @ fund, #2, #3] part of the group invariant establishes that the fundamental indices $\mathrm{SU}(3)_c$ of the second and third fields are contracted with a Kronecker delta. Furthermore, the fundamental $SU(2)_\mathrm{L}$ index of the Higgs (the first field) is contracted with an ϵ invariant of $\mathrm{SU}(2)_\mathrm{L}$ with the left-handed quark (second field) as specified by **eps**[SU2L @ fund, #1, #2].

The remaining free indices of the fields are the generation indices of the quarks. The coupling itself carries these indices, which we spell out with the option CouplingIndices → ({ gen[#2], gen[#3]} &). This option should be passed a pure function, of 3 arguments in the present case, which returns a list of the indices of the coupling.

The last option passed to the **AddYukawa** is more of an ease of life option, than strictly necessary. The Yukawa couplings in `RGBeta` are always given in terms of left-handed spinors, but conventionally the SM defines the Yukawa couplings on the right-handed spinors, and the complex conjugate of the coupling appears with the left-handed fermions. To tell `RGBeta` that it is the conjugated coupling that should be placed with the left-handed spinors, we need to pass it the option Chirality → Right (Left by default).

The other two Yukawa couplings follow in the same manner as the up-type Yukawa. The main difference is that they involve the complex conjugated Higgs field. This is specified by passing **Bar @**H to **AddYukawa** as the scalar field of the coupling. With the conjugated Higgs field, the $\mathrm{SU}(2)_\mathrm{L}$ contraction is done with a Kronecker delta instead of the anti-symmetric invariant.

The quartic Higgs coupling is added to the model with the function call

```
AddQuartic[λ, {Bar @ H, H, Bar @ H, H},
    GroupInvariant → (del[SU2L @ fund, #1, #2] del[SU2L @ fund, #3, #4] /2 &)]
```

Again the function takes the coupling and the fields (four scalars this time) as arguments. In contrast to the Yukawa couplings, λ is a scalar coupling, and no coupling indices have to be passed to the function. Note also that the $1/2$ normalization of the Higgs self coupling, cf. Eq. (9), is put in the group invariant. The package never assumes any normalization factors, even when, as is the case with the SM, there is a symmetry factor associated to the coupling.

In $\overline{\text{MS}}$ the relevant couplings do not influence the running of the marginal couplings. However, if we also wish to explore the running of the Higgs mass parameter, we can add it to the model with

```
AddScalarMass[M2, {Bar @ H, H},
    GroupInvariant → (del[SU2L @ fund, #1, #2] &)]
```

This concludes the implementation of the SM.

3.3 Producing the β-functions

Once the model has been loaded into the kernel, extracting the β-function is as simple as knowing what coupling you are looking for. To obtain, for instance, the 2-loop β-function of the down-type Yukawa coupling, simply use the function **BetaFunction**[yd, 2]. This function returns the β-function as defined in Eqs. (2–5).

A typical use of **BetaFunction** (selected for minimality) will look like

```
In[1] := BetaFunction[ye, 1]
```

$$\text{Out[1]} := \frac{1}{16\pi^2}\left(-\frac{15}{4}g1^2 ye_{f1,f2} - \frac{9}{4}g2^2 ye_{f1,f2} + 3\text{Tr}[yd.yd^\dagger] ye_{f1,f2} + \text{Tr}[ye.ye^\dagger] ye_{f1,f2} + 3\text{Tr}[yu.yu^\dagger] ye_{f1,f2} + \frac{3}{2} ye.ye^\dagger.ye_{f1,f2}\right)$$

for the lepton Yukawa β-function at 1-loop order. The formating of the output (in the Mathematica StandardForm) has been set up to make the output more human readable. To see the underlying Mathematica expression, one can use the **FullForm** command.

The f1 and f2 indices are the flavor indices carried by the ye. These are thus open indices of the ye β-function too. f1 is used to denote an index of the first fermion of the coupling and f2 is similarly the index of the second fermion (in the list specified by the user). With how we implemented the ye coupling, these are the l and e generation indices, respectively. For further manipulation with the β-function, one will typically wish to remove such explicit indices too leave the matrix structure implicit. this can be done with the **Finalize** function.

Examples of further manipulation of the β-functions can be found in `Documentation/Tutorial.nb`. Here one can also find an example of how matrix couplings can be parametrized, and one can obtain e.g. the β-function of the charm Yukawa coupling. Finally, we should mention that the **BetaTerm** function can be used to single out the contribution to a β-function at a particular loop order, $\beta_g^{(\ell)}$.

Appendix

Documentation for the functions provided by `RGBeta`

AddFermion[*field*, **Options**]

---

adds a fermion field to the model, with specified charge and flavor.

field is a symbol or a string with the name of the field.

Options

GaugeRep → {} is a list of representations under the gauge groups.

FlavorIndices → {} is a list of the flavor indices of the field.

AddFermionMass[*coupling*, {*psi1*, *psi2*}, **Options**]

---

defines a fermion mass term in the model.

coupling names the mass of the mass terms.

{*psi1*, *psi2*} is the list consisting of the two fermions *psi1* and *psi2* of the mass term.

Options

GroupInvariant → (1 **&**) is a pure function of 3 arguments defining the group invariants of the coupling.

CouplingIndices → (**Null &**) is a pure function of 3 arguments which specify the tensor indices of the coupling.

Chirality → Left sets whether the coupling appears with left-handed or right-handed fields in the Lagrangian.

AddGaugeGroup[*coupling*, *groupName*, *lieGroup*[*n*], **Options**]

---

adds a gauge group to the current model.

coupling specifies the coupling of the gauge group.

groupName specifies the reference name associated to the group and its representations.

lieGroup[*n*] specifies what Lie Group is gauged. The options are U1=U1[1], U1[n], SU[*n*], Sp[*n*], and SO[*n*], with *n* either an integer or a symbol.

Options

CouplingMatrix → **Automatic** determines the naming of the coupling matrix if the Lie group is $\mathrm{U}(1)^n$. Any symmetric $n \times n$ matrix can be supplied instead of the automatic naming.

AddQuartic[*coupling*, {*phi1*, *phi2*, *phi3*, *phi4*}, **Options**]

---

defines a quartic coupling in the model.

coupling specifies the coupling constant of the quartic interaction.

{*phi1*, *phi2*, *phi3*, *phi4*} is a list consisting of the four scalar fields involved in the interaction. They can be be conjugated with **Bar**.

Options

GroupInvariant → (1 **&**) is a pure function of 4 arguments defining the group invariants of the coupling.

CouplingIndices → (**Null &**) is a pure function of 4 arguments which specify the tensor indices of the coupling.

SelfConjugate → **True** sets if the interaction is real, or whether the Hermitian conjugate appears in the Lagrangian as well.

AddScalar[*field*, **Options**]

---

adds a scalar field to the model with specified charge and flavor.

field is a symbol or a string with the name of the field.

Options

GaugeRep → {} is a list of representations under the gauge groups.

FlavorIndices → {} is a list of the flavor indices of the field.

SelfConjugate → **False** determines if the fields is complex or real.

AddScalarMass[*coupling*, {*phi1*, *phi2*}, **Options**]

---

defines a scalar mass term in the model.

coupling denotes the mass parameter of the mass term *assumed to have mass-dimension two*.

{*phi1*, *phi2*} is the list consisting of two scalar fields. They can be be conjugated with **Bar**

Options

GroupInvariant → (1 **&**) is a pure function of 4 arguments defining the group invariants of the coupling.

CouplingIndices → (**Null &**) is a pure function of 4 arguments which specify the tensor indices of the coupling.

SelfConjugate → **True** sets if the interaction is real, or whether the Hermitian conjugate appears in the Lagrangian as well.

AddTrilinear[*coupling*, {*phi1*, *phi2*, *phi3*}, **Options**]

---

defines a trilinear coupling in the model.

coupling specifies the coupling constant of the Yukawa interaction.

{*phi1*, *phi2*, *phi3*} is the list consisting of three scalar fields. They can be be conjugated with **Bar**

Options

GroupInvariant → (1 **&**) is a pure function of 4 arguments defining the group invariants of the coupling.

CouplingIndices → (**Null &**) is a pure function of 4 arguments which specify the tensor indices of the coupling.

SelfConjugate → **True** sets if the interaction is real, or whether the Hermitian conjugate appears in the Lagrangian as well.

AddYukawa[*coupling*, {*phi*, *psi1*, *psi2*}, **Options**]

---

defines a Yukawa coupling in the model.

coupling specifies the coupling constant of the Yukawa interaction.

{*phi*, *psi1*, *psi2*} is the list consisting of the scalar *phi* and two fermions *psi1* and *psi2* of the interaction. The scalar can be conjugated with **Bar**.

Options

GroupInvariant → (1 **&**) is a pure function of 3 arguments defining the group invariants of the coupling.

CouplingIndices → (**Null &**) is a pure function of 3 arguments which specify the tensor indices of the coupling.

Chirality → Left sets whether the coupling appears with left-handed or right-handed fields in the Lagrangian.

BetaFunction[*coupling*, *loop*, **Options**]

gives the full β-function of a coupling up to the given loop order.

coupling is a symbol corresponding to the coupling of the relevant β-function.

loop is an integer specifying to what loop order.

Options

FourDimensions → **True** determines if the returned β-function is that of a strictly four-dimensional theory, or whether it is the one in $(4-\epsilon)$ dimensions.

RescaledCouplings → **False** determines whether the couplings should all be rescaled with $g \to 4\pi g$, $y \to 4\pi y$, and $\lambda \to (4\pi)^2\lambda$.

BetaTerm[*coupling*, *loop*]

gives the term $\beta_g^{(\ell)}$ in the β-function.

coupling is a symbol corresponding to the coupling of the relevant β-function.

loop is an integer specifying the loop order.

CheckProjection[*coupling*]

returns the value of the internal projection operator applied to the general coupling.

coupling specifies the coupling to be checked.

DefineLieGroup[*groupName*, *lieGroup*[*n*]]

sets up all group constants associated to the a Lie group of the specified kind.

groupName specifies the name of the added gauge group.

lieGroup[*n*] specifies what Lie Group is gauged. It can either be a SU[*n*], Sp[*n*], and SO[*n*], with *n* either as an integer or symbol.

Finalize[*expr*, **Options**]

gives an alternate form of the β-function.

expr an expression on the form such as given by **BetaTerm**.

Options

Parametrizations → {} a set of substitution rules, to replace the coupling symbols with e.g. coupling matrices.

BarToConjugate → **False** replaces the `RGBeta` head `bar` with the standard Mathematica **Conjugate** giving an expression more suitable for further numerical analysis.

QuarticBetaFunctions[*loop*, **Options**]

---

gives the full β-functions of all quartic couplings, using fully diagonalized projectors.

loop is an integer specifying to what loop order.

Options

FourDimensions → **True** determines if the returned β-function is that of a strictly four-dimensional theory, or whether it is the one in $(4-\epsilon)$ dimensions.

RescaledCouplings → **False** determines whether the couplings should all be rescaled with $g \to 4\pi g$, $y \to 4\pi y$, and $\lambda \to (4\pi)^2\lambda$.

ResetModel[ ]

---

clears all current model definitions, allowing for defining another model, without having to quit the kernel and reloading `RGBeta`.

SetReal[*symbol*,...]

---

instructs `RGBeta` to treat one or more symbols (typically couplings) as being real by setting **Bar @** symb = symb.

symbol the symbol that will be defined to be real.
