RGESolver

Generated by Doxygen 1.8.20

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 RGESolver Class Reference

A class that performs renormalization group evolution in the context of the SMEFT.

`#include <RGESolver.h>`

Public Member Functions

- RGESolver ()
  The default constructor.
- ~RGESolver ()
  The default destructor.

Parameters related to the numeric integration.

- double epsrel ()
  Getter for the relative error used in the numerical integration.
- double epsabs ()
  Getter for the absolute error used in the numerical integration.
- double step ()
  Getter for the step used in the numerical integration.
- void Set_epsrel (double epsrel)
  Setter for the relative error used in the numerical integration.
- void Set_epsabs (double epsabs)
  Setter for the absolute error used in the numerical integration.
- void Set_step (double step)
  Setter for the step used in the numerical integration.

Evolution

- void Evolve (std::string method, double muI, double muF)
  Performs the RGE evolution.
- void GenerateSMInitialConditions (double mu, std::string basis, std::string method, bool inputCKM=true)
  Generates the initial conditions for Standard Model's parameters (gauge couplings, Yukawa coupling, quartic coupling and Higgs' boson mass).
- void EvolveSMOnly (std::string method, double muI, double muF)

Same as Evolve, but only for the SM parameters. The user should use this method instead of Evolve when interested in pure SM running.

Input/output

Documentation for the input/output handling.

All the SMEFT coefficients are set using the
SetCoefficient methods and accessed with the GetCoefficient methods. There exist three different signatures for each method, depending on the number of flavour indices of the parameter (0,2,4).
These two routines must be used also for the SM parameters $g1, g2, g3, \lambda, m_h^2, \Re(\mathcal{Y}_u), \Im(\mathcal{Y}_u), \Re(\mathcal{Y}_d), \Im(\mathcal{Y}_d), \Re(\mathcal{Y}_e), \Im(\mathcal{Y}_e)$ *(we follow* https://arxiv.org/abs/1308.2627 *for what concerns the conventions in the Higgs' sector).*
If the user is interested in using the GenerateSMInitialConditions method, the input for the CKM matrix parameters and the fermion masses must be given with the methods SetCKMAngle(std::string name, double val), SetCKMPhase(double val) SetFermionMass(std::string name, double val).
A complete list of the keys that must be used to correctly invoke setter/getter methods are given in tables 2.1, 2.2, 2.3 and 2.4

- void ComputeCKMAndFermionMasses ()

  Compute CKM matrix and the mass of the fermions.
- void SetFermionMass (std::string name, double val)

  Setter function for the mass of the fermions. Assignation is allowed only if the inserted value is not negative.
- double GetFermionMass (std::string name)

  Getter function for the mass of the fermions.
- void SetCKMAngle (std::string name, double val)

  Setter function for the CKM matrix angles $\theta_{12}, \theta_{13}, \theta_{23}$*. The assignation is completed only if the inserted angle is* $\in [0, \frac{\pi}{2}]$.
- double GetCKMAngle (std::string name)

  Getter function for the CKM matrix angles $\theta_{12}, \theta_{13}, \theta_{23}$.
- void SetCKMPhase (double val)

  Setter function for the CKM matrix phase δ*. The assignation is completed only if* $\delta \in (-\pi, \pi]$.
- double GetCKMPhase ()

  Getter function for the CKM matrix phase δ.
- void SetSMInputScale (double mu)

  Setter method for the scale at which the method GenerateSMInitialConditions takes the input values for SM parameters.
- double GetSMInputScale ()

  Getter method for the scale at which the method GenerateSMInitialConditions takes the input values for SM parameters.
- void SetCoefficient (std::string name, double val)

  Setter function for scalar/0F parameters (no flavour indices).
- void SetCoefficient (std::string name, double val, int i, int j)

  Setter function for 2F parameters (2 flavour indices).
- void SetCoefficient (std::string name, double val, int i, int j, int k, int l)

  Setter function for 4F parameters (4 flavour indices).
- double GetCoefficient (std::string name)

  Getter function for scalar/0F parameters (no flavour indices).
- double GetCoefficient (std::string name, int i, int j)

  Getter function for 2F parameters (2 flavour indices).
- double GetCoefficient (std::string name, int i, int j, int k, int l)

  Getter function for 4F parameters (4 flavour indices).
- void Reset ()

  Resets all the SMEFT coefficients to 0 and the SM parameters to their default value.
- void SaveOutputFile (std::string filename, std::string format)

  Saves the current values of parameters in a file.

2.1.1 Detailed Description

A class that performs renormalization group evolution in the context of the SMEFT.

The class solves the Renormalization Group Equations (RGEs) both numerically and in the leading-log approximations. Only operators up to dimension six that preserve lepton and baryon numbers are considered. The operator basis is the Warsaw basis, defined in https://arxiv.org/abs/1008.4884.
The user must set separately real and imaginary part of each complex parameter.
In tables 2.1, 2.2, 2.3 and 2.4 are listed all the parameters, together with their name (that must be used to correctly invoke getter and setter functions).
The numerical integration is performed with an adaptive step-size routine (the Explicit embedded Runge-Kutta-↩ Fehlberg method), using the tools in the GNU Scientific Library.
See https://www.gnu.org/software/gsl/doc/html/ode-initval.html for all the details.
The accuracy level of the numerical integration can be tuned selecting the parameters $\epsilon_{rel}, \epsilon_{abs}$ and the integration step using the dedicated getter functions.

Author

HEPfit Collaboration

Copyright

GNU General Public License

Parameter	Name	Parameter	Name
g_1	g1	θ_{12}	CKM_theta12
g_2	g2	θ_{13}	CKM_theta13
g_3	g3	θ_{23}	CKM_theta23
λ	lambda	m_u [GeV]	mu
m_h^2 [GeV^2]	mh2	m_c [GeV]	mc
$\Re(\mathcal{Y}_u)$	YuR	m_t [GeV]	mt
$\Im(\mathcal{Y}_u)$	YuI	m_d [GeV]	md
$\Re(\mathcal{Y}_d)$	YdR	m_s [GeV]	ms
$\Im(\mathcal{Y}_d)$	YdI	m_b [GeV]	mb
$\Re(\mathcal{Y}_e)$	YeR	m_e [GeV]	mel
$\Im(\mathcal{Y}_e)$	YeI	m_μ [GeV]	mmu
		m_τ [GeV]	mtau

Table 2.1 Standard Model parameters. The parameters in the left column must be set (and accessed) with SetCoefficient (and GetCoefficient) methods. The ones in the right column must be set and accessed using other dedicated methods (see the specific documentation for input/output)

Classes 1-3		Class 4	
Coefficient	Name	Coefficient	Name
C_G	CG	C_{HG}	CHG
$C_{\tilde{G}}$	CGtilde	$C_{H\tilde{G}}$	CHGtilde
C_W	CW	C_{HW}	CHW
$C_{\tilde{W}}$	CWtilde	$C_{H\tilde{W}}$	CHWtilde
C_H	CH	C_{HB}	CHB
$C_{H\Box}$	CHbox	$C_{H\tilde{B}}$	CHBtilde
C_{HD}	CHD	C_{HWB}	CHWB
		$C_{H\tilde{W}B}$	CHWtildeB

Table 2.2 Scalar (and real) SMEFT operators. They must be set and accessed using SetCoefficient and GetCoefficient.

Class 5			Class 6			Class 7		
Coefficient	Name	Symmetry	Coefficient	Name	Symmetry	Coefficient	Name	Symmetry
$\Re(C_{eH})$	CeHR	WC1	$\Re(C_{eW})$	CeWR	WC1	$\Re(C_{Hl1})$	CHl1R	WC2R
$\Im(C_{eH})$	CeHI	WC1	$\Im(C_{eW})$	CeWI	WC1	$\Im(C_{Hl1})$	CHl1I	WC2I
$\Re(C_{uH})$	CuHR	WC1	$\Re(C_{eB})$	CeBR	WC1	$\Re(C_{Hl3})$	CHl3R	WC2R
$\Im(C_{uH})$	CuHI	WC1	$\Im(C_{eB})$	CeBI	WC1	$\Im(C_{Hl3})$	CHl3I	WC2I
$\Re(C_{dH})$	CdHR	WC1	$\Re(C_{uG})$	CuGR	WC1	$\Re(C_{He})$	CHeR	WC2R
$\Im(C_{dH})$	CdHI	WC1	$\Im(C_{uG})$	CuGI	WC1	$\Im(C_{He})$	CHeI	WC2I
			$\Re(C_{uW})$	CuWR	WC1	$\Re(C_{Hq1})$	CHq1R	WC2R
			$\Im(C_{uW})$	CuWI	WC1	$\Im(C_{Hq1})$	CHq1I	WC2I
			$\Re(C_{uB})$	CuBR	WC1	$\Re(C_{Hq3})$	CHq3R	WC2R
			$\Im(C_{uB})$	CuBI	WC1	$\Im(C_{Hq3})$	CHq3I	WC2I
			$\Re(C_{dG})$	CdGR	WC1	$\Re(C_{Hu})$	CHuR	WC2R
			$\Im(C_{dG})$	CdGI	WC1	$\Im(C_{Hu})$	CHuI	WC2I
			$\Re(C_{dW})$	CdWR	WC1	$\Re(C_{Hd})$	CHdR	WC2R
			$\Im(C_{dW})$	CdWI	WC1	$\Im(C_{Hd})$	CHdI	WC2I
			$\Re(C_{dB})$	CdBR	WC1	$\Re(C_{Hud})$	CHudR	WC1
			$\Im(C_{dB})$	CdBI	WC1	$\Im(C_{Hud})$	CHudI	WC1

Table 2.3 2F SMEFT operators. They must be set and accessed using SetCoefficient and GetCoefficient.

Class 8 $(\bar{L}L)(\bar{L}L)$

Coefficient	Name	Symmetry
$\Re(C_{ll})$	`CllR`	WC6R
$\Im(C_{ll})$	`CllI`	WC6I
$\Re(C_{qq1})$	`Cqq1R`	WC6R
$\Im(C_{qq1})$	`Cqq1I`	WC6I
$\Re(C_{qq3})$	`Cqq3R`	WC6R
$\Im(C_{qq3})$	`Cqq3I`	WC6I
$\Re(C_{lq1})$	`Clq1R`	WC7R
$\Im(C_{lq1})$	`Clq1I`	WC7I
$\Re(C_{lq3})$	`Clq3R`	WC7R
$\Im(C_{lq3})$	`Clq3I`	WC7I

Class 8 $(\bar{L}R)(\bar{L}R)$

Coefficient	Name	Symmetry
$\Re(C_{quqd1})$	`Cquqd1R`	WC5
$\Im(C_{quqd1})$	`Cquqd1I`	WC5
$\Re(C_{quqd8})$	`Cquqd8R`	WC5
$\Im(C_{quqd8})$	`Cquqs8I`	WC5
$\Re(C_{lequ1})$	`Clequ1R`	WC5
$\Im(C_{lequ1})$	`Clequ1I`	WC5
$\Re(C_{lequ3})$	`Clequ3R`	WC5
$\Im(C_{lequ3})$	`Clequ3I`	WC5

Class 8 $(\bar{R}R)(\bar{R}R)$

Coefficient	Name	Symmetry
$\Re(C_{ee})$	`CeeR`	WC8R
$\Im(C_{ee})$	`CeeI`	WC8I
$\Re(C_{uu})$	`CuuR`	WC6R
$\Im(C_{uu})$	`CuuI`	WC6I
$\Re(C_{dd})$	`CddR`	WC6R
$\Im(C_{dd})$	`CddI`	WC6I
$\Re(C_{eu})$	`CeuR`	WC7R
$\Im(C_{eu})$	`CeuI`	WC7I
$\Re(C_{ed})$	`CedR`	WC7R
$\Im(C_{ed})$	`CedI`	WC7I
$\Re(C_{ud1})$	`Cud1R`	WC7R
$\Im(C_{ud1})$	`Cud1I`	WC7I
$\Re(C_{ud8})$	`Cud8R`	WC7R
$\Im(C_{ud8})$	`Cud8I`	WC7I

Class 8 $(\bar{L}R)(\bar{R}L)$

Coefficient	Name	Symmetry
$\Re(C_{ledq})$	`CledqR`	WC5
$\Im(C_{ledq})$	`CledqI`	WC5

Class 8 $(\bar{L}L)(\bar{R}R)$

Coefficient	Name	Symmetry
$\Re(C_{le})$	`CleR`	WC7R
$\Im(C_{le})$	`CleI`	WC7I
$\Re(C_{lu})$	`CluR`	WC7R
$\Im(C_{lu})$	`CluI`	WC7I
$\Re(C_{ld})$	`CldR`	WC7R
$\Im(C_{ld})$	`CldI`	WC7I
$\Re(C_{qe})$	`CqeR`	WC7R
$\Im(C_{qe})$	`CqeI`	WC7I
$\Re(C_{qu1})$	`Cqu1R`	WC7R
$\Im(C_{qu1})$	`Cqu1I`	WC7I
$\Re(C_{qu8})$	`Cqu8R`	WC7R
$\Im(C_{qu8})$	`Cqu8I`	WC7I
$\Re(C_{qd1})$	`Cqd1R`	WC7R
$\Im(C_{qd1})$	`Cqd1I`	WC7I
$\Re(C_{qd8})$	`Cqd8R`	WC7R
$\Im(C_{qd8})$	`Cqd8I`	WC7I

Table 2.4 4F SMEFT Operators. They must be set and accessed using SetCoefficient and GetCoefficient.

2.1.2 Member Function Documentation

2.1.2.1 ComputeCKMAndFermionMasses()

```
void RGESolver::ComputeCKMAndFermionMasses ( )
```

Compute CKM matrix and the mass of the fermions.

The methods Evolve and EvolveSMOnly do not updates the value of CKM parameters and fermion masses after the evolution. This process require the diagonalization of the Yukawa matrices and may slow the evolution.
If the user is interested in these parameters (accessible with GetCKMAngle, GetCKMPhase, GetFermionMass) must invoke this method after the evolution.

2.1.2.2 Evolve()

```
void RGESolver::Evolve (
            std::string method,
            double muI,
            double muF )
```

Performs the RGE evolution.

RGEs are solved with the chosen method from `muI` to `muF`. Currently, the available methods are "Numeric" and "Leading-Log".
The evolutor takes as initial values the current values of the parameters, set with the `SetCoefficient(...)` function. After completing the evolution the values of the parameters are updated and are accessible with the `GetCoefficient(...)` function.

Parameters

method	resolution method
muI	initial energy scale
muF	final energy scale

2.1.2.3 EvolveSMOnly()

```
void RGESolver::EvolveSMOnly (
            std::string method,
            double muI,
            double muF )
```

Same as Evolve, but only for the SM parameters. The user should use this method instead of Evolve when interested in pure SM running.

Parameters

method	
muI	
muF	

2.1.2.4 GenerateSMInitialConditions()

```
void RGESolver::GenerateSMInitialConditions (
            double mu,
            std::string basis,
            std::string method,
            bool inputCKM = true )
```

Generates the initial conditions for Standard Model's parameters (gauge couplings, Yukawa coupling, quartic coupling and Higgs' boson mass).

After evolving the SM parameters up to the scale mu, CKM parameters and fermion masses are updated with the new values. If the flag CKMinput is set to `true` (default), the input for the Yukawa matrices will be generated from the current value of the CKM matrix and the masses of the fermions. If set to false, the current values of the Yukawa matrices will be used to generate the SM initial conditions at the chosen scale.

Parameters

mu	Scale (in GeV) at which the initial conditions are generated. If `mu` is different from the scale at which the input is given (SMInputScale), `RGESolver` will use the pure SM RGEs (at one-loop level) to run the parameters to the scale `mu`.
basis	Flavour basis (`"UP"or "DOWN"`)
method	`Method used by RGESolver to run the SM parameters to the scale mu ("Numeric" or "Leading-Log")`
inputCKM	`If set to true (default), the input for the Yukawa matrices will be generated from the current value of the CKM matrix and the masses of the fermions.`

2.1.2.5 GetCKMAngle()

```
double RGESolver::GetCKMAngle (
            std::string name )
```

Getter function for the CKM matrix angles $\theta_{12}, \theta_{13}, \theta_{23}$.

Returns

The selected CKM angle.

2.1.2.6 GetCKMPhase()

```
double RGESolver::GetCKMPhase ( )
```

Getter function for the CKM matrix phase δ.

Returns

δ.

2.1.2.7 GetCoefficient() [1/3]

```
double RGESolver::GetCoefficient (
            std::string name )
```

Getter function for scalar/0F parameters (no flavour indices).

If the parameter name does not match with any of the parameters, an error message is printed and the value 0 is returned.

Parameters

name	name of the parameter

Returns

the requested parameter (if it exists), otherwise returns 0.

2.1.2.8 GetCoefficient() [2/3]

```
double RGESolver::GetCoefficient (
            std::string name,
            int i,
            int j )
```

Getter function for 2F parameters (2 flavour indices).

If the parameter name does not match with any of the parameters or if at least one of the inserted indices is outside the [0:2] range,
an error message is printed and the value 0 is returned.

Parameters

name	name of the parameter
i	first flavour index
j	second flavour index

Returns

the requested parameter (if it exists), otherwise returns 0.

2.1.2.9 GetCoefficient() [3/3]

```
double RGESolver::GetCoefficient (
            std::string name,
            int i,
            int j,
            int k,
            int l )
```

Getter function for 4F parameters (4 flavour indices).

If the parameter name does not match with any of the parameters or if at least one of the inserted indices is outside the [0:2] range,
an error message is printed and the value 0 is returned.

Parameters

name	name of the parameter
i	first flavour index
j	second flavour index
k	third flavour index
l	fourth flavour index

Returns

the requested parameter (if it exists), otherwise returns 0.

2.1.2.10 SaveOutputFile()

```
void RGESolver::SaveOutputFile (
            std::string filename,
            std::string format )
```

Saves the current values of parameters in a file.

Currently, only "SLHA" format is implemented

Parameters

filename	Name of the output file
format	Format of the output file

2.1.2.11 SetCKMAngle()

```
void RGESolver::SetCKMAngle (
            std::string name,
            double val )
```

Setter function for the CKM matrix angles $\theta_{12}, \theta_{13}, \theta_{23}$. The assignation is completed only if the inserted angle is $\in [0, \frac{\pi}{2}]$.

Parameters

val	

2.1.2.12 SetCKMPhase()

```
void RGESolver::SetCKMPhase (
            double val )
```

Setter function for the CKM matrix phase δ. The assignation is completed only if $\delta \in (-\pi, \pi]$.

Parameters

val	

2.1.2.13 SetCoefficient() [1/3]

```
void RGESolver::SetCoefficient (
            std::string name,
            double val )
```

Setter function for scalar/0F parameters (no flavour indices).

If the parameter name does not match with any of the parameters, an error message is printed and no assignation is performed.

Parameters

name	name of the parameter
val	its value

2.1.2.14 SetCoefficient() [2/3]

```
void RGESolver::SetCoefficient (
            std::string name,
            double val,
            int i,
            int j )
```

Setter function for 2F parameters (2 flavour indices).

If the parameter name does not match with any of the parameters or if at least one of the inserted indices is outside the [0:2] range,

Parameters

name	name of the parameter
val	its value
i	first flavour index
j	second flavour index

2.1.2.15 SetCoefficient() [3/3]

```
void RGESolver::SetCoefficient (
            std::string name,
            double val,
            int i,
            int j,
            int k,
            int l )
```

Setter function for 4F parameters (4 flavour indices).

If the parameter name does not match with any of the parameters or if at least one of the inserted indices is outside the [0:2] range, an error message is printed and no assignation is performed.

Parameters

name	name of the parameter
val	its value
i	first flavour index
j	second flavour index
k	third flavour index
l	fourth flavour index

2.1.2.16 SetFermionMass()

```
void RGESolver::SetFermionMass (
            std::string name,
            double val )
```

Setter function for the mass of the fermions. Assignation is allowed only if the inserted value is not negative.

Parameters

val	

2.1.2.17 SetSMInputScale()

```
void RGESolver::SetSMInputScale (
            double mu )  [inline]
```

Setter method for the scale at which the method GenerateSMInitialConditions takes the input values for SM parameters.

Parameters

mu	

The documentation for this class was generated from the following files:

- RGESolver.h
- RGESolver.cc
- StaticMembers.cc
- BetaFunction.cc
- SettersAndGetters.cc

Index