RGESolver

Generated by Doxygen 1.8.20

Chapter 1

<tt>RGESolver</tt>

A C++ library to perform renormalization group evolution of SMEFT coefficients, both numerically and with the leading-log approximation. The general flavour case at dimension-six level is considered. Operators that violate lepton and/or baryon number conservation are not considered. The documentation for this library can be found here

RGESolver is a free software under the copyright of the GNU General Public License.

1.1 Dependencies

- BOOST : BOOST is a C++ library which can be obtained from the BOOST website or from Linux package managers or Mac ports. RGESolver only requires the BOOST headers, not the full libraries, so a header-only installation is sufficient.
- GSL : The GNU Scientific Library (GSL) is a C library for numerical computations. It can be found on the GSL website. Most Linux package managers will have a stable version as will any ports for Mac.
- C++11 : A compiler that supports at least C++11 standard is required.

1.2 Installation

The installation of RGESolver requires the availability of CMake in the system (version 3.1 or greater). A description of CMake and the instructions for its installation can be found in the CMakewebsite.

The installation can be performed writhing the following lines in a terminal session (in the RGESolver directory):

```
mkdir build && cd $_
cmake .. <options>
cmake --build .
cmake --install .
```

Note that depending on the setting of installation prefix (see below) the user might need root privileges to be able to install RGESolver.

1.2.1 Command line options for the installation

- -DLOCAL_INSTALL:BOOL=<ON or OFF> : to install RGESolver in the directory build/install (default: OFF).
- -DCMAKE_INSTALL_PREFIX:PATH=<RGESolver installation directory> : the directory in which RGESolver will be installed (default: /usr/local). This variable cannot be modified when -DLOCAL INSTALL ALL=ON is set.
- -DDEBUG_MODE:BOOL=<ON or OFF> : to enable the debug mode (default: OFF).
- -DBOOST_INCLUDE_DIR:PATH=<include path>/boost/ : CMakechecks for BOOST headers availability in the system and fails if they are not installed. Thus, if BOOST is not installed in the predefined search path, the user can specify where it is with this option. The path must end with the boost/directory which contains the headers.
- -DGSL_CONFIG_DIR:PATH=<gsl-config directory> : RGESolver uses gsl-config to get the GSL parameters. If this is not in the predefined search path, the user can specify it with this option.

1.3 Usage

The rgesolver-config script is available in the <CMAKE_INSTALL_PREFIX>/bin directory (default: /usr/local), which can be invoked with the following options:

- --cflags: to obtain the include path needed for compilation against the RGESolver.
- --libs: to obtain the flags needed for linking against the RGESolver.

If the path <CMAKE_INSTALL_PREFIX>/bin is not in the predefined search path, the compilation will (most likely) fail. if the user wants to use the compilation command above, it is suggested to add <CMAKE_INSTA↩ LL_PREFIX>/bin to the $PATH variable. Alternatively, the script can be invoked from a terminal session in <CMAKE_INSTALL_PREFIX>/bin to visualize the paths to the library and to the headers.

After the installation, the example program ExampleProgram.cpp (available in the Example Program directory) can be compiled with the command

```
g++ -o app ExampleProgram.cpp `rgesolver-config --cflags` `rgesolver-config --libs`
```

1.4 Uninstall

The user can uninstall the library typing in a terminal session in the build directory:

```
cmake --build . --target uninstall
```

Also in this case, depending on the setting of installation prefix, the user might need root privileges to be able to uninstall RGESolver.

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 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) at the scale `mu`, using one-loop pure SM beta functions.
- 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. Using this function is the same of using Evolve with all the SMEFT coefficients set to 0, but it is faster since it does compute only the evolution for the SM parameters.

- void ComputeCKMAndFermionMasses ()

  Compute CKM matrix and the mass of the fermions.

Input/output

- void SetFermionMass (std::string name, double val)

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

  Getter function for the mass of the fermions (in GeV).
- 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 (in GeV). 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.

3.1.1 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 approximation. 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 class splits real and imaginary part of each complex parameter.

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 setter functions.

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 $g_1, g_2, g_3, \lambda, m_h^2, \mathrm{Re}(\mathcal{Y}_u), \mathrm{Im}(\mathcal{Y}_u), \mathrm{Re}(\mathcal{Y}_d), \mathrm{Im}(\mathcal{Y}_d), \mathrm{Re}(\mathcal{Y}_e), \mathrm{Im}(\mathcal{Y}_e)$ (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, SetCKMPhase SetFermionMass.
A complete list of the keys that must be used to correctly invoke setter/getter methods are given in tables 3.1, 3.2, 3.3 and 3.4.

Author

S. Di Noi, L. Silvestrini.

Copyright

Table 3.1 Standard Model parameters.

Parameter	Name
g_1	g1
g_2	g2
g_3	g3
λ	lambda
m_h^2 [GeV^2]	mh2
$\mathrm{Re}(\mathcal{Y}_u)$	YuR
$\mathrm{Im}(\mathcal{Y}_u)$	YuI
$\mathrm{Re}(\mathcal{Y}_d)$	YdR
$\mathrm{Im}(\mathcal{Y}_d)$	YdI
$\mathrm{Re}(\mathcal{Y}_e)$	YeR
$\mathrm{Im}(\mathcal{Y}_e)$	YeI

Parameter	Name
θ_{12}	CKM_theta12
θ_{13}	CKM_theta13
θ_{23}	CKM_theta23
δ	CKM_delta
m_u [GeV]	mu
m_c [GeV]	mc
m_t [GeV]	mt
m_d [GeV]	md
m_s [GeV]	ms
m_b [GeV]	mb
m_e [GeV]	mel
m_μ [GeV]	mmu
m_τ [GeV]	mtau

Table 3.2 Scalar (and real) SMEFT operators.

Classes 1-3 Coefficient	Name
C_G	CG
$C_{\tilde{G}}$	CGtilde
C_W	CW
$C_{\tilde{W}}$	CWtilde
C_H	CH
$C_{H\Box}$	CHbox
C_{HD}	CHD

Class 4 Coefficient	Name
C_{HG}	CHG
$C_{H\tilde{G}}$	CHGtilde
C_{HW}	CHW
$C_{H\tilde{W}}$	CHWtilde
C_{HB}	CHB
$C_{H\tilde{B}}$	CHBtilde
C_{HWB}	CHWB
$C_{H\tilde{W}B}$	CHWtildeB

Table 3.3 2F SMEFT operators.

Class 5 Coefficient	Name	Symmetry
$\mathrm{Re}(C_{eH})$	CeHR	WC1
$\mathrm{Im}(C_{eH})$	CeHI	WC1
$\mathrm{Re}(C_{uH})$	CuHR	WC1
$\mathrm{Im}(C_{uH})$	CuHI	WC1
$\mathrm{Re}(C_{dH})$	CdHR	WC1
$\mathrm{Im}(C_{dH})$	CdHI	WC1

Class 6 Coefficient	Name	Symmetry
$\mathrm{Re}(C_{eW})$	CeWR	WC1
$\mathrm{Im}(C_{eW})$	CeWI	WC1
$\mathrm{Re}(C_{eB})$	CeBR	WC1
$\mathrm{Im}(C_{eB})$	CeBI	WC1
$\mathrm{Re}(C_{uG})$	CuGR	WC1
$\mathrm{Im}(C_{uG})$	CuGI	WC1
$\mathrm{Re}(C_{uW})$	CuWR	WC1
$\mathrm{Im}(C_{uW})$	CuWI	WC1
$\mathrm{Re}(C_{uB})$	CuBR	WC1
$\mathrm{Im}(C_{uB})$	CuBI	WC1
$\mathrm{Re}(C_{dG})$	CdGR	WC1
$\mathrm{Im}(C_{dG})$	CdGI	WC1
$\mathrm{Re}(C_{dW})$	CdWR	WC1
$\mathrm{Im}(C_{dW})$	CdWI	WC1
$\mathrm{Re}(C_{dB})$	CdBR	WC1
$\mathrm{Im}(C_{dB})$	CdBI	WC1

Class 7 Coefficient	Name	Symmetry
$\mathrm{Re}(C_{Hl1})$	CHl1R	WC2R
$\mathrm{Im}(C_{Hl1})$	CHl1I	WC2I
$\mathrm{Re}(C_{Hl3})$	CHl3R	WC2R
$\mathrm{Im}(C_{Hl3})$	CHl3I	WC2I
$\mathrm{Re}(C_{He})$	CHeR	WC2R
$\mathrm{Im}(C_{He})$	CHeI	WC2I
$\mathrm{Re}(C_{Hq1})$	CHq1R	WC2R
$\mathrm{Im}(C_{Hq1})$	CHq1I	WC2I
$\mathrm{Re}(C_{Hq3})$	CHq3R	WC2R
$\mathrm{Im}(C_{Hq3})$	CHq3I	WC2I
$\mathrm{Re}(C_{Hu})$	CHuR	WC2R
$\mathrm{Im}(C_{Hu})$	CHuI	WC2I
$\mathrm{Re}(C_{Hd})$	CHdR	WC2R
$\mathrm{Im}(C_{Hd})$	CHdI	WC2I
$\mathrm{Re}(C_{Hud})$	CHudR	WC1
$\mathrm{Im}(C_{Hud})$	CHudI	WC1

Table 3.4 4F SMEFT Operators.

Class 8 $(\bar{L}L)(\bar{L}L)$		
Coefficient	**Name**	**Symmetry**
$\mathrm{Re}(C_{ll})$	`CllR`	WC6R
$\mathrm{Im}(C_{ll})$	`CllI`	WC6I
$\mathrm{Re}(C_{qq1})$	`Cqq1R`	WC6R
$\mathrm{Im}(C_{qq1})$	`Cqq1I`	WC6I
$\mathrm{Re}(C_{qq3})$	`Cqq3R`	WC6R
$\mathrm{Im}(C_{qq3})$	`Cqq3I`	WC6I
$\mathrm{Re}(C_{lq1})$	`Clq1R`	WC7R
$\mathrm{Im}(C_{lq1})$	`Clq1I`	WC7I
$\mathrm{Re}(C_{lq3})$	`Clq3R`	WC7R
$\mathrm{Im}(C_{lq3})$	`Clq3I`	WC7I

Class 8 $(\bar{L}R)(\bar{L}R)$		
Coefficient	**Name**	**Symmetry**
$\mathrm{Re}(C_{quqd1})$	`Cquqd1R`	WC5
$\mathrm{Im}(C_{quqd1})$	`Cquqd1I`	WC5
$\mathrm{Re}(C_{quqd8})$	`Cquqd8R`	WC5
$\mathrm{Im}(C_{quqd8})$	`Cquqs8I`	WC5
$\mathrm{Re}(C_{lequ1})$	`Clequ1R`	WC5
$\mathrm{Im}(C_{lequ1})$	`Clequ1I`	WC5
$\mathrm{Re}(C_{lequ3})$	`Clequ3R`	WC5
$\mathrm{Im}(C_{lequ3})$	`Clequ3I`	WC5

Class 8 $(\bar{R}R)(\bar{R}R)$		
Coefficient	**Name**	**Symmetry**
$\mathrm{Re}(C_{ee})$	`CeeR`	WC8R
$\mathrm{Im}(C_{ee})$	`CeeI`	WC8I
$\mathrm{Re}(C_{uu})$	`CuuR`	WC6R
$\mathrm{Im}(C_{uu})$	`CuuI`	WC6I
$\mathrm{Re}(C_{dd})$	`CddR`	WC6R
$\mathrm{Im}(C_{dd})$	`CddI`	WC6I
$\mathrm{Re}(C_{eu})$	`CeuR`	WC7R
$\mathrm{Im}(C_{eu})$	`CeuI`	WC7I
$\mathrm{Re}(C_{ed})$	`CedR`	WC7R
$\mathrm{Im}(C_{ed})$	`CedI`	WC7I
$\mathrm{Re}(C_{ud1})$	`Cud1R`	WC7R
$\mathrm{Im}(C_{ud1})$	`Cud1I`	WC7I
$\mathrm{Re}(C_{ud8})$	`Cud8R`	WC7R
$\mathrm{Im}(C_{ud8})$	`Cud8I`	WC7I

Class 8 $(\bar{L}R)(\bar{R}L)$		
Coefficient	**Name**	**Symmetry**
$\mathrm{Re}(C_{ledq})$	`CledqR`	WC5
$\mathrm{Im}(C_{ledq})$	`CledqI`	WC5

Class 8 $(\bar{L}L)(\bar{R}R)$		
Coefficient	**Name**	**Symmetry**
$\mathrm{Re}(C_{le})$	`CleR`	WC7R
$\mathrm{Im}(C_{le})$	`CleI`	WC7I
$\mathrm{Re}(C_{lu})$	`CluR`	WC7R
$\mathrm{Im}(C_{lu})$	`CluI`	WC7I
$\mathrm{Re}(C_{ld})$	`CldR`	WC7R
$\mathrm{Im}(C_{ld})$	`CldI`	WC7I
$\mathrm{Re}(C_{qe})$	`CqeR`	WC7R
$\mathrm{Im}(C_{qe})$	`CqeI`	WC7I
$\mathrm{Re}(C_{qu1})$	`Cqu1R`	WC7R
$\mathrm{Im}(C_{qu1})$	`Cqu1I`	WC7I
$\mathrm{Re}(C_{qu8})$	`Cqu8R`	WC7R
$\mathrm{Im}(C_{qu8})$	`Cqu8I`	WC7I
$\mathrm{Re}(C_{qd1})$	`Cqd1R`	WC7R
$\mathrm{Im}(C_{qd1})$	`Cqd1I`	WC7I
$\mathrm{Re}(C_{qd8})$	`Cqd8R`	WC7R
$\mathrm{Im}(C_{qd8})$	`Cqd8I`	WC7I

3.1.2 Member Function Documentation

3.1.2.1 ComputeCKMAndFermionMasses()

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

Compute CKM matrix and the mass of the fermions.

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

3.1.2.2 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 functions. After completing the evolution the values of the parameters are updated and are accessible with the GetCoefficients function.

Parameters

method	resolution method
muI	initial energy scale (in GeV)
muF	final energy scale (in GeV)

3.1.2.3 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. Using this function is the same of using Evolve with all the SMEFT coefficients set to 0, but it is faster since it does compute only the evolution for the SM parameters.

Parameters

method	resolution method
muI	initial energy scale (in GeV)
muF	final energy scale (in GeV)

3.1.2.4 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) at the scale `mu`, using one-loop pure SM beta functions.

If the flag CKMinput is set to `true` (default), the input for the Yukawa matrices will be generated from the current values 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.
The usage of this method with CKMinput=`true` should be restricted to realistic cases, with usual fermion mass hierarchy (smallest mass for the 1st generation and largest mass for the 3rd generation and no mass degeneracy) and with non-zero CKM matrix angles.
For more particular cases, the user should set CKMinput=`false` and set the input in terms of the Yukawa matrices.

Parameters

mu	Scale (in GeV) at which the initial conditions are generate
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.

3.1.2.5 GetCKMAngle()

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

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

Parameters

name	of the angle (see table 3.1)

Returns

The selected CKM angle.

3.1.2.6 GetCKMPhase()

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

Getter function for the CKM matrix phase δ.

Returns

The CKM matrix phase δ.

3.1.2.7 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 (see table 3.2)

Returns

the requested parameter

3.1.2.8 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 (see table 3.3)
i	first flavour index
j	second flavour index

Returns

the requested parameter

3.1.2.9 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 (see table 3.4)
i	first flavour index
j	second flavour index
k	third flavour index
l	fourth flavour index

Returns

the requested parameter

3.1.2.10 GetFermionMass()

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

Getter function for the mass of the fermions (in GeV).

Parameters

name	name of the fermion (see table 3.1)

Returns

the requested fermion mass

3.1.2.11 GetSMInputScale()

```
double RGESolver::GetSMInputScale ( )  [inline]
```

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

Returns

`InputScale_SM` (in GeV).

3.1.2.12 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

3.1.2.13 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

name	of the angle (see table 3.1)
val	its value

3.1.2.14 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	its value

3.1.2.15 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 (see table 3.2)
val	its value

3.1.2.16 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 (see table 3.3)
val	its value
i	first flavour index
j	second flavour index

3.1.2.17 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 (see table 3.4)
val	its value
i	first flavour index
j	second flavour index
k	third flavour index
l	fourth flavour index

3.1.2.18 SetFermionMass()

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

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

Parameters

name	name of the fermion (see table 3.1)
val	its value

3.1.2.19 SetSMInputScale()

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

Setter method for the scale at which the method GenerateSMInitialConditions takes the input values (in GeV). for SM parameters.

Parameters

mu	

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

- Solver/src/RGESolver.h
- Solver/src/RGESolver.cpp
- Solver/src/StaticMembers.cpp
- Solver/src/BetaFunction.cpp
- Solver/src/SettersAndGetters.cpp

Index