

RGESolver

Generated by Doxygen 1.8.20



<b>1</b>	<b>&lt;tt&gt;RGESolver&lt;/tt&gt;</b>	<b>1</b>
1.1	Dependencies	1
1.2	Installation	1
1.2.1	Command line options for the installation	2
1.3	Usage	2
1.4	Uninstall	2
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	RGESolver Class Reference	5
3.1.1	Detailed Description	6
3.1.2	Member Function Documentation	10
3.1.2.1	ComputeCKMAndFermionMasses()	10
3.1.2.2	Evolve()	11
3.1.2.3	EvolveSMOnly()	11
3.1.2.4	GenerateSMInitialConditions()	12
3.1.2.5	GetCKMAngle()	12
3.1.2.6	GetCKMPhase()	13
3.1.2.7	GetCoefficient() [1/3]	13
3.1.2.8	GetCoefficient() [2/3]	13
3.1.2.9	GetCoefficient() [3/3]	14
3.1.2.10	GetFermionMass()	14
3.1.2.11	GetSMInputScale()	15
3.1.2.12	SaveOutputFile()	15
3.1.2.13	SetCKMAngle()	15
3.1.2.14	SetCKMPhase()	15
3.1.2.15	SetCoefficient() [1/3]	16
3.1.2.16	SetCoefficient() [2/3]	16
3.1.2.17	SetCoefficient() [3/3]	17
3.1.2.18	SetFermionMass()	17
3.1.2.19	SetSMInputScale()	17
<b>Index</b>		<b>19</b>



# 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/`: CMake checks 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_INSTALL_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:

[RGESolver](#)

A class that performs renormalization group evolution in the context of the SMEFT . . . . . [5](#)





## 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).*
- 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

- void [ComputeCKMAndFermionMasses](#) ()  
*Documentation for the input/output handling.*
- void [SetFermionMass](#) (std::string name, double val)  
*Setter function for the mass of the fermions. Assignment 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 assignment 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  $\delta$ . The assignment is completed only if  $\delta \in (-\pi, \pi]$ .*
- double [GetCKMPhase](#) ()  
*Getter function for the CKM matrix phase  $\delta$ .*
- 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.*

### 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 user must set separately real and imaginary part of each complex parameter.

In tables [3.1](#), [3.2](#), [3.3](#) and [3.4](#) we list all the parameters, together with their names (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 setter functions.

## Author

S. Di Noi, L. Silvestrini.

## Copyright

GNU General Public License

**Table 3.1 Standard Model parameters. The parameters in the left column must be set (and accessed) with `\ref SetCoefficient` (and `\ref GetCoefficient`) methods. The ones in the right column must be set and accessed using other dedicated methods (see the specific documentation for input/output)**

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

Parameter	Name
$\theta_{12}$	CKM_theta12
$\theta_{13}$	CKM_theta13
$\theta_{23}$	CKM_theta23
$\delta$	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]	me1
$m_\mu$ [GeV]	mmu
$m_\tau$ [GeV]	mtau

**Table 3.2 Scalar (and real) SMEFT operators. They must be set and accessed using `\ref SetCoefficient` and `\ref GetCoefficient`.**

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 3.3 2F SMEFT operators. They must be set and accessed using `\ref SetCoefficient` and `\ref GetCoefficient`.**

<table> <tr> <th>Class 5 Coeffi- cient</th><th>Name</th><th>Sym- metry</th></tr> <tr> <td><math>\text{Re}(C_{eH})</math></td><td>CeHR</td><td>WC1</td></tr> <tr> <td><math>\text{Im}(C_{eH})</math></td><td>CeHI</td><td>WC1</td></tr> <tr> <td><math>\text{Re}(C_{uH})</math></td><td>CuHR</td><td>WC1</td></tr> <tr> <td><math>\text{Im}(C_{uH})</math></td><td>CuHI</td><td>WC1</td></tr> <tr> <td><math>\text{Re}(C_{dH})</math></td><td>CdHR</td><td>WC1</td></tr> <tr> <td><math>\text{Im}(C_{dH})</math></td><td>CdHI</td><td>WC1</td></tr> </table>	Class 5 Coeffi- cient	Name	Sym- metry	$\text{Re}(C_{eH})$	CeHR	WC1	$\text{Im}(C_{eH})$	CeHI	WC1	$\text{Re}(C_{uH})$	CuHR	WC1	$\text{Im}(C_{uH})$	CuHI	WC1	$\text{Re}(C_{dH})$	CdHR	WC1	$\text{Im}(C_{dH})$	CdHI	WC1	<table> <tr> <th>Class 6 Coeffi- cient</th><th>Name</th><th>Sym- metry</th></tr> <tr> <td><math>\text{Re}(C_{eW})</math></td><td>CeWR</td><td>WC1</td></tr> <tr> <td><math>\text{Im}(C_{eW})</math></td><td>CeWI</td><td>WC1</td></tr> <tr> <td><math>\text{Re}(C_{eB})</math></td><td>CeBR</td><td>WC1</td></tr> <tr> <td><math>\text{Im}(C_{eB})</math></td><td>CeBI</td><td>WC1</td></tr> <tr> <td><math>\text{Re}(C_{uG})</math></td><td>CuGR</td><td>WC1</td></tr> <tr> <td><math>\text{Im}(C_{uG})</math></td><td>CuGI</td><td>WC1</td></tr> <tr> <td><math>\text{Re}(C_{uW})</math></td><td>CuWR</td><td>WC1</td></tr> <tr> <td><math>\text{Im}(C_{uW})</math></td><td>CuWI</td><td>WC1</td></tr> <tr> <td><math>\text{Re}(C_{uB})</math></td><td>CuBR</td><td>WC1</td></tr> <tr> <td><math>\text{Im}(C_{uB})</math></td><td>CuBI</td><td>WC1</td></tr> <tr> <td><math>\text{Re}(C_{dG})</math></td><td>CdGR</td><td>WC1</td></tr> <tr> <td><math>\text{Im}(C_{dG})</math></td><td>CdGI</td><td>WC1</td></tr> <tr> <td><math>\text{Re}(C_{dW})</math></td><td>CdWR</td><td>WC1</td></tr> <tr> <td><math>\text{Im}(C_{dW})</math></td><td>CdWI</td><td>WC1</td></tr> <tr> <td><math>\text{Re}(C_{dB})</math></td><td>CdBR</td><td>WC1</td></tr> <tr> <td><math>\text{Im}(C_{dB})</math></td><td>CdBI</td><td>WC1</td></tr> </table>	Class 6 Coeffi- cient	Name	Sym- metry	$\text{Re}(C_{eW})$	CeWR	WC1	$\text{Im}(C_{eW})$	CeWI	WC1	$\text{Re}(C_{eB})$	CeBR	WC1	$\text{Im}(C_{eB})$	CeBI	WC1	$\text{Re}(C_{uG})$	CuGR	WC1	$\text{Im}(C_{uG})$	CuGI	WC1	$\text{Re}(C_{uW})$	CuWR	WC1	$\text{Im}(C_{uW})$	CuWI	WC1	$\text{Re}(C_{uB})$	CuBR	WC1	$\text{Im}(C_{uB})$	CuBI	WC1	$\text{Re}(C_{dG})$	CdGR	WC1	$\text{Im}(C_{dG})$	CdGI	WC1	$\text{Re}(C_{dW})$	CdWR	WC1	$\text{Im}(C_{dW})$	CdWI	WC1	$\text{Re}(C_{dB})$	CdBR	WC1	$\text{Im}(C_{dB})$	CdBI	WC1	<table> <tr> <th>Class 7 Coeffi- cient</th><th>Name</th><th>Sym- metry</th></tr> <tr> <td><math>\text{Re}(C_{H11})</math></td><td>CH11R</td><td>WC2R</td></tr> <tr> <td><math>\text{Im}(C_{H11})</math></td><td>CH11I</td><td>WC2I</td></tr> <tr> <td><math>\text{Re}(C_{H13})</math></td><td>CH13R</td><td>WC2R</td></tr> <tr> <td><math>\text{Im}(C_{H13})</math></td><td>CH13I</td><td>WC2I</td></tr> <tr> <td><math>\text{Re}(C_{He})</math></td><td>CHeR</td><td>WC2R</td></tr> <tr> <td><math>\text{Im}(C_{He})</math></td><td>CHeI</td><td>WC2I</td></tr> <tr> <td><math>\text{Re}(C_{Hq1})</math></td><td>CHq1R</td><td>WC2R</td></tr> <tr> <td><math>\text{Im}(C_{Hq1})</math></td><td>CHq1I</td><td>WC2I</td></tr> <tr> <td><math>\text{Re}(C_{Hq3})</math></td><td>CHq3R</td><td>WC2R</td></tr> <tr> <td><math>\text{Im}(C_{Hq3})</math></td><td>CHq3I</td><td>WC2I</td></tr> <tr> <td><math>\text{Re}(C_{Hu})</math></td><td>CHuR</td><td>WC2R</td></tr> <tr> <td><math>\text{Im}(C_{Hu})</math></td><td>CHuI</td><td>WC2I</td></tr> <tr> <td><math>\text{Re}(C_{Hd})</math></td><td>CHdR</td><td>WC2R</td></tr> <tr> <td><math>\text{Im}(C_{Hd})</math></td><td>CHdI</td><td>WC2I</td></tr> <tr> <td><math>\text{Re}(C_{Hud})</math></td><td>CHudR</td><td>WC1</td></tr> <tr> <td><math>\text{Im}(C_{Hud})</math></td><td>CHudI</td><td>WC1</td></tr> </table>	Class 7 Coeffi- cient	Name	Sym- metry	$\text{Re}(C_{H11})$	CH11R	WC2R	$\text{Im}(C_{H11})$	CH11I	WC2I	$\text{Re}(C_{H13})$	CH13R	WC2R	$\text{Im}(C_{H13})$	CH13I	WC2I	$\text{Re}(C_{He})$	CHeR	WC2R	$\text{Im}(C_{He})$	CHeI	WC2I	$\text{Re}(C_{Hq1})$	CHq1R	WC2R	$\text{Im}(C_{Hq1})$	CHq1I	WC2I	$\text{Re}(C_{Hq3})$	CHq3R	WC2R	$\text{Im}(C_{Hq3})$	CHq3I	WC2I	$\text{Re}(C_{Hu})$	CHuR	WC2R	$\text{Im}(C_{Hu})$	CHuI	WC2I	$\text{Re}(C_{Hd})$	CHdR	WC2R	$\text{Im}(C_{Hd})$	CHdI	WC2I	$\text{Re}(C_{Hud})$	CHudR	WC1	$\text{Im}(C_{Hud})$	CHudI	WC1
Class 5 Coeffi- cient	Name	Sym- metry																																																																																																																											
$\text{Re}(C_{eH})$	CeHR	WC1																																																																																																																											
$\text{Im}(C_{eH})$	CeHI	WC1																																																																																																																											
$\text{Re}(C_{uH})$	CuHR	WC1																																																																																																																											
$\text{Im}(C_{uH})$	CuHI	WC1																																																																																																																											
$\text{Re}(C_{dH})$	CdHR	WC1																																																																																																																											
$\text{Im}(C_{dH})$	CdHI	WC1																																																																																																																											
Class 6 Coeffi- cient	Name	Sym- metry																																																																																																																											
$\text{Re}(C_{eW})$	CeWR	WC1																																																																																																																											
$\text{Im}(C_{eW})$	CeWI	WC1																																																																																																																											
$\text{Re}(C_{eB})$	CeBR	WC1																																																																																																																											
$\text{Im}(C_{eB})$	CeBI	WC1																																																																																																																											
$\text{Re}(C_{uG})$	CuGR	WC1																																																																																																																											
$\text{Im}(C_{uG})$	CuGI	WC1																																																																																																																											
$\text{Re}(C_{uW})$	CuWR	WC1																																																																																																																											
$\text{Im}(C_{uW})$	CuWI	WC1																																																																																																																											
$\text{Re}(C_{uB})$	CuBR	WC1																																																																																																																											
$\text{Im}(C_{uB})$	CuBI	WC1																																																																																																																											
$\text{Re}(C_{dG})$	CdGR	WC1																																																																																																																											
$\text{Im}(C_{dG})$	CdGI	WC1																																																																																																																											
$\text{Re}(C_{dW})$	CdWR	WC1																																																																																																																											
$\text{Im}(C_{dW})$	CdWI	WC1																																																																																																																											
$\text{Re}(C_{dB})$	CdBR	WC1																																																																																																																											
$\text{Im}(C_{dB})$	CdBI	WC1																																																																																																																											
Class 7 Coeffi- cient	Name	Sym- metry																																																																																																																											
$\text{Re}(C_{H11})$	CH11R	WC2R																																																																																																																											
$\text{Im}(C_{H11})$	CH11I	WC2I																																																																																																																											
$\text{Re}(C_{H13})$	CH13R	WC2R																																																																																																																											
$\text{Im}(C_{H13})$	CH13I	WC2I																																																																																																																											
$\text{Re}(C_{He})$	CHeR	WC2R																																																																																																																											
$\text{Im}(C_{He})$	CHeI	WC2I																																																																																																																											
$\text{Re}(C_{Hq1})$	CHq1R	WC2R																																																																																																																											
$\text{Im}(C_{Hq1})$	CHq1I	WC2I																																																																																																																											
$\text{Re}(C_{Hq3})$	CHq3R	WC2R																																																																																																																											
$\text{Im}(C_{Hq3})$	CHq3I	WC2I																																																																																																																											
$\text{Re}(C_{Hu})$	CHuR	WC2R																																																																																																																											
$\text{Im}(C_{Hu})$	CHuI	WC2I																																																																																																																											
$\text{Re}(C_{Hd})$	CHdR	WC2R																																																																																																																											
$\text{Im}(C_{Hd})$	CHdI	WC2I																																																																																																																											
$\text{Re}(C_{Hud})$	CHudR	WC1																																																																																																																											
$\text{Im}(C_{Hud})$	CHudI	WC1																																																																																																																											

Table 3.4 4F SMEFT Operators. They must be set and accessed using \ref SetCoefficient and \ref GetCoefficient.

<table> <tr> <th colspan="3">Class 8 <math>(\bar{L}L)(\bar{L}L)</math></th></tr> <tr> <th>Coefficient</th><th>Name</th><th>Symmetry</th></tr> <tr> <td><math>\text{Re}(C_{ll})</math></td><td>C11R</td><td>WC6R</td></tr> <tr> <td><math>\text{Im}(C_{ll})</math></td><td>C11I</td><td>WC6I</td></tr> <tr> <td><math>\text{Re}(C_{qq1})</math></td><td>Cqq1R</td><td>WC6R</td></tr> <tr> <td><math>\text{Im}(C_{qq1})</math></td><td>Cqq1I</td><td>WC6I</td></tr> <tr> <td><math>\text{Re}(C_{qq3})</math></td><td>Cqq3R</td><td>WC6R</td></tr> <tr> <td><math>\text{Im}(C_{qq3})</math></td><td>Cqq3I</td><td>WC6I</td></tr> <tr> <td><math>\text{Re}(C_{lq1})</math></td><td>Clq1R</td><td>WC7R</td></tr> <tr> <td><math>\text{Im}(C_{lq1})</math></td><td>Clq1I</td><td>WC7I</td></tr> <tr> <td><math>\text{Re}(C_{lq3})</math></td><td>Clq3R</td><td>WC7R</td></tr> <tr> <td><math>\text{Im}(C_{lq3})</math></td><td>Clq3I</td><td>WC7I</td></tr> <tr> <th colspan="3">Class 8 <math>(\bar{L}R)(\bar{L}R)</math></th></tr> <tr> <th>Coefficient</th><th>Name</th><th>Symmetry</th></tr> <tr> <td><math>\text{Re}(C_{quqd1})</math></td><td>Cquqd1R</td><td>WC5</td></tr> <tr> <td><math>\text{Im}(C_{quqd1})</math></td><td>Cquqd1I</td><td>WC5</td></tr> <tr> <td><math>\text{Re}(C_{quqd8})</math></td><td>Cquqd8R</td><td>WC5</td></tr> <tr> <td><math>\text{Im}(C_{quqd8})</math></td><td>Cquqd8I</td><td>WC5</td></tr> <tr> <td><math>\text{Re}(C_{lequ1})</math></td><td>Clequ1R</td><td>WC5</td></tr> <tr> <td><math>\text{Im}(C_{lequ1})</math></td><td>Clequ1I</td><td>WC5</td></tr> <tr> <td><math>\text{Re}(C_{lequ3})</math></td><td>Clequ3R</td><td>WC5</td></tr> <tr> <td><math>\text{Im}(C_{lequ3})</math></td><td>Clequ3I</td><td>WC5</td></tr> </table>	Class 8 $(\bar{L}L)(\bar{L}L)$			Coefficient	Name	Symmetry	$\text{Re}(C_{ll})$	C11R	WC6R	$\text{Im}(C_{ll})$	C11I	WC6I	$\text{Re}(C_{qq1})$	Cqq1R	WC6R	$\text{Im}(C_{qq1})$	Cqq1I	WC6I	$\text{Re}(C_{qq3})$	Cqq3R	WC6R	$\text{Im}(C_{qq3})$	Cqq3I	WC6I	$\text{Re}(C_{lq1})$	Clq1R	WC7R	$\text{Im}(C_{lq1})$	Clq1I	WC7I	$\text{Re}(C_{lq3})$	Clq3R	WC7R	$\text{Im}(C_{lq3})$	Clq3I	WC7I	Class 8 $(\bar{L}R)(\bar{L}R)$			Coefficient	Name	Symmetry	$\text{Re}(C_{quqd1})$	Cquqd1R	WC5	$\text{Im}(C_{quqd1})$	Cquqd1I	WC5	$\text{Re}(C_{quqd8})$	Cquqd8R	WC5	$\text{Im}(C_{quqd8})$	Cquqd8I	WC5	$\text{Re}(C_{lequ1})$	Clequ1R	WC5	$\text{Im}(C_{lequ1})$	Clequ1I	WC5	$\text{Re}(C_{lequ3})$	Clequ3R	WC5	$\text{Im}(C_{lequ3})$	Clequ3I	WC5	<table> <tr> <th colspan="3">Class 8 <math>(\bar{R}R)(\bar{R}R)</math></th></tr> <tr> <th>Coefficient</th><th>Name</th><th>Symmetry</th></tr> <tr> <td><math>\text{Re}(C_{ee})</math></td><td>CeeR</td><td>WC8R</td></tr> <tr> <td><math>\text{Im}(C_{ee})</math></td><td>CeeI</td><td>WC8I</td></tr> <tr> <td><math>\text{Re}(C_{uu})</math></td><td>CuuR</td><td>WC6R</td></tr> <tr> <td><math>\text{Im}(C_{uu})</math></td><td>CuuI</td><td>WC6I</td></tr> <tr> <td><math>\text{Re}(C_{dd})</math></td><td>CddR</td><td>WC6R</td></tr> <tr> <td><math>\text{Im}(C_{dd})</math></td><td>CddI</td><td>WC6I</td></tr> <tr> <td><math>\text{Re}(C_{eu})</math></td><td>CeuR</td><td>WC7R</td></tr> <tr> <td><math>\text{Im}(C_{eu})</math></td><td>CeuI</td><td>WC7I</td></tr> <tr> <td><math>\text{Re}(C_{ed})</math></td><td>CedR</td><td>WC7R</td></tr> <tr> <td><math>\text{Im}(C_{ed})</math></td><td>CedI</td><td>WC7I</td></tr> <tr> <td><math>\text{Re}(C_{ud1})</math></td><td>Cud1R</td><td>WC7R</td></tr> <tr> <td><math>\text{Im}(C_{ud1})</math></td><td>Cud1I</td><td>WC7I</td></tr> <tr> <td><math>\text{Re}(C_{ud8})</math></td><td>Cud8R</td><td>WC7R</td></tr> <tr> <td><math>\text{Im}(C_{ud8})</math></td><td>Cud8I</td><td>WC7I</td></tr> <tr> <th colspan="3">Class 8 <math>(\bar{L}R)(\bar{R}L)</math></th></tr> <tr> <th>Coefficient</th><th>Name</th><th>Symmetry</th></tr> <tr> <td><math>\text{Re}(C_{ledq})</math></td><td>CledqR</td><td>WC5</td></tr> <tr> <td><math>\text{Im}(C_{ledq})</math></td><td>CledqI</td><td>WC5</td></tr> </table>	Class 8 $(\bar{R}R)(\bar{R}R)$			Coefficient	Name	Symmetry	$\text{Re}(C_{ee})$	CeeR	WC8R	$\text{Im}(C_{ee})$	CeeI	WC8I	$\text{Re}(C_{uu})$	CuuR	WC6R	$\text{Im}(C_{uu})$	CuuI	WC6I	$\text{Re}(C_{dd})$	CddR	WC6R	$\text{Im}(C_{dd})$	CddI	WC6I	$\text{Re}(C_{eu})$	CeuR	WC7R	$\text{Im}(C_{eu})$	CeuI	WC7I	$\text{Re}(C_{ed})$	CedR	WC7R	$\text{Im}(C_{ed})$	CedI	WC7I	$\text{Re}(C_{ud1})$	Cud1R	WC7R	$\text{Im}(C_{ud1})$	Cud1I	WC7I	$\text{Re}(C_{ud8})$	Cud8R	WC7R	$\text{Im}(C_{ud8})$	Cud8I	WC7I	Class 8 $(\bar{L}R)(\bar{R}L)$			Coefficient	Name	Symmetry	$\text{Re}(C_{ledq})$	CledqR	WC5	$\text{Im}(C_{ledq})$	CledqI	WC5	<table> <tr> <th colspan="3">Class 8 <math>(\bar{L}L)(\bar{R}R)</math></th></tr> <tr> <th>Coefficient</th><th>Name</th><th>Symmetry</th></tr> <tr> <td><math>\text{Re}(C_{le})</math></td><td>CleR</td><td>WC7R</td></tr> <tr> <td><math>\text{Im}(C_{le})</math></td><td>CleI</td><td>WC7I</td></tr> <tr> <td><math>\text{Re}(C_{lu})</math></td><td>CluR</td><td>WC7R</td></tr> <tr> <td><math>\text{Im}(C_{lu})</math></td><td>CluI</td><td>WC7I</td></tr> <tr> <td><math>\text{Re}(C_{ld})</math></td><td>CldR</td><td>WC7R</td></tr> <tr> <td><math>\text{Im}(C_{ld})</math></td><td>CldI</td><td>WC7I</td></tr> <tr> <td><math>\text{Re}(C_{qe})</math></td><td>CqeR</td><td>WC7R</td></tr> <tr> <td><math>\text{Im}(C_{qe})</math></td><td>CqeI</td><td>WC7I</td></tr> <tr> <td><math>\text{Re}(C_{qu1})</math></td><td>Cqu1R</td><td>WC7R</td></tr> <tr> <td><math>\text{Im}(C_{qu1})</math></td><td>Cqu1I</td><td>WC7I</td></tr> <tr> <td><math>\text{Re}(C_{qu8})</math></td><td>Cqu8R</td><td>WC7R</td></tr> <tr> <td><math>\text{Im}(C_{qu8})</math></td><td>Cqu8I</td><td>WC7I</td></tr> <tr> <td><math>\text{Re}(C_{qd1})</math></td><td>Cqd1R</td><td>WC7R</td></tr> <tr> <td><math>\text{Im}(C_{qd1})</math></td><td>Cqd1I</td><td>WC7I</td></tr> <tr> <td><math>\text{Re}(C_{qd8})</math></td><td>Cqd8R</td><td>WC7R</td></tr> <tr> <td><math>\text{Im}(C_{qd8})</math></td><td>Cqd8I</td><td>WC7I</td></tr> </table>	Class 8 $(\bar{L}L)(\bar{R}R)$			Coefficient	Name	Symmetry	$\text{Re}(C_{le})$	CleR	WC7R	$\text{Im}(C_{le})$	CleI	WC7I	$\text{Re}(C_{lu})$	CluR	WC7R	$\text{Im}(C_{lu})$	CluI	WC7I	$\text{Re}(C_{ld})$	CldR	WC7R	$\text{Im}(C_{ld})$	CldI	WC7I	$\text{Re}(C_{qe})$	CqeR	WC7R	$\text{Im}(C_{qe})$	CqeI	WC7I	$\text{Re}(C_{qu1})$	Cqu1R	WC7R	$\text{Im}(C_{qu1})$	Cqu1I	WC7I	$\text{Re}(C_{qu8})$	Cqu8R	WC7R	$\text{Im}(C_{qu8})$	Cqu8I	WC7I	$\text{Re}(C_{qd1})$	Cqd1R	WC7R	$\text{Im}(C_{qd1})$	Cqd1I	WC7I	$\text{Re}(C_{qd8})$	Cqd8R	WC7R	$\text{Im}(C_{qd8})$	Cqd8I	WC7I
Class 8 $(\bar{L}L)(\bar{L}L)$																																																																																																																																																																																						
Coefficient	Name	Symmetry																																																																																																																																																																																				
$\text{Re}(C_{ll})$	C11R	WC6R																																																																																																																																																																																				
$\text{Im}(C_{ll})$	C11I	WC6I																																																																																																																																																																																				
$\text{Re}(C_{qq1})$	Cqq1R	WC6R																																																																																																																																																																																				
$\text{Im}(C_{qq1})$	Cqq1I	WC6I																																																																																																																																																																																				
$\text{Re}(C_{qq3})$	Cqq3R	WC6R																																																																																																																																																																																				
$\text{Im}(C_{qq3})$	Cqq3I	WC6I																																																																																																																																																																																				
$\text{Re}(C_{lq1})$	Clq1R	WC7R																																																																																																																																																																																				
$\text{Im}(C_{lq1})$	Clq1I	WC7I																																																																																																																																																																																				
$\text{Re}(C_{lq3})$	Clq3R	WC7R																																																																																																																																																																																				
$\text{Im}(C_{lq3})$	Clq3I	WC7I																																																																																																																																																																																				
Class 8 $(\bar{L}R)(\bar{L}R)$																																																																																																																																																																																						
Coefficient	Name	Symmetry																																																																																																																																																																																				
$\text{Re}(C_{quqd1})$	Cquqd1R	WC5																																																																																																																																																																																				
$\text{Im}(C_{quqd1})$	Cquqd1I	WC5																																																																																																																																																																																				
$\text{Re}(C_{quqd8})$	Cquqd8R	WC5																																																																																																																																																																																				
$\text{Im}(C_{quqd8})$	Cquqd8I	WC5																																																																																																																																																																																				
$\text{Re}(C_{lequ1})$	Clequ1R	WC5																																																																																																																																																																																				
$\text{Im}(C_{lequ1})$	Clequ1I	WC5																																																																																																																																																																																				
$\text{Re}(C_{lequ3})$	Clequ3R	WC5																																																																																																																																																																																				
$\text{Im}(C_{lequ3})$	Clequ3I	WC5																																																																																																																																																																																				
Class 8 $(\bar{R}R)(\bar{R}R)$																																																																																																																																																																																						
Coefficient	Name	Symmetry																																																																																																																																																																																				
$\text{Re}(C_{ee})$	CeeR	WC8R																																																																																																																																																																																				
$\text{Im}(C_{ee})$	CeeI	WC8I																																																																																																																																																																																				
$\text{Re}(C_{uu})$	CuuR	WC6R																																																																																																																																																																																				
$\text{Im}(C_{uu})$	CuuI	WC6I																																																																																																																																																																																				
$\text{Re}(C_{dd})$	CddR	WC6R																																																																																																																																																																																				
$\text{Im}(C_{dd})$	CddI	WC6I																																																																																																																																																																																				
$\text{Re}(C_{eu})$	CeuR	WC7R																																																																																																																																																																																				
$\text{Im}(C_{eu})$	CeuI	WC7I																																																																																																																																																																																				
$\text{Re}(C_{ed})$	CedR	WC7R																																																																																																																																																																																				
$\text{Im}(C_{ed})$	CedI	WC7I																																																																																																																																																																																				
$\text{Re}(C_{ud1})$	Cud1R	WC7R																																																																																																																																																																																				
$\text{Im}(C_{ud1})$	Cud1I	WC7I																																																																																																																																																																																				
$\text{Re}(C_{ud8})$	Cud8R	WC7R																																																																																																																																																																																				
$\text{Im}(C_{ud8})$	Cud8I	WC7I																																																																																																																																																																																				
Class 8 $(\bar{L}R)(\bar{R}L)$																																																																																																																																																																																						
Coefficient	Name	Symmetry																																																																																																																																																																																				
$\text{Re}(C_{ledq})$	CledqR	WC5																																																																																																																																																																																				
$\text{Im}(C_{ledq})$	CledqI	WC5																																																																																																																																																																																				
Class 8 $(\bar{L}L)(\bar{R}R)$																																																																																																																																																																																						
Coefficient	Name	Symmetry																																																																																																																																																																																				
$\text{Re}(C_{le})$	CleR	WC7R																																																																																																																																																																																				
$\text{Im}(C_{le})$	CleI	WC7I																																																																																																																																																																																				
$\text{Re}(C_{lu})$	CluR	WC7R																																																																																																																																																																																				
$\text{Im}(C_{lu})$	CluI	WC7I																																																																																																																																																																																				
$\text{Re}(C_{ld})$	CldR	WC7R																																																																																																																																																																																				
$\text{Im}(C_{ld})$	CldI	WC7I																																																																																																																																																																																				
$\text{Re}(C_{qe})$	CqeR	WC7R																																																																																																																																																																																				
$\text{Im}(C_{qe})$	CqeI	WC7I																																																																																																																																																																																				
$\text{Re}(C_{qu1})$	Cqu1R	WC7R																																																																																																																																																																																				
$\text{Im}(C_{qu1})$	Cqu1I	WC7I																																																																																																																																																																																				
$\text{Re}(C_{qu8})$	Cqu8R	WC7R																																																																																																																																																																																				
$\text{Im}(C_{qu8})$	Cqu8I	WC7I																																																																																																																																																																																				
$\text{Re}(C_{qd1})$	Cqd1R	WC7R																																																																																																																																																																																				
$\text{Im}(C_{qd1})$	Cqd1I	WC7I																																																																																																																																																																																				
$\text{Re}(C_{qd8})$	Cqd8R	WC7R																																																																																																																																																																																				
$\text{Im}(C_{qd8})$	Cqd8I	WC7I																																																																																																																																																																																				

### 3.1.2 Member Function Documentation

#### 3.1.2.1 ComputeCKMAndFermionMasses()

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

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  $g_1, g_2, g_3, \lambda, m_h^2, \text{Re}(\mathcal{Y}_u), \text{Im}(\mathcal{Y}_u), \text{Re}(\mathcal{Y}_d), \text{Im}(\mathcal{Y}_d), \text{Re}(\mathcal{Y}_e), \text{Im}(\mathcal{Y}_e)$  (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](#)

Compute CKM matrix and the mass of the fermions.

The methods [Evolve](#) and [EvolveSMAOnly](#) 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 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  $\mu_I$  to  $\mu_F$ . Currently, the available methods are "Numeric" and "Leading-Log".

The evolver 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

<i>method</i>	resolution method
<i>muI</i>	initial energy scale
<i>muF</i>	final energy scale

### 3.1.2.3 EvolveSMAOnly()

```
void RGESolver::EvolveSMAOnly (
    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

<i>method</i>	resolution method
<i>muI</i>	initial energy scale
<i>muF</i>	final energy scale

### 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).

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

<i>mu</i>	Scale (in GeV) at which the initial conditions are generated. If <i>mu</i> is different from the scale at which the input is given (SMInputScale), <a href="#">RGESolver</a> will use the pure SM RGEs (at one-loop level) to run the parameters to the scale <i>mu</i> .
<i>basis</i>	Flavour basis ( "UP" or "DOWN" )
<i>method</i>	Method used by <a href="#">RGESolver</a> to run the SM parameters to the scale <i>mu</i> ("Numeric" or "Leading-Log")
<i>inputCKM</i>	If set to <code>true</code> (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

<i>name</i>	of the angle (see table <a href="#">3.1</a> )
-------------	---

#### Returns

The selected CKM angle.



**3.1.2.6 GetCKMPhase()**

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

Getter function for the CKM matrix phase  $\delta$ .

**Returns**

$\delta$ .

**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**

<i>name</i>	name of the parameter (see table <a href="#">3.2</a> )
-------------	--

**Returns**

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

**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**

<i>name</i>	name of the parameter (see table <a href="#">3.3</a> )
<i>i</i>	first flavour index
<i>j</i>	second flavour index

**Returns**

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

**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**

<i>name</i>	name of the parameter (see table <a href="#">3.4</a> )
<i>i</i>	first flavour index
<i>j</i>	second flavour index
<i>k</i>	third flavour index
<i>l</i>	fourth flavour index

**Returns**

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

**3.1.2.10 GetFermionMass()**

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

Getter function for the mass of the fermions.

**Parameters**

<i>name</i>	name of the fermion (see table <a href="#">3.1</a> )
-------------	--

**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

**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**

<i>filename</i>	Name of the output file
<i>format</i>	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 assignment is completed only if the inserted angle is  $\in [0, \frac{\pi}{2}]$ .

**Parameters**

<i>name</i>	of the angle (see table <a href="#">3.1</a> )
<i>val</i>	

**3.1.2.14 SetCKMPhase()**

```
void RGESolver::SetCKMPhase (
```

```
double val )
```

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

#### Parameters

<i>val</i>	
------------	--

### 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

<i>name</i>	name of the parameter (see table <a href="#">3.2</a> )
<i>val</i>	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

<i>name</i>	name of the parameter (see table <a href="#">3.3</a> )
<i>val</i>	its value
<i>i</i>	first flavour index
<i>j</i>	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**

<i>name</i>	name of the parameter (see table <a href="#">3.4</a> )
<i>val</i>	its value
<i>i</i>	first flavour index
<i>j</i>	second flavour index
<i>k</i>	third flavour index
<i>l</i>	fourth flavour index

**3.1.2.18 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**

<i>name</i>	name of the fermion (see table <a href="#">3.1</a> )
<i>val</i>	

**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 for SM parameters.

**Parameters**

<i>mu</i>	
-----------	--

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

ComputeCKMAndFermionMasses

RGESolver, [10](#)

Evolve

RGESolver, [11](#)

EvolveSMAOnly

RGESolver, [11](#)

GenerateSMInitialConditions

RGESolver, [12](#)

GetCKMAngle

RGESolver, [12](#)

GetCKMPhase

RGESolver, [12](#)

GetCoefficient

RGESolver, [13](#), [14](#)

GetFermionMass

RGESolver, [14](#)

GetSMInputScale

RGESolver, [14](#)

RGESolver, [5](#)

ComputeCKMAndFermionMasses, [10](#)

Evolve, [11](#)

EvolveSMAOnly, [11](#)

GenerateSMInitialConditions, [12](#)

GetCKMAngle, [12](#)

GetCKMPhase, [12](#)

GetCoefficient, [13](#), [14](#)

GetFermionMass, [14](#)

GetSMInputScale, [14](#)

SaveOutputFile, [15](#)

SetCKMAngle, [15](#)

SetCKMPhase, [15](#)

SetCoefficient, [16](#)

SetFermionMass, [17](#)

SetSMInputScale, [17](#)

SaveOutputFile

RGESolver, [15](#)

SetCKMAngle

RGESolver, [15](#)

SetCKMPhase

RGESolver, [15](#)

SetCoefficient

RGESolver, [16](#)

SetFermionMass

RGESolver, [17](#)

SetSMInputScale

RGESolver, [17](#)