

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	Constructor & Destructor Documentation	11
3.1.2.1	RGESolver()	11
3.1.3	Member Function Documentation	11
3.1.3.1	Evolve()	12
3.1.3.2	EvolveSMOnly()	12
3.1.3.3	EvolveToBasis()	12
3.1.3.4	GenerateSMInitialConditions() [1/2]	13
3.1.3.5	GenerateSMInitialConditions() [2/2]	13
3.1.3.6	GetCKMAngle()	14
3.1.3.7	GetCKMImagPart()	15
3.1.3.8	GetCKMPhase()	15
3.1.3.9	GetCKMRealPart()	15
3.1.3.10	GetCoefficient() [1/3]	16
3.1.3.11	GetCoefficient() [2/3]	16
3.1.3.12	GetCoefficient() [3/3]	16
3.1.3.13	SaveOutputFile()	17
3.1.3.14	SetCoefficient() [1/3]	17
3.1.3.15	SetCoefficient() [2/3]	18
3.1.3.16	SetCoefficient() [3/3]	18
	<b>Index</b>	<b>21</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 [CMake website](#). Clone the repository with

```
git clone https://github.com/silvest/RGESolver --recursively
```

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 `Example1.cpp` (available in the `Examples` directory) can be compiled with the command

```
g++ -o app Example1.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.*
- void [Setepsrel](#) (double [epsrel](#))  
*Setter for the relative error used in the numerical integration (default value = 0.005)*
- void [Setepsabs](#) (double [epsabs](#))  
*Setter for the absolute error used in the numerical integration (default value = e-13)*

#### Evolution

- void [Evolve](#) (std::string method, double muI, double muF)  
*Performs the RGE evolution.*
- void [EvolveToBasis](#) (std::string method, double muI, double muF, std::string basis)  
*Performs the RGE evolution and the back rotation on the coefficients with flavour indices.*
- void [GenerateSMInitialConditions](#) (double mu, std::string basis, std::string method)  
*Generates the initial conditions for Standard Model's parameters (gauge couplings, Yukawa coupling, quartic coupling and Higgs' boson mass) at the scale  $\mu$  (in GeV), using one-loop pure SM beta functions. Default low-energy input is used.*
- void [GenerateSMInitialConditions](#) (double muIn, double muFin, std::string basis, std::string method, double g1in, double g2in, double g3in, double lambdain, double mh2in, double Muin[3], double Mdin[3], double Mein[3], double s12in, double s13in, double s23in, double deltain)

Generates the initial conditions for Standard Model's parameters (gauge couplings, Yukawa coupling, quartic coupling and Higgs' boson mass) at the scale  $\mu_U$  (in GeV), using one-loop pure SM beta functions. User-defined low energy input is used.

- void [EvolveSMOnly](#) (std::string method, double  $\mu_U$ , double  $\mu_F$ )

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.

## Input/output

- double [GetCKMAngle](#) (std::string name)  
Getter function for the CKM matrix angles  $\theta_{12}, \theta_{13}, \theta_{23}$ .
- double [GetCKMRealPart](#) (int i, int j)  
Getter function for the CKM matrix (real part)
- double [GetCKMImagPart](#) (int i, int j)  
Getter function for the CKM matrix (imaginary part)
- double [GetCKMPhase](#) ()  
Getter function for the CKM matrix phase  $\delta$ .
- 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). one of the inserted indices is outside the [0:2] range, an error message is printed and the value 0 is returned.
- void [Reset](#) ()  
Resets all the SMEFT coefficients to 0 and the SM parameters to their default value.  $\epsilon_{abs}$  and  $\epsilon_{rel}$  are reset 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>. [RGESolver](#) 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  $\epsilon_{rel}$  and  $\epsilon_{abs}$  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  $g_1, g_2, g_3, \lambda, m_h^2, \text{Re}(\mathcal{Y}_u), \text{Im}(\mathcal{Y}_u), \text{Re}(\mathcal{Y}_d), \text{Im}(\mathcal{Y}_d), \text{Re}(\mathcal{Y}_e), \text{Im}(\mathcal{Y}_e)$  (we follow <https://arxiv.org/abs/1308.2627> for what concerns the conventions in the Higgs' sector).

The routines [GetCKMAngle](#), [GetCKMPhase](#), [GetCKMRealPart](#), [GetCKMImagPart](#) should be used when interested in the CKM parameters or elements. The usage of this method is recommended after methods such [GenerateSMInitialConditions](#) or [EvolveToBasis](#) that choose a specific flavour basis ("UP" or "DOWN"), in which cases the CKM matrix is updated. 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](#).

A summary of the operators symmetry classes is given in table [3.5](#).

We follow <http://www.utfit.org/UTfit/Formalism> for what concerns the conventions for the CKM matrix.

#### Author

S. Di Noi, L. Silvestrini.

#### Copyright

GNU General Public License

**Table 3.1 Standard Model parameters. The labels in the left column must be used with the `GetCoefficient/SetCoefficient` methods, the ones in the right column must be used with `GetCKMAngle` methods.**

Parameter	Name		Parameter	Name
$g_1$	g1		$\sin(\theta_{12})$	s12
$g_2$	g2		$\sin(\theta_{13})$	s13
$g_3$	g3		$\sin(\theta_{23})$	s23
$\lambda$	lambda			
$m_h^2 [\text{GeV}^2]$	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			

**Table 3.2 Scalar (and real) SMEFT operators.**

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.

<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.

<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																																																																																																																																																																																				

**Table 3.5 Symmetry categories for operators in the SMEFT. nF indicates the number of flavour indices for each category.**

Parameter	Name
0	0F scalar object
WC1	2F generic real matrix
WC2R	2F Hermitian matrix (real part)
WC2I	2F Hermitian matrix (imaginary part)
WC5	4F generic real object
WC6R	4F two identical $\bar{\psi}\psi$ currents (real part)
WC6I	4F two identical $\bar{\psi}\psi$ currents (imaginary part)
WC7R	4F two independent $\bar{\psi}\psi$ currents (real part)

Parameter	Name
WC7I	4F two independent $\bar{\psi}\psi$ currents (imaginary part)
WC8R	$\mathcal{C}_{ee}$ (real part)
WC8I	$\mathcal{C}_{ee}$ (imaginary part)

**Table 3.6** SM parameters used by default to generate SM initial conditions at an arbitrary scale. The scale at which these parameters are given is  $\mu = 173.65$  GeV. We follow <http://www.utfit.org/UTfit/Formalism> for what concerns the conventions for the CKM matrix.

Parameter	Value	Parameter	Value [GeV]
$g_1$	0.3573	$m_u$	0.0012
$g_2$	0.6511	$m_c$	0.640
$g_3$	1.161	$m_t$	162.0
$\lambda$	0.1297	$m_d$	0.0027
$m_h^2$ [GeV <sup>2</sup> ]	15650	$m_s$	0.052
$\sin(\theta_{12})$	0.225	$m_b$	2.75
$\sin(\theta_{13})$	0.042	$m_e$	0.000511
$\sin(\theta_{23})$	0.003675	$m_\mu$	0.1057
$\delta$ [rad]	1.1676	$m_\tau$	1.776

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 RGESolver()

```
RGESolver::RGESolver ( )
```

The default constructor.

It initializes to 0 all the SMEFT coefficients.

### 3.1.3 Member Function Documentation

### 3.1.3.1 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 method 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 [GetCoefficient](#) functions.

#### Parameters

<i>method</i>	resolution method
<i>muI</i>	initial energy scale (in GeV)
<i>muF</i>	final energy scale (in GeV)

### 3.1.3.2 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

<i>method</i>	resolution method
<i>muI</i>	initial energy scale (in GeV)
<i>muF</i>	final energy scale (in GeV)

### 3.1.3.3 EvolveToBasis()

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



Performs the RGE evolution and the back rotation on the coefficients with flavour indices.

After the evolution, the CKM matrix is computed. A flavour rotation is performed on the coefficients to go in the chosen basis.

#### Parameters

<i>method</i>	resolution method
<i>muI</i>	initial energy scale (in GeV)
<i>muF</i>	final energy scale (in GeV)
<i>basis</i>	flavour basis after the evolution ("UP" or "DOWN").

#### 3.1.3.4 GenerateSMInitialConditions() [1/2]

```
void RGESolver::GenerateSMInitialConditions (
    double mu,
    std::string basis,
    std::string method )
```

Generates the initial conditions for Standard Model's parameters (gauge couplings, Yukawa coupling, quartic coupling and Higgs' boson mass) at the scale `mu` (in GeV), using one-loop pure SM beta functions. Default low-energy input is used.

The initial conditions are generated at the scale `mu` starting from the values at  $\mu = 173.65$  GeV in table 3.6. At the scale `mu` the CKM matrix is computed.

#### Parameters

<i>mu</i>	Scale (in GeV) at which the initial conditions are generated
<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 <code>mu</code> ("Numeric" or "Leading-Log")

#### 3.1.3.5 GenerateSMInitialConditions() [2/2]

```
void RGESolver::GenerateSMInitialConditions (
    double muIn,
    double muFin,
    std::string basis,
    std::string method,
    double g1in,
    double g2in,
    double g3in,
    double lambdain,
    double mh2in,
    double Muin[3],
```

```

double Mdin[3],
double Mein[3],
double s12in,
double s13in,
double s23in,
double deltain )

```

Generates the initial conditions for Standard Model's parameters (gauge couplings, Yukawa coupling, quartic coupling and Higgs' boson mass) at the scale  $\mu$  (in GeV), using one-loop pure SM beta functions. User-defined low energy input is used.

The initial conditions are generated at the scale  $\mu_{\text{Fin}}$  starting from the inserted parameters at the scale  $\mu_{\text{In}}$ . This method should be used with usual fermion hierarchy (smallest mass for the 1st generation and greatest mass for the 3rd without mass degeneracy for all up and down quarks and for charged leptons). The generation of the initial conditions is performed only if all the masses are non-negative and if  $\sin \theta_{ij} \in (0, 1)$ ,  $\delta \in (\pi, \pi]$ . We follow <http://www.utfit.org/UTfit> for what concerns the conventions for the CKM matrix. At the scale  $\mu$  the CKM matrix is computed.

#### Parameters

<i>muln</i>	Low-energy input scale (in GeV)
<i>muln</i>	Scale (in GeV) at which the initial conditions are generated
<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 $\mu$ ("Numeric" or "Leading-Log")
<i>g1in</i>	$g_1$
<i>g2in</i>	$g_2$
<i>g3in</i>	$g_3$
<i>lambdain</i>	$\lambda$
<i>mh2in</i>	$m_h^2$ (in $\text{GeV}^2$ )
<i>Muin</i>	Array containing the masses of the up-type quarks in GeV in the order $(m_u, m_c, m_t)$
<i>Mdin</i>	Array containing the masses of the down-type quarks in GeV in the order $(m_d, m_s, m_b)$
<i>Mein</i>	Array containing the masses of the charged leptons in GeV in the order $(m_e, m_\mu, m_\tau)$
<i>s12in</i>	The sine of the CKM matrix angle $\sin \theta_{12}$
<i>s13in</i>	The sine of the CKM matrix angle $\sin \theta_{13}$
<i>s23in</i>	The sine of the CKM matrix angle $\sin \theta_{23}$
<i>deltain</i>	The CKM matrix phase $\delta$

#### 3.1.3.6 GetCKMAngle()

```

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

```

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

This method should be called only after methods that choose a specific flavour basis (as [GenerateSMInitialConditions](#) or [EvolveToBasis](#) ), otherwise the CKM matrix is not updated.

## Parameters

<i>name</i>	of the angle (see table 3.1)
-------------	------------------------------

## Returns

The selected CKM angle.

## 3.1.3.7 GetCKMImagPart()

```
double RGESolver::GetCKMImagPart (
    int i,
    int j ) [inline]
```

Getter function for the CKM matrix (imaginary part)

This method should be called only after methods that choose a specific flavour basis (as [GenerateSMInitialConditions](#) or [EvolveToBasis](#) ), otherwise the CKM matrix is not updated.

## Returns

The imaginary part of the selected CKM matrix element.

## 3.1.3.8 GetCKMPhase()

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

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

This method should be called only after methods that choose a specific flavour basis (as [GenerateSMInitialConditions](#) or [EvolveToBasis](#) ), otherwise the CKM matrix is not updated.

## Returns

The CKM matrix phase  $\delta$ .

## 3.1.3.9 GetCKMRealPart()

```
double RGESolver::GetCKMRealPart (
    int i,
    int j ) [inline]
```

Getter function for the CKM matrix (real part)

This method should be called only after methods that choose a specific flavour basis (as [GenerateSMInitialConditions](#) or [EvolveToBasis](#) ), otherwise the CKM matrix is not updated.

## Returns

The real part of the selected CKM matrix element.

**3.1.3.10 GetCoefficient()** [1/3]

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

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

**Parameters**

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

**Returns**

the requested parameter

**3.1.3.11 GetCoefficient()** [2/3]

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

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

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

**3.1.3.12 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). 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

**3.1.3.13 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

**3.1.3.14 SetCoefficient() [1/3]**

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

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

**Parameters**

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

**3.1.3.15 SetCoefficient()** [2/3]

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

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

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="#">3.3</a> )
<i>val</i>	its value
<i>i</i>	first flavour index
<i>j</i>	second flavour index

**3.1.3.16 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 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

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

- Evolve
  - RGESolver, [11](#)
- EvolveSMAOnly
  - RGESolver, [12](#)
- EvolveToBasis
  - RGESolver, [12](#)
- GenerateSMInitialConditions
  - RGESolver, [13](#)
- GetCKMAngle
  - RGESolver, [14](#)
- GetCKMImagPart
  - RGESolver, [15](#)
- GetCKMPhase
  - RGESolver, [15](#)
- GetCKMRealPart
  - RGESolver, [15](#)
- GetCoefficient
  - RGESolver, [15](#), [16](#)
- RGESolver, [5](#)
  - Evolve, [11](#)
  - EvolveSMAOnly, [12](#)
  - EvolveToBasis, [12](#)
  - GenerateSMInitialConditions, [13](#)
  - GetCKMAngle, [14](#)
  - GetCKMImagPart, [15](#)
  - GetCKMPhase, [15](#)
  - GetCKMRealPart, [15](#)
  - GetCoefficient, [15](#), [16](#)
  - RGESolver, [11](#)
  - SaveOutputFile, [17](#)
  - SetCoefficient, [17](#), [18](#)
- SaveOutputFile
  - RGESolver, [17](#)
- SetCoefficient
  - RGESolver, [17](#), [18](#)