Eur. Phys. J. C (2023) 83:200
https://doi.org/10.1140/epjc/s10052-023-11189-4

THE EUROPEAN PHYSICAL JOURNAL C

Regular Article - Theoretical Physics

RGESolver: a C++ library to perform renormalization group evolution in the Standard Model Effective Theory

Stefano Di Noi[1,2,a], **Luca Silvestrini**[3,b]

[1] Dipartimento di Fisica e Astronomia, Universitá di Padova, Via Marzolo 8, 35131 Padua, Italy
[2] Sezione di Padova, Istituto Nazionale di Fisica Nucleare, Via Marzolo 8, 35131 Padua, Italy
[3] Sezione di Roma, Istituto Nazionale di Fisica Nucleare, Piazzale Aldo Moro 2, 00185 Rome, Italy

Received: 18 November 2022 / Accepted: 30 December 2022 / Published online: 7 March 2023

Abstract Renormalization group evolution above the electroweak scale is a crucial ingredient in the phenomenology of the Standard Model Effective Theory. The `RGESolver` open-source `C++` library performs the evolution at leading order for dimension-six operators in the most general flavour scenario (assuming lepton and baryon number conservation). Given its efficiency, `RGESolver` can be used to include the effects of renormalization group evolution in extensive phenomenological analyses in the framework of the Standard Model Effective Theory.

1 Introduction

The Standard Model (SM) is one of the biggest scientific successes of our time: it describes three (weak, strong and electromagnetic) of the four currently known elementary interactions in nature, closing a long path started between the nineteenth and the twentieth century. The particle content of the SM was completed in 2012 with the discovery of a Higgs-like boson with properties consistent with the SM within current experimental errors. In spite of its success, however, the SM leaves several phenomena unexplained. For example, neutrino masses, the origin of the baryon asymmetry in the universe and dark matter suggest the presence of physics beyond the SM. The absence of direct evidence of new particles at energies $\mathcal{O}$ (TeV) allows us to parametrize the effects of possible heavy New Physics (NP), lying beyond the reach of the LHC for direct production, with an effective field theory, known as the Standard Model Effective Field Theory (SMEFT) [1,2]. The SMEFT Lagrangian contains the SM Lagrangian plus a complete set of independent higher-dimensional operators. Working in the framework of the SMEFT makes it possible to search for NP through its virtual effects in a general, model-independent way.

Going from the SM to the SMEFT entails dramatic phenomenological consequences, since the SM enjoys several accidental symmetries that are potentially broken by higher-dimensional operators, such as Baryon (B) and Lepton (L) number conservation, or the absence of tree-level Flavour Changing Neutral Currents (FCNC). Phenomenology requires the coefficients of those higher-dimensional operators that violate accidental symmetries of the SM to be tiny, implying in turn that any NP not too far from the electroweak (EW) scale must be at least approximately invariant under the accidental symmetries of the SM. However, while SM interactions will not generate B or L violation perturbatively if the corresponding operators in the SMEFT have vanishing coefficients, FCNC's will always be generated, even if NP is invariant under the full $U(3)^5$ flavour symmetry group of SM gauge interactions, due to the SM Yukawa couplings. Therefore, the flavour properties of the SMEFT Wilson coefficients must be specified at the NP scale, and then the coefficients must be evolved using Renormalization group equations (RGE's) down to the scale relevant for the processes of interest in order to compute the NP contributions. Conversely, a phenomenological bound on low-scale Wilson coefficients can be turned into a bound on the coefficients at the NP scale, allowing to extract information on the viable values of SMEFT coefficients and, hopefully, on the symmetries of the NP models of interest.

Assuming B and L conservation but a general flavour structure, the SMEFT has 2499 independent operators, so that the full system of RGE's involves more than 2500 parameters. At Leading Order (LO), the renormalization group (RG) evolution is dictated by the Anomalous Dimension Matrices (ADM's) of the SMEFT operators, which in general receive

[a] e-mail: stefano.dinoi@phd.unipd.it (corresponding author)
[b] e-mail: Luca.Silvestrini@roma1.infn.it

contributions from scale-dependent gauge and Yukawa couplings, as well as from the Higgs self-coupling. Since the ADM contributions proportional to different couplings are in general non-commuting, an analytic resummation of logarithmic contributions cannot be achieved and the numerical solution of the full system of equations is the only possibility, in particular when the NP scale is much heavier than the EW one. `RGESolver` is an open-source `C++` library that performs the RG evolution of the SMEFT Wilson coefficients in a fast and easy-to-use manner, as detailed below. `RGESolver` will be also integrated in `HEPfit` [3], a flexible open-source tool which, given the Standard Model or any of its extensions, allows to fit the model parameters to a given set of experimental observables and to obtain predictions for observables.

This paper is organized as follows: we briefly introduce the theoretical framework of the SMEFT in Sect. 2, presenting the notation used in `RGESolver`. In Sect. 3 we describe in more detail the structure of the library, with a specific focus on the handling of the flavour structure and on the implementation of the numerical solution of the RGE's. Section 4 is devoted to describing the usage of `RGESolver`: we discuss the installation procedure and we describe how to use the basic functionalities of the library, together with a list of its most important methods. We discuss the efficiency of the library and the comparison with `DSixTools` [4,5] in Sect. 5. We present our conclusions in Sect. 6.

2 Theoretical framework: the SMEFT

As discussed above, the absence of new particles at energies at or above the TeV scale allows us to parametrize the effects of physics beyond the SM with a tower of dimension $\mathfrak{D} > 4$, Lorentz and gauge-invariant operators [1,2]. The resulting effective field theory is the SMEFT. The SM gauge group is

$$\mathcal{G}_{\rm SM} \equiv {\rm SU(3)_C} \otimes {\rm SU(2)_W} \otimes {\rm U(1)_Y}. \tag{1}$$

The three factors represent color, weak isospin and hypercharge gauge group. The quantum numbers of SM fields under $\mathcal{G}_{\rm SM}$ are listed in Table 1.

The Lagrangian of the Standard Model is (following the notation used in Refs. [6–8])

$$\begin{aligned}\mathcal{L}_{\rm SM} = &-\frac{1}{4}G^A_{\mu\nu}G^{A\mu\nu} - \frac{1}{4}W^I_{\mu\nu}W^{I\mu\nu} - \frac{1}{4}B_{\mu\nu}B^{\mu\nu} \\ &+ \sum_\psi \bar\psi i \not{D}\psi + (D_\mu H^\dagger)(D^\mu H) \\ &- \lambda\left(H^\dagger H - \frac{1}{2}v^2\right)^2 \\ &- \left[H^\dagger \bar d Y_d q + \tilde H^\dagger \bar u Y_u q + H^\dagger \bar e Y_e l + {\rm H.c.}\right].\end{aligned} \tag{2}$$

Table 1 $\mathrm{SU(3)_C} \otimes \mathrm{SU(2)_W} \otimes \mathrm{U(1)_Y}$ quantum numbers of SM fields. p is a flavour index

Field	$(\mathbf{R}_{\rm C}, \mathbf{R}_{\rm W})_{\rm y}$
q^p	$(\mathbf{3},\mathbf{2})_{+\frac{1}{6}}$
l^p	$(\mathbf{1},\mathbf{2})_{-\frac{1}{2}}$
u^p	$(\mathbf{3},\mathbf{1})_{+\frac{2}{3}}$
d^p	$(\mathbf{3},\mathbf{1})_{-\frac{1}{3}}$
e^p	$(\mathbf{1},\mathbf{1})_{-1}$
H	$(\mathbf{1},\mathbf{2})_{+\frac{1}{2}}$
G^A_μ	$(\mathbf{8},\mathbf{1})_0$
W^I_μ	$(\mathbf{1},\mathbf{3})_0$
B_μ	$(\mathbf{1},\mathbf{1})_0$

With this notation, when spontaneous symmetry breaking occurs, the physical Higgs boson acquires a mass given by $m_H^2 = 2\lambda v^2$, with $v = (\sqrt{2}G_F)^{-1/2} \sim 246$ GeV. The sum over ψ is extended to all fermion fields in the SM, both left-handed (q, l) and right-handed (u, d, e). The gauge covariant derivative is $D_\mu = \partial_\mu + ig_1 {\rm y} B_\mu + ig_2 t^I W^I_\mu + ig_3 T^A G^A_\mu$, with T^A and $t^I = \tau^I/2$[1] the generators in the fundamental representation of SU(3) and SU(2) respectively:

$$[T^A, T^B] = if^{ABC}T^C, \quad [t^I, t^J] = i\varepsilon^{IJK}t^K. \tag{3}$$

Finally, $\tilde H$ and $\tilde X$ (with $X = B, W^I, G^A$) are defined via the totally antisymmetric Levi-Civita symbol, with $\varepsilon_{12} = 1$ and $\varepsilon_{0123} = 1$:

$$\tilde H_j = \varepsilon_{jk} H^{*k}, \quad \tilde X^{\mu\nu} = \frac{1}{2}\varepsilon^{\mu\nu\rho\sigma}X_{\rho\sigma}. \tag{4}$$

Fermions appear in $n_g = 3$ different generations, so every fermion field carries, together with the gauge indices, a generation (or flavour) index that runs from 1 to 3. We did not write explicitly neither of these indices in (2) for the sake of simplicity. Yukawa couplings Y_ψ are thus complex matrices in flavour space.

At dimension six a complete basis of independent and gauge invariant operators that conserve B and L is given by the so-called *Warsaw basis*, defined in [2] and displayed in Table 2. Their independence means that no linear combination of them and their Hermitian conjugates vanishes due to the equations of motion (up to total derivatives). The equations of motion are used at $\mathcal{O}\left(1/\Lambda^2\right)$ level, so they can be derived from $\mathcal{L}_{\rm SM}$ alone.

The Warsaw basis consists of 59 operators, for a total of 2499 independent parameters in the generic flavour scenario (see Appendix A of Ref. [8]). The operators are divided in

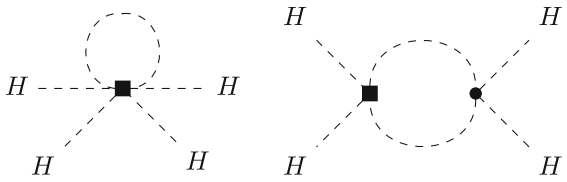
[1] τ_I, $I = 1, 2, 3$, are the Pauli matrices: $\tau^1 = \left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)$, $\tau^2 = \left(\begin{smallmatrix}0&-i\\i&0\end{smallmatrix}\right)$, $\tau^3 = \left(\begin{smallmatrix}1&0\\0&-1\end{smallmatrix}\right)$.

Table 2 The 59 independent dimension-six operators built from SM fields which conserve baryon and lepton number. p, r, s, t are fermion flavour indices, j, k are indices in the fundamental representation of $SU(2)_W$, I, J, K (A, B, C) are indices in the adjoint representation of $SU(2)_W$ $(SU(3)_C)$ and greek letters $(\mu, \nu \ldots)$ are Lorentz indices. Contraction of indices in the fundamental representation of $SU(3)_C$ is implicit

$1 : X^3$	
$\mathcal{O}_G$	$f^{ABC} G^{A\nu}_{\mu} G^{B\rho}_{\nu} G^{C\mu}_{\rho}$
$\mathcal{O}_{\tilde{G}}$	$f^{ABC} \tilde{G}^{A\nu}_{\mu} G^{B\rho}_{\nu} G^{C\mu}_{\rho}$
$\mathcal{O}_W$	$\varepsilon^{IJK} W^{I\nu}_{\mu} W^{J\rho}_{\nu} W^{K\mu}_{\rho}$
$\mathcal{O}_{\tilde{W}}$	$\varepsilon^{IJK} \tilde{W}^{I\nu}_{\mu} W^{J\rho}_{\nu} W^{K\mu}_{\rho}$
$2 : H^6$	
$\mathcal{O}_H$	$(H^\dagger H)^3$
$3 : H^4 D^2$	
$\mathcal{O}_{H\Box}$	$(H^\dagger H)\Box(H^\dagger H)$
$\mathcal{O}_{HD}$	$(H^\dagger D^\mu H)^*(H^\dagger D_\mu H)$
$4 : X^2 H^2$	
$\mathcal{O}_{HG}$	$(H^\dagger H) G^A_{\mu\nu} G^{A\mu\nu}$
$\mathcal{O}_{H\tilde{G}}$	$(H^\dagger H) \tilde{G}^A_{\mu\nu} G^{A\mu\nu}$
$\mathcal{O}_{HW}$	$(H^\dagger H) W^I_{\mu\nu} W^{I\mu\nu}$
$\mathcal{O}_{H\tilde{W}}$	$(H^\dagger H) \tilde{W}^I_{\mu\nu} W^{I\mu\nu}$
$\mathcal{O}_{HB}$	$(H^\dagger H) B_{\mu\nu} B^{\mu\nu}$
$\mathcal{O}_{H\tilde{B}}$	$(H^\dagger H) \tilde{B}_{\mu\nu} B^{\mu\nu}$
$\mathcal{O}_{HWB}$	$(H^\dagger \tau^I H) W^I_{\mu\nu} B^{\mu\nu}$
$\mathcal{O}_{H\tilde{W}B}$	$(H^\dagger \tau^I H) \tilde{W}^I_{\mu\nu} B^{\mu\nu}$
$5 : \psi^2 H^3$	
$\mathcal{O}_{eH}$	$(H^\dagger H)(\bar{l}_p e_r H)$
$\mathcal{O}_{uH}$	$(H^\dagger H)(\bar{q}_p u_r \tilde{H})$
$\mathcal{O}_{dH}$	$(H^\dagger H)(\bar{q}_p d_r H)$
$6 : \psi^2 X H$	
$\mathcal{O}_{eW}$	$(\bar{l}_p \sigma^{\mu\nu} e_r) \tau^I H W^I_{\mu\nu}$
$\mathcal{O}_{eB}$	$(\bar{l}_p \sigma^{\mu\nu} e_r) H B_{\mu\nu}$
$\mathcal{O}_{uG}$	$(\bar{q}_p T^A \sigma^{\mu\nu} u_r) \tilde{H} G^A_{\mu\nu}$
$\mathcal{O}_{uW}$	$(\bar{q}_p \sigma^{\mu\nu} u_r) \tau^I \tilde{H} W^I_{\mu\nu}$
$\mathcal{O}_{uB}$	$(\bar{q}_p \sigma^{\mu\nu} u_r) \tilde{H} B_{\mu\nu}$
$\mathcal{O}_{dG}$	$(\bar{q}_p T^A \sigma^{\mu\nu} d_r) H G^A_{\mu\nu}$
$\mathcal{O}_{dW}$	$(\bar{q}_p \sigma^{\mu\nu} d_r) \tau^I H W^I_{\mu\nu}$
$\mathcal{O}_{dB}$	$(\bar{q}_p \sigma^{\mu\nu} d_r) H B_{\mu\nu}$

Table 2 continued

$7 : \psi^2 H^2 D$	
$\mathcal{O}_{Hl(1)}$	$(H^\dagger i \overleftrightarrow{D}_\mu H)(\bar{l}_p \gamma^\mu l_r)$
$\mathcal{O}_{Hl(3)}$	$(H^\dagger i \overleftrightarrow{D}^I_\mu H)(\bar{l}_p \tau^I \gamma^\mu l_r)$
$\mathcal{O}_{He}$	$(H^\dagger i \overleftrightarrow{D}_\mu H)(\bar{e}_p \gamma^\mu e_r)$
$\mathcal{O}_{Hq(1)}$	$(H^\dagger i \overleftrightarrow{D}_\mu H)(\bar{q}_p \gamma^\mu q_r)$
$\mathcal{O}_{Hq(3)}$	$(H^\dagger i \overleftrightarrow{D}^I_\mu H)(\bar{q}_p \tau^I \gamma^\mu q_r)$
$\mathcal{O}_{Hu}$	$(H^\dagger i \overleftrightarrow{D}_\mu H)(\bar{u}_p \gamma^\mu u_r)$
$\mathcal{O}_{Hd}$	$(H^\dagger i \overleftrightarrow{D}_\mu H)(\bar{d}_p \gamma^\mu d_r)$
$\mathcal{O}_{Hud}$	$(\tilde{H}^\dagger i D_\mu H)(\bar{u}_p \gamma^\mu d_r)$
$8 : (\bar{L}L)(\bar{L}L)$	
$\mathcal{O}_{ll}$	$(\bar{l}_p \gamma_\mu l_r)(\bar{l}_s \gamma^\mu l_t)$
$\mathcal{O}_{qq(1)}$	$(\bar{q}_p \gamma_\mu q_r)(\bar{q}_s \gamma^\mu q_t)$
$\mathcal{O}_{qq(3)}$	$(\bar{q}_p \gamma_\mu \tau^I q_r)(\bar{q}_s \gamma^\mu \tau^I q_t)$
$\mathcal{O}_{lq(1)}$	$(\bar{l}_p \gamma_\mu l_r)(\bar{q}_s \gamma^\mu q_t)$
$\mathcal{O}_{lq(3)}$	$(\bar{l}_p \gamma_\mu \tau^I l_r)(\bar{q}_s \gamma^\mu \tau^I q_t)$
$8 : (\bar{R}R)(\bar{R}R)$	
$\mathcal{O}_{ee}$	$(\bar{e}_p \gamma_\mu e_r)(\bar{e}_s \gamma^\mu e_t)$
$\mathcal{O}_{uu}$	$(\bar{u}_p \gamma_\mu u_r)(\bar{u}_s \gamma^\mu u_t)$
$\mathcal{O}_{dd}$	$(\bar{d}_p \gamma_\mu d_r)(\bar{d}_s \gamma^\mu d_t)$
$\mathcal{O}_{eu}$	$(\bar{e}_p \gamma_\mu e_r)(\bar{u}_s \gamma^\mu u_t)$
$\mathcal{O}_{ed}$	$(\bar{e}_p \gamma_\mu e_r)(\bar{d}_s \gamma^\mu d_t)$
$\mathcal{O}_{ud(1)}$	$(\bar{u}_p \gamma_\mu u_r)(\bar{d}_s \gamma^\mu d_t)$
$\mathcal{O}_{ud(8)}$	$(\bar{u}_p \gamma_\mu T^A u_r)(\bar{d}_s \gamma^\mu T^A d_t)$
$8 : (\bar{L}L)(\bar{R}R)$	
$\mathcal{O}_{le}$	$(\bar{l}_p \gamma_\mu l_r)(\bar{e}_s \gamma^\mu e_t)$
$\mathcal{O}_{lu}$	$(\bar{l}_p \gamma_\mu l_r)(\bar{u}_s \gamma^\mu u_t)$
$\mathcal{O}_{ld}$	$(\bar{l}_p \gamma_\mu l_r)(\bar{d}_s \gamma^\mu d_t)$
$\mathcal{O}_{qe}$	$(\bar{q}_p \gamma_\mu q_r)(\bar{e}_s \gamma^\mu e_t)$
$\mathcal{O}_{qu(1)}$	$(\bar{q}_p \gamma_\mu q_r)(\bar{u}_s \gamma^\mu u_t)$
$\mathcal{O}_{qu(8)}$	$(\bar{q}_p \gamma_\mu T^A q_r)(\bar{u}_s \gamma^\mu T^A u_t)$
$\mathcal{O}_{qd(1)}$	$(\bar{q}_p \gamma_\mu q_r)(\bar{d}_s \gamma^\mu d_t)$
$\mathcal{O}_{qd(8)}$	$(\bar{q}_p \gamma_\mu T^A q_r)(\bar{d}_s \gamma^\mu T^A d_t)$
$8 : (\bar{L}R)(\bar{R}L)$	
$\mathcal{O}_{ledq}$	$(\bar{l}_p e_r)(\bar{d}_s q_t)$
$8 : (\bar{L}R)(\bar{L}R)$	
$\mathcal{O}_{quqd(1)}$	$(\bar{q}^j_p u_r)\varepsilon_{jk}(\bar{q}^k_s d_t)$
$\mathcal{O}_{quqd(8)}$	$(\bar{q}^j_p T^A u_r)\varepsilon_{jk}(\bar{q}^k_s T^A d_t)$
$\mathcal{O}_{lequ(1)}$	$(\bar{l}^j_p e_r)\varepsilon_{jk}(\bar{q}^k_s u_t)$
$\mathcal{O}_{lequ(3)}$	$(\bar{l}^j_p \sigma_{\mu\nu} e_r)\varepsilon_{jk}(\bar{q}^k_s \sigma^{\mu\nu} u_t)$



(a) SMEFT diagram contributing to the β function of λ with terms $\mathcal{O}\left(m_H^2 \mathcal{C}_H\right)$. (b) SMEFT diagram contributing to the β function of λ with terms $\mathcal{O}\left(\lambda m_H^2 \mathcal{C}_{HD}\right)$,$\mathcal{O}\left(\lambda m_H^2 \mathcal{C}_{H\Box}\right)$.

Fig. 1 Two SMEFT diagrams that contribute to the running of the Higgs quartic coupling λ. The solid square denotes a SMEFT vertex and the dot denotes a SM vertex

8 classes, depending on their field content, which schematically are

$$\begin{aligned} &1: X^3, \quad 2: H^6, \quad 3: H^4 D^2, \\ &4: X^2 H^2, \quad 5: \psi^2 H^3, \; 6: \psi^2 H X, \\ &7: \psi^2 H^2 D, \; 8: \psi^4, \end{aligned} \tag{5}$$

where X stands for a gauge field-strength tensor (or its dual), ψ for a fermion field, D for a derivative and H is the Higgs field.

To conclude the discussion about the SMEFT we note that higher dimensional operators contribute to observables not only through their matrix element, but also through their contribution to the running of SM couplings. An example is given by the diagrams in Fig. 1. Coherently with the power counting, their effects in the running of λ are suppressed by a factor of m_H^2/Λ^2. The contributions to the running of the SM parameters due to SMEFT operators are available in Ref. [6].

2.1 Flavour basis and back-rotation

In the SM a unitary transformation for each fermion field in flavour space,

$$\psi = R_\psi^\dagger \psi', \qquad \psi = q, l, u, d, e, \tag{6}$$

does not alter the structure of the Lagrangian, provided the Yukawa matrices are redefined as

$$\begin{cases} Y_u = R_u^\dagger Y_u' R_q, \\ Y_d = R_d^\dagger Y_d' R_q, \\ Y_e = R_e^\dagger Y_e' R_l. \end{cases} \tag{7}$$

This freedom allows to diagonalize simultaneously two out of the three matrices (Y_e and one between Y_u and Y_d).[2]

[2] The situation changes after the spontaneous symmetry breaking in the SM: fermions acquire a mass matrix proportional to the corresponding Yukawa matrix, $m_\psi^{pr} = v Y_\psi^{pr}/\sqrt{2}, \psi = u, d, e$. All the mass matrices can be simultaneously diagonalized because the up and down component of the iso-doublet q can be rotated indipendently.

Denoting with $\hat{Y}$ a diagonal Yukawa matrix we can define two reference bases: the *down basis* where $Y_e = \hat{Y}_e$, $Y_d = \hat{Y}_d$, $Y_u = \hat{Y}_u V$ and the *up basis* where $Y_e = \hat{Y}_e$, $Y_u = \hat{Y}_u$, $Y_d = \hat{Y}_d V^\dagger$, with V a unitary matrix. In the SM, V is the Cabibbo–Kobayashi–Maskawa matrix. In the SMEFT this is not true, since the mass matrices are not simply proportional to the Yukawa couplings due to the contributions from dimension-six operators. With a slight abuse of notation, in the following we still refer to the matrix V as the CKM matrix. It is worth noticing that the matrix V is not the one appearing in the interaction vertices between quarks and $W^\pm$ bosons, again due to effects from higher dimensional operators. A more detailed discussion is available in Section 2 of Ref. [9].

Notice that the up and down bases are not stable under renormalization group evolution, even in the SM. For example, the SM β function for Y_d (available in Ref. [10]) contains a term proportional to $Y_d(Y_d^\dagger Y_d - Y_u^\dagger Y_u)$, which is always non-diagonal in flavour space. Starting from a diagonal Y_d at the scale μ_I (down basis) leads to a non-diagonal Y_d at a different scale μ_F. To go back into the down basis, a further flavour rotation is thus required. Clearly, the same holds for the up basis.

In the SMEFT the flavour transformations in Eq. (6) imply not only a redefinition of the Yukawa couplings, but also a rotation of the Wilson coefficients. For example, the coefficient $\mathcal{C}_{dH}$ of operator $\mathcal{O}_{dH}$ is redefined as $\mathcal{C}_{dH} = R_q^\dagger \mathcal{C}'_{dH} R_d$. It should be noticed that the flavour rotations are not uniquely determined by the singular value decomposition. The three Yukawa matrices are diagonalized via six unitary matrices:

$$Y_\psi = U_\psi \hat{Y}_\psi V_\psi^\dagger, \quad \psi = u, d, e. \tag{8}$$

If U_ψ, V_ψ satisfy the relation (8), also $U_\psi \phi_\psi$, $V_\psi \phi_\psi$ with $\phi_\psi = \mathrm{diag}\left(e^{\alpha_1^\psi}, e^{\alpha_2^\psi}, e^{\alpha_3^\psi}\right)$ satisfy it. The phase matrices ϕ_u, ϕ_d are determined unambiguosly once the phase convention for the CKM matrix is chosen, while, in the absence of right-handed neutrinos, the phase of the lepton matrices cannot be determined. A possibility to fix the phases would be using one of the SMEFT coefficients that involve leptons, but this would not be a general solution. In fact, the user may want to set that coefficient to 0, frustrating the choice of phase convention. To overcome this problem, we choose $\phi_e = \mathrm{diag}\left(e^{-i\arg[(U_e)_{11}]}, e^{-i\arg[(U_e)_{22}]}, e^{-i\arg[(U_e)_{33}]}\right)$. This choice ensures that an evolution between two energy scales close to each other produces a small change in the coefficients. The user is then free to perform the rotation in the lepton sector choosing any other convention.

To go into the up basis, the rotation matrix R_q in (6) must be set equal to $V_u \phi_u$. To go into the down basis one must instead set $R_q = V_d \phi_d$. Obviously, in both bases one sets $R_u = U_u \phi_u$, $R_d = U_d \phi_u$, $R_l = V_e \phi_e$ and $R_e = U_e \phi_e$.

2.2 Solution of the renormalization group equations

The RGE for a parameter x_i, that in this context is a SMEFT coefficient $\mathcal{C}_i$ or a SM parameter, is

$$\mu \frac{dx_i}{d\mu} \equiv \beta_i \left(\{x_j\}\right), \tag{9}$$

that defines the β-function for the parameter. In general, when the theory has multiple parameters, the β-function for x_i can contain terms proportional to integer powers of x_i, but also terms proportional to other parameters x_j, $i \neq j$. In this article we always consider the β-functions at one-loop level.

The β-functions of SMEFT coefficients (as well as SMEFT contributions to the running of SM parameters) are available in Refs. [6–8], the SM β-functions of gauge couplings can be found for example in Ref. [11] and the β-functions of Yukawa couplings, m_H^2 and λ in Ref. [10].[3]

The β-function for SMEFT coefficients must be linear in the coefficients themselves by power counting. In fact, the product of two dimension-six SMEFT coefficients would be suppressed by a factor of $1/\Lambda^4$, negligible at $\mathcal{O}\left(m_H^2/\Lambda^2\right)$. This allows to write the RGE's for the $\mathcal{C}_i$s introducing the *anomalous dimension matrix* Γ (ADM),

$$\mu \frac{d\mathcal{C}_i}{d\mu} = \frac{1}{16\pi^2}\Gamma_{ij}\mathcal{C}_j. \tag{10}$$

The indices i, j in the previous equation run on the whole set of independent coefficients: Γ is a square 2499×2499 matrix. Clearly, the β-function for the i-th coefficient is $\beta_{\mathcal{C}_i} = \Gamma_{ij}\mathcal{C}_j/(16\pi^2)$.

A simple approximation consists in neglecting the μ dependence of the right-hand side of Eq. (10) and taking only linear order terms in $\ln(\mu_F/\mu_I)$, yielding

$$\mathcal{C}_i(\mu_F) = \mathcal{C}_i(\mu_I) + \Gamma_{ij}(\mu_I)\mathcal{C}_j(\mu_I)\frac{\ln(\mu_F/\mu_I)}{16\pi^2}. \tag{11}$$

While this drastic simplification may be appropriate for cases in which the two energy scales are not too different, so that the second term in Eq. (11) is a small correction, in general the log on the right-hand side will become large, calling for a resummation of log-enhanced terms.

To provide a more accurate solution of the system of first-order differential equations in Eq. (10) is a non-trivial task since the anomalous dimension matrix depends on μ through the SM couplings, the running of which is in turn influenced by SMEFT operators. The anomalous dimension matrix can be written as

$$\Gamma_{ij}(\mu) = g_1^2(\mu)\Gamma_{ij}^{(g_1^2)} + g_2^2(\mu)\Gamma_{ij}^{(g_2^2)} + \cdots, \tag{12}$$

[3] This reference uses a different convention for the Higgs sector and for g_1. The Higgs mass is $m_H^2 = -m^2/2$ and the replacements $g_1 \to g_1\sqrt{5/3}$, $\lambda \to 2\lambda$ are required to convert their results into our convention.

Table 3 Symmetry categories for operators in the SMEFT. nF indicates the number of flavour indices for each category

Category	Symmetry properties
0	0F scalar object
1	2F generic real matrix
2R	2F Hermitian matrix (Re)
2I	2F Hermitian matrix (Im)
5	4F generic real object
6R	4F two identical $\bar{\psi}\psi$ currents (Re)
6I	4F two identical $\bar{\psi}\psi$ currents (Im)
7R	4F two independent $\bar{\psi}\psi$ currents (Re)
7I	4F two independent $\bar{\psi}\psi$ currents (Im)
8R	$\mathcal{C}_{ee}$ (Re)
8I	$\mathcal{C}_{ee}$ (Im)

where the matrices on the right-hand side do not depend on the renormalization scale. In general the $\Gamma^{(g_i)}$ matrices cannot be diagonalized simultaneously: an exact analytic solution of the system is then impossibile. One could proceed with an hybrid scheme as commonly done in the Effective Theory for Weak interactions (WET) below the EW scale, where QCD and QED corrections are implemented by resumming the logs generated by the strong coupling only. However, in the SMEFT the top Yukawa coupling is also large, calling for the resummation of both strong and Yukawa contributions.

Thus, a numerical approach is required to obtain precise results in the general case.[4] More details about the implementation of this method are given in Sect. 3.2.

3 Description of the library

`RGESolver` is implemented in `C++`. We use the GNU Scientific Library for the integration of the RGE's (more information about the numerical methods used are given below). The whole code (including input and output) handles separately real and imaginary parts of the parameters.

3.1 Flavour structure

Operators (or equivalently their coefficients) in the SMEFT can be divided in different categories according to their symmetry properties as in Table 3,[5] where we partially modify the notation used in Table 14 of Ref. [5].

[4] An approximate resummation of strong and Yukawa contributions might be obtained by assuming the y_t^2/α_s ratio to be scale-invariant, as suggested in Ref. [12].

[5] It is worth noticing that there exists a symmetry class specific for the operator $\mathcal{O}_{ee}$, since it has an additional symmetry following from e^p being a singlet under $SU(3)_C \otimes SU(2)_W$ (see Ref. [8]).

We work splitting real and imaginary part for each $\mathcal{C}_i$. For example, the real part of the coefficient of an Hermitian 2F operator is a (real) symmetric matrix, while the imaginary part is a (real) antisymmetric matrix. Also the real and imaginary part of symmetric 4F operators are in different categories. This is not the case for non-Hermitian operators: the real and imaginary parts of their coefficient do not have any constraints, and they belong to categories 1 (2F) and 5 (4F). Clearly it is convenient to store and evolve only the independent parameters, but for ease of use it is convenient to be able to access the coefficients with any combination of indices, not only the independent ones. We achieve this flexibility by defining specific symmetry-aware getters and setters for each category. For example, setting $\mathrm{Re}[\mathcal{C}^{1,2}_{Hl(1)}] = 0.5$ via `S.SetCoefficient("CHl1R",0.5,1,2)` (where `S` is an instance of the `RGESolver` class) will lead to `S.GetCoefficient("CHl1R",2,1)` returning the value `0.5` due to Hermiticity.

3.2 Numerical implementation of renormalization group evolution

As already mentioned, we use the routines provided by the GNU Scientific Library (`GSL`, [13]) to perform the integration of the RGEs. Let $\vec{y}$ be the vector that contains all the independent parameters of the SMEFT. Introducing $t = \ln(\mu/\mathrm{GeV})$, the system in Eq. (9) can be rewritten as

$$\frac{d\vec{y}}{dt} = \vec{\mathrm{f}}(\vec{y}), \tag{13}$$

where the right-hand side of the equation depends on t only through $\vec{y}$. When all the symmetries of the operators are considered, the coefficients of operators at dimension-six level are completely determined through 2499 independent real parameters. The three gauge couplings g_1, g_2, g_3, the Higgs quartic coupling λ and the mass m_H of the Higgs boson are real scalars. Yukawa couplings Y_u, Y_d and Y_e are 3×3 complex matrices, each of which has 18 parameters.[6] This yelds 59 further parameters, raising the total number to 2558: this is the dimension of $\vec{y}$. The system in Eq. (9) is thus a 2558-dimensional system of first order coupled differential equations. We use an adaptive stepsize routine to solve the system: the stepsize h is changed throughout the integration in order to match the estimated local error E_i with a user-defined error level, to obtain the maximum efficiency. The desired error level D_i for each component is determined through four parameters a_y, a_{dydt}, ϵ_{abs} and ϵ_{rel} with the expression

$$D_i = \epsilon_{\mathrm{abs}} + \epsilon_{\mathrm{rel}} \left(a_y |y_i| + h a_{dydt} \left| \frac{dy_i}{dt} \right| \right) \tag{14}$$

The second term in brackets in the previous expression ensures a correct estimation also for situations where some components y_i are very close to zero. The error is determined not only from the value y_i but also from its increment. In particular, we use $a_y = a_{dydt} = 1$.

E_i can be estimated with the *step-doubling* technique, that consists in advancing the solution from t to $t + 2h$ in two different ways (performing two steps of length h or one step of length $2h$) and taking the error as the difference between the two. A more sofisticated and efficient error estimate is possible for the *embedded* integration methods, where the same evaluations of the function $\vec{\mathrm{f}}$ are used to compute two different values of the solution at $t + h$. The error is taken as the difference between the two values (see Section 16.2 of Ref. [14]).

If E_i exceeds the desired error D_i by more than 10% for any component, the stepsize is reduced according to

$$h \to h \times 0.9 \times \left(\frac{E}{D} \right)^{-\frac{1}{q}}, \tag{15}$$

where 0.9 is a safety factor (the error can only be estimated, not accurately determined), q is the consistency order of the method (i.e. the local error scales as h^{q+1}) and $E/D = \max_i (E_i/D_i)$ is the maximum ratio of estimated and desired error among the components.

If, instead, the estimated error is lower than the desired one (precisely, when $E/D < 50\%$) the stepsize is increased according to

$$h \to h \times 0.9 \times \left(\frac{E}{D} \right)^{-\frac{1}{q+1}}. \tag{16}$$

To avoid uncontrolled changes in the stepsize, the overall scaling factor is limited to the range (1/5, 5). In `RGESolver` the explicit embedded Runge–Kutta–Fehlberg method is used (with initial stepsize $h = \ln(\mu/\Lambda)/100$), for which $q = 4$.

It should be stressed that Eq. (14) refers to a local error: there is no simple relation that connects the four parameters to an estimate of the global error affecting the final result. We tested the accuracy of the adaptive stepsize integration comparing it with a fixed-step integration with an high number of steps. Using $\mu_{\mathrm{I}} = \Lambda = 1000$ TeV and $\mu_{\mathrm{F}} = 250$ GeV as energy scales, $\epsilon_{\mathrm{abs}} = 10^{-16}$ and $\epsilon_{\mathrm{rel}} = 10^{-4}$ for the adaptive integration and $N_{\mathrm{FS}} = 1000$ steps for the fixed stepsize integration we obtained percentual differences $\lesssim 10^{-5}$. The initial conditions for SM parameters at the high energy scale are obtained evolving them from $\mu \sim v$ to Λ with the pure SM β-functions (namely neglecting SMEFT contributions).

[6] In the Yukawa sector not all of the 54 real parameters are observable: only 13 parameters from this sector are independent (9 fermion masses plus the 3 angles and one phase in the CKM matrix). However, above the electroweak scale the β functions are naturally expressed in terms of the Yukawa matrices rather than of the fermion masses and the CKM matrix. Therefore, for the purpose of computing RG evolution it is more convenient to consider general Yukawa matrices.

The initial conditions for SMEFT coefficients are $\mathcal{O}\left(1/\Lambda^2\right)$, as prescribed by the power counting.

4 Using RGESolver

4.1 Installation

RGESolver is a free software released under the GNU General Public License. The download can be performed directly from the command line, typing in the terminal:

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

More details can be found on the dedicated GitHub webpage. The extended documentation is also available here

Dependencies

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. We list below the dependencies that need to be satisfied to succesfully install RGESolver:

- GSL: The GNU Scientific Library (GSL) is a C library for numerical computations. More details can be found on the GSL website.
- BOOST: BOOST is a set of libraries for the C++ programming language. RGESolver requires only the BOOST headers, not the full libraries, thus a header-only installation is sufficient. More details can be found on the BOOST website.
- C++11: a C++ compiler supporting at least the C++11 standard.

If all dependencies are satisfied, the installation procedure is completed typing in a terminal session in the downloaded RGESolver directory:

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

The available options are:

- -DLOCAL_INSTALL=ON: to install RGESolver in the directory build/install (default: OFF).
- -DCMAKE_INSTALL_PREFIX=<RGESolver installation directory>: the directory in which RGESolver will be installed (default: /usr/local). This variable cannot be modified when -DLOCAL_INSTALL=ON is set.
- -DDEBUG_MODE=ON: to enable the debug mode (default: OFF).
- -DBOOST_INCLUDE_DIR=<boost custom 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 to gsl-config>: 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.

If no <options> are specified, the default installation will be performed. Note that, depending on the setting of installation prefix, root privileges may be required (thus cmake --install . should be replaced with sudo cmake --install .). RGESolver can be uninstalled by typing in the build directory of the RGESolver library the command (sudo) cmake --build . --target uninstall.

4.2 Class methods

Here we briefly describe the most important methods available in RGESolver. The detailed documentation is available on GitHub.

Evolution

- void Evolve(std::string method, double muI, double muF): performs the one-loop renormalization group evolution from $\mu =$ muI to $\mu =$ muF (where muI and muF are expressed in GeV). The current values of the SMEFT parameters are taken as initial condition at $\mu =$ muI. After the evolution, they are updated with the new values at $\mu =$ muF. The available methods for the solution of the RGE's are "Approximate" and "Numeric". The first method is faster than the latter, but less accurate, as explained in Sect. 2.2.
- void EvolveSMOnly(std::string method, double muI, double muF): same as Evolve, but only for the SM parameters. The user should use this method instead of Evolve when interested in pure SM running. Using this function is the same of using Evolve with all the SMEFT coefficients set to 0, but it

Table 4 SM parameters used by default to generate SM initial conditions at an arbitrary scale. The scale at which these parameters are given is $\mu = 173.6$ GeV. We follow UTfit for what concerns the conventions for the CKM matrix

Parameter	Value
g_1	0.3573
g_2	0.6511
g_3	1.161
λ	0.1297
m_h^2 [GeV^2]	15650
$\sin\theta_{12}$	0.2252
$\sin\theta_{13}$	0.003675
$\sin\theta_{23}$	0.0420
δ [rad]	1.1676
m_u [GeV]	0.0012
m_c [GeV]	0.640
m_t [GeV]	162.0
m_d [GeV]	0.0027
m_s [GeV]	0.052
m_b [GeV]	2.75
m_e [GeV]	0.000511
m_μ [GeV]	0.1057
m_τ [GeV]	1.776

is faster since it computes only the evolution for the SM parameters.

- `void EvolveToBasis(std::string method, double muI, double muF, std::string basis)`: same as `Evolve`, but performs in addition the flavour back-rotation described in Sect. 2.1 to go into the selected basis (`"UP"` or `"DOWN"`). After the evolution, the CKM matrix is computed.
- `void GenerateSMInitialConditions(...)`: generates the initial conditions for Standard Model parameters ($g_1, g_2, g_3, \lambda, m_h^2, Y_u, Y_d, Y_e$) at the scale `muFin` (in GeV), using one-loop pure SM beta functions. At such scale, the CKM matrix is computed.
  The generation can be done starting from user-defined low-energy input or using the default values summarized in Table 4 (the arguments to be passed to the function are different in the two cases, see the documentation for the details and Sect. 4.3 for an example). In case of user-defined input, 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).

Input and output

Getters and setters take as first argument a string (`name`), that identifies the corresponding parameter. The names that must be used to correctly invoke these functions are reported in Tables 5, 6, 7 and 8. The functions for non-scalar coefficients take as additional arguments the flavour indices (2 or 4), that must be in the interval `[0:2]`.

Table 5 Standard Model parameters in `RGESolver`. The last 3 labels of this table must be used with the `GetCKMAngle` method. All the others must be used with the `GetCoefficient` method

Coefficient	Name
g_1	`g1`
g_2	`g2`
g_3	`g3`
λ	`lambda`
m_h^2 [GeV^2]	`mh2`
Re($\mathcal{Y}_u$)	`YuR`
Im($\mathcal{Y}_u$)	`YuI`
Re($\mathcal{Y}_d$)	`YdR`
Im($\mathcal{Y}_d$)	`YdI`
Re($\mathcal{Y}_e$)	`YeR`
Im($\mathcal{Y}_e$)	`YeI`
$\sin\theta_{12}$	`s12`
$\sin\theta_{13}$	`s13`
$\sin\theta_{23}$	`s23`

Table 6 SMEFT coefficients without flavour indices in `RGESolver`

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

- `void SetCoefficient(std::string name, double val)`: setter function for the scalar parameters. Sets the parameter `name` equal to `val`.
- `double GetCoefficient(std::string name)`: getter function for the scalar parameters. Returns the parameter `name`.

Table 7 SMEFT coefficients with two flavour indices in `RGESolver`

Coeff.	Name	Sym.
Class 5		
$\mathrm{Re}(C_{eH})$	`CeHR`	WC1
$\mathrm{Im}(C_{eH})$	`CeHI`	WC1
$\mathrm{Re}(C_{uH})$	`CuHR`	WC1
$\mathrm{Im}(C_{uH})$	`CuHI`	WC1
$\mathrm{Re}(C_{dH})$	`CdHR`	WC1
$\mathrm{Im}(C_{dH})$	`CdHI`	WC1
Class 6		
$\mathrm{Re}(C_{eW})$	`CeWR`	WC1
$\mathrm{Im}(C_{eW})$	`CeWI`	WC1
$\mathrm{Re}(C_{eB})$	`CeBR`	WC1
$\mathrm{Im}(C_{eB})$	`CeBI`	WC1
$\mathrm{Re}(C_{uG})$	`CuGR`	WC1
$\mathrm{Im}(C_{uG})$	`CuGI`	WC1
$\mathrm{Re}(C_{uW})$	`CuWR`	WC1
$\mathrm{Im}(C_{uW})$	`CuWI`	WC1
$\mathrm{Re}(C_{uB})$	`CuBR`	WC1
$\mathrm{Im}(C_{uB})$	`CuBI`	WC1
$\mathrm{Re}(C_{dG})$	`CdGR`	WC1
$\mathrm{Im}(C_{dG})$	`CdGI`	WC1
$\mathrm{Re}(C_{dW})$	`CdWR`	WC1
$\mathrm{Im}(C_{dW})$	`CdWI`	WC1
$\mathrm{Re}(C_{dB})$	`CdBR`	WC1
$\mathrm{Im}(C_{dB})$	`CdBI`	WC1
Class 7		
$\mathrm{Re}(C_{Hl(1)})$	`CHl1R`	WC2R
$\mathrm{Im}(C_{Hl(1)})$	`CHl1I`	WC2I
$\mathrm{Re}(C_{Hl(3)})$	`CHl3R`	WC2R
$\mathrm{Im}(C_{Hl(3)})$	`CHl3I`	WC2I
$\mathrm{Re}(C_{He})$	`CHeR`	WC2R
$\mathrm{Im}(C_{He})$	`CHeI`	WC2I
$\mathrm{Re}(C_{Hq(1)})$	`CHq1R`	WC2R
$\mathrm{Im}(C_{Hq(1)})$	`CHq1I`	WC2I
$\mathrm{Re}(C_{Hq(3)})$	`CHq3R`	WC2R
$\mathrm{Im}(C_{Hq(3)})$	`CHq3I`	WC2I
$\mathrm{Re}(C_{Hu})$	`CHuR`	WC2R
$\mathrm{Im}(C_{Hu})$	`CHuI`	WC2I
$\mathrm{Re}(C_{Hd})$	`CHdR`	WC2R
$\mathrm{Im}(C_{Hd})$	`CHdI`	WC2I
$\mathrm{Re}(C_{Hud})$	`CHudR`	WC1
$\mathrm{Im}(C_{Hud})$	`CHudI`	WC1

- `void SetCoefficient(std::string name, double val, int i, int j)`: setter function for the parameters with two flavour indices. Sets the parameter `name[i,j]` equal to `val`.

Table 8 SMEFT coefficients with four flavour indices in `RGESolver`

Coeff.	Name	Sym.
Class 8 $(\bar{L}L)(\bar{L}L)$		
$\mathrm{Re}(C_{ll})$	`CllR`	WC6R
$\mathrm{Im}(C_{ll})$	`CllI`	WC6I
$\mathrm{Re}(C_{qq(1)})$	`Cqq1R`	WC6R
$\mathrm{Im}(C_{qq(1)})$	`Cqq1I`	WC6I
$\mathrm{Re}(C_{qq(3)})$	`Cqq3R`	WC6R
$\mathrm{Im}(C_{qq(3)})$	`Cqq3I`	WC6I
$\mathrm{Re}(C_{lq(1)})$	`Clq1R`	WC7R
$\mathrm{Im}(C_{lq(1)})$	`Clq1I`	WC7I
$\mathrm{Re}(C_{lq(3)})$	`Clq3R`	WC7R
$\mathrm{Im}(C_{lq(3)})$	`Clq3I`	WC7I
Class 8 $(\bar{L}R)(\bar{L}R)$		
$\mathrm{Re}(C_{quqd(1)})$	`Cquqd1R`	WC5
$\mathrm{Im}(C_{quqd(1)})$	`Cquqd1I`	WC5
$\mathrm{Re}(C_{quqd(8)})$	`Cquqd8R`	WC5
$\mathrm{Im}(C_{quqd(8)})$	`Cquqd8I`	WC5
$\mathrm{Re}(C_{lequ(1)})$	`Clequ1R`	WC5
$\mathrm{Im}(C_{lequ(1)})$	`Clequ1I`	WC5
$\mathrm{Re}(C_{lequ(3)})$	`Clequ3R`	WC5
$\mathrm{Im}(C_{lequ(3)})$	`Clequ3I`	WC5
Class 8 $(\bar{R}R)(\bar{R}R)$		
$\mathrm{Re}(C_{ee})$	`CeeR`	WC8R
$\mathrm{Im}(C_{ee})$	`CeeI`	WC8I
$\mathrm{Re}(C_{uu})$	`CuuR`	WC6R
$\mathrm{Im}(C_{uu})$	`CuuI`	WC6I
$\mathrm{Re}(C_{dd})$	`CddR`	WC6R
$\mathrm{Im}(C_{dd})$	`CddI`	WC6I
$\mathrm{Re}(C_{eu})$	`CeuR`	WC7R
$\mathrm{Im}(C_{eu})$	`CeuI`	WC7I
$\mathrm{Re}(C_{ed})$	`CedR`	WC7R
$\mathrm{Im}(C_{ed})$	`CedI`	WC7I
$\mathrm{Re}(C_{ud(1)})$	`Cud1R`	WC7R
$\mathrm{Im}(C_{ud(1)})$	`Cud1I`	WC7I
$\mathrm{Re}(C_{ud(8)})$	`Cud8R`	WC7R
$\mathrm{Im}(C_{ud(8)})$	`Cud8I`	WC7I
Class 8 $(\bar{L}R)(\bar{R}L)$		
$\mathrm{Re}(C_{ledq})$	`CledqR`	WC5
$\mathrm{Im}(C_{ledq})$	`CledqI`	WC5
Class 8 $(\bar{L}L)(\bar{R}R)$		
$\mathrm{Re}(C_{le})$	`CleR`	WC7R
$\mathrm{Im}(C_{le})$	`CleI`	WC7I
$\mathrm{Re}(C_{lu})$	`CluR`	WC7R
$\mathrm{Im}(C_{lu})$	`CluI`	WC7I
$\mathrm{Re}(C_{ld})$	`CldR`	WC7R
$\mathrm{Im}(C_{ld})$	`CldI`	WC7I
$\mathrm{Re}(C_{qe})$	`CqeR`	WC7R
$\mathrm{Im}(C_{qe})$	`CqeI`	WC7I

Table 8 continued

Coeff.	Name	Sym.
$\mathrm{Re}(C_{qu(1)})$	`Cqu1R`	WC7R
$\mathrm{Im}(C_{qu(1)})$	`Cqu1I`	WC7I
$\mathrm{Re}(C_{qu(8)})$	`Cqu8R`	WC7R
$\mathrm{Im}(C_{qu(8)})$	`Cqu8I`	WC7I
$\mathrm{Re}(C_{qd(1)})$	`Cqd1R`	WC7R
$\mathrm{Im}(C_{qd(1)})$	`Cqd1I`	WC7I
$\mathrm{Re}(C_{qd(8)})$	`Cqd8R`	WC7R
$\mathrm{Im}(C_{qd(8)})$	`Cqd8I`	WC7I

- `double GetCoefficient(std::string name, int i, int j)`: getter function for the parameters with two flavour indices. Returns the parameter `name[i,j]`.
- `void SetCoefficient(std::string name, double val, int i, int j, int k, int l)`: setter function for the parameters with four flavour indices. Sets the parameter `name[i,j,k,l]` equal to `val`.
- `double GetCoefficient(std::string name, int i, int j, int k, int l)`: getter function for the parameters with four flavour indices. Returns the parameter `name[i,j,k,l]`.
- `double GetCKMAngle(std::string name)`, `double GetCKMPhase()`: getter methods to access CKM matrix parameters. These methods should be called only after methods that choose a specific flavour basis (as `GenerateSM` `InitialConditions()` or `EvolveToBasis()`), in which case the CKM matrix is updated.
- `double GetCKMRealPart(int i, int j)`, `double GetCKMImagPart(int i, int j)`: getter functions for the real and imaginary part of the (i, j) element of the CKM matrix. These methods should be called only after methods that choose a specific flavour basis (as `GenerateSM` `InitialConditions()` or `EvolveToBasis()`), in which case the CKM matrix is updated. The indices must be in the range `[0:2]`.
- `void SaveOutputFile(std::string filename, std::string format)`: saves the current values of the SMEFT parameters in the file `filename`. Currently, the only available format is Susy Les Houches Accord `"SLHA"` ([15]).

Numerical integration parameters

Here we list the methods related to the parameters that control the numerical evolution, as described in Sect. 3.2:

- `void Setepsrel(double epsrel)`: sets ϵ_{rel} equal to `epsrel` (the default value is $\epsilon_{\mathrm{rel}} = 5 \times 10^{-3}$).
- `double epsrel()`: returns ϵ_{rel}.
- `void Setepsabs(double epsabs)`: sets ϵ_{abs} equal to `epsabs` (the default value is $\epsilon_{\mathrm{abs}} = 10^{-13}$).
- `double epsabs()`: returns ϵ_{abs}.

General methods

- `RGESolver()`: the default constructor.
- `~RGESolver()`: the default destructor.
- `Reset()`: resets all the SMEFT coefficients to 0 and the SM parameters to their default value (in the up basis). ϵ_{abs} and ϵ_{rel} are reset to their default value. This function should be called after the evolution, if the same instance of the class is used for another run.

4.3 Writing and compiling a program using the library

We discuss here the usage of the main methods of the class, in order to make the reader acquainted with `RGESolver`'s functionalities. The examples discussed in this section can be found in the `Examples` directory repository. Let us start with the simplest case: the renormalization group evolution from an high-energy scale $\Lambda = 10{,}000$ GeV to $\mu_{\mathrm{Low}} = 250$ GeV with a single non-zero SMEFT coefficient at the starting scale, namely $\mathcal{C}_{HG}$. We do not report the whole program here, but we discuss just the crucial steps. This program is available under the name `ExampleEvolution.cpp`. The library must be included with

```
#include "RGESolver.h"
```

Inside the `main`, an instance of the class must be created:

```
RGESolver S;
```

We define the two energy scales (given in GeV) and we set the initial condition.

```
double Lambda = 10000.;
double muLow = 250.;
S.SetCoefficient("CHG",
1./(Lambda*Lambda));
```

The (numeric) evolution is then performed using the line

```
S.Evolve("Numeric", Lambda, muLow);
```

At this point, the user can access the evolved coefficients via the `GetCoefficient()` methods (see Tables 5, 6, 7

and 8 for a list of all the names to be used). For example, this line prints on the terminal the evolved value of $\mathcal{C}_{HG}$.

```
std::cout<<"CHG("<<muLow<<"GeV):"
<<S.GetCoefficient("CHG")<<std::endl;
```

The user should compile this program typing in the terminal[7]

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

If the `RGESolver` library is not in the predefined search path (as usually is the case for local installation), it may be necessary to specify the path needed for compilation and linking against `RGESolver`. A `rgesolver-config` script is available in the `<CMAKE_INSTALL_PREFIX>/bin` directory, which can be invoked with the following options:

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

A more advanced example can be found in `ExampleBackRotation.cpp`. The goal is to perform the evolution from $\Lambda = 10{,}000$ GeV to $\mu_{\text{Low}} = 250$ GeV and perform the back-rotation. The default initial conditions for the SM parameters in Table 4 are given at the scale $\mu = 173.65$ GeV. It is possible to generate the initial conditions for the SM parameters at any scale, both evolving the default input or using a different set of values.
We thus define our custom input, starting from the energy scale at which the input is given:

```
double SMinputScale = 91.0;
```

We then define the masses of the fermions (in GeV), the CKM parameters (the CKM phase must be expressed in radians), the gauge couplings, the quartic coupling and the Higgs' mass squared (in GeV^2):

```
double Muin[3] = {0.002, 1.2, 170.};
double Mdin[3] = {0.006, 0.05, 5.2};
double Mein[3] = {0.0005, 0.1, 1.2};
double s12in = 0.225;
double s13in = 0.003675;
double s23in = 0.042;
double deltain = 1.17;
double g1in = .35;
double g2in = .65;
double g3in = 1.2;
double lambdain = 0.14;
double mh2in = 15625.;
```

The initial conditions at the scale Λ are thus generated in the down basis through the solution of the pure SM RGE's (via numeric integration) simply using:

```
S.GenerateSMInitialConditions(
SMinputScale,Lambda,"DOWN","Numeric",
g1in,g2in,g3in,lambdain,mh2in,
Muin,Mdin,Mein,
s12in,s13in,s23in,deltain);
```

As anticipated, the user can also take advantage of the default low-energy input for the SM parameters via:

```
S.GenerateSMInitialConditions(
Lambda,"DOWN","Numeric");
```

Having generated the initial conditions for the SM parameters, the user can thus set the values of the SMEFT coefficients at the scale Λ and perform the evolution to μ_{Low}, as done before. We don't discuss this since the syntax is the same of the previous case.

The evolution and the back-rotation (in the down basis) are performed via:

```
S.EvolveToBasis(
"Numeric",Lambda,muLow,"DOWN");
```

Since this method computes also the CKM matrix, we can access its parameters or its elements using the dedicated getter functions. For example, the following lines print on-screen $\sin\theta_{12}$ at the scale `muLow`:

```
std::cout<<"Sin(theta12)("<<
muLow<< " GeV): "
<<S.GetCKMAngle("s12")<< std::endl;
```

5 Execution times and comparison with `DSixTools`

We tested our code against `DSixTools` ([4,5]), a `Mathematica` package for the handling of the SMEFT (and

[7] Mac users might need to add the flag `-std=c++11`: (or greater).

Table 9 Comparison between DSixTools and RGESolver. The first (last) three values refer to the approximate solution (numeric solution)

$\Delta^{\max}$	Λ (TeV)
4.8×10^{-6}	1
4.8×10^{-6}	10
5.1×10^{-6}	100
1.2×10^{-5}	1
1.2×10^{-5}	10
4.2×10^{-5}	100

the Low-energy Effective Field Theory, but we focus on the SMEFT part of the package). DSixTools solves the SMEFT RGE's numerically and via the approximation described in Sect. 2.2, as RGESolver does. We performed an evolution from Λ to $\mu_{\text{Low}} = 250$ GeV with both codes to compare the values of the coefficients after the evolution. The initial conditions at the high energy scale Λ for the SM parameters are obtained using pure SM RGE's to run them up to such scale. We set $\mathcal{O}\left(1/\Lambda^2\right)$ initial conditions for every SMEFT coefficient at $\mu = \Lambda$.[8] The result of this comparison is displayed in Table 9.

To compare the two results, we introduce $\Delta^{\max} = \max|\mathcal{C}_i^{\text{R}}(\mu_{\text{Low}}) - \mathcal{C}_i^{\text{D}}(\mu_{\text{Low}})|/(1/\Lambda^2)$, with $\mathcal{C}_i^{\text{R(D)}}(\mu_{\text{Low}})$ the evolved coefficient computed by RGESolver (DSixTools).

We observe that the two codes produce the same results up to $\mathcal{O}\left(10^{-5}/\Lambda^2\right)$ ($\mathcal{O}\left(10^{-6}/\Lambda^2\right)$) in the case of numeric solution (approximate solution). Clearly, the agreement is better for the simpler solution, namely the approximate one.

Let us briefly discuss RGESolver's efficiency performances. We report in Table 10 the execution times that we measured (referring to a PC whose specifications are shown in Table 11). The execution times consider not only the evolution, but also the input/output (setting the initial conditions for the SMEFT parameters and accessing them after the evolution). These times should not be taken as exact, but as an order-of-magnitude estimate.

This result should be compared to the execution times of DSixTools, that can be up to $\mathcal{O}(20\,\text{s})$ ($\mathcal{O}(10\,\text{s})$) for the numeric (approximate) solution. This package performs a consistency check on the input that affects heavily the execution time for large inputs. Such impact is $\sim 8\,\text{s}$ for the input used in the comparison between the two programs.

Conversely, considering just 3 non vanishing independent operators at $\mu = \Lambda$, the execution time is $\mathcal{O}(10\,\text{s})$ ($\mathcal{O}(2\,\text{s})$) for the numeric (approximate) solution.

[8] For SMEFT coefficients with flavour indices, we set $\neq 0$ at least one entry for each operator.

Table 10 Execution times of C++ programs that use RGESolver

Operation	Time
Approximate solution	$\mathcal{O}(2\,\text{ms})$
Numeric solution	$\mathcal{O}(50\,\text{ms})$
Numeric solution + generation of SM initial conditions + back-rotation	$\mathcal{O}(100\,\text{ms})$

Table 11 Technical specifications of the computer used for the test

CPU	Intel© Core™ i7-6500U CPU @2.50GHz×4
RAM	7,7 GiB
OS	Ubuntu 20.04.2 LTS

6 Conclusions

RGESolver is an open-source C++ library that performs the renormalization group evolution in the context of the SMEFT at dimension-six level, in the most general flavour scenario. Only operators that preserve baryon and lepton number are considered. RGESolver was designed to be easy to use, highly efficient and suitable to perform extensive phenomenological analysis. To this aim, it will be included in HEPfit ([3]), a multipurpose and flexible analysis framework that can be used for fitting models to experimental and theoretical constraints. RGESolver outperforms DSixTools in execution time by two orders of magnitude, while retaining an excellent accuracy. Further details and the full documentation can be found on GitHub.

Acknowledgements SDN would like to thank his colleagues (and friends) from Rome and Padua for their contributions to early tests of RGESolver and A. Vicente, whose help was essential to complete the comparison between RGESolver and DSixTools. This work was supported in part by the Italian Ministry of Research (MIUR) under grant PRIN 20172LNEEZ. The Feynman diagrams shown in this work were drawn with TikZ-Feynman [16].

Data Availibility Statement The manuscript has associated data in a data repository. [Authors' comment: The code release described in this manuscript is available at https://doi.org/10.5281/zenodo.7639023.]

Funded by SCOAP[3]. SCOAP[3] supports the goals of the International Year of Basic Sciences for Sustainable Development.

References

1. W. Buchmuller, D. Wyler, Effective Lagrangian analysis of new interactions and flavor conservation. Nucl. Phys. B **268**, 621–653 (1986). https://doi.org/10.1016/0550-3213(86)90262-2
2. B. Grzadkowski, M. Iskrzynski, M. Misiak, J. Rosiek, Dimension-six terms in the standard model Lagrangian. JHEP **10**, 085 (2010). https://doi.org/10.1007/JHEP10(2010)085. arXiv:1008.4884 [hep-ph]
3. J. De Blas et al., `HEPfit`: a code for the combination of indirect and direct constraints on high energy physics models. Eur. Phys. J. C **80**(5), 456 (2020). https://doi.org/10.1140/epjc/s10052-020-7904-z. arXiv:1910.14012 [hep-ph]
4. A. Celis, J. Fuentes-Martin, A. Vicente, J. Virto, DsixTools: the standard model effective field theory toolkit. Eur. Phys. J. C **77**(6), 405 (2017). https://doi.org/10.1140/epjc/s10052-017-4967-6. arXiv:1704.04504 [hep-ph]
5. J. Fuentes-Martin, P. Ruiz-Femenia, A. Vicente, J. Virto, DsixTools 2.0: the effective field theory toolkit. Eur. Phys. J. C **81**(2), 167 (2021). https://doi.org/10.1140/epjc/s10052-020-08778-y. arXiv:2010.16341 [hep-ph]
6. E.E. Jenkins, A.V. Manohar, M. Trott, Renormalization group evolution of the standard model dimension six operators I: formalism and lambda dependence. JHEP **10**, 087 (2013). https://doi.org/10.1007/JHEP10(2013)087. arXiv:1308.2627v4 [hep-ph]
7. E.E. Jenkins, A.V. Manohar, M. Trott, Renormalization group evolution of the standard model dimension six operators II: Yukawa dependence. JHEP **01**, 035 (2014). https://doi.org/10.1007/JHEP01(2014)035. arXiv:1310.4838v3 [hep-ph]
8. R. Alonso, E.E. Jenkins, A.V. Manohar, M. Trott, Renormalization group evolution of the standard model dimension six operators III: Gauge coupling dependence and phenomenology. JHEP **04**, 159 (2014). https://doi.org/10.1007/JHEP04(2014)159. arXiv:1312.2014v4 [hep-ph]
9. E.E. Jenkins, A.V. Manohar, P. Stoffer, Low-energy effective field theory below the electroweak scale: operators and matching. JHEP **03**, 016 (2018). https://doi.org/10.1007/JHEP03(2018)016. arXiv:1709.04486 [hep-ph]
10. Luo Mx, Y. Xiao, Two loop renormalization group equations in the standard model. Phys. Rev. Lett. **90**, 011601 (2003). https://doi.org/10.1103/PhysRevLett.90.011601. arXiv:hep-ph/0207271
11. M. Srednicki, *Quantum Field Theory* (Cambridge University Press, Cambridge, 2007). https://doi.org/10.1017/cbo9780511813917.031
12. A.J. Buras, M. Jung, Analytic inclusion of the scale dependence of the anomalous dimension matrix in Standard Model Effective Theory. JHEP **06**, 067 (2018). https://doi.org/10.1007/JHEP06(2018)067. arXiv:1804.05852 [hep-ph]
13. M.C. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, F. Rossi, R. Ulerich, GNU Scientific Library. Network Theory, Ltd. (2019)
14. W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, *Numerical Recipes in C (2nd Ed.): The Art of Scientific Computing*, 2nd edn. (Cambridge University Press, Cambridge, 1992)
15. B.C. Allanach, SUSY Les Houches Accord 2. Comput. Phys. Commun. **180**, 8–25 (2009). https://doi.org/10.1016/j.cpc.2008.08.004. arXiv:0801.0045 [hep-ph]
16. J. Ellis, TikZ-Feynman: Feynman diagrams with TikZ. Comput. Phys. Commun. **210**, 103–123 (2017). https://doi.org/10.1016/j.cpc.2016.08.019. arXiv:1601.05437 [hep-ph]