CERN-LHCEFTWG-20xx-yyy
CERN-LPCC-20xx-zz

November, 2020

LHC EFT WG Note: Precision matching of microscopic physics to the SMEFT

Sally Dawson[1,16], *Admir Greljo*[2,15,16], *Kristin Lohwasser*[3,16], *Jason Aebischer*[4], *Supratim Das Bakshi*[5], *Adrián Carmona*[5], *Joydeep Chakrabortty*[6], *Timothy Cohen*[7], *Juan Carlos Criado*[8], *Javier Fuentes-Martín*[5], *Achilleas Lazopoulos*[9], *Xiaochuan Lu*[7], *Pablo Olgoso*[5], *Sunando Kumar Patra*[10], *José Santiago*[5], *Anders Eller Thomsen*[2], *Zhengkang Zhang*[11], *Stefano Di Noi*[12,13], *Luca Silvestrini*[14]

[1] Department of Physics, Brookhaven National Laboratory, Upton, NY, United States
[2] Albert Einstein Center for Fundamental Physics, Institute for Theoretical Physics, University of Bern, Bern, Switzerland
[3] Department of Physics and Astronomy, University of Sheffield, Sheffield, United Kingdom
[4] Physik-Institut, Universität Zürich, Zürich, Switzerland
[5] Departmento Fisica Teorica y del Cosmos, Universidad de Granada, Granada, Spain
[6] Department of Physics, Indian Institute of Technology, Kanpur, 208016, India
[7] Institute for Fundamental Science, University of Oregon, Eugene, OR, United States
[8] Department of Physics, Durham University, Durham, United Kingdom
[9] Institute for Theoretical Physics, ETH Zürich, Zürich, Switzerland
[10] Bangabasi Evening College, Kolkata, India
[11] Department of Physics, University of California, Santa Barbara, CA, United States
[12] Dipartimento di Fisica e Astronomia "G. Galilei", Università degli Studi di Padova, Padua, Italy
[13] Istituto Nazionale di Fisica Nucleare, Sezione di Padova, Padua, Italy
[14] Istituto Nazionale di Fisica Nucleare, Sezione di Roma, Rome, Italy
[15] Department of Physics, University of Basel, Klingelbergstrasse 82, CH-4056 Basel, Switzerland
[16] Convenors of the LHC EFT working group area 5

Abstract
This note gives an overview of the tools for the precision matching of ultraviolet theories to the Standard Model effective field theory (SMEFT) at the tree level and one loop. Several semi- and fully automated codes are presented, as well as some supplementary codes for the basis conversion and the subsequent running and matching at low energies. A suggestion to collect information for cross-validations of current and future codes is made.

Keywords
EFT, One-Loop Matching, Running, UV models

Contents

1 Introduction and Motivation

The Standard Model effective field theory (SMEFT) describes physics at energies below the new mass scale which is assumed to be above the electroweak scale. The imprints of ultraviolet (UV) physics are encoded in the Wilson coefficients (WC) of the SMEFT. Measuring these coefficients and their correlations allows for discriminating between different UV models. The important technical step in this procedure is the *matching*, where the heavy degrees of freedom are integrated out and their effects are represented by local operators. The resulting WC are expressed in terms of the parameters of the UV theory such as couplings and masses. This facilitates the interpretation of the SMEFT analyses in explicit UV models.

Matching beyond the tree level is important since many interesting observables are generated only at the one-loop level. However, this task is not only technically challenging but given the number of possible UV models, repetitive and time-consuming. To address the issue, several dedicated tools have been developed recently. For example, the SuperTracer [1], Matchete (to be released), STrEAM [2] and CoDEx [3] packages aim at facilitating the one-loop EFT matching of generic UV models using path-integral methods. Matchmakereft [4], instead, automates the diagrammatic EFT matching of generic UV models. These tools are introduced in Sec. 2 where also possible avenues for code validation and benchmarking are described.

Furthermore, there are several codes on the market that deal with Renormalization Group Evolution (RGE) and the treatment of numerical Wilson coefficient values. Such tools are especially important in phenomenological analyses but also when comparing analytic matching results obtained from the above-mentioned matching codes. In Sec. 3 some of these numerical tools are discussed. Namely, the (match)runner codes DsixTools [5, 6], RGESolver [7] and wilson [8], and the Wilson coefficient exchange format (WCxf) [9].

2 Matching Codes

Codes to (semi-)automatically match a concrete UV model to the SMEFT are important tools for constraining beyond the Standard Model (BSM) theories by the global SMEFT fits. An overview of the different codes and their primary functions is given below. The codes are introduced according to strict alphabetical order. Users should therefore study the full list before deciding which code is best suited for their needs.

2.1 `CoDEx`

`CoDEx` [3] is a Mathematica [10] package that integrates out heavy fields of spin-$0, 1/2, 1$ and computes the effective operators up to mass dimension-6 and associated Wilson coefficients (WCs) in terms of the model parameters. Relying on the functional method, it can perform the integration out at both tree- and 1-loop-levels. It offers the effective operators in both `SILH` [11, 12] and `Warsaw` [13, 14] bases. `CoDEx` can deal with BSM scenarios containing single or multiple mass-degenerate heavy fields of the same spin. To run the program, it requires very minimal input within a user-friendly format. The user needs to provide only the relevant part of the BSM Lagrangian that involves the heavy field(s) to be integrated out. `CoDEx` generates the effective action using functional method [1,2,15–23], and an internal program is used to identify the effective operators and accompanying WCs. The operators are computed at the energy scale where the integration out is performed, i.e., the mass of the heavy field(s). `CoDEx` provides an option to invoke the RGE of the effective operators in Warsaw basis using the anomalous dimension matrices [24–26] and note down the set of operators that emerge at any other scale. `CoDEx` with its installation instructions, web documentation, and model examples is available on GitHub. .

<u>**User Inputs & Possible Outputs**</u>

The input information to run `CoDEx` for any given BSM scenario is minimal. Here, we depict a step-by-step procedure to compute the effective operators and the internal computation that is carried out at each step in `CoDEx`:

– User needs to provide the following information about the heavy field(s): Color, Isospin, Hyper-charge, Mass, and Spin, based on which the representation(s) of the heavy field(s) are evaluated by the package internally. On top of that, the relevant part of the BSM Lagrangian that involves the heavy field(s) must be supplied by the user.

– Relying on these inputs, the package can integrate out propagators from the tree- and 1-loop-level processes of that BSM theory. The derivative term and the mass term of the heavy field are internally constructed in the package with the help of quantum numbers provided by the user.

– The `CoDEx`-function – `treeOutput` integrates out the heavy tree-level propagators only and generates the tree-level WCs and effective operators. Internally, `CoDEx` computes the heavy field classical solution by solving the Euler-Lagrange equation upto an order that will contribute to the dimension-6 operators. Then, the solution is substituted back in the BSM Lagrangian to generate the effective Lagrangian. After implementing appropriate mass-dimension cuts and operator identities to match with the given SMEFT basis, the desired output is generated.

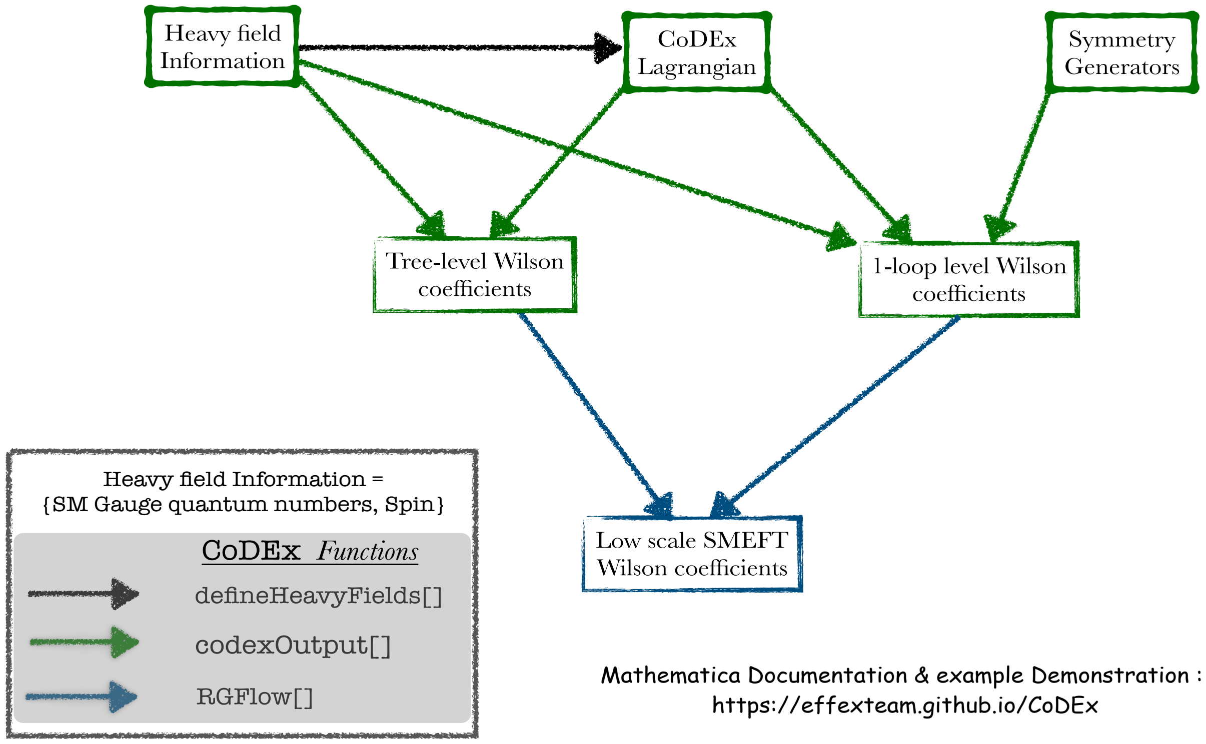

Fig. 1: Flow-chart for `CoDEx`. The inputs and outputs of the `CoDEx`-functions are represented by coloured lines.

– The `CoDEx`-function – `loopOutput` integrates out heavy propagators from loops and generates the 1-loop-level WCs and effective operators. In addition to the inputs required to generate tree-level operators, `loopOutput` needs the symmetry generators of the gauge groups, under which the heavy field is charged. The symmetry generators of the SM gauge groups for frequently used representation are available in `CoDEx`, and thus the user does not need to provide that information externally. To compute the 1-loop generated effective operators, `CoDEx` internally computes the trace of the effective action formulae [19, 27, 28]. The package recognizes the terms quadratic in the heavy field from the BSM Lagrangian. It builds the covariant derivative operator using the quantum numbers of the heavy field.

– Following the above steps, the user generates the effective operators at the matching scale. But one can also find the operators at the electroweak or other suitable scale using `CoDEx`-function:`RGFlow`. This function computes RGE for `Warsaw` basis effective operators using anomalous dimension matrices available in Refs. [24–26].

Developers' version: yet to be released

– **WCxF [9]:** We have added two `CoDEx`-functions: `wcxfOut` and `wcxfIn` to export and import the WCs in a format compatible with other codes, see Refs. [29]. There exist several packages with different EFT utilities (facilitating WC matching, renormalisation group running, and calculating observables). However, not all of them are on the same footing, e.g., they use different EFT bases and operator normalization. It is important to have a data exchange interface among these programs, and WCxF provides that. Thus, it is desirable for any package to have import & export functions in this format to interface the program with others.

– **Heavy-light mixed WCs:** The mixed heavy-light contribution, for scalars only, is included in the matching result by expanding the UV action around the light field solution obtained using the Euler-Lagrange equation, similar to the pure heavy-loop approach. Using the 'covariant diagrams' methodology presented in Ref. [28], we have calculated the formula for the mixed heavy-light contributions and cross-checked it with that of Ref. [30] (see Tables 1–5 in there). We implement this formula in `CoDEx` along with the 16 BSMs to generate the mixed heavy-light Wilson coefficients [31–33].

– **Identities:** Evaluating the effective action for a UV model may generate gauge-invariant structures which do not directly resemble the desired effective operator basis. Then, we require the implementation of operator identities and equations of motion (EOMs) of light degrees of freedom on the effective Lagrangian to transform the gauge-invariant terms to desired structures. The implementation of these identities depends upon the choice of the effective operator basis. These transformations like Fierz identities, SM fields' equations of motion, and SMEFT dimension-6 operator identities are introduced in the developer version of `CoDEx`. These transformations are necessary to represent the effective Lagrangian in terms of the SMEFT operators and their WCs.

2.2 `Matchete` and `SuperTracer`

`Matchete` and `SuperTracer` are Mathematica packages aimed at automating the complete one-loop matching of arbitrary UV models into their EFTs, using the functional-matching procedure described in [1]. The workflow of these packages is summarized in Fig. 2. `SuperTracer` allows for the evaluation of generic supertraces, one of the most time-consuming and repetitive tasks at the center of functional matching computations. In the future, `Matchete` is planned to supersede `SuperTracer` and provide a comprehensive and fully automated matching tool, with a user-friendly interface that will only require the UV Lagrangian as user input. A proof of concept for `Matchete` will be made publicly available soon [34].

The functional one-loop matching procedure is performed by evaluating the hard region [23] of two types of functional supertraces, log-type, and power-type supertraces, corresponding respectively to the first and second term in the following expression:

$$S^{(1)}_{\text{EFT}} = \frac{i}{2}\text{STr}\ln\Delta^{-1}\Big|_{\text{hard}} - \frac{i}{2}\sum_{n=1}^{\infty}\text{STr}[(\Delta X)^n]\Big|_{\text{hard}}, \tag{1}$$

where $S^{(1)}_{\text{EFT}}$ is the one-loop EFT action, Δ is the gauge-invariant kinetic operator, and X the interaction terms. These can be derived directly from the UV Lagrangian by

$$\frac{\delta^2\mathcal{L}_{\text{UV}}}{\delta\eta_j\delta\bar{\eta}_i} = \delta_{ij}\Delta_i^{-1} - X_{ij}. \tag{2}$$

In this formalism, η_i runs over all fields of the theory (counting also conjugate fields). The calculation of the functional supertraces is kept explicitly gauge invariant by doing a Covariant Derivative Expansion (CDE) [15, 17, 35] of both propagators and interaction terms.

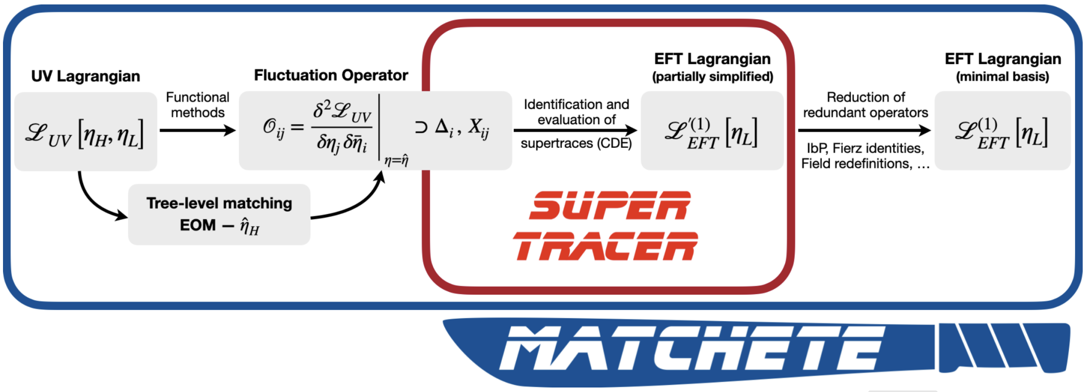

Fig. 2: Workflow for functional one-loop matching in the `Matchete` and `SuperTracer` packages.

`Supertracer` can perform the CDE and loop integration of the supertraces for arbitrary interaction terms to get generic expressions for the one-loop EFT. To install the package simply run the following command in a new Mathematica notebook:

```
In[1]:= Import["https://gitlab.com/supertracer/supertracer/-/raw/master/
          install.m"]
```

This will download and install `Supertracer` in the Applications folder in the base directory of Mathematica. After the package has been installed, it can be loaded into a Mathematica kernel with the command

```
In[2]:= << SuperTracer`
```

`Supertracer` operates with an expansion in light mass dimensions, which provides an expansion parameter for the power-type supertraces and for the CDE. The kinematics of the supertraces change depending on the spin and masses of the propagating fields. Accordingly, `Supertracer` distinguishes between heavy and light scalars, fermions, vectors, and ghost fields, denoted as `Φ,ϕ,Ψ,ψ,V,A,cV,cA`, respectively. The generic form of the log-type supertrace for heavy fermions up to dimension 6, is then found with the routine

```
In[3]:= LogTerm[Ψ,6]
```

$$\text{Out[3]=}\quad -\frac{1}{6}\,\text{Log}\left[\frac{\overline{\mu}^2}{\text{M}_\text{H}^2}\right]\text{G}^{\mu\nu}{**}\ \text{G}^{\mu\nu} + \frac{1}{15}\frac{1}{\text{M}_\text{H}^2}\text{D}_\mu\text{G}^{\mu\nu}{**}\ \text{D}_\rho\text{G}^{\nu\rho} + \frac{1}{90}\,\text{i}\,\frac{1}{\text{M}_\text{H}^2}\text{G}^{\mu\nu}{**}\ \text{G}^{\mu\rho}{**}\ \text{G}^{\nu\rho}$$

Power-type supertraces depend on the X terms that are involved in the trace. To extract the generic form, one has to put in a list of X terms defining the types of the propagating fields in the loop, and the light dimension of each X. Thus, to extract the trace with a heavy fermion and a gauge field with interaction terms of dimension $5/2$ (the smallest light dimension of a heavy fermion field), up to dimension 6, we call

```
In[4]:= STrTerm[{X[{Ψ,A}, 5/2],X[{A,Ψ}, 5/2]}, 6]
```

$$\text{Out[4]=}\quad \frac{1}{8}\,\text{i}\left(3 + 2\ \text{Log}\left[\frac{\overline{\mu}^2}{\text{M}_\text{H}^2}\right]\right)\gamma_\mu{**}\text{D}_\mu\text{X}_{\Psi_\text{i}\text{A}_\text{j}}{**}\text{X}_{\text{A}_\text{j}\Psi_\text{i}} + \frac{1}{2}\left(1 + \text{Log}\left[\frac{\overline{\mu}^2}{\text{M}_\text{H}^2}\right]\right)\text{M}_\text{H}\text{X}_{\Psi_\text{i}\text{A}_\text{j}}{**}\ \text{X}_{\text{A}_\text{j}\Psi_\text{i}}$$

With the generic formulas for the supertraces, there is limited possibility for simplifications of the expressions, as e.g. contractions in the Dirac algebra and covariant derivatives cannot be fully resolved. Furthermore, in more realistic models, the X terms, being matrices in field space, are large objects, and expanding the traces by hand is a huge effort. Supertracer provides the option of directly substituting explicit expressions for the X terms into the traces, which allows for the internal routines to simplify the results as much as possible. This makes it feasible to use SuperTracer for realistic BSM matching computations.

For proper handling of indices and to include the action of the gauge fields on the various matter fields, the user has to define various objects to properly perform the substitution of the X terms. One has to be rather careful when doing this, and we refer the user to the full manual [1]. Here we restrict ourselves to a simple model where, as in the previous supertrace, the interactions are between a heavy fermion and an Abelian gauge field:

```
In[5]:=  STrTerm[{X[{Ψ,A}, 5/2], X[{A,Ψ}, 5/2]}, 6,
           {
            {Ψ,A}->{{-e γ[α[j]]**ψh[]}, {e γ[α[j]]**CConj[ψh[]]}},
            {A,Ψ}->{{-e Bar[ψh[]]**γ[α[i]], e Bar[CConj[ψh[]]]**γ[α[
         i]]}},
            M[Ψ]->{Mh, Mh},
            G[Ψ]->{{e[1]},{[e[-1]]}},
            G[A]->{{}}
           }
         ]
```

Out[5]= $\frac{1}{2} i\, e^2 \left(1 + 2\, \text{Log}\left[\frac{\overline{\mu}^2}{\text{Mh}^2}\right]\right) \overline{\psi\text{h}}{**}\gamma_\mu{**}D_\mu\psi\text{h} - 2\, e^2\, \text{Mh} \left(1 + 2\, \text{Log}\left[\frac{\overline{\mu}^2}{\text{Mh}^2}\right]\right) \overline{\psi\text{h}}{**}\psi\text{h}$

The design goal of Matchete is to completely automate the implementation of functional matching. This will include a simple user interface for the UV Lagrangian input, and the implementation of functional derivatives to automate the computation of fluctuation operators and the solution of the EOMs for the heavy fields. For the user, this will result in a tremendous simplification, as the input is directly on the Lagrangian level, rather than having to provide the much more complicated functional objects. The second pillar of Matchete is the integration of robust simplification routines utilizing integration-by-parts (IBP) identities, Fierzing, and field redefinitions to bring the output of the supertraces to an operator basis. Whereas SuperTracer requires the user to be familiar with functional matching, Matchete will be approachable even with rudimentary knowledge.

On a practical level, both SuperTracer and Matchete use dimensional regularization with $\overline{\text{MS}}$ renormalization, with γ_5 being treated in the naive dimensional regularization (NDR) scheme. Gauge invariance of the resulting EFT is ensured by performing the matching computation in the background field gauge. Furthermore, the metric signature is chosen to be $g_{\mu\nu} = (+, -, -, -)$ and the Levi-Civita tensor convention to be $\varepsilon^{0123} = +1$.

2.3 Matchmakereft

Matchmakereft is a Python tool to perform the matching of arbitrary models onto arbitrary effective theories up to one-loop order in an automated way. The matching is performed in a diagrammatic fashion by matching one-light-particle-irreducible (1LPI) off-shell amplitudes functions in the background field method. We currently use dimensional regularization with $\overline{\text{MS}}$ renormalization and use an anti-commuting γ^5 convention, which is enough for most of the relevant calculations in the SMEFT and its simple generalizations. Further information can be found in the project web page https://ftae.ugr.es/matchmakereft/ and in the manual [4]. Its code is publicly available at https://gitlab.com/m4103/matchmaker-eft and it can be installed via `PyPI`

```
> python3 -m pip install matchmakereft
```

or `Conda`

```
> conda install -c matchmakers matchmakereft
```

`Matchmakereft` uses well-tested tools for the different parts of the matching calculation and has therefore some dependencies that have to be met by the user. These include Mathematica, `FeynRules` [36], `FORM` [37] and `QGRAF` [38] (see the manual for details on how to install them).

Models (both UV and effective) are defined via the Mathematica package `FeynRules` following the standard rules of that package. One particularity is that every particle has to be defined as heavy or light by using the following rule in their definition `FullName -> "light"` for light particles and `FullName -> "heavy"` for heavy ones. This defines models as *light* models when no heavy particles are present, and *heavy* models, when there are some heavy particles present. EFT models have to be light models whereas UV models can be heavy or light. In the former case, the finite tree-level and one-loop matching is performed, in the latter, the one-loop anomalous dimensions are computed. Depending on the type of model and its properties, further information has to be provided. This includes gauge information, possible symmetry properties of the different couplings, and the explicit reduction from a Green basis to a physical one in the case of the EFT. This latter point is very important. Given that `Matchmakereft` performs the matching off-shell it is essential that a full Green basis is defined as the EFT. Similarly, the use of the background field method is crucial for the matching to be gauge-independent. Given that the model definition is the only step of the process that is not fully automated and therefore more error-prone, we encourage the reader to consult the manual for all the details.

Once installed, `Matchmakereft` can be started by typing `matchmakereft` in the terminal. This loads the CLI that looks like this

```
Checking for updates.
matchmakereft is up-to-date.

Welcome to matchmakereft v1.0.2
Please refer to arXiv:2112.10787 when using this code.

matchmakereft>
```

The following commands are currently available:

```
matchmakereft> test_installation
```

This command tests the installation by running three sample calculations and comparing the results against the known ones in the literature.

```
matchmakereft>copy_models modellocation
```

This command creates a directory `MatchMakerEFT` under `modellocation` with some sample models that can be used as starting points for further model generation. In particular the B-preserving SMEFT model is provided.

```
matchmakereft>create_model modefile1.fr ... modefilen.fr
```

This command creates a `Matchmakereft` model under directory `modefilen_MM`. Some extra files could be needed for model creation. Please check the manual for details.

```
matchmakereft>match_model_to_eft UVModelName EFTModelName
```

This command computes the hard-region contribution to all the required amplitudes to match the UV model under directory `UVModelName` onto the EFT under directory `EFTModelName` and compares the two calculations to perform the matching. The corresponding matching is stored in `UVModelName/MatchingResult.dat` at three different levels of the calculation. First the matching in the Green basis, then the same one after canonical normalization and finally the matching in the physical basis (provided the corresponding reduction was provided during model generation). The matching of the gauge couplings in the background field method is also provided. A significant number of cross checks are performed using the redundancy inherent to the off-shell matching in the background field method. If any inconsistency is found, it is reported with some details stored in the file `UVModelName/MatchingProblems.dat`.

```
matchmakereft>compute_rge_model_to_eft UVModelName EFTModelName
```

This command computes the one-loop anomalous dimensions of the Wilson coefficients of the EFT under directory `EFTModelName` assuming the (also light) UV model under directory `UVModelName`. The corresponding anomalous dimensions are stored in `UVModelName/RGEResult.dat`.

```
matchmakereft>clean_model Model
```

This command restarts model `Model` to repeat the matching calculation from scratch.

```
matchmakereft>check_linear_dependence EFTModelName
```

This command checks if the operators defined in EFTModelName are (off-shell) linearly independent or not and if they are not, it provides the linear relations among them.

In order to add flexibility, `Matchmakereft` adds some commands to split the calculation of the matching in smaller steps (amplitude calculation and Wilson coefficient calculation) and also includes the possibility of performing only tree-level matching. Further details can be found in the manual.

`Matchmakereft` is in very active development and we encourage the user to always update to the latest version and to check the manual or the web page for updates on its functionality.

2.4 `MatchingTools`

`MatchingTools` [39] is a Python package for performing tree-level matching calculations between general EFTs, and for implementing the algebraic manipulations of effective Lagrangians needed to re-write them in terms of a basis of operators. Its code is publicly available at github.com/jccriado/matchingtools, and it can be installed through:

```
> pip3 install matchingtools
```

The main focus of `MatchingTools` is on applications related to the SMEFT, but it works in a more general setting. It can integrate out fields of spin 0, 1/2 or 1 out of the box. Fields of higher spin can also be included by providing the corresponding propagators. No assumptions are made about their transformation properties under internal symmetry groups. Regarding their interactions, the only condition is that the interaction Lagrangian is a Lorentz-invariant polynomial in the fields and their derivatives. In particular, operators of any dimension in the UV theory and EFT can be included.

We first overview here the methods used internally by `MatchingTools`. Given a UV theory defined by an action $S_{\text{UV}}[\phi, \Phi]$, with light fields ϕ and heavy fields Φ, `MatchingTools` integrates out Φ at tree level by solving its equation of motion and replacing it in S_{UV}. That is, the effective action is given by

$$S_{\text{EFT}}[\phi] = S_{\text{UV}}[\phi, \Phi_c(\phi)], \qquad \text{where } \left.\frac{\delta S}{\delta \Phi}\right|_{\Phi=\Phi_c(\phi)} = 0. \tag{3}$$

`MatchingTools` computes the solution $\Phi_c(\phi)$ as a perturbative expansion in inverse powers of the mass M of Φ. It does so by means of an iterative procedure that generates a sequence of solutions $\Phi_n(\phi)$, starting with $\Phi_0(\phi) \equiv 0$ and given by

$$\Phi_n(\phi) \equiv P \left.\frac{\delta S_{\text{int}}}{\delta \Phi}\right|_{\Phi=\Phi_{n-1}(\phi)}, \tag{4}$$

where P is the propagator for Φ, expanded in powers of $1/M$, and $S_{\text{int}}[\phi, \Phi]$ is the interaction part of the UV action. Each $\Phi_n(\phi)$ is a solution of the equations of motion only to a finite order in $1/M$, but this order increases with n. `MatchingTools` can thus iterate this procedure to compute the solution to any order in $1/M$, which in turn gives the effective Lagrangian to any desired order.

The effective Lagrangian obtained from this method will contain in general a set of operators that are not independent. In order to re-write it in terms of a set of independent operators, a basis, three different types of operations can be applied to it: algebraic/group theory identities, field redefinitions (or, equivalently at leading order, using equations of motion), and integration by parts. `MatchingTools` unifies all of them under a general system for finding and replacing patterns in the effective Lagrangian. The patterns that can be replaced are products of fields and constant tensors, with arbitrary index contractions.

We now consider a simple example to illustrate the usage and features of MatchingTools. The UV theory has a $SU(2) \times U(1)$ symmetry, and it contains two scalar multiplets: ϕ, a light doublet with hypercharge 1/2; and Φ, a heavy triplet with vanishing hypercharge. Their interactions are given by

$$\mathcal{L}_{\text{int}} \supset -\kappa\, \Xi^a(\phi^\dagger \sigma^a \phi) - \lambda\, (\Xi^a \Xi^a)(\phi^\dagger \phi). \tag{5}$$

This theory can be defined in MatchingTools using the following code:

```
import matchingtools as mt

sigma = mt.TensorBuilder("sigma")
kappa = mt.TensorBuilder("kappa")
lamb = mt.TensorBuilder("lamb")

phi = mt.FieldBuilder("phi", 1, mt.boson)
phic = mt.FieldBuilder("phic", 1, mt.boson)
Xi = mt.FieldBuilder("Xi", 1, mt.boson)

L_int = -mt.OpSum(
  mt.Op(kappa(), Xi(0), phic(1), sigma(0, 1, 2), phi(2)),
  mt.Op(lamb(), Xi(0), Xi(0), phic(1), phi(1)),
)
```


First, the different symbols that appear in the Lagrangian are defined: the Pauli matrices σ, the coupling constants κ and λ, and the fields ϕ and Ξ. All these objects are viewed by MatchingTools as tensors, possibly with zero indices, as in the case of the coupling constants in this example. For the fields, their canonical dimension and commutation properties have to be specified. Finally, the interaction Lagrangian is constructed as a sum (OpSum) of operators (Op). Each operator is given by the list of its factors, which can be both fields and constant tensors. The index structure of the operator is expressed by placing a non-negative integer in each position corresponding to an index, with repeated integers denoting contraction.

The program is now ready to integrate out Ξ. To do so, one can write:

```
heavy_Xi = mt.RealScalar("Xi", 1, has_flavor=False)
L_eff = mt.integrate(
  heavy_fields=[heavy_Xi], interaction_lagrangian=L_int, max_dim=6
)
print(mt.Writer(L_eff, []))
```

This produces a list of all the terms in the resulting effective Lagrangian. To re-write it in terms of a basis of operators, one can make use of the find-and-replace system provided by MatchingTools. As an example, we will use the $SU(2)$ Fierz identity:

$$\sigma^a_{ij}\sigma^a_{kl} = 2\delta_{il}\delta_{jk} - \delta_{ij}\delta_{kl}, \tag{6}$$

to simplify the Lagrangian, by replacing every occurrence of the left-hand side of this equation by its right-hand side. To do this, we define the corresponding rule, which is a tuple whose first element is the pattern and whose second element is the replacement. We then apply this rule to the effective Lagrangian:

```
fierz_rule = (
  mt.Op(sigma(0, -1, -2), sigma(0, -3, -4)),
  mt.OpSum(
    number_op(2) * mt.Op(mt.kdelta(-1, -4), mt.kdelta(-3, -2)),
    -mt.Op(mt.kdelta(-1, -2), mt.kdelta(-3, -4))
  )
)
L_eff = mt.apply_rules(L_eff, [fierz_rule], max_iterations=1)
print(mt.Writer(mt.simplify(L_eff), []))
```

This code outputs the list of terms of the transformed Lagrangian. After all redundant terms have been removed and the Lagrangian is written as a linear combination of operators in the desired basis, the final step is usually to identify the coefficients of the operators in the basis. To do this, one can define rules to replace the explicit expression of each operator by a single symbol and then instruct `MatchingTools` to extract the coefficients of these symbols. For the purpose of our example, we do so for an overcomplete set operators, since we have not reduced all redundancies yet. The overcomplete basis is:

$$\mathcal{O}_{\phi 6} = (\phi^\dagger \phi)^3, \qquad \mathcal{O}_{\phi 4} = (\phi^\dagger \phi)^2, \tag{7}$$

$$\mathcal{O}_{\phi}^{(1)} = \phi^\dagger \phi (D_\mu \phi)^\dagger D^\mu \phi, \qquad \mathcal{O}_{\phi}^{(3)} = (\phi^\dagger D_\mu \phi)(D^\mu \phi)^\dagger \phi, \tag{8}$$

$$\mathcal{O}_{D\phi} = \phi^\dagger (D_\mu \phi) \phi^\dagger D^\mu \phi, \qquad \mathcal{O}_{D\phi}^{*} = (D_\mu \phi)^\dagger \phi (D^\mu \phi)^\dagger \phi, \tag{9}$$

and the code to obtain the corresponding coefficients:

```
Ophi6 = mt.tensor_op("Ophi6")
Ophi4 = mt.tensor_op("Ophi4")
...

definition_rules = [
  (mt.Op(phic(0), phi(0), phic(1), phi(1), phic(2), phi(2)), mt.OpSum(
    Ophi6)),
  (mt.Op(phic(0), phi(0), phic(1), phi(1)), mt.OpSum(Ophi4)),
  (mt.Op(mt.D(2, phic(0)), mt.D(2, phi(0)), phic(1), phi(1)), mt.OpSum(
    O1phi)),
  (mt.Op(phic(0), mt.D(2, phi(0)), mt.D(2, phic(1)), phi(1)), mt.OpSum(
    O3phi)),
  (mt.Op(phic(0), mt.D(2, phi(0)), phic(1), mt.D(2, phi(1))), mt.OpSum(
    ODphi)),
  (mt.Op(mt.D(2, phic(0)), phi(0), mt.D(2, phic(1)), phi(1)), mt.OpSum(
    ODphic))
]

L_eff = mt.apply_rules(L_eff, definition_rules, 1)
final_coef_names = ["Ophi6", "Ophi4", "O1phi", "O3phi", "ODphi", "ODphic
    "]
print(mt.Writer(L_eff, final_coef_names))
```

With output:

```
O1phi:
  2 (MXi^(-4)) kappa kappa
```

```
O3phi:
  -1 (MXi^(-4)) kappa kappa
...
```

Indicating that the coefficient of the operator $\mathcal{O}_\phi^{(1)}$ is $2\kappa^2/M_\Xi^4$, the coefficient of $\mathcal{O}_\phi^{(3)}$ is $-\kappa^2/M_\Xi^4$, ... This can be converted to LaTeX code using the method `write_latex` from the `Writer` class.

`MatchingTools` also provides an `extras` subpackage containing modules for SMEFT-related applications, including the definition of tensors and rules relevant for $SU(2)$, $SU(3)$ and Lorentz group theory, the definitions of the SM fields, the rules for applying the SM equations of motion, and the definitions of the Warsaw basis operators. More information on these modules and other `MatchingTools` features can be found at matchingtools.readthedocs.io.

2.5 `STrEAM`

`STrEAM` (**S**uper**Tr**ace **E**valuation **A**utomated for **M**atching) is a Mathematica package that automates the evaluation of functional supertraces that could arise when one matches a generic UV theory onto a relativistic EFT. `STrEAM` implements the covariant derivative expansion method and could provide the result to arbitrary order in the heavy mass expansion.

According to the streamlined functional matching prescription presented in Ref. [40], the matching result at the one-loop level can be computed by evaluating the functional supertraces

$$\int \mathrm{d}^4x\, \mathcal{L}_{\text{EFT}}^{\text{(1-loop)}}[\phi] = \frac{i}{2}\,\mathrm{STr}\log \boldsymbol{K}\Big|_{\text{hard}} - \frac{i}{2}\sum_{n=1}^{\infty}\frac{1}{n}\,\mathrm{STr}\Big[\big(\boldsymbol{K}^{-1}\boldsymbol{X}\big)^n\Big]\Big|_{\text{hard}}, \tag{10}$$

where the inverse (covariant) propagator matrix $\boldsymbol{K}$ is diagonal

$$K_i = \begin{cases} P^2 - m_i^2 & (\text{spin-}0) \\ \not{P} - m_i & (\text{spin-}\tfrac{1}{2}) \\ -\eta^{\mu\nu}(P^2 - m_i^2) & (\text{spin-}1) \end{cases}, \tag{11}$$

and the interaction matrix $\boldsymbol{X}$ can be organized into a derivative expansion:

$$\boldsymbol{X}(\phi, P_\mu) = \boldsymbol{U}[\phi] + \big(P_\mu \boldsymbol{Z}^\mu[\phi] + \bar{\boldsymbol{Z}}^\mu[\phi] P_\mu\big) + \cdots, \tag{12}$$

with $P_\mu \equiv iD_\mu$ the "open" covariant derivative. Therefore, a general power-type supertrace in Eq. (10)

$$-i\,\mathrm{STr}\left[\frac{1}{K_{i_1}} X_{i_1 i_2} \frac{1}{K_{i_2}} X_{i_2 i_3} \cdots \frac{1}{K_{i_n}} X_{i_n i_1}\right], \tag{13}$$

consists of a product sequence of segments of the form

$$\frac{1}{K_i}\big(P_{\mu_1}\cdots P_{\mu_n}\big)\, U_k \,\big(P_{\nu_1}\cdots P_{\nu_m}\big)\,. \tag{14}$$

Using Δ_i and Λ_i to denote the bosonic and fermionic versions of K_i^{-1}, respectively

$$\Delta_i \equiv \frac{1}{P^2 - m_i^2}\,, \qquad \Lambda_i \equiv \frac{1}{\not{P} - m_i}\,, \tag{15}$$

the concrete scope of `STrEAM` can be summarized as following:

STrEAM automates the evaluation of functional supertraces of the form

$$-i\,\mathrm{STr}\Big[\,f\big(P_\mu\,,\,\{U_k\}\,\big)\Big]\Big|_{\text{hard}}\,, \tag{16}$$

where f is a product sequence of P_μ, U_k, Δ_i and Λ_i, consisting of an arbitrary number of “propagator blocks”:

$$f \;=\; \Big[\;\cdots\;\left(P_{\mu_1}\ldots P_{\mu_n}\right)\left(\Delta_i \text{ or } \Lambda_i\right)\left(P_{\nu_1}\ldots P_{\nu_m}\right)U_k\;\cdots\;\Big]\,. \tag{17}$$

The last block in f is allowed to have a trivial U factor, i.e., $U = 1$, such that the log-type supertraces in Eq. (10) can also be covered upon taking a mass derivative

$$\frac{\partial}{\partial m_\Phi^2}\Big[i\,\mathrm{STr}\log\left(P^2-m_\Phi^2\right)\Big] = -i\,\mathrm{STr}\left[\frac{1}{P^2-m_\Phi^2}\right] = -i\,\mathrm{STr}\big[\Delta_\Phi\big]\big|_{\text{hard}}\,, \tag{18a}$$

$$\frac{\partial}{\partial m_\Phi}\Big[i\,\mathrm{STr}\log\left(\slashed{P}-m_\Phi\right)\Big] = -i\,\mathrm{STr}\left[\frac{1}{\slashed{P}-m_\Phi}\right] = -i\,\mathrm{STr}\big[\Lambda_\Phi\big]\big|_{\text{hard}}\,. \tag{18b}$$

The STrEAM package can be downloaded from GitHub at

https://www.github.com/EFTMatching/STrEAM

After placing the file “STrEAM.m” at the user’s own choice of directory “/path/to/package/”, one can load it with the usual Mathematica command:

```
In[1]:= <<"/path/to/package/STrEAM.m";
```

STrEAM is a compact package with a single main function SuperTrace. It has a simple syntax:

```
In[2]:= SuperTrace[dim, flist]
```

with two mandatory arguments: dim is an Integer that specifies the desired operator dimension in the evaluation result; flist is a List that specifies the functional operator $f\big(P_\mu, \{U_k\}\big)$ to be traced over; it consists of $\mathtt{P}_\mu$, $\mathtt{U}_\mathtt{k}$, $\Delta_\mathtt{i}$, and $\Lambda_\mathtt{i}$, organized in the form of Eq. (17). The main function SuperTrace also has a few options; see Sec. 4 in Ref. [2] for a list of them.

As a simple demonstration example, the result

$$-i\,\mathrm{STr}\left[\frac{1}{P^2-m_1^2}U_1^{[2]}\right]\Bigg|_{\text{hard}} = \int d^4x\,\tfrac{1}{16\pi^2}\,\mathrm{tr}\left[m_1^2\Big(1-\log\tfrac{m_1^2}{\mu^2}\Big)U_1 + \tfrac{1}{12m_1^2}F_{\mu\nu}F^{\mu\nu}U_1\right], \tag{19}$$

can be obtained by calling SuperTrace as

```
In[3]:= SuperTrace[6, {Δ1, U1}, Udimlist->{2}, display->True];
```

which will print

$$\texttt{-iSTr}\left[\frac{1}{\texttt{P}^2\texttt{-m}_1^2}\texttt{U}_1\right]\Big|_{\texttt{hard}} = \int d^4x \; \frac{1}{16\pi^2} \; \texttt{tr}\{$$

$$\texttt{m}_1^2\left(\texttt{1-Log}\left[\frac{\texttt{m}_1^2}{\mu^2}\right]\right) \qquad (\texttt{U}_1) \qquad (\texttt{dim-2})$$

$$\frac{1}{\texttt{12m}_1^2} \qquad (\texttt{F}_{\mu_1\mu_2})(\texttt{F}_{\mu_1\mu_2})(\texttt{U}_1) \qquad (\texttt{dim-6})$$

$$\}$$

With the option `display->True`, `SuperTrace` will print the evaluation result in `TableForm`, together with the input supertrace, as shown above. More demonstration examples, as well as a more detailed manual of `STrEAM` can be found in Sec. 4 of Ref. [2].

2.6 General procedure for code comparison

A key step in the development of automated matching tools is cross-validation and comparison between different theoretical approaches and code implementations. Indeed, a variety of cross-checks have already been implemented for the different matching codes. For instance, the CDE of multiple supertraces has been validated by direct comparison of `Supertracer` and `STrEAM` outputs. Moreover, both `MatchmakerEFT` and `Supertracer` have already been used and partially cross-checked in the context of specific UV models [41–52]. CoDEx generated matching results for sixteen SM extensions with a single heavy scalar are available here. SILH basis matching results are cross-checked with the models given in Ref. [19] and they agree. Warsaw basis matching result for singlet real scalar extensions of the SM is cross-checked with Refs. [53, 54] and it agrees. However, as matching codes become more mature, it becomes highly desirable to have more systematic and comprehensive cross-checks.

In an effort to establish a well-defined standard for cross-validation, we have created the GitLab repository `https://gitlab.com/modelmatch/ModelMatch`. This repository will be used as an archive for BSM to SMEFT (and possibly other EFTs down the line) matching calculations, at the time that it will provide a transparent and open-access framework for comparison among the different implementations. To this end, any matching calculation presented in this repository will contain the following three files:[1]

1. **Matching results:** in any format that the authors deem appropriate.
2. **Validation:** consisting of a WCxf file with numerical matching coefficients for a given set of benchmark parameters for comparison with other implementations.
3. **Additional information:** provided in the form of a document clearly stating (at least) the following information:
   - Corresponding author(s).
   - All theory assumptions entering into the one-loop matching computation, including renormalization scheme, γ_5 prescription, gauge-fixing procedure, metric signature, and Levi-Civita convention.
   - The complete UV Lagrangian. In case of heavy vectors, an additional Lagrangian in the broken phase is highly encouraged.

[1] More details will be provided in the GitLab repository with some basic information also publically available at `https://twiki.cern.ch/twiki/bin/view/LHCPhysics/EFTAC5`.

- The set of benchmark parameter values used in the validation file. To avoid possible numerical issues, factorizing the loop factors and using rational values for the model parameters would be preferred.

We propose the following representative examples of BSM models including, respectively, heavy scalars, fermions, and vectors: $S_1 + S_3$ scalar leptoquark extension, a heavy vector-like lepton transforming under the SM gauge group as $E \sim (\mathbf{1}, \mathbf{1}, -1)$, and a heavy vector triplet from the symmetry breaking $\mathrm{SU}(2)'_{\mathrm{L}} \times \mathrm{SU}(2)_X \to \mathrm{SU}(2)_{\mathrm{L}}$. As an ultimate test for the long-term future, we will also consider the matching of the SMEFT to the Low-Energy Effective Field Theory (LEFT), where the Higgs, top, W, and Z are integrated out. This latter example is particularly comprehensive as it involves simultaneously heavy scalars, fermions, and vectors.

2.7 Outlook

The dream is that one could take a Lagrangian (e.g. implemented in FeynRules) and then pass it through a code that spits out the 1-loop Wilson coefficients in the Warsaw basis automatically. Putting all of this together is an area of active interest. The field will surely move forwards by leaps and bounds, once the next generation of automated matching tools becomes operational at which point cross-validation of the codes will become very important.

3 Supplementary numerical Codes

In this section, we discuss additional codes that are used in phenomenological analyses for SMEFT and LEFT running or to compare different matching results.

3.1 `DsixTools`

`DsixTools` [5, 6] is an open-source Mathematica package that automates one-loop RGE in the SMEFT [24–26, 55] and in the LEFT [56], as well as one-loop SMEFT-to-LEFT matching [57–59]. One of the main features of `DsixTools` is that it contains not only numerical but also analytical routines, allowing for simple manipulation of beta functions and matching expressions.

`DsixTools 2.1` aims for a more visual and user-friendly experience. Together with the usual matching and running routines, it also provides an interface with useful information for all operators and parameters of the SMEFT and the LEFT, such as their flavor symmetries and the number of degrees of freedom. Routines for the implementation of this information on global expressions (like decay amplitudes and cross-sections) are also provided. Furthermore, `DsixTools` includes a user-friendly input that performs automatic consistency checks, simplifying the user's task.

The package can be simply installed by running the following command in a Mathematica notebook (it is advised to use a fresh kernel for the installation):

```
In[1]:= Import["https://raw.githubusercontent.com/DsixTools/DsixTools/
          master/install.m"]
```

This will download and install `DsixTools` in the Applications folder of the Mathematica base directory. It will also create Mathematica documentation for all the package routines.

The following lines provide a basic (yet complete) usage example:

```
In[2]:= Needs["DsixTools`"]
        NewScale[{HIGHSCALE -> 10000}];
        NewInput[{Clq1[1 ,1 ,1 ,2] -> 1/HIGHSCALE^2, Clq1[1 ,1 ,2 ,1] ->
            1/HIGHSCALE^2, CH -> -0.5/HIGHSCALE^2}];
        RunDsixTools;
        D6run[LeuVLL[2 ,2 ,1 ,1]] /. \[Mu] -> LOWSCALE
```

which illustrates how to load the package, provide the input for a given SMEFT Lagrangian at the UV scale $\Lambda_{\rm UV} =$ `HIGHSCALE` (in units of GeV), and calculate the LEFT WCs at the IR scale $\Lambda_{\rm IR} =$ `LOWSCALE` (by default, `LOWSCALE` = 5 GeV).

`DsixTools` also offers multiple options to interface with other EFT tools. In particular, it can import and export JSON and YAML files in the `WCxf` exchange format [9] (see Sec. 3.3 for further details). It also admits as an input the output generated by `Matchmakereft` (see Sec. 2.3). For instance, a `Matchmakereft` output file (MMEfile.dat) is loaded into `DsixTools` using the command line:

```
In[3]:= NewInput[{MMEfile -> "MMEfile.dat" , MS -> 1000 , lam2 -> 0.1 ,
            lam3 -> 0.2}, HIGHSCALE -> 1000]
```

where all the UV-model parameters (in this example MS, lam2 and lam3) and the `HIGHSCALE` must be assigned numerical values.

Additional information and an up-to-date version of the user's manual can be found on the package webpage: https://dsixtools.github.io/, or in the project's GitHub repository.

3.2 RGESolver

`RGESolver` [7] is an open-source `C++` library that performs the renormalization group evolution of the SMEFT Wilson coefficients in a fast and easy-to-use manner. The library deals with the most generic flavor scenario, assuming only lepton and baryon number conservation.

`RGESolver` has been developed with a specific focus on extensive phenomenological analyses. Two methods are available in order to solve the differential equations: a numerical solution or an approximate one (first leading log). Furthermore, after the RGE `RGESolver` can perform the so-called flavor back-rotation [60]. It also provides a routine to perform the evolution and the back-rotation with a simple command. `RGESolver` has been tested against `DSixTools 2.1` [5, 6] for both the numerical and the approximate method, obtaining differences between the two codes $\lesssim 10^{-5}\,1/\Lambda^2$ for initial conditions $\mathcal{O}(1/\Lambda^2)$.

`RGESolver` can generate initial conditions for the Standard Model parameters (gauge couplings, Higgs sector parameters, and Yukawa matrices) at any given scale solving pure Standard Model renormalization group equations.

We give a simple example: these few lines generate the initial conditions (in the basis where the down Yukawa matrix is diagonal) for the Standard Model parameters at the scale $\mu = \Lambda = 10$ TeV, set $\mathcal{C}^{dH}_{1,2}(\Lambda) = 1/\Lambda^2$, solve numerically the renormalization group equations down to $\mu = 250$ GeV and perform the back-rotation to get back into the original basis. Finally, the real part of the evolved coefficient $\mathcal{C}^{dH}_{1,2}(250\,\text{GeV})$ can be accessed via the dedicated getter method.

```
double Lambda = 10000.;
```

```
S.GenerateSMInitialConditions(Lambda, "DOWN", "Numeric");
S.SetCoefficient("CHdR", 1. / (Lambda * Lambda), 1, 2);
S.EvolveToBasis("Numeric", Lambda, 250., "DOWN");
double EvolvedCHdR_12 = S.GetCoefficient("CHdR", 1, 2);
```

All the details about the installation, the extended documentation, and some examples can be found in the GitHub dedicated page.

3.3 WCxf

The Wilson coefficient exchange format (WCxf) defines a standard for Wilson coefficients used in computer codes. Since many different conventions are being used in the literature, it is important to have a collection of unique definitions for the different bases and the corresponding Wilson coefficients, especially when comparisons between different codes are performed. A list of all the computer tools that already support WCxf as well as their corresponding bases can be found on the WCxf GitHub website https://wcxf.github.io/. The format is extensible in the sense that any new basis can be added to the predefined bases, by providing simply a yaml file in which the convention for the Wilson coefficients together with the underlying EFT are specified. Further details on how to extend WCxf can be found in [9] and on the GitHub webpage.

Furthermore, there is the Python module WCxf, which allows changing numerical values of Wilson coefficients between different operator bases. Especially when comparing results obtained with different computer codes this module comes in handy, as it allows translation of the Wilson coefficients from one code in an automated way into the basis of the other, which facilitates cross-checks and comparisons. In the following we will show how to translate Wilson coefficients from one basis into another using WCxf:

The package can be easily installed, using

```
python3 -m pip install wcxf --user
```

Importing a given WCxf yaml file that specifies the Wilson coefficient values is done by:

```
import wcxf

with open('my_wcxf_input_file.yml', 'r') as f:
  wc = wcxf.WC.load(f)
```

Translating the Wilson coefficients of the imported WCxf file into another basis is achieved by

```
wc_new = wc.translate('My target basis')
```

Further details on WCxf as well as further commands can be found in [9].

3.4 wilson

The Python package wilson [8] is a matchrunning tool, which is mainly used in phenomenological codes such as flavio [61] and smelli [62], but can also be used independently. It allows to numerically run all Wilson coefficients in the SMEFT to arbitrary scales, as well as matching them onto the LEFT. Furthermore, the full LEFT running below the electroweak scale is implemented in `wilson`. The implementation entails

- The complete one-loop SMEFT beta functions [24–26].
- The complete tree-level [57, 59] and one-loop [58] matching from SMEFT onto LEFT.
- The complete one-loop LEFT running [56, 63].

`wilson` also supports the `WCxf` format [9] and takes back-rotation [60] effects into account, which result when mass matrices have to be rediagonalized after RGE running.

The package can be installed using

```
python3 -m pip install wilson --user
```

To set a Wilson coefficient to a certain value at a particular scale in a given EFT the Wilson class is used. For instance, setting the Wilson coefficient of the SMEFT operator $\mathcal{O}_{dG}^{23} = \left(\bar{q}_2 \sigma^{\mu\nu} T^A d_3\right) \varphi \, G_{\mu\nu}^A$ to one at the scale $\Lambda = 1$ TeV in the Warsaw basis one writes:

```
from wilson import Wilson
mywilson = Wilson({'dG_23': 1e-6}, scale=1e3, eft='SMEFT', basis='Warsaw
    ')
```

This Wilson coefficient can then be run down to the EW scale, matched onto the Weak Effective Theory (WET), which is equivalent to the LEFT, and further run down to a lower scale. For example, matching onto the JMS basis (introduced in [57]) and running to 100 GeV is achieved by

```
wc_JMS = mywilson.match_run(scale=100, eft='WET', basis='JMS')
```

Further information and updates can be found on the project website https://wilson-eft.github.io/ and in [8].

4 Conclusions and Outlook

Several codes aiming towards a full automatization of matching the short-distance BSM models to the SMEFT have been presented alongside some numerical codes for the basis conversion and the low-energy matching and running. A procedure to cross-validate the different matching approaches has been suggested for future work, identifying several interesting models.

Acknowledgments: This work was done on behalf of the LHC EFT WG and we would like to thank members of the LHC EFT WG for stimulating discussions which led to this document.

References

[1] J. Fuentes-Martin, M. König, J. Pagès, A. E. Thomsen, and F. Wilsch, *SuperTracer: A Calculator of Functional Supertraces for One-Loop EFT Matching*, *JHEP* **04** (2021) 281, [arXiv:2012.08506].

[2] T. Cohen, X. Lu, and Z. Zhang, *STrEAMlining EFT Matching*, *SciPost Phys.* **10** (2021), no. 5 098, [arXiv:2012.07851].

[3] S. Das Bakshi, J. Chakrabortty, and S. K. Patra, *CoDEx: Wilson coefficient calculator connecting SMEFT to UV theory*, *Eur. Phys. J. C* **79** (2019), no. 1 21, [arXiv:1808.04403].

[4] A. Carmona, A. Lazopoulos, P. Olgoso, and J. Santiago, *Matchmakereft: automated tree-level and one-loop matching*, arXiv:2112.10787.

[5] A. Celis, J. Fuentes-Martin, A. Vicente, and J. Virto, *DsixTools: The Standard Model Effective Field Theory Toolkit*, *Eur. Phys. J. C* **77** (2017), no. 6 405, [arXiv:1704.04504].

[6] J. Fuentes-Martin, P. Ruiz-Femenia, A. Vicente, and J. Virto, *DsixTools 2.0: The Effective Field Theory Toolkit*, *Eur. Phys. J. C* **81** (2021), no. 2 167, [arXiv:2010.16341].

[7] S. Di Noi and L. Silvestrini, *RGESolver : a C++ library to perform Renormalization Group evolution in the Standard Model Effective Theory*, arXiv:2210.06838.

[8] J. Aebischer, J. Kumar, and D. M. Straub, *Wilson: a Python package for the running and matching of Wilson coefficients above and below the electroweak scale*, *Eur. Phys. J. C* **78** (2018), no. 12 1026, [arXiv:1804.05033].

[9] J. Aebischer et al., *WCxf: an exchange format for Wilson coefficients beyond the Standard Model*, *Comput. Phys. Commun.* **232** (2018) 71–83, [arXiv:1712.05298].

[10] I. Wolfram Research, "Mathematica, Champaign, IL." https://www.wolfram.com/language/.

[11] G. F. Giudice, C. Grojean, A. Pomarol, and R. Rattazzi, *The Strongly-Interacting Light Higgs*, *JHEP* **06** (2007) 045, [hep-ph/0703164].

[12] J. Elias-Miro, J. R. Espinosa, E. Masso, and A. Pomarol, *Higgs windows to new physics through d=6 operators: constraints and one-loop anomalous dimensions*, *JHEP* **11** (2013) 066, [arXiv:1308.1879].

[13] W. Buchmuller and D. Wyler, *Effective Lagrangian Analysis of New Interactions and Flavor Conservation*, *Nucl. Phys. B* **268** (1986) 621–653.

[14] B. Grzadkowski, M. Iskrzynski, M. Misiak, and J. Rosiek, *Dimension-Six Terms in the Standard Model Lagrangian*, *JHEP* **10** (2010) 085, [arXiv:1008.4884].

[15] M. K. Gaillard, *The Effective One Loop Lagrangian With Derivative Couplings*, *Nucl. Phys. B* **268** (1986) 669–692. [iNSPIRE].

[16] L.-H. Chan, *Effective-action expansion in perturbation theory*, *Phys. Rev. Lett.* **54** (Mar, 1985) 1222–1225.

[17] O. Cheyette, *Effective Action for the Standard Model With Large Higgs Mass*, *Nucl. Phys. B* **297** (1988) 183–204.

[18] M. S. Bilenky and A. Santamaria, *One loop effective Lagrangian for a standard model with a heavy charged scalar singlet*, *Nucl. Phys. B* **420** (1994) 47–93, [hep-ph/9310302].

[19] B. Henning, X. Lu, and H. Murayama, *How to use the Standard Model effective field theory*, *JHEP* **01** (2016) 023, [arXiv:1412.1837].

[20] A. Drozd, J. Ellis, J. Quevillon, and T. You, *The Universal One-Loop Effective Action*, *JHEP* **03** (2016) 180, [arXiv:1512.03003].

[21] M. Krämer, B. Summ, and A. Voigt, *Completing the scalar and fermionic Universal

One-Loop Effective Action, *JHEP* **01** (2020) 079, [arXiv:1908.04798].

[22] A. Angelescu and P. Huang, *Integrating Out New Fermions at One Loop*, *JHEP* **01** (2021) 049, [arXiv:2006.16532].

[23] J. Fuentes-Martin, J. Portoles, and P. Ruiz-Femenia, *Integrating out heavy particles with functional methods: a simplified framework*, *JHEP* **09** (2016) 156, [arXiv:1607.02142].

[24] E. E. Jenkins, A. V. Manohar, and M. Trott, *Renormalization Group Evolution of the Standard Model Dimension Six Operators I: Formalism and lambda Dependence*, *JHEP* **10** (2013) 087, [arXiv:1308.2627].

[25] E. E. Jenkins, A. V. Manohar, and M. Trott, *Renormalization Group Evolution of the Standard Model Dimension Six Operators II: Yukawa Dependence*, *JHEP* **01** (2014) 035, [arXiv:1310.4838].

[26] R. Alonso, E. E. Jenkins, A. V. Manohar, and M. Trott, *Renormalization Group Evolution of the Standard Model Dimension Six Operators III: Gauge Coupling Dependence and Phenomenology*, *JHEP* **04** (2014) 159, [arXiv:1312.2014].

[27] B. Henning, X. Lu, and H. Murayama, *One-loop Matching and Running with Covariant Derivative Expansion*, *JHEP* **01** (2018) 123, [arXiv:1604.01019].

[28] Z. Zhang, *Covariant diagrams for one-loop matching*, *JHEP* **05** (2017) 152, [arXiv:1610.00710].

[29] J. Aebischer, M. Fael, A. Lenz, M. Spannowsky, and J. Virto, eds., *Computing Tools for the SMEFT*, 10, 2019.

[30] S. A. R. Ellis, J. Quevillon, T. You, and Z. Zhang, *Extending the Universal One-Loop Effective Action: Heavy-Light Coefficients*, *JHEP* **08** (2017) 054, [arXiv:1706.07765].

[31] S. Das Bakshi, J. Chakrabortty, and M. Spannowsky, *Classifying Standard Model Extensions Effectively with Precision Observables*, arXiv:2012.03839.

[32] Anisha, S. Das Bakshi, J. Chakrabortty, and S. K. Patra, *A Step Toward Model Comparison: Connecting Electroweak-Scale Observables to BSM through EFT and Bayesian Statistics*, arXiv:2010.04088.

[33] Anisha, S. Das Bakshi, S. Banerjee, A. Biekötter, J. Chakrabortty, S. Kumar Patra, and M. Spannowsky, *Effective limits on single scalar extensions in the light of recent LHC data*, arXiv:2111.05876.

[34] J. Fuentes-Martín, M. König, J. Pagès, A. E. Thomsen, and F. Wilsch, *A Proof of Concept for Matchete: An Automated Tool for Matching Effective Theories (in preparation)*, https://gitlab.com/matchete/matchete.

[35] L.-H. Chan, *Derivative Expansion for the One Loop Effective Actions With Internal Symmetry*, *Phys. Rev. Lett.* **57** (1986) 1199.

[36] A. Alloul, N. D. Christensen, C. Degrande, C. Duhr, and B. Fuks, *FeynRules 2.0 - A complete toolbox for tree-level phenomenology*, *Comput. Phys. Commun.* **185** (2014) 2250–2300, [arXiv:1310.1921].

[37] B. Ruijl, T. Ueda, and J. Vermaseren, *FORM version 4.2*, arXiv:1707.06453.

[38] P. Nogueira, *Automatic Feynman graph generation*, *J. Comput. Phys.* **105** (1993) 279–289.

[39] J. C. Criado, *MatchingTools: a Python library for symbolic effective field theory calculations*, *Comput. Phys. Commun.* **227** (2018) 42–50, [arXiv:1710.06445].

[40] T. Cohen, X. Lu, and Z. Zhang, *Functional Prescription for EFT Matching*, *JHEP* **02**

(2021) 228, [arXiv:2011.02484].

[41] V. Gherardi, D. Marzocca, and E. Venturini, *Matching scalar leptoquarks to the SMEFT at one loop*, *JHEP* **07** (2020) 225, [arXiv:2003.12525]. [Erratum: JHEP 01, 006 (2021)].

[42] M. Chala, G. Guedes, M. Ramos, and J. Santiago, *Running in the ALPs*, *Eur. Phys. J. C* **81** (2021), no. 2 181, [arXiv:2012.09017].

[43] M. Chala, G. Guedes, M. Ramos, and J. Santiago, *Towards the renormalisation of the Standard Model effective field theory to dimension eight: Bosonic interactions I*, *SciPost Phys.* **11** (2021) 065, [arXiv:2106.05291].

[44] M. Chala and J. Santiago, *Positivity bounds in the Standard Model effective field theory beyond tree level*, arXiv:2110.01624.

[45] M. Chala, A. Díaz-Carmona, and G. Guedes, *A Green's basis for the bosonic SMEFT to dimension 8*, arXiv:2112.12724.

[46] D. Zhang and S. Zhou, *Complete one-loop matching of the type-I seesaw model onto the Standard Model effective field theory*, *JHEP* **09** (2021) 163, [arXiv:2107.12133].

[47] A. Dedes and K. Mantzaropoulos, *Universal scalar leptoquark action for matching*, *JHEP* **11** (2021) 166, [arXiv:2108.10055].

[48] A. Crivellin, M. Kirk, T. Kitahara, and F. Mescia, *Correlating $t \to cZ$ to the W Mass and B Physics with Vector-Like Quarks*, arXiv:2204.05962.

[49] S. Bakshi Das, M. Chala, A. Díaz-Carmona, and G. Guedes, *Towards the renormalisation of the Standard Model effective field theory to dimension eight: Bosonic interactions II*, arXiv:2205.03301.

[50] G. Guedes and P. Olgoso, *A bridge to new physics: proposing new – and reviving old – explanations of a_μ*, arXiv:2205.04480.

[51] Y. Du, X.-X. Li, and J.-H. Yu, *Neutrino seesaw models at one-loop matching: Discrimination by effective operator*, arXiv:2201.04646.

[52] X. Li, D. Zhang, and S. Zhou, *One-loop matching of the type-II seesaw model onto the Standard Model effective field theory*, *JHEP* **04** (2022) 038, [arXiv:2201.05082].

[53] M. Jiang, N. Craig, Y.-Y. Li, and D. Sutherland, *Complete one-loop matching for a singlet scalar in the Standard Model EFT*, *JHEP* **02** (2019) 031, [arXiv:1811.08878]. [Erratum: JHEP 01, 135 (2021)].

[54] U. Haisch, M. Ruhdorfer, E. Salvioni, E. Venturini, and A. Weiler, *Singlet night in Feynman-ville: one-loop matching of a real scalar*, *JHEP* **04** (2020) 164, [arXiv:2003.05936]. [Erratum: JHEP 07, 066 (2020)].

[55] R. Alonso, H.-M. Chang, E. E. Jenkins, A. V. Manohar, and B. Shotwell, *Renormalization group evolution of dimension-six baryon number violating operators*, *Phys. Lett. B* **734** (2014) 302–307, [arXiv:1405.0486].

[56] E. E. Jenkins, A. V. Manohar, and P. Stoffer, *Low-Energy Effective Field Theory below the Electroweak Scale: Anomalous Dimensions*, *JHEP* **01** (2018) 084, [arXiv:1711.05270].

[57] E. E. Jenkins, A. V. Manohar, and P. Stoffer, *Low-Energy Effective Field Theory below the Electroweak Scale: Operators and Matching*, *JHEP* **03** (2018) 016, [arXiv:1709.04486].

[58] W. Dekens and P. Stoffer, *Low-energy effective field theory below the electroweak scale:*

matching at one loop, *JHEP* **10** (2019) 197, [arXiv:1908.05295].

[59] J. Aebischer, A. Crivellin, M. Fael, and C. Greub, *Matching of gauge invariant dimension-six operators for $b \to s$ and $b \to c$ transitions*, *JHEP* **05** (2016) 037, [arXiv:1512.02830].

[60] J. Aebischer and J. Kumar, *Flavour violating effects of Yukawa running in SMEFT*, *JHEP* **09** (2020) 187, [arXiv:2005.12283].

[61] D. M. Straub, *flavio: a Python package for flavour and precision phenomenology in the Standard Model and beyond*, arXiv:1810.08132.

[62] J. Aebischer, J. Kumar, P. Stangl, and D. M. Straub, *A Global Likelihood for Precision Constraints and Flavour Anomalies*, *Eur. Phys. J. C* **79** (2019), no. 6 509, [arXiv:1810.07698].

[63] J. Aebischer, M. Fael, C. Greub, and J. Virto, *B physics Beyond the Standard Model at One Loop: Complete Renormalization Group Evolution below the Electroweak Scale*, *JHEP* **09** (2017) 158, [arXiv:1704.06639].