Computer Physics Communications 300 (2024) 109198

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Computer Programs in Physics

AutoEFT: Automated operator construction for effective field theories

Robert V. Harlander, Magnus C. Schaaf *

Institute for Theoretical Particle Physics and Cosmology, RWTH Aachen University, 52056 Aachen, Germany

ARTICLE INFO

Dataset link: 10.17632/z8xm2hpbsp.1

Keywords:
EFT
SMEFT
Operator basis

ABSTRACT

The program `AutoEFT` is described. It allows one to generate Effective Field Theories (EFTs) from a given set of fields and symmetries. Allowed fields include scalars, spinors, gauge bosons, and gravitons. The symmetries can be local or global Lie groups based on $U(1)$ and $SU(N)$. The mass dimension of the EFT is limited only by the available computing resources. The operators are stored in a compact, human and machine-readable format. Aside from the program itself, we provide input files for EFTs based on the Standard Model and a number of its extensions. These include additional particles and symmetries, EFTs with minimal flavor violation, and gravitons.

Program summary

Program title: `AutoEFT`

CPC Library link to program files: https://doi.org/10.17632/z8xm2hpbsp.1

Developer's repository link: https://gitlab.com/auto_eft/autoeft

Licensing provisions: MIT license

Programming language: `Python`

Nature of problem: The *bottom-up* construction of an Effective Field Theory (EFT) which describes physics below a certain energy scale Λ requires obtaining a set of operators, composed of fields with mass $m \ll \Lambda$, that are invariant under certain symmetries. One is primarily interested in complete sets of independent operators, called *operator bases*. Their construction for a given mass dimension of the operators is nontrivial due to algebraic and kinematic relations that may render different operators redundant. Except for the lowest mass dimensions, the number of operators is so large that the task of constructing an explicit EFT operator basis requires a high degree of automation on a computer. In addition, an automated approach will allow one to immediately take into account newly postulated or discovered light particles beyond the Standard Model.

Solution method: Based on the group theoretical techniques and concepts established in Refs. [1–9] and in particular Refs. [10,11], we developed the program `AutoEFT`, capable of constructing a non-redundant on-shell operator basis for general EFTs and arbitrary mass dimension. Provided a suitable *model file*, the respective operator basis is generated explicitly, including contractions of the symmetry group indices, in a fully automated fashion. Due to the generality of the algorithm, it can be applied to a variety of low-energy scenarios. The underlying low-energy theory is encoded in a model file which defines the symmetries and the field content. The fairly simple format enables the user to compose their own model files and to construct the respective operator basis with minimal effort.

Additional comments including restrictions and unusual features: In its current form, `AutoEFT` is restricted to theories including particles with spin 0, 1/2, 1, and 2, where the latter two are considered massless. In addition, `AutoEFT` only constructs operators that mediate proper interactions, meaning that any operator must be composed of at least three fields. The internal symmetries must be given as factors of $U(1)$ and $SU(N)$ groups. In principle, operator bases can be generated for any mass dimension, which is, however, limited by the available computing resources.

References

[1] Y. Shadmi, Y. Weiss, Effective field theory amplitudes the on-shell way: scalar and vector couplings to gluons, J. High Energy Phys. 02 (2019) 165, arXiv:1809.09644 [hep-ph].

☆ The review of this paper was arranged by Prof. Z. Was.
☆☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: magnus.schaaf@rwth-aachen.de (M.C. Schaaf).

https://doi.org/10.1016/j.cpc.2024.109198
Received 6 October 2023; Received in revised form 3 March 2024; Accepted 5 April 2024
Available online 12 April 2024

[2] B. Henning, T. Melia, Conformal-helicity duality & the Hilbert space of free CFTs, arXiv:1902.06747 [hep-th].
[3] T. Ma, J. Shu, M.-L. Xiao, Standard model effective field theory from on-shell amplitudes, Chin. Phys. C 47 (2023) 023105, arXiv:1902.06752 [hep-ph].
[4] B. Henning, T. Melia, Constructing effective field theories via their harmonics, Phys. Rev. D 100 (2019) 016015, arXiv:1902.06754 [hep-ph].
[5] R. Aoude, C.S. Machado, The Rise of SMEFT on-shell amplitudes, J. High Energy Phys. 12 (2019) 058, arXiv:1905.11433 [hep-ph].
[6] R.M. Fonseca, Enumerating the operators of an effective field theory, Phys. Rev. D 101 (2020) 035040, arXiv:1907.12584 [hep-ph].
[7] G. Durieux, T. Kitahara, Y. Shadmi, Y. Weiss, The electroweak effective field theory from on-shell amplitudes, J. High Energy Phys. 01 (2020) 119, arXiv:1909.10551 [hep-ph].
[8] A. Falkowski, Bases of massless EFTs via momentum twistors, arXiv:1912.07865 [hep-ph].
[9] G. Durieux, C.S. Machado, Enumerating higher-dimensional operators with on-shell amplitudes, Phys. Rev. D 101 (2020) 095021, arXiv:1912.08827 [hep-ph].
[10] H.-L. Li, Z. Ren, J. Shu, M.-L. Xiao, J.-H. Yu, Y.-H. Zheng, Complete set of dimension-eight operators in the standard model effective field theory, Phys. Rev. D 104 (2021) 015026, arXiv:2005.00008 [hep-ph].
[11] H.-L. Li, Z. Ren, M.-L. Xiao, J.-H. Yu, Y.-H. Zheng, Complete set of dimension-nine operators in the standard model effective field theory, Phys. Rev. D 104 (2021) 015025, arXiv:2007.07899 [hep-ph].

Contents

1. Introduction

The mass of the Higgs boson is in a region where the Standard Model (SM) remains free of theoretical inconsistencies up to very large mass scales [1–4]. Disregarding arguments of naturalness which have lost much of their persuasive power due to the absence of any new particle discoveries in the TeV range, one may have to face the possibility that on-shell discoveries of particles belong to the past, and fundamental physics beyond the Standard Model will manifest itself at (current or future) particle colliders only through virtual effects [5].

Fortunately, such effects can be parameterized in a systematic way in terms of Effective Field Theories (EFTs). Ideally, the free parameters of an EFT, the Wilson coefficients, or characteristic subsets thereof, can be determined experimentally via precision measurements. Comparison to theoretical calculations of these coefficients via matching to theoretical models of the heavy physics could lead to new fundamental insights about the nature of UV physics.[1]

An EFT is based on the field content of a renormalizable Lagrangian $\mathcal{L}_{\leq 4}$, and incorporates effects up to order $(E/\Lambda)^N$ in processes at energies E, where Λ is the scale of new physics. The corresponding effective Lagrangian can be written as

$$\mathcal{L} = \mathcal{L}_{\leq 4} + \sum_{d=5}^{N+4} \sum_{n} \frac{C_n^{(d)}}{\Lambda^{d-4}} \mathcal{O}_n^{(d)}\,, \tag{1}$$

where the higher-dimensional operators $\mathcal{O}_n^{(d)}$ are composed of all fields of $\mathcal{L}_{\leq 4}$, and $C_n^{(d)}$ are the Wilson coefficients. For example, if $\mathcal{L}_{\leq 4}$ is the SM Lagrangian, then the higher-dimensional operators are composed of all SM fields, and $\mathcal{L}$ is referred to as the Standard Model Effective Field Theory (SMEFT) Lagrangian.

In a top-down approach, the effective operators follow from a UV-complete theory $\mathcal{L}_{\text{UV}}$ by integrating out the heavy degrees of freedom in the path integral of the generating functional. Such an approach is pursued in the Universal One-Loop Effective Action (UOLEA), for example, which also provides the Wilson coefficients in terms of the parameters of $\mathcal{L}_{\text{UV}}$ [6–10].

More common, however, is a bottom-up approach, which will be adopted in this paper. Here, one constructs all higher-dimensional operators by combining the fields of the low-energy theory $\mathcal{L}_{\leq 4}$ in such a way that they obey all symmetry constraints. At the same time, however, one requires the set of operators to be non-redundant in order to ensure that the Wilson coefficients are well-defined. Redundancies among the set of operators can arise from several sources. First, since total derivatives in the Lagrangian do not contribute to the action, operators could be related by integration-by-parts (IbP) identities. Second, operators could be linearly dependent due to algebraic relations such as Fierz or Schouten identities. Third, higher-dimensional operators that vanish due to equations-of-motion (EoMs) can be eliminated from the EFT by field redefinitions in the path integral of the generating functional [11,12]. Finally, operators could be related by permutations of the fields transforming in equal representations [13].

Obviously, the EFT which is most relevant from a phenomenological point of view is SMEFT. In fact, the dimension-five operators are strong candidates for being the source of neutrino masses [14]. The multiple attempts needed to arrive at the SMEFT-bases at dimension six and seven testify to the complexity of constructing a complete and non-redundant set of operators, despite the fact that their numbers are still quite manageable (84 and 30, respectively, for a single generation of fermions) [15–18].[2] Towards higher mass dimension, this number increases roughly exponentially. It can actually be computed exactly using Hilbert-series techniques [19–25], but also by more direct methods [26–28].[3] Despite the fact that, for three generations of fermions, the number of operators in the SMEFT basis at mass dimensions eight and nine already amounts to 44807 and 90456, respectively, it was still possible to construct them by largely manual efforts [30,31]. Nevertheless, an algorithmic procedure clearly becomes desirable.

Important steps towards the systematic construction of EFT operators were made in Refs. [13,32–40]. A complete algorithm was presented in Refs. [41,42] and used to construct the SMEFT basis at mass dimensions eight and nine. Its implementation is available as a `Mathematica` package [43]. In Ref. [44], we reported on an independent implementation of that algorithm and used it to derive for the first time the SMEFT operator bases at mass dimensions 10, 11, and 12. The current paper accompanies the publication of the associated computer program, named `AutoEFT`.[4] It is available as open source under the MIT license,[5] is based on `Python`,[6] and uses only publicly available software libraries, in particular `SageMath`.[7] Since the algorithm of Refs. [41,42] is not specific to SMEFT, it is possible to use `AutoEFT` also in extended theories with additional light particles beyond the SM spectrum (see, e.g., Refs. [45–52]). For example, Ref. [44] also includes the operators of the gravity-extension of SMEFT (GRSMEFT) [53] up to mass dimension 12.

This paper provides an introduction to `AutoEFT`, describing the necessary notation, the preparation of the input file, the commands to generate the operator basis, and the format of the output files. Section 2 introduces the theoretical and notational background required to interpret the `AutoEFT` input and output. The installation of `AutoEFT` is described in Section 3. Section 4 explains the structure of the model file to be processed by `AutoEFT`, and provides comprehensive examples for various models. The operator construction using `AutoEFT` is showcased in Section 5, including a discussion on the output format, as well as `AutoEFT`'s current limitations. Section 6 contains examples on how `AutoEFT` can further process the output. In addition, we include a reference manual in the appendix, which can be used to look up particular features or specifications related to the usage of `AutoEFT`.

2. Preliminaries

For fixed values of the Wilson coefficients and the parameters of the low-energy theory, an EFT can be considered as a vector in the space of all higher-dimensional operators. `AutoEFT` constructs a basis in this space of operators for a fixed, but in principle arbitrary value of d. In doing so, it takes into account the constraints arising from external (Lorentz) and internal symmetries. It ensures that the basis is non-redundant, meaning that no two operators are interrelated through EoMs, IbP or algebraic identities.

The field content and the symmetry groups of the low-energy theory $\mathcal{L}_{\leq 4}$ are supplied to `AutoEFT` via an input file, referred to as *model file* in the following. Its detailed structure will be defined in Section 4 and Appendix C. In this section, we provide the notational background for its contents.

`AutoEFT` allows for particles with spin 0, 1/2, 1, and 2 in the spectrum of $\mathcal{L}_{\leq 4}$.[8] For spin 0 and spin 1/2, it makes no difference for the

[1] Dubbed the "Cinderella approach" in Ref. [5].

[2] See Section 5.3 concerning the counting of operators.

[3] See also Ref. [29] for a summary of EFT software tools.

[4] https://gitlab.com/auto_eft/autoeft.

[5] https://spdx.org/licenses/MIT.html.

[6] https://www.python.org/.

[7] https://www.sagemath.org.

[8] We use "fields" and "particles" interchangeably in this paper.

Table 1
The list of irreducible representations of the Lorentz group supported by AutoEFT. Each representation is associated with a placeholder symbol for the field, and a unique value for the helicity h.

field	(j_l, j_r)	h	name
ϕ	$(0,0)$	0	scalar
ψ_{L}	$(1/2,0)$	$-1/2$	left-handed spinor
ψ_{R}	$(0,1/2)$	$+1/2$	right-handed spinor
F_{L}	$(1,0)$	-1	left-handed field-strength tensor
F_{R}	$(0,1)$	$+1$	right-handed field-strength tensor
C_{L}	$(2,0)$	-2	left-handed Weyl tensor
C_{R}	$(0,2)$	$+2$	right-handed Weyl tensor

construction of an EFT [47] whether they are massive or massless, and thus also for AutoEFT. Higher-spin particles represented by vector or tensor fields are currently restricted to the massless case though. Note that this is in line with SMEFT, which is formulated in the unbroken phase of the SM Lagrangian. The massive vector bosons are recovered by performing the electroweak symmetry breaking in SMEFT explicitly. Furthermore, since scalar and spinor fields are allowed to be massive, AutoEFT can also be used to generate EFTs in which all massive vector bosons are integrated out (e.g., Low-Energy Effective Field Theory (LEFT)/Weak Effective Theory (WET), parameterizing effects between the electroweak scale and Λ_{QCD}). For AutoEFT, a particle is thus uniquely identified by its $U(1)$ charges, the representations according to which it transforms under the Lorentz and the non-abelian internal symmetry groups, and a possible generation index.

The irreducible representations of the Lorentz group—which can be identified with $SU(2)_l \times SU(2)_r$ for our purpose—are characterized by (j_l, j_r), where $j_{l/r}$ are non-negative integers or half-integers. The most important irreducible representations are given by $(0,0)$, $(1/2,0)$, and $(1,0)$, corresponding to scalars ϕ, left-handed Weyl spinors $\psi_{\mathrm{L}\alpha}$, and self-dual 2-forms $F_{\mathrm{L}\alpha\beta}$. For simplicity, we will refer to the latter also as “left-handed field-strength tensors” in the following. In addition, we consider self-dual (“left-handed”) Weyl tensors $C_{\mathrm{L}\alpha\beta\gamma\delta}$ transforming as $(2,0)$ which are required for gravity. Since $j_r = 0$ for all of these “elemental” representations, they can also be characterized by their *helicity*

$$h = j_r - j_l\,. \tag{2}$$

The conjugate (“right-handed”) fields $\psi_{\mathrm{R}}{}^{\dot\alpha}$, $F_{\mathrm{R}}^{\dot\alpha\dot\beta}$, and $C_{\mathrm{R}}^{\dot\alpha\dot\beta\dot\gamma\dot\delta}$ transform as $(0,1/2)$, $(0,1)$ and $(0,2)$ under the Lorentz group and thus carry the negative helicity of the corresponding left-handed fields. Here and in the following, α, β, … and $\dot\alpha$, $\dot\beta$, … denote fundamental $SU(2)_l$ and $SU(2)_r$ spinor indices, respectively, unless indicated otherwise.[9]

All other fields which occur in common Quantum Field Theories (QFTs) transform in representations which can be composed of these elemental representations $(|h|,0)$ and their conjugate versions $(0,|h|)$. For example, the bispinor and the field-strength tensor transform in the direct sums of the left- and right-handed Weyl spinor representations $(1/2,0) \oplus (0,1/2)$, and the self- and anti-self-dual 2-form representations $(1,0) \oplus (0,1)$, respectively. In AutoEFT, however, one simply defines each irreducible component as a separate field. Concrete examples will be given in Section 4.

The output of AutoEFT is thus formulated in terms of the objects summarized in Table 1, as well as the covariant derivative $D_\alpha^{\dot\alpha}$. The action of n derivatives on a field Φ is understood in the AutoEFT output as the combined object

$$(D^n\Phi)^{(\dot\alpha\dot\beta\dots)}_{(\alpha\beta\dots)} \sim (D^n\Phi)^{\dot\alpha\dot\beta\dots}_{\alpha\beta\dots} + (D^n\Phi)^{\dot\alpha\dot\beta\dots}_{\beta\alpha\dots} + (D^n\Phi)^{\dot\beta\dot\alpha\dots}_{\alpha\beta\dots} + (D^n\Phi)^{\dot\beta\dot\alpha\dots}_{\beta\alpha\dots} + \dots\,, \tag{3}$$

where the dotted and undotted indices are separately symmetrized. For each term on the right hand side of Eq. (3), the first n pairs of dotted and undotted indices belong to the covariant derivatives, whereas all remaining indices are part of the field Φ.

In order to facilitate the translation of the operators into the more common notation of bispinors Ψ, field-strength tensors $F^{\mu\nu}$, Weyl tensors $C^{\mu\nu\rho\sigma}$, and covariant derivatives D^μ, with Lorentz four-vector indices $\mu, \nu, \dots$, we collect the necessary relations in Appendix A.

Concerning internal symmetries, AutoEFT allows for local and global $U(1)$ and $SU(N)$ groups.[10] All fields are assumed to transform in an irreducible representation of the internal symmetry groups. AutoEFT requires that each $U(1)$ charge of a field is given by a (fractional) multiple of some elementary charge (which does not need to be specified further). In the model file, the $U(1)$ charges are thus defined by rational numbers. Examples will be given in Section 4.

The irreducible representations of $SU(N)$ are encoded via their one-to-one correspondence to Young diagrams, which can be represented by lists of non-increasing positive integers (also referred to as *integer partitions* in the following).[11] For example, the fundamental representation of $SU(N)$ can be specified as

$$\square \quad \sim \quad [1]\,. \tag{4}$$

For the anti-fundamental representation, it is

$$N-1\left\{\begin{array}{c}\square\\ \square\\ \vdots\\ \square\end{array}\right. \quad \sim \quad \underbrace{[1,1,\dots,1]}_{N-1} \equiv [1^{N-1}]\,, \tag{5}$$

and for the adjoint representation, the correspondence is

$$N-1\left\{\begin{array}{l}\square\square\\ \square\\ \vdots\\ \square\end{array}\right. \quad \sim \quad [2,\underbrace{1,\dots,1}_{N-2}] \equiv [2,1^{N-2}]\,, \tag{6}$$

where we used a common short-hand notation for integer partitions with long sequences of the same number.

In the SM context, it is more common to refer to the irreducible representations of $SU(N)$ by their dimensionality rather than by integer partitions or Dynkin labels. For example, $\mathbf{3}$, $\bar{\mathbf{3}}$, and $\mathbf{8}$ denote the fundamental, anti-fundamental, and adjoint representations of $SU(3)$, respectively. However, this characterization becomes ambiguous in the case where several non-equivalent irreducible representations with the same dimensionality exist (e.g., $SU(3)$ has four 15-dimensional irreducible representations: $[4], [4,4], [3,1], [3,2]$). Such a characterization is, therefore, not suitable for a systematic approach, and we refrain from it in the context of AutoEFT.

Similar to the Lorentz group, the fields composing the operators in the output of AutoEFT carry only fundamental indices of the internal symmetry groups. For fields transforming in the anti-fundamental or adjoint representations, one can translate this directly to a more common notation using the relations provided in Appendix A. While this is sufficient for SMEFT, it may be desirable to translate other representations in extended theories with light fields. In this case, the corresponding Clebsch-Gordan coefficients need to be taken into account.[12]

---

[9] We do not consider the irreducible representation $(1/2,1/2)$ corresponding to Lorentz four-vectors explicitly, because we assume that vector fields always arise as gauge fields and thus appear only as part of a field-strength tensor or the covariant derivative. For more details, see the subsequent main text.

[10] Concerning $U(N)$, see Section 4.4.

[11] AutoEFT also supports the characterization of these representations by Dynkin labels; see Section 4.2.2, for a concrete example.

[12] For example, the sextet representation $\square\square \sim [2]$ of $SU(3)$ can be related to the symmetric product of two fundamental representations using the Clebsch-Gordan coefficients computed in Ref. [54]. Consequently, a field transforming in this representation can be denoted either by one sextet index or two fundamental indices, related by the Clebsch-Gordan coefficients.

3. Installation

AutoEFT is implemented in Python and makes use of several functions provided by the free open-source mathematics software system SageMath. Since intermediate expressions during the construction procedure can become exceedingly large, certain algebraic operations are passed to FORM [55,56]. All remaining dependencies are third-party Python libraries and are included for the user's convenience, such as input validation and console markup. For a standard installation of AutoEFT, the following software needs to be installed on the system:

Python (*version 3.8* or later)
: This requirement is fulfilled by default in most cases. There is either a system wide Python installation that is also used by SageMath, or SageMath does come with its own version of the Python interpreter. If the installation is done via the conda/mamba package management system, a suitable Python version is automatically included in the virtual environment.

SageMath (*version 9.3* or later)
: The SageMath library only needs to be installed explicitly if AutoEFT is *not* installed using the conda/mamba package management system.[13] Installation details can be found at https://doc.sagemath.org/html/en/installation/index.html.

FORM (*version 4.3* or later)
: The FORM home page can be found at https://www.nikhef.nl/~form/. To use AutoEFT together with FORM, make sure that there is an executable named form on the system path or on a path specified by the environment variable AUTOEFT_PATH (cf. Appendix B.2).

In case of problems with the installation, the user is advised to contact the authors via email or the AutoEFT repository, see Footnote 4. The latter also collects several potential installation issues and their resolution.

3.1. Installing AutoEFT from PyPI

This is the recommended installation method. It requires an existing and running version of SageMath though. Given that, AutoEFT and its dependencies can be installed from the *Python Package Index (PyPI)*[14] by simply running:[15]

```
sage -pip install autoeft
```

3.2. Installing AutoEFT from conda-forge

Since the SageMath distribution is part of the *conda-forge* [57] channel, there is no requirement for a prior installation. Using the conda[16] package manager, AutoEFT and its dependencies can be installed from the *conda-forge* channel by running:

```
conda install autoeft -c conda-forge
```

If the mamba[17] package manager is used instead, the *conda-forge* channel is enabled by default. Hence, AutoEFT and its dependencies can be installed by running:

```
mamba install autoeft
```

3.3. Building AutoEFT from source code

To build AutoEFT from its source code, make sure the latest version of the Python Packaging Authority's build[18] is installed. The distribution packages can then be generated by running:

```
git clone \
  https://gitlab.com/auto_eft/autoeft.git autoeft
cd autoeft/
python -m build
```

Note that the last command must be executed in the directory containing the file pyproject.toml. After this, there should be two archive files in the newly created dist/ directory: The source distribution autoeft-1.0.0.tar.gz as well as the build distribution autoeft-1.0.0-py3-none-any.whl. To install the local package, run:

```
sage -pip install \
  dist/autoeft-1.0.0-py3-none-any.whl
```

As AutoEFT is developing, the version number will have to be replaced accordingly in these commands, of course.

3.4. Validating the installation

A successful installation of AutoEFT can be validated by running

```
autoeft check
```

In the current version, this constructs the SMEFT operator basis for mass dimension six and compares it to a pre-constructed result.

4. The model file

To construct an EFT operator basis, the user must define a model describing the relevant details of the low-energy theory. This is done via the *model file* which encodes all information about the symmetries and field content of the model.[19] A detailed description of all keywords and their type can be found in Appendix C.

4.1. Basic structure

A valid model file has to contain a minimal set of keywords (simply referred to as *keys* in the following), which must be assigned appropriate values. In particular, every model file must contain the key name, set to a valid string that identifies the model. The other required keys are symmetries and fields. These three keys are sufficient to define a valid model file that AutoEFT can process. For example, in Listing 1, both symmetries and fields are set to the empty set '{ }', corresponding to the trivial model without any fields.[20]

[13] Although there is some effort towards modularizing SageMath into separate distributions, the packages required by AutoEFT are only available in the complete library for now. We advice to either install SageMath by "hand", or to use the conda/mamba package management system, which installs SageMath automatically in a virtual environment.

[14] https://pypi.org/.

[15] On macOS using Homebrew, it may be necessary to precede this statement by PYTHONEXECUTABLE=</path/to/sage> with the proper path to the SageMath executable inserted. In addition, it may be necessary to add the path to SageMath's executables to the $PATH environment variable.

[16] https://conda.io/.

[17] https://github.com/mamba-org/mamba.

[18] https://pypi.org/project/build/.

[19] Technically, the format of the model file is YAML (https://yaml.org/); all required specifications will be implicitly discussed below though.

[20] We adopt the convention that variable input provided by the user is set in type-writer font and surrounded by single quotes in the main text. In the code listings, they are set in black color. The single quotes are missing for fixed code words that are not to be changed by the user (blue color in the listings).

```
# AutoEFT model file
name: Minimal Model
symmetries: {}
fields: {}
```

Listing 1: Minimal data required in a model file.

As a non-trivial model, let us consider scalar Quantum Electrodynamics (QED), i.e. a $U(1)$ gauge theory of a charged scalar field. The $U(1)$ symmetry is implied by adding the sub-key `u1_groups` to `symmetries`, as displayed in Listing 2.

```
symmetries:
  u1_groups:
    QED: {}
```

Listing 2: Symmetry definition of the scalar QED model file.

Note that the actual symmetry, identified by the string '`QED`', has been added as another sub-key to `u1_groups`. In principle, we could specify additional attributes for this group (e.g., an allowed violation, a residual charge, or a LaTeX symbol; see Appendix C) by assigning it a non-trivial value. For our purposes, however, this is not necessary and we assign to it the empty set '{}'.

Next, we include a single complex scalar field ϕ in the model, by adding the entry '`phi`' to `fields` as shown in Listing 3. Again, the name '`phi`' is arbitrary. To define the transformation properties of the field under the symmetry groups, the key `representations` must be added to '`phi`'. In our case, there is only one symmetry group, so we add the entry '`QED: -1`' to `representations`, which means that ϕ carries one negative unit of the elementary $U(1)$ charge, see line 9 in Listing 3. For every field defined in the model file, `AutoEFT` automatically takes into account the conjugate version and denotes it by appending the symbol '+' to the original field name. Thus, in our example, the conjugate field $\phi^\dagger$ is taken into account automatically by `AutoEFT`, and it will be denoted by '`phi+`' in the output.[21]

```
6 fields:
7   phi:
8     representations:
9       QED: -1
```

Listing 3: Definition of the scalar field in the model file.

To make this theory an actual gauge theory, the $U(1)$ gauge boson has to be defined as well. Gauge bosons can appear in two instances: encoded in field strength tensors or as part of the covariant derivative. The latter is automatically included by `AutoEFT`, while the former is decomposed into two separate fields which transform in irreducible representations of the Lorentz group, see Section 2. The first one, '`FL`' $\hat{=}$ F_L, transforming as $(1,0)$, can be defined in the model file by adding another entry to `fields`; see Listing 4. By default, the Lorentz group is identified by the literal string '`Lorentz`',[22] and it is assigned the helicity value $h=-1$ in this case. The second component of the QED field-strength tensor, $F_\mathrm{R} \in (0,1)$, is again included automatically, as it is the conjugate of F_L ($F_\mathrm{R} = F_\mathrm{L}^\dagger$). Note that we did not explicitly have to specify the helicity for the scalar field ϕ in Listing 3, nor the QED charge for the field strength F_L in Listing 4. If unspecified, `AutoEFT` assumes that the fields are singlets under the corresponding symmetry groups, which means that ϕ is defined as a Lorentz scalar, and F_L does not carry a $U(1)$ charge, as desired.

```
  FL:
    representations:
      Lorentz: -1
```

Listing 4: Definition of the gauge boson in the model file.

[21] The exception to this are fields all of whose representations are real (or combine to form a real representation). In this case, no conjugate field is generated. One can also prevent `AutoEFT` from including the conjugate field—for whatever reason one may have—by using the `conjugate` property; see Appendix C.

[22] This can be overwritten by the user in the model file; see Appendix C.

Combining the symmetry definition in Listing 2 and the field content definitions in Listings 3 and 4—and giving the model a suitable name—results in the entire model file, displayed in Listing 5.

```
# AutoEFT model file
name: sQED-EFT
symmetries:
  u1_groups:
    QED: {}
fields:
  phi:
    representations:
      QED: -1
  FL:
    representations:
      Lorentz: -1
```

Listing 5: Scalar QED model file.

Note that, in the model file, an explicit association of the field strength tensor '`FL`' (or '`FR`') to the gauge group '`QED`' is not necessary. Its role as a gauge field will originate from the proper interpretation of the covariant derivative in the resulting operators. If it includes the photon field, the symmetry is local; otherwise, it is a global symmetry, and '`FL`' represents a vector boson which transforms as a singlet under the symmetry group. It will still couple to the fermion in higher-dimensional operators.

4.2. *Realistic examples*

In this section, more realistic examples will be considered, starting from QED, generalizing to Quantum Chromodynamics (QCD), and finally the SM. In the course of this, we will discuss the definition of spinors and non-abelian $SU(N)$ symmetry groups in the model file.

4.2.1. *QED*

To promote the example of scalar QED from the previous section to actual QED, one needs to introduce Dirac fermions. As described in Section 2, the Lorentz representation of bispinors is given by $(1/2,0) \oplus (0,1/2)$. The model file for QED with a single charged electron can thus be written as shown in Listing 6. Here, e_L $\hat{=}$ '`eL`' and e_R $\hat{=}$ '`eR`' denote left- and right-handed Weyl spinors $e_\mathrm{L} \in (1/2,0)$ and $e_\mathrm{R} \in (0,1/2)$ with helicity $-1/2$ and $+1/2$, respectively. In QED, both of them carry the same charge, and thus can be considered as components of the same Dirac spinor

$$\Psi = \begin{pmatrix} e_\mathrm{L} \\ e_\mathrm{R} \end{pmatrix} . \tag{7}$$

```
# AutoEFT model file
name: QED-EFT
description: Effective Field Theory of QED interactions

symmetries:
  u1_groups:
    QED: {}

fields:
  eL: # EL = (eL, 0)^T, ELbar = (0, eL+)
    representations:
      Lorentz: -1/2
      QED: -1
  eR: # ER = (0, eR)^T, ERbar = (eR+, 0)
    representations:
      Lorentz: 1/2
      QED: -1
  FL:
    representations:
      Lorentz: -1
```

Listing 6: QED model file.

`AutoEFT` by default also takes into account the conjugate Weyl spinors '`eL+`' $\hat{=}$ $e_\mathrm{L}^\dagger \in (0,1/2)$ and '`eR+`' $\hat{=}$ $e_\mathrm{R}^\dagger \in (1/2,0)$. Note that for a Dirac spinor $e_\mathrm{L}^\dagger \neq e_\mathrm{R}$, which is why both e_L and e_R need to be defined in the model file. In contrast, a Majorana spinor is represented in the model file by a single Weyl spinor which transforms in real representations of all internal symmetries.

In the literature it is quite common to adopt the *all-left* chirality notation for the fundamental building blocks of an EFT. In this convention, all Weyl spinors are defined to be left-handed, so that the index "L" can be dropped. The right-handed components are then acquired by conjugation. In the above example this would mean that one defines 'e' $\hat{=}\ e \equiv e_\text{L}$ and its charge conjugate 'eC' $\hat{=}\ e_\mathbb{C} \equiv e_\text{R}^\dagger$. One could thus define QED in AutoEFT also by replacing lines 10–17 in Listing 6 by the content of Listing 7.

```
10  e: # EL = (e, 0)^T, ELbar = (0, e+)
11    representations:
12      Lorentz: -1/2
13      QED: -1
14  eC: # ER = (0, eC+)^T, ERbar = (eC, 0)
15    representations:
16      Lorentz: -1/2
17      QED: 1
```

Listing 7: All-left notation for the electron.

4.2.2. QCD

To generalize the example of QED to a non-abelian theory like QCD, $SU(N)$ symmetries need to be introduced. They are defined in a similar way to $U(1)$ symmetries in the model file but require additional information like their degree N. A model file for QCD with a single quark flavor could be defined as displayed in Listing 8. The $SU(3)$ symmetry of QCD is imposed by the lines 5–8. Under the keyword `sun_groups`, all $SU(N)$ symmetry groups of the model are listed; here, we only have 'QCD', for which we specify the degree by the entry 'N: 3' (note the indentation of line 8).

```
1   # AutoEFT model file
2   name: QCD-EFT
3   description: Effective Field Theory of QCD interactions
4
5   symmetries:
6     sun_groups:
7       QCD:
8         N: 3
9
10  fields:
11    qL:
12      representations:
13        Lorentz: -1/2
14        QCD: [1]
15    qR:
16      representations:
17        Lorentz: 1/2
18        QCD: [1]
19    GL:
20      representations:
21        Lorentz: -1
22        QCD: [2,1]
```

Listing 8: QCD model file.

Lines 11–22 declare the field content of the model. Analogously to the example of QED discussed in Section 4.2.1, a Dirac quark spinor is implemented by specifying its left- and right-handed components, named $q_\text{L}\ \hat{=}$ 'qL' and $q_\text{R}\ \hat{=}$ 'qR' here. The fact that they transform in the fundamental representation of QCD is encoded by specifying the integer partition '[1]' in lines 14 and 18, cf. Eq. (4). As discussed above, AutoEFT automatically takes into account the corresponding conjugate fields 'qL+' $\hat{=}\ q_\text{L}^\dagger$ and 'qR+' $\hat{=}\ q_\text{R}^\dagger$ which transform in the anti-fundamental representation $[1,1]$ of QCD, cf. Eq. (5). Since the adjoint representation $[2,1]$ is real, only the left-handed component of the gluon field-strength tensor 'GL' must be defined in the model file explicitly, see lines 19–22 of Listing 8.

Instead of integer partitions, one may also use *Dynkin labels* to specify the irreducible representation of $SU(N)$ in which a field transforms. For AutoEFT, the difference is indicated by using round brackets instead of square ones. The fundamental and adjoint representations of $SU(3)$ are denoted by the Dynkin labels (10) and (11), respectively. Lines 14 and 18 of Listing 8 could thus also be written as 'QCD: (1,0)', for example, and line 22 as 'QCD: (1,1)'.[23] Internally, any Dynkin label is converted to the respective partition.

4.2.3. Standard Model

The previous sections provide all the information required to compose a model file for the entire SM in the unbroken phase—including all symmetries and fields. The transition to the broken phase can be performed at the level of the operators by appropriate replacements of the Higgs field.

The SM gauge group is given by $SU(3)\times SU(2)\times U(1)$ which can be defined in just a few lines in the model file, see lines 6–10 in Listing 9 below. Each gauge group is equipped with an associated multiplet of gauge bosons by defining the components 'GL' $\hat{=}\ G_\text{L}$, 'WL' $\hat{=}\ W_\text{L}$, and 'BL' $\hat{=}\ B_\text{L}$, respectively (cf. lines 13–23).

The matter fields of the SM come in five distinct representations. Taking the first generation of fermions as an example, they are characterized by the Weyl spinors

$$\text{'QL'}\ \hat{=}\ Q_\text{L}\,,\quad \text{'uR'}\ \hat{=}\ u_\text{R}\,,\quad \text{'dR'}\ \hat{=}\ d_\text{R}\,,\quad \text{'LL'}\ \hat{=}\ L_\text{L}\,,\quad \text{'eR'}\ \hat{=}\ e_\text{R} \tag{8}$$

and their Hermitian conjugate. Their representations w.r.t. the Lorentz and the SM gauge group are defined in lines 24–53 of the model file.[24]

In principle, the second and third generation of fermions could be implemented as separate copies of Eq. (8). More conveniently though, one may add the entry 'generations: 3' to every fermion declaration, see Listing 9. By using this option, AutoEFT will associate a generation index with these fields, which leads to a much more compact form of the output, of course. Note that, even though the sum over generation indices is not carried out explicitly in this case, the output does depend on the actual number of generations. This is because the external and internal symmetries may induce redundancies which depend on this number (see Ref. [13] for details).

To complete the SM, the complex Higgs doublet $H\ \hat{=}$ 'H' must be included as well. This is simply done by defining it as an $SU(2)$ doublet and assigning an appropriate hypercharge, see lines 54–57 of Listing 9.

The entire model file for the SM with three generations of fermions is then given by Listing 9.[25]

```
1   # AutoEFT model file
2   name: SMEFT
3   description: Standard Model Effective Field Theory
4
5   symmetries:
6     sun_groups:
7       SU3: {N: 3}
8       SU2: {N: 2}
9     u1_groups:
10      U1: {}
11
12  fields:
13    GL:
14      representations:
15        Lorentz: -1
16        SU3: [2,1]
17    WL:
18      representations:
19        Lorentz: -1
20        SU2: [2]
21    BL:
```

[23] This is not to be confused with the (j_r, j_l) notation for the representations of the Lorentz group defined in Section 2.

[24] In the supplementary model files, the electromagnetic charge Q is defined by the relation $Q = I_3 + Y$ where I_3 and Y are the 3rd component of weak-isospin and the $U(1)$-hypercharge, respectively.

[25] As supplementary material, we supply the model file `sm.yml`. Besides the information displayed in Listing 9, this file contains additional keywords which, however, only affect the LaTeX markup of the operators. For consistency with other literature and Ref. [44], we also supply the model file `all-left_sm.yml` that defines the fields in the all-left chirality convention (cf. Section 4.2.1).


```
    representations:
      Lorentz: -1
  QL:
    representations:
      Lorentz: -1/2
      SU3: [1]
      SU2: [1]
      U1: 1/6
    generations: 3
  uR:
    representations:
      Lorentz: 1/2
      SU3: [1]
      U1: 2/3
    generations: 3
  dR:
    representations:
      Lorentz: 1/2
      SU3: [1]
      U1: -1/3
    generations: 3
  LL:
    representations:
      Lorentz: -1/2
      SU2: [1]
      U1: -1/2
    generations: 3
  eR:
    representations:
      Lorentz: 1/2
      U1: -1
    generations: 3
  H:
    representations:
      SU2: [1]
      U1: 1/2
```

Listing 9: SM model file.

4.3. Extended models

After reading Sections 4.1 and 4.2, and optionally consulting Appendix C, the user should be able to assemble custom model files from scratch. However, AutoEFT offers an alternative approach of creating model files using the `sample-model` command. Running this command will print the content of a predefined SM model file to the standard output (e.g., the terminal). Therefore, a custom model can also be obtained by running the command

```
autoeft sample-model > custom.yml
```

and subsequently modifying the newly created file `custom.yml` as desired. Alternatively, the user may base the custom model on one of the sample model files supplied with this paper. In the following, we consider specific examples for extending the SM as the low-energy theory.

4.3.1. Additional particles

Ref. [58] defines a list of possible extensions of the SM by adding new particles. In order to illustrate the simplicity of preparing a specific model file for AutoEFT, we explicitly describe the necessary modifications of the SM model file for all examples provided in this paper. Each model file can also be found in the supplementary material of this paper, or in the AutoEFT repository, see Footnote 4. It allows one to reconstruct the operator bases provided in Ref. [58], and to extend them to higher mass dimension.

Quite in general, new particles can be included in the EFT construction by adding new entries under the keyword `fields` and assigning them appropriate representations of the existing symmetry groups. In the following examples, we only show the lines that need to be added to the very end of the default model file produced by the `sample-model` command. Following Ref. [58] and adopting their notation, let us first consider the addition of uncolored particles.

A scalar $\delta^+ \hat{=}$ ‘`del`’ which only carries one unit of the hyper charge and otherwise transforms as a singlet can be implemented as:

```
  del:
    representations:
      U1: 1
```

for example. Of course, other (rational) values of the hyper charge can be incorporated in an analogous way. For example, the doubly charged scalar named $\rho^{++} \hat{=}$ ‘`rho`’ in Ref. [58] is obtained from:

```
  rho:
    representations:
      U1: 2
```

Similarly, the complex scalar $SU(2)$-triplet $\Delta \hat{=}$ ‘`Del`’ can be added as:

```
  Del:
    representations:
      SU2: [2]
      U1: 1
```

and the left-handed fermion triplet $\Sigma \hat{=}$ ‘`Sig`’ is defined as:

```
  Sig:
    representations:
      Lorentz: -1/2
      SU2: [2]
```

For vector-like leptons of various charges $(V_{\mathrm{L,R}}, E_{\mathrm{L,R}}, N_{\mathrm{L,R}}) \hat{=}$ (‘`VL`’,‘`VR`’, ...), one also needs to define the right-handed components:

```
  VL:
    representations:
      Lorentz: -1/2
      SU2: [1]
      U1: -1/2
    generations: 3
  VR:
    representations:
      Lorentz: 1/2
      SU2: [1]
      U1: -1/2
    generations: 3
  EL:
    representations:
      Lorentz: -1/2
      U1: -1
    generations: 3
  ER:
    representations:
      Lorentz: 1/2
      U1: -1
    generations: 3
  NL:
    representations:
      Lorentz: -1/2
    generations: 3
  NR:
    representations:
      Lorentz: 1/2
    generations: 3
```

Finally, also higher representations of the gauge group can be accounted for. For example, the scalar $SU(2)$-quadruplet $\Theta \hat{=}$ ‘`The`’ is given by:

```
  The:
    representations:
      SU2: [3]
      U1: 3/2
```

New *colored* particles can be included in exactly the same way by assigning appropriate $SU(3)$ representations. Again, we only show the lines that need to be added to the very end of the default model file. In particular, the various versions of lepto-quarks defined in Ref. [58] can be implemented as:

Lepto-Quark ($\chi_1 \hat{=}$ ‘`chi1`’)

```
  chi1:
    representations:
      SU3: [1]
      SU2: [1]
      U1: 1/6
```

Lepto-Quark ($\varphi_1 \hat{=}$ ‘`phi1`’)

```
phi1:
  representations:
    SU3: [1]
    U1: 2/3
```

Lepto-Quark ($\chi_2 \;\widehat{=}$ ‘chi2’)

```
chi2:
  representations:
    SU3: [1]
    SU2: [1]
    U1: 7/6
```

Lepto-Quark ($\varphi_2 \;\widehat{=}$ ‘phi2’)

```
phi2:
  representations:
    SU3: [1]
    U1: -1/3
```

4.3.2. Additional gauge symmetries

Additional gauge groups can be added by simply including their definition under the keyword `symmetries`. In the following example, there are two new abelian gauge groups $U(1)'$ and $U(1)''$, extending the SM gauge group. Their respective gauge bosons are denoted by X and Y (corresponding to ‘XL’, ‘YL’ in the model file, plus the automatically included conjugate fields). In addition, global symmetries—like baryon- and lepton-number conservation—can be added in exactly the same way, with the only difference that there are no associated gauge bosons. In this example, each fermion gets assigned a specific baryon and lepton number and the resulting operators must conserve the total numbers exactly. Using the optional keys `violation` and `residual`, it would also be possible to allow for a certain degree of violation of the global $U(1)$ symmetries, see Appendix C.

The entire model file is displayed in Listing 10, including the `tex`, `tex_hc`, and `indices` keys that tell `AutoEFT` how to represent the symmetries, fields, and indices in LaTeX format; see Appendix C. New non-abelian gauge groups can be added in close analogy to the procedure described above.

```
# AutoEFT model file
name: U(1)-U(1)-SMEFT
description: U(1)' x U(1)'' extended Standard Model Effective
    ↪ Field Theory

symmetries:
  lorentz_group:
    tex: SO^+(1,3)
  sun_groups:
    SU3:
      N: 3
      tex: SU(3)
      indices: [a,b,c,d,e,f,g,h]
    SU2:
      N: 2
      tex: SU(2)
      indices: [i,j,k,l,m,n,p,q]
  u1_groups:
    U1:
      tex: U(1)
    U1p:
      tex: U(1)^\prime
    U1pp:
      tex: U(1)^{\prime\prime}
    Bno: {}
    Lno: {}

fields:
  GL:
    representations:
      Lorentz: -1
      SU3: [2,1]
    tex: G_{\mathrm{L}}
    tex_hc: G_{\mathrm{R}}
  WL:
    representations:
      Lorentz: -1
      SU2: [2]
    tex: W_{\mathrm{L}}
    tex_hc: W_{\mathrm{R}}
  BL:
    representations:
      Lorentz: -1
    tex: B_{\mathrm{L}}
    tex_hc: B_{\mathrm{R}}
  XL:
    representations:
      Lorentz: -1
    tex: X_{\mathrm{L}}
    tex_hc: X_{\mathrm{R}}
  YL:
    representations:
      Lorentz: -1
    tex: Y_{\mathrm{L}}
    tex_hc: Y_{\mathrm{R}}
  QL:
    representations:
      Lorentz: -1/2
      SU3: [1]
      SU2: [1]
      U1: 1/6
      Bno: 1/3
    generations: 3
    tex: Q_{\mathrm{L}}
  uR:
    representations:
      Lorentz: 1/2
      SU3: [1]
      U1: 2/3
      Bno: 1/3
    generations: 3
    tex: u_{\mathrm{R}}
  dR:
    representations:
      Lorentz: 1/2
      SU3: [1]
      U1: -1/3
      Bno: 1/3
    generations: 3
    tex: d_{\mathrm{R}}
  LL:
    representations:
      Lorentz: -1/2
      SU2: [1]
      U1: -1/2
      Lno: -1
    generations: 3
    tex: L_{\mathrm{L}}
  eR:
    representations:
      Lorentz: 1/2
      U1: -1
      Lno: -1
    generations: 3
    tex: e_{\mathrm{R}}
  H:
    representations:
      SU2: [1]
      U1: 1/2
    tex: H
```

Listing 10: $U(1)' \times U(1)''$ extended model file.

4.4. MFV model

Instead of considering the three generations of fermions as independent entities, one can also introduce so-called flavor symmetries. In these models, the approximate flavor symmetry of the SM—which is only broken by the Yukawa sector—is also imposed on the EFT. A prominent example is Minimal Flavor Violation (MFV) [59,60], which introduces a global $U(3)^5 \sim U(3)_Q \times U(3)_u \times U(3)_d \times U(3)_L \times U(3)_e$ flavor symmetry. Although `AutoEFT` does not support $U(N)$ symmetries directly, there is a Lie algebra isomorphism to $SU(N) \times U(1)$. Hence, MFV is realized by assigning an $SU(3)_f \;\widehat{=}$ ‘SU3<f>’ fundamental representation and a $U(1)_f \;\widehat{=}$ ‘U1<f>’ (<f>∈ {q,u,d,l,e}) charge of unity to every fermion:

$$Q \sim \square_{SU(3)_Q} \otimes 1_{U(1)_Q}\,, \quad u \sim \square_{SU(3)_u} \otimes 1_{U(1)_u}\,, \quad d \sim \square_{SU(3)_d} \otimes 1_{U(1)_d}\,, \quad L \sim \square_{SU(3)_L} \otimes 1_{U(1)_L}\,, \quad e \sim \square_{SU(3)_e} \otimes 1_{U(1)_e}\,. \tag{9}$$

```
# AutoEFT model file
name: MFV-SMEFT
description: Minimal Flavor Violation Standard Model Effective Field Theory

symmetries:
  sun_groups:
    SU3: {N: 3}
    SU2: {N: 2}
    SU3q: {N: 3}
    SU3u: {N: 3}
    SU3d: {N: 3}
    SU3l: {N: 3}
    SU3e: {N: 3}
  u1_groups: {U1: {}, U1q: {}, U1u: {}, U1d: {}, U1l: {}, U1e: {}}

fields:
  GL:
    representations: {Lorentz: -1, SU3: [2,1]}
  WL:
    representations: {Lorentz: -1, SU2: [2]}
  BL:
    representations: {Lorentz: -1}
  QL:
    representations: {Lorentz: -1/2, SU3: [1], SU2: [1], U1: 1/6, SU3q: [1], U1q: 1}
  uR:
    representations: {Lorentz: 1/2, SU3: [1], U1: 2/3, SU3u: [1], U1u: 1}
  dR:
    representations: {Lorentz: 1/2, SU3: [1], U1: -1/3, SU3d: [1], U1d: 1}
  LL:
    representations: {Lorentz: -1/2, SU2: [1], U1: -1/2, SU3l: [1], U1l: 1}
  eR:
    representations: {Lorentz: 1/2, U1: -1, SU3e: [1], U1e: 1}
  H:
    representations: {SU2: [1], U1: 1/2}
```

Listing 11: MFV model file.

Since now every fermion carries a fundamental $SU(3)_f$ index, one must remove the entry '`generations: 3`' of Listing 9 from all fermion declarations. The entire model file encoding MFV is shown in Listing 11. It can be used to construct the leading (i.e., flavor symmetric) terms in the MFV EFT basis.[26] Of course, other realizations of flavor symmetry can be implemented in a similar fashion. For example, Refs. [61–63] examine various flavor symmetries in an EFT context.

5. Constructing operators

5.1. Running `AutoEFT`

Given a valid model file, `AutoEFT` can be used to construct an EFT basis for a certain mass dimension. For example, to construct the SMEFT dimension-six operators, run the command:

```
autoeft construct sm.yml 6
```

where `sm.yml` denotes the model file of Listing 9. `AutoEFT` will first display a disclaimer followed by a summary of the loaded model. The summary includes the name and description of the model as well as a table containing all fields of the model, including the automatically generated conjugate fields. The table can be used to verify that the model file has been loaded correctly and the field representations are set up as desired. Afterwards, the operator construction starts and the number of families, types, terms, and operators is displayed in a live preview (see Appendix E and Ref. [44] for the meaning of these expressions). After the operator construction is finished, `AutoEFT` terminates and returns to the shell prompt. During each run, `AutoEFT` writes a *log file* called `autoeft.log` to the current working directory, capturing the console output.

During the construction, `AutoEFT` creates the output directory `efts/sm-eft/6/` in the current working directory. The substring '`sm`' is derived from the name of the model file `sm.yml`, and '`6`' is the requested mass dimension. All output files of `AutoEFT` will be written into this directory or its subdirectories. If during the construction an operator type which is already present in the output directory is encountered, `AutoEFT` will skip the construction of this particular type.[27]

The operator basis itself is written into the subdirectory `basis/`. This directory always contains the file `model.json`, serving as a reference to the model used during the construction, and the hidden file `.autoeft` containing metadata of the generation. All constructed operator files of a given `family` and `type` (cf. Appendix E) are included in further subdirectories of the form `<N>/<family>/<type>.yml`, where `N` denotes the total number of fields in the operator. The format of the operator files is explained in the next section.

A detailed description of all command-line options of the `construct` (short: `c`) command can be found in Appendix B.1.3. Here, we only mention the optional `--select` (short: `-s`) and `--ignore` (short: `-i`) options, which are particularly useful if only a specific subset of operators should be constructed. For example, to only construct dimension-six operators containing exactly two Higgs doublets, run the command:

```
autoeft c sm.yml 6 -s "{H: 2, H+: 0}" \
                   -s "{H: 0, H+: 2}" \
                   -s "{H: 1, H+: 1}"
```

On the other hand, the command

```
autoeft c sm.yml 6 -i "{GL: +}" -i "{GL+: +}"
```

will exclude all operators containing gluons. The `-s` and `-i` options can be combined, of course, whereupon the latter overrides the former in case of conflicts.

After a successful run, `AutoEFT` writes the file `stats.yml` to the output directory, containing the total number of families, types, terms, and operators in the basis. These numbers can also be obtained using the `count` command; see Appendix B.1.4.

5.2. Output format

The *operator files* contain all information needed to reconstruct the EFT basis type-by-type. Here, we demonstrate how their content can be interpreted using the SMEFT operator type $L_L^1 Q_L^3$ as an example. The entire operator file, named `1LL_3QL.yml`, is displayed in Listing 12.[28]

```
# '1LL_3QL.yml' generated by AutoEFT 1.0.0
version: 1.0.0
type:
- {LL: 1, QL: 3}
- complex
generations: {LL: 3, QL: 3}
n_terms: 3
n_operators: 57
invariants:
  Lorentz:
    O(Lorentz,1): +eps(1_1,3_1)*eps(2_1,4_1) * LL(1_1)*QL(2_1)*QL(3_1)*QL(4_1)
```

[26] In principle, it would be possible to include the Yukawa couplings as spurion fields—also transforming under the flavor symmetry. This would allow to construct the MFV EFT basis beyond the leading terms. However, the Yukawa couplings are dimensionless and should instead be expanded by some other small quantity. Such a declaration is not included in the model file specifications yet, but we intend to implement this feature in the next release of `AutoEFT`.

[27] Unless the `--overwrite` flag is set; see Appendix B.1.3.

[28] In the supplemental material accompanying Ref. [44], which adopts the all-left notation for the fields, the corresponding file is named `1L_3Q.yml`.

```
12     O(Lorentz,2): +eps(1_1,2_1)*eps(3_1,4_1) * LL(1_1)*QL(2_1)*QL
          ↪ (3_1)*QL(4_1)
13   SU3:
14     O(SU3,1): +eps(2_1,3_1,4_1) * LL*QL(2_1)*QL(3_1)*QL(4_1)
15   SU2:
16     O(SU2,1): +eps(1_1,3_1)*eps(2_1,4_1) * LL(1_1)*QL(2_1)*QL(3_1
          ↪ )*QL(4_1)
17     O(SU2,2): +eps(1_1,2_1)*eps(3_1,4_1) * LL(1_1)*QL(2_1)*QL(3_1
          ↪ )*QL(4_1)
18 permutation_symmetries:
19 - vector: Lorentz * SU3 * SU2
20 - symmetry: {LL: [1], QL: [1, 1, 1]}
21   n_terms: 1
22   n_operators: 3
23   matrix: |-
24     [ 0 -1  1  0]
25 - symmetry: {LL: [1], QL: [2, 1]}
26   n_terms: 1
27   n_operators: 24
28   matrix: |-
29     [-1  2  2 -1]
30 - symmetry: {LL: [1], QL: [3]}
31   n_terms: 1
32   n_operators: 30
33   matrix: |-
34     [ 2 -1 -1  2]
```

Listing 12: $L_{\rm L}^1 Q_{\rm L}^3$ operator file.

A summary of all keywords appearing in the operator files is included in Appendix D. For this particular example, they can be interpreted in the following way:

`version`:
The version of `AutoEFT` that was used to produce the output file.

`type`:
The first entry denotes the operator type, in this example $L_{\rm L}^1 Q_{\rm L}^3$. The second entry states that this type is '`complex`', meaning there is a distinct Hermitian conjugate type (which is contained in `1LL+_3QL+.yml`).

`generations`:
For reference, the number of generations for each field is also displayed in the operator files. In the present case, the file was generated for three generations of leptons and quarks.

`n_terms`:
The total number of operators with independent Lorentz and internal index contractions and definite permutation symmetry of the repeated fields (i.e. fields which differ at most in their generation index). It does not take into account the different generations though. In this example, the generation indices of the quarks can be decomposed into totally anti-symmetric $[1,1,1]$, mixed symmetric $[2,1]$, and totally symmetric $[3]$ tensors.

`n_operators`:
The total number of independent operators, taking into account the independent values the generation indices can assume. Here, there are $3\cdot(1+8+10)=57$ independent combinations of the $L_{\rm L}$ and $Q_{\rm L}$ generations.

`invariants`:
The invariant contractions are given by:

$$\begin{aligned}
\mathcal{O}_1^{\rm Lorentz} &= \epsilon^{\alpha\gamma}\,\epsilon^{\beta\delta}\,L_{{\rm L}\alpha}\,Q_{{\rm L}\beta}\,Q_{{\rm L}\gamma}\,Q_{{\rm L}\delta}\,,\\
\mathcal{O}_2^{\rm Lorentz} &= \epsilon^{\alpha\beta}\,\epsilon^{\gamma\delta}\,L_{{\rm L}\alpha}\,Q_{{\rm L}\beta}\,Q_{{\rm L}\gamma}\,Q_{{\rm L}\delta}\,,\\
\mathcal{O}_1^{SU(3)} &= \epsilon^{bcd}\,L_{\rm L}\,Q_{{\rm L}b}\,Q_{{\rm L}c}\,Q_{{\rm L}d}\,,\\
\mathcal{O}_1^{SU(2)} &= \epsilon^{ik}\,\epsilon^{jl}\,L_{{\rm L}i}\,Q_{{\rm L}j}\,Q_{{\rm L}k}\,Q_{{\rm L}l}\,,\\
\mathcal{O}_2^{SU(2)} &= \epsilon^{ij}\,\epsilon^{kl}\,L_{{\rm L}i}\,Q_{{\rm L}j}\,Q_{{\rm L}k}\,Q_{{\rm L}l}\,,
\end{aligned}\tag{10}$$

where only the relevant set of indices is displayed in each case.

`permutation_symmetries`:
The first entry always denotes the order of the tensor product. In this case, the combination is given by

$$\texttt{vector}:\quad \vec{\mathcal{O}} \equiv \mathcal{O}^{\rm Lorentz}\otimes\mathcal{O}^{SU(3)}\otimes\mathcal{O}^{SU(2)} = \begin{pmatrix} \mathcal{O}_1^{\rm Lorentz}\otimes\mathcal{O}_1^{SU(3)}\otimes\mathcal{O}_1^{SU(2)}\\ \mathcal{O}_1^{\rm Lorentz}\otimes\mathcal{O}_1^{SU(3)}\otimes\mathcal{O}_2^{SU(2)}\\ \mathcal{O}_2^{\rm Lorentz}\otimes\mathcal{O}_1^{SU(3)}\otimes\mathcal{O}_1^{SU(2)}\\ \mathcal{O}_2^{\rm Lorentz}\otimes\mathcal{O}_1^{SU(3)}\otimes\mathcal{O}_2^{SU(2)} \end{pmatrix}.\tag{11}$$

The first element of this vector is to be read as

$$\mathcal{O}_1^{\rm Lorentz}\otimes\mathcal{O}_1^{SU(3)}\otimes\mathcal{O}_1^{SU(2)} = = \epsilon^{\alpha\gamma}\,\epsilon^{\beta\delta}\,\epsilon^{ik}\,\epsilon^{jl}\,\epsilon^{bcd}\,L^w_{{\rm L}\,\alpha i}\,Q^x_{{\rm L}\beta bj}\,Q^y_{{\rm L}\gamma ck}\,Q^z_{{\rm L}\delta dl}\,,\tag{12}$$

for example, with generation indices $w,x,y,z\in\{1,2,3\}$, while the other indices are those of Eq. (10).

The choice of generation indices in the repeated fields is not arbitrary though. The remaining entries take this into account via the permutation symmetries of the repeated fields and the associated linearly independent combinations of the invariant contractions. In this case, there are three distinct permutation symmetries of the quark generation indices:

- $\lambda_{L_{\rm L}}\sim[1]$ & $\lambda_{Q_{\rm L}}\sim[1,1,1]$
$$\texttt{matrix}:\quad \mathcal{K}^{[1],[1,1,1]}\equiv\begin{pmatrix}0 & -1 & 1 & 0\end{pmatrix}.\tag{13}$$
- $\lambda_{L_{\rm L}}\sim[1]$ & $\lambda_{Q_{\rm L}}\sim[2,1]$
$$\texttt{matrix}:\quad \mathcal{K}^{[1],[2,1]}\equiv\begin{pmatrix}-1 & 2 & 2 & -1\end{pmatrix}.\tag{14}$$
- $\lambda_{L_{\rm L}}\sim[1]$ & $\lambda_{Q_{\rm L}}\sim[3]$
$$\texttt{matrix}:\quad \mathcal{K}^{[1],[3]}\equiv\begin{pmatrix}2 & -1 & -1 & 2\end{pmatrix}.\tag{15}$$

Combining the information of the model file, one arrives at three independent terms, each with definite permutation symmetry. The first one is

$$\begin{aligned}\mathcal{O}^{[1],[1,1,1]} &\equiv \mathcal{K}^{[1],[1,1,1]}\cdot\vec{\mathcal{O}}\\ &= -\,\mathcal{O}_1^{\rm Lorentz}\otimes\mathcal{O}_1^{SU(3)}\otimes\mathcal{O}_2^{SU(2)}\\ &\quad +\,\mathcal{O}_2^{\rm Lorentz}\otimes\mathcal{O}_1^{SU(3)}\otimes\mathcal{O}_1^{SU(2)}\,.\end{aligned}\tag{16}$$

While the generation index $w\in\{1,2,3\}$ for the lepton can be any of these three values, the quark generation indices can only assume a single combination of values in this case; one choice is $(x,y,z)=(1,2,3)$, for example. Thus, $\mathcal{O}^{[1],[1,1,1]}$ represents $3\cdot 1=3$ different operators (see line 22 of Listing 12), if the generation of the fields is taken into account.

The second term is

$$\begin{aligned}\mathcal{O}^{[1],[2,1]} &\equiv \mathcal{K}^{[1],[2,1]}\cdot\vec{\mathcal{O}}\\ &= -\,\mathcal{O}_1^{\rm Lorentz}\otimes\mathcal{O}_1^{SU(3)}\otimes\mathcal{O}_1^{SU(2)}\\ &\quad +2\,\mathcal{O}_1^{\rm Lorentz}\otimes\mathcal{O}_1^{SU(3)}\otimes\mathcal{O}_2^{SU(2)}\\ &\quad +2\,\mathcal{O}_2^{\rm Lorentz}\otimes\mathcal{O}_1^{SU(3)}\otimes\mathcal{O}_1^{SU(2)}\\ &\quad -\,\mathcal{O}_2^{\rm Lorentz}\otimes\mathcal{O}_1^{SU(3)}\otimes\mathcal{O}_2^{SU(2)}\,.\end{aligned}\tag{17}$$

For this permutation symmetry, there are eight independent combinations of the quark generation indices; one may choose them to be[29]

$$\begin{aligned}(x,y,z)\in\{&(1,1,2),(1,1,3),(1,2,2),(1,2,3),(1,3,2),\\ &(1,3,3),(2,2,3),(2,3,3)\}\,.\end{aligned}\tag{18}$$

Again taking into account the multiplicity of the lepton generations, this term represents $3\cdot 8=24$ operators (see line 27).

[29] They can be determined from the associated semi-standard Young tableaux $\begin{smallmatrix}x & y\\ z\end{smallmatrix}$; see Ref. [44] for details.

The third term is

$$\begin{aligned}\mathcal{O}^{[1],[3]} &\equiv \mathcal{K}^{[1],[3]} \cdot \vec{\mathcal{O}} \\ &= 2\,\mathcal{O}_1^{\text{Lorentz}} \otimes \mathcal{O}_1^{SU(3)} \otimes \mathcal{O}_1^{SU(2)} \\ &\quad - \mathcal{O}_1^{\text{Lorentz}} \otimes \mathcal{O}_1^{SU(3)} \otimes \mathcal{O}_2^{SU(2)} \\ &\quad - \mathcal{O}_2^{\text{Lorentz}} \otimes \mathcal{O}_1^{SU(3)} \otimes \mathcal{O}_1^{SU(2)} \\ &\quad + 2\,\mathcal{O}_2^{\text{Lorentz}} \otimes \mathcal{O}_1^{SU(3)} \otimes \mathcal{O}_2^{SU(2)}\,,\end{aligned} \tag{19}$$

where one can choose the following ten combinations of the quark generation indices:

$$\begin{aligned}(x,y,z) \in \{&(1,1,1),(1,1,2),(1,1,3),(1,2,2),(1,2,3),\\ &(1,3,3),(2,2,2),(2,2,3),(2,3,3),(3,3,3)\}\,,\end{aligned} \tag{20}$$

and therefore this term represents $3 \cdot 10 = 30$ operators (see line 32).

More examples and detailed descriptions of the output format can be found in Ref. [44]. See also Section 6.1 for an example on how `AutoEFT` can be used to perform this expansion automatically.

5.3. Counting SMEFT operators

The number of families, types, terms, and operators (cf. Appendix E) can be obtained from an existing basis using the `count` command;[30] see Appendix B.1.4. `AutoEFT` also writes these numbers to the file `stats.yml` after each basis construction, see Section 5.1. In order to arrive at a well-defined number, operators (families, types, terms) and their distinct conjugate version are counted separately. This means that, for SMEFT at mass dimension five, `AutoEFT` counts two operators (equaling the number of families, terms, and types) if only one generation of leptons is taken into account: the Weinberg operator $\sim L_L L_L H H$ and its conjugate $\sim H^\dagger H^\dagger L_L^\dagger L_L^\dagger$. For three generations, each of the two terms leads to six operators (permutation symmetry eliminates three out of the nine possible combinations, cf. Section 5.2).

For SMEFT at mass dimension six, various different numbers can be found in the literature, depending on the way of counting, or possible additional symmetries that have been imposed. For example, assuming baryon-number conservation, `AutoEFT` generates 76 operators for a single generation of fermions. This is in line with the 59 operators reported in Ref. [16] if one counts the 17 Hermitian conjugate operators of Q_{uG}, Q_{dG}, $Q_{\varphi ud}$, Q_{ledq}, $Q^{(1)}_{quqd}$, $Q^{(8)}_{quqd}$, $Q^{(1)}_{lequ}$, $Q^{(3)}_{lequ}$, and Q_{ij} with $i \in \{e,u,d\}$, $j \in \{\varphi, W, B\}$, as independent degrees of freedom. Relaxing the assumption of baryon-number conservation, `AutoEFT` reports a total of 84 operators, corresponding to the addition of the four baryon-number violating operators Q_{duq}, Q_{qqu}, Q_{qqq}, and Q_{duu}, quoted in Ref. [16], and their Hermitian conjugate versions. In the case of three fermion generations, `AutoEFT` generates 2499 operators if baryon-number conservation is imposed; otherwise the number increases to 3045. Enforcing flavor conservation as described in Section 4.4, `AutoEFT` only generates 47 operators at mass dimension six (cf. Ref. [62]).

The number of operators calculated with the Hilbert series [22,23] matches the counting of `AutoEFT` for any mass dimension exactly, as we have verified for SMEFT up to mass dimension 12 [44].

5.4. Hermitian conjugate operators

To write down a *real* EFT Lagrangian, it is required to include the Hermitian conjugate version of each term. Given a complex Wilson coefficient C and an operator $\mathcal{O}$, this means that the Lagrangian must contain the sum

$$C\mathcal{O} + C^*\mathcal{O}^\dagger \in \mathcal{L}\,. \tag{21}$$

If the operator $\mathcal{O}$ is (anti-)Hermitian, Eq. (21) simplifies to $(C \pm C^*)\mathcal{O}$ so that only the real (imaginary) part of the Wilson coefficient contributes to the Lagrangian, and the operator represents a single degree of freedom. If the operator is not (anti-)Hermitian, both real and imaginary parts of the Wilson coefficient appear in the Lagrangian, and the operator represents two degrees of freedom. Of course, one could choose a basis of only Hermitian operators (implying only real Wilson coefficients) which is, however, not very practical for most applications.

`AutoEFT` treats operators that are obtained by Hermitian conjugation as independent degrees of freedom—except for (anti-)Hermitian operators. By default, it thus includes the Hermitian conjugate operator types in the basis. Note that the Hermitian conjugate operators of a given type are not necessarily the same as the operators of the conjugate type, but rather linear combinations thereof. Future versions of `AutoEFT` will provide the functionality to obtain the Hermitian conjugate of individual operators, rather than operator types.

`AutoEFT` can be prevented from constructing the Hermitian conjugate types by the command-line option `--no_hc`. The user then needs to conjugate each operator that is constructed by `AutoEFT` and add it to the Lagrangian with the complex conjugate Wilson coefficient, see Eq. (21).

5.5. Limitations of AutoEFT

5.5.1. Computing resources

The algorithm implemented in `AutoEFT` works for an arbitrary mass dimension d. However, the computational efforts increase exponentially with d. For SMEFT, for example, while the basis at mass dimension 10 could be constructed within a few hours, dimension 12 took of the order of months. Currently, CPU time is therefore probably the most severe limiting factor for going to even higher mass dimension.

Other hardware limitations may arise from memory requirements, which mostly come from algebraic operations when projecting the general Lorentz and $SU(N)$ tensors onto the tensor basis. Also the storage of the output files may exceed the available hardware. In Ref. [44], we estimated the required disk space for SMEFT at mass dimension 26 to amount to about 1 PB.

5.5.2. Generators of the symmetric group

The elimination of redundancies due to the occurrence of repeated fields in an operator requires representation matrices of the symmetric group S_n. Since these are independent of the specific EFT under consideration, `AutoEFT` comes with a hard-coded version of generator matrices that are used to generate all required representation matrices up to $n = 9$, which is sufficient for problems with up to nine repeated fields in an operator.[31] If the construction of an EFT requires a representation matrix which is not hard-coded, `AutoEFT` will terminate and request the missing matrices. `AutoEFT` also provides a functionality to (re-)compute these matrices, using the `generators` command, see Appendix B.1. Calculating them beyond $n = 9$ is very CPU expensive though.

5.5.3. Conceptual limitations

As indicated already in Section 2, `AutoEFT` is currently limited to fields with spin 0, 1/2, 1, and 2, where the latter two must be massless. Furthermore, internal symmetries must only be due to $U(1)$ or $SU(N)$ groups.

The operator basis will be non-redundant on-shell, which means that operators which are related by equations-of-motion and integration-by-parts identities have been identified. For the purpose of renormalization, operators proportional to equations-of-motion as well as gauge-variant operators are required in general though, but currently these

[30] See Appendix E for the definition of a valid operator basis that can be processed by `AutoEFT`.

[31] For SMEFT, this would be sufficient to generate the basis up to mass dimension 18.

cannot be generated by AutoEFT. In addition, AutoEFT only constructs operators that mediate proper interactions, meaning that any operator must be composed of at least three fields. Furthermore, AutoEFT does not take into account evanescent operators, as they may be required for calculations in dimensional regularization (see, e.g., Ref. [64]).

6. Working with operator files

Besides providing a command-line script for the operator basis construction, AutoEFT also serves as a SageMath library to work with EFT operators. In this section, we illustrate some of the features available by importing AutoEFT into SageMath, which will be extended further in future releases of AutoEFT. We also introduce some additional command-line functionalities that are not directly related to the operator construction but may be useful for the user.

6.1. Loading operator files

Once an operator basis is constructed using the construct command, it is possible to load the operator files into SageMath for further manipulation. A *valid* operator basis that can be processed by AutoEFT is represented by a directory containing the file model.json referencing the model, a (hidden) file .autoeft containing metadata, and the respective operator files in the format described in Section 5.2.[32] By default, the directories called basis created by AutoEFT are structured such that they can be directly loaded into SageMath. See also Appendix E.

Consider for example Listing 13, displaying how one can load the mass dimension-six SMEFT basis located at efts/sm-eft/6/basis (cf. line 5). The command lines in this listing can be entered into an interactive SageMath session, for example, or stored in a <file> first and passed to sage via a shell command ‘sage <file>’.

AutoEFT provides the class

```
autoeft.io.basis.BasisFile
```

to interact with the basis stored on the disk. It must be initialized with the path pointing to the basis as displayed in line 6. Afterwards, the entire basis can be loaded into memory with the get_basis() function (cf. line 7), returning a dictionary that maps each operator type to the contents of the respective operator file. For example, to access the operators of type $L_{\mathrm{L}}^{1} Q_{\mathrm{L}}^{3}$ (see also Section 5.2), the type can be identified by the mapping ‘{"LL": 1, "QL": 3}’, following the same convention as the output files; see line 4 in Listing 12. The resulting object, assigned to the variable ‘LQQQ’ in line 9 of Listing 13, is an instance of the

```
autoeft.base.basis.OperatorInfoPermutation
```

class. This class contains all the information of the operator files as object properties, see for example lines 10–14. However, it also contains further properties and functions. For example, to obtain the actual terms of this type, the object can be expanded using the expanded() function, returning a new object that contains the terms as displayed in Eq. (16) (cf. lines 16–22 of Listing 13).

```
1 from pathlib import Path
2
3 from autoeft.io.basis import BasisFile
4
5 basis_path = Path("efts/sm-eft/6/basis")
6 basis_file = BasisFile(basis_path)
7 basis = basis_file.get_basis()
8
9 LQQQ = basis[{"LL": 1, "QL": 3}]
10 print(LQQQ)
11 # LL(1) QL(3)
12
13 print(LQQQ.n_terms, LQQQ.n_operators, sep=" & ")
14 # 3 & 57
15
16 for term in LQQQ.expanded():
17     print(term.symmetry)
18     print(term)
19     print(term.operators)
20 # {'LL': [1], 'QL': [1,1,1]}
21 # (-1) * eps(Lorentz_1_1,Lorentz_3_1)*eps(Lorentz_2_1,Lorentz_4_1
     ↪ )*eps(SU3_2_1,SU3_3_1,SU3_4_1)*eps(SU2_1_1,SU2_2_1)*eps(
     ↪ SU2_3_1,SU2_4_1) * LL(Lorentz_1_1;SU2_1_1)*QL(Lorentz_2_1
     ↪ ;SU3_2_1;SU2_2_1)*QL(Lorentz_3_1;SU3_3_1;SU2_3_1)*QL(
     ↪ Lorentz_4_1;SU3_4_1;SU2_4_1) + (1) * eps(Lorentz_1_1,
     ↪ Lorentz_2_1)*eps(Lorentz_3_1,Lorentz_4_1)*eps(SU3_2_1,
     ↪ SU3_3_1,SU3_4_1)*eps(SU2_1_1,SU2_3_1)*eps(SU2_2_1,SU2_4_1
     ↪ ) * LL(Lorentz_1_1;SU2_1_1)*QL(Lorentz_2_1;SU3_2_1;
     ↪ SU2_2_1)*QL(Lorentz_3_1;SU3_3_1;SU2_3_1)*QL(Lorentz_4_1;
     ↪ SU3_4_1;SU2_4_1)
22 # [(1, 1, 2, 3), (2, 1, 2, 3), (3, 1, 2, 3)]
23
24 # {'LL': [1], 'QL': [2,1]}
25 # (-1) * eps(Lorentz_1_1,Lorentz_3_1)*eps(Lorentz_2_1,Lorentz_4_1
     ↪ )*eps(SU3_2_1,SU3_3_1,SU3_4_1)*eps(SU2_1_1,SU2_3_1)*eps(
     ↪ SU2_2_1,SU2_4_1) * LL(Lorentz_1_1;SU2_1_1)*QL(Lorentz_2_1
     ↪ ;SU3_2_1;SU2_2_1)*QL(Lorentz_3_1;SU3_3_1;SU2_3_1)*QL(
     ↪ Lorentz_4_1;SU3_4_1;SU2_4_1) + (2) * eps(Lorentz_1_1,
     ↪ Lorentz_3_1)*eps(Lorentz_2_1,Lorentz_4_1)*eps(SU3_2_1,
     ↪ SU3_3_1,SU3_4_1)*eps(SU2_1_1,SU2_2_1)*eps(SU2_3_1,SU2_4_1
     ↪ ) * LL(Lorentz_1_1;SU2_1_1)*QL(Lorentz_2_1;SU3_2_1;
     ↪ SU2_2_1)*QL(Lorentz_3_1;SU3_3_1;SU2_3_1)*QL(Lorentz_4_1;
     ↪ SU3_4_1;SU2_4_1) + (2) * eps(Lorentz_1_1,Lorentz_2_1)*eps
     ↪ (Lorentz_3_1,Lorentz_4_1)*eps(SU3_2_1,SU3_3_1,SU3_4_1)*
     ↪ eps(SU2_1_1,SU2_3_1)*eps(SU2_2_1,SU2_4_1) * LL(
     ↪ Lorentz_1_1;SU2_1_1)*QL(Lorentz_2_1;SU3_2_1;SU2_2_1)*QL(
     ↪ Lorentz_3_1;SU3_3_1;SU2_3_1)*QL(Lorentz_4_1;SU3_4_1;
     ↪ SU2_4_1) + (-1) * eps(Lorentz_1_1,Lorentz_2_1)*eps(
     ↪ Lorentz_3_1,Lorentz_4_1)*eps(SU3_2_1,SU3_3_1,SU3_4_1)*eps
     ↪ (SU2_1_1,SU2_2_1)*eps(SU2_3_1,SU2_4_1) * LL(Lorentz_1_1;
     ↪ SU2_1_1)*QL(Lorentz_2_1;SU3_2_1;SU2_2_1)*QL(Lorentz_3_1;
     ↪ SU3_3_1;SU2_3_1)*QL(Lorentz_4_1;SU3_4_1;SU2_4_1)
26 # [(1, 1, 1, 2), (1, 1, 1, 3), (1, 1, 2, 2), (1, 1, 2, 3), (1, 1,
     ↪  3, 2), (1, 1, 3, 3), (1, 2, 2, 3), (1, 2, 3, 3), (2, 1,
     ↪ 1, 2), (2, 1, 1, 3), (2, 1, 2, 2), (2, 1, 2, 3), (2, 1,
     ↪ 3, 2), (2, 1, 3, 3), (2, 2, 2, 3), (2, 2, 3, 3), (3, 1,
     ↪ 1, 2), (3, 1, 1, 3), (3, 1, 2, 2), (3, 1, 2, 3), (3, 1,
     ↪ 3, 2), (3, 1, 3, 3), (3, 2, 2, 3), (3, 2, 3, 3)]
27
28 # {'LL': [1], 'QL': [3]}
29 # (2) * eps(Lorentz_1_1,Lorentz_3_1)*eps(Lorentz_2_1,Lorentz_4_1)
     ↪ *eps(SU3_2_1,SU3_3_1,SU3_4_1)*eps(SU2_1_1,SU2_3_1)*eps(
     ↪ SU2_2_1,SU2_4_1) * LL(Lorentz_1_1;SU2_1_1)*QL(Lorentz_2_1
     ↪ ;SU3_2_1;SU2_2_1)*QL(Lorentz_3_1;SU3_3_1;SU2_3_1)*QL(
     ↪ Lorentz_4_1;SU3_4_1;SU2_4_1) + (-1) * eps(Lorentz_1_1,
     ↪ Lorentz_3_1)*eps(Lorentz_2_1,Lorentz_4_1)*eps(SU3_2_1,
     ↪ SU3_3_1,SU3_4_1)*eps(SU2_1_1,SU2_2_1)*eps(SU2_3_1,SU2_4_1
     ↪ ) * LL(Lorentz_1_1;SU2_1_1)*QL(Lorentz_2_1;SU3_2_1;
     ↪ SU2_2_1)*QL(Lorentz_3_1;SU3_3_1;SU2_3_1)*QL(Lorentz_4_1;
     ↪ SU3_4_1;SU2_4_1) + (-1) * eps(Lorentz_1_1,Lorentz_2_1)*
     ↪ eps(Lorentz_3_1,Lorentz_4_1)*eps(SU3_2_1,SU3_3_1,SU3_4_1)
     ↪ *eps(SU2_1_1,SU2_3_1)*eps(SU2_2_1,SU2_4_1) * LL(
     ↪ Lorentz_1_1;SU2_1_1)*QL(Lorentz_2_1;SU3_2_1;SU2_2_1)*QL(
     ↪ Lorentz_3_1;SU3_3_1;SU2_3_1)*QL(Lorentz_4_1;SU3_4_1;
     ↪ SU2_4_1) + (2) * eps(Lorentz_1_1,Lorentz_2_1)*eps(
     ↪ Lorentz_3_1,Lorentz_4_1)*eps(SU3_2_1,SU3_3_1,SU3_4_1)*eps
     ↪ (SU2_1_1,SU2_2_1)*eps(SU2_3_1,SU2_4_1) * LL(Lorentz_1_1;
     ↪ SU2_1_1)*QL(Lorentz_2_1;SU3_2_1;SU2_2_1)*QL(Lorentz_3_1;
     ↪ SU3_3_1;SU2_3_1)*QL(Lorentz_4_1;SU3_4_1;SU2_4_1)
30 # [(1, 1, 1, 1), (1, 1, 1, 2), (1, 1, 1, 3), (1, 1, 2, 2), (1, 1,
     ↪  2, 3), (1, 1, 3, 3), (1, 2, 2, 2), (1, 2, 2, 3), (1, 2,
     ↪ 3, 3), (1, 3, 3, 3), (2, 1, 1, 1), (2, 1, 1, 2), (2, 1,
     ↪ 1, 3), (2, 1, 2, 2), (2, 1, 2, 3), (2, 1, 3, 3), (2, 2,
     ↪ 2, 2), (2, 2, 2, 3), (2, 2, 3, 3), (2, 3, 3, 3), (3, 1,
     ↪ 1, 1), (3, 1, 1, 2), (3, 1, 1, 3), (3, 1, 2, 2), (3, 1,
     ↪ 2, 3), (3, 1, 3, 3), (3, 2, 2, 2), (3, 2, 2, 3), (3, 2,
     ↪ 3, 3), (3, 3, 3, 3)]
```

Listing 13: Example script loading an operator basis created by AutoEFT. The operator type $L_{\mathrm{L}}^{1} Q_{\mathrm{L}}^{3}$ is selected and explicitly expanded into its terms as illustrated in Section 5.2.

[32] Note that, while the files model.json and .autoeft must be contained in the top-level directory of the basis, the operator files can be structured in subdirectories. The get_basis function searches all subdirectories for files with extension ‘.yml’ and loads them as operator file.

6.2. LaTeX output

AutoEFT provides the `latex` (short: `l`) command for the automatic LaTeX markup of the constructed operators. For example, to produce the TeX files corresponding to the operator basis located at `efts/sm-eft/6/basis`,[33] run the command:

```
autoeft latex efts/sm-eft/6/basis
```

By default, AutoEFT stores all `.tex` files under the directory `tex/sm-eft/6/`. If this directory does not contain a file called `main.tex`, an appropriate file will be generated automatically. From this LaTeX file (for example with the help of `pdflatex`), one can produce a PDF document which contains a table encoding the model; a table representing the numbers of types, terms, and operators per family; the respective Hilbert series; and the information encoded in the operator files for each type.[34] The `latex` command also supports the `--select` and `--ignore` options, to restrict the generation of TeX files to a subset of the entire basis (cf. Section 5 and Appendix B.1.5).

7. Conclusion

We have presented AutoEFT, a completely independent implementation of the algorithms and concepts proposed in Refs. [13,41,42] for the automated bottom-up construction of on-shell operator bases for generic EFTs. AutoEFT has been successfully used to construct—for the first time—the complete and non-redundant SMEFT and GRSMEFT operator bases for mass dimensions 10, 11, and 12 [44]. Besides the SM, AutoEFT can accommodate various low-energy scenarios and generate the respective EFT operator bases, in principle up to arbitrary mass dimension. Due to the simple format of the input files and the command-line utilities provided by AutoEFT, the user can compose custom models in a straight-forward way and construct EFT operator bases with minimal effort. Its phenomenological purpose is to eliminate the task of manually constructing EFTs from low-energy theories which may involve as-of-yet undiscovered light particles. But AutoEFT may also help to understand deeper structures of EFTs at the general level (see, e.g., Ref. [65]).

AutoEFT provides a foundation for future EFT frameworks and we plan to extend its capabilities in various respects, including the automated translation of the output to the FeynRules [66,67] format, or the capability to transform Wilson coefficients between different bases. This latter feature will also allow for a detailed comparison to programs of similar purpose, which is currently impeded by different choices of the bases.

CRediT authorship contribution statement

Robert V. Harlander: Conceptualization, Software, Writing – original draft, Writing – review & editing. **Magnus C. Schaaf:** Conceptualization, Software, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The source code is attached to this submission.

[33] See Appendix E for the definition of a valid operator basis that can be processed by AutoEFT.

[34] Using the option `-c`, one can directly compile `main.tex` with the AutoEFT call, see Appendix B.1.5.

Acknowledgements

We would like to thank Jakob Linder and Maximilian Rzehak for helpful comments and extensive β-tests, and Tim Kempkens for the inspiring collaboration at the earlier stages of the project. This research was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant 400140256 – *GRK 2497: The physics of the heaviest particles at the LHC*, and grant 396021762 – *TRR 257: P3H – Particle Physics Phenomenology after the Higgs Discovery*.

Appendix A. Relations to conventional notation

In this section, we describe how the output of AutoEFT can be related to a more conventional notation, containing bispinors, field-strength tensors, and Weyl tensors with Lorentz four-vector indices as well as fields carrying anti-fundamental and adjoint indices of the internal symmetry groups. Applying the following translation rules is straightforward but may result in large expressions. Their systematic simplification requires a well-defined criterion for "simplicity", for example, and achieving maximal simplicity may require computationally expensive permutation operations which can exceed the capability of the available hardware. We therefore defer its implementation to future versions of AutoEFT.

The four-component bispinors can be decomposed into left- and right-handed part, such that they are equally represented by a single two component Weyl spinor:

$$\begin{aligned}\Psi_{\rm L} &= \begin{pmatrix}\psi_{{\rm L}\alpha}\\ 0\end{pmatrix}, \quad \bar\Psi_{\rm L} = \left(0, \psi_{{\rm L}\dot\alpha}^{\dagger}\right),\\ \Psi_{\rm R} &= \begin{pmatrix}0\\ {\psi_{\rm R}}^{\dot\alpha}\end{pmatrix}, \quad \bar\Psi_{\rm R} = \left({\psi_{\rm R}^{\dagger}}^{\alpha}, 0\right).\end{aligned} \tag{A.1}$$

Note that in the all-left chirality notation, $\psi \equiv \psi_{\rm L}$ and $\psi_{\rm C} \equiv \psi_{\rm R}^{\dagger}$.

The output of AutoEFT containing Weyl spinors can be expressed in terms of bilinears involving the bispinors

$$\Psi = \begin{pmatrix}\psi_{{\rm L}\alpha}\\ {\psi_{\rm R}}^{\dot\alpha}\end{pmatrix}, \qquad \chi = \begin{pmatrix}\chi_{{\rm L}\alpha}\\ {\chi_{\rm R}}^{\dot\alpha}\end{pmatrix}, \tag{A.2}$$

using the relations

$$\begin{aligned}
\bar\Psi_{\rm R}\chi_{\rm L} &= {\psi_{\rm R}^{\dagger}}^{\alpha}\chi_{{\rm L}\alpha}, & \bar\Psi_{\rm L}\chi_{\rm R} &= \psi_{{\rm L}\dot\alpha}^{\dagger}{\chi_{\rm R}}^{\dot\alpha},\\
\bar\Psi_{\rm R}\gamma^\mu\chi_{\rm R} &= {\psi_{\rm R}^{\dagger}}^{\alpha}(\sigma^\mu)_{\alpha\dot\alpha}{\chi_{\rm R}}^{\dot\alpha}, & \bar\Psi_{\rm L}\gamma^\mu\chi_{\rm L} &= \psi_{{\rm L}\dot\alpha}^{\dagger}(\bar\sigma^\mu)^{\dot\alpha\alpha}\chi_{{\rm L}\alpha},\\
\bar\Psi_{\rm R}\sigma^{\mu\nu}\chi_{\rm L} &= {\psi_{\rm R}^{\dagger}}^{\alpha}(\sigma^{\mu\nu})_\alpha{}^\beta\chi_{{\rm L}\beta}, & \bar\Psi_{\rm L}\sigma^{\mu\nu}\chi_{\rm R} &= \psi_{{\rm L}\dot\alpha}^{\dagger}(\bar\sigma^{\mu\nu})^{\dot\alpha}{}_{\dot\beta}{\chi_{\rm R}}^{\dot\beta},\\
\Psi_{\rm R}^{\rm T}{\rm C}\chi_{\rm R} &= \psi_{{\rm R}\dot\alpha}{\chi_{\rm R}}^{\dot\alpha}, & \Psi_{\rm L}^{\rm T}{\rm C}\chi_{\rm L} &= {\psi_{\rm L}}^{\alpha}\psi_{{\rm L}\alpha},\\
\Psi_{\rm R}^{\rm T}{\rm C}\gamma^\mu\chi_{\rm L} &= \psi_{{\rm R}\dot\alpha}(\bar\sigma^\mu)^{\dot\alpha\alpha}\chi_{{\rm L}\alpha}, & \Psi_{\rm L}^{\rm T}{\rm C}\gamma^\mu\chi_{\rm R} &= {\psi_{\rm L}}^{\alpha}(\sigma^\mu)_{\alpha\dot\alpha}{\chi_{\rm R}}^{\dot\alpha},\\
\Psi_{\rm R}^{\rm T}{\rm C}\sigma^{\mu\nu}\chi_{\rm R} &= \psi_{{\rm R}\dot\alpha}(\bar\sigma^{\mu\nu})^{\dot\alpha}{}_{\dot\beta}{\chi_{\rm R}}^{\dot\beta}, & \Psi_{\rm L}^{\rm T}{\rm C}\sigma^{\mu\nu}\chi_{\rm L} &= {\psi_{\rm L}}^{\alpha}(\sigma^{\mu\nu})_\alpha{}^\beta\psi_{{\rm L}\beta},\\
\bar\Psi_{\rm R}{\rm C}\bar\chi_{\rm R}^{\rm T} &= {\psi_{\rm R}^{\dagger}}^{\alpha}\chi_{{\rm R}\alpha}^{\dagger}, & \bar\Psi_{\rm L}{\rm C}\bar\chi_{\rm L}^{\rm T} &= \psi_{{\rm L}\dot\alpha}^{\dagger}{\chi_{\rm L}^{\dagger}}^{\dot\alpha},\\
\bar\Psi_{\rm R}\gamma^\mu{\rm C}\bar\chi_{\rm L}^{\rm T} &= {\psi_{\rm R}^{\dagger}}^{\alpha}(\sigma^\mu)_{\alpha\dot\alpha}{\chi_{\rm L}^{\dagger}}^{\dot\alpha}, & \bar\Psi_{\rm L}\gamma^\mu{\rm C}\bar\chi_{\rm R}^{\rm T} &= \psi_{{\rm L}\dot\alpha}^{\dagger}(\bar\sigma^\mu)^{\dot\alpha\alpha}\chi_{{\rm R}\alpha}^{\dagger},\\
\bar\Psi_{\rm R}\sigma^{\mu\nu}{\rm C}\bar\chi_{\rm R}^{\rm T} &= {\psi_{\rm R}^{\dagger}}^{\alpha}(\sigma^{\mu\nu})_\alpha{}^\beta\chi_{{\rm R}\beta}^{\dagger}, & \bar\Psi_{\rm L}\sigma^{\mu\nu}{\rm C}\bar\chi_{\rm L}^{\rm T} &= \psi_{{\rm L}\dot\alpha}^{\dagger}(\bar\sigma^{\mu\nu})^{\dot\alpha}{}_{\dot\beta}{\chi_{\rm L}^{\dagger}}^{\dot\beta},
\end{aligned} \tag{A.3}$$

where

$$\begin{aligned}(\sigma^\mu)_{\alpha\dot\alpha} &= (I, \vec\sigma)_{\alpha\dot\alpha}, & (\bar\sigma^\mu)^{\dot\alpha\alpha} &= (I, -\vec\sigma)^{\dot\alpha\alpha},\\ (\sigma^{\mu\nu})_\alpha{}^\beta &= \frac{i}{2}(\sigma^\mu\bar\sigma^\nu - \sigma^\nu\bar\sigma^\mu)_\alpha{}^\beta, & (\bar\sigma_{\mu\nu})^{\dot\alpha}{}_{\dot\beta} &= \frac{i}{2}\left(\bar\sigma_\mu\sigma_\nu - \bar\sigma_\nu\sigma_\mu\right)^{\dot\alpha}{}_{\dot\beta},\end{aligned} \tag{A.4}$$

and

$$\gamma^\mu = \begin{pmatrix}0 & (\sigma^\mu)_{\alpha\dot\beta}\\ (\bar\sigma^\mu)^{\dot\alpha\beta} & 0\end{pmatrix}, \qquad \sigma^{\mu\nu} = \frac{i}{2}\left[\gamma^\mu, \gamma^\nu\right], \qquad {\rm C} = i\gamma^0\gamma^2. \tag{A.5}$$

Here, I is the 2×2 identity matrix and $\vec\sigma = \left(\sigma^1, \sigma^2, \sigma^3\right)$ denotes the Pauli matrices.

Using the normalization of Li et al. [49], the covariant derivative, field-strength tensor, and Weyl tensor with Lorentz four-vector indices are given by

$$D_\mu = -\frac{1}{2} D^{\dot\alpha}_{\alpha}(\sigma_\mu)^{\alpha}_{\dot\alpha}\,, \qquad D^{\dot\alpha}_{\alpha} = D_\mu (\sigma^\mu)^{\dot\alpha}_{\alpha}\,, \tag{A.6}$$

$$\begin{aligned} F^{\mu\nu} &= \frac{i}{4} F_{\mathrm{L}}^{\alpha\beta}\sigma^{\mu\nu}_{\alpha\beta} - \frac{i}{4} F_{\mathrm{R}}^{\dot\alpha\dot\beta}\bar\sigma^{\mu\nu}_{\dot\alpha\dot\beta}\,, \\ F_{\mathrm{L}\,\alpha\beta} &= \frac{i}{2} F_{\mu\nu}\sigma^{\mu\nu}_{\alpha\beta}\,, \qquad F_{\mathrm{R}}^{\dot\alpha\dot\beta} = -\frac{i}{2} F^{\mu\nu}\bar\sigma^{\dot\alpha\dot\beta}_{\mu\nu}\,, \end{aligned} \tag{A.7}$$

and

$$\begin{aligned} C^{\mu\nu\rho\sigma} &= -\frac{1}{16} C_{\mathrm{L}}^{\alpha\beta\gamma\delta}\sigma^{\mu\nu}_{\alpha\beta}\sigma^{\rho\sigma}_{\gamma\delta} - \frac{1}{16} C_{\mathrm{R}}^{\dot\alpha\dot\beta\dot\gamma\dot\delta}\bar\sigma^{\mu\nu}_{\dot\alpha\dot\beta}\bar\sigma^{\rho\sigma}_{\dot\gamma\dot\delta}\,, \\ C_{\mathrm{L}\,\alpha\beta\gamma\delta} &= -\frac{1}{4} C_{\mu\nu\rho\sigma}\sigma^{\mu\nu}_{\alpha\beta}\sigma^{\rho\sigma}_{\gamma\delta}\,, \qquad C_{\mathrm{R}}^{\dot\alpha\dot\beta\dot\gamma\dot\delta} = -\frac{1}{4} C^{\mu\nu\rho\sigma}\bar\sigma^{\dot\alpha\dot\beta}_{\mu\nu}\bar\sigma^{\dot\gamma\dot\delta}_{\rho\sigma}\,, \end{aligned} \tag{A.8}$$

respectively.

Similarly, fields with $SU(N)$ anti-fundamental and adjoint indices are given by

$$(\psi^\dagger)^b = \frac{1}{(N-1)!}\epsilon^{a_1\ldots a_{N-1}b}\psi^\dagger_{a_1\ldots a_{N-1}}\,, \qquad \psi^\dagger_{a_1\ldots a_{N-1}} = \epsilon_{a_1\ldots a_{N-1}b}(\psi^\dagger)^b\,, \tag{A.9}$$

and

$$\begin{aligned} F^A &= \frac{1}{T_F (N-1)!}\epsilon^{a_1 a_3\ldots a_{N-1}b}(T^A)^{a_2}{}_b F_{a_1 a_2 a_3\ldots a_N}\,, \\ F_{a_1 a_2 a_3\ldots a_N} &= \epsilon_{a_1 a_3\ldots a_{N-1}b}(T^A)^b{}_{a_2} F^A\,, \end{aligned} \tag{A.10}$$

respectively. Here, T^A denotes the $SU(N)$ generators in the fundamental representation, and T_F defines their normalization via $Tr(T^A T^B) = T_F\,\delta^{AB}$.

Appendix B. Invoking `autoeft`

Synopsis

autoeft [*options*] *command* [*args*]

Description This is the main command starting `autoeft`. Its general behavior is affected by the *options*. `autoeft` offers several commands which are explained in the following sections. The behavior of each *command* is controlled by *args*, including positional arguments and further options.

Command-line options

`-h`
`--help`
Print a usage message briefly summarizing the command-line options.

`-v`
`--version`
Print the version number of `autoeft` to the standard output stream.

`-q`
`--quiet`
Suppress all output to the standard output stream.

B.1. `autoeft` commands

`autoeft check`
Validate the installation of `autoeft` by comparing to a pre-determined basis.

`autoeft sample-model`
Print the Standard Model definition in YAML format to the standard output stream.

`autoeft construct`
`autoeft c`
Construct an operator basis for a given model and mass dimension.

`autoeft count`
Count the number of families, types, terms, and operators for a given basis.

`autoeft latex`
`autoeft l`
Generate and compile TEX files for a given basis.

`autoeft generators`
`autoeft g`
View or create the symmetric group representation generators.

B.1.1. `check` command

Synopsis

```
autoeft check
```

Description This command constructs the mass dimension-six operator basis for the Standard Model Effective Field Theory and compares it to a pre-determined basis. If both bases agree, the installation is reported as successful.

B.1.2. `sample-model` command

Synopsis

```
autoeft sample-model
```

Description This command prints the content of the model file (see [Model], page 17) for the Standard Model to the standard output stream. To obtain the model file `sm.yml` in the current working directory, simply run:

```
autoeft sample-model > sm.yml
```

B.1.3. `construct` command

Synopsis

autoeft construct [*options*] *model* *dimension*

Description This command starts the construction of an operator basis. The details of the basis depend on the provided model file `model.yml` (see [Model], page 17) and mass dimension `D`. The result will be saved in the (default) output directory under <model>-eft/<D>/basis/. The operators are further collected by their `family` (see [*family*], page 19) and `type` (see [*type*], page 19) in subdirectories of structure <N>/<family>/<type>.yml, where `N` is the number of fields in the operator. See [Output], page 18 for the format of the output.

Positional arguments

model
Path to the model file that should be used for the construction (see [Model], page 17).

dimension
The integer mass dimension the constructed operators should have.

Options

`-h`
`-help`
Print a usage message briefly summarizing the command-line options.

`-v`
`--verbose`
Print a tree-like structure of operator families and types during the construction.

`-t` *n*
`--threads=`*n*
Set the number of threads *n* that start `form` processes. By default, *n* is the number of CPUs in the system.

`-n` *name*
`--name=`*name*
Set the EFT *name*. By default, the name is `<model>-eft`.
`-o` *path*
`--output=`*path*
Set the output *path*, where the operators will be saved. By default, the operators are saved under `efts/` in the current working directory.
`-s` *pattern*
`--select=`*pattern*
Only construct operator types (see [*type*], page 19) that match with *pattern*. The pattern must be given as a string representing a mapping—denoted by curly braces—from the fields to their number of occurrences in the desired type (e.g., '`{H: 3, H+: 3}`'). Besides explicit numbers, the symbol '+' can be used to require at least one occurrence. Alternatively, a range can be provided by '`x..y`', meaning there must be at least '`x`' and at most '`y`' occurrences. If one of the bounds is omitted (e.g., '`x..`', or '`..y`'), only the remaining one is enforced. By default, no selection is performed and any operator will be constructed. If this option is used multiple times, an operator type must match at least one pattern to be constructed. If this option is combined with the `-i`(`--ignore`) option, *select* is applied before *ignore*.
`-i` *pattern*
`--ignore=`*pattern*
Do not construct operator types (see [*type*], page 19) that match with *pattern*. The pattern must be given as a string representing a mapping—denoted by curly braces—from the fields to their number of occurrences in the desired type (e.g., '`{H: 3, H+: 3}`'). Besides explicit numbers, the symbol '+' can be used to require at least one occurrence. Alternatively, a range can be provided by '`x..y`', meaning there must be at least '`x`' and at most '`y`' occurrences. If one of the bounds is omitted (e.g., '`x..`', or '`..y`'), only the remaining one is enforced. By default, no exclusion is performed and all operators will be constructed. If this option is used multiple times, an operator type that matches at least one pattern will not be constructed. If this option is combined with the `-s`(`--select`) option, *select* is applied before *ignore*.
`--dry-run`
Only list the operator types (see [*type*], page 19) that match the provided selection. No explicit construction is performed.
`--generators=`*path*
Set the generators *path*, where `autoeft` searches for the symmetric group representation generators (see [Generators], page 17). By default, this is set to `gens/` in the current working directory. If the directory does not exist or some generator files are missing, `autoeft` loads fallback generators for the representations up to S_9.
`--overwrite`
Overwrite existing operator files in the output directory.
`--no_hc`
Prevent the explicit construction of conjugate operator types.

B.1.4. `count` *command*
Synopsis

```
autoeft count [options] basis
```

Description This command counts the number of families, types, terms, and operators for a given basis (see [Vocabulary], page 19).

Positional arguments

basis
Path to the basis containing the operators (see [*basis*], page 19).

Options

`-h`
`--help`
Print a usage message briefly summarizing the command-line options.
`-v`
`--verbose`
Print a tree-like structure of operator families and types.
`-o` *file*
`--output=`*file*
Set the output *file*, where the numbers will be saved. By default, the numbers are saved in the file `counts.yml` in the current working directory.
`-s` *pattern*
`--select=`*pattern*
Only count operator types (see [*type*], page 19) that match with *pattern*. The pattern must be given as a string representing a mapping—denoted by curly braces—from the fields to their number of occurrences in the desired type (e.g., '`{H: 3, H+: 3}`'). Besides explicit numbers, the symbol '+' can be used to require at least one occurrence. Alternatively, a range can be provided by '`x..y`', meaning there must be at least '`x`' and at most '`y`' occurrences. If one of the bounds is omitted (e.g., '`x..`', or '`..y`'), only the remaining one is enforced. By default, no selection is performed and any operator will be counted. If this option is used multiple times, an operator type must match at least one pattern to be counted. If this option is combined with the `-i`(`--ignore`) option, *select* is applied before *ignore*.
`-i` *pattern*
`--ignore=`*pattern*
Do not count operator types (see [*type*], page 19) that match with *pattern*. The pattern must be given as a string representing a mapping—denoted by curly braces—from the fields to their number of occurrences in the desired type (e.g., '`{H: 3, H+: 3}`'). Besides explicit numbers, the symbol '+' can be used to require at least one occurrence. Alternatively, a range can be provided by '`x..y`', meaning there must be at least '`x`' and at most '`y`' occurrences. If one of the bounds is omitted (e.g., '`x..`', or '`..y`'), only the remaining one is enforced. By default, no exclusion is performed and all operators will be counted. If this option is used multiple times, an operator type that matches at least one pattern will not be counted. If this option is combined with the `-s`(`--select`) option, *select* is applied before *ignore*.
`--dry-run`
Only list the operator types (see [*type*], page 19) that match the provided selection. No explicit counting is performed.
`--no_hc`
Prevent the implicit counting of conjugate operator types.

B.1.5. `latex` *command*
Synopsis

```
autoeft latex [options] basis
```

Description This command generates TEX files for a given basis (see [*basis*], page 19). The TEX files represent all the information encoded in the operator files as LATEX markup. The resulting files compose a valid LATEX document that can be compiled to a single PDF file.

Positional arguments

basis
Path to the basis containing the operators (see [*basis*], page 19).

Options

`-h`
`--help`
Print a usage message briefly summarizing the command-line options.

`-c` *command*
`--compile=`*command*
Compile the TEX files by invoking the *command* in the output directory.

`-o` *path*
`--output=`*path*
Set the output *path*, where the TEX files will be saved. By default, the TEX files are saved under `tex/` in the current working directory.

`-s` *pattern*
`--select=`*pattern*
Only generate TEX files for operator types (see [*type*], page 19) that match with *pattern*. The pattern must be given as a string representing a mapping—denoted by curly braces—from the fields to their number of occurrences in the desired type (e.g., '`{H: 3, H+: 3}`'). Besides explicit numbers, the symbol '+' can be used to require at least one occurrence. Alternatively, a range can be provided by '`x..y`', meaning there must be at least '`x`' and at most '`y`' occurrences. If one of the bounds is omitted (e.g., '`x..`', or '`..y`'), only the remaining one is enforced. By default, no selection is performed and any operator will be included. If this option is used multiple times, an operator type must match at least one pattern to be included. If this option is combined with the `-i`(`--ignore`) option, *select* is applied before *ignore*.

`-i` *pattern*
`--ignore=`*pattern*
Do not generate TEX files for operator types (see [*type*], page 19) that match with *pattern*. The pattern must be given as a string representing a mapping—denoted by curly braces—from the fields to their number of occurrences in the desired type (e.g., '`{H: 3, H+: 3}`'). Besides explicit numbers, the symbol '+' can be used to require at least one occurrence. Alternatively, a range can be provided by '`x..y`', meaning there must be at least '`x`' and at most '`y`' occurrences. If one of the bounds is omitted (e.g., '`x..`', or '`..y`'), only the remaining one is enforced. By default, no exclusion is performed and all operators will be included. If this option is used multiple times, an operator type that matches at least one pattern will be excluded. If this option is combined with the `-s`(`--select`) option, *select* is applied before *ignore*.

`--dry-run`
Only list the operator types (see [*type*], page 19) that match the provided selection. No explicit TEX files are generated.

B.1.6. `generators` command
Synopsis

```
autoeft generators [options]
```

Description This command handles the pre-computed generator matrices for the symmetric group representations. If this command is executed *without* the `-S` or `-P` options, a table of all generators that `autoeft` would load with the current options is printed to the standard output stream. Otherwise, the respective generators are computed and stored.

Options

`-h`
`--help`
Print a usage message briefly summarizing the command-line options.

`-o` *path*
`--output=`*path*
Set the output *path*, where the generators will be saved. By default, the generators are saved under `gens/` in the current working directory.

`-S` *N*
Create the generators for all irreducible representations of the symmetric group S_N of degree N.

`-P` *p* [*p* ...]
Create the generators for the irreducible representation given by the partition as a non-increasing list of integers *p*.

`--overwrite`
Overwrite existing generator files in the output directory.

B.2. Environment variables

`AUTOEFT_PATH`
The environment variable `AUTOEFT_PATH` can be set to a path (or a list of paths, separated by '`:`'), to specify where `autoeft` searches for the `form` executable. If `AUTOEFT_PATH` is not set, the system `PATH` will be used instead.

`AUTOEFT_CS`
The symbol appended to the field name to denote conjugate fields. By default, the symbol '`+`' is used.

`AUTOEFT_DS`
The symbol appended to spinor indices to denote dotted indices. By default, the symbol '`~`' is used.

Appendix C. Model file

`name` *Scalar*
The name of the model. This is usually a short identifier like '`SMEFT`' or '`LEFT`'.

`description` *Scalar*
Optional (longer) description of the model.

`symmetries` *Mapping*
Definition of the model's symmetries. The symmetries are divided into the sub-entries `lorentz_group`, `sun_groups`, and `u1_groups`.

`lorentz_group` *Mapping*
Properties of the Lorentz group—realized as $SU(2)_l \times SU(2)_r$. If omitted, `autoeft` loads the Lorentz group with default values.

`name` *Scalar*
The name associated with the Lorentz group.

- Group names must start with a letter ('`A-z`').
- Group names can only contain alpha-numeric characters and parentheses ('`A-z`', '`0-9`', '`(`', and '`)`').
- Group names must end with an alpha-numeric character or a parenthesis ('`A-z`', '`0-9`', '`(`', or '`)`').

By default, the name '`Lorentz`' is used.

`tex` *Scalar*
The TEX string associated with the Lorentz group. By default, the group name surrounded by `\mathtt{•}` is used.

`indices` *Sequence*
The list of TEX (spinor) indices associated with the Lorentz group. By default, the Greek letters '$\alpha, \beta, \ldots, \lambda$' are used.

`sun_groups` *Mapping*
Definition of the model's non-abelian symmetries—realized as

$SU(N)$ groups. Each entry is a mapping from the symmetry's name (e.g., 'QCD' or 'SU(3)') to its properties.

- Group names must start with a letter ('A-z').
- Group names can only contain alpha-numeric characters and parentheses ('A-z', '0-9', '(', and ')').
- Group names must end with an alpha-numeric character or a parenthesis ('A-z', '0-9', '(', or ')').

By default, `sun_groups` is the empty mapping '{}'.

`N` *Scalar*
The degree of the respective $SU(N)$ group. This must be a positive integer greater than '1'.

`tex` *Scalar*
The TEX string associated with the respective $SU(N)$ group. By default, the group name surrounded by `\mathtt{•}` is used.

`indices` *Sequence*
The list of TEX (fundamental) indices associated with the respective $SU(N)$ group. By default, the characters 'a,b,...,k' are used.

`u1_groups` *Mapping*
Definition of the model's abelian symmetries—realized as $U(1)$ groups. Each entry is a mapping from the symmetry's name (e.g., 'QED' or 'U(1)') to its properties.

- Group names must start with a letter ('A-z').
- Group names can only contain alpha-numeric characters and parentheses ('A-z', '0-9', '(', and ')').
- Group names must end with an alpha-numeric character or a parenthesis ('A-z', '0-9', '(', or ')').

By default, `u1_groups` is the empty mapping '{}'.

`violation` *Scalar*
A unit of charge the operators are **allowed** to deviate from the exact $U(1)$ symmetry. The unit can be an integer value (e.g., '1') or a fractional value (e.g., '1/2'). The `violation` v can be combined with a `residual` charge R, such that only operators with total charge Q satisfying $|Q-R| \leq v$ are allowed. The default value for `violation` is '0'.

`residual` *Scalar*
A unit of charge the operators **must** deviate from the exact $U(1)$ symmetry. The unit can be an integer value (e.g., '1') or a fractional value (e.g., '1/2'). See also `violation`. The default value for `residual` is '0'.

`tex` *Scalar*
The TEX string associated with the respective $U(1)$ group. By default, the group name surrounded by `\mathtt{•}` is used.

`fields` *Mapping*
Definition of the model's field content. Each entry is a mapping from the field's name (e.g., 'Q' or 'H') to its properties.

- Field names must start with a letter ('A-z').
- Field names can only contain alpha-numeric characters ('A-z' and '0-9').
- Field names must end with an alpha-numeric character or a plus ('A-z', '0-9', or '+'). If the environment variable `AUTOEFT_CS` is set, it replaces the plus symbol '+' in the above restrictions (see [Environment], page 17).
- The field name cannot be 'D', as this symbol is reserved for the covariant derivative.

By default, `fields` is the empty mapping '{}'.

`representations` *Mapping*
Definition of the field's representations. Each entry is a mapping from a group name—as defined under `symmetries`—to an irreducible representation associated with this group. Depending on the group, the irreducible representation is expressed as

- an integer/fraction (e.g., '1'/'1/2') denoting the helicity—Lorentz,
- a partition (e.g., '[2,1]') denoting the Young Diagram—$SU(N)$,
- an integer/fraction (e.g., '1'/'1/2') denoting the charge—$U(1)$.

The field is assumed to transform like a singlet under every group defined under `symmetries` that is not explicitly listed under `representations`. Hence, every representation not explicitly defined assumes an appropriate default value ('0' for the helicity/charge and '[]' for the Young Diagram).

`anticommute` *Scalar*
If set to 'False' ('True'), the field is treated as commuting (anticommuting). By default, this property is derived from the field's helicity, respecting spin-statistics.

`conjugate` *Scalar*
Whether to automatically include the conjugate field. By default, this property is derived from the field's representations. If there is at least one complex representation or an odd number of pseudo-real representations, the conjugate field is included as well. Otherwise, only the field which is explicitly defined in the model file is included.

`generations` *Scalar*
The number of copies with the same representations appearing in the model. The number of `generations` must be a positive integer. By default, each field has just a single generation.

`tex` *Scalar*

`tex_hc` *Scalar*
The TEX string associated with the field and its conjugate. By default, `tex` is set to the field name surrounded by `\mathtt{•}`. If `tex_hc` is omitted, it is set to the value of `tex` and—depending on the value of `conjugate`—optionally appended by '`^\dagger`'.

Appendix D. Output files

`version` *Scalar*
The version of `autoeft` used to produce the file.

`type` *Sequence*
Details of the operator type. The first entry denotes how often each field—identified by its name as defined under `fields` in the model file—and the covariant derivative appears in the type. The second entry denotes if the type is 'real' or 'complex' (see [Vocabulary], page 19).

`generations` *Mapping*
The number of generations for each field—identified by its name as defined under `fields` in the model file.

`n_terms` *Scalar*
The total number of independent contractions with definite permutation symmetry of the repeated fields, not including multiple generations.

`n_operators` *Scalar*
The total number of independent contractions with definite permutation symmetry of the repeated fields, including multiple generations.

`invariants` *Mapping*
Details of the type's invariant contractions. Each entry is a mapping from a group name—as defined under `symmetries` in the model file—to the invariant contractions associated with this group. Each independent contraction is labeled by ‘`O(<groupname>,<m>)`’, where ‘`m`’ enumerates the contractions. The indices are denoted by ‘`<i>_<j>`’, where ‘`i`’ is the position of the field that carries this index, and ‘`j`’ is the position of the index *on* the field. Per invariant contraction, each index appears exactly twice and summation is implied. For the Lorentz group, ‘`<i>_<j>~`’ denotes dotted indices. If the environment variable `AUTOEFT_DS` is set, it replaces the tilde symbol ‘`~`’ in the above notation (see [Environment], page 17). Note that the dotted and undotted spinor indices of the building blocks (i.e. fields plus derivatives) are understood to be (separately) symmetrized. The symbol ‘`eps`’ denotes the ϵ-tensor with `eps(1,2)=eps(2~,1~)=1` for the Lorentz group and `eps(1,2,...,n)=1` for any internal $SU(N)$ group. All indices not associated with the symmetry group in question are suppressed on the fields. If the operators contain only fields that are singlets under a particular symmetry group, there is no index to be contracted and the entry contains just a single element ‘`+1`’ multiplied by the fields without indices.

`permutation_symmetries` *Sequence*
Details of the type's permutation symmetries. The first entry is always a mapping from ‘`vector`’ to a product of group names—as defined under `symmetries` in the model file—separated by ‘ `*` ’. This denotes the order of the tensor product of the invariant contractions given under `invariants`. All other entries represent explicit permutation symmetries.

`symmetry` *Mapping*
The permutation symmetry of each field, identified by an integer partition (e.g., ‘`[2,1]`’) corresponding to an irreducible representation of the symmetric group.

`n_terms` *Scalar*
The number of independent contractions respecting the permutation symmetry, not including multiple generations.

`n_operators` *Scalar*
The number of independent contractions respecting the permutation symmetry, including multiple generations.

`matrix` *Scalar*
The matrix representing the independent linear combinations of the invariant contractions respecting the permutation symmetry.

Appendix E. Vocabulary glossary

family
A family represents all operators with the same **Lorentz** representations of the fields (i.e., same *kind* of fields). For each field, the Lorentz representation can be identified with the helicity value h and `autoeft` assigns the following symbols to each representation:

object	helicity	symbol
scalar	0	‘`phi`’
spinor	$-1/2$	‘`psiL`’
	$+1/2$	‘`psiR`’
rank-2 tensor	-1	‘`FL`’
	$+1$	‘`FR`’
rank-4 tensor	-2	‘`CL`’
	$+2$	‘`CR`’
covariant derivative		‘`D`’

Each family is then represented by a string consisting of these symbols, each preceded by its number of occurrences in the family and separated by an underscore ‘`_`’. To identify a family uniquely, the symbols are sorted by their helicity value and the covariant derivative is always added to the end. Representations not appearing in the family are simply dropped.

For example, the family of the dimension-5 Weinberg operator—consisting of two spinors and two scalars—is given by ‘`2psiL_2phi`’ and its conjugate by ‘`2phi_2psiR`’.

type
A type represents all operators with the same **Lorentz** and **internal** representations of the fields (i.e., same *content* of fields). Each type is then represented by a string consisting of the field name—as defined under `fields` in the model file—and each field is preceded by its number of occurrences in the type and separated by an underscore ‘`_`’. To identify a type uniquely, the fields are first sorted by their helicity value and fields with the same helicity are sorted by their name alpha-numerically. The covariant derivative is always added to the end. Representations not appearing in the type are simply dropped.

For example, the type of the dimension-5 Weinberg operator—consisting of two lepton doublets ‘`L`’ and two Higgs doublets ‘`H`’—is given by ‘`2L_2H`’ and its conjugate by ‘`2H+_2L+`’.

term
A term represents all operators with the same explicit contraction of the fields with the invariant tensor structures of the external and internal symmetries, retaining open generation indices.

For the number of terms to be unambiguous, a term is required to have a definite permutation symmetry for all repeated fields. That means, that general terms are always decomposed into their irreducible representations under the symmetric group with respect to the repeated fields.

operator
An operator is a particular instance of a *term* with fixed generation indices for all fields. Since any term has a definite permutation symmetry for any repeated field, the independent operators correspond to the independent components of the respective Wilson coefficient. This means that the number of operators is equal to the independent degrees of freedom of the EFT (Effective Field Theory).

basis
A valid operator *basis* that can be processed further is represented by a directory containing the file `model.json` referencing the model, a (hidden) file `.autoeft` containing metadata, and the respective operator files (see [Output], page 18). Note that, while the files `model.json` and `.autoeft` must be contained in the top-level directory of the basis, the operator files can be structured in subdirectories. By default, the directories called `basis` created by `autoeft construct` compose valid bases.

real family/type
A *real* family contains types that are either real or both the type and its conjugate are part of the same family.

A *real* type contains terms that are either Hermitian or the Hermitian conjugate terms are not independent and can be expressed as a combination of the terms of the same type.

complex family/type
A *complex* family only contains complex types and the conjugate types are part of the distinct conjugate family.

A *complex* type only contains terms that are not Hermitian and the Hermitian conjugate terms can be expressed as a combination of the terms of the distinct conjugate type.

References

[1] The ATLAS Collaboration, A detailed map of Higgs boson interactions by the ATLAS experiment ten years after the discovery, Nature 607 (2022) 52–59, arXiv:2207.00092 [hep-ex], Correction: Nature 612 (2022) E24.

[2] The CMS Collaboration, A. Tumasyan, et al., A portrait of the Higgs boson by the CMS experiment ten years after the discovery, Nature 607 (2022) 60–68, arXiv:2207.00043 [hep-ex].
[3] T. Hambye, K. Riesselmann, Matching conditions and Higgs mass upper bounds reexamined, Phys. Rev. D 55 (1997) 7255, arXiv:hep-ph/9610272.
[4] G. Degrassi, S. Di Vita, J. Elias-Miró, J.R. Espinosa, G.F. Giudice, G. Isidori, A. Strumia, Higgs mass and vacuum stability in the standard model at NNLO, J. High Energy Phys. 08 (2012) 098, arXiv:1205.6497 [hep-ph].
[5] R. Harlander, J.-P. Martinez, G. Schiemann, The end of the particle era?, Eur. Phys. J. H 48 (2023) 6, Correction: Eur. Phys. J. H 48 (2023) 8.
[6] A. Drozd, J. Ellis, J. Quevillon, T. You, The universal one-loop effective action, J. High Energy Phys. 03 (2016) 180, arXiv:1512.03003 [hep-ph].
[7] S.A.R. Ellis, J. Quevillon, T. You, Z. Zhang, Extending the universal one-loop effective action: heavy-light coefficients, J. High Energy Phys. 08 (2017) 054, arXiv:1706.07765 [hep-ph].
[8] M. Krämer, B. Summ, A. Voigt, Completing the scalar and fermionic universal one-loop effective action, J. High Energy Phys. 01 (2020) 079, arXiv:1908.04798 [hep-ph].
[9] U. Banerjee, J. Chakrabortty, S.U. Rahaman, K. Ramkumar, One-loop effective action up to dimension eight: integrating out heavy scalar(s), arXiv:2306.09103 [hep-ph].
[10] J. Chakrabortty, S.U. Rahaman, K. Ramkumar, One-loop effective action up to dimension eight: integrating out heavy fermion(s), arXiv:2308.03849 [hep-ph].
[11] C. Arzt, Reduced effective Lagrangians, Phys. Lett. B 342 (1995) 189–195, arXiv:hep-ph/9304230.
[12] J.C. Criado, M. Pérez-Victoria, Field redefinitions in effective theories at higher orders, J. High Energy Phys. 03 (2019) 038, arXiv:1811.09413 [hep-ph].
[13] R.M. Fonseca, Enumerating the operators of an effective field theory, Phys. Rev. D 101 (2020) 035040, arXiv:1907.12584 [hep-ph].
[14] S. Weinberg, Baryon- and lepton-nonconserving processes, Phys. Rev. Lett. 43 (1979) 1566.
[15] W. Buchmüller, D. Wyler, Effective Lagrangian analysis of new interactions and flavor conservation, Nucl. Phys. B 268 (1986) 621–653.
[16] B. Grzadkowski, M. Iskrzyński, M. Misiak, J. Rosiek, Dimension-six terms in the standard model Lagrangian, J. High Energy Phys. 10 (2010) 085, arXiv:1008.4884 [hep-ph].
[17] L. Lehman, Extending the standard model effective field theory with the complete set of dimension-7 operators, Phys. Rev. D 90 (2014) 125023, arXiv:1410.4193 [hep-ph].
[18] Y. Liao, X.-D. Ma, Renormalization group evolution of dimension-seven baryon- and lepton-number-violating operators, J. High Energy Phys. 11 (2016) 043, arXiv:1607.07309 [hep-ph].
[19] L. Lehman, A. Martin, Hilbert series for constructing Lagrangians: expanding the phenomenologist's toolbox, Phys. Rev. D 91 (2015) 105014, arXiv:1503.07537 [hep-ph].
[20] L. Lehman, A. Martin, Low-derivative operators of the standard model effective field theory via Hilbert series methods, J. High Energy Phys. 02 (2016) 081, arXiv:1510.00372 [hep-ph].
[21] B. Henning, X. Lu, T. Melia, H. Murayama, Hilbert series and operator bases with derivatives in effective field theories, Commun. Math. Phys. 347 (2016) 363–388, arXiv:1507.07240 [hep-th].
[22] B. Henning, X. Lu, T. Melia, H. Murayama, 2, 84, 30, 993, 560, 15456, 11962, 261485, ..: higher dimension operators in the SM EFT, J. High Energy Phys. 08 (2017) 016, arXiv:1512.03433 [hep-ph], Erratum: J. High Energy Phys. 09 (2019) 019.
[23] C.B. Marinissen, R. Rahn, W.J. Waalewijn, ., 83106786, 114382724, 1509048322, 2343463290, 27410087742, ... efficient Hilbert series for effective theories, Phys. Lett. B 808 (2020) 135632, arXiv:2004.09521 [hep-ph].
[24] U. Banerjee, J. Chakrabortty, S. Prakash, S.U. Rahaman, Characters and group invariant polynomials of (super)fields: road to "Lagrangian", Eur. Phys. J. C 80 (2020) 938, arXiv:2004.12830 [hep-ph].
[25] S. Calò, C. Marinissen, R. Rahn, Discrete symmetries and efficient counting of operators, J. High Energy Phys. 05 (2023) 215, arXiv:2212.04395 [hep-ph].
[26] B. Gripaios, D. Sutherland, DEFT: a program for operators in EFT, J. High Energy Phys. 01 (2019) 128, arXiv:1807.07546 [hep-ph].
[27] R.M. Fonseca, The Sym2Int program: going from symmetries to interactions, J. Phys. Conf. Ser. 873 (2017) 012045, arXiv:1703.05221 [hep-ph].
[28] J.C. Criado, BasisGen: automatic generation of operator bases, Eur. Phys. J. C 79 (2019) 256, arXiv:1901.03501 [hep-ph].
[29] J. Aebischer, et al., Computing tools for effective field theories, arXiv:2307.08745 [hep-ph].
[30] C.W. Murphy, Dimension-8 operators in the standard model eective field theory, J. High Energy Phys. 10 (2020) 174, arXiv:2005.00059 [hep-ph].
[31] Y. Liao, X.-D. Ma, An explicit construction of the dimension-9 operator basis in the standard model effective field theory, J. High Energy Phys. 11 (2020) 152, arXiv:2007.08125 [hep-ph].
[32] Y. Shadmi, Y. Weiss, Effective field theory amplitudes the on-shell way: scalar and vector couplings to gluons, J. High Energy Phys. 02 (2019) 165, arXiv:1809.09644 [hep-ph].
[33] B. Henning, T. Melia, Conformal-helicity duality & the Hilbert space of free CFTs, arXiv:1902.06747 [hep-th].
[34] T. Ma, J. Shu, M.-L. Xiao, Standard model effective field theory from on-shell amplitudes, Chin. Phys. C 47 (2023) 023105, arXiv:1902.06752 [hep-ph].
[35] B. Henning, T. Melia, Constructing effective field theories via their harmonics, Phys. Rev. D 100 (2019) 016015, arXiv:1902.06754 [hep-ph].
[36] R. Aoude, C.S. Machado, The Rise of SMEFT on-shell amplitudes, J. High Energy Phys. 12 (2019) 058, arXiv:1905.11433 [hep-ph].
[37] G. Durieux, T. Kitahara, Y. Shadmi, Y. Weiss, The electroweak effective field theory from on-shell amplitudes, J. High Energy Phys. 01 (2020) 119, arXiv:1909.10551 [hep-ph].
[38] A. Falkowski, Bases of massless EFTs via momentum twistors, arXiv:1912.07865 [hep-ph].
[39] G. Durieux, C.S. Machado, Enumerating higher-dimensional operators with on-shell amplitudes, Phys. Rev. D 101 (2020) 095021, arXiv:1912.08827 [hep-ph].
[40] R.M. Fonseca, GroupMath: a mathematica package for group theory calculations, Comput. Phys. Commun. 267 (2021) 108085, arXiv:2011.01764 [hep-th].
[41] H.-L. Li, Z. Ren, J. Shu, M.-L. Xiao, J.-H. Yu, Y.-H. Zheng, Complete set of dimension-eight operators in the standard model effective field theory, Phys. Rev. D 104 (2021) 015026, arXiv:2005.00008 [hep-ph].
[42] H.-L. Li, Z. Ren, M.-L. Xiao, J.-H. Yu, Y.-H. Zheng, Complete set of dimension-nine operators in the standard model effective field theory, Phys. Rev. D 104 (2021) 015025, arXiv:2007.07899 [hep-ph].
[43] H.-L. Li, Z. Ren, M.-L. Xiao, J.-H. Yu, Y.-H. Zheng, Operators for generic effective field theory at any dimension: on-shell amplitude basis construction, J. High Energy Phys. 04 (2022) 140, arXiv:2201.04639 [hep-ph].
[44] R.V. Harlander, T. Kempkens, M.C. Schaaf, Standard model effective field theory up to mass dimension 12, Phys. Rev. D 108 (2023) 055020, arXiv:2305.06832 [hep-ph].
[45] Y. Liao, X.-D. Ma, Operators up to dimension seven in standard model effective field theory extended with sterile neutrinos, Phys. Rev. D 96 (2017) 015012, arXiv:1612.04527 [hep-ph].
[46] Y. Liao, X.-D. Ma, Q.-Y. Wang, Extending low energy effective field theory with a complete set of dimension-7 operators, J. High Energy Phys. 08 (2020) 162, arXiv:2005.08013 [hep-ph].
[47] H.-L. Li, Z. Ren, M.-L. Xiao, J.-H. Yu, Y.-H. Zheng, Low energy effective field theory operator basis at $d \leq 9$, J. High Energy Phys. 06 (2021) 138, arXiv:2012.09188 [hep-ph].
[48] H.-L. Li, Z. Ren, M.-L. Xiao, J.-H. Yu, Y.-H. Zheng, Operator bases in effective field theories with sterile neutrinos: $d \leq 9$, J. High Energy Phys. 11 (2021) 003, arXiv:2105.09329 [hep-ph].
[49] H.-L. Li, Z. Ren, M.-L. Xiao, J.-H. Yu, Y.-H. Zheng, On-shell operator construction in the effective field theory of gravity, arXiv:2305.10481 [gr-qc].
[50] H. Song, H. Sun, J.-H. Yu, Effective field theories of axion, ALP and dark photon, arXiv:2305.16770 [hep-ph].
[51] H. Song, H. Sun, J.-H. Yu, Complete EFT operator bases for dark matter and weakly-interacting light particle, arXiv:2306.05999 [hep-ph].
[52] J.-H. Liang, Y. Liao, X.-D. Ma, H.-L. Wang, Dark sector effective field theory, arXiv:2309.12166 [hep-ph].
[53] M. Ruhdorfer, J. Serra, A. Weiler, Effective field theory of gravity to all orders, J. High Energy Phys. 05 (2020) 083, arXiv:1908.08050 [hep-ph].
[54] T. Han, I. Lewis, T. McElmurry, QCD corrections to scalar diquark production at hadron colliders, J. High Energy Phys. 01 (2010) 123, arXiv:0909.2666 [hep-ph].
[55] J.A.M. Vermaseren, New features of FORM, arXiv:math-ph/0010025.
[56] J. Kuipers, T. Ueda, J.A.M. Vermaseren, J. Vollinga, FORM version 4.0, Comput. Phys. Commun. 184 (2013) 1453–1467, arXiv:1203.6543 [cs.SC].
[57] conda-forge community, The conda-forge project: community-based software distribution built on the conda package format and ecosystem, https://doi.org/10.5281/zenodo.4774216, July 2015.
[58] U. Banerjee, J. Chakrabortty, S. Prakash, S.U. Rahaman, M. Spannowsky, Effective operator bases for beyond standard model scenarios: an EFT compendium for discoveries, J. High Energy Phys. 01 (2021) 028, arXiv:2008.11512 [hep-ph].
[59] R.S. Chivukula, H. Georgi, Composite-technicolor standard model, Phys. Lett. B 188 (1987) 99–104.
[60] G. D'Ambrosio, G.F. Giudice, G. Isidori, A. Strumia, Minimal flavor violation: an effective field theory approach, Nucl. Phys. B 645 (2002) 155–187, arXiv:hep-ph/0207036.
[61] D.A. Faroughy, G. Isidori, F. Wilsch, K. Yamamoto, Flavour symmetries in the SMEFT, J. High Energy Phys. 08 (2020) 166, arXiv:2005.05366 [hep-ph].
[62] A. Greljo, A. Palavrić, A.E. Thomsen, Adding flavor to the SMEFT, J. High Energy Phys. 10 (2022) 010, arXiv:2203.09561 [hep-ph].
[63] A. Greljo, A. Palavrić, Leading directions in the SMEFT, J. High Energy Phys. 09 (2023) 009, arXiv:2305.08898 [hep-ph].
[64] S. Herrlich, U. Nierste, Evanescent operators, scheme dependences and double insertions, Nucl. Phys. B 455 (1995) 39–58, arXiv:hep-ph/9412375.
[65] M. Alminawi, I. Brivio, J. Davighi, Jet bundle geometry of scalar field theories, arXiv:2308.00017 [hep-ph].
[66] N.D. Christensen, C. Duhr, FeynRules - Feynman rules made easy, Comput. Phys. Commun. 180 (2009) 1614–1641, arXiv:0806.4194 [hep-ph].
[67] A. Alloul, N.D. Christensen, C. Degrande, C. Duhr, B. Fuks, FeynRules 2.0 - a complete toolbox for tree-level phenomenology, Comput. Phys. Commun. 185 (2014) 2250–2300, arXiv:1310.1921 [hep-ph].