MATCH2FIT User's Manual

Alejo N. Rossia*

Department of Physics and Astronomy, University of Manchester
Oxford Road, Manchester M13 9PL, United Kingdom

August 4, 2023

*alejo.rossia@manchester.ac.uk

Contents

1 Introduction

The epoch before HL-LHC in theoretical particle physics is marked by our efforts to extract every possible bit of information out of the combined LHC/HL-LHC experimental program. The absence of direct evidence for new fundamental particles calls for the use of Effective Field Theories (EFTs) as the ideal framework to parametrize any deviation from the Standard Model of particle physics.

The most used EFT for phenomenological analyses at LHC is the Standard Model Effective Field Theory (SMEFT). It is built upon the SM, preserves all the defining features of the SM and agrees with it when considering only operators of dimension up to 4.

The use of EFTs allows us to separate the process of constraining a hypothetical UV model in mostly independent steps. The experimental observables have to be computed as a function of the EFT parameters, the so-called Wilson Coefficients (WCs) to the desired precision. Then, global fits accounting for hundreds or more experimental measurements can be used to constrain the EFT parameters. Once one has selected a UV model to constrain, this can be matched to the EFT, resulting in relations between the EFT WCs and the parameters of the UV model. Experimental constraints on the latter can be derived from those relations and the aforementioned global fits.

A plethora of tools has been developed to automate the different steps in the EFT programme, for a recent review see [1]. In particular, several groups have performed global fits of experimental data to SMEFT with varying datasets and degrees of precision in the theoretical predictions [2–8] and most of them have released public versions of their fitting codes. Some of them included results for pre-defined UV models matched to SMEFT. On the other hand, the matching process has been fully automated at tree-level and partially at 1-loop level, even with the use of 2 different techniques [9–12]. However, there is no automatic link between the fitting and matching codes that allows the user to define a UV model, match it and fit its parameters in a few commands. Tools with those features have been developed for scalar-extended BSM models, e.g. see [13], but not within the SMEFT community.

The MATCH2FIT package aims at bridging that remaining gap in the EFT programme by offering a first interface between matching and fitting tools. We have chosen `matchmakereft` [11] and SMEFiT [8] as the matching and fitting codes to interface respectively, with plans to extend the support to other codes on both sides. The companion paper [**?**] explains how SMEFiThas been updated to support fits on the space parameter of the UV models and shows the fit results for several UV models.

In this manual, we explain in detail how to use MATCH2FIT to produce the run cards needed by SMEFiTto fit the UV models. The goal is that any user can install `matchmakereft`, MATCH2FIT, and SMEFiTand then compute the bounds on the UV parameters of their UV model of interest.

2 Conceptual description

The package has 2 working modes. The first one simply reads the matching results in the output format used by `matchmakereft` and parses it to the format of run cards that can be fed into the SMEFiT fitting code. In this mode, the mandatory inputs to the package are the location of the file containing the matching results and a numerical value (in TeV) for the mass of the heavy particle. Optionally, one can also specify a name for the UV model and a name for the "collection", where a collection is a set of UV models with some common characteristics. Depending on the executed function, the code will print the run

card for a scan on the UV parameters with or without the accompanying file that defines the UV invariants (see [**?**]).

The second working mode runs `matchmakereft` to perform the tree-level matching of a certain model to SMEFT and generates the same final output than the previous mode. It is also possible to just perform the matching without producing the run cards for SMEFiT. The input required in this mode is the one that `matchmakereft` needs to describe the heavy particles that will be integrated out. More precisely, it needs the `.fr`, `.red` and `.gauge` files, and optionally the `.red` file. The `.fr` file is just a Feynrules file that defines the heavy particle(s), its (their) free Lagrangian and its (their) interactions with the SM. The SM and SMEFT models are included in `matchmakereft`and we use them without modification. For more details on how to write the `.fr` file and what the other files must contain, see [11].

3 Prerequisites and installation.

This is a Wolfram Mathematica Package designed and tested in version 12.1 or later. To unlock its full functionality, this package requires a working installation of `matchmakereft`, see its installation instructions in the corresponding paper [11] or in its website. Besides the functions that use `matchmakereft`, the rest of the package works in any operative system.

To install this package, clone the Github repository and paste the MATCH2FIT.WL file on **$BaseDirectory**/Applications or **$UserBaseDirectory**/Applications to be able to load it from any notebook. The accompanying Mathematica notebook shows how to load the package and execute the commands with the included sample models and matching results. It also shows how to use the options of each function.

4 Public commands

4.1 Reading matching results

- parametersList [*directory* , *model*]: Both arguments should be strings. This function reads the file `model.fr` in `directory`, recognizes the masses and couplings of the heavy particles to be integrated out, and gives back an array with 2 elements. The first element is a list with the symbolical expression of the masses of the heavy particles. The second element is a list of the couplings defined for these heavy particles, excluding gauge couplings but not self-interactions.

- parametersListFromMatchingResult[*matchResFile*]: It takes as only argument a string with the address of the file containing the matching results. It recognizes the masses and couplings of the heavy particles from those results and gives an output in the same format as parametersList. This code assumes that any parameter with a name starting by m or M corresponds to a mass and identifies any other parameter as a coupling. If the input of parametersListFromMatchingResult is the matching result obtained with the model fed into parametersList, any difference in their outputs should be only due to couplings (or whole particles) that do not affect the tree-level matching result.

- flavourSymChecker[*matchResFile*,**Options**]: It takes the file with matching results specified as input and checks if those results are compatible with the SMEFiTflavour symmetry, $\mathrm{U}(2)_q \times \mathrm{U}(2)_u \times \mathrm{U}(3)_d \times (\mathrm{U}(1)_\ell \times \mathrm{U}(1)_e)^3$. If the constraints are satisfied,

it returns YES. If not, it returns NO and it prints the first WC for which it found a symmetry violation. The option "UVFlavourAssumption" allows the user to specify a replacement list that can be used to apply flavour assumptions on the UV parameters. The left-hand side of the replacement rule should contain some of the UV parameters listed by parametersListFromMatchingResult or parametersList with or without numerical indices, according to how they appear in the matching results file. The code supports up to 4 numerical indices in a single group, i.e. couplings such as gUV, gUV[1], gUV[1, 3] and gUV[1, 3, 2, 4] are supported, but gUV[1][3] or gUV[1][2, 3] are not. The default value of "UVFlavourAssumption" is an empty list. An example of how to set this option is,

```
{"UVFlavourAssumption" -> {gWtiQ[i_, j_] :> KroneckerDelta[i,j] * KroneckerDelta
    [i, 3] gWqf[3, 3] }}
```

- flavourSolver [*matchResFile*, **Options**]: It takes the file with matching results specified as input and tries to solve the constraints imposed by the SMEFiTflavour symmetry, $\mathrm{U}(2)_q \times \mathrm{U}(2)_u \times \mathrm{U}(3)_d \times (\mathrm{U}(1)_\ell \times \mathrm{U}(1)_e)^3$, for the UV couplings. The running time, the number of solutions and their complexity depend on the model. It returns all found solutions. The only Option of this function is "UVFlavourAssumption", which follows the same description given in the function flavourSymChecker. This function considers the SM Yukawa couplings as symbolical variables and they can be set to zero with the option "UVFlavourAssumption".

4.2 Using MatchmakerEFT

- matcher[*directory* , *model*]: it runs `matchmakereft` and performs the tree-level matching without printing any run card for SMEFiT. It takes two strings as arguments, *directory* and *model*. The first one is the directory where the package will look for the files `model.fr`, `model.red` and `model.gauge`. If the code does not find one of those files, it will print a warning. It does not check for the existence of `model.herm`. The expected content of each of those files is specified in the documentation of `matchmakereft` [11]. `matchmakereft` will create the folder `directory/model_MM`, inside which the matching results will be stored as `MatchingResult.dat`. The code will check if `matchmakereft` reported any problem during the matching and will print a warning if so. After performing the matching, the package will remove most of the files and directories created by itself or `matchmakereft` for the sake of tidiness. It will only leave the directory `model_MM` and 2 files inside: `MatchingResult.dat` and `MatchingProblems.dat`.

4.3 Generating the output for SMEFiT

- matchResToUVscanCard[*matchResFile*,*mass*, **Options**]: Function that reads the file with the tree-level matching results and prints the cards required for a UV scan. *matchResFile* must be a string with the exact address of the file that contains the matching results to be used. The format of that file should be exactly like the file `MatchingResult.dat` produced by `MatchmakerEFT`. The argument *mass* should be the value in TeV that the mass(es) of the UV particle(s) will be set to. *mass* can be one numerical value or a list of them $\{m_1, .., m_N\}$, the latter being useful in the case of a multiparticle model. The order of the masses is the one returned by parametersListFromMatchingResult. If the user specifies only one numerical mass value for a multiparticle model, all the particles will be assigned the same mass.

If parametersListFromMatchingResult identifies K masses and $N < K$, the code will assume $m_i = m_N$ for $N \leqslant i \leqslant K$. If $N > K$, the values m_i with $K < i \leqslant N$ will be ignored. The mass values are also printed on the card names and inside the cards. For multiple masses, the convention is to take the integer part of each value and stick them together in sequence. This function has 3 options. The first one is "UVFlavourAssumption", which is identical to the one of the function flavourSymChecker , see its description for details on this option. The second option is "Collection", a string indicating the Collection to which the model belongs. Its default value is "UserCollection". Finally, the option "Model" is a string that specifies the model name to be printed on the run cards, with default value `"UserModel"`. An example of how to set these options is replacing the argument **Options** by:

```
{"UVFlavourAssumption" -> {lambdaT1[a_] :> lambdaT1[3]},
 "Collection" -> "TestMatchingCollection","Model"->"TestModel"}
```

4.4 Combining both steps

Finally, the package includes 2 functions that integrate the steps of matching and printing all the run cards according to the result of said matching.

- modelToUVscanCard[*directory*,*model*,*mass*, **Options**]: Function that takes the files that define the UV model, performs the tree-level matching by running `matchmakereft`, and prints the run cards needed for a UV scan. The first two arguments are exactly like in `matcher`, i.e. the program will look for the files `model.fr`, `model.red`, and `model.gauge` in `directory` and will do the tree-level matching based on them. The argument `model` also defines the name of the model. `mass` should be the value(s) in TeV that the mass(es) of the UV particle(s) will be set to. The handling of several mass values is equal to the function matchResToUVscanCard. This function has 2 options. The first one is "UVFlavourAssumption", which is identical to the one of the function flavourSymChecker, see its description for details on this option. The second option is "Collection", a string indicating the Collection to which the model belongs. Its default value is "UserCollection". An example of how to set these options is replacing the argument **Options** by:

```
{"UVFlavourAssumption" -> {lambdaT1[a_] :> lambdaT1[3]},
 "Collection" -> "TestMatchingCollection"}
```

5 Limitations and outlook

All SM couplings are numerically evaluated when printing the run cards for the global fit. Their values are hard coded in the package and we summarise them in Appendix A. Future versions should allow the user to check and change easily these values.

The code does not check for the fulfilment of the SMEFiT flavour assumptions automatically when printing the run cards for SMEFiT. This generates a degree of arbitrariness, e.g. the code uses the matching result for $(c_{\varphi q}^{(3)})_{22}$ as the result for $(c_{\varphi q}^{(3)})_{ii}$ without checking that $(c_{\varphi q}^{(3)})_{22} = (c_{\varphi q}^{(3)})_{11}$. We provide all these choices in Appendix B. The user should be aware of this and compensate for it with the required assumptions on the UV couplings. The support for different flavour symmetries on the WCs will be added in the future. This is key to ensure full compatibility with SMEFiT and/or be fully compatible with other fitting codes.

The code assumes that all UV couplings are real and applies this assumption when interpreting the matching results. The support for complex UV couplings and WCs will be added in future releases.

An important but more long-term upgrade is to read matching results in the xWCF format. This would allow for interfacing with other matching codes such as CoDEx and Matchete. This would allow for additional interfacing with codes that implement the SMEFT RGE running.

Acknowledgements

The author thanks to M. Chala, G. Magni, Y. Oda, J. Rojo, J. Santiago, C. Severi, J. ter Hoeve, and E. Vryonidou for support and useful discussions during the development of this package. The author is grateful to the High Energy Theory Group of Universidad de Granada and Theory Group of Nikhef for their hospitality during the early stages of this work. The development of this package has been supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 949451) and by a Royal Society University Research Fellowship through grant URF/R1/201553.

A Numerical parameters

The package uses numerical values for SM parameters in order to convert the matching expressions into functions of only the UV parameters to fit. The values are hard-coded inside the code. The default values are shown in Table 1. If the user wishes to change these values, they can do it by modifying the file match2fit.wl, section SM numerical inputs.

Parameter	Value
α_s	0.1179
$\sin^2\theta_W$	0.23121
G_F	$1.1663787 \cdot 10^{-5}$ GeV^{-2}
g_3	$\sqrt{4\pi\alpha_s} = 1.2172$
g_2	$\sqrt{\frac{8}{\sqrt{2}} m_W^2 G_F} = 0.652905$
g_1	$g_2\sqrt{\frac{\sin^2\theta_W}{1-\sin^2\theta_W}} = 0.358055$
m_W	$8.0379 \cdot 10^{-2}$ TeV
m_Z	$9.11876 \cdot 10^{-2}$ TeV
m_h	$1.2525 \cdot 10^{-1}$ TeV
v	0.24622 TeV
λ_h	$\frac{1}{2}\left(\frac{m_h}{v}\right)^2$
m_ψ	$\delta_{\psi,t}\, 0.17276$ TeV
$(y_\psi^{\rm SM})_{ij}$	$\delta_{ij}\sqrt{2}\frac{m_\psi}{v}$

Table 1: Numerical parameters used by the package. Values extracted from [14]

SMEFiT WC	WC read by MATCH2FIT
$c_{\varphi q}^{(-)}$	$\left(c_{\varphi q}^{(1)}\right)_{22} - \left(c_{\varphi q}^{(3)}\right)_{22}$
$c_{\varphi q}^{(3)}$	$\left(c_{\varphi q}^{(3)}\right)_{22}$
$c_{\varphi u_i}$	$\left(c_{\varphi u}\right)_{22}$
$c_{\varphi d_i}$	$\left(c_{\varphi d}\right)_{22}$
$c_{\varphi Qq}^{1,8}$	$\left(c_{qq}^{1}\right)_{1331} + 3\left(c_{qq}^{3}\right)_{1331}$
$c_{\varphi Qq}^{1,1}$	$\left(c_{qq}^{1}\right)_{1133} + \frac{1}{6}\left(c_{qq}^{1}\right)_{1331} + \frac{1}{2}\left(c_{qq}^{3}\right)_{1331}$
$c_{\varphi Qq}^{3,8}$	$\left(c_{qq}^{1}\right)_{1331} - \left(c_{qq}^{3}\right)_{1331}$
$c_{\varphi Qq}^{3,1}$	$\left(c_{qq}^{3}\right)_{1133} + \frac{1}{6}\left(\left(c_{qq}^{1}\right)_{1331} - \left(c_{qq}^{3}\right)_{1331}\right)$
$c_{\varphi tq}^{8}$	$\left(c_{qu}^{8}\right)_{1133}$
$c_{\varphi tq}^{1}$	$\left(c_{qu}^{1}\right)_{1133}$
$c_{\varphi tu}^{8}$	$2\left(c_{uu}\right)_{1331}$
$c_{\varphi tu}^{1}$	$\left(c_{uu}^{8}\right)_{1133} + \frac{1}{3}\left(c_{uu}^{8}\right)_{1331}$
$c_{\varphi Qu}^{8}$	$\left(c_{qu}^{8}\right)_{3311}$
$c_{\varphi Qu}^{1}$	$\left(c_{qu}^{1}\right)_{3311}$
$c_{\varphi td}^{8}$	$\left(c_{ud}\right)_{3333}$
$c_{\varphi td}^{1}$	$\left(c_{ud}^{1}\right)_{3333}$
$c_{\varphi Qd}^{8}$	$\left(c_{qd}^{8}\right)_{3333}$
$c_{\varphi Qd}^{1}$	$\left(c_{qd}^{1}\right)_{3333}$

Table 2: WCs in the Warsaw basis used by MATCH2FITto determine the WCs in the SMEFiTbasis. The definition of the latter ones can be found in [5].

There is an internal switch to turn on the masses and Yukawa couplings of all the third-generation fermions in the SM. The variable is called flavourOption and defined in section SM numerical inputs, subsection Masses and Yukawas, of the file match2fit.wl. The default value "SMEFiTop" enables the top mass as the only non-vanishing one. If the value of this variable is changed to "SMEFiT3", the bottom and τ masses are also set as non-zero, with values $m_b = 4.18$ GeV and $m_\tau = 1.78$ GeV.

B Choices due to flavour symmetry

The SMEFiTflavour symmetry forces certain WCs to be the same for certain values of their flavour indices. The code, when printing the run cards, reads the matching result for one particular value of those indices and assumes that all the other values give the same expression. This procedure leads to ambiguities if the model after matching does not follow the SMEFiTflavour assumption on the WCs even after applying the UV flavour assumptions specified by the user. We provide in Table 2 a list of the flavour indices that the code uses to print the run card so the user can keep track of these possible ambiguities.

References

[1] J. Aebischer *et al.*, *Computing Tools for Effective Field Theories* (2023), 2307.08745.

[2] J. Aebischer, J. Kumar, P. Stangl and D. M. Straub, *A Global Likelihood for Precision Constraints and Flavour Anomalies*, Eur. Phys. J. C **79**(6), 509 (2019), doi:10.1140/epjc/s10052-019-6977-z, 1810.07698.

[3] J. De Blas *et al.*, *HEPfit: a code for the combination of indirect and direct constraints on high energy physics models*, Eur. Phys. J. C **80**(5), 456 (2020), doi:10.1140/epjc/s10052-020-7904-z, 1910.14012.

[4] J. Ellis, M. Madigan, K. Mimasu, V. Sanz and T. You, *Top, Higgs, Diboson and Electroweak Fit to the Standard Model Effective Field Theory*, JHEP **04**, 279 (2021), doi:10.1007/JHEP04(2021)279, 2012.02779.

[5] J. J. Ethier, G. Magni, F. Maltoni, L. Mantani, E. R. Nocera, J. Rojo, E. Slade, E. Vryonidou and C. Zhang, *Combined SMEFT interpretation of Higgs, diboson, and top quark data from the LHC*, JHEP **11**, 089 (2021), doi:10.1007/JHEP11(2021)089, 2105.00006.

[6] I. Brivio, S. Bruggisser, E. Geoffray, W. Killian, M. Krämer, M. Luchmann, T. Plehn and B. Summ, *From models to SMEFT and back?*, SciPost Phys. **12**(1), 036 (2022), doi:10.21468/SciPostPhys.12.1.036, 2108.01094.

[7] Anisha, S. Das Bakshi, S. Banerjee, A. Biekötter, J. Chakrabortty, S. Kumar Patra and M. Spannowsky, *Effective limits on single scalar extensions in the light of recent LHC data*, Phys. Rev. D **107**(5), 055028 (2023), doi:10.1103/PhysRevD.107.055028, 2111.05876.

[8] T. Giani, G. Magni and J. Rojo, *SMEFiT: a flexible toolbox for global interpretations of particle physics data with effective field theories*, Eur. Phys. J. C **83**(5), 393 (2023), doi:10.1140/epjc/s10052-023-11534-7, 2302.06660.

[9] J. C. Criado, *MatchingTools: a Python library for symbolic effective field theory calculations*, Comput. Phys. Commun. **227**, 42 (2018), doi:10.1016/j.cpc.2018.02.016, 1710.06445.

[10] S. Das Bakshi, J. Chakrabortty and S. K. Patra, *CoDEx: Wilson coefficient calculator connecting SMEFT to UV theory*, Eur. Phys. J. C **79**(1), 21 (2019), doi:10.1140/epjc/s10052-018-6444-2, 1808.04403.

[11] A. Carmona, A. Lazopoulos, P. Olgoso and J. Santiago, *Matchmakereft: automated tree-level and one-loop matching*, SciPost Phys. **12**(6), 198 (2022), doi:10.21468/SciPostPhys.12.6.198, 2112.10787.

[12] J. Fuentes-Martín, M. König, J. Pagès, A. E. Thomsen and F. Wilsch, *A proof of concept for matchete: an automated tool for matching effective theories*, Eur. Phys. J. C **83**(7), 662 (2023), doi:10.1140/epjc/s10052-023-11726-1, 2212.04510.

[13] H. Bahl, T. Biekötter, S. Heinemeyer, C. Li, S. Paasch, G. Weiglein and J. Wittbrodt, *HiggsTools: BSM scalar phenomenology with new versions of HiggsBounds and HiggsSignals*, Comput. Phys. Commun. **291**, 108803 (2023), doi:10.1016/j.cpc.2023.108803, 2210.09332.

[14] R. L. Workman *et al.*, *Review of Particle Physics*, PTEP **2022**, 083C01 (2022), doi:10.1093/ptep/ptac097.