`MatchMaker`, Making one-loop matching simple

Charalampos Anastasiou[a], Adrián Carmona[b], Achilleas Lazopoulos[a], Jose Santiago[c]

[a]*Institute for Theoretical Physics, ETH Zürich, 8093 Zürich, Switzerland, Emails: babis@phys.ethz.ch, lazopoulos@itp.phys.ethz.ch*
[b]*CERN, Theoretical Physics Department, 1211 Geneva 23, Switzerland, Email: adrian.carmona@cern.ch*
[c] *Departamento de Física Teórica y del Cosmos and CAFPE, Universidad de Granada, E-18071 Granada, Spain, Email: jsantiago@ugr.es*

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Keywords: Effective field theory, One-loop matching

PROGRAM SUMMARY

Manuscript Title: `MatchMaker`, Making one-loop matching simple
Authors: Charalampos Anastasiou, Adrián Carmona, Achilleas Lazopoulos, Jose Santiago
Program title: `MatchMaker`
Programming language: `Python 2.7`, `FORM`, `QGRAF`, `Mathematica`
Computer: Platforms on which the used languages are available.
Operating systems: Operating systems on which the used languages are available.
Keywords: Effective field theory, One-loop matching.
CPC Library Classification:
External routines/libraries:
Nature of problem:
Solution method: Restrictions: `Mathematica` version 10 or higher. `Python 2.7`. `pip` for `Python 2.7`.
Unusual features: None.
Running time:

1. Introduction

bla, blah, intro to why EFTs are important, bottom-up vs top-down, we do top-down in an automated way up to one loop. Current version only for the SMEFT but the algorithms `MatchMaker` is based on will be applicable to a wider range of theories and EFTs.

2. Installation

An updated version of this manual can be found, together with some sample BSM `MatchMaker` models, in https://cafpe.ugr.es/matchmaker/. Once `MatchMaker` is installed it can also be found in the directory

`Match_Maker-location/matchmaker/docs/`

where `Match_Maker-location` is the directory listed under `Location` when the command `pip show Match_Maker` is used (see below).

2.1. Prerequisites

In order to be able to run `MatchMaker`, some prerequisites need to be met. First of all, the following programs need to be installed:

- `Mathematica` version 10 or higher.
- `FORM` : Installation can be checked by typing `form -v` in a terminal. Binaries or the source code can be downloaded from http://www.nikhef.nl/~form/
- `QGRAF` : Installation can be checked by typing `qgraf` in a terminal. Binaries or the source code can be downloaded from http://cfif.ist.utl.pt/~paulo/qgraf.html. Installation from source can be easily achieved by byping in a terminal

  ```
  > (COMPILER) -o qgraf qgraf-3.xxx.f
  ```

  where `(COMPILER)` is any fortran compiler (for instance `gfortran`) and then by copying the `QGRAF` executable in a directory included in the path.

The binaries of both `FORM` and `QGRAF` need to be located in some path that is included in the binary path of the system, in such a way that they can be executed from any possible location. Finally, it is also necessary to have installed

- `Python` (`2.7` or higher, including `3.x`): Installation can be checked by typing `python --version` in a terminal. In most Linux distributions it will be install by default. In Debian/Ubuntu this can be done e.g. by writing

```
sudo apt-get install python
```

- `pip`. Installation can be checked by typing `pip -V` in a terminal. In Debian/Ubuntu this can be done by writing

  ```
  sudo apt-get install python-pip python-dev build-essential
  ```

- `virtualenv` : This is not compulsory but it can be useful to avoid possible conflicts between `Python` packages. In Debian/Ubuntu it can be installed by writing

  ```
  sudo apt-get install python-virtualenv
  ```

Finally, although technically not compulsory, there are two mathematica packages that greatly simplify the creation of `MatchMaker` models and their matching. These are:

- `FeynRules` : Can be used to simplify the creation of `MatchMaker` models. It can be downloaded from https://feynrules.irmp.ucl.ac.be/.
- `Susyno` : Can be used to provide the explicit group theory factors involved in the `MatchMaker` models. The code and installation instructions can be found in http://renatofonseca.net/susyno.php.

2.2. Installing `MatchMaker`

Once `pip` is installed in the system, `MatchMaker` can be installed by just typing in a terminal (we denote the terminal prompt as `>`)

```
> pip install Match_Maker --user
```

We can get information about `MatchMaker` by writing

```
> pip show Match_Maker
Name: Match-Maker
Version: 0.1.2
Summary: One loop matching
Home-page: https://cafpe.ugr.es/matchmaker/
Author: Charalampos Anastasiou, Adrian Carmona,
Achilleas Lazopoulos, Jose Santiago
Author-email: babis@phys.ethz.ch, adrian.carmona@cern.ch,
lazopoulos@itp.phys.ethz.ch, jsantiago@ugr.es
License: Creative Commons Attribution-Noncommercial-Share Alike license
Location: /home/adrian/.local/lib/python2.7/site-packages
Requires:
```

If `MatchMaker` is already installed in the system, it is possible to check for possible updates by writing

```
> pip install --upgrade Match_Maker --user
```

whereas one can remove it by making

```
> pip uninstall Match_Maker
```

2.3. Updating the system path

Once `MatchMaker` is installed, we have to ensure that the corresponding executable is included in the user path. This can be easily achieved by copying the following two lines in the `~/.bashrc` file (or the equivalent one if a different shell is used)

```
export PY_USER_BIN=$(python -c 'import site; print(site.USER_BASE + "/bin")')
export PATH=$PY_USER_BIN:$PATH
```

If the user prefers to install without the `--user` option (s)he should ensure that the directory where the executables are installed is included in the path. Finally, the `Mathematica` packages included in `MatchMaker` should be included in the `Mathematica` path. This can be done by adding to the User Mathematica Initialization file the following line

```
$Path = Join[{FileNameJoin[{"path-to-location","matchmaker","core"}]}, $Path]
```

where `path-to-location` is the directory under Location when the command `pip show Match_Maker` is issued. In the example above it corresponds to

```
/home/adrian/.local/lib/python2.7/site-packages
```

The location of the User Mathematica Initialization file can be found by running the following command in a Mathematica Notebook:

```
FileNameJoin[{$UserBaseDirectory, "Kernel", "init.m"}]
```

and is typically in `~/.Mathematica/Kernel/init.m` for a linux distribution and in `~/Library/Mathematica/Kernel/init.m` for a mac os one.

3. Running `MatchMaker`

`MatchMaker` can be started from any directory (provided that the correspondings executables are included in the path as described in the previous section) by simply typing in a terminal

```
> MatchMaker
```

This will perform some basic checks, including whether a newer version is available and if `FORM` and `QGRAF` are installed in the system, and opens an interactive command prompt

```
MatchMaker>
```

In this prompt help can be obtained by using the command `help`

```
MatchMaker> help

Documented commands (type help <topic>):
========================================
EOF  download_models  exit  help  hist  match_model  quit  shell

MatchMaker>
```

The two most relevant commands are:

- `download_models path-to-download-directory` downloads the file `MatchMaker.tar.gz` inside the directory `path-to-download-directory` (tab completion is activated). This file contains (upon extraction) the directory `MatchMaker` with some examples of `FeynRules` models that can be used to create `MatchMaker` models and two mathematica notebooks, to create `MatchMaker` models out of `FeynRules` models and to compute Wilson coefficients in a `MatchMaker` model that has been already matched.

- `match_model path-to-Match_Maker-model` computes all the relevant amplitudes to match the `MatchMaker` model in `path-to-Match_Maker-model` (tab completion activated).

3.1. Creating `MatchMaker` models from `FeynRules` models

It is strongly recommended to use `FeynRules` to automatically create `MatchMaker` models. This will minimize the possibility of errors and greatly simplify the creation of the model. Nevertheless `MatchMaker` models can also by directly written by the user and a detailed description of their structure is given in the appendix.

The current version of `MatchMaker` supports new physics models that consist of extensions of the SM in the unbroken electrowek phase. In the directory obtained by means of the `download_models` command we distribute two files called `UnbrokenSM_BFM_Particle_Content.fr` and `UnbrokenSM_BFM_Lagrangian.fr` that contain the definition of the SM in the unbroken phase and in the Background field method version of the Feynman gauge. These two models have to be always loaded into `FeynRules` before the model containing the corresponding extension. In the same distribution we also provide as an example two models corresponding to simple SM extensions:

- `VL_Quark_Singlet_Y_2_3_BFM.fr` containing a vector-like quark singlet with hypercharge 2/3.

- `Scalar_Singlet_BFM.fr` containing a hypercharge zero scalar singlet.

A `MatchMaker` model can be easily created by means of the `Mathematica` notebook `create_MM_model.nb` also provided with the models directory. Nevertheless we reproduce here its contents for simplicity.

```
thisdir = NotebookDirectory[];
$FeynRulesPath =
  SetDirectory["path-to-feynrules"];
<< FeynRules`;
SetDirectory[thisdir];
FR$Loop = True;

LoadModel["UnbrokenSM_BFM_Particle_Content.fr",
          "UnbrokenSM_BFM_Lagrangian.fr",
          "Scalar_Singlet_BFM.fr"]
```

```
<< FR2MM`;

WriteMM[Ltot]
```

It is important to note that the package `FR2MM` has to be in the `Mathematica` path as described in Section 2 and that it should be installed after the model is loaded into `FeynRules`. This package provides the function `WriteMM[Ltot,MMModelName]` that writes a `MatchMaker` model in a directory `MMModelName` (this argument is optional, if not included `M$ModelName<>"_MM"` is used) from the `FeynRules` model loaded with Lagrangian `Ltot`.

3.2. Computing Wilson Coefficients from a matched `MatchMaker` model

Once the model is matched via the `MatchMaker` command `match_model` the Wilson coefficients can be computed by typing the following in a `Mathematica` notebook:

```
Get["matcher`"];
modeldir="full-path-to-matchmaker-model";
ReplaceGaugeData[modeldir];
TotalMatcher[modeldir];
```

where `full-path-to-matchmaker-model` is the absolute path of the directory containing the matched `MatchMaker` model. For simplicity we provide the mathematica notebook `read_results.nb` with the working directory downloaded by `MatchMaker` via the command `download_models`.

4. How the matching is performed in `MatchMaker`

The matching in `MatchMaker` is performed off-shell, requiring that all one-light-particle-irreducible (1lPI) Green functions agree in the full and effective theories. The actual matching is performed in two steps:

1. **Amplitude calculation**. The calculation of all the 1lPI amplitudes needed to perform the full matching is performed via the `MatchMaker` command `match_model MMmodel`. The process is as follows. Generic expressions for the 1lPI (with at least one heavy particle) Green functions are generated by `QGRAF` (at tree level for the EFT and both at tree and one-loop levels for the full theory). The resulting amplitudes are dressed by `MatchMaker` using the data in the `MatchMaker` model and fed to `FORM` for the actual calculation of the amplitudes. The manipulations performed by form are identical for tree and one-loop calculations and proceed as follows:
   - **Expansion in external momenta**. The integrand is expanded in a power series in the external moenta, up to the order corresponding

to dimension-six operators. This is done by by iterating the following identity for the propagators

$$\frac{1}{(k+p)^2-m^2} = \frac{1}{k^2-m^2}\left[1-\frac{p^2+2k\cdot p}{(k+p)^2-m^2}\right], \tag{1}$$

after which all external momenta appears as powers outside the momentum integration.

- **Tensor reduction**. Tensor reduction is performed by means of the following identities

$$k^{\mu_1}k^{\mu_2} = g^{\mu_1\mu_2}\frac{k^2}{D}, \tag{2}$$

$$k^{\mu_1}k^{\mu_2}k^{\mu_3}k^{\mu_4} = g^{\mu_1\mu_2\mu_3\mu_4}\frac{k^4}{D^2+2D}, \tag{3}$$

$$k^{\mu_1}\dots k^{\mu_6} = g^{\mu_1\cdots\mu_6}\frac{k^6}{D^3+6D^2+8D}, \tag{4}$$

$$k^{\mu_1}\dots k^{\mu_8} = g^{\mu_1\cdots\mu_8}\frac{k^8}{D^4+12D^3+44D^2+48D}, \tag{5}$$

  where $g^{\mu_1\cdots\mu_n}$ is the totally symmetric combination of metric tensors.

- **Dirac algebra**. We have to describe in detail what we do here with traces, gamma 5, open lines, etc.

- **Partial fractioning**. The following identity is used to separate propagators with different masses

$$\frac{1}{(k^2-m_1^2)(k^2-m_2^2)} = \frac{1}{m_1^2-m_2^2}\left[\frac{1}{k^2-m_1^2}-\frac{1}{k^2-m_2^2}\right], \tag{6}$$

  where one of the two masses can be vanishing.

- **Integration by parts**. After partial fractioning scaleless integrals are set to zero and the following identity is used to reduce the massive integrals to tadpoles

$$\frac{1}{(k^2-m^2)^{n+1}} = \frac{D-2n}{2nm^2}\frac{1}{(k^2-m^2)^n}. \tag{7}$$

  At this point we are left with a tadpole integral

$$a_0(m) = \int \frac{1}{k^2-m^2} = \mathrm{i}\frac{m^2}{16\pi^2}\left[\frac{1}{\bar{\epsilon}}+1-\log\left(\frac{m^2}{\mu^2}\right)\right]. \tag{8}$$

  The resulting amplitudes are written in a format suitable for further processing.

2. **Wilson Coefficient calculation**. The final step in the calculation is the computation of the Wilson coefficients, which is performed via the

`MatchMaker` command `compute_wilson_coefficients MMmodel`. This is performed, currently with `Mathematica`, as follows. Every amplitude that is present in the full and EFT model directories is loaded and Laurent expanded around $\bar{\epsilon} = 0$. The pole part in $\bar{\epsilon}$ is removed and the amplitudes in the full and effective theories are subtracted. The finite part (back to $D = 4$) is then expanded in all possible kinematic structures and the coefficients of every such kinematic structure is set to zero. This gives an over-constrained system of equations for the Wilson coefficients that is solved for further cross-check. An example will be provided below (we'll provide the full kinematic structure of the four-higgs-two-derivative example).

A few remarks on the process of matching are due.

- Since we match off-shell Green functions we have to include redundant operators in the process of matching. We have defined the SMEFT model containing the full set of operators, including those that are redundant via EoM. Once the full off-shell matching is performed, the Wilson coefficients of redundant operators are written in terms of the minimal basis of choice. The corresponding list of reduntant operators and their relation to a minimal basis are provided for the SMEFT by `MatchMaker`. This process makes it trivial to express the results in any basis chosen by the user. Users who need to define a different EFT model have to provide both.

- The EFT model has to contain all operators of dimension 6 *and less*, and they all have to be properly matched. In particular, operators of dimension 2 and 4, with the structure of the operators in the SM Lagrangian but with arbitrary coefficients. This corresponds to the calculation of *decoupling constants* and, for the case of the matching of kinetic terms, has a direct implication on the tree-level dimension-6 operators, as the new contribution has to be reabsorbed via wave-function renormalization.

- Evanescent operators (operators that vanish for $D = 4$ but not for arbitrary D) are also included. The complete list of operators that would be redundant due to relations that are valid only in $D = 4$ are kept in `MatchMaker` and included in the matching. Again, the relation between the different operators is provided by a separate file. Thus, the adoption of any specific basis of evanescent operators is trivial in `MatchMaker`.

- The matching is performed by subtracting renormalized (*i.e.* UV-finite) Green functions in the full and EFT theories. The IR behaviour, including IR divergences and non-analytic dependence on light masses or external momenta, is identical on the full and EFT sides and cancells in the matching. The only logs that can appear in the matching is $\log \mu/M_i$ with M_i the mass of one of the particles we are integrating out. Thus, the Wilson coefficients are automatically finite.

- In the current version of `MatchMaker`, all particles in the EFT side are massless. Thus, after the expansion in the external momenta all one-loop integrals on the EFT side are scaleless and therefore vanish. Given this, no one-loop calculation is currently performed on the EFT side.

4.1. On the UFO models

It is the responsibility of the user to provide correct UFO models. If these models are generated using FeynRules, some care should be exercise when implementing the models. In particular, here are some common potenial sources of errors:

- Ensure the masses for the heavy particles are implemented in the proper way, using `Mass->{MHeavy,Internal}`.

- Define tensorial parameters explicitly as complex whenever they are. FeynRules takes them complex by default but when the UFO model is written, they are translated to scalar paremters which are assumed real unless otherwise explicitly defined.

5. Technical aspects

5.1. Renormalization of $\mathcal{L}_4$ and gauge invariance

In the process of matching $\mathcal{L}_4$ can also receive contributions from the heavy physics. This amounts to two effects. First, some couplings will be redefined, receiving contribution from the new particles; second, the kinetic terms of the light particles might receive corrections which, after canonical normalization will contribute to all tree level processes.

Another aspect that is relevant in the process of matching is that gauge invariance guarantees that all contributions can be written in terms of gauge covariant derivatives. However, there is still the possibility of a universal renormalization of the corresponding couplings. To give a concrete example let us consider the kinetic term of a light fermion

$$\mathcal{L}_4 = \bar{\psi}\mathrm{i}\not{D}\psi + \ldots = \bar{\psi}[\mathrm{i}\not{\partial} + g\not{A}]\psi + \ldots \ . \tag{9}$$

Once we integrate out the new physics at one loop, the dimension-4 effective Lagrangian will have the general form

$$\mathcal{L}_4^{\text{eff}} = \alpha_\psi \bar{\psi}[\mathrm{i}\not{\partial} + \tilde{g}\not{A}]\psi + \ldots \ . \tag{10}$$

This same feature, the possibility of a universal rescaling of the gauge coupling constants, also occurs at higher dimensions and has to be taken into account in the process of matching.

What `MatchMaker` does is to first consider the matching of $\mathcal{L}_4$ by computing all relevant amplitudes but considering only one per gauge invariant operator. This is used to compute the different Wilson coefficients in $\mathcal{L}_4$ (the α_ψ in the example above). We then include at least three (one per gauge coupling)[1] amplitudes related to the ones used before by gauge invariance. In this way we fix the value of $\tilde{g}$.

5.2. Global sign convention in QGRAF

An important technical aspect of this process is the global sign convention used in QGRAF. In the case of fermions, the above process requires the use of both two and three point functions to compute the renormalization of the coupling constant. Two point functions at tree level are not computed by QGRAF and have to be hard-coded in `MatchMaker`. In order to get a consistent sign convention for the covariant derivative, the global sign convention in QGRAF and in the hard-coded two point function have to be the same. `MatchMaker` uses all incoming particles and QGRAF conventionally writes those in decreasing order for the indices, which in practice means in inverse order with respect to the one used to define the Green function. Thus, an extra global (-1) sign has to be implemented in the two point function. In particular, for the example above, the corresponding two point function reads (recall that all particles are incoming, so $p_1 + p_2 = 0$)

$$\bar{\psi}\psi : (-1)\alpha_\psi \bar{v}(p_1) \mathrm{i}\not{p}_2 u(p_2), \tag{11}$$

where the (-1) sign comes from the QGRAF convention.

Acknowledgments

We would like to thank useful discussions with F. Águila, C. Bobeth, M. Ciuchini, A. Freytas, M. Gorbahn, U. Haisch, Z. Kunstz, U. Nierste, A. Signer, M. Trott.

[1]We actually use a redundant set of amplitudes to guarantee an appently zero contribution from a zero value of the corresponding Wilson coefficient.