MatchingTools: a Python library for symbolic effective field theory calculations

Juan Carlos Criado,
University of Granada

April 16, 2018

Introduction

Matching

$$\text{High-energy theory} \xrightarrow{\text{Integrate out fields with } M > \Lambda} \text{Effective theory with cutoff } \Lambda$$

Removing redundancies

- Group theory identities.
- Integration by parts.
- Field redefinitions (EOMs for the light fields).

Tree-level matching

From an action:

$$S(\phi, \Phi) = -\int \Phi^\dagger \mathcal{D} \Phi + S_{int}(\phi, \Phi)$$

Iterative solution of the EOMs ($1/M_\Phi$ expansion):

$$\Phi = \mathcal{D}^{-1} \frac{\delta S_{in}}{\delta \Phi^\dagger}(\phi, \Phi) \quad \longrightarrow \quad \Phi_c(\phi)$$

Effective action:

$$S_{\text{eff}}(\phi) = S\left(\phi, \Phi_c(\phi)\right)$$

Application: from general extensions of the SM to the SMEFT

Complete tree-level dictionary [1711.10391]

Extensions with new particles $\longleftrightarrow$ Dimension-6 SMEFT

We consider a general SM extension with:

- $SU(3) \times SU(2) \times U(1)$-invariance.
- SM fields $+$ all new ones with contributions to the dimension-6 SMEFT.

and integrate out all the new fields.

Application: from general extensions of the SM to the SMEFT

Size of the problem:

- 47 new fields (apart from the SM ones).
- The interaction Lagrangian contains hundreds of terms.
- After integration, before simplifying, thousands of terms.

`MatchingTools` helps reducing the possibility of errors and the time of the calculations.

Outline

1	**Overview**	Organization, basic tools, matching, extras.
2	**Toy example**	Integrating out a vector-like quark in a toy model.
3	**Application**	From general extensions of the SM to the dim-6 SMEFT
4	**Links**	Repository, installation.
5	**Conclusions**	Future work, summary.

Overview

Organization of the library

`matchingtools`

`.core`	The tools for defining a model and the basic simbolic tensor algebraic operations.
`.integration`	The classes to define heavy fields and the function `integrate`.
`.transformations`	The functions `apply_rules`, `simplify`, ...
`.output`	The class `Writer`, which provides methods for representing results in plain text or LaTeX.
`.extras`	Package with tools for SMEFT applications.

Core tools

Basic objects

- Lagrangians: sums of terms (operators).
- Operators: products of tensors with arbitrary index contractions.

Basic operation

Search and replace a pattern in each term. Examples:

- Substitute heavy field by its EOM solution.
- Group theory identities (e.g. $\sigma^a_{ij}\sigma^a_{kl} \to 2\delta_{il}\delta_{kj} - \delta_{ij}\delta_{kl}$)

Encoding polynomials of tensors

Tensors (fields) with some indices:

```
    tensor(index_label_1, index_label_2, ...)
```

Operators (products of tensors):

```
    Op(tensor1(i1, i2, ...), tensor2(j1, ...), ...)
```

Lagrangians (and other sums of operators) as:

```
    OpSum(operator_1, operator_2, ...)
```

Index labelling. Repeated indices

Repeated index labels

Non-negative integers repeated exactly twice inside each operator to indicate contraction.

$$\mathcal{L}_{example} = \phi_{ij}\psi_{iab}F_{jab} + X_{mn}Y^{nm}$$

```
Lexample = OpSum(
    Op(phi(0, 1), psi(0, 2, 3), F(1, 2, 3)),
    Op(X(0, 1), Y(1, 0)))
```

Index labelling. Free indices

Free index labels

Negative integers. Used in substitution rules: match the ones in the pattern with the ones in the replacement.

$$\sigma^a_{ij}\sigma^a_{kl} \to 2\delta_{il}\delta_{kj} - \delta_{ij}\delta_{kl}$$

```
SU2Fierz = (
    Op(sigma(0, -1, -2), sigma(0, -3, -4)),
    OpSum(
        2 * Op(kdelta(-1, -4), kdelta(-3, -2)),
        - Op(kdelta(-1, -2), kdelta(-3, -4))))
```

Derivatives

Notation for (covariant) derivatives:

```
D(vector_index_label, field(i1, i2, ...))
```

Example:

$$\mathcal{O} = V^{\mu}\phi_i D_{\mu}\phi_i$$

```
O = Op(V(0), phi(1), D(0, phi(1)))
```

Tree-level matching using `MatchingTools`

Tree-level matching in any **Lorentz invariant** field theory.

Heavy fields

- Predefined: scalars, Dirac or Majorana fermions and vectors.
- Other kinds of heavy fields can be added by the user.

The user specifies (unrestricted in priciple):

- Order in $1/M$ for the solution of the EOMs.
- Max. dim. for the operators in the effective Lagrangian.

The `extras` subpackage

`extras`	
`.SU2`	Common tensors and identities for $SU(2)$.
`.SU3`	Common tensors and identities for $SU(3)$.
`.Lorentz`	Common tensors and identities for the Lorentz group.
`.SM`	The SM fields and their equations of motion.
`.SM_dim_6_basis`	Definition of a basis for the dimension-6 SMEFT [arXiv:1412.8480]. (Other bases will be included soon in other modules)

Workflow

1. Define fields and coupling constants.
2. Define the **interaction Lagrangian**.
3. Specify which fields are heavy.
4. **Integrate out** the heavy fields.
5. Define substitution rules **rewrite** the effective Lagrangian and apply them.
6. [Define LaTeX representation of the coupling constants and Wilson coefficients and output to a `.tex` file].

Toy example

Example: integrating out a vector-like quark

Consider an extension of the SM with vector-like quark doublet Q of hypercharge $7/6$ and interaction Lagrangian:

$$\mathcal{L}_{\text{int}} = -(y_Q)_i \bar{Q}_L \phi u_{Ri} + \text{h.c.}$$

When integrated out, it'll give contributions to the effective Lagrangian as $\sim \bar{u}_R \phi^\dagger \not{D}(\phi u_R) \sim \bar{u}_R \phi^\dagger (\not{D}\phi) u_R + \bar{u}_R \phi^\dagger \phi (\not{D} u_R)$.

We can then use the EOM of u_R to write the result in terms of:

$$\mathcal{O}_{\phi u} = \bar{u}_R \phi^\dagger (\not{D}\phi) u_R,$$
$$\mathcal{O}_{u\phi} = \bar{q}_L \tilde{\phi} u_R \phi^\dagger \phi.$$

Definition of the model and matching

```
phi = FieldBuilder('phi', 1, boson)
phic = FieldBuilder('phic', 1, boson)

uR = FieldBuilder('uR', 1.5, fermion)
uRc = FieldBuilder('uRc', 1.5, fermion)

QL = FieldBuilder('QL', 1.5, fermion)
QLc = FieldBuilder('QLc', 1.5, fermion)
QR = FieldBuilder('QR', 1.5, fermion)
QRc = FieldBuilder('QRc', 1.5, fermion)

yQ = TensorBuilder('yQ')
yQc = TensorBuilder('yQc')

Lint = -OpSum(
    Op(yQ(0), QLc(1, 2, 3), phi(3), uR(1, 2, 0)),
    Op(yQc(0), uRc(1, 2, 0), phic(3), QL(1, 2, 3)))

heavy_Q = VectorLikeFermion(
    'Q', 'QL', 'QR', 'QLc', 'QRc', 3, has_flavor=False)
Leff = integrate([heavy_Q], Lint, 6)

Leff_writer = Writer(Leff, {})
print(Leff_writer)
```

Rewriting operators and defining a basis

```
isigma2 = TensorBuilder("isigma2")
yu = TensorBuilder("yu"); yu_dagger = TensorBuilder("yu_dagger")
qL = FieldBuilder("qL", 1.5, fermion); qLc = FieldBuilder("qLc", 1.5, fermion)

rules_uR_eom = [
    (Op(sigma4(0, -1, 1), D(0, uR(1, -2, -3))),
     OpSum(Op(yu_dagger(-3, 0), isigma2(1, 2), phi(2), qL(-1, -2, 1, 0)))),
    (Op(sigma4(0, 1, -1), D(0, uRc(1, -2, -3))),
     OpSum(Op(yu(0, -3), isigma2(1, 2), phic(2), qLc(-1, -2, 1, 0))))]

Ophiu = flavor_tensor_op("Ophiu"); Ophiuc = flavor_tensor_op("Ophiuc")
Ouphi = flavor_tensor_op("Ouphi"); Ouphic = flavor_tensor_op("Ouphic")

rules_ops = [
    (Op(uRc(0, 1, -1), phic(2), sigma4(3, 0, 4), D(3, phi(2)), uR(4, 1, -2)),
     OpSum(Ophiu(-1, -2))),
    (Op(uRc(0, 1, -2), D(3, phic(2)), sigma4(3, 0, 4), phi(2), uR(4, 1, -1)),
     OpSum(Ophiuc(-1, -2))),
    (Op(qLc(0, 1, 2, -1), isigma2(2, 3), phic(3), uR(0, 1, -2), phic(4), phi(4)),
     OpSum(Ouphi(-1, -2))),
    (Op(uRc(0, 1, -2), qL(0, 1, 2, -1), isigma2(2, 3), phi(3), phic(4), phi(4)),
     OpSum(Ouphic(-1, -2)))]

Lfinal = apply_rules(Leff, rules_uR_eom + rules_ops, 1)
```

Outputting

```python
Lfinal_writer = Writer(Lfinal, ["Ophiu", "Ophiuc", "Ouphi", "Ouphic"])
print(Lfinal_writer)

latex_structures = {
    "yQ": "(y_Q)_{}", "yQc": "(y_Q)^*_{}",
    "yu": "(y_u)_{{{}{}}}", "yu_dagger": "(y_u)^\dagger_{{{}{}}}",
    "MQ": "M_Q"}

latex_ops = {
    "Ophiu": r"(C_{{\phi u}})_{{{}{}}}",
    "Ophiuc": r"(C_{{\phi u}})^*_{{{}{}}}",
    "Ouphi": r"(C_{{u \phi}})_{{{}{}}}",
    "Ouphic": r"(C_{{u \phi}})^*_{{{}{}}}"}

Lfinal_writer.write_latex(
    "VLQ_example", latex_structures, latex_ops,
    list(map(chr, range(ord('i'), ord('z')))))
```

Output

The final LaTeX output of the program is:

$$(C_{\phi u})_{ij} = + \frac{i(y_Q)_i^*(y_Q)_j}{2\left(M_Q\right)^2}$$

$$(C_{\phi u})_{ij}^* = - \frac{i(y_Q)_i(y_Q)_j^*}{2\left(M_Q\right)^2}$$

$$(C_{u\phi})_{ij} = - \frac{i(y_u)_{ik}(y_Q)_j(y_Q)_k^*}{2\left(M_Q\right)^2}$$

$$(C_{u\phi})_{ij}^* = + \frac{i(y_u)_{ki}^\dagger(y_Q)_j^*(y_Q)_k}{2\left(M_Q\right)^2}$$

An application to a complex case

From general extensions of the SM to the SMEFT

Complete tree-level dictionary [1711.10391]

Extensions with new particles $\longleftrightarrow$ Dimension-6 SMEFT

Size:

- ~ 50 multiplets to integrate out.
- $1000 - 10000$ terms in some intermediate steps.

`MatchingTools` takes **less than a minute** in a i5 to do the complete calculation.

UV/IR dictionary: bottom-up

$$
\begin{aligned}
(C_{le})_{ijkl} = & -\frac{(y_\varphi^e)^*_{rli}(y_\varphi^e)_{rkj}}{2M_{\varphi_r}^2} - \frac{(g_\mathcal{B}^e)_{rkl}(g_\mathcal{B}^l)_{rij}}{M_{\mathcal{B}_r}^2} + \frac{(g_{\mathcal{L}_3})^*_{rki}(g_{\mathcal{L}_3})_{rlj}}{M_{\mathcal{L}_{3r}}^2} \\
& - \frac{\hat{y}_{li}^{e*}(\delta_{\mathcal{L}_1\varphi})_{sr}(\gamma_{\mathcal{L}_1})^*_s(y_\varphi^e)_{rkj}}{2M_{\varphi_r}^2 M_{\mathcal{L}_{1s}}^2} - \frac{\hat{y}_{kj}^{e}(\delta_{\mathcal{L}_1\varphi})^*_{sr}(\gamma_{\mathcal{L}_1})_s(y_\varphi^e)^*_{rli}}{2M_{\varphi_r}^2 M_{\mathcal{L}_{1s}}^2} \\
& - \frac{\hat{y}_{kj}^{e}\hat{y}_{li}^{e*}(\delta_{\mathcal{L}_1\varphi})^*_{ts}(\gamma_{\mathcal{L}_1})_t(\delta_{\mathcal{L}_1\varphi})_{rs}(\gamma_{\mathcal{L}_1})^*_r}{2M_{\mathcal{L}_{1r}}^2 M_{\varphi_s}^2 M_{\mathcal{L}_{1t}}^2} \\
& + \frac{1}{f}\left\{ \frac{\hat{y}_{li}^{e*}(\tilde{g}_{\mathcal{L}_1}^{eDl})_{rkj}(\gamma_{\mathcal{L}_1})^*_r}{4M_{\mathcal{L}_{1r}}^2} + \frac{\hat{y}_{li}^{e*}(\tilde{g}_{\mathcal{L}_1}^{Del})_{rkj}(\gamma_{\mathcal{L}_1})^*_r}{4M_{\mathcal{L}_{1r}}^2} \right. \\
& \qquad \left. + \frac{\hat{y}_{kj}^{e}(\tilde{g}_{\mathcal{L}_1}^{eDl})^*_{rli}(\gamma_{\mathcal{L}_1})_r}{4M_{\mathcal{L}_{1r}}^2} + \frac{\hat{y}_{kj}^{e}(\tilde{g}_{\mathcal{L}_1}^{Del})^*_{rli}(\gamma_{\mathcal{L}_1})_r}{4M_{\mathcal{L}_{1r}}^2} \right\},
\end{aligned}
$$

$$
(C_{ld})_{ijkl} = -\frac{(y_{\Pi_1})^*_{rjk}(y_{\Pi_1})_{ril}}{2M_{\Pi_{1r}}^2} - \frac{(g_\mathcal{B}^d)_{rkl}(g_\mathcal{B}^l)_{rij}}{M_{\mathcal{B}_r}^2} + \frac{(g_{\mathcal{Q}_5}^{dl})^*_{rki}(g_{\mathcal{Q}_5}^{dl})_{rlj}}{M_{\mathcal{Q}_{5r}}^2},
$$

$$
(C_{lu})_{ijkl} = -\frac{(y_{\Pi_7}^{lu})^*_{rjk}(y_{\Pi_7}^{lu})_{ril}}{2M_{\Pi_{7r}}^2} - \frac{(g_\mathcal{B}^u)_{rkl}(g_\mathcal{B}^l)_{rij}}{M_{\mathcal{B}_r}^2} + \frac{(g_{\mathcal{Q}_1}^{ul})^*_{rki}(g_{\mathcal{Q}_1}^{ul})_{rlj}}{M_{\mathcal{Q}_{1r}}^2},
$$

...

UV/IR dictionary: top-down

Fields	Operators
$\mathcal{S}$	$\mathcal{O}_{\phi 4}$, $\mathcal{O}_{\phi}$, $\mathcal{O}_{\phi\Box}$, $\mathcal{O}_{\phi B}$, $\mathcal{O}_{\phi\tilde{B}}$, $\mathcal{O}_{\phi W}$, $\mathcal{O}_{\phi\tilde{W}}$, $\mathcal{O}_{\phi G}$, $\mathcal{O}_{\phi\tilde{G}}$, $\mathcal{O}_{e\phi}$, $\mathcal{O}_{d\phi}$, $\mathcal{O}_{u\phi}$
$\mathcal{S}_1$	$\mathcal{O}_{ll}$
$\mathcal{S}_2$	$\mathcal{O}_{ee}$
φ	$\mathcal{O}_{le}$, $\mathcal{O}_{qu}^{(1)}$, $\mathcal{O}_{qu}^{(8)}$, $\mathcal{O}_{qd}^{(1)}$, $\mathcal{O}_{qd}^{(8)}$, $\mathcal{O}_{ledq}$, $\mathcal{O}_{quqd}^{(1)}$, $\mathcal{O}_{lequ}^{(1)}$, $\mathcal{O}_{\phi}$, $\mathcal{O}_{e\phi}$, $\mathcal{O}_{d\phi}$, $\mathcal{O}_{u\phi}$
Ξ	$\mathcal{O}_{\phi 4}$, $\mathcal{O}_{\phi}$, $\mathcal{O}_{\phi D}$, $\mathcal{O}_{\phi\Box}$, $\mathcal{O}_{\phi WB}$, $\mathcal{O}_{\phi W\tilde{B}}$, $\mathcal{O}_{e\phi}$, $\mathcal{O}_{d\phi}$, $\mathcal{O}_{u\phi}$
Ξ_1	$\mathcal{O}_{\phi 4}$, $\mathcal{O}_{5}$, $\mathcal{O}_{ll}$, $\mathcal{O}_{\phi}$, $\mathcal{O}_{\phi D}$, $\mathcal{O}_{\phi\Box}$, $\mathcal{O}_{e\phi}$, $\mathcal{O}_{d\phi}$, $\mathcal{O}_{u\phi}$
Θ_1	$\mathcal{O}_{\phi}$
Θ_3	$\mathcal{O}_{\phi}$
ω_1	$\mathcal{O}_{qq}^{(1)}$, $\mathcal{O}_{qq}^{(3)}$, $\mathcal{O}_{lq}^{(1)}$, $\mathcal{O}_{lq}^{(3)}$, $\mathcal{O}_{eu}$, $\mathcal{O}_{ud}^{(1)}$, $\mathcal{O}_{ud}^{(8)}$, $\mathcal{O}_{quqd}^{(1)}$, $\mathcal{O}_{quqd}^{(8)}$, $\mathcal{O}_{lequ}^{(1)}$ $\mathcal{O}_{lequ}^{(3)}$, $\mathcal{O}_{duq}$, $\mathcal{O}_{qqu}$, $\mathcal{O}_{qqq}$, $\mathcal{O}_{duu}$
ω_2	$\mathcal{O}_{dd}$
ω_4	$\mathcal{O}_{uu}$, $\mathcal{O}_{ed}$, $\mathcal{O}_{duu}$
Π_1	$\mathcal{O}_{ld}$
Π_7	$\mathcal{O}_{lu}$, $\mathcal{O}_{qe}$, $\mathcal{O}_{lequ}^{(1)}$, $\mathcal{O}_{lequ}^{(3)}$
ζ	$\mathcal{O}_{qq}^{(1)}$, $\mathcal{O}_{qq}^{(3)}$, $\mathcal{O}_{lq}^{(1)}$, $\mathcal{O}_{lq}^{(3)}$, $\mathcal{O}_{qqq}$
Ω_1	$\mathcal{O}_{qq}^{(1)}$, $\mathcal{O}_{qq}^{(3)}$, $\mathcal{O}_{ud}^{(1)}$, $\mathcal{O}_{ud}^{(8)}$, $\mathcal{O}_{quqd}^{(1)}$, $\mathcal{O}_{quqd}^{(8)}$
Ω_2	$\mathcal{O}_{dd}$
Ω_4	$\mathcal{O}_{uu}$
Υ	$\mathcal{O}_{qq}^{(1)}$, $\mathcal{O}_{qq}^{(3)}$
Φ	$\mathcal{O}_{qu}^{(1)}$, $\mathcal{O}_{qu}^{(8)}$, $\mathcal{O}_{qd}^{(1)}$, $\mathcal{O}_{qd}^{(8)}$, $\mathcal{O}_{quqd}^{(8)}$

+ fermions, vectors

Links and installation

Links and installation

GitHub repository:

https://github.com/jccriado/matchingtools

Available at `PyPI`:

```
pip install matchingtools
```

Documentation:

http://matchingtools.readthedocs.io/en/latest/

arXiv:1710.06445

Future work and conclusions

Conclusions

Future work:

- Include more application-specific tools in `extras`.
- Connection with other software.

With `MatchingTools` we can automatize the process of:

- Tree-level matching.
- Rewritting the effective Lagrangian in a chosen basis.

This lets us reduce the possibility of errors and the time it takes to do the calculations.