runDM v1.0 - Manual

April 27, 2016

1 Overview

The `runDM` code is a tool for calculating the low energy couplings of Dark Matter (DM) to the Standard Model (SM) in Simplified Models with vector mediators. By specifying the mass of the mediator and the couplings of the mediator to SM fields at high energy, the code outputs the couplings at a different energy scale, fully taking into account the mixing of all dimension-6 operators.

At present, the code is written in two languages: *Mathematica* and *Python*. For installation instructions and example code in each language, skip straight to Sec. 3. If you are interested in an implementation in another language, please get in touch and we'll do what we can to add it. Please contact Bradley Kavanagh (bradkav@gmail.com) for any questions, problems, bugs and suggestions.

If you make use of `runDM` in your work, please cite it as:

> F. D'Eramo, B. J. Kavanagh & P. Panci (2016). *runDM* (Version X.X) [Computer software]. Available at https://github.com/bradkav/runDM/ ,

making sure to include the correct version number. Please also cite the associated papers:

> F. D'Eramo & M. Procura, *Connecting Dark Matter UV Complete Models to Direct Detection Rates via Effective Field Theory*, JHEP **1504** (2015) 054 [arXiv:1411.3342 [hep-ph]] ,

> F. D'Eramo & B. J. Kavanagh, *You can hide but you have to run: direct detection with vector mediators*, (2016) [arXiv:1605.XXXXX] .

2 General framework

This section describes the general framework of `runDM`, describing the general usage, inputs and outputs of `runDM`, as well as pseudocode for how to use the most important functions. For implementation-specific information, please see Sec. 3.

Functions in `runDM` accept as input a vector of couplings $\mathbf{c}$. This vector has 16 elements, the coefficients of the dimension-6 DM-SM operators of the form:

$$\mathcal{O}_{\mathrm{DM},\mu}\,\mathcal{O}^{\mu}_{\mathrm{SM}} \tag{1}$$

$$\mathbf{c} = \Big(\begin{array}{ccc:cc|ccc:cc|ccc:cc\|c} c_q^{(1)} & c_u^{(1)} & c_d^{(1)} & c_l^{(1)} & c_e^{(1)} & c_q^{(2)} & c_u^{(2)} & c_d^{(2)} & c_l^{(2)} & c_e^{(2)} & c_q^{(3)} & c_u^{(3)} & c_d^{(3)} & c_l^{(3)} & c_e^{(3)} & c_H \end{array} \Big), \tag{2}$$

where

$$\begin{aligned} \mathcal{O}^{\mu}_{q^{(i)}} &= \overline{q}^{(i)}_L \gamma^\mu q^{(i)}_L\,, & \mathcal{O}^{\mu}_{l^{(i)}} &= \overline{l}^{(i)}_L \gamma^\mu l^{(i)}_L\,, \\ \mathcal{O}^{\mu}_{u^{(i)}} &= \overline{u}^{(i)}_R \gamma^\mu u^{(i)}_R\,, & \mathcal{O}^{\mu}_{e^{(i)}} &= \overline{e}^{(i)}_R \gamma^\mu e^{(i)}_R\,, \\ \mathcal{O}^{\mu}_{d^{(i)}} &= \overline{d}^{(i)}_R \gamma^\mu d^{(i)}_R\,, & \mathcal{O}^{\mu}_{H} &= i H^\dagger \overleftrightarrow{D}_\mu H\,. \end{aligned} \tag{3}$$

$$\mathcal{C} = \Big(\begin{array}{ccccc|ccc\|ccccc|ccc} c_{Vu}^{(1)} & c_{Vd}^{(1)} & c_{Vu}^{(2)} & c_{Vd}^{(2)} & c_{Vd}^{(3)} & c_{Ve}^{(1)} & c_{Ve}^{(2)} & c_{Ve}^{(3)} & c_{Au}^{(1)} & c_{Ad}^{(1)} & c_{Au}^{(2)} & c_{Ad}^{(2)} & c_{Ad}^{(3)} & c_{Ae}^{(1)} & c_{Ae}^{(2)} & c_{Ae}^{(3)} \end{array} \Big). \tag{4}$$

$$\begin{aligned} \mathcal{O}^{\mu}_{Vu^{(i)}} &= \overline{u}^{(i)} \gamma^\mu u^{(i)}\,, & \mathcal{O}^{\mu}_{Au^{(i)}} &= \overline{u}^{(i)} \gamma^\mu \gamma^5 u^{(i)}\,, \\ \mathcal{O}^{\mu}_{Vd^{(i)}} &= \overline{d}^{(i)} \gamma^\mu d^{(i)}\,, & \mathcal{O}^{\mu}_{Ad^{(i)}} &= \overline{d}^{(i)} \gamma^\mu \gamma^5 d^{(i)}\,, \\ \mathcal{O}^{\mu}_{Ve^{(i)}} &= \overline{e}^{(i)} \gamma^\mu e^{(i)}\,, & \mathcal{O}^{\mu}_{Ae^{(i)}} &= \overline{e}^{(i)} \gamma^\mu \gamma^5 e^{(i)}\,. \end{aligned} \tag{5}$$

2.1 Initialisation

The input coupling vector $\mathbf{c}$ must be in the form of an array with 16 elements. To help with initialisation, `runDM` includes the function `initCouplings()`, which returns such an array, filled with zeroes. The user is then free to specify each of the 16 couplings in Eq. 2.

Alternatively, `setBenchmark(benchmarkID)` returns a coupling vector $\mathbf{c}$ corresponding to one of a number of preset benchmarks, specified using the string `benchmarkID`. The available benchmarks are:

- `"Higgs"` - coupling only to the Higgs current operator:

  $c_H = 1$, all other couplings zero.

- `"UniversalVector"` - universal vector coupling to all fermions:

  $c_q^{(i)} = c_u^{(i)} = c_d^{(i)} = c_l^{(i)} = c_e^{(i)} = 1$, $c_H = 0$.

- `"UniversalAxial"` - universal axial-vector coupling to all fermions:

  $-c_q^{(i)} = c_u^{(i)} = c_d^{(i)} = -c_l^{(i)} = c_e^{(i)} = 1$, $c_H = 0$.

- `"QuarksVector"` - vector coupling to all quarks:
$$c_q^{(i)} = c_u^{(i)} = c_d^{(i)} = 1,\ c_l^{(i)} = c_e^{(i)} = c_H = 0.$$

- `"QuarksAxial"` - axial-vector coupling to all quarks:
$$-c_q^{(i)} = c_u^{(i)} = c_d^{(i)} = 1,\ c_l^{(i)} = c_e^{(i)} = c_H = 0.$$

- `"LeptonsVector"` - vector coupling to all leptons:
$$c_l^{(i)} = c_e^{(i)} = 1,\ c_q^{(i)} = c_u^{(i)} = c_d^{(i)} = c_H = 0.$$

- `"LeptonsAxial"` - axial-vector coupling to all leptons:
$$-c_l^{(i)} = c_e^{(i)} = 1,\ c_q^{(i)} = c_u^{(i)} = c_d^{(i)} = c_H = 0.$$

- `"ThirdVector"` - vector coupling to third generation fermions:
$$c_q^{(3)} = c_u^{(3)} = c_d^{(3)} = c_l^{(3)} = c_e^{(3)} = 1, \text{ all remaining couplings zero.}$$

- `"ThirdAxial"` - axial-vector coupling to third generation fermions:
$$-c_q^{(3)} = c_u^{(3)} = c_d^{(3)} = -c_l^{(3)} = c_e^{(3)} = 1, \text{ all remaining couplings zero.}$$

If the value of `benchmarkID` is not recognised, `setBenchmark` simply returns a coupling vector filled with zeroes.

2.2 runCouplings

The function `runCouplings`($\mathbf{c}$, E_1, E_2) accepts as input a vector of couplings $\mathbf{c}$, defined at energy E_1 (in GeV) and returns a vector of couplings evaluated at energy E_2 (in GeV), taking into account the running, matching and mixing between the two energies. If $E_2 > m_Z$, `runCouplings` returns the elements of $\mathbf{c}$, the vector of couplings above the EWSB scale defined in Eq. 2. Alternatively, if $E_2 < m_Z$, `runCouplings` returns the elements of $\mathcal{C}$, the vector of couplings in the EFT below the EWSB scale, defined in Eq. 4.

2.3 DDCouplings

The function `DDCouplings`($\mathbf{c}$, E_1) accepts as input a vector of couplings $\mathbf{c}$, defined at energy E_1 (in GeV). The running is performed down to the nuclear energy scale of 1 GeV and the function returns a vector of the low-energy couplings to light quarks (u, d, s):

$$\mathcal{C}_q = \left(c_V^{(u)}, c_V^{(d)}, c_V^{(s)}, c_A^{(u)}, c_A^{(d)}, c_A^{(s)}\right) . \tag{6}$$

3 Implementations

3.1 Mathematica

In order to use the *mathematica* implementation of `runDM`, simply copy the *mathematica* package file `runDM.m` into the same directory as your notebook and then use `Get`:

```
Get[NotebookDirectory[] <> "runDM.m"];
```

Below is an example code snippet for calculating (and plotting) the low energy couplings to light quarks (relevant in direct detection experiments) at the nuclear energy scale, assuming a mediator of mass m_V which couples through the axial-vector current to all SM quarks:

```
Get[NotebookDirectory[] <> "runDM.m"];

chigh = setBenchmark["QuarksAxial"];

clabels = {"cVu", "cVd", "cVs", "cAu", "cAd", "cAs"};

Table[Plot[{DDCouplings[chigh, 10^lmv][[k]]}, {lmv, 0.0, 6.0},
  PlotTheme -> "Scientific",
  FrameLabel -> {"Log10[mv/GeV]", clabels[[k]]},
  PlotRangePadding -> .1, ImageSize -> 200,
  BaseStyle -> {FontSize -> 16}], {k, 6}]
```

3.2 Python

```
import runDM

#Z mass
mZ = 91.1875

#Nuclear energy scale
E_N = 1

c = sr.InitCouplings()
c[0] = 1.0
#c[-1] = 1.0

mV = np.logspace(0, 5, 100)
```