Getting Started with `DirectDM`

Installation

Basic information about the package can be found at https://directdm.github.io/

Both the `Mathematica` and `Python` versions of `DirectDM` are available on `Github`:

- `Mathematica`: https://github.com/DirectDM/directdm-mma.git
- `Python`: https://github.com/DirectDM/directdm-py.git

For this quick-start guide we will use the `Mathematica` version so please clone the `git` repository as follows:

```
git clone https://github.com/DirectDM/directdm-mma.git <directory>
```

Where `<directory>` is an optional directory name to clone into; if unspecified, it will be `directdm-mma`

Load `DirectDM`

Move into this directory

```
cd directdm-mma
```

Open a new `Mathematica` notebook and save it to the same directory so that it know it lives here, then load the package

```
$DirectDMDirectory=NotebookDirectory[];
AppendTo[$Path,$DirectDMDirectory];
<<DirectDM`
```

Set the Wilson coefficients above EW scale

Assuming a generic Z' model with a Lagrangian,

$$\begin{aligned}\mathcal{L} = &- Z'_\mu \left[g_u \bar{u}_R \gamma^\mu u_R + g_d \bar{d}_R \gamma^\mu d_R \right] - Z'_\mu g_Q \bar{Q}_L \gamma^\mu Q_L \\ &- m_\chi \bar{\chi}\chi - Z'_\mu \bar{\chi}\gamma^\mu \left(g_\chi^V + g_\chi^A \gamma_5 \right) \chi \,.\end{aligned}$$

Matching at tree-level gives the following Wilson coefficients

$$\begin{aligned} C_{2,i}^{(6)} &= -\frac{1}{M_{Z'}^2} g_Q g_\chi^V \,, & C_{6,i}^{(6)} &= -\frac{1}{M_{Z'}^2} g_Q g_\chi^A \,, \\ C_{3,i}^{(6)} &= -\frac{1}{M_{Z'}^2} g_u g_\chi^V \,, & C_{7,i}^{(6)} &= -\frac{1}{M_{Z'}^2} g_u g_\chi^A \\ C_{4,i}^{(6)} &= -\frac{1}{M_{Z'}^2} g_d g_\chi^V \,, & C_{8,i}^{(6)} &= -\frac{1}{M_{Z'}^2} g_d g_\chi^A \,. \end{aligned}$$

Set the DM properties

The DM in this model is a weak-isospin and hypercharge singlet Dirac fermion

```
SetDMType["D"]
SetDMIsospin[0]
SetDMHypercharge[0]
SetDMMass[MX]
```

Set the coefficients

In `DirectDM`, this can be done via

```
(* Vector DM current *)
SetCoeff["6Flavor",Q6[2,1],-gq*gxv/mz^2]
SetCoeff["6Flavor",Q6[3,1],-gu*gxv/mz^2]
SetCoeff["6Flavor",Q6[4,1],-gd*gxv/mz^2]
(* Axial DM current *)
SetCoeff["6Flavor",Q6[6,1],-gq*gxa/mz^2]
SetCoeff["6Flavor",Q6[7,1],-gu*gxa/mz^2]
SetCoeff["6Flavor",Q6[8,1],-gd*gxa/mz^2]
```

where the operator naming `Q6[i,g]` means dimension 6 operator number `i` for generation `g`. This operator basis is defined in

Help for the various "public" functions can be obtained in the usual way

```
?SetCoeff
```

Let's only set the coefficients for the first generation for simplicity.

At any point, if the user so desires, all the coefficients can be reset to zero by invoking `ResetBasis["6Flavor"]`.

Once the coefficients are set, all that is left to do is to compute the non-relativistic EFT coefficients. All the intermediate matching and running is done automatically.

To keep things simple to compare with analytic result, let's turn off the running by issuing the `Running->False` option in `ComputeCoeffs`

```
ComputeCoeffs["6Flavor","NR"]
```

That's it!

Extracting the NREFT can be accomplished in two ways: either by producing a list, or via the `GetCoeff` function to extract them one at a time.

N.B., in the `NR` basis, the *proton* and *neutron* coefficients are provided separately via the `NR_p` and `NR_n` options in the `GetCoeff` or `CoeffsList` functions.

Let's check a couple of them

```
GetCoeff["NR_p",1]//Factor
GetCoeff["NR_n",1]//Factor
```

Compare the output to the analytic result (make sure you `Running->False` in `ComputeCoeffs`)

$$c^p_{NR,1} = -\frac{g^V_\chi}{2M^2_{Z'}}(3g_Q + 2g_u + g_d)$$

The coefficient for the neutron can be obtained from that of the proton by the replacement $p \to n$ and $u \leftrightarrow d$.