Quick guide to running python code for computing relic abundance for the Quintuplet

June 13, 2024

1. Requirements: **python3, scipy, numpy, mpmath, matplotlib**.
2. To compute the abundance for a fixed mass (say 13 TeV) execute the following

   ```
   python boltzmann_eqs_solve.py 13.e3
   ```

   The output is as follows:

   ```
   mdm Omega_eff Omega_eff(gs_const) Omega_network Omega_network (with I→J) time(s)
   13000.0 0.12941000226553 0.12404274053778493 0.11032009304201815 0.11104055336618876 67.53104209899902
   ```

   The average execution time is about 60 seconds to compute all the four cases above.
3. To perform a scan over the masses (default $m_\chi \in [2, 15]$ TeV)

   ```
   python mass_scan.py
   ```

   The output table is saved in file **mx_omega_huelthen_network.dat**, in the same column order as above.
4. Plot results with

   ```
   python plot_omega.py
   ```
5. To extract the yield curve for a fixed mass (say 13 TeV)

   ```
   python yield_compare.py 13.e3
   ```

0.1 Basic structure of the code:

1. Main file **boltzmann_eqs_solve.py**: imports degrees of freedom from folder **dof**. Thermally averaged cross section tables imported from folder **bsf_sigmas**. Summary in table 1
2. Folder **dof**: degrees of freedom as a function of T, including the derivative term.
3. Folder **bsf_sigmas**: All thermally averaged cross sections are computed here.

Function name	purpose	some details
dydzbseff	Effective dY/dz	t=x=z, yi =Y_χ
dydzbseffgsc	Effective dY/dz	(const. gs)
dydz9	Eqs. for dY/dz	with DM and 9 bound states t=x=z, yi[0] =Y_χ, yi[1]-yi[9] = all bound states dydx0 for DM, and dydxi[0]-dydxi[8] eqs for bound states
dydz9ibs	Eqs. for dY/dz	same as above + inter bound-state transitions
solve_ivp	Solves the above differential eqs.	Method='Radau' (an implicit method) y0ia = initial values, teval = integration range , $z = 3 - 10^6$

Table 1: Summary of the main code boltzmann_eqs_solve.py