Augusto Hernandez-Solis, December 2014 @ KTH, Sweden
auhs@kth.se

The python script "DRAG_MOC.py" contains a class that reads self-shielded microscopic cross-sections from an ASCII library produced by the DRAGON transport code. Such script also computes self-shielded macroscopic cross-sections for each of the mixtures previously defined in the DRAGON calculation. Please, remember that from the Version 4 of the DRAGON code, this is able to handle two types of library formats: the WIMS and the DRAGLIB format. Thus, if you are not familiarized with such formats, it is recommended to get acquainted with them. For more info regarding this issue visit:

https://www-nds.iaea.org/wimsd/

www.polymtl.ca/merlin/version5.htm

The main objective of the script is to produce different files after its execution, i.e.:

1) "mat_isot.py" -> It contains material information per mixture assuming isotropic scattering. It comes as a Python nested dictionary and is ready to feed OpenMOC.

2) "mat_isot_hdf5.py" -> If executed as "python mat_isot_hdf5.py", it creates the isotropic material file that is ready to be converted to HDF5 format (just type “python mat_isot_hdf5.py” to create the mat_isot.h5 file).

3) "mat_tranc.py" -> It contains material information per mixture based on the following transport correction:

NTOT_TRANC = NTOT0 - SIGS01 and SCAT_TRANC_MTX = SCAT00 - SCAT01

This means that each term of the P0 matrix is subtracted to its correspondent term of the P1 matrix. The idea behind this is to conserve the n_tot revision done by OpenMOC. Nevertheless, remember that in general, the transport correction is made just by subtracting the SIGS01 (e.g. energy-integrated P1 values) only to the main diagonal of the P0 matrix, i.e.
SCAT00 (sig00_tranc_g -> g = sig00_g -> g - SIGS01g). Still, you have been WARNED and should be aware: The way the transport correction is made in the python script might not be correct and should be avoided or used with caution. However, the idea behind the script is to extract as much information as possible from DRAGON and therefore, due to the fact that both P0 and P1 scattering matrices are available, the user can defined the transport correction as they like. As it is now, at the end of the Keff computation, the only valid comparison between DRAGON and OpenMOC should be done and trusted only for the isotropic case. You may judge this at your best.
Please, remember that the WIMS library ONLY define the P1 scattering matrix for moderator nuclides (e.g. H1, O16, etc.).

4) "mat_tranc_hdf5.py"

5) "XS.m" - > File for easy post-processing and plotting in Matlab/Octave. Here the different XS for the different mixtures can be read and manipulated if desired.

In order to run the script, an input file that imports the DRAG_MOC class needs to be executed. For example, the module can be called and executed as follows: “python example_DRAG_MOC.py”.

Parameters to be defined by the user in the python input file

“GROUPS” -> No. of energy groups contained in the library given by DRAGON

“mixtures” -> Names defined by the user for the different atomic mixtures. These can be any given name the user likes.

“mix_drag” -> Names of the mixtures in DRAGON format. For instance, if in the DRAGON input deck "Mix 1" is defined under the LIB module, its string in this list will be '0001'

“isotopes” -> Names of the different isotopes as exactly defined in the DRAGON input deck.

“atom_dens” -> Atomic densities. The position of each string in this list should match the isotopes string locations.

“XS_*.py” -> ALL the names of the different files that the module will create.

CASES directory

Some test cases have been computed, and the benchmark between DRAGON and OpenMOC has been made. Different energy groups, library formats and nuclear data libraries have been tested. They aim to help the user to get familiarized very quickly with the tool. Therefore, each test case directory contains the following files:

1) *.c2m → This is the DRAGON input deck. The Keff comparison between DRAGON and OpenMOC can be found at the top this file.

2) *_DRAG_MOC.py → This is the input file that is needed to run the DRAG_MOC.py module.

3) library_ss → ASCII library file that is read by DRAG_MOC.py. (This is the only file that is included ONLY in certain test cases due to size restrictions).

4) *.result → This is the output file of the DRAGON run for that particular case

5) All the output files created by DRAG_MOC.py