dchain_tools.py

Hunter Ratliff Last updated: 28 July 2020

This document details the `dchain_tools.py` Python 3 library of functions. Note that this module is simply a collection of functions I have developed over time to speed up my usage of PHITS and DCHAIN-PHITS; these functions are not officially supported by the PHITS development team. They were developed to serve my own needs, and I am just publicly sharing them because others may also find utility in them. I may more professionally repackage and redistribute these functions in the future in a more standard way. If I write any other notably useful functions related to DCHAIN in the future, I will add them to this module.

While `dchain_tools.py` contains a variety of functions, its primary intended usage is with the `process_dchain_simulation_output()` function which parses all of the relevant DCHAIN output files for a simulation and returns a single Python dictionary object (which can also be accessed in an "attribute-style" thanks to the munch library) containing all of the relevant output information in a much more accessible form.

1 Installation

The DCHAIN tools module is a single Python file called `dchain_tools.py` and is just a collection of functions. Place the `dchain_tools.py` file in a desired location and add the path to that folder to your PYTHONPATH system environmental variable. Then, the functions can be accessed just like with any other Python library by placing the following code at the top of any Python script:

```
from dchain_tools import *
```

And that's it! There are a number of libraries which importing `dchain_tools` will also automatically import:

```
import numpy as np
import os
import sys
import matplotlib.pyplot as plt
import time
import re
import bisect
import unicodedata as ud
from munch import *
```

These libraries will likely be installed already in any Python environment, especially if coming from a distribution including libraries such as Anaconda, unless it is brand new.

2 Available functions

All of these functions are reasonably well-documented within the `dchain_tools.py` script itself, so this document will not seek to rehash everything already written in the comment block at the start of each function. Instead, it serves to just highlight what functions are available. How the main `process_dchain_simulation_output()` function is used is covered in the following section.

2.1 Nuclide/element formatting functions

- `nuclide_plain_str_ZZZAAAM` - convert a plaintext string for a nuclide (written in any of a large variety of sensible formats) to an integer ZZZAAAM nuclide identifying number (ZZZAAAM = Z·10^4 + A·10 + M)
- `Dname_to_ZAM` - converts a DCHAIN-formatted nuclide name to a ZZZAAAM
- `ZAM_to_Dname` - converts a ZZZAAAM to a DCHAIN-formatted nuclide name
- `Dname_to_Latex` - converts a DCHAIN-formatted nuclide name to LaTeX format
- `nuclide_plain_str_to_latex_str` - convert a plaintext string for a nuclide to a LaTeX-formatted raw string
- `Element_Z_to_Sym` - returns elemental symbol provided the atomic number Z
- `Element_Sym_to_Z` - returns an atomic number Z provided the elemental symbol
- `Element_ZorSym_to_name` - returns a string of the name of an element provided its atomic number Z or symbol
- `Element_ZorSym_to_mass` - returns the average atomic mass of an element provided its atomic number Z or symbol
- `nuclide_to_Latex_form` - form a LaTeX-formatted string of a nuclide provided its Z, A, and metastable information

2.2 Relating to DCHAIN data libraries

- `rxn_to_dchain_str` - converts a reaction to the format used by the neutron reaction cross section libraries in DCHAIN
- `ZZZAAAM_to_dchain_xs_lib_str` - converts a ZZZAAAM number to the 7-character nuclide string used by the neutron reaction cross section libraries
- `ECCO1968_Ebins` - returns the n highest energy bins of ECCO 1968-group structure
- `retrieve_rxn_xs_from_lib` - returns cross section for a reaction from a provided neutron reaction cross section library file
- `calc_one_group_nrxn_xs_dchain` - provided a neutron flux, reaction, and library file, determine the single-group neutron reaction cross section

2.3 DCHAIN output file parsing

- `parse_DCHAIN_act_file` - parser for the *.act file from DCHAIN
- `generate_nuclide_time_profiles` - processes output from `parse_DCHAIN_act_file` to turn nuclide output into a more usable format
- `parse_DCHAIN_act_file_legacy` - legacy version of `parse_DCHAIN_act_file` from before statistical uncertainty propagation implementation
- `generate_nuclide_time_profiles_legacy` - legacy version from before statistical uncertainty propagation implementation of the `generate_nuclide_time_profiles` function
- `parse_DCS_file_from_DCHAIN` - parser for the *.dcs file from DCHAIN
- `parse_dtrk_file` - parser for the *.dtrk file from PHITS meant for DCHAIN
- `parse_dyld_files` - parser for the *.dyld files from PHITS meant for DCHAIN
- `process_dchain_simulation_output` - the main master function for parsing **ALL** output from DCHAIN
- `plot_top10_nuclides` - generates a nice visualization on the ranking of nuclides

3 Primary function usage

As mentioned earlier, `process_dchain_simulation_output()` is the primary function in this library; most of the other functions serve to just be called by this one (but still work fine standalone). It returns one output, referred to just as `dchain_output`, which will be discussed in detail later. The five inputs (2 mandatory, 3 optional) are detailed below.

Inputs for `process_dchain_simulation_output()`	
`simulation_folder_path`	text string of path to folder containing simulation output (should end with / or \)
`simulation_basename`	common string of the DCHAIN simulations; output files are named `simulation_basename.*`. This string is also what is entered in the `file` variable in the [T-Dchain] tally, without any extension included.
`dtrk_filepath`	(*optional*) file path to *.dtrk file, only necessary if it has a different basename and there are multiple *.dtrk files in the simulation folder
`dyld_filepath`	(*optional*) file path to *.dyld file(s), only necessary if it has a different basename and there are multiple *.dyld files in the simulation folder
`process_DCS_file`	(*optional*) Boolean variable specifying whether the DCS file should be processed too. As this file can be quite large, for performance reasons its processing is disabled by default. (default=`False`)

The process_dchain_simulation_output() function primarily serves to parse the *.act main output file of DCHAIN but can also read the *.dcs file and will automatically attempt to read the PHITS-generated *.dtrk and *.dyld files. (The PHITS-generated files are mostly useful for diagnostic and secondary purposes.)

To make this usage more clear, a brief example will now be covered. In this example, the initial PHITS simulation input file was located in a folder whose path is C://path/to/simulation/folder/, and the [T-Dchain] tally had file = example.in set as the name of the DCHAIN input file to be automatically generated. After completing the PHITS simulation (resulting in example.in, example.dyld, example.dtrk, and dch_link.dat being written), the DCHAIN input file was ran through DCHAIN without moving or renaming any files (resulting in a number of files named example.* being produced). After this, the below Python code is used to extract the results from the output files.

```
from dchain_tools import *

folder_path = r'C:\path\to\simulation\folder\\'
simulation_name = 'example'  # note: no extension included

dchain_output = process_dchain_simulation_output(folder_path,
    simulation_name,process_DCS_file=True)

# print list of all nuclides (formatted as plain text strings) found in
    the first region across all times
print(dchain_output['nuclides']['names'][0]) # dictionary-style access

# print the total activity and its absolute error in the first region
    and first time step
print(dchain_output.nuclides.total.activity.value[0][0], dchain_output.
    nuclides.total.activity.error[0][0]) # attribute-style access
```

The dchain_output variable is a Python dictionary containing all of the relevant values outputted by DCHAIN. It has been "munchified" by the munch library too, meaning it can be accessed both as a normal Python dictionary or in the attribute style associated with classes. Generally, all entries are either single values, lists of values, or lists of NumPy arrays. For most outputs, this is of the form of a list containing elements for each region where each element is either a single string/value or a NumPy array/list further dividing results by time, nuclide species, etc.

This dictionary is structured as outlined below. While the parts relevant to the *.act file are always written, the sections relevant to the other files are only written if the files are found / their parsing is requested (in the case of the *.dcs file). A more convenient version of this information for general reference or printing can be found in the accompanying dchain_tools_output_structure.pdf file.

```

# Notation for output array dimensions
#   R  regions
#   T  time steps
#   N  max number of nuclides found in a single region
#   E  number of gamma energy bins
# le10 for top 10 lists, a number <= 10

```

```
dchain_output = {
'time':{                            #  ~  Time information
    'from_start_sec'                #  [T] list of times from start time [sec]
    'from_EOB_sec'                  #  [T] list of times from end of final bombardment [sec]
    'of_EOB_sec'                    #      scalar time marking end of final bombardment [sec]
    }

'region':{                          #  ~  Information which only varies with region
    'numbers'                       #  [R] region numbers
    'number'                        #  [R] region numbers
    'irradiation_time_sec'          #  [R] irradiation time per region
    'volume'                        #  [R] volume in [cc] per region
    'neutron_flux'                  #  [R] neutron flux in [n/cm^2/s] per region
    'beam_power_MW'                 #  [R] beam power in [MW] per region
    'beam_energy_GeV'               #  [R] beam energy in [GeV] per region
    'beam_current_mA'               #  [R] beam current in [mA] per region
    }

'nuclides':{                        #  ~  Main nuclide results from *.act file
    'names'                         #  [R][N] names of nuclides produced in each region
    'TeX_names'                     #  [R][N] LaTeX-formatted names of nuclides produced
    'ZZZAAAM'                       #  [R][N] ZZZAAAM values (=10000*Z+10*A+M) of nuclides
                                    #         (ground state m=0, metastable m=1,2,etc.)
    'half_life'                     #  [R][N] half lives of nuclides produced [sec]
    'inventory':{'value'            #  [R][T,N] atoms [#/cc]
                 'error'}           #  [R][T,N] atoms [#/cc]
    'activity':{'value'             #  [R][T,N] activity [Bq/cc]
                'error'}            #  [R][T,N] activity [Bq/cc]
    'dose_rate':{'value'            #  [R][T,N] dose-rate [uSv/h*m^2]
                 'error'}           #  [R][T,N] dose-rate [uSv/h*m^2]
    'decay_heat':{
        'total':{'value'            #  [R][T,N] total decay heat [W/cc]
                 'error'}           #  [R][T,N] total decay heat [W/cc]
        'beta':{'value'             #  [R][T,N] beta decay heat [W/cc]
                'error'}            #  [R][T,N] beta decay heat [W/cc]
        'gamma':{'value'            #  [R][T,N] gamma decay heat [W/cc]
                 'error'}           #  [R][T,N] gamma decay heat [W/cc]
        'alpha':{'value'            #  [R][T,N] alpha decay heat [W/cc]
                 'error'}           #  [R][T,N] alpha decay heat [W/cc]
        }
    'column_headers'                # Length 7 list of the *.act columns' descriptions
    'total':{                       #  ~  Total values summed over all nuclides
        'activity':{'value'         #  [R][T] total activity [Bq/cc]
                    'error'}        #  [R][T] total activity [Bq/cc]
        'decay_heat':{'value'       #  [R][T] total decay heat [W/cc]
                      'error'}      #  [R][T] total decay heat [W/cc]
        'beta_heat':{'value'        #  [R][T] total beta decay heat [W/cc]
                     'error'}       #  [R][T] total beta decay heat [W/cc]
        'gamma_heat':{'value'       #  [R][T] total gamma decay heat [W/cc]
                      'error'}      #  [R][T] total gamma decay heat [W/cc]
        'alpha_heat':{'value'       #  [R][T] total alpha decay heat [W/cc]
                      'error'}      #  [R][T] total alpha decay heat [W/cc]
        'activated_atoms':{'value'  #  [R][T] total activated atoms [#/cc]
                           'error'} #  [R][T] total activated atoms [#/cc]
        'gamma_dose_rate':{'value'  #  [R][T] total gamma dose rate [uSV/h*m^2]
                           'error'} #  [R][T] total gamma dose rate [uSV/h*m^2]
        }
    }


'gamma':{                           #  ~  Gamma spectra and totals
    'spectra':{
        'group_number'              #  [R][T,E] group number
        'E_lower'                   #  [R][T,E] bin energy lower-bound [MeV]
        'E_upper'                   #  [R][T,E] bin energy upper-bound [MeV]
        'flux':{'value'             #  [R][T,E] flux [#/s/cc]
                'error'}            #  [R][T,E] flux [#/s/cc]
        'energy_flux':{'value'      #  [R][T,E] energy flux [MeV/s/cc]
                       'error'}     #  [R][T,E] energy flux [MeV/s/cc]
        }
    'total_flux':{'value'           #  [R][T] total gamma flux [#/s/cc]
                  'error'}          #  [R][T] total gamma flux [#/s/cc]
```

```
    'total_energy_flux':{'value'    # [R][T] total gamma energy flux [MeV/s/cc]
                         'error'}   # [R][T] total gamma energy flux [MeV/s/cc]
    'annihilation_flux':{'value'    # [R][T] annihilation gamma flux [#/s/cc]
                         'error'}   # [R][T] annihilation gamma flux [#/s/cc]
    'current_underflow':{'value'    # [R][T] gamma current underflow [#/s]
                         'error'}   # no error reported
    'current_overflow':{'value'     # [R][T] gamma current overflow [#/s]
                        'error'}    # no error reported
    }

'top10':{                           #  ~  Top 10 lists from *.act file
    'activity':{
        'rank'                      # [R][T,le10] rank
        'nuclide'                   # [R][T,le10] nuclide name
        'value'                     # [R][T,le10] activity [Bq/cc]
        'error'                     # [R][T,le10] activity [Bq/cc]
        'percent'                   # [R][T,le10] percent of total activity
        }
    'decay_heat':{
        'rank'                      # [R][T,le10] rank
        'nuclide'                   # [R][T,le10] nuclide name
        'value'                     # [R][T,le10] decay heat [W/cc]
        'error'                     # [R][T,le10] decay heat [W/cc]
        'percent'                   # [R][T,le10] percent of total decay heat
        }
    'gamma_dose':{
        'rank':                     # [R][T,le10] rank
        'nuclide'                   # [R][T,le10] nuclide name
        'value'                     # [R][T,le10] dose-rate [uSv/h*m^2]
        'error'                     # [R][T,le10] dose-rate [uSv/h*m^2]
        'percent'                   # [R][T,le10] percent of total gamma dose rate
        }
    }
'number_of':{                       #  ~  Maximum values of R, T, N, and E
    'regions'                       #  R  = total number of regions
    'time_steps'                    #  T  = total number of time steps
    'max_nuclides_in_any_region'    #  N  = maximum unique nuclides found in any region
    'gamma_energy_bins'             #  E  = number of gamma energy bins (default=42)
    }
}

if process_dtrk_file: dchain_output.update({
'neutron':{                         #  ~  Neutron spectra and totals
    'spectra':{                     #  -  Actual values used in DCHAIN
        'E_lower'                   # [R][E] bin energy lower-bound [MeV]
        'E_upper'                   # [R][E] bin energy upper-bound [MeV]
        'flux':{'value'             # [R][E] neutron flux [#/s/cm^2]
                'error'}            # [R][E] neutron flux [#/s/cm^2]
        }
    'total_flux':{'value'           # [R]    total neutron flux [#/s/cm^2]
                  'error'}          # [R]    total neutron flux [#/s/cm^2]
    'unit_spectra':{                #  -  Flux per unit source particle (raw *.dtrk output)
        'E_lower'                   # [R][E] bin energy lower-bound [MeV]
        'E_upper'                   # [R][E] bin energy upper-bound [MeV]
        'flux':{'value'             # [R][E] neutron flux [#/s/cm^2/s.p.]
                'error'}            # [R][E] neutron flux [#/s/cm^2/s.p.]
}})

if process_dyld_files:
    dchain_output.update({
    'yields':{                      #  ~  Yield spectra
        'all_names'                 # [N] names of all nuclides produced
        'names'                     # [R][N] names of nuclides produced in each region
        'TeX_names'                 # [R][N] LaTeX-formatted names of nuclides produced
        'ZZZAAAM'                   # [R][N] ZZZAAAM values (=10000Z+10A+M) of nuclides
                                    #        (ground state m=0, metastable m=1,2,etc.)
        'rate':{                    #  -  Actual values used in DCHAIN (at 100% beam power)
               'value'              # [R][E] nuclide yield rate [#/s/cm^3]
               'error'              # [R][E] nuclide yield rate [#/s/cm^3]
               }
        'unit_rate':{               #  -  Yields per unit source particle
               'value'              # [R][E] nuclide yield rate [#/s.p.]
               'error'              # [R][E] nuclide yield rate [#/s.p.]
```

```
}}})

if process_DCS_file: # add extra information
    # Notation for output array dimensions
    #   R  (n_reg)    regions
    #   Td (ntsteps)  time steps in DCS file (usually different from that of *.act file!)
    #   Nd (nnuc_max) max number of nuclides (this index differs from the *.act N index)
    #   C  (chni_max) maximum index of relevant chains
    #   L  (chln_max) maximum number of links per chain

dchain_output.update({
'DCS':{
    'time':{
        'from_start_sec'            # [Td] list of times from start time [sec]
        'from_EOB_sec'              # [Td] list of times from end of final bombardment [sec]
        'of_EOB_sec'                #      scalar time marking end of final bombardment [sec]
        }

    'number_of':{                   #  ~  Maximum values of R, Td, Nd, C, and L
        'regions'                   #  R  = total number of regions
        'time_steps'                #  Td = total number of time steps
        'max_nuclides'              #  Nd = max number of end nuclides in any time step
        'max_number_of_chains'      #  C  = highest index of a relevant chain found
        'max_chain_length'          #  L  = max number of links (nuclides) in any chain
        }

    'end_nuclide':{                 #  ~  Informtaion on nuclides at the end of each chain
        'names'                     # [R][Td,Nd] nuclide names
        'inventory':{
            'N_previous'            # [R][Td,Nd,C] inventory in previous time step [atoms/cc]
            'N_now'                 # [R][Td,Nd,C] inventory in current time step [atoms/cc]
            'dN'                    # [R][Td,Nd,C] change in inventory of end nuclide from
                                    #              previous to current time step [atoms/cc]
            }
        'activity':{
            'A_previous'            # [R][Td,Nd,C] activity in previous time step [Bq/cc]
            'A_now'                 # [R][Td,Nd,C] activity in the current time step [Bq/cc]
            'dA'                    # [R][Td,Nd,C] change in activity of end nuclide from
                                    #              previous to current time step [Bq/cc]
            }
        }

    'chains':{                      #  ~  Chains, individual links, and their contributions
        'indices_of_printed_chains' # [R][Td,Nd]     the chain indices which were printed
        'length'                    # [R][Td,Nd,C]   length of listed chain
        'link_nuclides'             # [R][Td,Nd,C,L] strings of the nuclides in each chain
        'link_decay_modes'          # [R][Td,Nd,C,L] strings of the decay modes each link
                                    #                undergoes to produce the next link
        'link_dN':{                 # (only generated if values in file, 'None' otherwise)
            'beam'                  # [R][Td,Nd,C,L] beam contribution to dN from each link
            'decay_nrxn'            # [R][Td,Nd,C,L] decay + neutron rxn contribution to dN
            'total'                 # [R][Td,Nd,C,L] total contribution to dN from each link
            }
        }

    'relevant_nuclides':{           #  ~  A vs t profiles of nuclides over relevancy threshold
        'names'                     # [R]        list of relevant nuclides per region
        'times'                     # [R][Td,Nd] time [s]
        'inventory'                 # [R][Td,Nd] inventory [atm/cc]
        'activity'                  # [R][Td,Nd] activity [Bq/cc]
        }
    }
})
```

dchain_tools_output_structure