PHITS_tools.py

Hunter Ratliff

Last updated: 22 October 2024

This document details the `PHITS_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; 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.

While `PHITS_tools.py` contains a variety of functions, its primary intended usage is with the `parse_tally_output_file()` function, which is a general parser for standard PHITS tally output files, and the `parse_tally_dump_file()` function, which is a general parser for "dump" output files when using the `dump` parameter in [T-Cross], [T-Time], and [T-Track] tallies.

These tallies are able to output parsed results both in a "dense" format (a 10-dimensional NumPy array for the standard output parser and a list of namedtuples for the dump file parser) and as Pandas DataFrames. The standard output parser also provides a dictionary of metadata relevant to the tally.

1 Installation

The PHITS Tools module is a single Python file called `PHITS_tools.py` and is just a collection of functions. Place the `PHITS_tools.py` file in a desired location and add the path to that folder to your PYTHONPATH system environmental variable if not already included in it. 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:

```python
from PHITS_tools import *
```

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

```python
import sys
import os
import numpy as np
from munch import *
from pathlib import Path
```

And depending on which functions are used, the following imports may also be made:

```python
from collections import namedtuple
from typing import NamedTuple
from scipy.io import FortranFile
import pandas as pd
import pickle
import dill
```

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

Detailed documentation on all of the functions can be found online at lindt8.github.io/PHITS-Tools/ and within the `PHITS_tools.py` module itself, so this document will not seek to rehash everything already written there. Instead, it serves to just highlight what functions are available. How the main `parse_tally_output_file()` function is used is covered in the following section.

2.1 Main PHITS Output Parsing Functions

- `parse_tally_output_file` - general parser for standard output files for all PHITS tallies
- `parse_tally_dump_file` - parser for dump files from `dump` flag in PHITS [T-Cross], [T-Time], and [T-Track] tallies
- `parse_all_tally_output_in_dir` - NOT YET IMPLEMENTED

2.2 General Purpose Functions

- `is_number` - returns Boolean denoting whether provided string is that of a number

2.3 Subfunctions for PHITS output parsing

(These are meant as dependencies more so than for standalone usage.)

- `split_into_header_and_content` - initial reading of PHITS tally output, dividing it into header and "content" sections
- `parse_tally_header` - extract metadata from tally output header section
- `parse_tally_content` - extract tally results/values from tally content section
- `initialize_tally_array` - initialize NumPy array for storing tally results
- `calculate_tally_absolute_errors` - calculate absolute uncertainties from read values and relative errors
- `build_tally_Pandas_dataframe` - make Pandas dataframe from the main results NumPy array and the metadata
- `extract_data_from_header_line` - extract metadata key/value pairs from tally output header lines
- `data_row_to_num_list` - extract numeric values from a line in the tally content section
- `parse_group_string` - split a string containing "groups" (e.g., regions) into a list of them
- `split_str_of_equalities` - split a string containing equalities (e.g., `reg = 100`) into a list of them

3 Primary function usage

As mentioned earlier, `parse_tally_output_file()` 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 `tally_output`, which will be discussed in detail later. Nominally, its only input is the file path to the PHITS tally output file to be processed; its additional inputs are all optional Booleans (set `=True` by default) which control whether the Pandas DataFrame is constructed from the "compact" NumPy array (`make_PandasDF`), whether absolute uncertainties are to be calculated from the tally values and relative errors (`calculate_absolute_errors`), and whether the function's output dictionary is also saved to a pickle file for easy future access (`save_output_pickle`).

This function seeks to be a general/universal PHITS tally output parser. Broadly speaking, all PHITS output is either formatted ASCII files intended for processing by the ANGEL code for generation of visual output (*.eps files), which this function processes, or dump files (ASCII or binary), which the `parse_tally_dump_file` function parses. For special [T-Dchain] outputs and the output from the DCHAIN code, the separate Python module DCHAIN tools (github.com/Lindt8/DCHAIN-Tools) is available.

To be able to process output from *any* PHITS tally standard output formatted for ANGEL, one must understand all of the different variables, dimensions, and axes any tally may contain and account for all of the different formatting styles possible. On the first point, the general idea was to have some N-dimensional array which could be shaped to contain the output from any tally; the important part is determining what that N should be and what should be assigned to its indices.

At a basic level, every tally has some geometric component, and most also have an additional axis (or more) of output. Furthermore, most tallies can output results for different particles or sets of particles. A few tallies, like [T-Deposit2] and [T-Yield] (and hidden in [T-Track] for DCHAIN) have extra possible output axes as well, but no more than one of these would ever be active at a time. Thus, in all, the **ten** axes/dimensions below are able to form an array capable of holding the results from any PHITS tally.

index N	index variable	axis/dimension meaning
0	`ir`	Geometry mesh: `reg` / `x` / `r` / `tet`
1	`iy`	Geometry mesh: `1` / `y` / `1` ([T-Cross] `ir surf`)
2	`iz`	`1` / `z` / `z` ([T-Cross] `iz surf` if (`mesh=xyz`) OR `iz` if (`mesh=r-z` AND `enclos=1`))
3	`ie`	Energy mesh: `eng` ([T-Deposit2] `eng1`)
4	`it`	Time mesh
5	`ia`	Angle mesh
6	`il`	LET mesh
7	`ip`	Particle type (`part =`)
8	`ic`	Special: [T-Cross] `iz surf` (if `mesh=r-z` AND `enclos=0`); [T-Deposit2] `eng2`; [T-Yield] `mass`, `charge`, `chart`, `dchain`
9	`ierr`	`=0/1/2` for Value / relative error / absolute error

The output dictionary `tally_output` from this function has the following three keys: `'tally_data'`, `'tally_metadata'`, and `'tally_dataframe'`. `tally_output['tally_data']` is the 10-dimensional NumPy array formatted as shown above and can be accessed as `tally_output['tally_data'][ ir, iy, iz, ie, it, ia, il, ip, ic, ierr ]`.