LA-UR-21-26562
Approved for public release; distribution is unlimited.

Title: Particle Track Output (PTRAC) Improvements, Parallelism, and Post-Processing

Author(s):
Bolding, Simon R.
Kulesza, Joel A.
Marcath, Matthew James
Rising, Michael Evan

Intended for: MCNP User Symposium 2021, 2021-07-12/2021-07-16 (Los Alamos, New Mexico, United States)

Issued: 2021-07-12 (Draft)

Particle Track Output (PTRAC) Improvements, Parallelism, and Post-Processing

Simon Bolding, Joel Kulesza, Matthew Marcath, and Michael Rising

13 July 2021

LAUR -TODO

Outline

- PTRAC overview
- What is new in MCNP6.3?
  - Parallel PTRAC
  - HDF5 format
- Post-processing with MCNPTools and Python
- Future changes

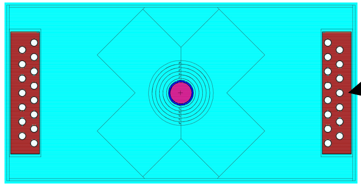
Outline

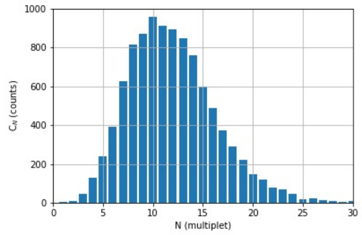
- PTRAC overview
- What is new in MCNP6.3?
  - Parallel PTRAC
  - HDF5 format
- Post-processing with MCNPTools and Python
- Future changes

Particle track output (PTRAC) is used for advanced post-processing of histories on an event-by-event basis

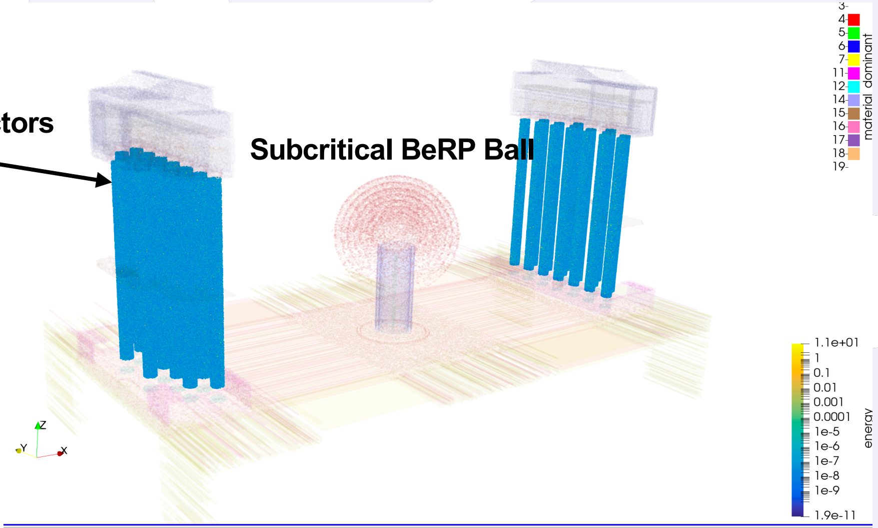
- PTRAC is often used for coincident and time-dependent detector analysis
  - The output data is fed into other post-processing software
- Examples uses include
  - Advanced detector response simulations with DRiFT [1]
  - Subcritical multiplicity counting

Multiplicity Detectors





Subcritical BeRP Ball



Los Alamos NATIONAL LABORATORY

PTRAC output is a list of events as they occurred in the code, for each history

- Output includes source, termination, collision, bank, and surface events
  - With secondary particles, not ordered with physical time (LIFO)
  - Reconstructing branching of histories is typically possible, but onerous
- Can filter which events are written based on history and event data
  - Limits the size of output files and memory usage
- Example input card

```
PTRAC EVENT=SRC,COL,TER CELL=1 FILTER=ERG,1,14 FILE=HDF5 FLUSHNPS=1E5
```

  - Writes events with particle energy in [1,14] MeV, for histories that traversed cell 1

For MCNP6.3 an HDF5 format is available that can be generated via parallel execution

- Simpler output structure makes the feature more accessible to users
  - Reduces processing errors of legacy formats
  - More flexible, so it can be extended in the future
- **PTRAC simulations with HDF5 can be executed in parallel, removing a computational bottleneck**
  - Even in serial, HDF5 PRAC is up to 10x faster for large problems
- Improved the interface for some features
  - e.g., cell and surface event filters now read user IDs
- MCNP6.3 includes several PTRAC bug fixes detailed in release notes
  - Two infrequently used features have also been deprecated

Outline

- PTRAC overview
- What is new in MCNP6.3?
  - Parallel PTRAC
  - HDF5 format
- Post-processing with MCNPTools and Python
- Future changes

MCNP6.3 with PTRAC can be executed in parallel with detailed guidance given in the manual

- HDF5 PTRAC files can be produced with MPI, threads, or both
  - MPI-parallel PTRAC **requires** an MPI enabled installation of HDF5
  - Task (OpenMP) parallel PTRAC is available with any HDF5 build
- Each process buffers data into memory over multiple histories periodically writing to disk
- The memory usage can be **very large**
  - A `std::bad_alloc` error will appear if memory is exhausted
    - Inefficient memory swapping may occur instead
  - This is mitigated with the FLUSHNPS option as detailed in next slide

User must specify FLUSHNPS to control how much PTRAC data is buffered in memory between writes

- FLUSHNPS is the maximum number of histories between file writes
  - User values of FLUSHNPS vary greatly with simulation and computer size
- An estimate of maximum memory usage is printed by MCNP6
  - Only PTRAC usage, so must be balanced with problem size
  - Writes may occur more frequently from other rendezvous
- As a rule of thumb (for current hardware) use `FLUSHNPS=100,000` and check the memory usage is < 2 GB, per computational resource
  - Higher memory usage will not necessarily improve efficiency
  - **When in doubt, write more often than necessary**

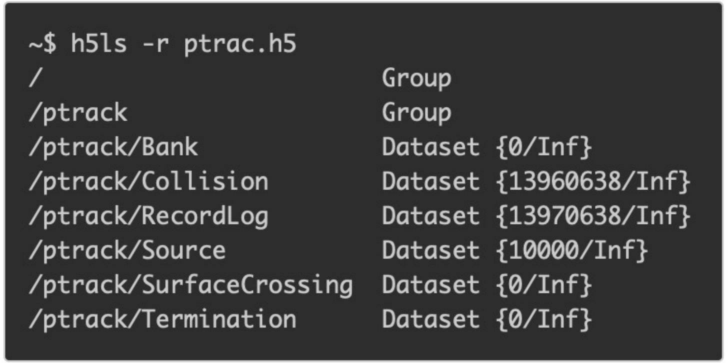
There are a couple other things to remember specific to parallel PTRAC execution

- MPI parallel HDF5 will be **slow or hang on most Network File Systems** in ways that MCNP6 cannot detect
  - **Always** try a quick initial run to verify that a file is written correctly
  - Can use task-based parallelism on any file system

- For *task*-based parallelism, order of NPS in file is non-deterministic
  - A side-effect of how MCNP does shared memory parallelism
  - Each history is independent, so this **does not** affect results
    - All events for each history are always ordered
  - If you want to exactly reproduce the order of an equivalent serial simulation, the record log should be *stably* sorted by NPS

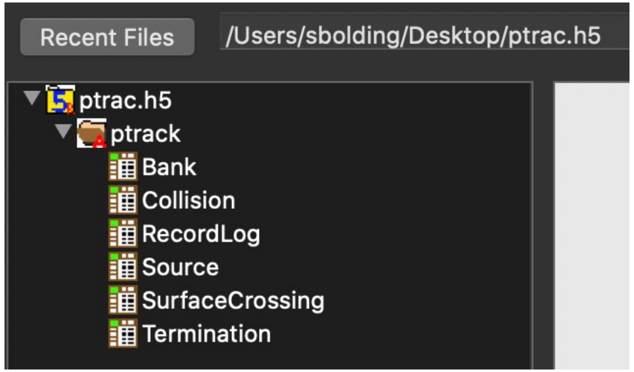
The output file ptrac.h5 is binary, but there are many standardized toolsets for inspecting it

– **h5ls** and **h5dump** for terminal usage



```
~$ h5ls -r ptrac.h5
/                        Group
/ptrack                  Group
/ptrack/Bank             Dataset {0/Inf}
/ptrack/Collision        Dataset {13960638/Inf}
/ptrack/RecordLog        Dataset {13970638/Inf}
/ptrack/Source           Dataset {10000/Inf}
/ptrack/SurfaceCrossing  Dataset {0/Inf}
/ptrack/Termination      Dataset {0/Inf}
```

– **HDFView** for graphical exploration



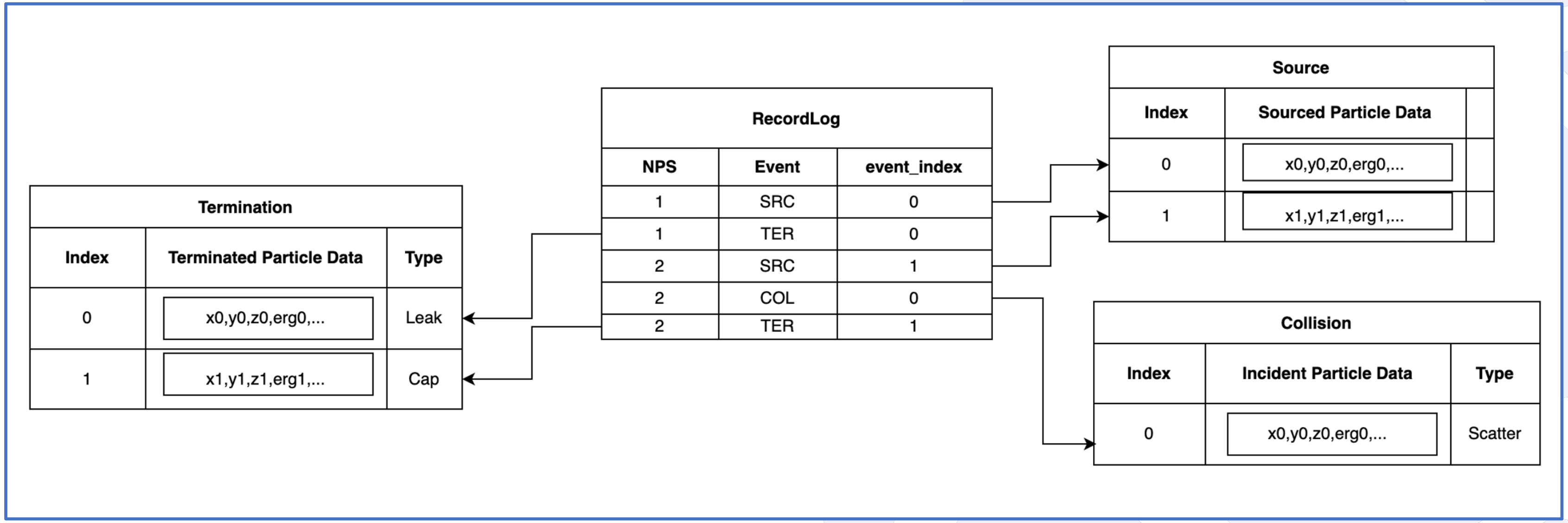
```
▼ ptrac.h5
  ▼ ptrack
      Bank
      Collision
      RecordLog
      Source
      SurfaceCrossing
      Termination
```

• Each group (e.g., /ptrack/) is like a folder on your computer and each dataset is just an array of data that can be post-processed

The new format has a flattened output structure compared to the convoluted ASCII data

- Each dataset array lists all events for a particular type of event
  - Have direct access to entire array for each event type
- The RecordLog dataset provides the event-by-event ordering of the entries in the event arrays, ordered by the history identifier NPS
  - Similar to foreign keys of relational databases
- *Each* entry in a dataset is a compound data type
  - Contains all particle data, e.g., x, y, z, energy, etc.
  - Contains event specific data, e.g., collision type

The RecordLog lists the order of the events present in the other arrays



RecordLog

NPS	Event	event_index
1	SRC	0
1	TER	0
2	SRC	1
2	COL	0
2	TER	1

Termination

Index	Terminated Particle Data	Type
0	x0,y0,z0,erg0,...	Leak
1	x1,y1,z1,erg1,...	Cap

Source

Index	Sourced Particle Data
0	x0,y0,z0,erg0,...
1	x1,y1,z1,erg1,...

Collision

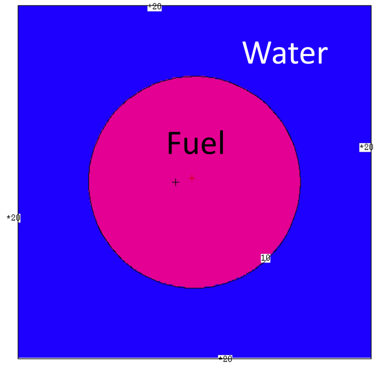
Index	Incident Particle Data	Type
0	x0,y0,z0,erg0,...	Scatter

Outline

- PTRAC overview
- What is new in MCNP6.3?
  - Parallel PTRAC
  - HDF5 format
- Post-processing with MCNPTools and Python
- Future changes

Example problem: *simplified* subcritical pin cell lattice, with a spontaneous fission source in fuel

- 12% enriched uranium surrounded by water
  - Reflective boundaries
  - ^{238}U Watt spectrum source, distributed over fuel volume
  - K-effective for geometry is 0.96
- **Goal:** using PTRAC, plot fission sites by energy
  - Note: example code snippets in slides are incomplete. See materials attached to presentation for full scripts



PTRAC input card and memory usage message

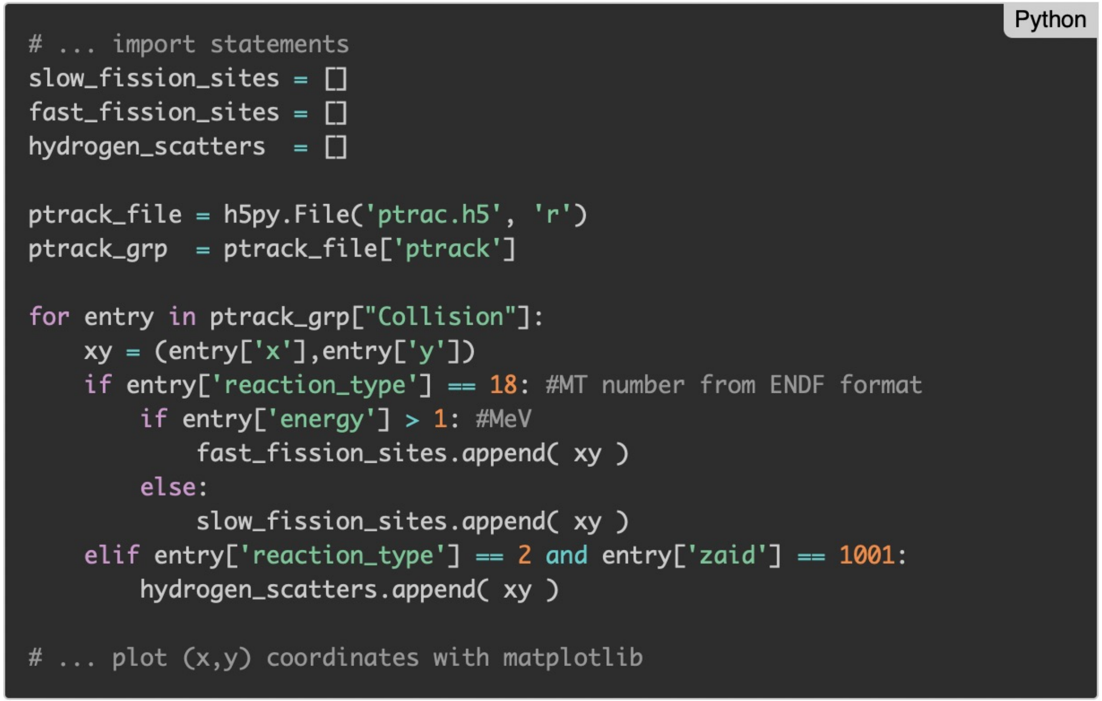
- Ptrac input card for generating collision and source events

```
ptrac file=hdf5 flushnps=1000 event=col,src
```

- This problem produces many collisions per history, so a relatively low value of FLUSHNPS was used:

```
dump    1 on file runtpe.h5   nps =         0  coll =           0
xact is done

cp0 = 0.05

    >>> ptrac data buffer peak memory usage is approx. 1.46E+03 MB between writes
    >>> estimated over the first 1000 histories
    ...
```

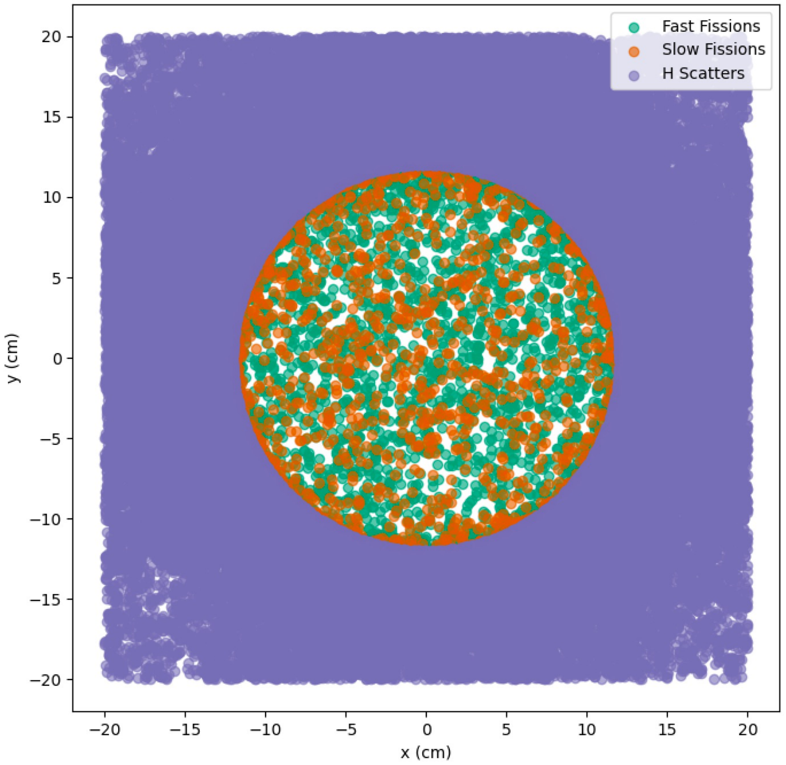
Use h5py to process ptrac.h5; plot fission sites by incident energy and load scatters by reaction ZAID



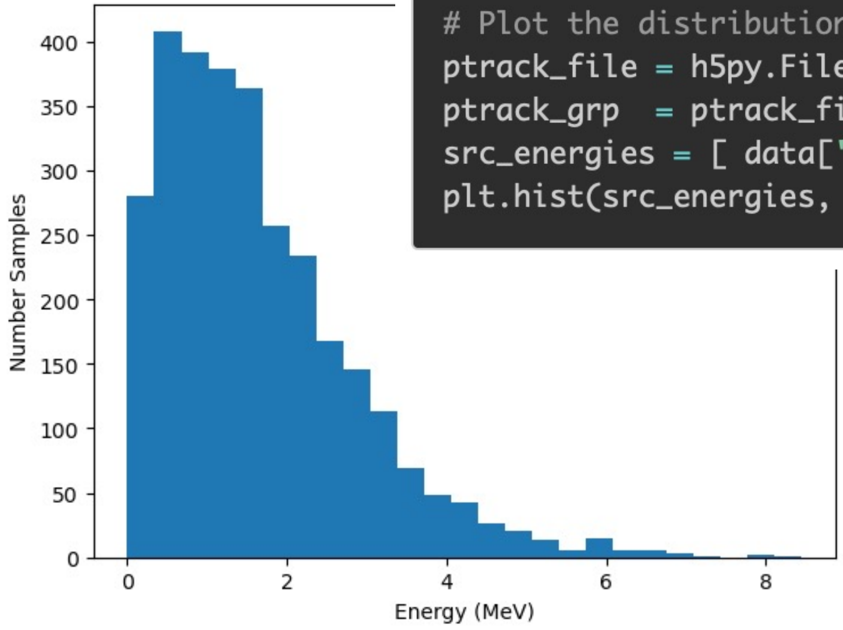
Python

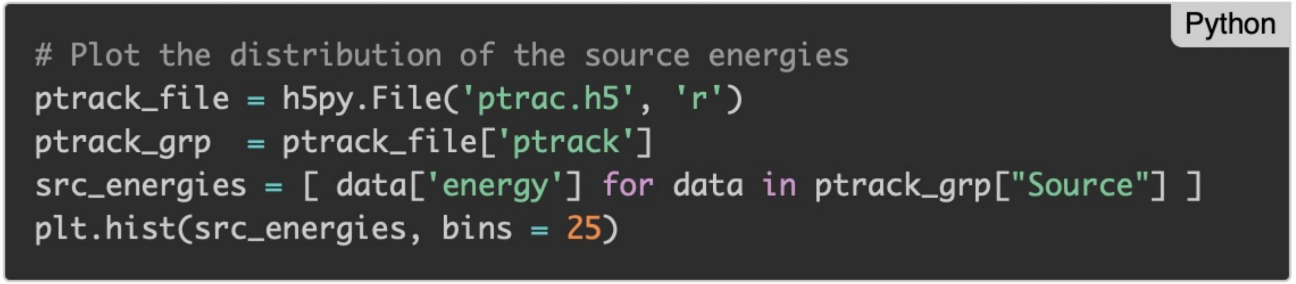
```python
# ... import statements
slow_fission_sites = []
fast_fission_sites = []
hydrogen_scatters  = []

ptrack_file = h5py.File('ptrac.h5', 'r')
ptrack_grp  = ptrack_file['ptrack']

for entry in ptrack_grp["Collision"]:
    xy = (entry['x'],entry['y'])
    if entry['reaction_type'] == 18: #MT number from ENDF format
        if entry['energy'] > 1: #MeV
            fast_fission_sites.append( xy )
        else:
            slow_fission_sites.append( xy )
    elif entry['reaction_type'] == 2 and entry['zaid'] == 1001:
        hydrogen_scatters.append( xy )

# ... plot (x,y) coordinates with matplotlib
```

Plot of fast (> 1 MeV) and slow fission, as well as location of scatters off of hydrogen



Creating a histogram of the source spectra is straightforward with direct access

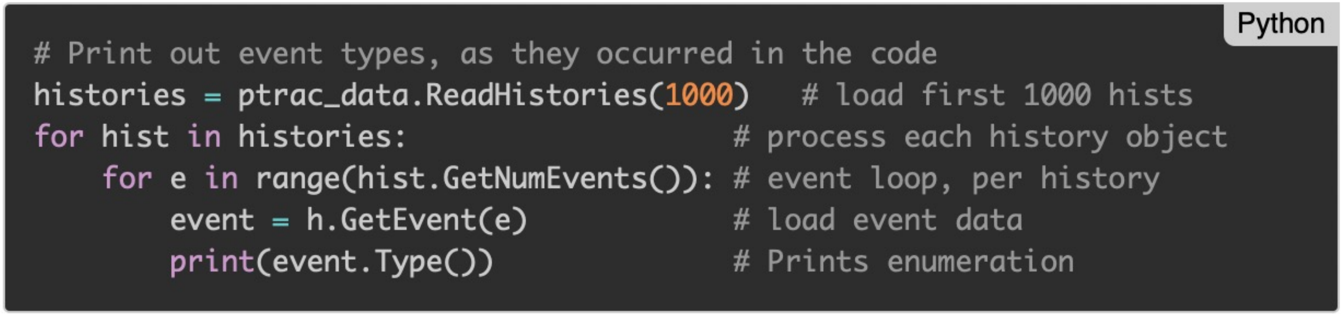




```python
# Plot the distribution of the source energies
ptrack_file = h5py.File('ptrac.h5', 'r')
ptrack_grp  = ptrack_file['ptrack']
src_energies = [ data['energy'] for data in ptrack_grp["Source"] ]
plt.hist(src_energies, bins = 25)
```

PTRAC files (HDF5 and legacy) can also be processed with the MCNPTools library

- MCNPTools is a collection of python and C++ post-processing software
  - Will be posted to LANL Github around MCNP6.3 release
- While file formats in MCNP6 may change over time, the MCNPTools interface will require minimal changes
- For PTRAC files, MCNPTools simplifies access to events in simulation order, for all histories


Python

```python
# Print out event types, as they occurred in the code
histories = ptrac_data.ReadHistories(1000)   # load first 1000 hists
for hist in histories:                       # process each history object
    for e in range(hist.GetNumEvents()): # event loop, per history
        event = h.GetEvent(e)            # load event data
        print(event.Type())              # Prints enumeration
```

Outline

- PTRAC overview
- What is new in MCNP6.3?
  - Parallel PTRAC
  - HDF5 format
- Post-processing with MCNPTools and Python
- Future changes

Take-aways and future plans

- MCNP6.3 has a new HDF5 format that can be executed in parallel
  - Use MCNPTools or h5py to analyze results
- The legacy formats are serial-only and **deprecated**
  - We are not planning to support EVENT=CAP and COINC keywords. Please contact mcnp_help@lanl.gov if you *need* these features.
- Several helpful features are (hopefully) coming in the future
  - Remove need for user-specified `FLUSHNPS`, in most cases
  - Tracking the generation of secondary particles and their events
  - Expanded MCNPTools support to help replace deprecated features (EVENT=CAP)
  - Ability for continue runs and expanded filtering options

Questions?

Contact: sbolding@lanl.gov
mcnp_help@lanl.gov

Citations

1. "DRiFT – A Detector Response Function Toolkit for MCNP Output", M.T. Andrews, et. al. Los Alamos National Laboratory, LA-UR-16-27166.

Backup Content

- Attached to this document are the full scripts that can be used to generate the results shown.

Accessing the fields of a compound dataset

- It can be a little tricky to get access to the available fields of a compound dataset, particularly in older versions of h5py:

```
import h5py
import matplotlib as mpl
#mpl.use('Agg') # For certain HPC backends
import matplotlib.pyplot as plt

# -------------------------------------------------------
# NOTE: rename this file to plot_ptrac.py. It could not
# be attached to the PDF unless it had extension .txt
#
# This file assumes you have a valid installation of
# MCNPTools available, and have ran a corresponding input
# file `pincell.txt` to produce a ptrac.h5 output file. It
# is also necessary to have h5py and matplotlib installed
# with the python distribution.
#
# To run: python3 plot_ptrac.py
#
# It is straightforward to extend the parsing in this file
# to other uses. If you do not have MCNPTools, the first
# part of the script will successfully complete, and the
# mcnptools code below it can be removed.
# -------------------------------------------------------

def plot_points(ax, points, color, label=None):
    """Plot several points over geometry"""
    x = [pair[0] for pair in points]
    y = [pair[1] for pair in points]
    ax.scatter(x,y,c=color,alpha=0.6,label=label)
    return


def plot_results_with_h5py():
    """Directly plot results in h5py"""
    ptrack_file = h5py.File('ptrac.h5', 'r')
    ptrack_grp  = ptrack_file['ptrack']

    # In older versions of h5py, it is a little tricky to access the
datatype:
    print("What fields are available?: ",
ptrack_grp["Collision"].dtype.fields.keys())

    # Plot the distribution of the source energies
    src_energies = [ data['energy'] for data in ptrack_grp["Source"] ]
    fig = plt.figure()
    ax  = fig.add_subplot(111)
    ax.hist(src_energies, bins = 25)
    ax.set_xlabel("Energy (MeV)")
    ax.set_ylabel("Number Samples")
    fig.savefig("energy_spectra.pdf", bbox_inches='tight')
    # Bin fission sites by incident neutron enery, and plot H scatters
    slow_fission_sites = []
    fast_fission_sites = []
    hydrogen_scatters  = []

    for entry in ptrack_grp["Collision"][1:1000000]: #load subset to
reduce plotting strain
        xy = (entry['x'],entry['y'])
        if entry['reaction_type'] == 18: #MT number from ENDF format
            if entry['energy'] > 1: #MeV
                fast_fission_sites.append( xy )
            else:
                slow_fission_sites.append( xy )
        elif entry['reaction_type'] == 2 and entry['zaid'] == 1001:
            hydrogen_scatters.append( xy )

    # Plot the points of fast and slow fissions, as well as hydrogen
scatters
    # simply to outline the geometry
    fig = plt.figure()
    ax  = fig.add_subplot(111)
    plot_points(ax, fast_fission_sites,'#1b9e77',label="Fast
Fissions")
    plot_points(ax, slow_fission_sites,'#d95f02',label="Slow
Fissions")
    plot_points(ax, hydrogen_scatters, '#7570b3', label="H Scatters")
    ax.set_xlabel("x (cm)")
    ax.set_ylabel("y (cm)")
    ax.legend(loc='upper right')
    ax.set_aspect(1)
    fig.savefig("h5py_plot.pdf", bbox_inches='tight')

    print("Number of fast fissions:
{}".format(len(fast_fission_sites)))
    print("Number of slower fissions:
{}".format(len(slow_fission_sites)))
    print("Number of scatters: {}".format(len(hydrogen_scatters)))


def plot_results_with_mcnptools():
    """Plot same results as before but with MCNPTools"""
    slow_fission_sites = []
    fast_fission_sites = []
    hydrogen_scatters  = []

    from mcnptools import Ptrac

    # Open in mcnptools
    pdata = Ptrac("ptrac.h5", Ptrac.HDF5_PTRAC)
    # Read in batches
    num_collisions = 0
    while True:

        # Read histories in iterations, until 3M collisions have been
processed
        hists = pdata.ReadHistories(1000)
        if len(hists) == 0 or num_collisions > 3000000:
            break

        for h in hists: # history loop
            for e in range(h.GetNumEvents()): # event loop, per
history
                event = h.GetEvent(e)
                xy = (event.Get(Ptrac.X), event.Get(Ptrac.Y))
                if event.Type() == Ptrac.COL:
                    num_collisions += 1
                    mt_number = event.Get(Ptrac.RXN) # See manual
                    if mt_number == 18:
                        if event.Get(Ptrac.ENERGY) > 1.0:
                            fast_fission_sites.append(xy)
                        else:
                            slow_fission_sites.append(xy)

    fig = plt.figure()
    ax  = fig.add_subplot(111)
    plot_points(ax, fast_fission_sites,'#1b9e77',label="Fast
Fissions")
    plot_points(ax, slow_fission_sites,'#d95f02',label="Slow
Fissions")
    plot_points(ax, hydrogen_scatters, '#7570b3', label="H Scatters")
    ax.legend(loc='upper right')
    ax.set_aspect(1)
    fig.savefig("mcnptools_plot.pdf", bbox_inches='tight')

plot_results_with_h5py()
plot_results_with_mcnptools()

# Display all the photos, a copy is also saved in the working
directory
plt.show()
```

```
Reflected box of Uranium and neutron source
c >> Cell Cards
1  1000 -19.1   -10 -20 imp:n=1   $ Uranium Fuel
2  2000 -1.0     10 -20 imp:n=1   $ Water
3  0             20     imp:n=0   $ void

c >> Surface Cards
10 CZ   12.0
*20 RPP   -20. 20. -20 20 -20 20

c >> Data Cards
mode n
sdef pos=0 0 0 ext=d1 rad=d2 erg=d3 axs=0 0 1
si1 -20 20 $ uniform height
sp1  0  1
si2  0 12.0         $ uniform in volume
sp2 -21 1
sp3 -3 0.648 6.81057    $ U238 spontaneous fission
c kcode 10000 1.0 10 50 $ For criticality calculation
nps 3000                $ For fixed source
m1000  92235.70c  -0.12
       92238.70c  -0.88
m2000  1001.70c   2
       8016.70c   1
mt2000 lwtr.10t
print
c Write collision and source events for post processing
ptrac file=hdf5 flushnps=1000 event=col,src
```