LA-UR-25-26049

Approved for public release; distribution is unlimited.

Title: A Python Tool for Reconstructing MCNP6 Particle Histories from an HDF5 PTRAC File

Author(s): Weaver, Colin Andrew
Vaquer, Pablo Andres

Intended for: 2025 MCNP User Symposium, 2025-07-07/2025-07-10 (Los Alamos, New Mexico, UNITED STATES)

Issued: 2025-08-04 (rev.1)

Los Alamos NATIONAL LABORATORY

Los Alamos NATIONAL LABORATORY

A Python Tool for Reconstructing MCNP6 Particle Histories from an HDF5 PTRAC File

C.A. Weaver and P.A. Vaquer

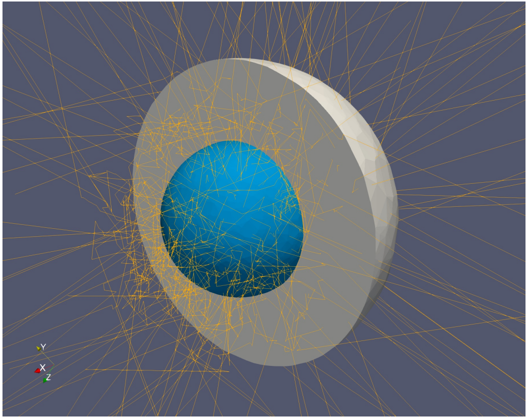
2025 MCNP User Symposium
July 9, 2025

LA-UR-25-26049

Abstract

A Python tool for converting the MCNP6 HDF5 PTRAC file to a list of Python trees is presented. The particle trees store MCNP6 simulated events for each history using parent-child relationships, which ensures that branching processes are accurately reproduced. A variety of post-processing scripts are presented and used in conjunction with the Python particle trees to make special tallies that are currently not available in the MCNP6 software and visualize the particle tracks.

Table of Contents



Overview

PTRAC: Particle Track Output

> *The `PTRAC` card generates an output file of user-filtered particle events referred to as a particle track file. Adding a `FILE=HDF5` entry will produce an HDF5 output file...*
>
> — *MCNP® Code Version 6.3.1 Theory & User Manual [1]*

```
/ ............................................................ (root)
├── config_control ........................................... (group)
├── problem_info ............................................. (group)
└── ptrack ................................................... (group)
    ├── RecordLog .......................................... (dataset)
    ├── Bank ............................................... (dataset)
    ├── Collision .......................................... (dataset)
    ├── Source ............................................. (dataset)
    ├── SurfaceCrossing .................................... (dataset)
    └── Termination ........................................ (dataset)
```

Los Alamos
NATIONAL LABORATORY

Particle Trees

- The MCNP6 HDF5 [2] PTRAC file stores events in the order that they occured in the Monte Carlo random walk. Different histories may be out of order, if the MCNP6 input file is executed in parallel, but this does not affect the order of within-history events
- Bank events are transported after their progenitor is tracked to termination. The PTRAC file does not store progenitor information and it is not known *a priori* which event made the bank event
- A Python tool is developed with the `anytree` library to construct “particle trees” that accurately preserve the MCNP6 branching processes
- Bank events are connected to their progenitor event by matching the (x, y, z) coordinates of the bank event to the progenitor event. The same applies to source events in a `KCODE` problem

Scope and Limitations

The standalone implementation of this tool is called PyTrac (Python PTRAC).
Disclaimer: This name may change when it migrates to the MCNP Python package.

Fixed Source

```
from pytrac import PyTrac

pytrac = PyTrac('ptrac.h5')
```

k-Eigenvalue

```
from pytrac import PyTrac

pytrac = PyTrac('ptrac.h5',
                kcode=True)
```

- PyTrac has been developed and tested for neutron transport
- It successfully reconstructs the Monte Carlo random walk with branching processes for both fixed-source and k-eigenvalue problems
- It does not work for problems that use the “condensed history” Monte Carlo method or similar methods that do not record progenitor events in the PTRAC file

Special Tallies

Source Sensitivity Coefficients

Consider an alternative form of Taylor's series:

$$\frac{f(x)-f(a)}{f(a)} = \sum_{n=1}^{\infty} S_n \left(\frac{x-a}{a}\right)^n,$$

where the n-th order sensitivity coefficient is

$$S_n \equiv \frac{a^n}{n!}\frac{f^{(n)}(a)}{f(a)}.$$

Example, *evaporation energy spectrum*:

$$p(E) = CE\exp(-E/a)$$

Thermal Surface Flux with 10% Error

Response	0.07858 ± 0.15574
Order 1	-0.94652 ± 0.20611
Order 2	0.78699 ± 0.21912
Propag. Unc.	-0.86783

Fast Surface Flux with 10% Error

Response	1.17835 ± 0.02738
Order 1	0.05086 ± 0.04856
Order 2	-0.16130 ± 0.04956
Propag. Unc.	0.03473

PyTrac for Source Sensitivity Coefficients

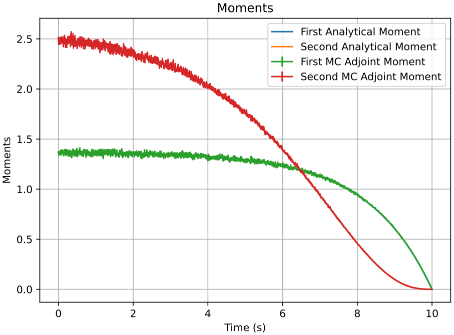
```
from pytrac import PyTrac
from pytrac.processors.source_sensitivity import SourceSensitivity
```

```
pytrac = PyTrac('ptrac.h5')

sen = SourceSensitivity(
    pytrac,
    sensitivity_filters={
        "material_id": [1],
        "cell_id": [10],
        "particle_type": [1],
        "source_type": [40],
        "order": [1,2]
    }
)
```

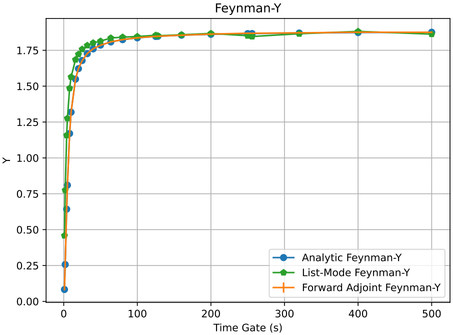
```
thermal =
    lambda p: 2.5e-8 < p.energy < 1e-6
    and p.surface_id == 100
sen.calculate_sensitivity(thermal)
sen.calculate_error(0.1)

fast =
    lambda p: 0.001 < p.energy < 10
    and p.surface_id == 100
sen.calculate_sensitivity(fast)
sen.calculate_error(0.1)
```

Adjoint-based Feynman-Y Calculations
Figures provided by R.T. Johnson

Ongoing doctoral research is studying how the MCNP software can be used to calculate stochastic neutron transport moments and Feynman-Y values.



Stochastic neutron transport moments



Feynman-Y time-gate profile

Visualization

PyTrac for Terminal Visualization

```
from pytrac import PyTrac
from pytrac.processors.print_trees import print_trees

pytrac = PyTrac('ptrac.h5')
print_trees(
    pytrac.tree_root_nodes,
    print_particle_fields=['reaction_type'],
    particle_filter=lambda p: p.reaction_type == 16
)

pytrac = PyTrac('ptrac.h5', kcode=True)
print_trees(
    pytrac.tree_root_nodes,
    print_particle_fields=['cell_id', 'energy']
)
```

Terminal Output for a Fixed-Source Problem

```
SRC: reaction_type=None
+-- COL: reaction_type=16 [FILTER]
    |-- COL: reaction_type=2
    |   +-- SUR: reaction_type=None
    |       +-- TER: reaction_type=None
    +-- BNK: reaction_type=2
        +-- COL: reaction_type=2
            +-- COL: reaction_type=2
                +-- COL: reaction_type=2
                    +-- COL: reaction_type=2
                        +-- SUR: reaction_type=None
                            +-- TER: reaction_type=None
```

Los Alamos NATIONAL LABORATORY

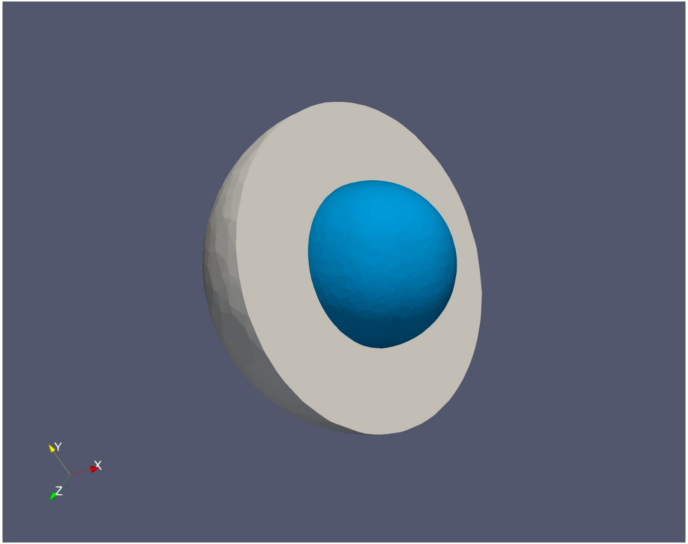
Terminal Output for a k-Eigenvalue Problem

```
SRC: cell_id=10, energy=0.0745
+-- COL: cell_id=10, energy=0.0740
    +-- COL: cell_id=10, energy=0.0735
        +-- COL: cell_id=10, energy=0.0728
            |-- COL: cell_id=10, energy=0.0720
            |   |-- COL: cell_id=10, energy=0.0718
            |   |   +-- COL: cell_id=10, energy=0.0714
            |   |       +-- COL: cell_id=10, energy=0.0703
            |   |           +-- COL: cell_id=10, energy=0.0703
            |   |               +-- TER: cell_id=10, energy=0.0703
            |   +-- SRC: cell_id=10, energy=2.5896
            |       +-- COL: cell_id=10, energy=2.5759
            |           +-- COL: cell_id=10, energy=2.5749
            |               +-- COL: cell_id=10, energy=2.5678
            |                   +-- COL: cell_id=10, energy=2.5678
            |                       +-- TER: cell_id=10, energy=2.5678
            +-- SRC: cell_id=10, energy=0.7702
                +-- COL: cell_id=10, energy=0.3962
                    +-- COL: cell_id=10, energy=0.3951
                        +-- COL: cell_id=10, energy=0.0240
                            +-- COL: cell_id=10, energy=0.0240
                                +-- TER: cell_id=10, energy=0.0240
```

PyTrac for Unstructured Mesh Generation

```
from pytrac import PyTrac
from pytrac.processors.reconum import reconum

pytrac = PyTrac('ptrac.h5')

cell_ids = [10, 20]

for cell_id in cell_ids:
    reconum(
        pytrac.tree_root_nodes,
        cell_id = cell_id
    )
```

ParaView [3] Visualization of Two Nested Spheres

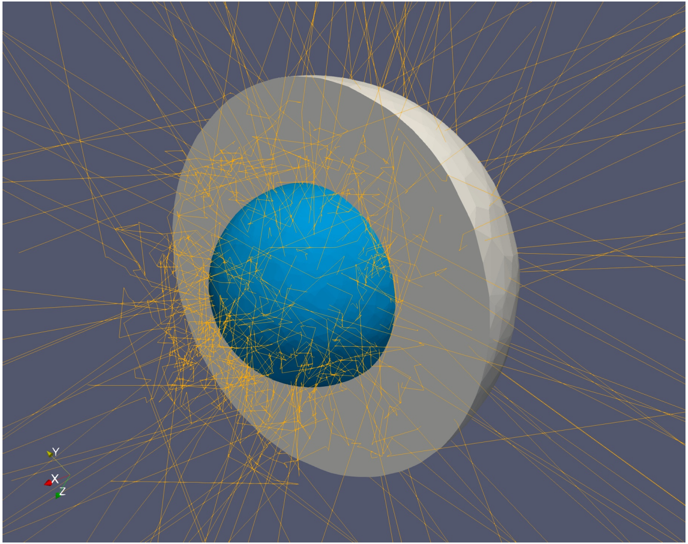


PyTrac for Particle Track Visualization

```
from pytrac import PyTrac
from pytrac.processors import make_vtp_and_pvtp_files, make_pvd_file
pytrac = PyTrac('ptrac.h5')
vtp_files, pvtp_files = make_vtp_and_pvtp_files(pytrac, "vtk_tracks")
make_pvd_file(vtp_files, "tracks.pvd")
make_pvd_file(pvtp_files, "total_tracks.pvd")
```

- `VTP`: A VTK PolyData file
  - Contains tracks for single particle history
- `PVTP`: A Parallel VTK PolyData file
  - Contains tracks for particle histories 1 through N
  - Simply references a set of VTP files
- `PVD`: A ParaView Data collection file
  - Used for stepping through a series of datasets/snapshots
  - Partically useful for time-dependent simulations
  - Simply references a set of files (VTP, PVTP, etc.)

ParaView Visualization of Particle Tracks



PyTrac to Visualize Tracks in k-Eigenvalue Problems

Step 1: Run MCNP with `KCODE` card to generate a SRCTP file

```
> mcnp6 i=mcnp_input_file.inp tasks 12
```

Step 2: Add a `PTRAC` card to the same input file and run MCNP again using existing SRCTP as initial guess

```
> mcnp6 i=mcnp_input_file.inp src=srctp tasks 12
```

Step 3: Use PyTrac with `kcode=True`

```
from pytrac import PyTrac
from pytrac.processors import make_vtp_and_pvtp_files, make_pvd_file
pytrac = PyTrac('ptrac.h5', kcode=True)
vtp_files, pvtp_files = make_vtp_and_pvtp_files(pytrac, "vtk_tracks")
make_pvd_file(vtp_files, "tracks.pvd")
make_pvd_file(pvtp_files, "total_tracks.pvd")
```

PyTrac to Make Movies for Visualizing Tracks

```
from pytrac import PyTrac
from pytrac.processors import make_vtp_and_pvtp_files, make_pvd_file
import pyvista as pv
import imageio

# Load the ptrac.h5 file and create VTP files
pytrac = PyTrac('ptrac.h5', max_nps=200)
vtp_files, pvtp_files = make_vtp_and_pvtp_files(pytrac, "vtk_tracks")

# Create a plotter and specify as camera view
pv.start_xvfb()
plotter = pv.Plotter(off_screen=True)
plotter.camera_position = [
    (0, -250, 0),  # camera location
    (0,    0, 0),  # focal point
    (0,    0, 1)   # "upwards" direction
]
images = []

# Loop through each VTP file and save a screenshot
for vtp_file in vtp_files:
    mesh = pv.read(vtp_file)
    plotter.add_mesh(mesh)
    img = plotter.screenshot()
    images.append(img)

# Write the images to a GIF file
imageio.mimsave("animated.gif", images, duration=0.05)
```

Conclusions

Summary and Outlook

- The MCNP6 HDF5 PTRAC file is used to reproduce the Monte Carlo random walk of a particle history with branching processes
- This is useful for making special tallies that are currently unavailable in the MCNP software and visualizing problems
- The standalone instance of this tool is called PyTrac but the intent is to relocate it to the new MCNP Python package under a more generic name

References

[1] Joel A. Kulesza et al. *MCNP® Code Version 6.3.1 Theory & User Manual*. Tech. rep. LA-UR-24-24602, Rev. 1. Los Alamos, NM, USA: Los Alamos National Laboratory, May 2024. DOI: 10.2172/2372634. URL: https://www.osti.gov/biblio/2372634.

[2] The HDF Group. *Hierarchical Data Format, version 5*. URL: https://github.com/HDFGroup/hdf5.

[3] Utkarsh Ayachit. *The ParaView Guide: A Parallel Visualization Application*. Clifton Park, NY, USA: Kitware, Inc., 2015. ISBN: 1930934300.

Questions?

Thank you!

caweaver@lanl.gov

vaquer@lanl.gov