LA-UR-22-30927

Approved for public release; distribution is unlimited.

Title: Coincident Capture through Post-processing PTRAC

Author(s): Rising, Michael Evan
Bolding, Simon R.

Intended for: 2022 MCNP User Symposium, 2022-10-17/2022-10-21 (Los Alamos, New Mexico, United States)

Issued: 2022-10-25 (rev.1)

Los Alamos
NATIONAL LABORATORY

Coincident Capture through Post-processing PTRAC

Michael E. Rising and Simon Bolding, XCP-3, LANL

2022 MCNP® User Symposium

October 17–21, 2022

LA-UR-22-30927

MCNP® Trademark

MCNP® and Monte Carlo N-Particle® are registered trademarks owned by Triad National Security, LLC, manager and operator of Los Alamos National Laboratory. Any third party use of such registered marks should be properly attributed to Triad National Security, LLC, including the use of the ® designation as appropriate.

- Please note that trademarks are adjectives and should not be pluralized or used as a noun or a verb in any context for any reason.
- Any questions regarding licensing, proper use, and/or proper attribution of Triad National Security, LLC marks should be directed to trademarks@lanl.gov.

Thanks and Acknowledgements

This work is supported by the LANL ISTI Program.

This work is supported by the Department of Energy through Los Alamos National Laboratory (LANL) operated by Triad National Security, LLC, for the National Nuclear Security Administration (NNSA) under Contract No. 89233218CNA000001.

Outline

Motivation

PTRAC Updates: Features and Format

Development of Nuclear Safeguards Examples

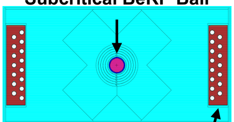
Coincident Counting through PTRAC

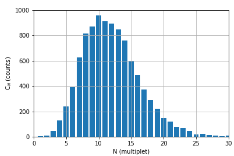
Summary

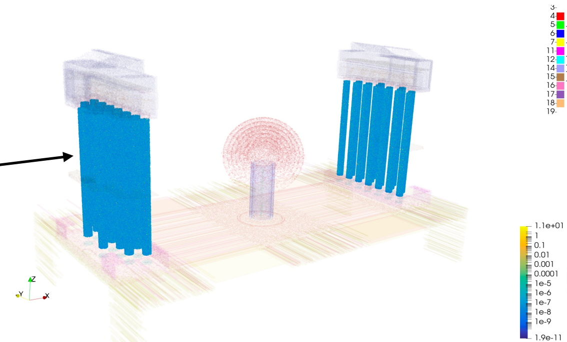
Particle Track Output (PTRAC) Usage

- PTRAC is often used for advanced detector responses, where correlated or time-dependent analysis is needed
- The PTRAC file is used as input for custom post-processing software
  - Examples include Advanced detector response simulation framework DRiFT [1]
  - Subcritical multiplicity experiments

Subcritical Multiplication Analysis and Visualization







PTRAC Updates: Features and Format

PTRAC Input Card (1)

- For separate output file printing of all or partial (filtered) histories and events from a transport calculation
- Allows greater user control for specialized result processing when standard and special treatment tallies are inadequate
- Use PTRAC input card and keyword-value pairs (more on next slide):

keyword	value(s)	description
file	bin, asc, hdf5	bin=binary, asc=ASCII, hdf5=HDF5
max	integer	maximum of number of events written
write	pos, all	pos=x,y,z particle info only, all=x,y,z,u,v,w,wgt,tme,erg info
coinc	col	print tally scores by history (need tally keyword also)
flushnps	integer	controls write frequency for HDF5 output file type

red = deprecated in MCNP6.3, blue = new in MCNP6.3

Los Alamos NATIONAL LABORATORY

PTRAC Input Card (2)

- Event-based filtering on the PTRAC input card:

keyword	value(s)	description
event	src, bnk, sur, col, ter, cap	event-type filter: src=source, bnk=bank, sur=surface, col=collision, ter=termination, cap=coincident capture
filter	PBL	particle state variables
type	$\mathscr{P}$	particle-type filter

- History-based filtering on the PTRAC input card:

keyword	value(s)	description
nps	integer	range of nps history numbers
cell	integer	list of cell numbers
surface	integer	list of surface numbers
tally	integer	list of tally numbers
value	float	list of tally contribution thresholds

red = deprecated in MCNP6.3

HDF5-formatted `PTRAC` in MCNP6.3

- HDF5 `PTRAC` simulations can be executed in parallel
  - Removes a significant computational bottleneck
  - Even in serial, HDF5 `PTRAC` is faster for large problems
- Organized output structure makes post-processing more accessible
  - Reduces processing errors of legacy formats
  - More flexible, so it can be extended in the future
- The MCNP6.3 release notes provide more detail on the feature
  - Several `PTRAC` bug fixes
  - Legacy formats and two infrequently used features are DEPRECATED
  - Improved interface for event-wise cell and surface features

HDF5 is Binary, but Easily Interrogated

- New HDF5 PTRAC file format described in MCNP6.3 user manual

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

- Each group (e.g., /ptrack/) is like a filesystem directory
- Each dataset (e.g., Bank) is just an array of data that can be processed
- Interrogate the HDF5 file from the command line:
  - h5ls and h5dump for terminal usage

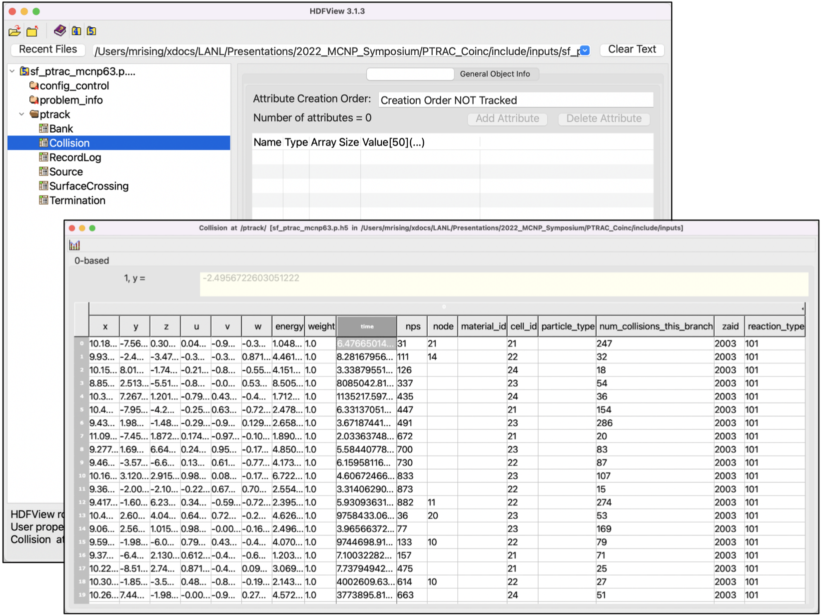
```
h5ls -r sf_ptrac_mcnp63.p.h5
/                            Group
/config_control              Group
/problem_info                Group
/ptrack                      Group
/ptrack/Bank                 Dataset {0/Inf}
/ptrack/Collision            Dataset {67170/Inf}
/ptrack/RecordLog            Dataset {67170/Inf}
/ptrack/Source               Dataset {0/Inf}
/ptrack/SurfaceCrossing      Dataset {0/Inf}
/ptrack/Termination          Dataset {0/Inf}
```

Compound-data Structure: Layout

Collision event compound data type fields:

Data Field	Description
x	x coordinate of the particle position
y	y coordinate of the particle position
z	z coordinate of the particle position
u	Particle direction cosine relative to $+x$ axis
v	Particle direction cosine relative to $+y$ axis
w	Particle direction cosine relative to $+z$ axis
energy	Particle energy
weight	Particle weight
time	Time at particle position
nps	History identifier
node	Number of nodes in track from source to here
material_id	Program material ID of the cell containing event
cell_id	Problem number of the cell containing event
num_collisions_this_branch	Count of collisions per track
reaction_type	Number identifying the reaction type
zaid	**ZZZAAA** for reaction isotope
particle_type	particle type enumeration

Compound Data Structure: HDFView



Access to HDF5 PTRAC Data

MCNPTools

- Version 3.8 released with MCNP6.2
- **Latest version, supports both legacy and HDF5 formats, now available as open source at https://github.com/lanl/mcnptools**
- Example below:

Direct access through HDF5 API's

- Official APIs: C, C++, Fortran, Java
- Unofficial APIs: Julia, Matlab, Mathematica, Perl, Python, R
- Python `h5py` example to follow

MCNPTools PTRAC Processing Example

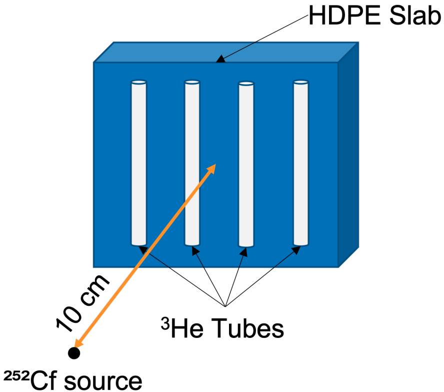
Python

```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
```

Development of Nuclear Safeguards Examples

Safeguards Example for Simple Neutron Detector Coincident Counting (1)

- With our LANL nuclear safeguards colleagues in NEN-1, we developed a new MCNP safeguards-specific class
- Exercises include a simplified neutron detector system for coincident neutorn counting
  - 4 He-3 tubes
  - High Density Polyethylene (HDPE)
  - Cf-252 spontaneous fission (SF) source
- Example **safeguards.inp** MCNP6.3 input file is attached



Safeguards Example for Simple Neutron Detector Coincident Counting (2)

- Consider the options on how a coincident neutron counting simulation can be done

1. Using the pulse-height tally capture (`CAP`) special treatment option
   - Note that this option automatically turns off implicit capture
2. Using `PTRAC` data card that writes all particle data
   - Need to turn off implicit capture (otherwise capture events will NEVER occur and appear in a `PTRAC` file)
   - Could use an event-based collision filter (i.e., `event=col`)
   - Could use an event-based particle cell filter (i.e., `filter=21,24,cel`) within the detector cells

Listing 1: Safeguards MCNP6.3 PTRAC Card

```
c MCNP6.3 ptrac card
c
ptrac file=hdf5  flushnps=1e6
      event=col  filter=21,24,cel
```

Coincident Counting through PTRAC

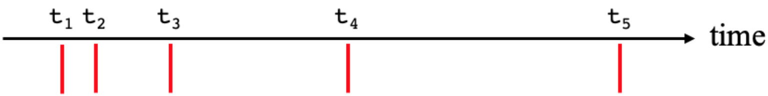
Safeguards Example for Simple Neutron Detector Coincident Counting (1)

Listing 2: Coincident Counting Python (ptrac_coinc.py.txt attached) Example: Main

```python
def main():
    """Main execution script"""

    parser = parse_args()
    args = parser.parse_args()

    if args.verbose == 1:
        level = logging.INFO
    elif args.verbose > 1:
        level = logging.DEBUG
    else:
        level = logging.WARNING
    logging.basicConfig(format="%(levelname)s: %(message)s", level=level)
    logging.debug(args)

    ptrac_file = {"name": args.input, "fmt": args.format}
    history_filter = {"cells": args.cells, "zas": args.isotopes, "rxns": args.reactions}
    times_filter = {"predelay_time": args.predelay, "gate_width": args.gatewidth}

    print("\nIndividual history coincident counting results via MCNPTools")
    count_histogram = process_ptrac_via_mcnptools(
        ptrac_file, history_filter, times_filter
    )
    output_counts(count_histogram, file_prefix=f"{args.output}_mcnptools")

    print("\nAll histories coincident counting results via h5py")
    count_histogram = process_ptrac_via_h5py(ptrac_file, history_filter, times_filter)
    output_counts(count_histogram, file_prefix=f"{args.output}_h5py")
```

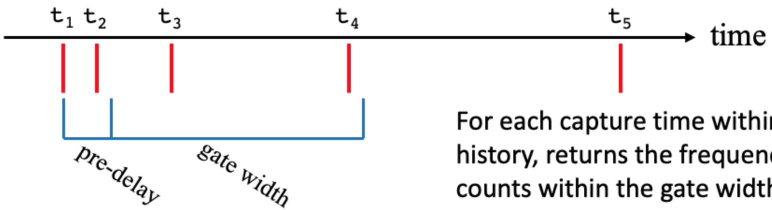
Safeguards Example for Simple Neutron Detector Coincident Counting (2)

Need a list of times the capture events took place in the detectors:



Returns list of sorted capture times: `[t1,t2,t3,t4,t5]`

Apply pre-delay and gate-width time filters to simulate detector effects:



For each capture time within the history, returns the frequency of counts within the gate width

Using MCNPTools to Process Individual Histories (1)

Listing 3: Coincident Counting Python Example: Processing Histories

```
def process_ptrac_via_mcnptools(ptrac_file, history_filter, times_filter):
    """Processing PTRAC file into a histogram of coincident counts using MCNPTools."""

    fmt_to_mcnptools = {
        "ascii": Ptrac.ASC_PTRAC,
        "binary": Ptrac.BIN_PTRAC,
        "hdf5": Ptrac.HDF5_PTRAC,
    }
    count_histogram = np.array([0])

    # Open file and then read a chunk of 1000 histories
    ptrac_handle = Ptrac(ptrac_file["name"], fmt_to_mcnptools[ptrac_file["fmt"]])
    ptrac_hists = ptrac_handle.ReadHistories(1000)
    while ptrac_hists:

        # Iterate through each individual history
        for history in ptrac_hists:

            # Call time filter function to get sorted list of capture times
            times = filter_history_times(history, history_filter)

            # Call histogram function to process capture times
            counts = histogram_time_gate(times, times_filter)

            # Accumulate history histogram into total histogram
            extend = len(counts) - len(count_histogram)
            if extend > 0:
                extend_histogram = np.zeros(extend, dtype=int)
                count_histogram = np.concatenate((count_histogram, extend_histogram))
            for icount, count in enumerate(counts):
                count_histogram[icount] += count

        # Read next chunk of 1000 histories
        ptrac_hists = ptrac_handle.ReadHistories(1000)

    return count_histogram
```

Using MCNPTools to Process Individual Histories (2)

Listing 4: Coincident Counting Python Example: Cell/Isotope/Reaction Filtering

```
    def filter_history_times(history, history_filter):
        """Function to process a single MCNPTools PTRAC history.
           Returns a sorted list of reaction times.
        """

        number_events = history.GetNumEvents()

        times = list()
        for ievent in range(number_events):
            event = history.GetEvent(ievent)

            if event.Type() == Ptrac.COL:
                # Gather particle collision cell, isotope, and reaction
                cell = int(event.Get(Ptrac.CELL))
                za = int(event.Get(Ptrac.ZAID))
                rxn = int(event.Get(Ptrac.RXN))

                # Filter all capture reactions within cells and isotopes
                if (
                    cell in history_filter["cells"]
                    and za in history_filter["zas"]
                    and rxn in history_filter["rxns"]
                ):
                    times.append(event.Get(Ptrac.TIME))

        return sorted(times)
```

Using Python h5py to Process All Histories (1)

Listing 5: Coincident Counting Python Example: Processing Histories

```
def process_ptrac_via_h5py(ptrac_file, history_filter, times_filter):
    """Processing PTRAC file into a histogram of coincident counts using h5py."""

    # Open HDF5 file and iterate over collision group
    if ptrac_file["fmt"] != "hdf5":
        logging.error("HDF5 file required for h5py processing")
    ptrac_handle = h5py.File(ptrac_file["name"], "r")
    ptrac_group = ptrac_handle["ptrack"]

    # Call time filter function to get sorted list of capture times
    times = filter_history_times(ptrac_group, history_filter)

    # Call histogram function to process capture times
    count_histogram = histogram_time_gate(times, times_filter)

    return count_histogram
```

Using Python `h5py` to Process All Histories (2)

Listing 6: Coincident Counting Python Example: Cell/Isotope/Reaction Filtering

```python
    def filter_history_times(group, history_filter):
        """Function to process all histories in HDF5 collision group.
           Returns a sorted list of reaction times.
        """

        times = list()
        for event in group["Collision"][:]:

            # Gather particle collision cell, isotope, and reaction
            cell = event["cell_id"]
            za = event["zaid"]
            rxn = event["reaction_type"]

            # Filter all capture reactions within cells and isotopes
            if (
                cell in history_filter["cells"]
                and za in history_filter["zas"]
                and rxn in history_filter["rxns"]
            ):
                times.append(event["time"])

        return sorted(times)
```

Los Alamos NATIONAL LABORATORY

Running the Example (1)

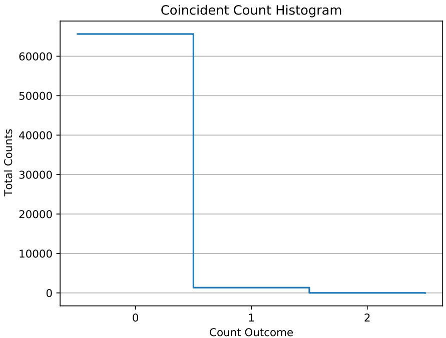
- Executing MCNP6.3

  `> mcnp6 i=safeguards.inp n=sf_ptrac_mcnp63.  tasks 8`

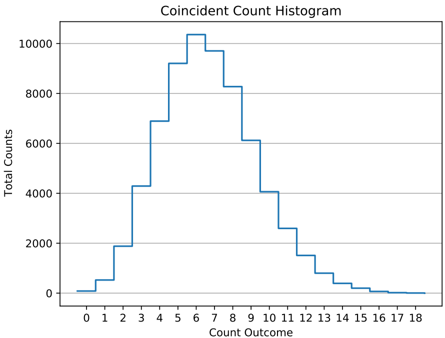
- And then executing the provided Python script

```
> python ptrac_coinc.py.txt -i sf_ptrac_mcnp63.p.h5 -f hdf5 -o high_activity \
 --cells 21 22 23 24 --isotopes 2003 --reactions 101 \
 --predelay 500 --gatewidth 10000
```

MCNPTools Results



h5py Results



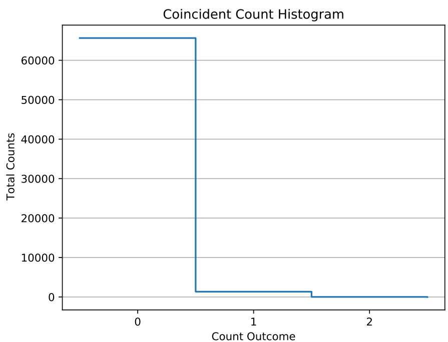
Running the Example (2)

- The individual history processing ignores cross-history effects
- 1M histories simulated uniformly in 1s interval

Listing 7: Safeguards MCNP6.3 PTRAC Card

```
sdef tme=d1 pos=-5 0 0 par=sf
si1  0 1e8
sp1 0 1
fmult data=3 method=3 shift=1
nps 1e6
```

- Changing to a “lower” source activity, i.e., a larger 10^8 s time interval, results in equivalent coincident capture count histograms



Summary

Summary of New PTRAC Capabilities and Workflows

- The PTRAC capability in MCNP6.3 has seen a massive overhaul since MCNP6.2
- The new HDF5 file format allows for both MPI- and thread-based parallelism
- MCNPTools has been updated to handle the new HDF5 PTRAC format and is now open-sourced on GitHub
- Built-in capabilities, such as the pulse-height tally coincident capture special treatment, can largely be replicated through separate postprocesing scripts that leverage both PTRAC and MCNPTools
  - Allows for greater flexibility in user-specified and controlled detector response functionality, ultimately using the MCNP code for what it is best at (i.e. particle transport)
- Look at the MCNP6.3 User Manual [2] and S. Bolding's presentation at the 2021 MCNP User Symposium [3] for more information on the new HDF5 PTRAC format and capabilities

Questions?

Backup Slides

References

[1] M. T. Andrews, C. R. Bates, E. A. Mckigney, A. D. Mullen, S. F. Woldegiorgis, M. E. Rising, M. J. Marcath, and A. Sood, "DRiFT - RELEASE 1.0.0 ORGANIC SCINTILLATORS," Tech. Rep. LA-UR-21-29114, Los Alamos National Laboratory, Sept. 2021.

[2] J. A. Kulesza, T. R. Adams, J. C. Armstrong, S. R. Bolding, F. B. Brown, J. S. Bull, T. P. Burke, A. R. Clark, R. A. Forster, III, J. F. Giron, A. S. Grieve, C. J. Josey, R. L. Martz, G. W. McKinney, E. J. Pearson, M. E. Rising, C. J. Solomon, Jr., S. Swaminarayan, T. J. Trahan, S. C. Wilson, and A. J. Zukaitis, "MCNP® Code Version 6.3.0 Theory & User Manual," Tech. Rep. LA-UR-22-30006, Rev. 1, Los Alamos National Laboratory, Los Alamos, NM, USA, Sept. 2022.

[3] S. R. Bolding, J. A. Kulesza, M. J. Marcath, and M. E. Rising, "Particle Track Output (PTRAC) Improvements, Parallelism, and Post-Processing," Tech. Rep. LA-UR-21-26562, Los Alamos National Laboratory, Aug. 2021.