Los Alamos
NATIONAL LABORATORY

MCNP6.3 Workshop

Michael E. Rising and Avery Grieve, XCP-3, LANL

14th International Conference on Radiation Shielding and 21st Topical Meeting of the Radiation Protection and Shielding Division (ICRS14/RPSD-2022)

Sunday, September 25, 2022

LA-UR-22-29859

Acknowledgements

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.

This work is supported by the LANL MCNP Site Support Project, the LANL LDRD Program, the LANL ISTI Program, the Department of Energy (DOE) Advanced Scientific Computing Program, the DOE Nuclear Criticality Safety Program, and the DOE NA-22 Nuclear Nonproliferation R&D Program.

Thanks to the ICRS14/RPSD-2022 Topical meeting organizers for including us here.

Thanks to those who provided content to this workshop, including Jennifer Alwin, Jerawan Armstrong, Simon Bolding, Forrest Brown, Jeffrey Bull, Alexander Clark, Micky Dzur, Jeff Favorite, Colin Josey, Joel Kulesza, Scott Mosher, Sriram Swaminarayan, and Mara Watson.

Thanks to Jeremy Sweezy for reviewing the content of this workshop.

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.

Outline

MCNP6.3 Overview

New Features

- Fission-matrix-based convergence testing and acceleration
- ***Multigroup cross section tallies for reactor analysis***
- Doppler broadening resonance correction
- Stochastic $S(\alpha, \beta)$ temperature mixing
- Mixed-material treatment for structured meshes
- ***New FMESH tally backend options***
- ***Parallel PTRAC support***
- CMake build system
- ***HDF5-formatted output files and XDMF support***

Covered or partially discussed in this workshop

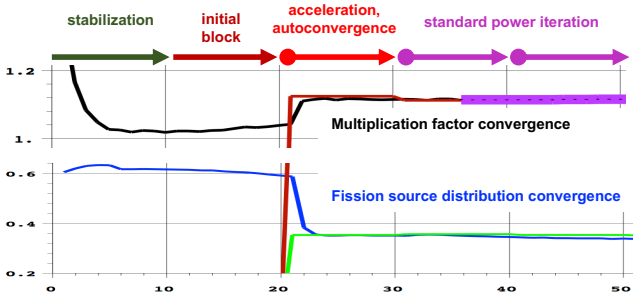
Improvements

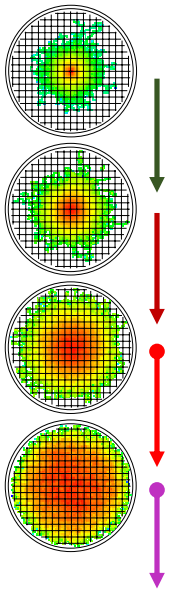
- ***New Qt-based plotter preview available for alpha-testing***
- All of the MCNP code meets Fortran 2008 and C++ 17 standard requirements
- Preliminary support for upcoming ENDF/B-VIII.1 data changes
  - For new $S(\alpha, \beta)$ format options
  - For photonuclear physics
- ***Upgrade to use open-source version of CGMF 1.1.1***
- ***Improved unstructured mesh input file processing***
- ***Added unstructured mesh quality metrics and reporting***
- Decrease of delayed-gamma-line memory usage
- ***Various deprecated features with plans for future removal***

Covered or partially discussed in this workshop

Fission Matrix Convergence Testing and Acceleration

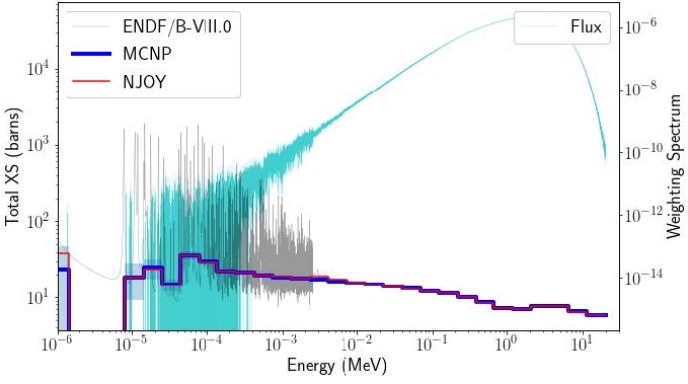
- Automated acceleration of the convergence of the fission source distribution
  - Eliminates user intervention and trial-and-error testing
  - Saves computational cost
- More robust statistical convergence and population testing
  - Ensuring the simulation has converged





Features for Advanced Reactor Analysis

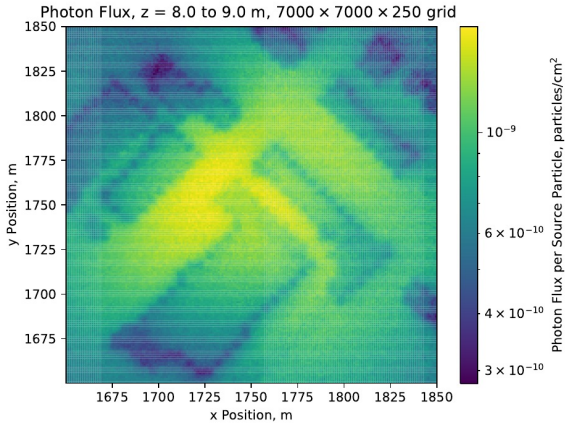
- Unstructured mesh developments for high-fidelity reactor simulations
  - MCNP material properties based on element sets
  - Developed MCNP and ABAQUS based Reactor Multiphysics (MARM) framework for high temperature reactor analysis [1]
- Multigroup cross section tallies computed for coupling to deterministic, time-dependent reactor feedback codes [2]



Mesh Tally Backend Improvements Enabling Extreme-scale, High-fidelity Simulations [3]

- History and batch statistics
- Advanced MPI parallelism options
- Better performance on current problems
- Scaling to extreme scale for higher resolution future problems
  - E.g., full core reactor depletion

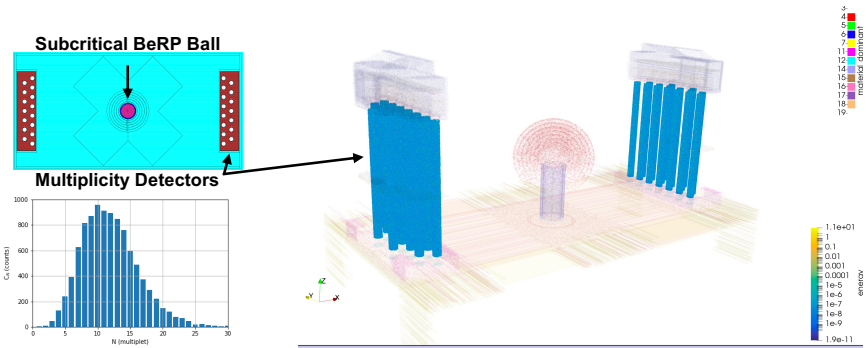
12.2 billion tally region problem run with new FMESH capability



Improved Particle Track Outputs for More Efficient Advanced Detector Response Simulations [4]

- New implementation produces a new file format that makes the particle track output information far simpler and more accessible to users
  - Improving workflow and reduces processing errors
- Particle track outputs are now parallel, removing a computational bottleneck

Subcritical Multiplication Analysis and Visualization



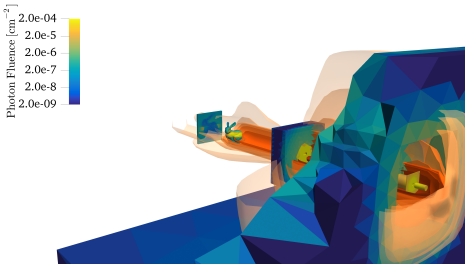
Introduction of the New File Formats (1)

- Introduced a flexible, Hierarchical Data Format (HDF5) to replace legacy binary and ASCII files
  - Permits natural organization
  - Parallel input/output can provide substantial performance
  - Parallel data transfer and distribution provides extensive mesh tally scalability
  - Intrinsic data-compression capability can provide 10–100× file size savings
  - Accessible via C, C++, Fortran, Python, Matlab, etc.
- The restart file (runtpe) now an HDF5 file in MCNP6.3.
- The ptrac file and unstructured mesh model / elemental edit files all now have an HDF5 option in MCNP6.3.

Introduction of the New File Formats (2)

- XDMF visualization through openly available software (e.g., ParaView, VisIt)
  - XML-formatted ASCII XDMF file can be easily interrogated via standard text editor
  - Permits many geometric representations: structured, unstructured, polyhedral, etc.
  - Low-overhead roadmap into the aforementioned HDF5 data file(s)
- HDF5+XDMF: directly accessible data files providing immediate visualization

Mesh Tally Results Visualized via ParaView



Simple Godiva Example (1)

Include last line if running with MCNP6.3, otherwise exclude.

Listing 1: godiva.inp

```
 Godiva  Solid Bare HEU sphere  HEU-MET-FAST-001
1    100    4.7984e-02    -10     imp:n=1
2    0                     10     imp:n=0

10   so     8.7407

kcode  5000  1.0  50   250
ksrc   0 0 0
c
m100   92234.00c   4.9184e-04
       92235.00c   4.4994e-02
       92238.00c   2.4984e-03
c
fmesh4:n   geom=xyz  origin=-10 -10 -10
           imesh=10  iints=200
           jmesh=10  jints=200
           kmesh=10  kints=200
           out=xdmf              $ New out option in MCNP6.3
```

Simple Godiva Example (2)

MCNP6.2

- Execution command:

```
mcnp6 i=godiva.inp n=godiva.  tasks 8
```

- Directory listing:

```
/
 |- godiva.inp
 |- godiva.o
 |- godiva.r
 |- godiva.s
 |- godiva.msht
```

- ~ 78 M Histories / hr

MCNP6.3

- Execution command:

```
mcnp6 i=godiva.inp n=godiva.  tasks 8
```

- Directory listing:

```
/
 |- godiva.inp
 |- godiva.o
 |- godiva.r.h5
 |- godiva.s
 |- godiva.msht.xdmf
```

  - `.h5` file extension for new HDF5 files
  - `.xdmf` file extension for new XDMF files
- ~ 710 M Histories / hr

Bug Fixes

- Allow $S(\alpha, \beta)$ cross section to be applied to fissile isotopes
- Cacheing of $S(\alpha, \beta)$ cross section in a material
- Electron scattering angle distribution with multiple elements
- ***Energy deposition fixes below energy cutoff (non-n/p/e particles)***
- Various source sampling and normalization fixes
- ***Energy-dependent perturbation (PERT) error***
- Fix `mwhere` treatment on WWP card

Note that there are many more features, code enhancements, and bug fixes both within the code and the separate utilities not discussed in this workshop. The MCNP6.3 release notes exhaustively cover all of the important changes.

Energy Deposition Improvements

In MCNP6.3, there are three primary changes to energy deposition:

1. Consistent treatment of particle energy deposition as they pass through or are born below the energy cutoff
   - Particles crossing below the energy cutoff contribute the remainder of their energy to the local energy deposition tally
   - Particles born below energy cutoff contribute their energy to the local energy deposition tally
2. Fixed charged-particle TMESH tally energy deposition in a magnetic field
3. Added warning message about electron energy deposition in a magnetic field
   - Due to the electron energy straggling, the proper fix is complicated and may require some refactoring and algorithmic changes

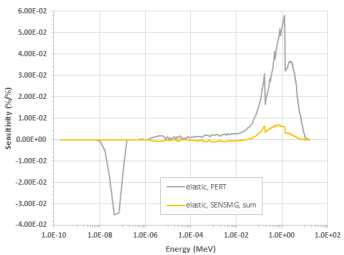
Energy-dependent PERT Card Fixed

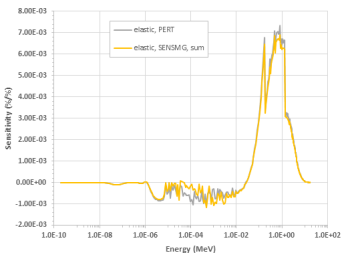
- Energy-dependent perturbations and/or sensitivities calculated with the PERT card can be significantly wrong in MCNP6.2 (fixed in MCNP6.3).

```
pert004:n cell=1 mat=2 rho=9.6E-02 rxn=2 erg=0.1 1.0 method=2
```

- At a collision point, the incident energy of a particle placed in the bank was not cached properly, making contributions to the wrong energy bin.

Before (left) and After (right) PERT Bugfix





Thanks to Jeff Favorite (XCP-7, LANL) for finding this bug and providing substantial details to make fixing this issue straightforward.

Deprecated Features

- ***FMESH output formats***
- Legacy unstructured mesh EEOUT file formats
- Embedded geometry background and matcell flexibility
- ***Legacy PTRAC file formats***
- ***PTRAC options COINC and CAP***
- Legacy unstructured mesh utilities
- MCNPUM and GMV UM file formats
- TIR, TIC, PI, MPN input cards

Covered or partially discussed in this workshop

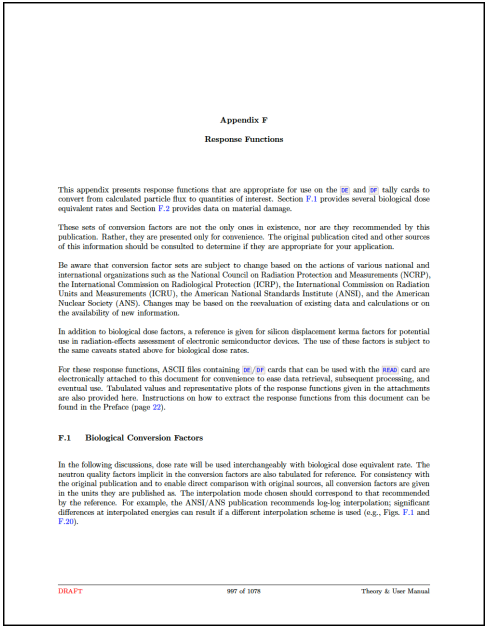
Removed Features

- Random number generator options only set through RAND card, not DBCN
- Removed HTAPE utility
- Removed MCNP_RANDOM utility
- ***Removed built-in fluence-to-dose response functions***

Covered or partially discussed in this workshop

Built-in Response Functions Extracted (1)

- Due to both copyright concerns and maintainability of the built-in fluence-to-dose response functions, the IC=10–40 built-in DE/DF functions are no longer available in the MCNP source code.
- The fluence-to-dose response functions are available in the Response Functions Appendix of the MCNP6.3 user manual.



Appendix F

Response Functions

This appendix presents response functions that are appropriate for use on the DE and DF tally cards to convert from calculated particle flux to quantities of interest. Section F.1 provides several biological dose equivalent rates and Section F.2 provides data on material damage.

These sets of conversion factors are not the only ones in existence, nor are they recommended by this publication. Rather, they are presented only for convenience. The original publication cited and other sources of this information should be consulted to determine if they are appropriate for your application.

Be aware that conversion factor sets are subject to change based on the actions of various national and international organizations such as the National Council on Radiation Protection and Measurements (NCRP), the International Commission on Radiological Protection (ICRP), the International Commission on Radiation Units and Measurements (ICRU), the American National Standards Institute (ANSI), and the American Nuclear Society (ANS). Changes may be based on the reevaluation of existing data and calculations or on the availability of new information.

In addition to biological dose factors, a reference is given for silicon displacement kerma factors for potential use in radiation-effects assessment of electronic semiconductor devices. The use of these factors is subject to the same caveats stated above for biological dose rates.

For these response functions, ASCII files containing DE/DF cards that can be used with the READ card are electronically attached to this document for convenience to ease data retrieval, subsequent processing, and eventual use. Tabulated values and representative plots of the response functions given in the attachments are also provided here. Instructions on how to extract the response functions from this document can be found in the Preface (page 22).

F.1 Biological Conversion Factors

In the following discussions, dose rate will be used interchangeably with biological dose equivalent rate. The neutron quality factors implicit in the conversion factors are also tabulated for reference. For consistency with the original publication and to enable direct comparison with original sources, all conversion factors are given in the units they are published as. The interpolation mode chosen should correspond to that recommended by the reference. For example, the ANSI/ANS publication recommends log-log interpolation; significant differences at interpolated energies can result if a different interpolation scheme is used (e.g., Figs. F.1 and F.20).

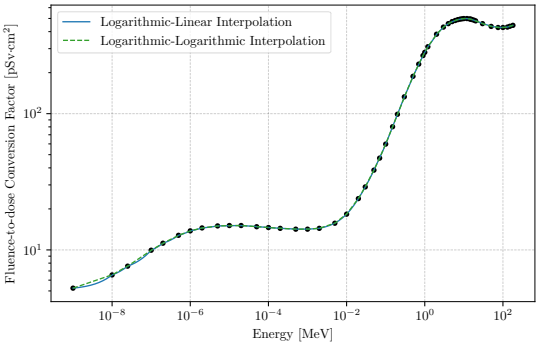
DRAFT 997 of 1078 Theory & User Manual

Built-in Response Functions Extracted (2)

- Both table listings and plots are available in the Response Functions Appendix

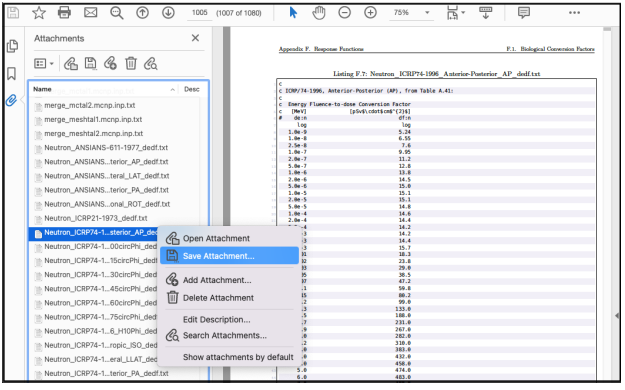
Neutron ICRP74-1996 Anterior-Posterior AP dedf Cards and Plots

```
c
c ICRP/74-1996, Anterior-Posterior (AP), from Table A.41:
c
c  Energy Fluence-to-dose Conversion Factor
c    [MeV]                    [pSv$\cdot$cm$^{2}$]
#      de:n                              df:n
         log                               log
    1.0e-9                             5.24
    1.0e-8                             6.55
    2.5e-8                              7.6
    1.0e-7                             9.95
    2.0e-7                             11.2
    5.0e-7                             12.8
    1.0e-6                             13.8
    2.0e-6                             14.5
    5.0e-6                             15.0
    1.0e-5                             15.1
    2.0e-5                             15.1
    5.0e-5                             14.8
    1.0e-4                             14.6
    2.0e-4                             14.4
    5.0e-4                             14.2
    1.0e-3                             14.2
    2.0e-3                             14.4
    5.0e-3                             15.7
      0.01                             18.3
      0.02                             23.8
      0.03                             29.0
      0.05                             38.5
      0.07                             47.2
       0.1                             59.8
      0.15                             80.2
       0.2                             99.0
       0.3                            133.0
       0.5                            188.0
       0.7                            231.0
       0.9                            267.0
       1.0                            282.0
       1.2                            310.0
       2.0                            383.0
       3.0                            432.0
       4.0                            458.0
       5.0                            474.0
       6.0                            483.0
       7.0                            490.0
       8.0                            494.0
       9.0                            497.0
      10.0                            499.0
      12.0                            499.0
      14.0                            496.0
      15.0                            494.0
      16.0                            491.0
      18.0                            486.0
      20.0                            480.0
      30.0                            458.0
      50.0                            437.0
      75.0                            429.0
     100.0                            429.0
     130.0                            432.0
     150.0                            438.0
     180.0                            445.0
c
```



Extracted Response Function Example (1)

- In addition to the discussions, tables, and plots, the fluence-to-dose response functions are included as attachments to the user manual PDF.
- Save the attached (to the user manual and this presentation) **NEUTRON_ICRP74-1996_ANTERIOR-POSTERIOR_AP_DEDF.TXT** file to use immediately as input.



Extracted Response Function Example (2)

- Can easily use READ card to include extracted DE/DF cards in problem input

Listing 2: shield.inp

```
c ### tally specification
fc2     1 m from shield, source rate = 3e8 neutrons
f2:n    10.1
fm2     3.e8          $ source strength 3.e8 n/s
c
read file=Neutron_ICRP74-1996_Anterior-Posterior_AP_dedf.txt
```

- Take care when using these attached fluence-to-dose response functions as you may want to modify which problem-specific tally you want to apply the conversion functions to.

Qt-based Geometry and Tally Plotter

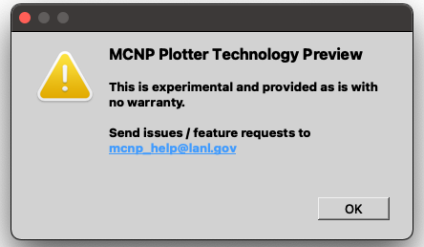
Outline

- Geometry Plotter
  - Introduction of new features
  - Brief look at each panel
  - Live demo
- Tally Plotter
  - New viewport
  - Live demo

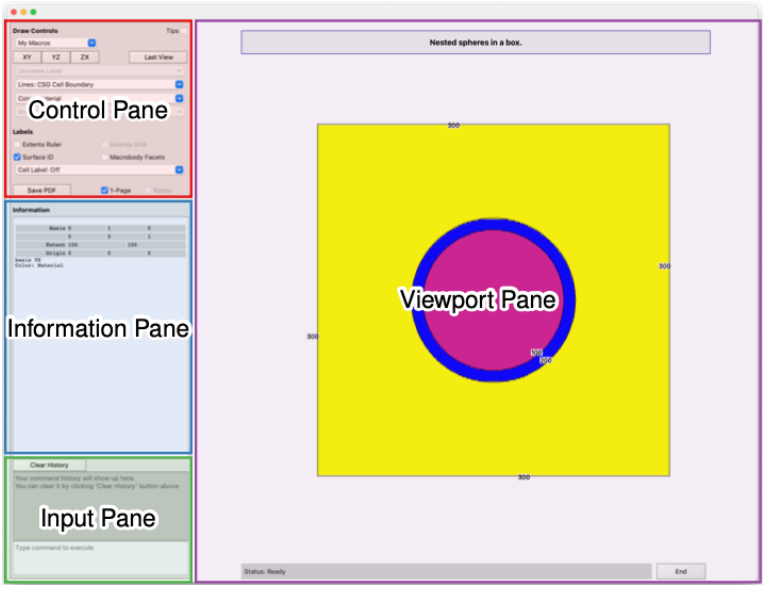
New Plotter Features

- Qt-based render window (no X-Server required)
- Modern buttons (geometry plotter)
- Command box supporting > 29 characters
- Mouse control to pan, zoom, and rotate slices
- New plotter view file formats (png, pdf)
- Macro support

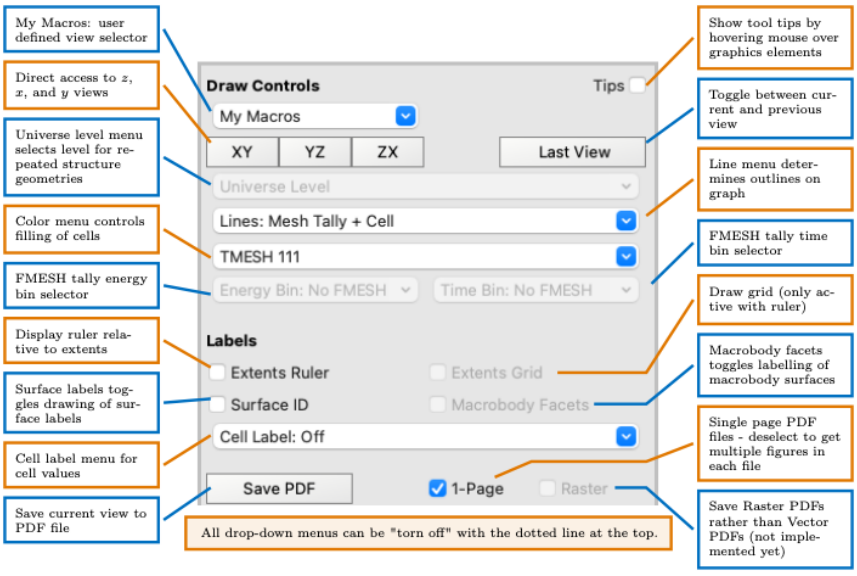
New plotter is shipping with MCNP6.3



Geometry Plotter First Look



Control Pane (1)

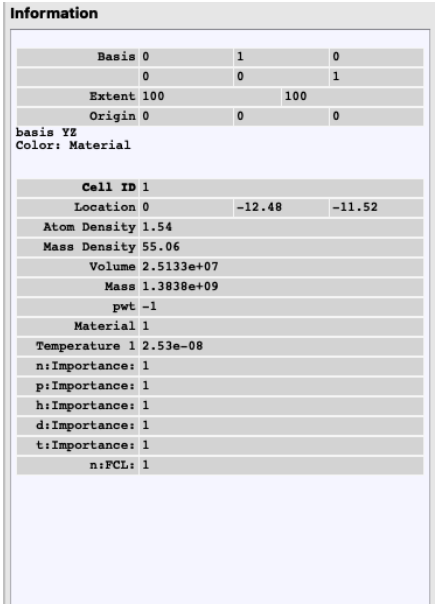


Control Pane (2)

- “My Macros” functionality
  - Similar to reading a COM file on the execution line
  - More later
- “Last View” expands on limitations of “Restore” button
  - Resets all parameter(s) to previous parameter(s) instead of a select few
- “Tips” button allows user to hover over a feature for quick help text
- “Save PDF” button
  - Available as a new command: `savepdf`
  - Raster PDF functionality to come

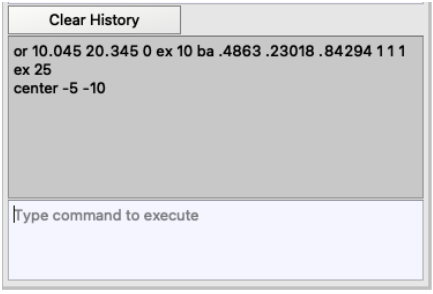
Information Pane

- Contains all information present in X-based plotter by default
  - Basis, Extent, Origin
- Click in a cell for more information on that cell.
  - Old plotter only showed what was selected in the right-hand control panel
  - Now shows all information relevant to problem
- Will also show information on an overlaid FMESH mesh
  - Runtape information, tally value/error of selected voxel, voxel indices



Input Pane

- The old “Click here or picture or menu” box only supported 29 characters of input
  - or 10.045 20.345 0 ex 10 ba .4863 .23018 .84294 1 1 1
- New input is “infinite” for all practical one-liners
  - Also supports command recall with up- and down-arrows
- Messy history can be cleared with “Clear History” button, but command recall with arrow keys still works
- Error messages are still presented in the command-prompt window



Viewport Pane

- Can now interact directly with keyboard and mouse:
  - Click and drag to pan
  - Shift-Click and drag to rotate
  - Ctrl-Click and drag to zoom in and out
  - Ctrl-[Arrow Key] to pan
    - Ctrl-Shift-[Arrow Key] to pan more
  - Ctrl-[+/-] to zoom at origin
    - Ctrl-Shift-[+/-] to zoom more
- Status bar now tells you when the plotter is working

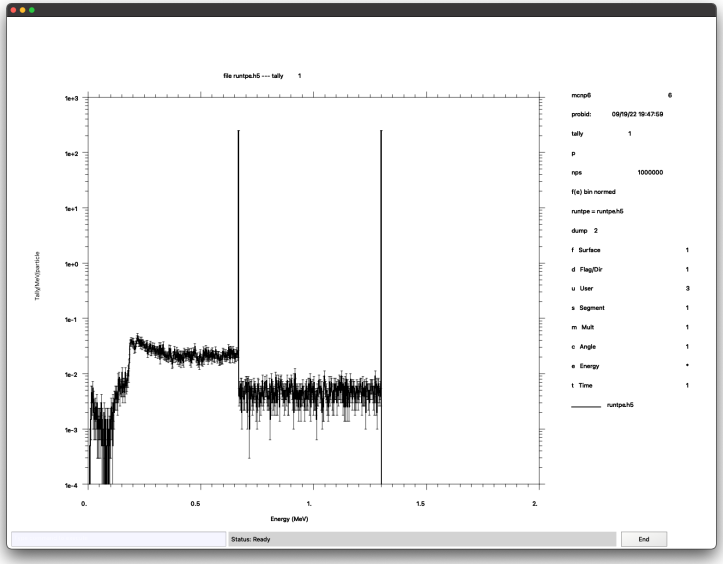
These features will be shown in the demo

Geometry Plotter Live Demo

- Show macro functionality with lost particle use-case
- Demo more complex geometry
  - Compare X-plotter speed
  - Show new `savepdf`, `savepng` commands

Tally Plotter Viewport

- Largely the same look as the old tally plotter, though modernized



Tally Plotter Live Demo

- Show classic tallies
  - Arrow key recall
- Show FMESH tally
  - Tie into geometry plotter

HDF5, MCNPTools, and New PTRAC Options

HDF5 Overview

Motivation

- MCNP output exists in many forms with various uses
  - Output file (`outp`)
    - Human-readable collection of relevant results
  - Data files (EEOUT, `mctal`, `meshtal`, `runtpe`, `ptrac`, etc.)
    - Uniquely formatted to support post-processing (plots, tables, etc.)
- Several of the data files can be written as binary
  - Fast, efficient storage
  - Non-standard formats; hard to parse (needs custom applications)
- MCNP data files are migrating toward HDF5 to eliminate this downside
  - Mesh tally (`fmesh`), `ptrac`, `runtape`, and UM output are implemented
  - Other files undergoing development and implementation

Introduction to HDF5

- HDF: Hierarchical Data Format
- Developed by The HDF Group – Non-profit organization
  - Spun off from Nat'l Center for Supercomp. Appl. at Univ. of Illinois
  - Central authority to ensure quality and prevent fragmentation
- BSD-like license, freely available, portable, numerous APIs
  - Official APIs: C, C++, Fortran, Java
  - Unofficial APIs: Julia, Matlab, Mathematica, Perl, Python, R
- Developed with speed and scalability in mind
- Binary format, which requires a program or API to read the data
- Three major objects: groups, datasets, and attributes

Groups	Containers for datasets or other groups (like a file system)
Datasets	Homogeneous n-dimensional arrays
	▸ Can contain complex objects, e.g., images
Attributes	Can be added to either groups or datasets

History of HDF5

- 1987: work to develop all-encompassing hierarchical object-oriented file
- 1990 and 1992: NSF grants provided crucial funding
  - NSF wanted to harmonize netCDF and HDF formats
  - Drove improved V&V basis
  - NASA selected HDF as its standard data and information system
- 1996: major redesign: went to current group & dataset approach
- More information in videos at: https://www.hdfgroup.org/about-us/

- HDF4 is older but actively supported
- HDF5 is current (and actively supported)
  - Attempts to address some HDF4 limitations

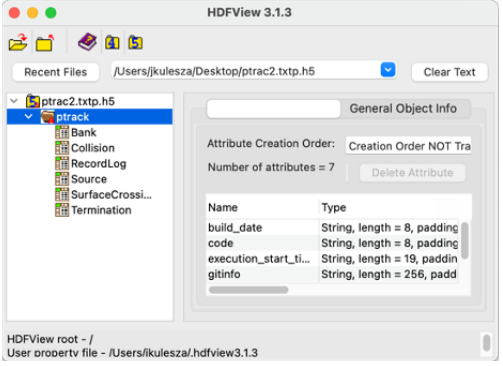
HDF5 is Binary, but Easily Interrogated

- 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

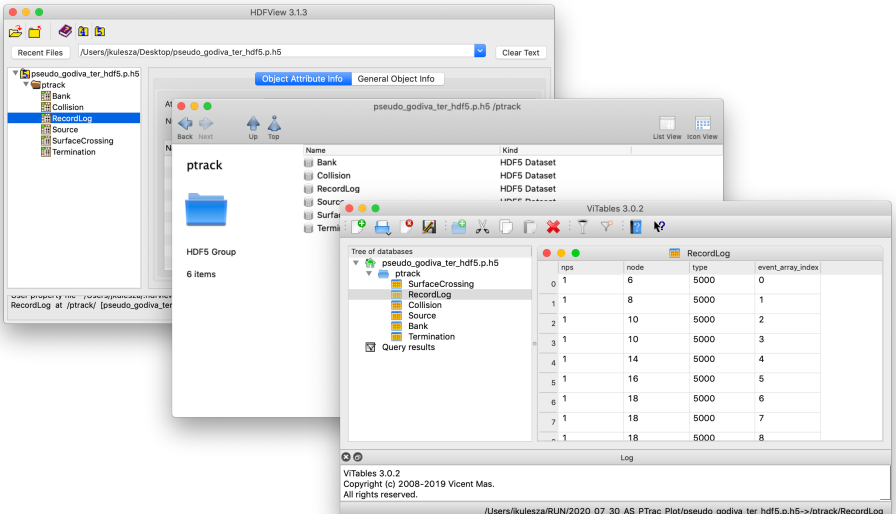
`h5ls` and `h5dump` for terminal usage

```
h5ls -r ptrac2.txtp.h5
/                        Group
/ptrack                  Group
/ptrack/Bank             Dataset {0/Inf}
/ptrack/Collision        Dataset {12920729/Inf}
/ptrack/RecordLog        Dataset {12923729/Inf}
/ptrack/Source           Dataset {3000/Inf}
/ptrack/SurfaceCrossing  Dataset {0/Inf}
/ptrack/Termination      Dataset {0/Inf}
```

`HDFView` for graphical exploration



GUI Options: HDFView, HDF Compass, and ViTables



PTRAC Overview

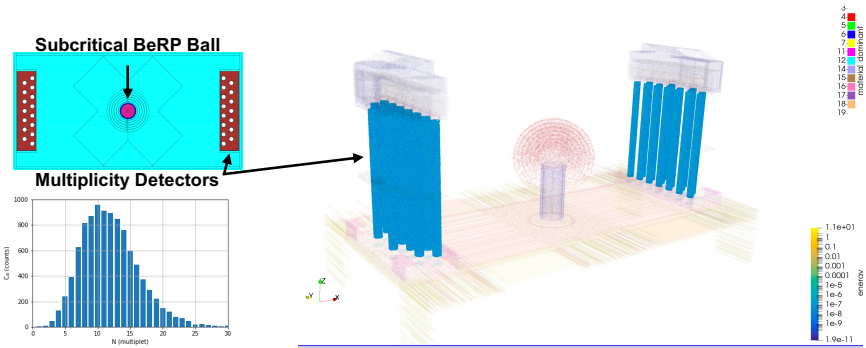
Introduction to Particle Track Output (PTRAC) (1)

- PTRAC output details the physical processes each particle underwent during the simulation, allowing for advanced post-processing
- Includes data for source, termination, collision, bank, and surface events
  - Secondary particles' events are not in chronological order
  - Reconstructed branching of histories is typically possible, but onerous
- User-specified filters control when events are written, per event or history
  - Limits the size of output files and memory usage
- Example MCNP6.2 input card
  - `PTRAC EVENT=SRC,COL,TER CELL=1 FILTER=1,14,ERG FILE=BIN`
  - Write events with energy of 1–14 MeV, for histories that traversed cell 1
- Example MCNP6.3 input card
  - `PTRAC EVENT=SRC,COL,TER CELL=1 FILTER=1,14,ERG FILE=HDF5 FLUSHNPS=1E5`
  - Same behavior as before, different output file format

Introduction to Particle Track Output (PTRAC) (2)

- 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 [5]
  - Subcritical multiplicity experiments

Subcritical Multiplication Analysis and Visualization



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

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, blue = new in MCNP6.3

MCNPTools PTRAC Interface

MCNPTools Background

- MCNPTools was born out the continual need to process MCNP outputs
- A collection of Python and C++ post-processing software
  - Version 3.8 released with MCNP6.2
  - **Latest version now available as open source at https://github.com/lanl/mcnptools**
- The MCNPTools library provides easy access to events in simulation order (HDF5 and legacy)
- While file formats in MCNP6 may change over time, the MCNPTools interface will require minimal changes
- Example processing:

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
```

MCNPTools `Ptrac` Overview

- The `Ptrac` class
  - Manages data for MCNP's ptrac files
  - Can handle all file format versions (assumes binary as default)
- The `PtracHistory` class manages data for an individual history
- The `PtracNps` class handles the starting information for a history
- The `PtracEvent` class handles information for particle events

MCNPTools `Ptrac` Class Accessors

- `ReadHistories(NUM)`
  - Reads `NUM` histories and all associated events into memory as a list
  - Allows “chunk” processing of a file without filling memory

MCNPTools `PtracHistory` Class Accessors

- `GetNPS()`
  - Return a `PtracNPS` class to the history (`NPS`) information
- `GetNumEvents()`
  - Return the number of events recorded for the history
- `GetEvents(NUM)`
  - Return a `PtracEvent` class to the history's `NUM`th event

MCNPTools `PtracNps` Class Accessors

- `NPS()`
  - Return the history number
- `Cell()*`
  - Return the filter cell from `CELL` keyword (if present)
- `Surface()*`
  - Return the filtering surface from `SURFACE` keyword (if present)
- `Tally()`
  - Return the filter tally from `TALLY` keyword (if present)
- `Value()`
  - Return the tally score from `TALLY` keyword (if present)

*Not currently present in the HDF5 output file

MCNPTools `PtracEvent` Class Accessors

- `Type()`
  - Return the event type
- `Has(DATA)`
  - Return `TRUE` if the event has type `DATA`
- `Get(DATA)`
  - Return the value of the data type `DATA`

New Parallel PTRAC

HDF5-formatted PTRAC (1)

- 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-formatted PTRAC (2)

- HDF5 PTRAC files can be produced with MPI, threads, or both
  - MPI-parallel PTRAC requires an MPI enabled installation of HDF5
  - Thread-parallel PTRAC (tasks option) works with any HDF5 build
- Each process buffers data into memory over multiple histories
  - Periodically all processes write data to the file
  - Provides greater speed than writing semicontinuously
- The buffers' memory usage can be **very large**
  - A std::bad_alloc error will appear if memory is exhausted
    - Memory swapping may occur instead
  - Prevented with the FLUSHNPS option detailed on next slide

HDF5-formatted PTRAC (3)

- FLUSHNPS controls how much 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 PTRAC memory usage is printed by MCNP6
  - Only includes PTRAC memory usage, so must balance with problem size
  - Writes may occur more frequently from other rendezvous
- As a rule of thumb (for current hardware) use FLUSHNPS = 100000
  - Check the memory usage is less than 1–2 GB per computational resource
  - Higher memory usage than this will not substantially improve efficiency
- **When in doubt, write more often than necessary**

HDF5-formatted PTRAC (4)

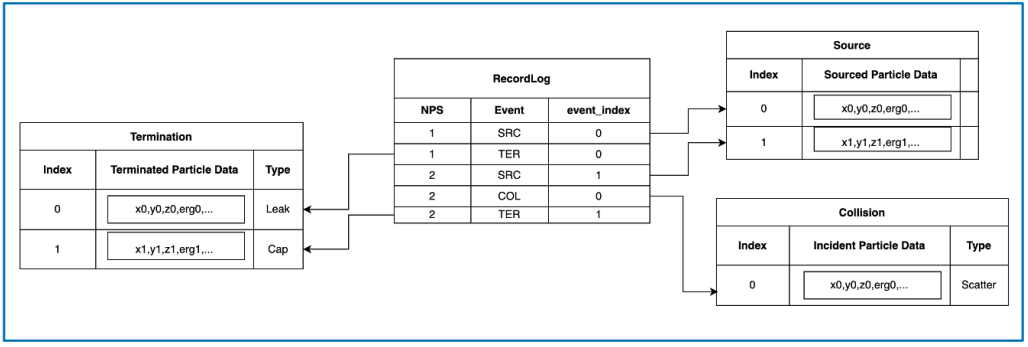
- Several important MPI-parallel HDF5 considerations
  - Intended for use on parallel file systems, e.g., Lustre
  - Will be **slow or hang** on most Network File Systems (NFS)
    - The MCNP code **cannot** detect or predict this misbehavior
- Can use thread-based parallelism on any file system
- **Always try a quick initial run to verify that a file is written correctly**

Hierarchical and Structured Layout

- The data for each type of event are a separate dataset array
  - Have direct access to all events, for each event category
- Each entry in a dataset is a compound data type
  - Contains all particle data: x, y, z, energy, etc.
  - Contains event specific data: collision type, bank type, etc.

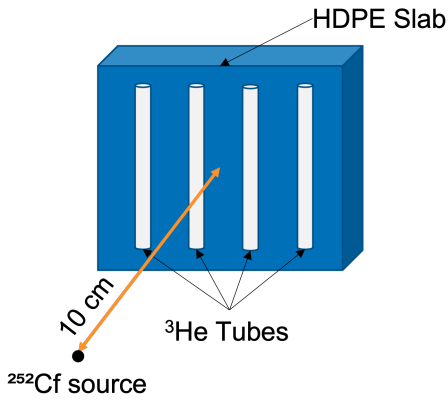
Event Ordering from the `RecordLog`

- The `RecordLog` dataset
  - Provides the event-by-event order of entries in the event arrays
  - Uses the history identifier `NPS`
  - Similar to foreign keys of relational databases



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

Safeguards Example for Simple Neutron Detector Coincident Counting (4)

Let's take a look at the attached **ptrac_mcnptools.py.txt** script

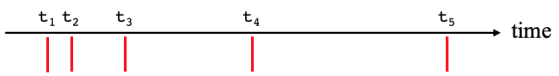
Python

```
max_counts = 0
count_histogram = np.zeros(100)

# Open file and then read a chunk of 1000 histories
ptrac_handle = Ptrac("ptrac.p.h5", Ptrac.HDF5_PTRAC)
ptrac_hists = ptrac_handle.ReadHistories(1000)

while ptrac_hists:

    # Iterate through each individual history
    for hist in ptrac_hists:

        # Call time filter function to get sorted list of capture times
        times = filter_history_times(hist, cells=[21, 22, 23, 24], zas=[2003], rxns=[101])

        # Call histogram function to process capture times
        counts = histogram_time_gate(times, predelay_time=500, gate_width=10000)
        max_counts = max(max_counts, len(counts))

        # Accumulate history histogram into total histogram
        for icount, count in enumerate(counts):
            count_histogram[icount] += count

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

# Print total histogram
for icount, count in enumerate(count_histogram[:max_counts]):
    print(f"Count Outcome = {icount}, Total Counts = {int(count)}")
if max_counts == 0 and count_histogram[0] == 0:
    print("Zero Capture Events Found!!!")

# Plot total histogram
# ...
# See the rest in the ptrac3_mcnptools.py.txt script
# ...
```

Safeguards Example for Simple Neutron Detector Coincident Counting (5)

Python

```
# Call time filter function to get sorted list of capture times
times = filter_history_times(hist, cells=[21, 22, 23, 24], zas=[2003], rxns=[101])

# ===============================================================================
def filter_history_times(history, cells, zas, rxns):
    """ Function to process a Ptrac history into 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 cells and za in zas and rxn in rxns:
                times.append(event.Get(Ptrac.TIME))

    return sorted(times)
```

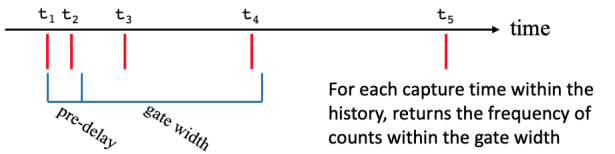


Returns list of sorted capture times: `[t1,t2,t3,t4,t5]`

Safeguards Example for Simple Neutron Detector Coincident Counting (6)

Python

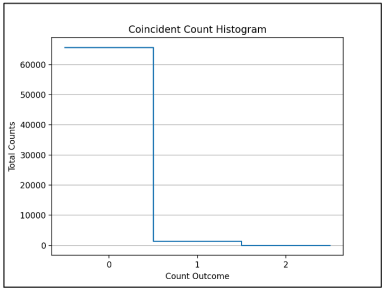
```
# Call histogram function to process capture times
counts = histogram_time_gate(times, predelay_time=500, gate_width=10000)

# ============================================================================
def histogram_time_gate(times, predelay_time, gate_width):
    """ Function to process a list of times into a histogram of coincident counts. """

    number_times = len(times)

    counts = np.zeros(number_times)
    for itime in range(number_times):
        # Time when gate is opened
        t0 = times[itime]

        count = 0
        for jtime in range(itime + 1, number_times):
            # Next time before pre-delay time... no count
            if times[jtime] < t0 + predelay_time:
                pass
            # Next time within pre-delay and gate time... count
            elif times[jtime] <= t0 + predelay_time + gate_width:
                count += 1
            # Next time after pre-delay and gate time... no count
            else:
                break
        counts[count] += 1

    return counts
```



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

Safeguards Example for Simple Neutron Detector Coincident Counting (7)

- Executing MCNP6.3
  - `mcnp6 i=safeguards.inp ptrac=ptrac.p.h5 tasks 8`
- And then executing the provided Python script
  - `python3 ptrac_mcnptools.py.txt`
- Results in:
  - `python3 ptrac_mcnptools.py.txt`
  - `Count Outcome = 0, Total Counts = 65662`
  - `Count Outcome = 1, Total Counts = 1348`
  - `Count Outcome = 2, Total Counts = 19`



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)

New Tally Features

FMESH Upgrades and HDF5+XDMF

Introduction

Since 6.2, `FMESH` has undergone a substantial revision in capabilities.

- The default `FMESH` configuration was heavily optimized.
- 3 new tally backends were added for various needs (mainly, larger tallies).
- MESHTAL is deprecated and replaced with an HDF5 + XDMF output format.
  (This format allows for much faster and easier postprocessing and analysis)

Tally Algorithms

It started as a side project - can we use MPI remote memory access to scale further than ever before?

Implemented 4 algorithms:

History	Basic history statistics without optimization.
Fast History	A tuned version of 6.2's `FMESH` algorithm, tracks changed indices to reduce memory bandwidth usage.
Batch	Threads share a tally array, so memory usage is reduced.
Batch RMA	The Batch algorithm, but using MPI-3 RMA to distribute tallies over all MPI ranks.

Infrastructure Changes

The MCNP code didn't support batch statistics in any way. A number of changes had to be made:

- `NPS` now has a batch size option.
- When any batch tallies are enabled, `KCODE` will resample the fission bank to a fixed size.

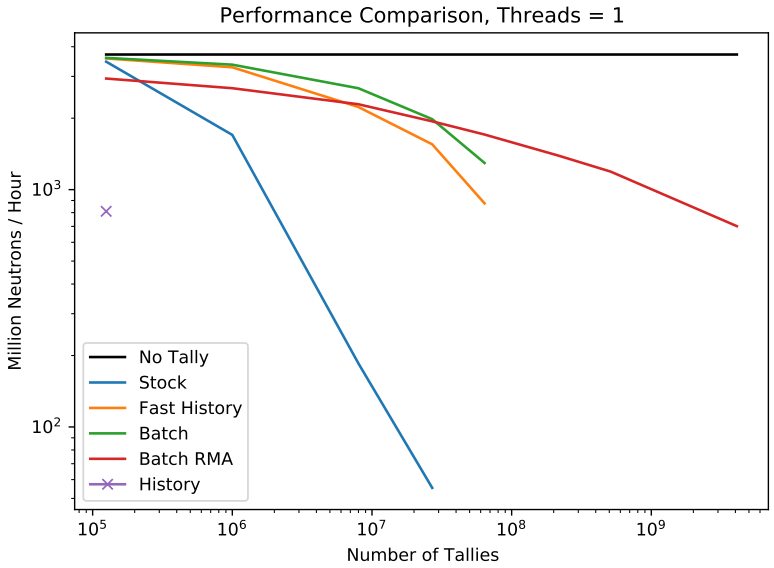
This means the RNG sequence will change if batch tallies are added!

Performance

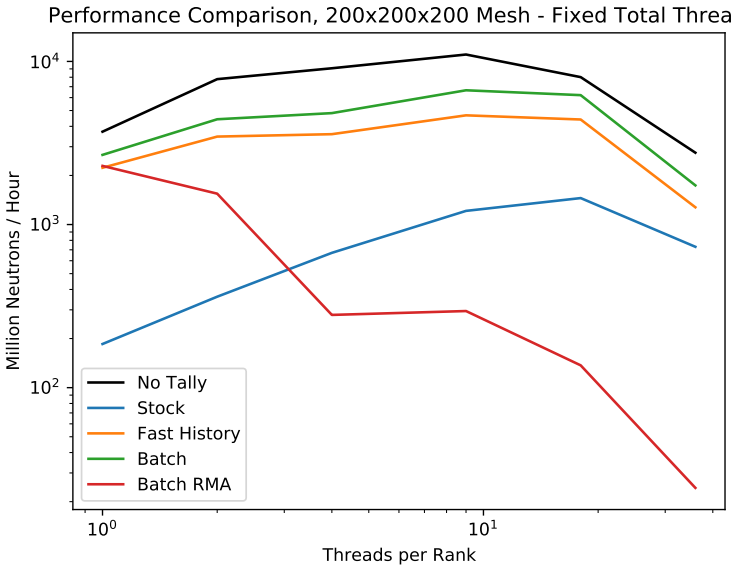
- Tested on a k-eigenvalue problem with a 10-cm, 10-g/cc ball of ^{235}U.
- Maximizes the effect of tally performance on the problem.
- Mesh was scaled from $50 \times 50 \times 50$ to $1600 \times 1600 \times 1600$
- Neutrons/hr and memory usage tallied
- Tested on 6 nodes of a cluster with 2 sockets, 18 cores each, 128 GB memory.
- Ran combinations of MPI, OpenMP.

Note: OpenMPI 3.1.6 + Omni-Path does not support `MPI_THREAD_MULTIPLE`, so threading performance is poor for Batch RMA.

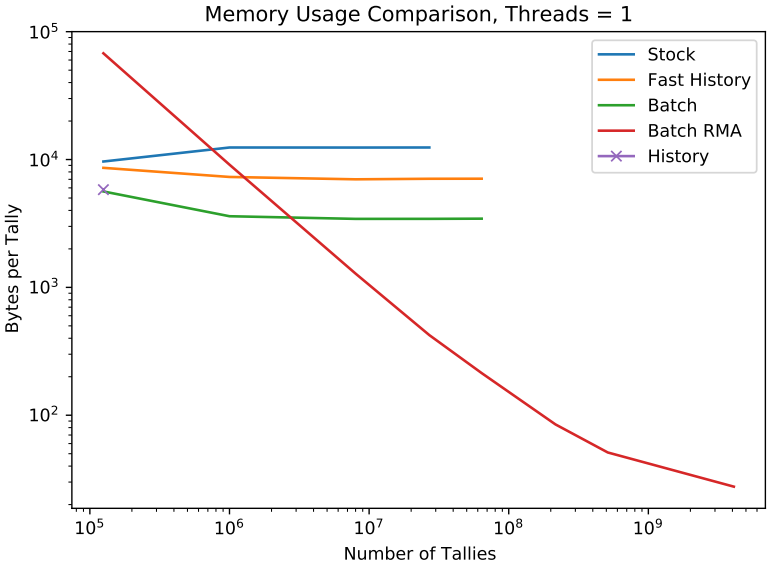
Performance



Performance

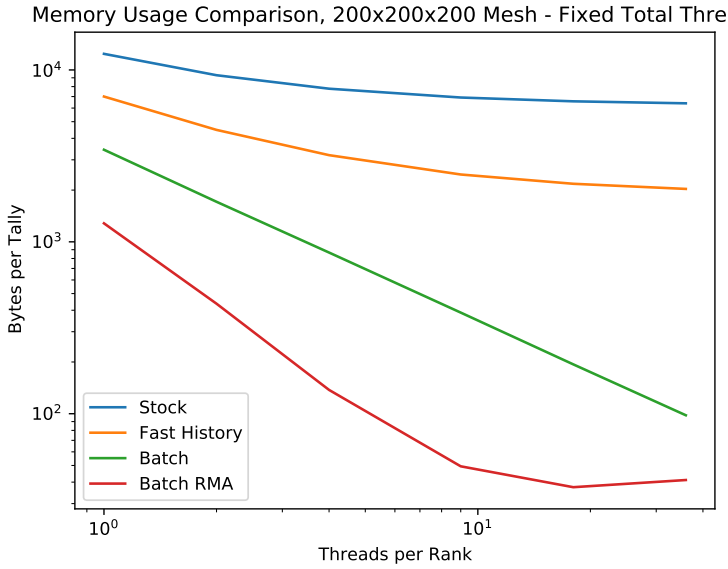


Memory



Batch RMA has high overhead that dissipates for large problems.

Memory


Memory Usage Comparison, 200x200x200 Mesh - Fixed Total Threads
Bytes per Tally
10^4
10^3
10^2
10^0
10^1
Threads per Rank
Stock
Fast History
Batch
Batch RMA

File Formats

Previous versions of the MCNP code used the MESHTAL format:

- ASCII output results in large file sizes.
- The binary to ASCII conversion was generally slow.
- It is tricky to bring into other tools (needs a processing script).

Version 6.3 uses HDF5 + XDMF:

- Binary file format for smaller sizes and faster IO.
- Trivial to load into ParaView, VisIt[1], Python, etc.
- (Optional) parallel HDF5 for even faster performance.

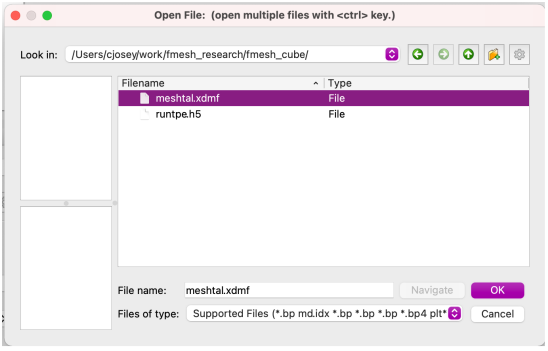
[1] Note that VisIt uses HDF5 1.8 at the time of this writing. MCNP outputs files that use the 1.10 format. Future versions of VisIt will use 1.10+. For now, `h5repack` can be used to convert to a 1.8 file.

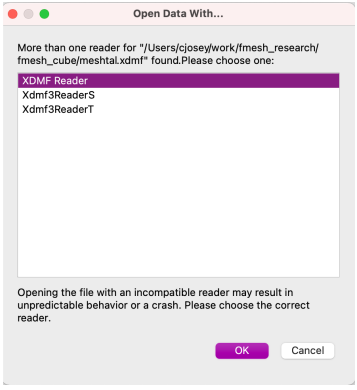
File IO Performance

216 million cell mesh, Lustre filesystem, 8 stripes, 1M stripe size, 8 MPI Ranks:

Method	Time (s)	File Size
MESHTAL	617.5	12 GB
HDF5 + XDMF	18.1	3.2 GB (in runtape)
Parallel HDF5 + XDMF	7.5	3.2 GB (in runtape)

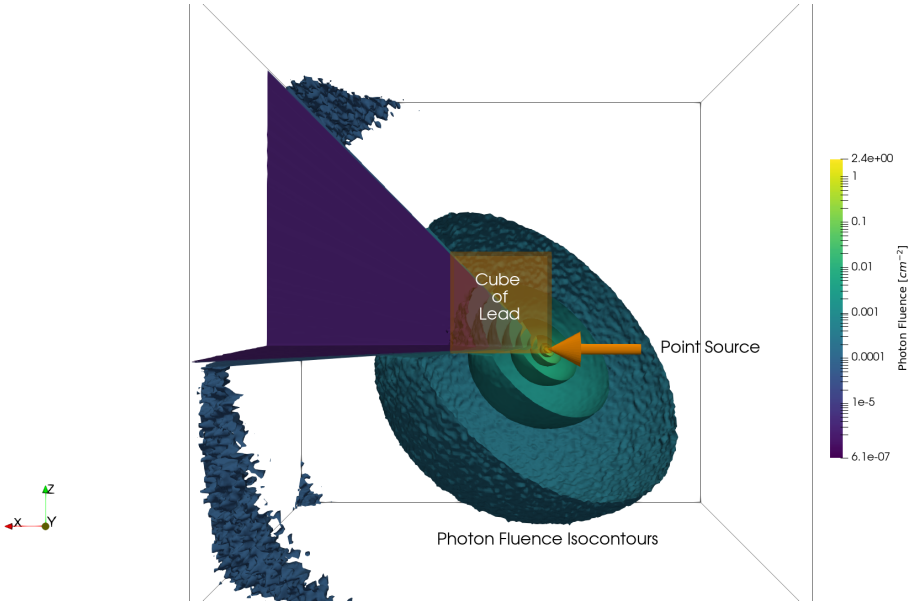
ParaView Example





Make sure to open with “XDMF Reader”, which is the reader for XDMF version 2 files.

ParaView Example - Point Source on Cube Corner



Python Example

Listing 3: Python 3.6+ Example for Reading FMESH

```python
import h5py
import numpy

def read_fmesh(filename, tally_id):
    with h5py.File(filename, 'r') as handle:
        group = handle[f"/results/mesh_tally/mesh_tally_{tally_id}"]

        data = {}
        # Transpose converts indices to x, y, z, e, t
        data["mean"] = numpy.transpose(group["mean"][()])
        data["relative_standard_error"] = \
            numpy.transpose(group["relative_standard_error"][()])
        data["grid_x"] = group["grid_x"][()]
        data["grid_y"] = group["grid_y"][()]
        data["grid_z"] = group["grid_z"][()]
        data["grid_energy"] = group["grid_energy"][()]
        data["grid_time"] = group["grid_time"][()]

        return data

data = read_fmesh("runtpe.h5", 4)
```

By slicing `group[‘‘mean’’]` instead of using `[()]`, one can load a portion into memory without loading all of it.

Summary

- New FMESH outperforms 6.2's in most workloads.
- New modes allow for much lower memory usage for large problems.
- File formats are fast and easy to work with.

In the future, we expect to extend this capability to more parts of the MCNP code.

Special Tally Treatments for Multigroup Cross Sections

Motivation

- Historically, MCNP development has not been focused on nuclear reactor applications.
- Recent institutional investiments through the LANL Laboratory Directed Research & Development (LDRD) Program have focused on developing capabilities for nuclear reactor applications.
- **A new special tally treatment for multigroup cross section calculations is in production for MCNP6.3**
- Work is continuing in this area, with a currect focus on improved tracking algorithms (i.e., Delta tracking), improved energy deposition and burn-up/activation tallies, and new tooling to support efficient workflows for nuclear reactor applications
  - Note that several of these newly developed tools and/or improved capabilities will likely have an impact outside of nuclear reactor applications (e.g., radiation protection and shielding, criticality safety)

Multigroup Cross Section Tally Options

Four new tally special treatment options (FT card) have been added to assist with reactor analyses:

- SPM Collision exit energy-angle scatter probability matrices
- MGC Flux weighted multigroup cross sections
- FNS Induced fission neutron spectra
- LCS Legendre coefficients for scatter reactions

These new multigroup tally capabilities have been thoroughly described and verified via code-to-code comparisons [2].

Flux-weighted Multigroup Cross Sections

`FTn MGC fg`

`MGC` Flux weighted multigroup cross sections

`fg` Flag for microscopic (barns) or macroscopic (1/cm) cross section calculation

Description of the Multiplier Bins for the `MGC FT` Option.

Bin #	Units	Values
1	$n/(cm^2 \cdot s)$	Flux (used as a divisor for the other bins)
2	sh/cm	Inverse velocity
3	barns	Total cross section
4	barns	Absorption cross section
5	barns	Fission cross section
6	barns	Total or prompt fission production cross section
7	barns	Delayed fission production cross section
8	barns	Fission heat production cross section
9	barns	Capture cross section (Absorption + Fission)
10	barns	Scatter cross section [Total − (Absorption + Fission)]

Flux-weighted Scattering Matrices and Fission Spectra

Multigroup scattering matrix options

`FTn SPM na` (cosine-binned scattering matrices)

- `SPM` Collision exit energy-angle scatter probability matrices
- `na` Integer number of equally-spaced cosine bins

`FTn LCS lo` (Legendre coefficient scattering matrices)

- `LCS` Legendre coefficients for scatter reactions
- `lo` Integer number of maximum Legendre scattering order

Multigroup fission energy spectra

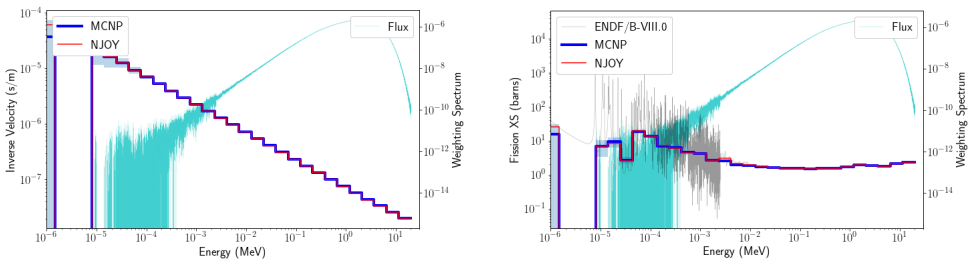
`FTn FNS nt`

- `FNS` Induced fission neutron spectra
- `nt` Integer number of delayed neutron time bins
  - If `nt` is not specified, then a `T` card needs to be used to specify time binning to separate various prompt and delayed neutrons emitted from fission.

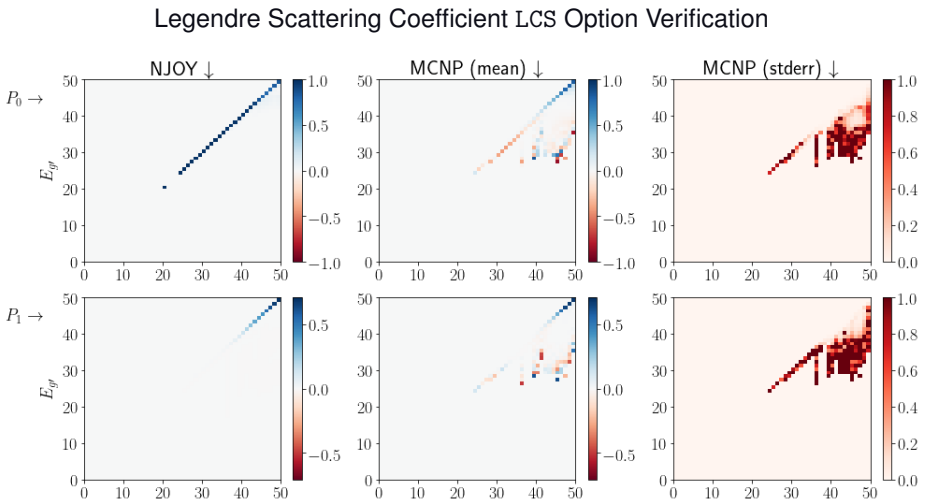
Code-to-code Verification Efforts (1)

- Compared new multigroup special treatments to those produced using NJOY
  - Calculated a fine-group energy weighting spectrum with MCNP
  - Inserted the weighting spectrum into NJOY
  - Compared NJOY multigroup data to MCNP multigroup data
- Compared to other Monte Carlo codes that calculate multigroup cross sections

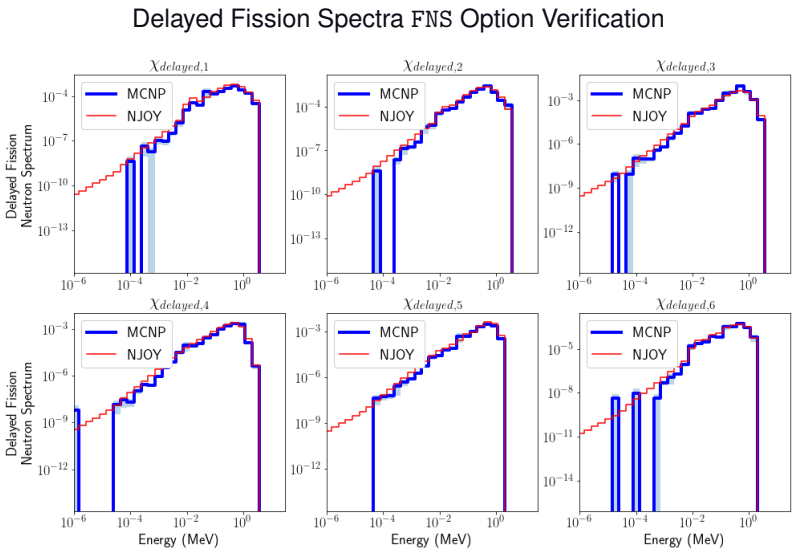
Multigroup Cross Section `MGC` Option Verification



Code-to-code Verification Efforts (2)



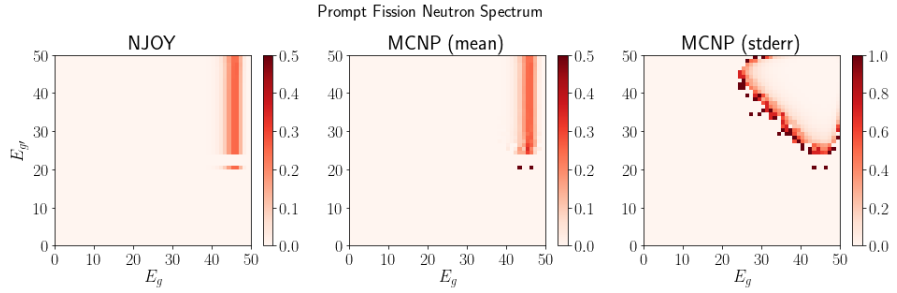
Code-to-code Verification Efforts (3)



Summary

- The `SPM` and `LCS` options were compared to each other for internal consistency
- Besides any issues using NJOY with a fine-enough weighting spectrum and dealing with the statistical noise in the Monte Carlo tallies, everything looks good between NJOY and MCNP
- Some reactor pin-cell-like problems were used to compare to multigroup capabilities in other Monte Carlo codes

Prompt Fission Spectra `FNS` Option Verification



Unstructured Mesh Improvements

Overview of Unstructured Mesh in MCNP6.3

- Significant refactoring and modernization work done to clean-up existing and unused code paths
- Added mesh element quality assessment to the MCNP6.3 code
- Additional regression, verification, and validation testing with additional comment, warning, and fatal error messages
- New HDF5 input format option and HDF5+XDMF elemental edit output format option
  - The Abaqus input format can be used to produce the new HDF5 input format with the MCNP input (`mcnp6 i ...`) processing option
  - The elemental edit HDF5+XDMF output builds on top of the input HDF5 format such that it can be used as input for an initial calculation or for a restart calculation
- Reduced mesh input processing times and memory consumption
- Deprecated legacy UM Fortran utilities in favor of new Python-based scripts

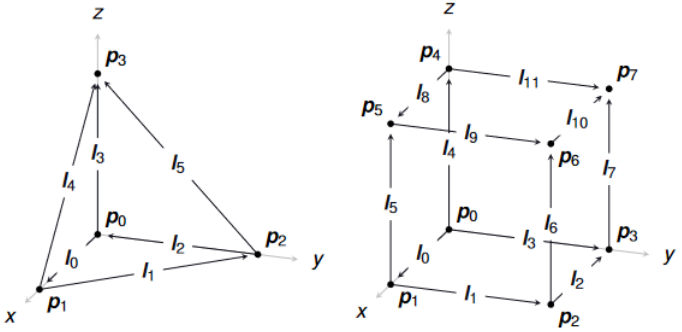
UM Element Quality Assessment

Mesh Elemental Quality Assessment

- Recent review MCNP UM quality assessment and reporting work [6]
- Detailed discussion on calculating element quality [7, 8]
- Mesh-element quality is a thoroughly studied topic, e.g., [7, 9–11]
  - Also: International Meshing Roundtable [https://imr.sandia.gov]
  - Origins in mesh generation: calculate metrics and remesh toward optima
- The MCNP um_pre_op utility [12] has an “element checker”
  - Evaluates elemental Jacobian matrix determinant: go/no-go criterion
  - Must be run/inspected manually by a user; not implicitly in workflow
  - Invoked as, e.g., `um_pre_op -ec -o inp1034.ec.out um1034.abaq`
  - Looking for final line: “Total number of failed elements: 0”
- MCNP6.3 Advances in mesh quality assessment
  - Quality metrics implemented for MCNP UM element types
    - Reproduced um_pre_op capability
    - Most-commonly used elements emphasized
    - Default enabled, opt-out available via embed card `elementchk` toggle

Approach to UM Quality Assessment (1)

- Start by focusing on linear tetrahedra and hexahedra
  - Most commonly used elements by MCNP UM practitioners
  - Most thoroughly studied elements by mesh-quality community
  - Identify metrics that can be characterized versus recommended ranges
  - All metrics exist on continua (i.e., no discrete and/or Boolean metrics)
  - Recommended ranges usually support FEA mesh-quality needs
  - This may lead to overly conservative recommended min/max values
  - Most consolidated source identified: Verdict library



Approach to UM Quality Assessment (2)

- Provide enough information to know if more information is needed
  - Provide information implicitly; don't require more workflow tasks
  - Ability to opt out in case mesh is "known to be good"
  - Elemental quality metric distribution information (a balancing act)
  - Number of elements inside / outside recommended ranges
  - Ample warning that more attention may be needed
  - Conservatism, until better guidance is available, is good
  - Education on how to learn more about mesh quality
- Warnings and Output Tables
  - Functional approach: warnings (below) and output tables
    - Warnings in standard output prompt user to inspect output tables
    - Output tables inform whether additional, external, study is necessary
  - (Conservative) warnings sent to standard output and output file...

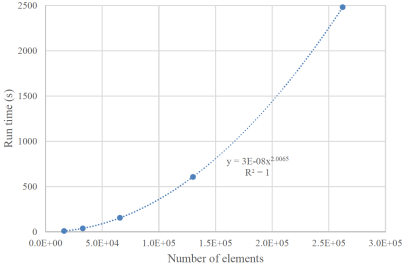
```
warning.  at least one element Aspect Frobenius outside recommended range.
warning.  at least one element Min. Dihedral Ang. outside recommended range.
warning.  at least one element Scaled Jacobian outside recommended range.
```

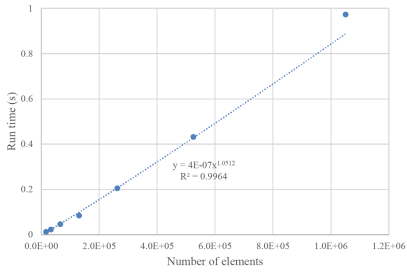
UM Performance Improvements

Reduction in UM Initialization Computational Time

- For efficient particle tracking, the UM input processing includes a step to store neighbor-to-neighbor element information
- In MCNP6.2 this step is $O(N^2)$ and in MCNP6.3 this step is $O(N)$
- For large UM models, this is a significant improvement
  - For example, a city-sized UM model with 700k+ linear hexahedral element model had a 20x speedup in input processing time

UM Input Processing Times for MCNP6.2 (left) vs. MCNP6.3 (right)





Reduction in UM Memory Consumption

- In MCNP6.2, UM models with many parts that have a large variance in the number elements can lead to excessive amounts of unnecessary memory allocated
- This can be especially important when running MPI parallel calculations with a large number of ranks
- The example below is using an UM model of the ORNL PCA Problem

UM Memory Usage for MCNP6.2 (Devel) vs. MCNP6.3 (Mem)

Branch	Case	Mesh Init Time	Memory reported by top	RSS reported by top	Actual Savings	Expected Savings
Devel	1x1	49s	3.3 GB	2.8 GB	**1.7 GB**	**1.4 GB**
Mem	1x1	49s	1.6 GB	1.2 GB		
Devel	1x36	22s	6.0 GB	3.1 GB	**1.3 GB**	**1.4GB**
Mem	1x36	22s	4.7 GB	1.9 GB		
Devel	144x1	130s	416 GB	332 GB	**192 GB**	**201 GB**
Mem	144x1	37s	224 GB	144 GB		
Devel	72x1	70s	199 GB	161 GB	**94 GB**	**101 GB**
Mem	72x1	31s	105 GB	70 GB		
Devel	72x2	70s	193 GB	162 GB	**119 GB**	**101 GB**
Mem	72x2	28s	74. GB	27 GB		

UM Validation

MCNP Unstructured Mesh Geometry

- The MCNP unstructured mesh (UM) feature was implemented to allow MCNP to read in an Abaqus mesh file, generated by Abaqus, a multiphysics and finite element analysis code.
- The use of a mesh in MCNP allows a multiphysics code such as Abaqus to be used in tandem to perform neutronics analysis with MCNP and other analyses such as heat transfer with a separate code.
- Mesh Quality is important
  - Mass/volume may not be preserved
  - Especially important for criticality calculations
- It is possible to generate a mesh, which reflects the geometry adequately for most purposes, and yet does not properly preserve mass and/or volume to the degree necessary for correct alidation efforts using criticality safety benchmarks and shielding benchmarks.

Shielding Benchmark Example: Oktavian Experiments

- The Oktavian suite of shielding experiments consist of inner and outer spherical steel shells encasing some shielding sample.
- The experiments used a D-T fusion source at the center of the shield, with an aperture leading to a hollow interior containing a tritium target. A deuteron beam was fired through the aperture to strike the target and cause fusion.

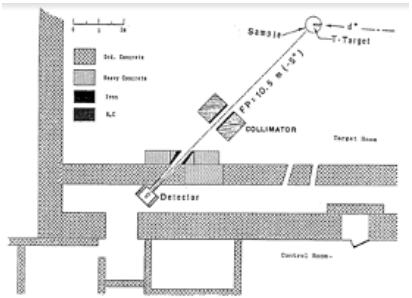


Fig. 1. Experimental arrangement in the OKTAVIAN Facility.

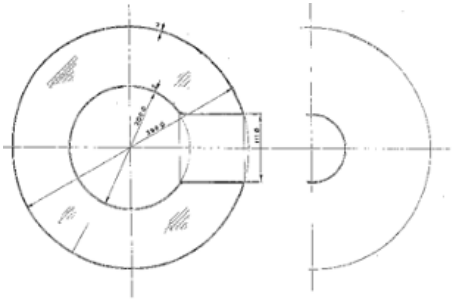
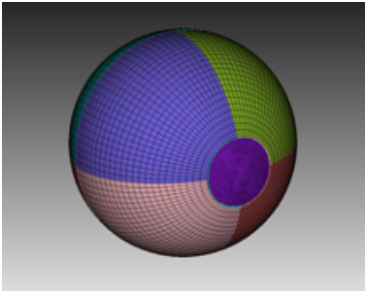
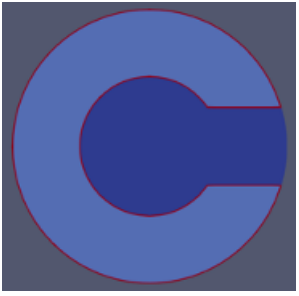


Fig. 2. 40 cm diameter vessel (Type-II).

Initial Oktavian Meshing

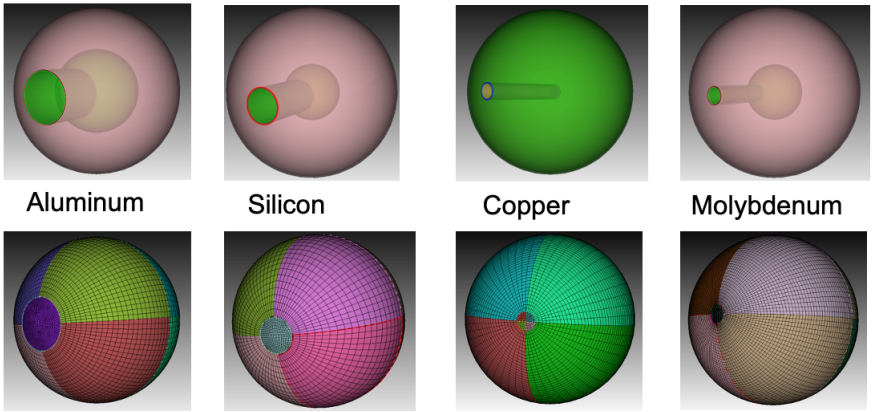
- In general, hex meshing can be difficult, requiring a geometry to be split into simpler components to be meshed.
- CUBIT was used for this task, a meshing software developed by Sandia National Labs. Requires a model to be split into pieces that can be “swept”, a meshing algorithm that CUBIT employs.
- The method used for the Oktavians was to cut a core out of the center using the cylindrical aperture as a guide. The remaining spherical components could then be cut into eighths and the entire model then meshed.





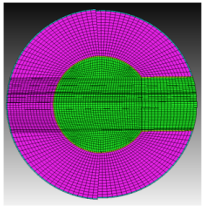
UM vs. CSG Oktavian Verification Work

- Hex meshed four Oktavian models using CUBIT for processing in MCNP
- Repeated using a tet mesh, allowing for an easier meshing process, but requiring greater computational time
- Both meshing methods to be compared to CSG models from MCNP VERIFICATION_SHLD_SVDM

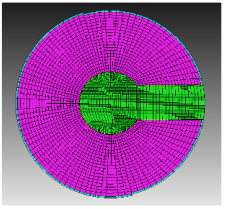


UM Oktavian Mesh Quality and Size

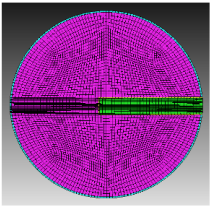
	Aluminum	Silicon	Copper	Molybdenum
Volume Difference (%)	0.13	0.21	0.27	0.07
Number of Elements	195848	111200	93788	702800
Average Quality	0.85	0.84	0.70	0.80



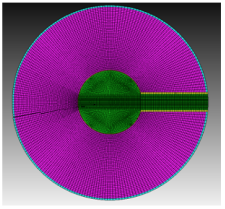
Aluminum



Silicon



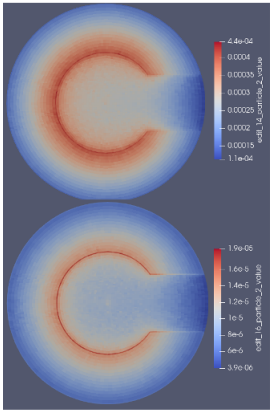
Copper



Molybdenum

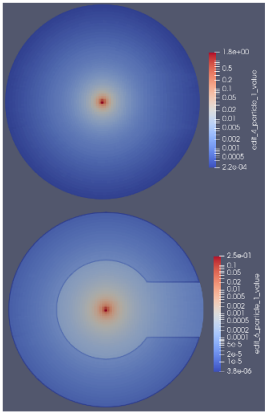
UM Oktavian Paraview Visualization

- Mesh geometries allow the placement of elemental edits on meshes, functioning similarly to a tally
- HDF5 elemental edit output (eeout) files created by MCNP6.3 allow results to be viewed using a program such as Paraview



Photon Flux

Photon Energy Deposition

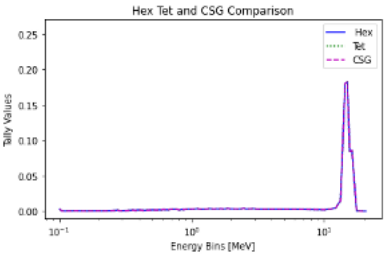


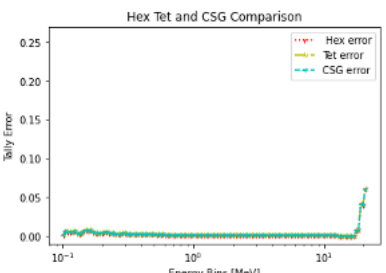
Neutron Flux

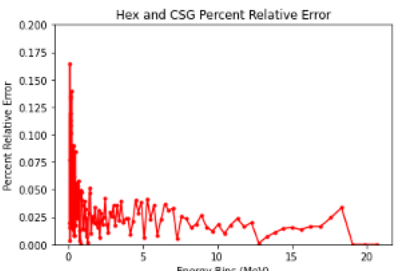
Neutron Energy Deposition

Aluminum Oktavian UM vs. CSG Comparison

- Plots comparing the F1 tally results from the hex, tet, and CSG models, as well as comparing the error of these three tally results.
- The graph in red shows the percent relative error between the hex mesh and CSG models.







UM Verification and Validation Summary

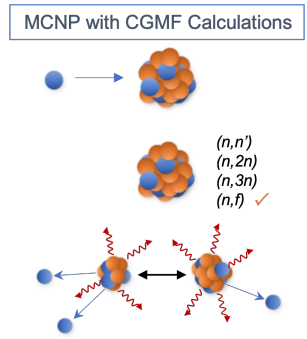
- In the past couple of years, a dedicated UM verification and validation effort has been ongoing
- Focused on CSG vs. UM comparisons to ultimately understand the mesh quality requirements to obtain comparable results to a proper CSG benchmark model
- Both criticality safety benchmarks (not shown here) and shielding benchmarks have been studied
- UM V&V will continue and ultimately become part of the `vnvstats` efforts in the future

Physics and Data Changes

CGMF 1.1.1 Upgrade

Motivation for Correlated Fission in MCNP6

- Limitations
  - Expected secondary production of particles
    - Good for criticality safety, shielding, reactor physics applications, etc.
    - Cannot correlate incident reaction with secondary production
  - Cannot perform time-coincident detector response calculations
- Previous workarounds
  - Sampling P(n) in MCNP
  - LLNL fission library
  - Detector response simulations in MCNPX-PoliMi

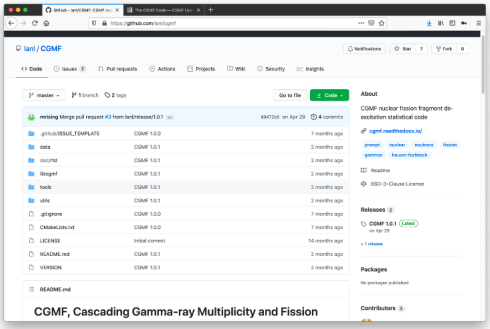


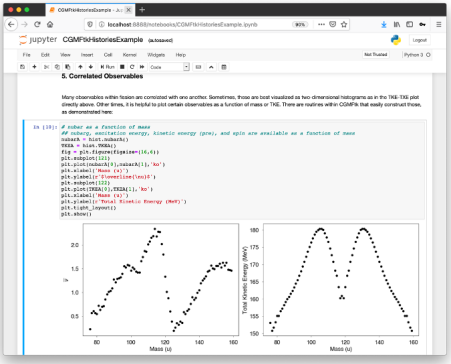
Integrated “low-energy” fission physics models to simulate event-by-event nuclear reactions.

CGMF on GitHub (1)

Includes source code, data, and Python Jupyter notebooks and tools

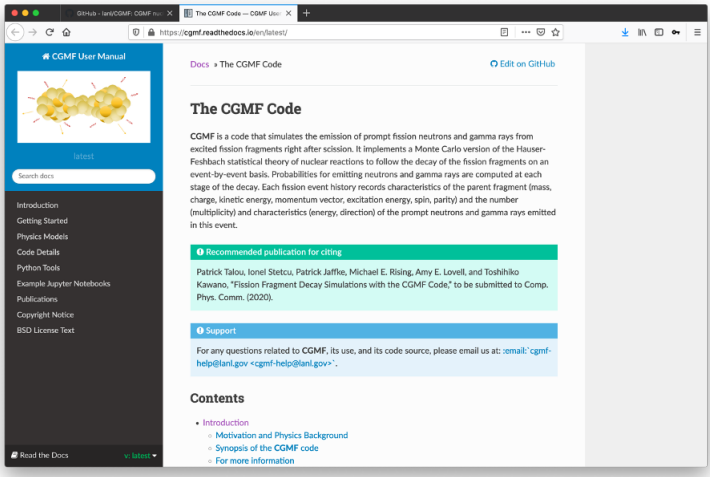
- https://github.com/lanl/cgmf





CGMF on GitHub (2)

- ReadTheDocs documentation published online

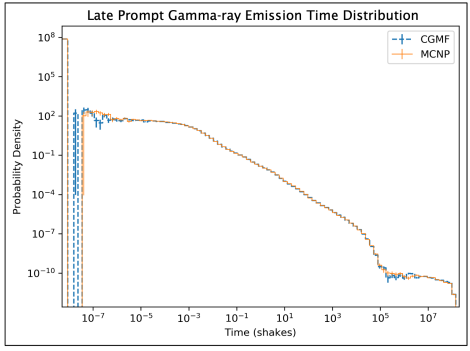


Physics and user manual published in Computer Physics Communications [13]

Updates for the MCNP6.3 Release (1)

CGMF updates (new fission systems in red)

- Spontaneous fission
  - Pu-238, -240, -242, -244, and Cf-252, -254
- Neutron-induced fission
  - U-233, -234, -235, -238, Np-237, and Pu-239, -241
- Late-time prompt fission gamma rays
- Fission fragment angular distributions
- Pre-equilibrium neutron emission

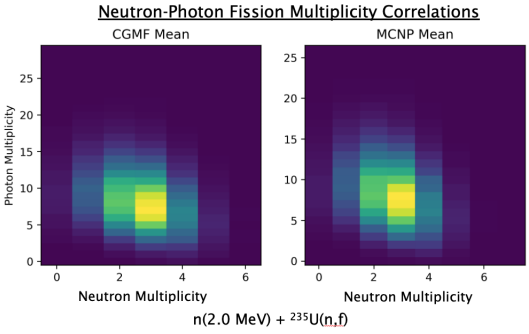


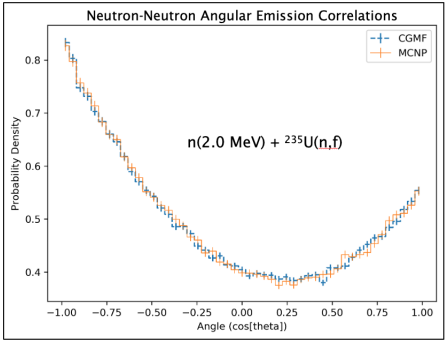
CGMF-MCNP Integration Updates

- Through the new MCNP CMake build system (`find_package`)
- Built as a library, linked to MCNP executable
- Same library can also be linked to make the CGMF executable

Updates for the MCNP6.3 Release (2)

- Verification of the integrated CGMF code and MCNP interface
  - Done with new HDF5 PTRAC and MPI





No change to MCNP input options. To use CGMF → `FMULT METHOD=7`

Additional Minor Updates

- For MCNP6.3
  - The logic in MCNP6.2 to handle the combination of prompt neutrons from the correlated fission models (CGMF, FREYA, and LLNL Fission Library) and delayed neutrons from ACE data tables is flawed. This has been corrected in MCNP6.3.

- For CGMF 1.1
  - A patch to fix data file reading in mixed Windows/Linux environments (e.g. WSL) was pushed to GitHub. This patch version CGMF 1.1.1 was released in April 2022.

- Moving forward with MCNP6.3 + CGMF 1.1.x should be easy to manage if you have the MCNP6.3 source code
  - The CGMF 1.1.x interface will not change
  - Therefore, future patch releases of CGMF 1.1.x should be drop-in replacements without changes required to the MCNP6.3 source code itself

Summary and Future Plans

- As a result of a multi-year NA-22 project,
  - CGMF was integrated into MCNP6.2 and publicly released
  - CGMF was open-sourced and publicly released
  - MCNP6.3 was updated to include the latest version and is in the process of being publicly released

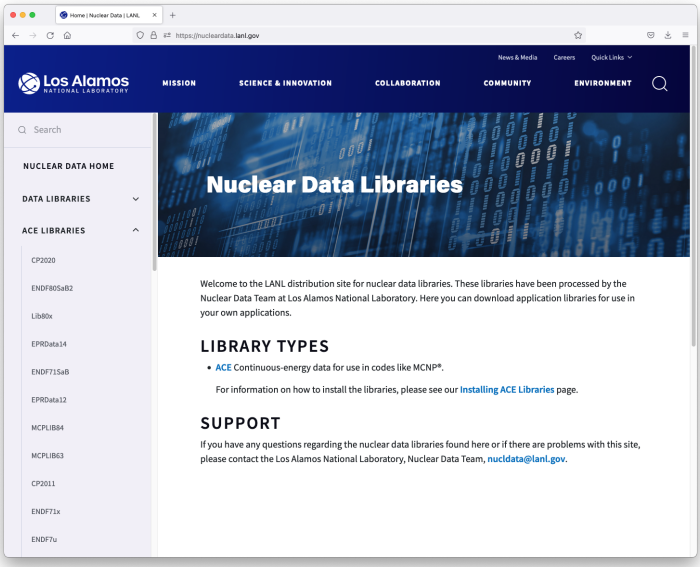
- Current and future plans
  - A.E. Lovell was awarded a LANL Early Career LDRD to work on global optimization and uncertainty quantification within CGMF
  - D. Neudecker and A.E. Lovell have been working on model parameter fitting such that CGMF may be used in ENDF/B evaluations
  - T. Kawano and M.E. Rising are collaborating with RPI to improve both standalone and MCNP-integrated CGM (non-fission) simulations

Contact the LANL CGMF Developers at cgmf-help@lanl.gov

Nuclear Data Downloading Tool

Motivation: Making Nuclear Data More Accessible (1)

Nuclear Data Available to Download - https://nucleardata.lanl.gov/



Motivation: Making Nuclear Data More Accessible (2)

- Nuclear data libraries have historically been shipped with the MCNP code
- Getting updated nuclear data usually required
  - Waiting for new MCNP code release
  - Downloading the ENDF-format nuclear data and doing the NJOY processing manually

- Nuclear data team at LANL (XCP-5) have made available several nuclear data libraries on their website (https://nucleardata.lanl.gov/)
  - More easily provide nuclear data libraries
  - More timely delivery of nuclear data updates

- Colin Josey (XCP-3) has developed a Python tool for downloading and installing specific nuclear data libraries
  - Makes it simple to install and uninstall specific nuclear data library(ies)
  - Generates the nuclear data directory (XSDIR) file with user-specified ordering

`nd_manager` Usage: Initial Configuration

- The Python script, `nd_manager.py`, and required atomic weight ratio data will ship with the MCNP code
- On first run, the script will generate a configuration file and define where to get nuclear data and where it will be installed

```
(base)                  :/mnt/c/Users/      /staff_xcp3/mcnp_classes/criticality/data_downloader$ ./nd_manager.py update
No configuration found. Generating.
Where would you like your data to be stored [Default: ./MCNP_DATA]?
Where would you like your downloaded files to be stored [Default: ./Downloads]?
Would you like to keep the archived data after installation [Y/n]? y
The current enabled repositories are: LANL_NDT.
Would you like to add another [y/N]? n
(base)                  :/mnt/c/Users/      /staff_xcp3/mcnp_classes/criticality/data_downloader$ ls -l
total 120
-rwxrwxrwx 1 arclark arclark 86956 Jun  1 16:11 awr_data
-rwxrwxrwx 1 arclark arclark   304 Jun  3 14:53 config.json
drwxrwxrwx 1 arclark arclark   512 Jun  3 14:53 Downloads
-rwxrwxrwx 1 arclark arclark 25069 Jun  1 16:11 nd_manager.py
-rwxrwxrwx 1 arclark arclark  2789 Jun  1 16:11 README.md
```

`nd_manager` Usage: Downloading Nuclear Data

- Straightforward to view all available libraries and choose which ones to install
- Can choose to download nuclear data libraries either by name (e.g. “Lib80x”) or using the “--all available” or “--all production” flags

```
(base)                  :/mnt/c/Users/      /staff_xcp3/mcnp_classes/criticality/data_downloader$ ./nd_manager.py list
All libraries:
==============

TEST ERRATA LIBRARY - TEST ERRATA LIBRARY DESCRIPTION... [Production]

CP2020              - This library contains data for incident charged particles
                      for low Z isotopes and their interactions with other low
                      Z isotopes [Production]

ENDF80SaB2          - ENDF/B-VIII.0-based thermal scattering library.
                      [Production]

Lib80x              - The general ACE library based on ENDF/B-VIII.0 containing
                      continuous-energy incident neutron data [Production]
```

```
(base)                  :/mnt/c/Users/      /staff_xcp3/mcnp_classes/criticality/data_downloader$ ./nd_manager.py download Lib80x
Downloading https://nucleardata.lanl.gov/lib/Lib80x.tgz
[                                                  ] [0.19 GiB / 6.57 GiB], 2.95%
```

nd_manager Usage: Installing Nuclear Data

- Once downloaded, the script is used to install the nuclear data and generate an XSDIR file

```
(base)                  :/mnt/c/Users/      /staff_xcp3/mcnp_classes/criticality/data_downloader$ ./nd_manager.py install Lib80x
Installing Lib80x
```

```
(base)                  :/mnt/c/Users/      /staff_xcp3/mcnp_classes/criticality/data_downloader$ ./nd_manager.py create-xsdir Lib80x
(base)                  :/mnt/c/Users/      /staff_xcp3/mcnp_classes/criticality/data_downloader$ ls -l MCNP_DATA/
total 408
drwxrwxrwx 1 arclark arclark    512 Jun  4 13:50 database
drwxrwxrwx 1 arclark arclark    512 Jun  4 13:50 docs
drwxrwxrwx 1 arclark arclark    512 Jun  4 13:50 Lib80x
-rwxrwxrwx 1 arclark arclark 417169 Jun  4 14:09 xsdir_mcnp6.3
```

- Specifying the “--all available” or “--all production” flags will install the corresponding nuclear data libraries that were previously downloaded
- Order of libraries in the XSDIR file is set in the config
  - Order of repositories in config, then order of libraries in each repository
  - Updated whenever new nuclear data libraries are installed/uninstalled

```
(base)                  :/mnt/c/Users/      /staff_xcp3/mcnp_classes/criticality/data_downloader$ ./nd_manager.py uninstall Lib80x
Uninstalling Lib80x
Update XSDIR? [y/N]
y
```

nd_manager Usage: XSDIR File Creation

- XSDIR file for MCNP usage can be generated, based on what was downloaded and installed

```
1  #-------------------------------------------------------------------------------
2  # xsdir
3  # This file was automatically generated by the nuclear data downloader tool.
4  #-------------------------------------------------------------------------------
5  atomic weight ratios
6     0001  1.000000    0001  1.000000
7     1000  0.99931697 1001  0.99916733   1002  1.99679968   1003  2.99013997
8                       1004  3.99320563   1005  4.99205575   1006  5.99301364
9                       1007  6.99216250
10    2000  3.96821760 2003  2.99012018   2004  3.96821897   2005  4.96916622
11                      2006  5.96718398   2007  6.96764655   2008  7.96490665
12                      2009  8.96625841   2010  9.96604391
13    3000  6.88131188 3003  3.00473919   3004  3.99259010   3005  4.96947769
14                      3006  5.96345000   3007  6.95573370   3008  7.95357035
15                      3009  8.94924505   3010  9.94927166   3011  10.94892619
16                      3012  11.95023224
17    4000  8.93476310 4005  4.99748719   4006  5.96801398   4007  6.95665103
18                      4008  7.93653569   4009  8.93476323   4010  9.92751276
19                      4011  10.92697642  4012  11.92360372  4013  12.92371015
20                      4014  13.92225773  4015  14.92414355  4016  15.92394040
```

```
1129 06/04/2022
1130 directory
1131 1001.00c 0.999167 Lib80x/H/1001.800nc 0 1 3 10257 0 0 2.530100E-08
1132 1001.01c 0.999167 Lib80x/H/1001.801nc 0 1 3 10257 0 0 5.170400E-08
1133 1001.02c 0.999167 Lib80x/H/1001.802nc 0 1 3 10257 0 0 7.755600E-08
1134 1001.03c 0.999167 Lib80x/H/1001.803nc 0 1 3 10257 0 0 1.034100E-07
1135 1001.04c 0.999167 Lib80x/H/1001.804nc 0 1 3 10257 0 0 2.154300E-07
1136 1001.05c 0.999167 Lib80x/H/1001.805nc 0 1 3 10257 0 0 8.617400E-12
1137 1001.06c 0.999167 Lib80x/H/1001.806nc 0 1 3 10257 0 0 2.154300E-08
1138 1002.00c 1.996800 Lib80x/H/1002.800nc 0 1 3 43756 0 0 2.530100E-08
1139 1002.01c 1.996800 Lib80x/H/1002.801nc 0 1 3 43922 0 0 5.170400E-08
1140 1002.02c 1.996800 Lib80x/H/1002.802nc 0 1 3 43987 0 0 7.755600E-08
1141 1002.03c 1.996800 Lib80x/H/1002.803nc 0 1 3 44076 0 0 1.034100E-07
1142 1002.04c 1.996800 Lib80x/H/1002.804nc 0 1 3 44208 0 0 2.154300E-07
1143 1002.05c 1.996800 Lib80x/H/1002.805nc 0 1 3 43008 0 0 8.617400E-12
1144 1002.06c 1.996800 Lib80x/H/1002.806nc 0 1 3 43712 0 0 2.154300E-08
1145 1003.00c 2.989596 Lib80x/H/1003.800nc 0 1 3 17764 0 0 2.530100E-08
1146 1003.01c 2.989596 Lib80x/H/1003.801nc 0 1 3 17864 0 0 5.170400E-08
1147 1003.02c 2.989596 Lib80x/H/1003.802nc 0 1 3 17919 0 0 7.755600E-08
1148 1003.03c 2.989596 Lib80x/H/1003.803nc 0 1 3 17944 0 0 1.034100E-07
1149 1003.04c 2.989596 Lib80x/H/1003.804nc 0 1 3 18014 0 0 2.154300E-07
1150 1003.05c 2.989596 Lib80x/H/1003.805nc 0 1 3 16869 0 0 8.617400E-12
1151 1003.06c 2.989596 Lib80x/H/1003.806nc 0 1 3 17744 0 0 2.154300E-08
1152 2003.00c 2.989032 Lib80x/He/2003.800nc 0 1 3 10004 0 0 2.530100E-08
```

nd_manager Summary

- Allowing the LANL Nuclear Data and Monte Carlo teams to work and distribute their products independently
- Updates, fixes (errata), and newly released data will be made available in a much faster timescale than ever before
- Demo (if time permits)

Verification and Validation Framework

MCNP Testing

All of these techniques are used during MCNP development, either daily or weekly, in the pursuit of ultimately providing a robust MCNP product.

Regression	With saved templates of output and result files, the current execution of the MCNP code is compared to these stored templates to ensure nothing changed unexpectedly. Intended to run quickly, so there is no sense in trying to understand any statistical convergence or aggregate behavior of the code.
Verification	Where analytical and semi-analytical solutions to the transport equation may exist, we want to ensure that MCNP is solving the correct equations
Validation	Combination of code (MCNP) and nuclear data (ENDF/NJOY/ACE) work together to produce results comparable to reality

Long-standing reputation can be linked to extensive verification and validation (V&V). The remainder of this presentation will focus on V&V.

Previously Released V&V Suites

V&V Suites in MCNP6.1, MCNP6.1.1, and MCNP6.2

KOBAYASHI	3-D fixed-source streaming problems [14]
VERIFICATION_KEFF	Analytic k-effective test problems [15]
VERIFICATION_SHLD_SVDM	Various shielding benchmarks from SINBAD [16]
VALIDATION_CRITICALITY	31 ICSBEP criticality benchmarks [17]
VALIDATION_CRIT_EXPANDED	119 ICSBEP criticality benchmarks [18]
VALIDATION_SHIELDING	Neutron and photon fixed-source benchmarks [19]

Additional V&V Suites in MCNP6.1 and MCNP6.1.1

VALIDATION_CEM	Differential tests of the high-energy physics code CEM [20]
VALIDATION_LAQGSM	Differential tests of the high-energy physics code LAQGSM [21]
VALIDATION_ROSSI_ALPHA	Integral tests of the 13 ICSBEP benchmarks with Rossi-α measurements [22]

Limitations of the Previously Released V&V Suites

- Mixture of Makefile, Perl, Windows .bat scripts used to execute problems (ALL)
  - Missing execution scripts entirely (CEM and LAQGSM)
- Problems cannot be run directly without preprocessing or suite-specific XSDIR files (VALIDATION_CRITICALITY and VALIDATION_CRIT_EXPANDED)
- Misleading suite not doing actual verification (VERIFICATION_SHLD_SVDM)
- Postprocessing results scripts inconsistent and/or missing (VALIDATION_SHIELDING, CEM and LAQGSM)
- No job submission / cluster support (ALL)
- Plotting / visualization support missing, broken, or incomplete (ALL)
- Any sort of documentation requires manual intervention (ALL)

New Python-based Framework

- **Consistency** across suites
- **Extensible** to more suites and problem types
- **Automated** for all steps
  - Setup
  - Execute
  - Postprocess
  - Document
- Requires Python3 (and various standard packages)
- Requires MCNPTools
- Runs on Linux, MacOS, and Windows

Listing 4: Sample description.json File

```
{
  "general_info": {
    "name": "GODIVA",
    "icsbep_name": {
      "material": "HEU",
      "form": "MET",
      "spectrum": "FAST",
      "number": "001",
      "case": ""
    },
    "description": "Bare HEU sphere"
  },
  "execution_info": {
    "arguments": {
      "i": "GODIVA",
      "n": "GODIVA"
    },
    "outputs": {
      "outp": "GODIVAo",
      "mctal": "GODIVAm"
    },
    "inputs": {
      "inp": "GODIVA"
    }
  },
  "experiment_data": {
    "k-eff": {
      "val": 1.0,
      "std": 0.001
    }
  }
}
```

New Python-based Framework

- Can be immediately used for any version of the code (input and data options must be considered)
- For developers
  - Can test code and data frequently
  - V&V reports are essential for a release
- For everyone else
  - Can add application-specific V&V suites
  - Can support SQA needs

Listing 5: vnvstats Directory Structure

```
vnvstats /
 |- README.md
 |- support /
   |- mcnpvnv.py       (MCNP-specific V&V functionality)
   |- vnv /            (Generic V&V functionality)
     |- README.md
     |- __init__.py
     |- benchcalc.py     (Benchmark/experiment handling)
     |- commandline.py   (Command line parser/execution)
     |- compare.py       (Compare results)
     |- plotndoc.py      (Tables, plots and docs)
     |- slurmin.py       (Support SLURM submission)
 |- validation /
   |- criticality /
     |- README.md
     |- VnV.py         (Drives criticality suite)
     |- experiments /
       |- GODIVA /
         |- GODIVA             (Benchmark model)
         |- description.json   (Experiment info)
       |- ...
   |- pulsed_spheres /
   |- lockwood /
   |- ...
 |- verification /
   |- keff /
   |- kobayashi /
   |- ...
```

New Python-based Framework

List - Query test suite for available test problems

```
$ python VnV.py list
```

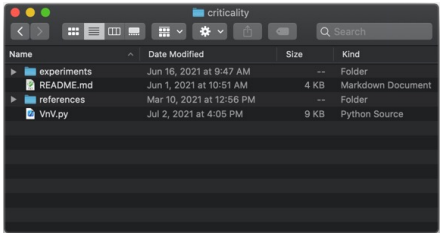
Listing 6: vnvstats Listing Example

```
criticality $ python VnV.py list
All available tests in validation criticality:
  BAWXI2
  BIGTEN
  FLAT23
  FLAT25
  FLATPU
  FLSTF1
  GODIVA
  GODIVR
  HISHPG
  ICT2C3
  IMF03
  IMF04
  JEZ233
  JEZ240
  JEZPU
  LST2C2
  ORNL10
  ORNL11
  PNL2
  PNL33
  PUBTNS
  PUSH20
  SB25
  SB5RN3
  STACY36
  THOR
  TT2C11
  UH3C6
  UMF5C2
  ZEBR8H
  ZEUS2
```

New Python-based Framework

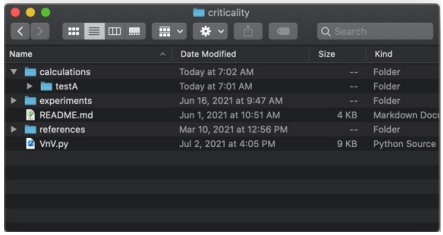
Setup - Creates a calculation tree of benchmarks selected

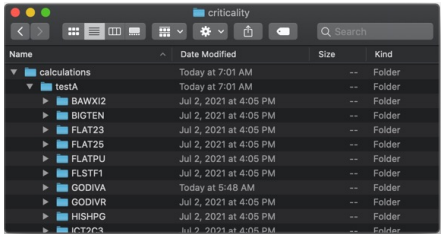
```
$ python VnV.py setup \
$ --calcdir_name testA
```



Example of calculation tree with only listed benchmarks:

```
$ python VnV.py setup \
$ --calcdir_name testB GODIVA JEZPU
```





New Python-based Framework

Execute - Runs all problems in existing calculation directory

```
$ python VnV.py execute --calcdir_name testA
```

- **Option examples**
  - `--executable_name mcnp6`
  - `--jobs 2` (concurrent execution)
  - `--ntrd 8` (threads for each job)
  - `--nmpi 4` (ranks for each job)

- Builds command line from `execution_info` group

Listing 7: Sample description.json Execution Info

```
1  "execution_info": {
2    "arguments": {
3      "i": "GODIVA",
4      "n": "GODIVA"
5    },
```

New Python-based Framework

Execution Submission - Submits all problems in existing calculation directory via slurm/sbatch

```
$ python VnV.py execute_slurm --calcdir_name testA
```

- **Option examples**
  - `--nodes 1` (node allocation)
  - `--time 120` (time allocation in minutes)
  - `--stride 8` (jobs per sbatch job submitted)
  - `--wait` (flag to wait for execution to complete before proceeding)
  - `--pre_cmd` and `--post_cmd`
    (commands to run before and/or after MCNP execution within sbatch submission script)

New Python-based Framework

Postprocess - Reads calculation output files and processes results into calculation description.json

```
$ python VnV.py postprocess --calcdir_name testA
```

- Adds `calculation_data` and `calculation_info` objects to JSON file
- `experiment_data` and `calculation_data` directly comparable
- All suites will likely postprocess MCNP results differently
- Using MCNPTools wherever possible

Listing 8: Sample description.json Calculation Data and Info

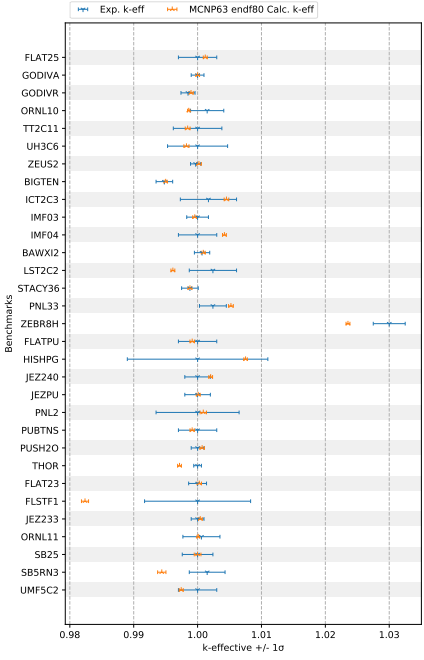
```
1  "calculation_data": {
2    "k-eff": {
3      "val": 1.00002,
4      "std": 0.000283388
5    }
6  },
7  "calculation_info": {
8    "code": "mcnp6",
9    "version": "6",
10   "date": "05/04/22 23:40:20"
11 }
```

New Python-based Framework

Documentation - Retrieves experiment and simulation results from calculation description.json and prepares documentation

```
$ python VnV.py document \
$ --calcdir_name testA
```

- Results are tabulated into text and LaTeX form
- Plots are generated into PNG outputs
- Between LaTeX text, tables, and PNG plots, a V&V report is nearly done



New Python-based Framework

Nominal workflow - Setup, execute, postprocess, and document a suite of test problems

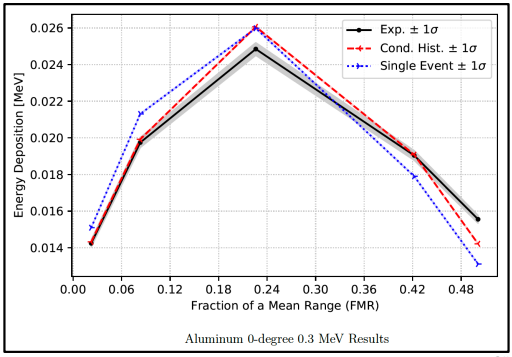
1. `python VnV.py setup --calcdir_name MCNP63_VV`
2. `python VnV.py execute --calcdir_name MCNP63_VV`
3. `python VnV.py postprocess --calcdir_name MCNP63_VV`
4. `python VnV.py document --calcdir_name MCNP63_VV`

Additional Test Suite(s)

- Beyond the actual MCNP input files, two ingredients are required to create a new suite:
  - **description.json** files, each benchmark (easy)
    - `execution_info` : maps to MCNP command line options/arguments and input/output files
    - `experiment_data` : benchmark results used to compare to calculation results
  - **VnV.py** script, each suite (medium/hard)
    - `list` : same for all test suites
    - `setup` : same for all test suites
    - `execute` : same for all test suites
    - `execute_slurm` : same for all test suites
    - `postprocess` : unique to every test suite
    - `document` : unique to every test suite

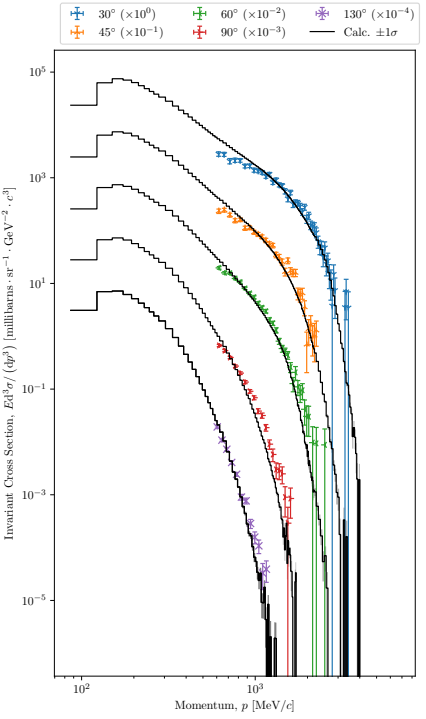
Additional Test Suite(s)

- Finished incorporating Lockwood validation test suite
  - Electron transport energy deposition
    - Condensed history algorithm
    - Single event electrons
  - Several materials, 334 separate MCNP inputs
  - Reasonably computationally expensive (need cluster / high performance computing)



Additional Test Suite(s)

- Resurrecting LAQGSM validation test suites
  - No Makefile or other scripts to execute code and/or postprocess results
  - Gaining experience through old tests, documentation and trail of bread crumbs. . .



Summary

- All MCNP team supported V&V test suites are now developed in a separate repository from the MCNP source code within a Python-based framework including:
  - Python tools and scripts
  - Benchmark inputs and description JSON files
- This entire framework will be distributed with the MCNP6.3 release
- Most V&V test suites distributed with MCNP6.2 will be distributed in new framework
- New V&V test suites have been added for the MCNP6.3 release
- Looking forward to feedback and potential contributions

Q&A

Backup Slides

References

[1] V. K. Mehta, J. Armstrong, D. V. Rao, and D. Kotlyar, “Capturing Multiphysics Effects in Hydride Moderated Microreactors using MARM,” *Annals of Nuclear Energy*, vol. 172, p. 109067, 2022.

[2] R. B. Wilkerson, G. McKinney, M. E. Rising, and J. A. Kulesza, “MCNP Reactor Multigroup Tally Options Verification,” Tech. Rep. LA-UR-20-27819, Los Alamos National Laboratory, Oct. 2020.

[3] C. J. Josey and J. A. Kulesza, “Improved FMESH Capabilities in the MCNP 6.3 Code,” Tech. Rep. LA-UR-21-26363, Los Alamos National Laboratory, July 2021.

[4] 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.

[5] 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.

References

[6] J. A. Kulesza, J. C. Armstrong, T. C. McClanahan, and S. Swaminarayan, "MCNP Unstructured Mesh Elemental Quality Assessment," Tech. Rep. LA-UR-21-26362, Los Alamos National Laboratory, Los Alamos, NM, USA, July 2021. MCNP User Symposium Presentation.

[7] C. J. Stimpson, C. D. Ernst, P. K. and. P. P. Pébay, and D. Thompson, "The Verdict Geometric Quality Library," Tech. Rep. SAND2007-1751, Sandia National Laboratories, Albuquerque, NM ,USA, Mar. 2007.

[8] J. A. Kulesza, "MCNP Code Version 6.3.0 Unstructured-mesh Quality Metrics & Assessment," Tech. Rep. LA-UR-20-27150, Los Alamos National Laboratory, Los Alamos, NM, USA, Sept. 2020.

[9] P. M. Knupp, "Achieving Finite Element Mesh Quality Via Optimization of the Jacobian Matrix Norm and Associated Quantities. Part II—A Framework for Volume Mesh Optimization and the Condition Number of the Jacobian Matrix," *International Journal for Numerical Methods in Engineering*, vol. 48, pp. 1165–1185, May 2000.

References

[10] P. M. Knupp, “Algebraic Mesh Quality Metrics for Unstructured Initial Meshes,” *Finite Elements in Analysis and Design*, vol. 39, pp. 217–241, Jan. 2003.

[11] P. P. Pébay, D. Thompson, J. Shepherd, P. Knupp, C. Lisle, V. A. Magnotta, and N. M. Grosland, “New Applications of the Verdict Library for Standardized Mesh Verification Pre, Post, and End-to-End Processing,” in *Proceedings of the 16th International Meshing Roundtable* (M. L. Brewer and D. Marcum, eds.), (Seattle, WA, USA; October 15–17), pp. 535–552, 2008.

[12] R. L. Martz, “The MCNP6 Book On Unstructured Mesh Geometry: User’s Guide For MCNP 6.2,” Tech. Rep. LA-UR-17-22442, Los Alamos National Laboratory, Los Alamos, NM, USA, Mar. 2017.

[13] P. Talou, I. Stetcu, P. Jaffke, M. E. Rising, A. E. Lovell, and T. Kawano, “Fission Fragment Decay Simulations with the CGMF Code,” *Computer Physics Communications*, vol. 269, p. 108087, 2021.

[14] K. Kobayashi, N. Sugimura, and Y. Nagaya, “3D Radiation Transport Benchmark Problems and Results for Simple Geometries with Void Region,” *Progress in Nuclear Energy*, vol. 39, no. 2, pp. 119–144, 2001.

References

[15] A. Sood, R. A. Forster, and D. K. Parsons, “Analytical Benchmark Test Set for Criticality Code Verification,” *Progress in Nuclear Energy*, vol. 42, no. 1, pp. 55–106, 2003. Also: LA-UR-01-3082.

[16] I. A. Kodeli and E. Sartori, “SINBAD – Radiation Shielding Benchmark Experiments,” *Annals of Nuclear Energy*, vol. 159, p. 108254, 2021.

[17] R. D. Mosteller, “Comparison of Results from the MCNP(TM) Criticality Validation Suite Using ENDF/B-VI and Preliminary ENDF/B-VII Nuclear Data,” *AIP Conference Proceedings*, vol. 769, May 2005.

[18] R. D. Mosteller, F. B. Brown, and B. C. Kiedrowski, “An Expanded Criticality Validation Suite for MCNP,” July 2011.

[19] B. C. Kiedrowski, F. B. Brown, N. A. Gibson, A. S. Bennett, and M. A. Gonzales, “MCNP6 Shielding Validation Suite: Past, Present, and Future,” *Transactions of the American Nuclear Society*, vol. 105, pp. 559–561, 2011.

References

[20] S. G. Mashnik, “Validation and Verification of MCNP6 Against High-Energy Experimental Data and Calculations by Other Codes. I. The CEM Testing Primer,” Tech. Rep. LA-UR-11-05129, Los Alamos National Laboratory, Los Alamos, NM, USA, 2011.

[21] S. G. Mashnik, “Validation and Verification of MCNP6 Against High-Energy Experimental Data and Calculations by Other Codes. II. The LAQGSM Testing Primer,” Tech. Rep. LA-UR-11-05627, Los Alamos National Laboratory, Los Alamos, NM, USA, 2011.

[22] R. D. Mosteller and B. C. Kiedrowski, “The Rossi Alpha Validation Suite for MCNP,” July 2011.

[23] U. Ayachit, *The ParaView Guide*. Kitware, Inc., community ed., June 2018.

[24] J. A. Kulesza and T. C. McClanahan, “A Python Script to Convert MCNP Unstructured Mesh Elemental Edit Output Files to XML-based VTK Files,” Tech. Rep. LA-UR-19-20291, Rev. 2, Los Alamos National Laboratory, Los Alamos, NM, USA, Nov. 2019.

References

[25] J. A. Kulesza, J. L. Alwin, J. D. Hutchinson, E. F. Shores, and R. C. Little, "l3d2vtk: An MCNPTools Utility to Enable LNK3DNT File Visualization & Post-processing," in *Transactions of the Winter 2019 American Nuclear Society Meeting*, vol. 121, pp. 1233–1236, Nov. 2019.

[26] H. Childs, E. Brugger, B. Whitlock, J. Meredith, S. Ahern, D. Pugmire, K. Biagas, M. Miller, C. Harrison, G. H. Weber, H. Krishnan, T. Fogal, A. Sanderson, C. Garth, E. W. Bethel, D. Camp, O. Rübel, M. Durant, J. M. Favre, and P. Navrátil, "VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data," in *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, pp. 357–372, Oct. 2012.

[27] J. A. Clarke and E. R. Mark, "Enhancements to the eXtensible Data Model and Format (XDMF)," in *HPCMP User's Group Conference 2007. High Performance Computing Modernization Program: A Bridge to Future Defense*, (Pittsburgh, PA, USA; June 18–21), pp. 322–327, 2007.

[28] "XDMF Model and Format." Website, Mar. 2019.

MCNP6.3 Visualization Approaches

How to Use These Slides

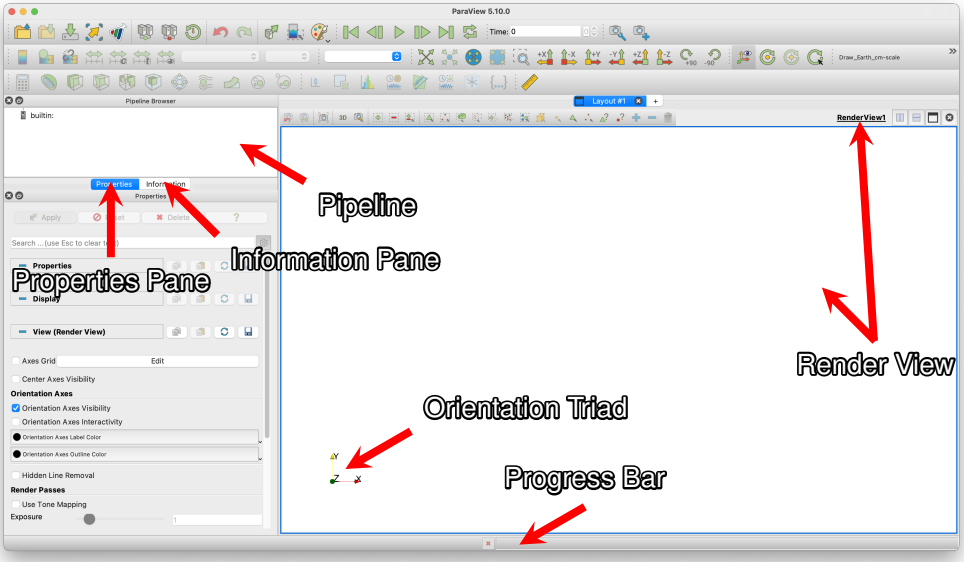
- Intended to be useful as a reference
  - Can be used independent of ParaView [23] being open
- GUI elements labeled to establish nomenclature
  - When “clicks” are necessary, steps are numbered to give click order
  - Additional information in lower-right “callout” boxes
- Recommended approach: focus on presentation and experiment after
  - Prevents “getting lost” while the presentation is underway
  - Experimentation is how skill and “muscle memory” is built

ParaView Introduction

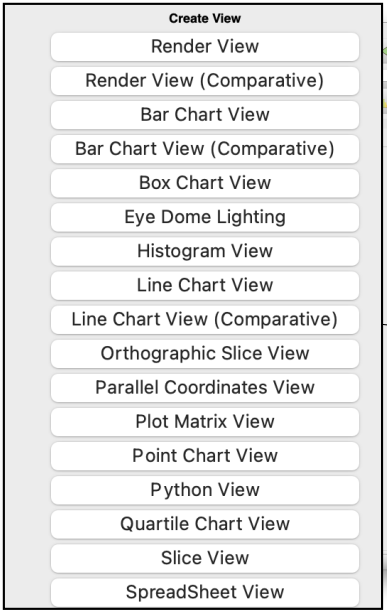
- Open-source & cross-platform software
- Maintained by Kitware Inc.
  - Kitware also maintains CMake, VTK, etc.
- Supported by, among others:
  - Advanced Simulation and Computing Program
  - Army Research Laboratory
  - Los Alamos National Laboratory
  - Sandia National Laboratories

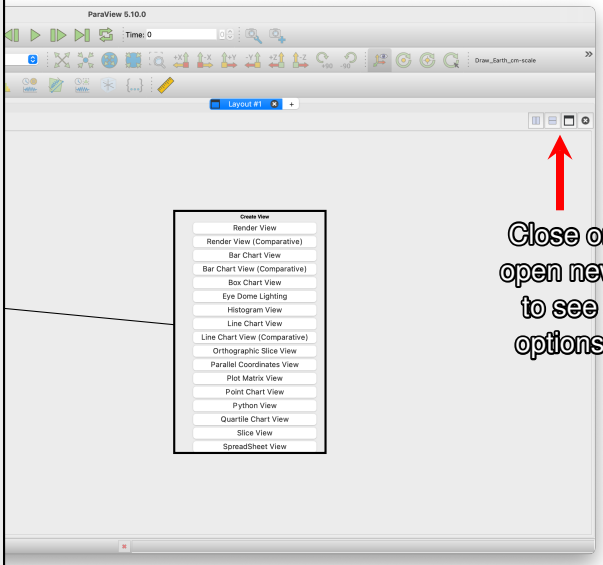
- Getting it: https://www.paraview.org/download
- Getting help with it: https://www.paraview.org/community-support
- Issues: https://gitlab.kitware.com/paraview/paraview/-/issues

ParaView Interface Overview—Main Components

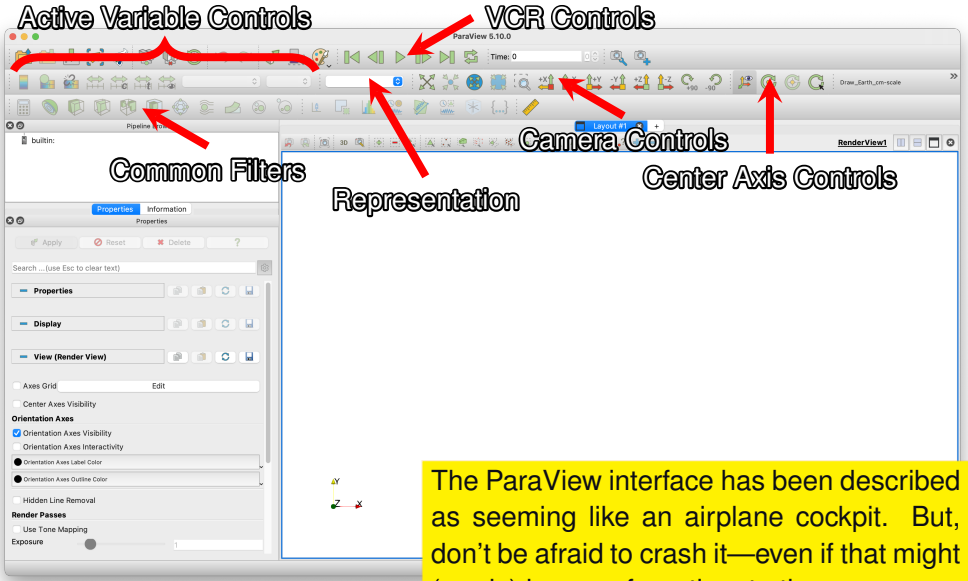


Views other than the Render View



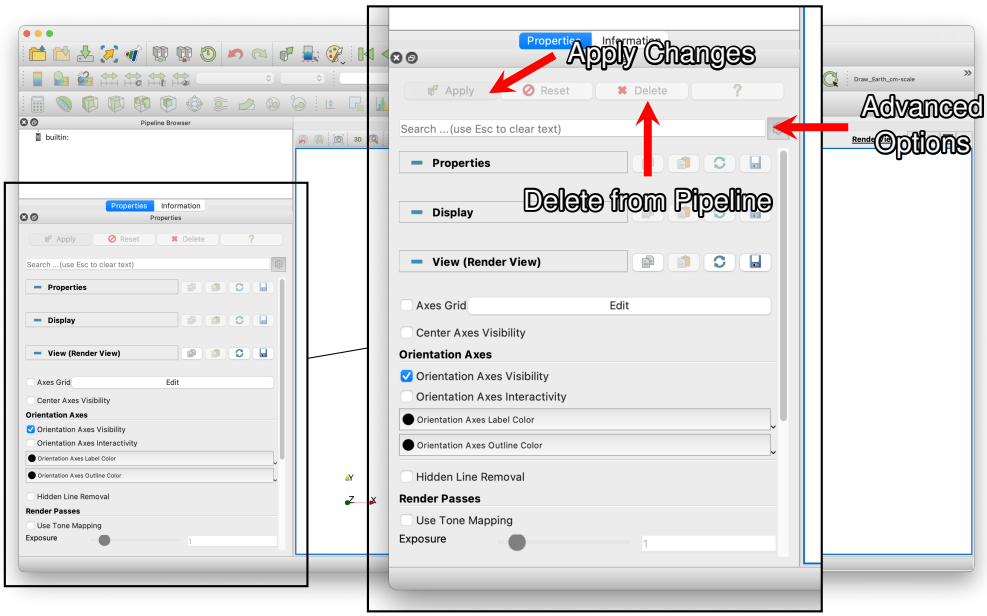


ParaView Interface Overview—Toolbars

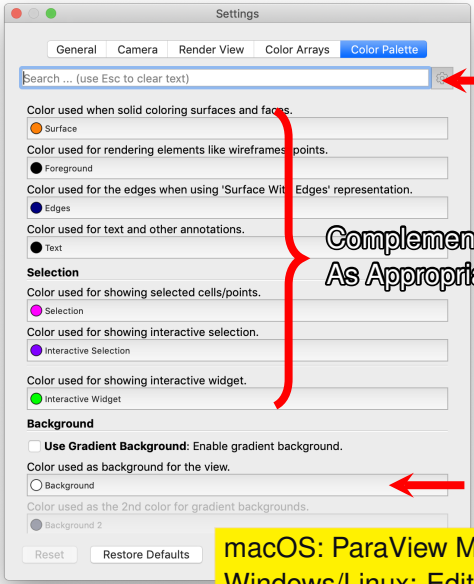


The ParaView interface has been described as seeming like an airplane cockpit. But, don't be afraid to crash it—even if that might (rarely) happen from time to time...

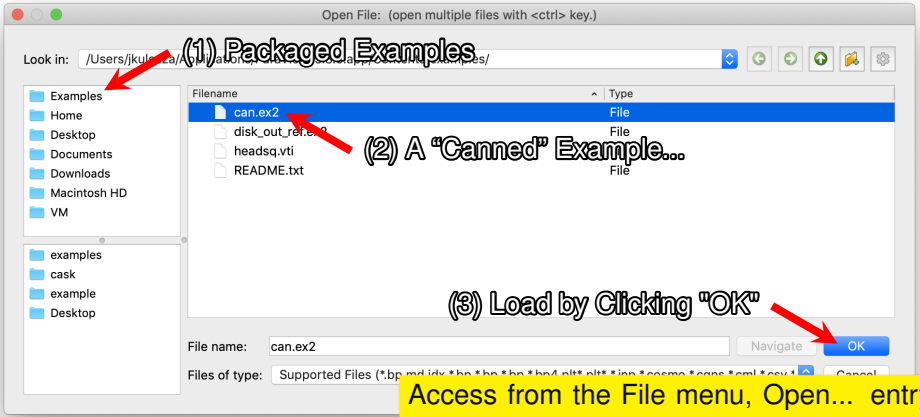
ParaView Interface Overview—Properties Pane



Render View Colors (Settings, Color Palette Tab)

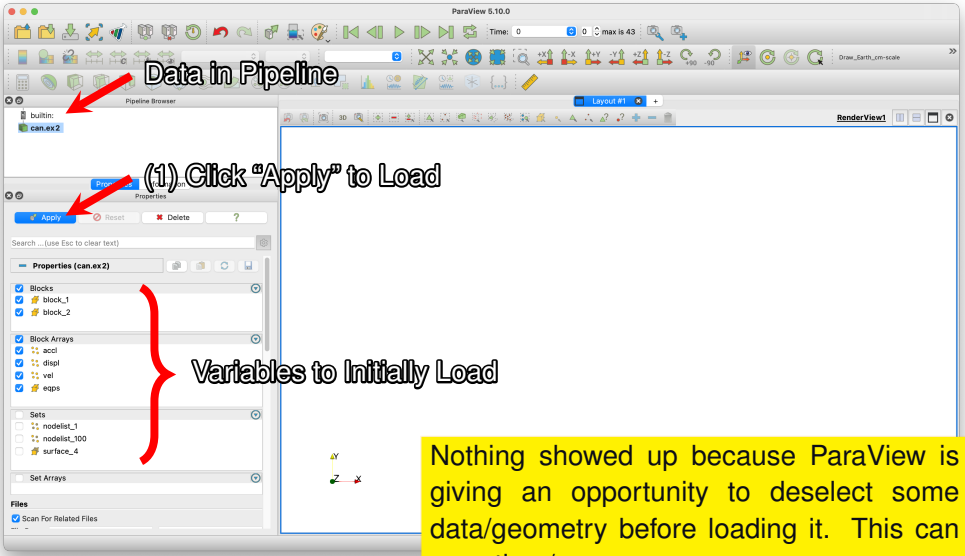


Loading (Example) Data to View

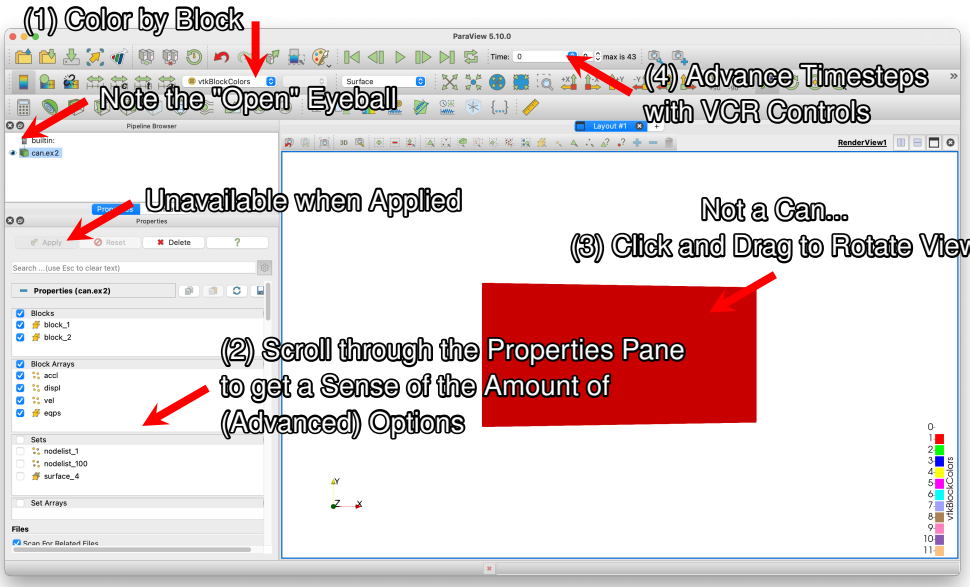


Access from the File menu, Open... entry. Some simple data files are provided with ParaView that can be used for familiarization and/or experimentation.

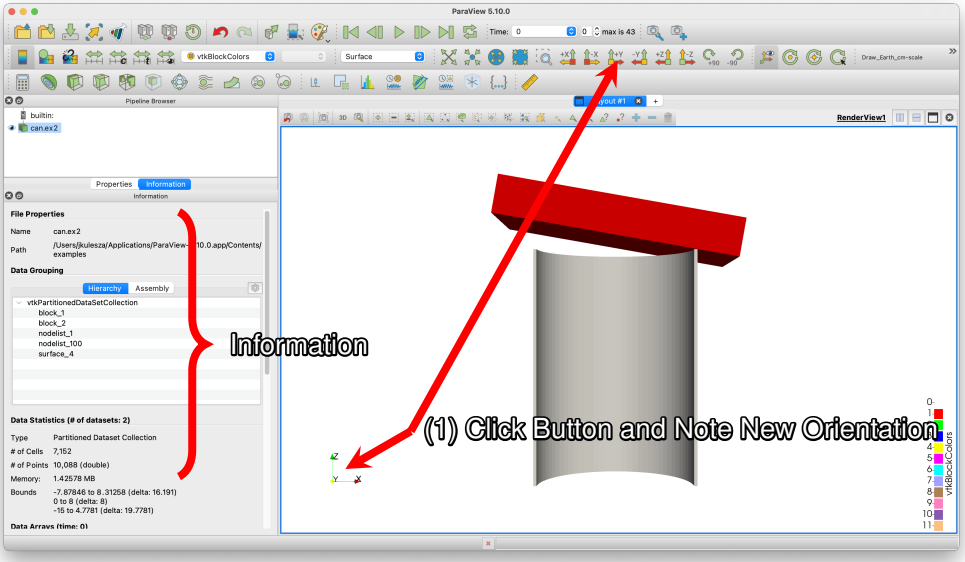
Moving Around / Controls



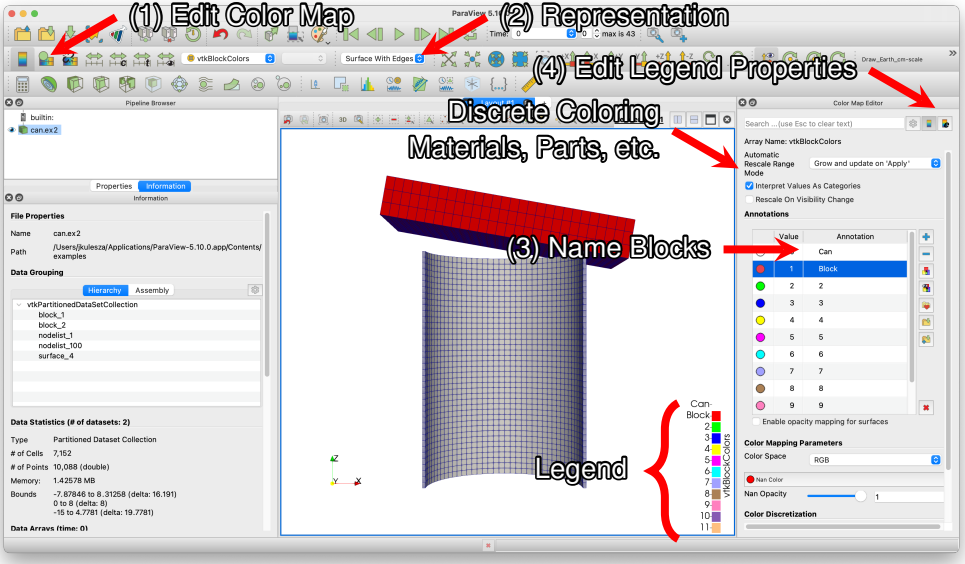
Moving Around / Controls (Properties Tab)



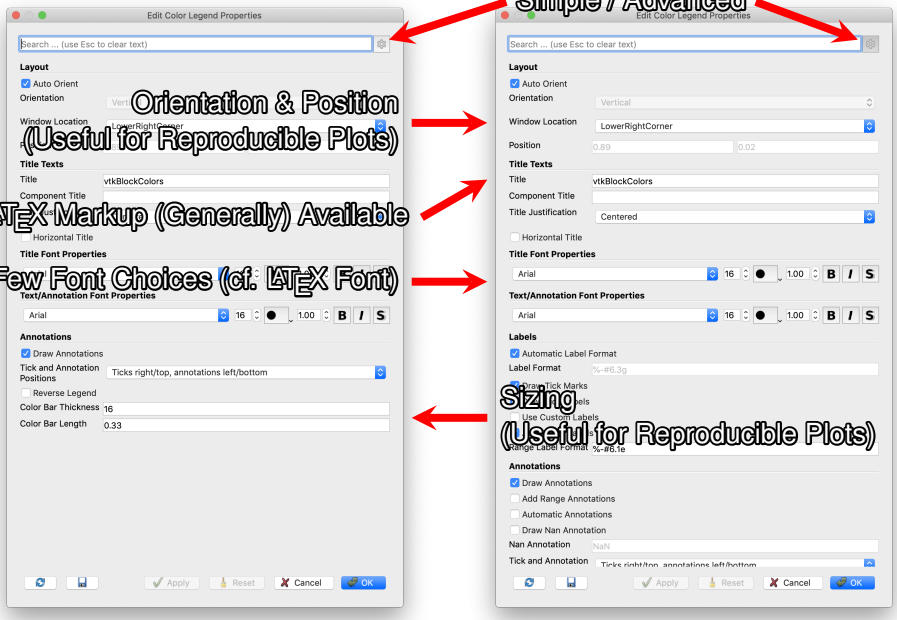
Moving Around / Controls (Information Tab)



Moving Around / Controls (Coloring & Legends)



Edit Color Map



Common MCNP Output Operations / Manipulations

Motivating Calculation Geometry & Results

- Focus: UM geometry and results
  - EEOUT file converted to an ASCII Unstructured VTK File (.vtu) [24]
  - HDF5+XDMF output files (not available in MCNP6.2 and earlier)
- Focus: FMESH-based mesh tallies
  - MESHTAL file converted to an ASCII Structured VTK File (.vts)
  - HDF5+XDMF output files (not available in MCNP6.2 and earlier)

- The needed HDF5+XDMF files are provided
  - Conversion processes shown next for completeness

- Other possibilities (not covered here)
  - Particle tracks
  - Point data (collisions, fissions, etc.)
  - Voxelized CSG representations [25]
  - Weight-window mesh files

MCNP UM File Format Conversion

- MCNP6.2 and earlier: use conversion script electronically attached to [24]
- MCNP6.3: direct `embed` XDMF+HDF5 output via `hdf5file=`*`filename`*
- Example conversion execution looks like:

```
> ./Convert_MCNP_eeout_to_VTK.py caas_hybrid.mcnp.eeout
Processing caas_hybrid.mcnp.eeout...
  Found 1 edit(s).
  Processing ENERGY_6 edit...
    Processing & Validating EDIT_6_RESULT_...
    Maximum           value: 1.19052e-02
    Minimum positive  value: 3.21896e-14
    Minimum           value: 0.00000e+00
    Processing & Validating EDIT_6_ERROR_...
    Maximum           value: 1.00000e+00
    Minimum positive  value: 8.86076e-04
    Minimum           value: 8.86076e-04
```

- Resulting file: `caas_hybrid.mcnp.eeout.vtu`

MCNP Mesh Tally File Format Conversion

- MCNP6.2 and earlier: use `meshtal2vtk` script with MCNPTools 5.2.1+
- MCNP6.3: direct `fmesh` XDMF+HDF5 output via `out=xdmf`
- Example `meshtal2vtk` execution looks like:

```
> meshtal2vtk meshtal
Processing mesh tally: 14 -> meshtal_14.vts
  Smallest non-zero tally value:  1.58150e-03
            Largest tally value:  3.57728e-02
```

- As before: the values are useful for setting log-scaled colorbar bounds

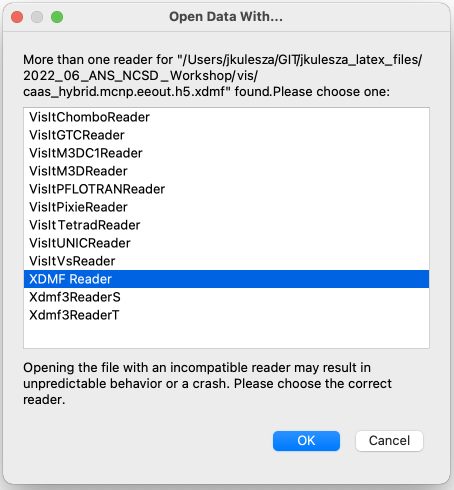
Introduction to XDMF [27, 28]

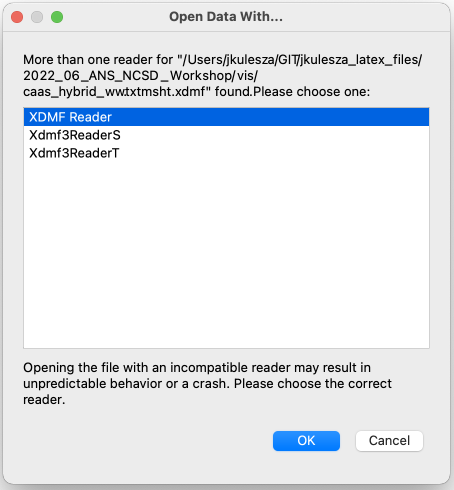
- ASCII, XML-formatted, file
  - Standard, albeit not actively developed, file format
  - Supports various structured and unstructured geometries
  - Can contain data and/or **contain pointers into HDF5 files**

- Cross-platform & open-source support (direct visualization of output)
  - ParaView [23]
  - VisIt [26]

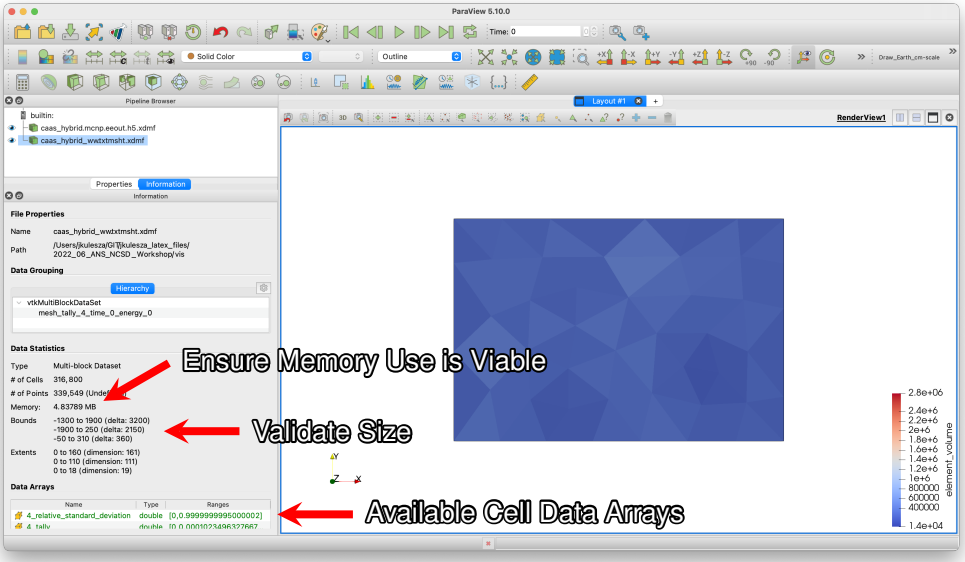
- No additional MCNP library dependencies
  - MCNP outputs are XDMF version 2

Loading XDMF Version 2 Files

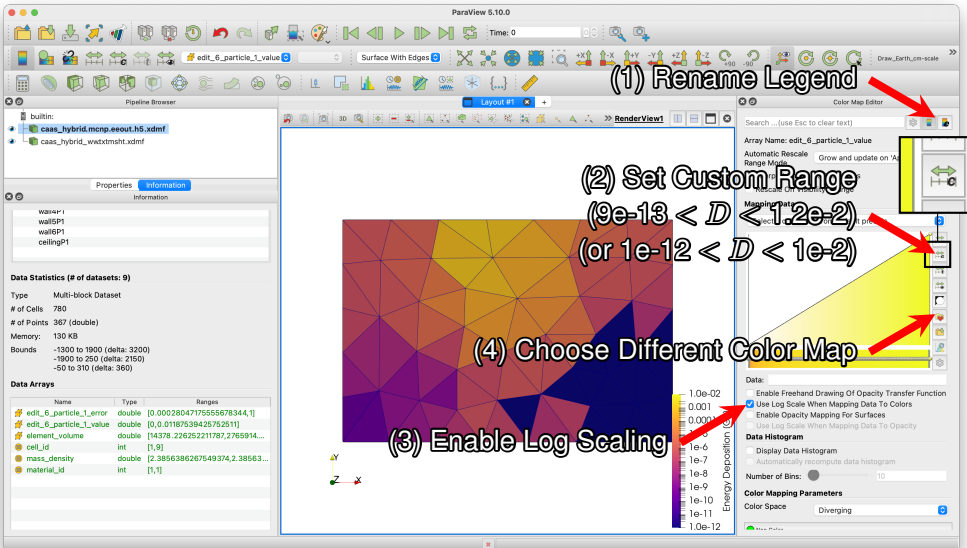




After Loading, Check the File “Information”

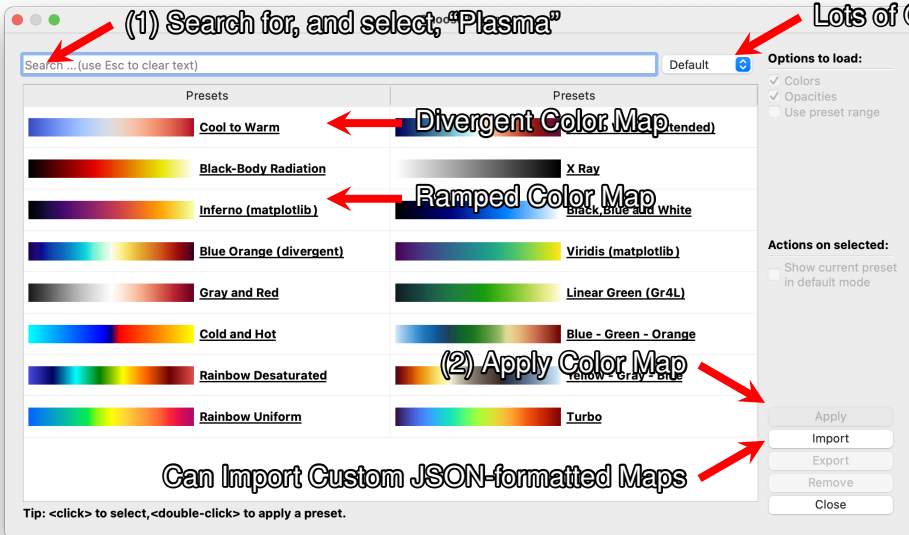


Logarithmically Scaled Coloring & Rename Legend

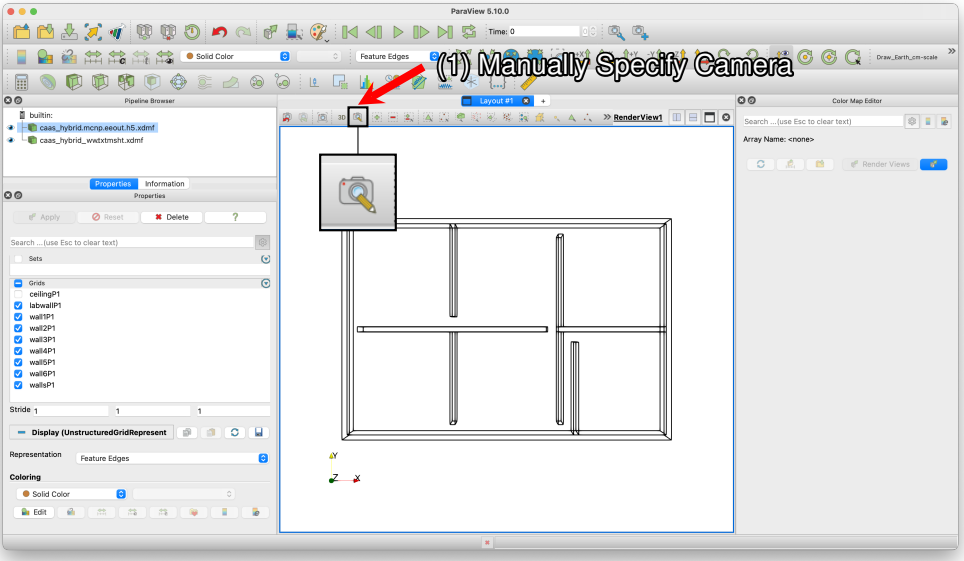


Select Edit 6 values to color by.
Units are calculated as gray from
MeV/g $\times 1.6022 \times 10^{-10}$ Gy · g/MeV $\times$ 2.9 fission^{-1} $\times 10^{15}$ fissions.

Color Map Selection

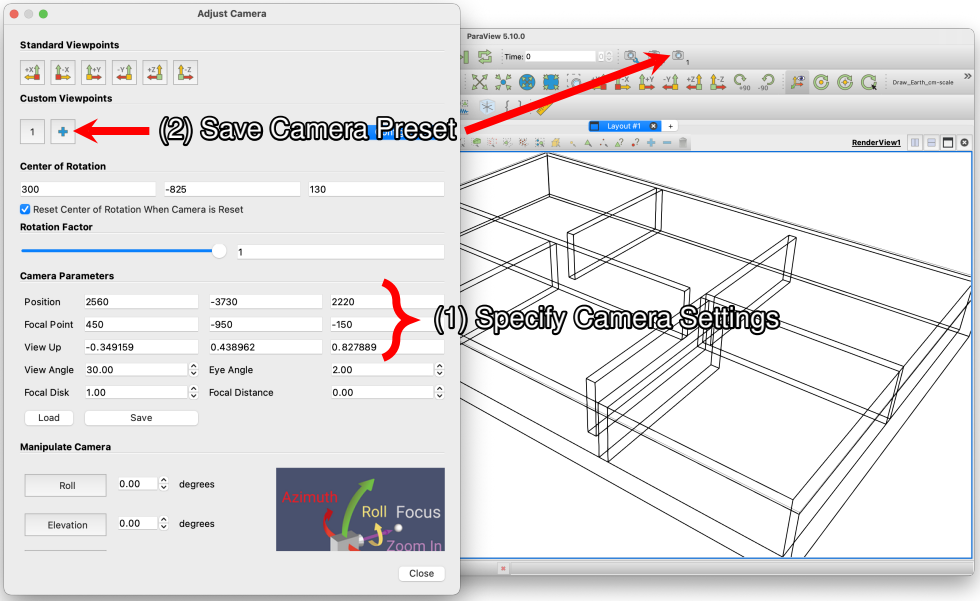


UM Feature-edges Only



Deselect ceiling grid, change UM representation to "Feature Edges", color as black, set line width to 3.

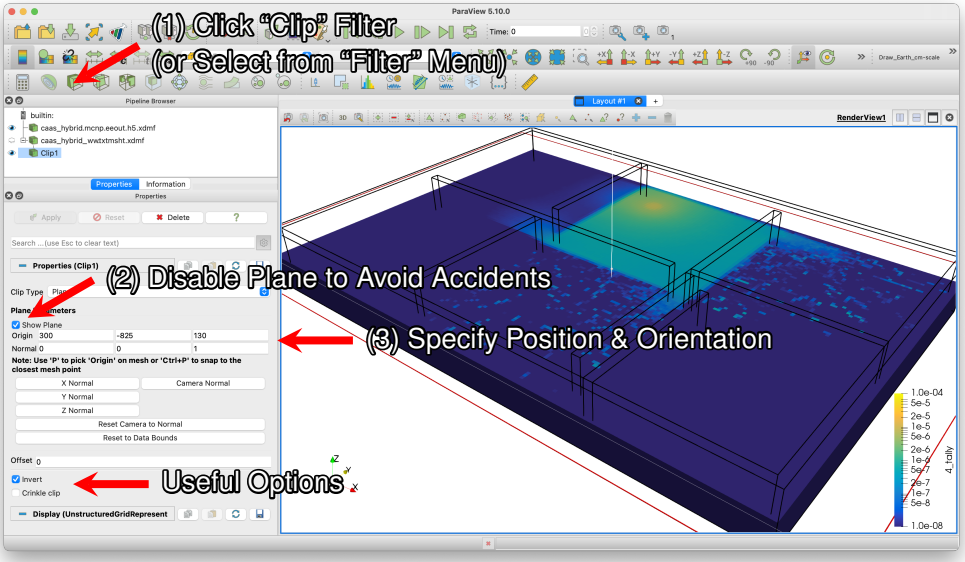
Specify Camera Settings



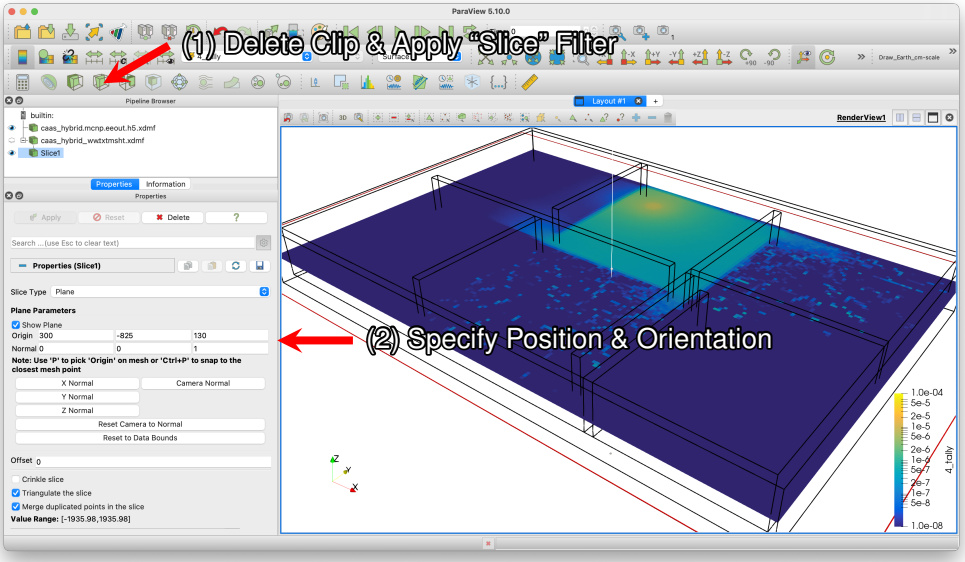
Operations of Interest

- Clips and slices
- Isocontours
  - Cell-to-point data conversion
  - Lighting considerations
  - Overlaying relative fractional uncertainties
  - Probing & manually annotating values
- Assessing mesh quality
  - Segregating and finding particular elements

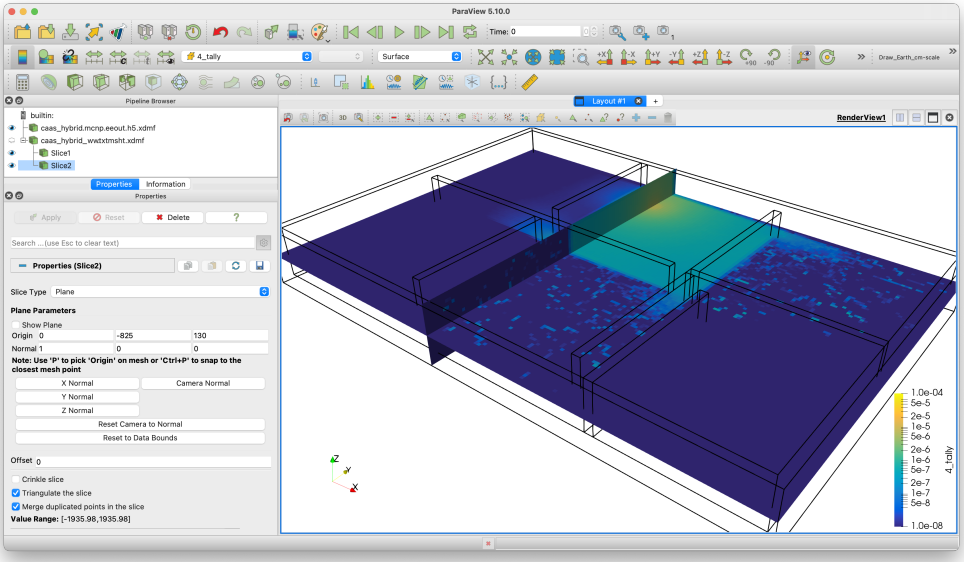
Clip (Hide Half Space)



Slice

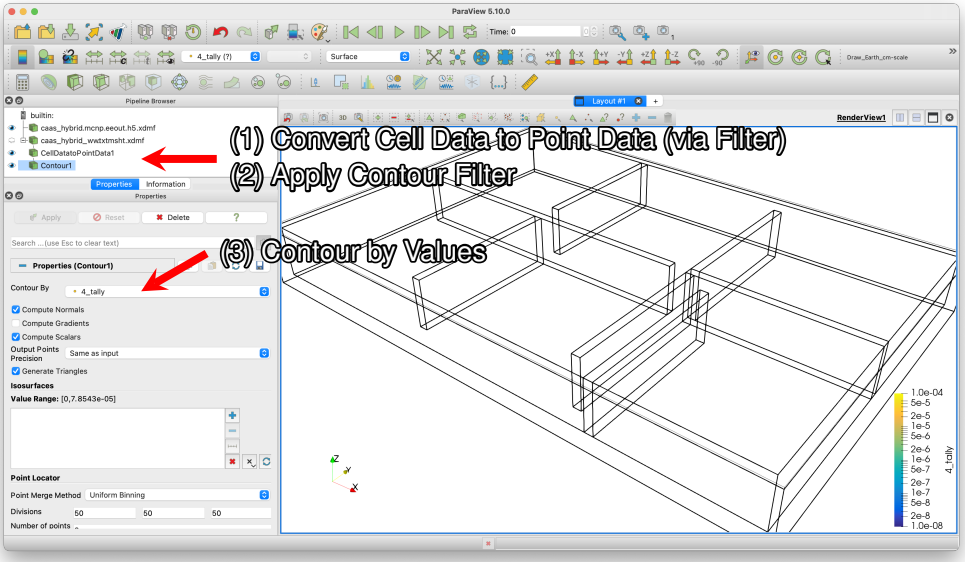

(1) Delete Clip & Apply "Slice" Filter
(2) Specify Position & Orientation
ParaView 5.10.0
Pipeline Browser
builtin:
caas_hybrid.mcnp.eeout.h5.xdmf
caas_hybrid_wwtxtmsht.xdmf
Slice1
Properties
Information
Apply
Reset
Delete
Search ...(use Esc to clear text)
Properties (Slice1)
Slice Type
Plane
Plane Parameters
Show Plane
Origin 300 -825 130
Normal 0 0 1
Note: Use 'P' to pick 'Origin' on mesh or 'Ctrl+P' to snap to the closest mesh point
X Normal
Camera Normal
Y Normal
Z Normal
Reset Camera to Normal
Reset to Data Bounds
Offset 0
Crinkle slice
Triangulate the slice
Merge duplicated points in the slice
Value Range: [-1935.98,1935.98]
Layout #1
RenderView1
1.0e-04
5e-5
2e-5
1e-5
5e-6
2e-6
1e-6
5e-7
2e-7
1e-7
5e-8
1.0e-08
4_tally

Another Slice ($x, y, z = 0, -825, 130 \perp x$)

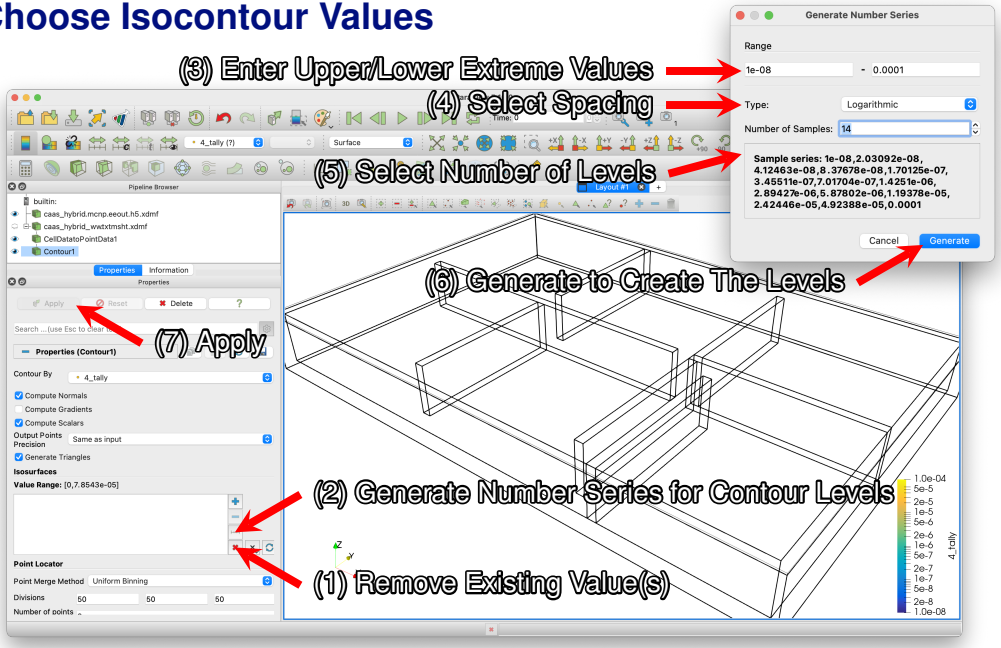


Isocontours & Lighting Considerations

Apply Contour Filter and “Contour By” Value

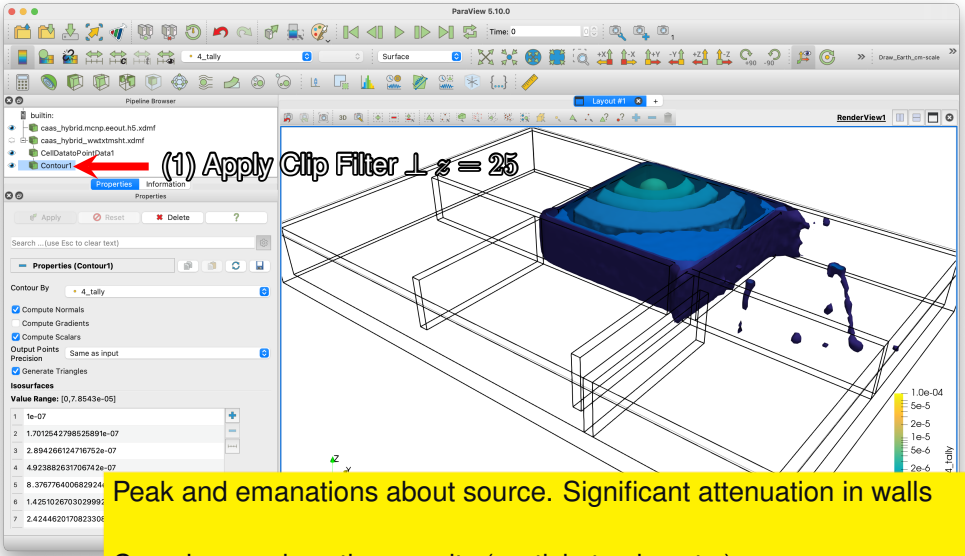


Choose Isocontour Values



Also adjust coloring by value to be consistent with contour range.

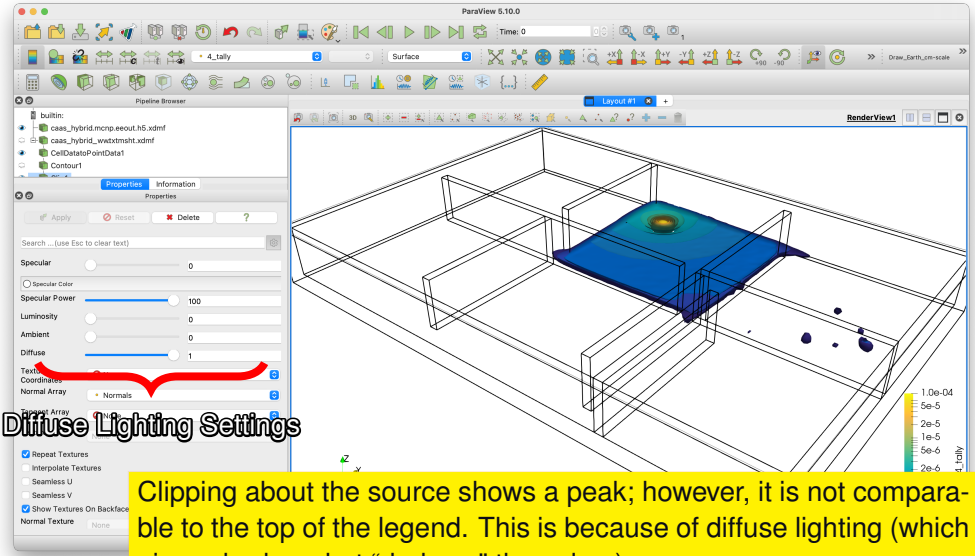
Observe Isocontour Results & Trends



Peak and emanations about source. Significant attenuation in walls

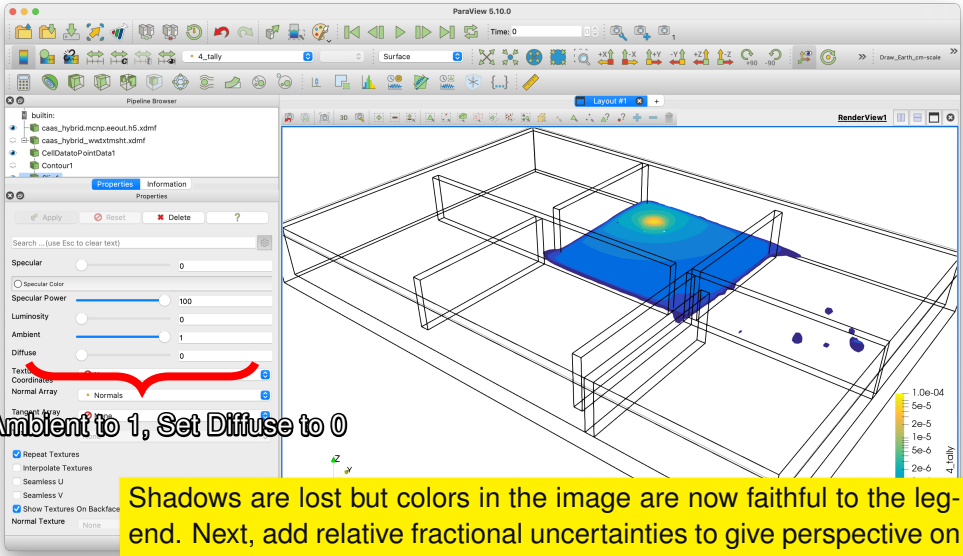
Can also overlay other results (particle tracks, etc.).

Clip Isocontour & Observe Shadows



Clipping about the source shows a peak; however, it is not comparable to the top of the legend. This is because of diffuse lighting (which gives shadows but “darkens” the colors).

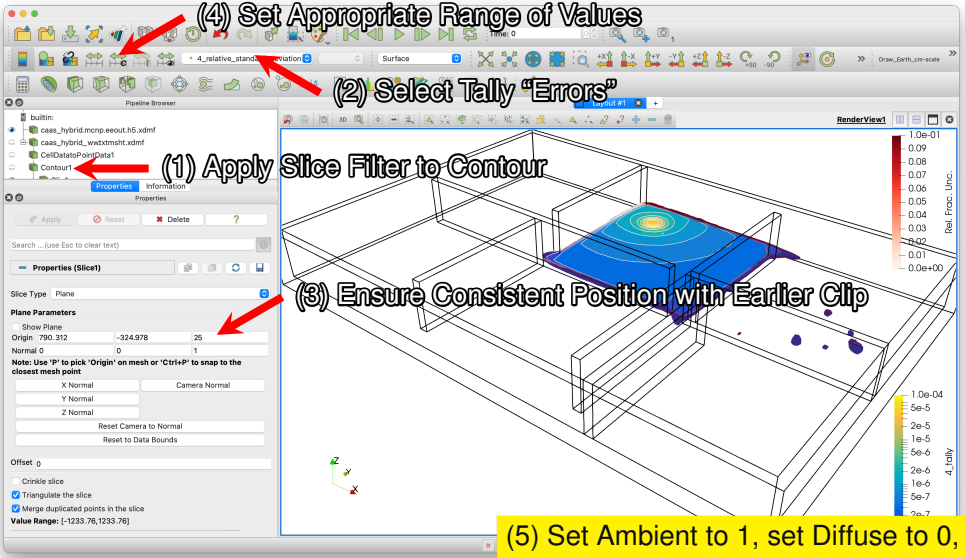
Adjust from Diffuse to Ambient Lighting



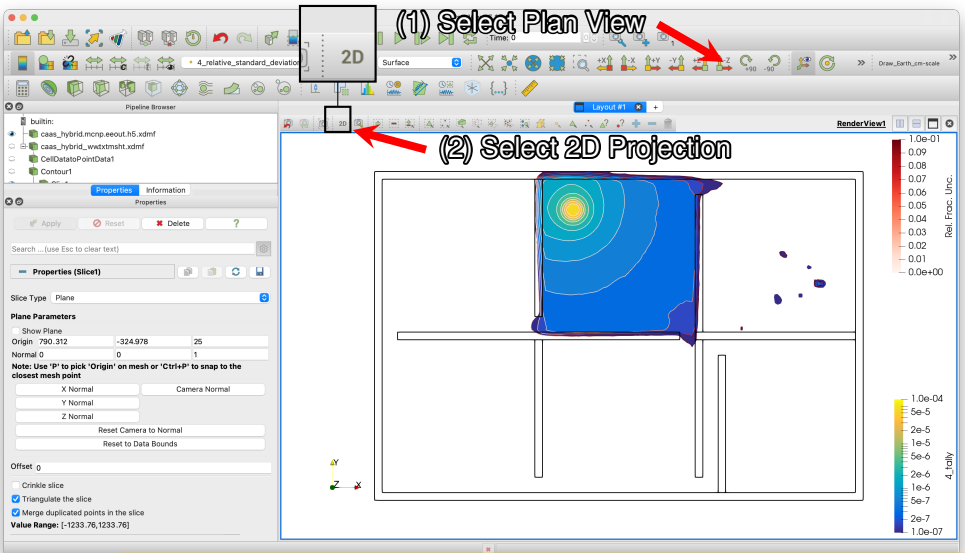
Set Ambient to 1, Set Diffuse to 0

Shadows are lost but colors in the image are now faithful to the legend. Next, add relative fractional uncertainties to give perspective on results.

Add Relative Fractional Uncertainties



2D Plan View



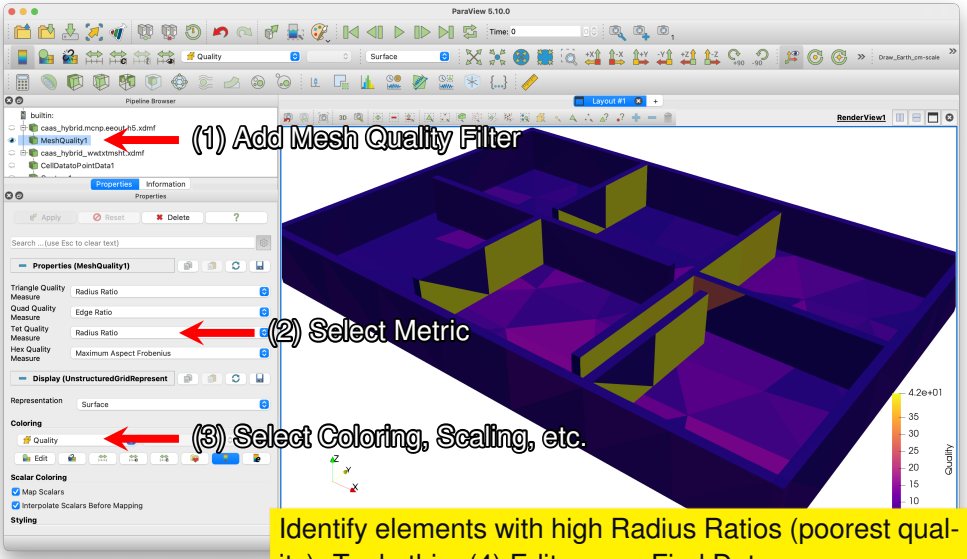
Gives a more “traditional” representation of geometry and results.

Assessing Mesh Quality

What is “Mesh Quality”

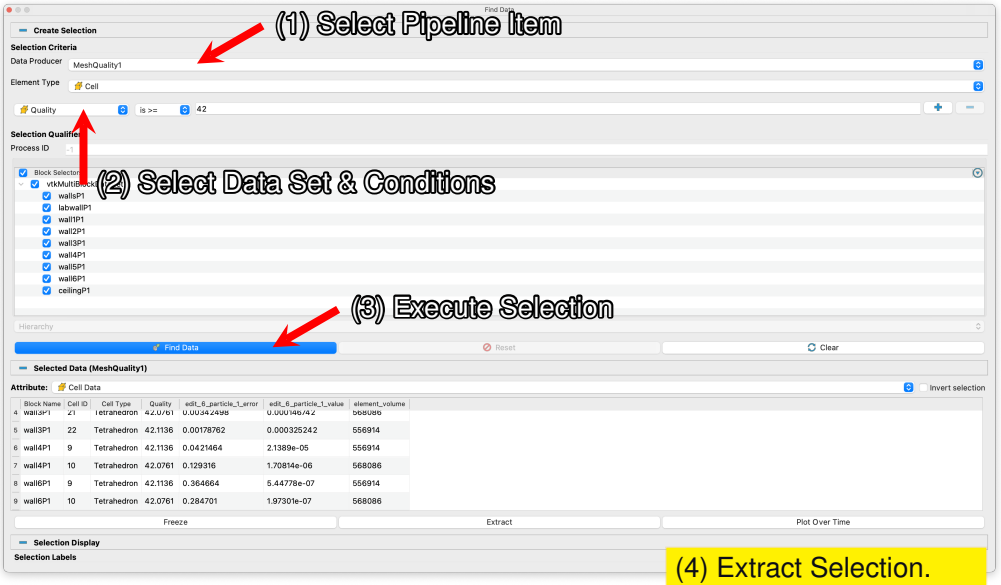
- Mesh element “quality” can be assessed by a variety of metrics
  - Deterministically calculated
  - Useful for mesh-generation algorithms
  - Useful for deterministic engineering calculations
- Body of knowledge mainly for linear tetrahedra and hexahedra
  - Verdict library [7] summarizes these metrics
  - ParaView incorporates Verdict
- Useful for Monte Carlo transport to identify undesirable elements
  - Negative-Jacobian elements have negative volume
    - Problematic for edits
  - “Paper-thin” elements can challenge particle tracking

Mesh Quality Filter & Finding Elements



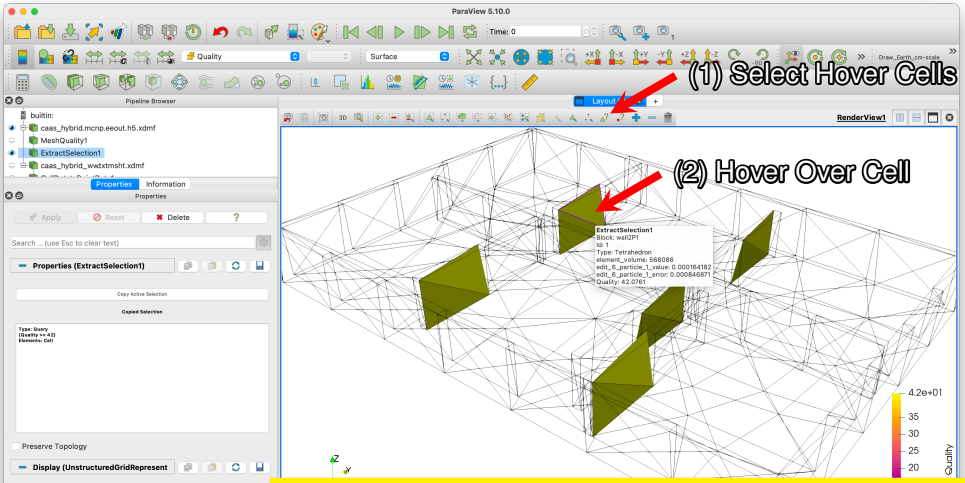
Identify elements with high Radius Ratios (poorest quality). To do this: (4) Edit menu: Find Data.

Find Data Dialog Use



Note that this dialog may need to be scrolled vertically to expose all elements.

Viewing & Interrogating Specific Elements



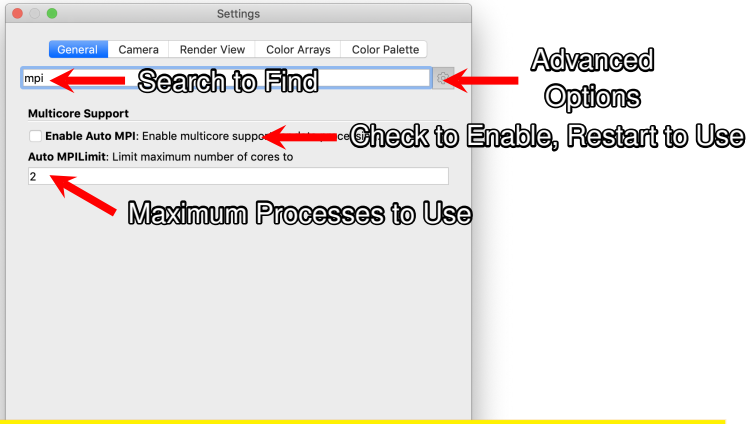
Information on the extracted data is in the Properties pane. One can also hover on points to get location information.

Other (Advanced?) ParaView Topics

Brief Overview of Other Topics

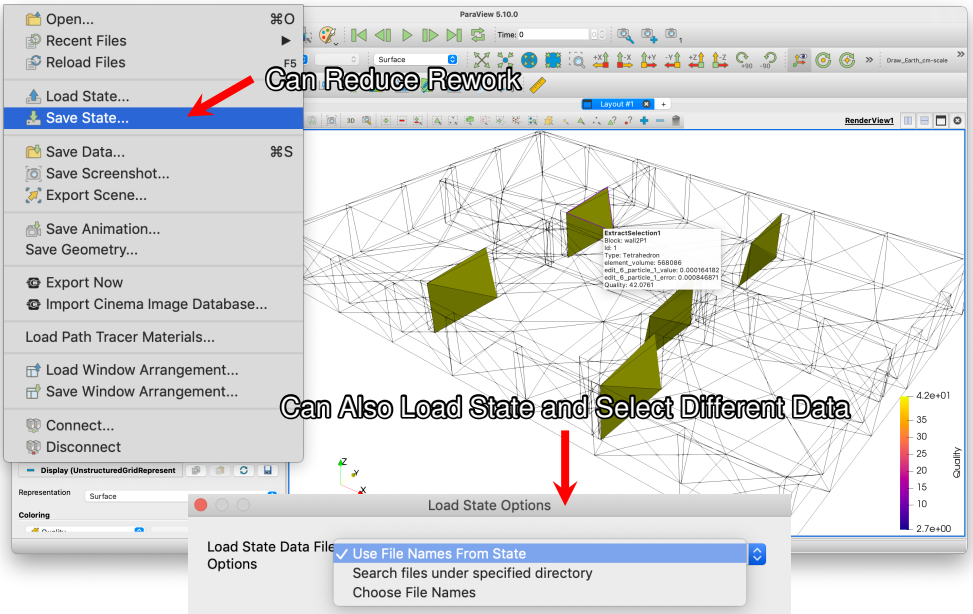
- MPI Parallelism
- Toggling Geometry via Blocks
- Saving ParaView State
- Camera Linking
- Recording Traces (Macros)
- Client-server Connections
- Animation

MPI Parallelism (Settings, General Tab)

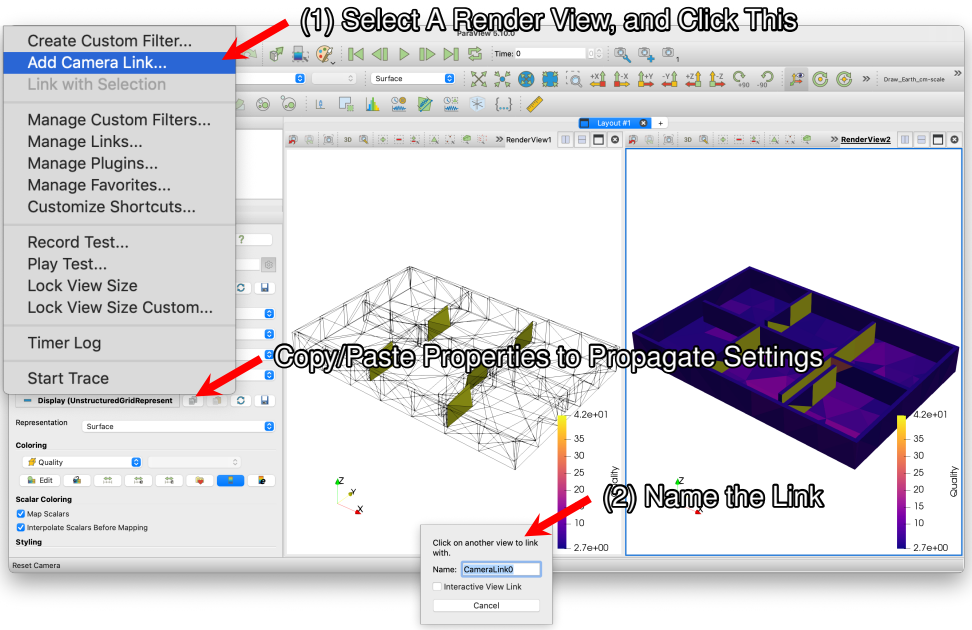


Requires MPI on your system (and running). Also, depending on your system firewall settings, you may see a number of MPI processes launch and be forced to dismiss allow/block requests. This means: start with a few processes to see how your system behaves.

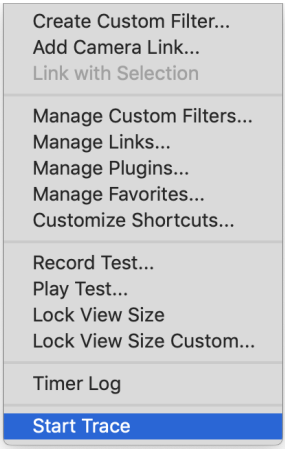
Saving ParaView State (File Menu)

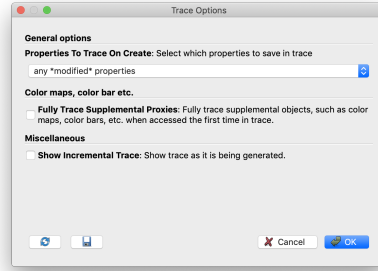


Camera Linking (Tools Menu)

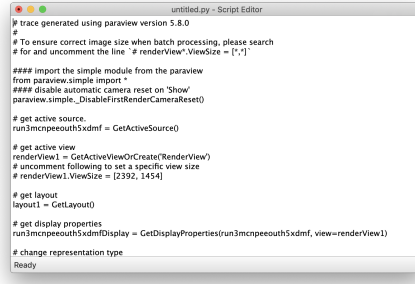


Recording Macros (Traces, Tools Menu)



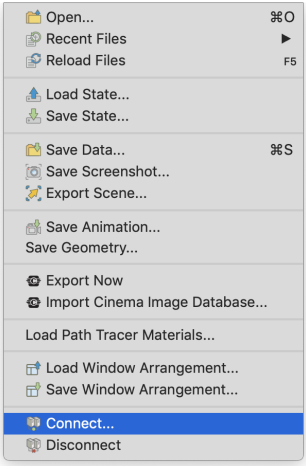


Stop Trace

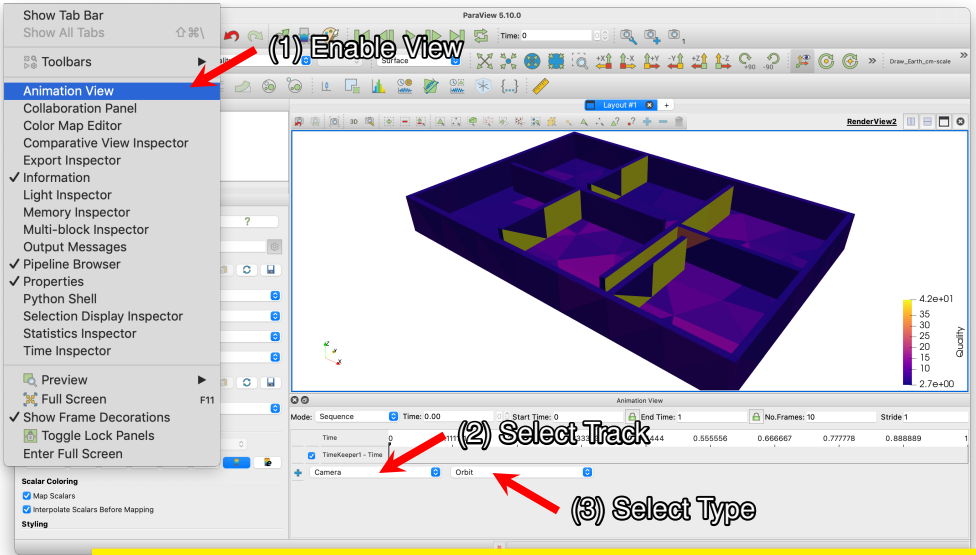


Client-server Connections

- Negates downloading data
- Provide parallel processing via MPI
- Institution-specific configuration
- LANL uses a PVSC configuration file
- https://hpc.lanl.gov/paraview_usage
- Remote memory monitoring
  - View: Memory Inspector



Animation



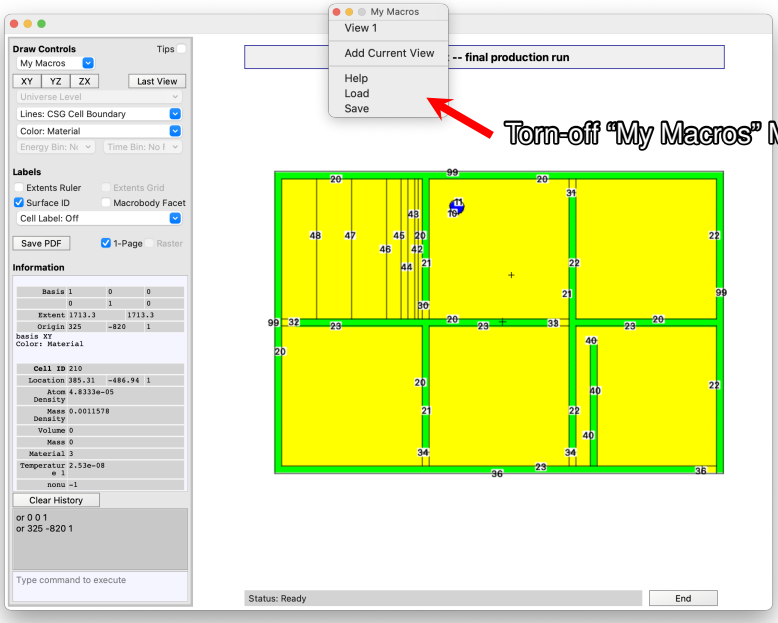
Python tracks can perform scripted actions every frame (every tick).

Qt Plotter Technology Preview

Qt Plotter Technology Preview

- Why a new plotter?
  - "Flat" interface elements: text drawn on a window
  - Missing "basic" features: command history, mouse click & drag
  - Less-than-optimal performance
  - Relies on X11, which is unconventional on MS Windows
- Notable new features
  - Mouse-based interaction for translation and zoom
  - Standard UI elements:
    - Buttons/menus for common tasks
    - Checkboxes to control *and* indicate state
  - Comprehensive cell-property information
  - Nested menus for particle-type queries
  - Direct button export to PDF file (also: `saveps`/`savepng`/`savepdf`)
  - Command history (i.e., use an up arrow to recall last typed command)
  - Customizable and retrievable view information via "My Macros"

Interface Overview



UM EEOUT Extreme-value Identifier

61_Users_mrising_xdocs_LANL_Presentations_2022_RPSD–Workshop_Seattle_vis_get_eeout_value_range.py

```python
#!/usr/bin/env python

"""A utility to find the range of values for MCNP UM edits."""

import sys

import h5py
import numpy as np


def get_eeout_size_range(filename):
    """Find range of EEOUT values."""

    class H5Datasets:
        """Provide recursive method and container to find/store dataset names."""

        def __init__(self):
            """Prepare to store names of datasets."""
            self.names = []

        def __call__(self, name, node):
            """If a dataset node is found, then append its name."""
            if isinstance(node, h5py.Dataset) and not name in self.names:
                self.names.append(name)

    datasets = H5Datasets()
    values = np.infty * np.ones(3)
    values[-1] *= -1  # Flip sign on last entry.
    with h5py.File(filename, "r") as myfile:
        myfile.visititems(datasets)

        for name in datasets.names:
            if "value" not in name:
                continue
            tmp = myfile[name][()]
            values[0] = np.min((np.min(tmp), values[0]))
            values[1] = np.min((np.min(tmp[np.nonzero(tmp)]), values[1]))
            values[2] = np.max((np.max(tmp), values[2]))
    return values[0], values[1], values[2]


infilename = sys.argv[1]
val_min, val_min_nonzero, val_max = get_eeout_size_range(infilename)

print(f"Maximum          value: {val_max:.3e}")
print(f"Minimum non-zero value: {val_min_nonzero:.3e}")
print(f"Minimum          value: {val_min:.3e}")
```

Runtape Repacking Script

62_Users_mrising_xdocs_LANL_Presentations_2022_RPSD–Workshop_Seattle_vis_repack_runtape.py

```python
#!/usr/bin/env python

"""A utility to repack datasets as compressed."""

import logging
import os
import subprocess
import sys

import h5py

logging.basicConfig(level=logging.INFO)


def repack_runtape(filename, compression="gzip", compression_opts=4):
    """Delete non-results groups and compress remaining datasets."""
    filesize = os.path.getsize(filename)
    logging.info(f"Original size of {filename}: {filesize} bytes.")

    class H5Datasets:
        """Provide recursive method and container to find/store dataset names."""

        def __init__(self):
            """Prepare to store names of datasets."""
            self.names = []

        def __call__(self, name, node):
            """If a dataset node is found, then append its name."""
            if isinstance(node, h5py.Dataset) and not name in self.names:
                self.names.append(name)

    datasets = H5Datasets()
    with h5py.File(filename, "a") as myfile:
        # Delete non-results and non-QA groups.
        for group in ["fixed", "header", "restart", "variable"]:
            if group in myfile.keys():
                del myfile[group]

        # Find and compress remaining datasets.
        myfile.visititems(datasets)
        for name in datasets.names:
            tmp = myfile[name][()]
            del myfile[name]
            myfile.create_dataset(
                f"{name}_compressed",
                data=tmp,
                compression=compression,
                compression_opts=compression_opts,
            )

    # Move names to be consistent with uncompressed datasets.
    with h5py.File(filename, "r+") as myfile:
        for name in datasets.names:
            myfile.move(f"{name}_compressed", name)

    return filesize


infilename = sys.argv[1]
original_size = repack_runtape(infilename)

try:
    outfilename = f"{infilename}_repacked"
    subprocess.run(["h5repack", infilename, outfilename], check=True)
    os.replace(outfilename, infilename)
except RuntimeError:
    logging.error("Failed to 'h5repack' {infilename} and then overwrite it.")

final_size = os.path.getsize(infilename)
logging.info(
    f"Final size of {infilename}: {final_size} bytes"
    f" ({final_size/original_size*100:.2f}% of original)."
)
```