LA-UR-19-20291
Approved for public release; distribution is unlimited.

Title: A Python Script to Convert MCNP Unstructured Mesh Elemental Edit Output Files to XML-based VTK Files

Author(s): Kulesza, Joel A.
McClanahan, Tucker Caden

Intended for: Report

Issued: 2019-11-13 (rev.2)

Disclaimer:
Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by Triad National Security, LLC for the National Nuclear Security Administration of U.S. Department of Energy under contract 89233218CNA000001. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

A Python Script to Convert MCNP Unstructured Mesh Elemental Edit Output Files to XML-based VTK Files

Joel A. Kulesza and Tucker C. McClanahan
Monte Carlo Methods, Codes, and Applications Group
X Computational Physics Division
Los Alamos National Laboratory

September 23, 2019

1 Introduction

This report provides a Python script to convert an MCNP® [1] ASCII unstructured mesh (UM) elemental edit output (EEOUT) file [2] geometry and results to an ASCII XML-based unstructured mesh VTK (`.vtu`) file [3, Section 19.3]. The resulting VTK file can be directly visualized with applications such as ParaView [4] and VisIt [5]. This report also describes accompanying verification work that shows the script performing as required. However, users of the enclosed script must still verify that the script is behaving correctly for their own work.

This report is organized as follows: Section 2 describes how to execute the enclosed script and its resulting output, Section 3 gives requirements of the script, Section 4 describes design and implementation considerations, and Section 5 describes verification testing performed to demonstrate that the requirements in Section 3 are met. The script itself is listed in Appendix A and provided as a PDF attachment for convenience. Select files used in testing are listed in Appendix B. For convenience, Appendix C provides several ParaView macros that the first author has found useful.

An early version of the enclosed script was distributed, to a limited extent, to MCNP UM analysts within Los Alamos National Laboratory. This document is the result of several requests for the script from the MCNP UM user community outside Los Alamos National Laboratory. This script is thus provided now as a stopgap measure until such functionality is provided by a compiled application that provides superior speed and a robust verification and validation basis (e.g., via `um_post_op` or MCNPTools [6]) or until MCNP UM output is produced in a format that can be directly visualized.

Revision 1 of this document incorporates changes to the script that gracefully handle NaN and extreme value ($10^{\pm 100}$ and beyond) edit quantities. If a NaN is encountered, a warning is issued and it is converted to 10^{308}. Furthermore, all Float32-valued VTK fields have been converted to Float64.

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. Any questions regarding licensing, proper use, and/or proper attribution of Triad National Security, LLC marks should be directed to trademarks@lanl.gov.

Revision 2 of this document incorporates changes to the script that handle the processing of energy and/or time bins along with their errors and totals. The result is a file with multiple datasets corresponding to each energy/time/error/total bin.

2 Script Usage

The enclosed script, named `Convert_MCNP_eeout_to_VTK.py`, is executed from the terminal command line followed by the filename of the EEOUT file as, for example:

```
python3 Convert_MCNP_eeout_to_VTK.py test.eeout
```

Output to the terminal will be of the form:

```
Processing test.eeout...
  Found 1 edit(s).
  Processing FLUX_4 edit...
    Processing & Validating EDIT_4_RESULT...
    Maximum          value: 2.86644e-01
    Minimum non-zero value: 4.78513e-07
    Minimum          value: 0.00000e+00
```

The label `EDIT_4_RESULT` is expanded to the name of the resulting dataset including time/energy identifiers. The maximum, minimum non-zero, and minimum values are provided for two reasons. First, the maximum and minimum non-zero values give appropriate bounds for setting color scales when log-scaling is used. Second, one can use the minimum value as a sanity check. If the minimum is non-zero, then it should be the same as the minimum non-zero value. If the minimum is zero, some elements may not have had particles enter them so the user should confirm that results are appropriate. In this example, the EEOUT file name is `test.eeout` so the resulting ASCII XML-based unstructured mesh VTK will be automatically named `test.eeout.vtu`. If this file exists, it will be overwritten without warning.

3 Functional and Performance Requirements

The functional requirements for this script are:

1. The script shall be able to convert arbitrary combinations of parts containing first- and second-order tetrahedral, pentahedral, and hexahedral geometry elements from the EEOUT file produced by MCNP code version 6.2 to an XML-based unstructured mesh VTK (`.vtu`) file.
2. The resulting `.vtu` file shall be ASCII, serial-formatted, and entirely self-contained. As such, it can be readily interrogated by a user with a text editor and loaded in a visualization application such as ParaView or VisIt without additional post-processing. This script is only required to work with ASCII EEOUT files, which is the default output format.
3. The script shall be able to convert element-wise edits and optionally enabled corresponding relative uncertainties from EEOUT to `.vtu`.

4. The script shall be able to convert element-wise volume, density, and materials from EEOUT to `.vtu`.
5. The script shall be able to convert element-wise edits that are binned in energy and/or time along with their corresponding relative uncertainties and totals from EEOUT to `.vtu`.

There are no firm performance requirements. However, this script should be able to process EEOUT files containing millions of elements without undue slowness.

4 Design and Implementation

Because this script was developed for personal use by the first author, speed of development and flexibility were informal design and implementation requirements. Accordingly, the Python language was selected and an evolutionary development approach was used. The script does not leverage object-oriented functionality; however, it is recommended that a data structure be constructed to represent the EEOUT information in the future. As the script was used, it was checked by the authors on a case-by-case basis. This report is the first time substantial verification is performed. Despite the verification work in this report, it is incumbent on the user of the enclosed script to still verify that results are correct on a case-by-case basis.

This script is Python 3 compliant (most recently executed with Python version 3.7.2 provided by Homebrew[1] on macOS version 10.12.6). Furthermore, it was an informal goal that minimal Python modules be required so the script would be as portable as possible. As such, this script only depends on the `os`, `re`, and `sys` modules. Because of the Python 3 compliance and minimal number of module dependencies, the enclosed script should work with any recent Python 3 interpreter on any operating system.

5 Testing

A collection of 216 individual calculations is used to verify that this script fulfills requirements 1–4 in Section 3. The 216 separate MCNP calculations consist of all permutations of UM cells composed of first- and second-order tetrahedral, pentahedral, and hexahedral elements within a three pseudo-cell analysis.

Each calculation features a UM three-cell geometry composed of a $50 \times 50 \times 50$-mm cube with an adjacent right-triangular prism and semicylindrical cap. The geometry is shown in Fig. 1. The cube is composed of ^{252}Cf at 15.1 g·cm^{-3} with fission disabled (with MCNP input: `nonu`), the triangular prism is ^{27}Al at 2.6989 g·cm^{-3}, and the cylindrical cap is water at 0.998207 g·cm^{-3}. A spontaneous-fission neutron source is distributed throughout the cube using the MCNP UM `volumer` capability and is assigned a ^{252}Cf Watt fission spectrum (with MCNP input: `sp1 -3 1.180000 1.03419`). An example input file is given in Listing 2.

Four elemental edit outputs are specified with `embee` cards:

```
embee4:n embed=1
embee14:n embed=1 errors=yes
embee6:n embed=1
embee16:n embed=1 errors=yes
```

[1] https://brew.sh/

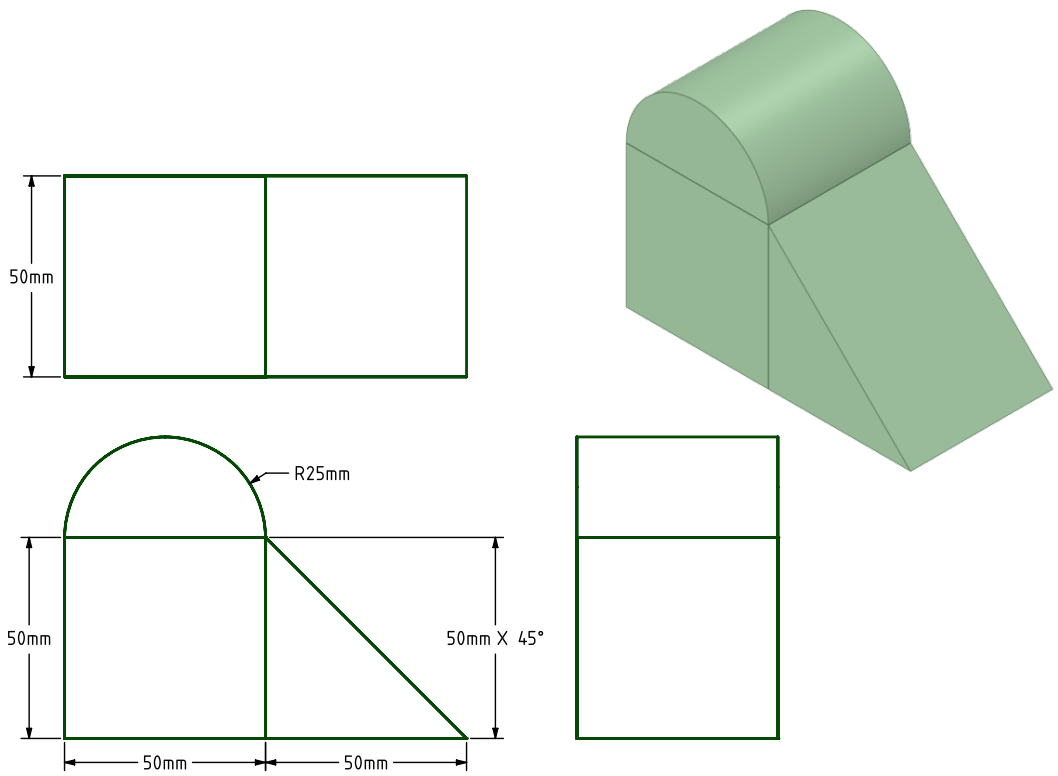


Figure 1: Test Case Geometry

This collection of elemental edits provides UM element-wise F4- and F6-type edits (corresponding to track-length and energy-deposition tallies) both with and without associated relative uncertainties in the resulting EEOUT file. By default, element-wise material, volume, and density values are available in the EEOUT file.

Following each MCNP calculation, three steps are used to verify that requirements 1–4 are met. First, each EEOUT file is converted to a `.vtu` file using the script in Listing 1 [Req. 1]. Next, each `.vtu` is loaded in ParaView [Req. 2]. Finally, the element-wise results [Req. 3], associated relative uncertainties (if applicable) [Req. 3], material numbers [Req. 4], densities [Req. 4], and volumes [Req. 4] are visualized. Examples of the visualized quantities are given in Fig. 2 for a variety of element-type combinations.

In order to test the energy and/or time binning functionality of this script, 6 individual calculations are used to verify that the script fulfills requirement 5 in Section 3. The 6 calculations include all permutations of time and energy binning, with and without relative errors. Each test calculation includes a $1 \times 1 \times 1$-cm cube represented by 8 hexahedral mesh elements, and the cube is centered about the origin. The geometry is shown in Fig. 3. The cube is composed of ^{27}Al at 2.7 g·cm^{-3}. A point source of neutrons located at the point (12.071, 12.071, 12.071) with arbitrary energy and time distributions that are detailed in Appendix B.3. A similar procedure to verify the energy/time binning test cases as stated above was followed for each test case. Fig. 4 shows some examples of energy and time bins plotted in ParaView.

5.1 Comments on Performance

This script is also tested on a UM terrain geometry that contains approximately 10 million elements (with approximately 3 million nodes) with 2 elemental edits and required up to 16 GB of free RAM. The conversion process from EEOUT to VTK takes approximately 4 minutes. Loading and viewing the resulting `.vtu` file in ParaView takes approximately 1 minute. The resulting geometry shaded by UM element volume is shown in Fig. 5. This level of performance is deemed acceptable.

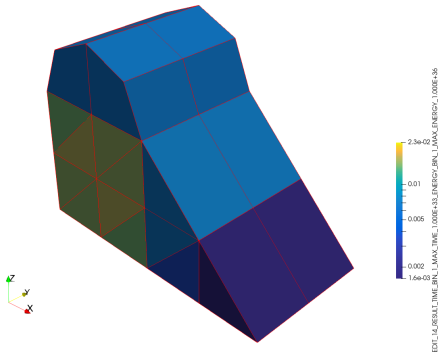
6 Conclusions

The test case results given in Section 5 demonstrate that the script fulfills the requirements stipulated in Section 3. While verification of the script is still incumbent on the user on a case-by-case basis, this document suggests that it is implemented correctly and should properly convert MCNP code version 6.2 elemental edit output files to XML-based unstructured VTK (`.vtu`) files.

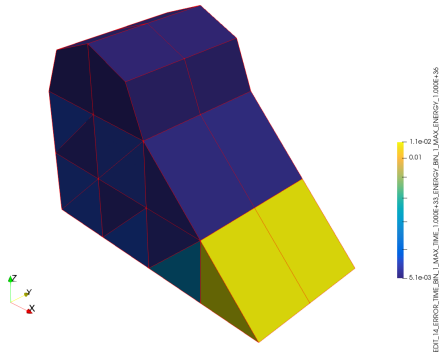
References

[1] C. J. Werner, J. Armstrong, F. B. Brown, J. S. Bull, L. Casswell, L. J. Cox, D. Dixon, R. A. Forster, J. T. Goorley, H. G. Hughes, J. Favorite, R. Martz, S. G. Mashnik, M. E. Rising, C. J. Solomon, A. Sood, J. E. Sweezy, A. Zukaitis, C. Anderson, J. S. Elson, J. W. Durkee, R. C. Johns, G. W. McKinney, G. E. McMath, J. S. Hendricks, D. B. Pelowitz, R. E. Prael, T. E. Booth, M. R. James, M. L. Fensin, T. A. Wilcox, and B. C. Kiedrowski, "MCNP User's Manual, Code Version 6.2," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-17-29981, Oct. 2017. URL: http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-17-29981

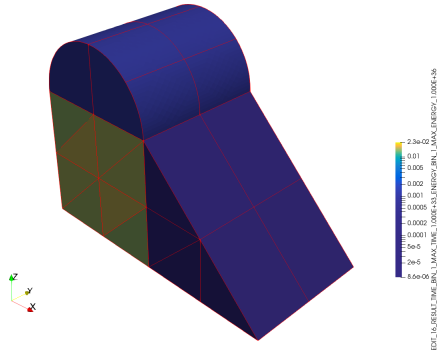
[2] R. L. Martz, "The MCNP6 Book On Unstructured Mesh Geometry: User's Guide For MCNP 6.2,"



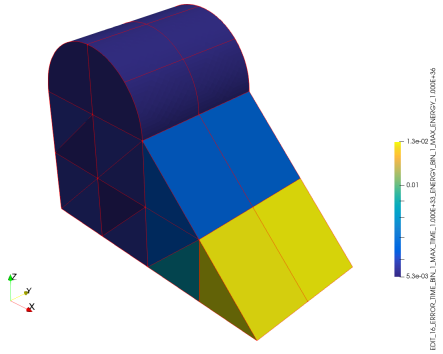
(a) embee14 Results, Linear Elements



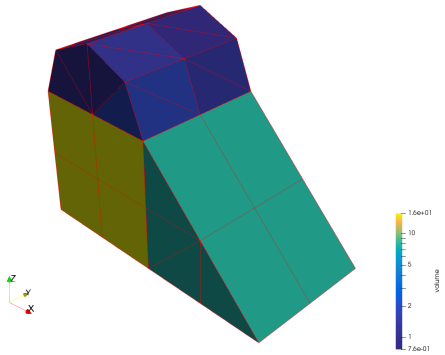
(b) embee14 Uncertainties, Linear Elements



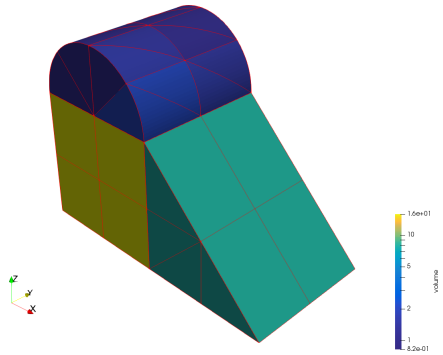
(c) embee16 Results, Quadratic Elements



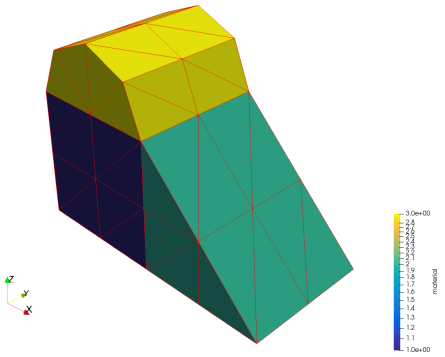
(d) embee16 Uncertainties, Quadratic Elements



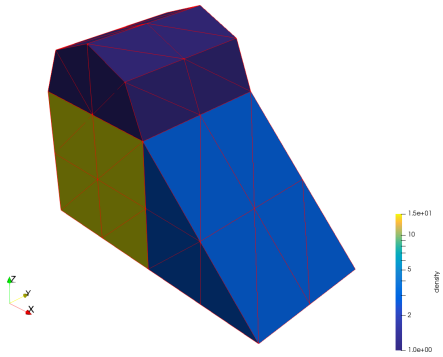
(e) Elemental Volumes, Linear Elements



(f) Elemental Volumes, Quadratic Elements



(g) Elemental Material Numbers



(h) Elemental Mass Densities

Figure 2: Example Visualization Results

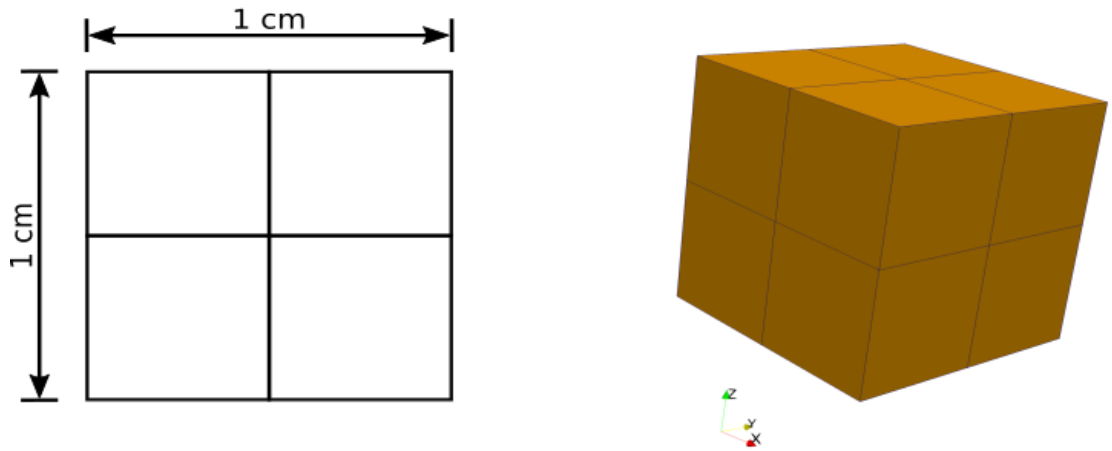
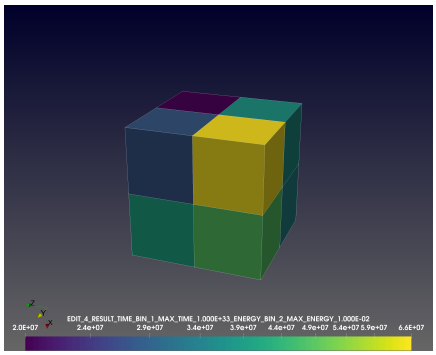
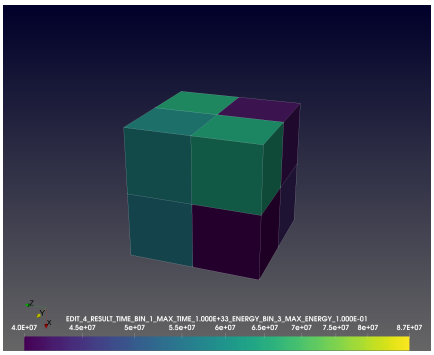


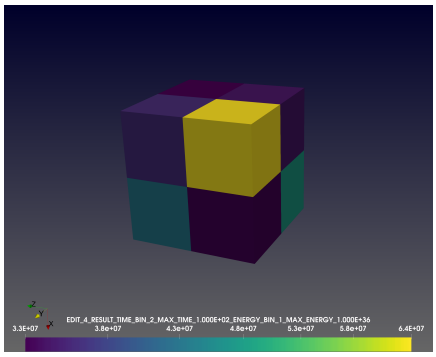
Figure 3: Binning Test Case Geometry



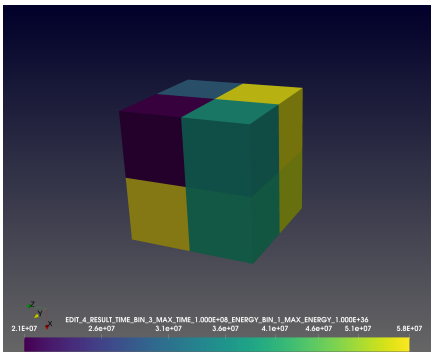
(a) Energy Bin 2 Results



(b) Energy Bin 3 Results



(c) Time Bin 2 Results



(d) Time Bin 3 Results

Figure 4: Example Visualization Results for Energy/Time Bins

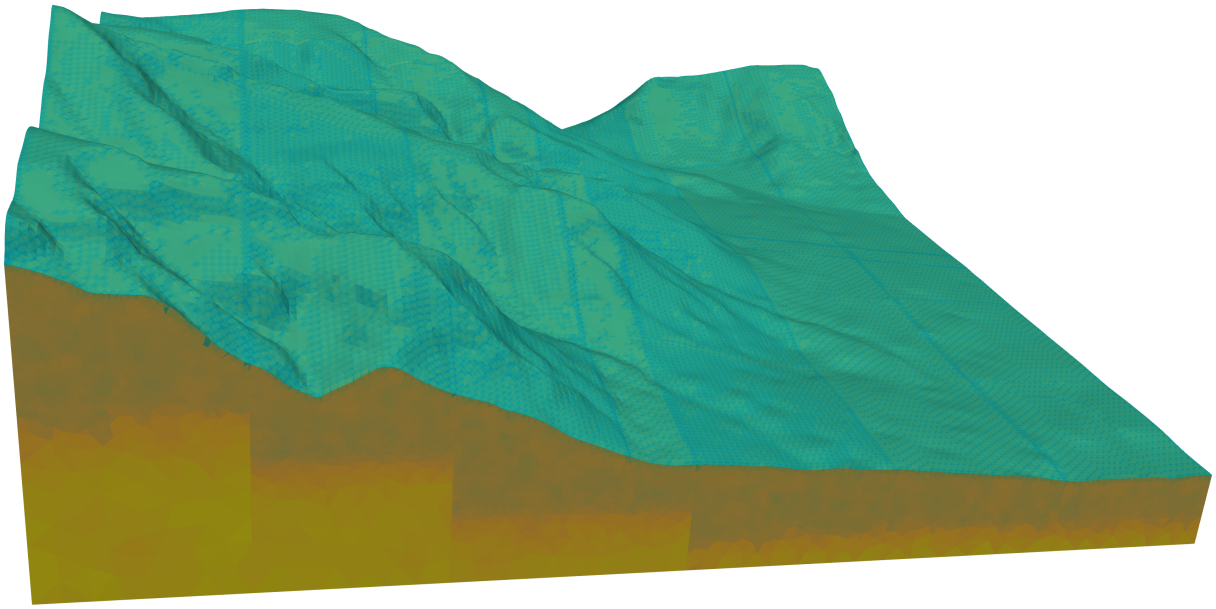


Figure 5: UM Terrain Demonstration, Shaded by UM Element Volume, ~10 Million Elements

Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-17-22442, Mar. 2017. URL: http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-17-22442

[3] L. S. Avila, U. Ayachit, S. Barré, J. Baumes, F. Bertel, R. Blue, D. Cole, D. DeMarle, B. Geveci, W. A. Hoffman, B. King, K. Krishnan, C. C. Law, K. M. Martin, W. McLendon, P. Pebay, N. Russell, W. J. Schroeder, T. Shead, J. Shepherd, A. Wilson, and B. Wylie, *The VTK User's Guide*, 11th ed. Kitware, Inc., 2010. URL: http://www.vtk.org/vtk-users-guide/

[4] U. Ayachit, *The ParaView Guide*, community ed., L. Avila, K. Osterdahl, S. McKenzie, and S. Jordan, Eds. Kitware, Inc., Jun. 2018. URL: https://www.paraview.org/paraview-guide/

[5] "VisIt User's Manual," Lawrence Livermore National Laboratory, Livermore, CA, USA, Tech. Rep. UCRL-SM-220449, Oct. 2005. URL: https://wci.llnl.gov/simulation/computer-codes/visit/manuals

[6] C. J. Solomon, C. R. Bates, and J. A. Kulesza, "The MCNPTools Package: Installation and Use," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-17-21779, Mar. 2017.

[7] J. L. Alwin, J. B. Spencer, and G. A. Failla, "Criticality Acccident Alarm System (CAAS) CSG-UM Hybrid Example," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-18-24235, May 2018. URL: http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-18-24235

A Script Source Code

The source code for the script described herein is given in Listing 1. For convenience, it is also provided as an attachment to this PDF, which can be accessed using Adobe Acrobat through the menu path shown in Fig. 6.

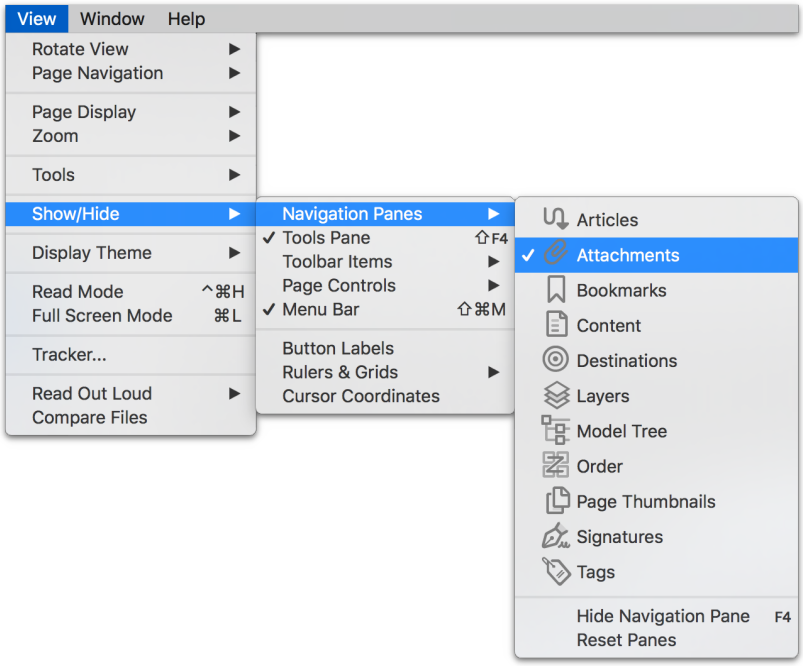


Figure 6: Adobe Acrobat Menu Path to Access PDF Attachments

Listing 1: Script Source Code

```python
#!/usr/bin/env python
#
# Execute as: Convert_MCNP_eeout_to_VTK.py <file.eeout>
#
# Code:     Convert_MCNP_eeout_to_VTK, version 1.2.0
#
# Authors:  Joel A. Kulesza (jkulesza@lanl.gov)
#           Tucker C. McClanahan (tcmcclan@lanl.gov)
#           Monte Carlo Methods, Codes & Applications
#           X Computational Physics Division
#           Los Alamos National Laboratory
#
# Copyright (c) 2019 Triad National Security, LLC.  All rights reserved.
#
# This material was produced under U.S. Government contract 89233218NCA000001
# for Los Alamos National Laboratory, which is operated by Triad National
# Security, LLC for the U.S. Department of Energy.  The Government is granted
# for itself and others acting on its behalf a paid-up, nonexclusive,
# irrevocable worldwide license in this material to reproduce, prepare
# derivative works, and perform publicly and display publicly.  Beginning five
# (5) years after February 14, 2018, subject to additional five-year worldwide
# renewals, the Government is granted for itself and others acting on its behalf
# a paid-up, nonexclusive, irrevocable worldwide license in this material to
# reproduce, prepare derivative works, distribute copies to the public, perform
# publicly and display publicly, and to permit others to do so.  NEITHER THE
# UNITED STATES NOR THE UNITED STATES DEPARTMENT OF ENERGY, NOR TRIAD NATIONAL
# SECURITY, LLC, NOR ANY OF THEIR EMPLOYEES, MAKES ANY WARRANTY, EXPRESS OR
# IMPLIED, OR ASSUMES ANY LEGAL LIABILITY OR RESPONSIBILITY FOR THE ACCURACY,
# COMPLETENESS, OR USEFULNESS OF ANY INFORMATION, APPARATUS, PRODUCT, OR PROCESS
# DISCLOSED, OR REPRESENTS THAT ITS USE WOULD NOT INFRINGE PRIVATELY OWNED
# RIGHTS.

import os
import re
import sys

# Find the start and end positions for the first matching sublist in a list.
def find_sublist( sl, l ):
    sll=len( sl )
    for ind in ( i for i,e in enumerate( l ) if e == sl[ 0 ] ):
        if( l[ ind:ind+sll ] == sl ):
            return ind,ind+sll-1

# Extract a certain number of entries (length) following a sublist (sublist) for
# a given input list (inlist).
```

```python
def extract_sublist_entries( sublist, length, inlist ):
    lp1 = find_sublist( sublist, inlist )[ 1 ] + 1
    lp2 = lp1 + length
    return inlist[ lp1:lp2 ]

# Calculate the connected nodes for various element types.
def calculate_connectivity_list_length( e_types ):
    connectivity_list_length = []
    for e in e_types:
        if( e == '4' ):
            connectivity_list_length.append( 4 )
        elif( e == '5' ):
            connectivity_list_length.append( 6 )
        elif( e == '6' ):
            connectivity_list_length.append( 8 )
        elif( e == '14' ):
            connectivity_list_length.append( 10 )
        elif( e == '15' ):
            connectivity_list_length.append( 15 )
        elif( e == '16' ):
            connectivity_list_length.append( 20 )
    return connectivity_list_length

# Convert eeout element types to VTK element types.
def calculate_vtk_e_types( e_types ):
    vtk_e_types = []
    for e in e_types:
        if( e == '4'):
            vtk_e_types.append( '10' )
        elif( e == '5' ):
            vtk_e_types.append( '13' )
        elif( e == '6' ):
            vtk_e_types.append( '12' )
        elif( e == '14' ):
            vtk_e_types.append( '24' )
        elif( e == '15' ):
            vtk_e_types.append( '26' )
        elif( e == '16' ):
            vtk_e_types.append( '25' )
    return vtk_e_types

# Create flat list of 3D vertices from individual coordinate lists.
def create_vertices( xs, ys, zs ):
    v = []
    for n,x in enumerate( xs ):
        v.append( xs[ n ] )
```

```
        v.append( ys[ n ] )
        v.append( zs[ n ] )
    return(v)

# Reformat list to print its elements nicely within the XML file.
def pretty_print_list( indent, cols, colwidths, inlist ):
    pretty_list_string = indent * ' '
    for n,i in enumerate( inlist ):
        pretty_list_string += str( i ).rjust( colwidths ) + ' '
        if( n % cols == cols - 1 ):
            pretty_list_string += '\n' + indent * ' '
    pretty_list_string += '\n'
    return pretty_list_string

# Perform various sanity checks on edit results.
def perform_edit_checks( edit_values, total_elements, check_gap = True ):
    found_negative = False
    found_nan = False
    max_val = -1e308
    min_nz_val = 1e308
    min_val = 1e308

    # Custom float conversion for Fortran-formatted numbers missing an "e" and
    # with three digits in the exponent.
    def floatf( x ):
        try:
            rv = float( x )
        except:
            rv = float( x[0:-4] + 'e' + x[-4:] )
        return rv

    edit_values = [ floatf(i) for i in edit_values ]

    if( len( edit_values ) == total_elements + 1 ):
        gap_value = edit_values[ 0 ]
        if( gap_value > 0 and check_gap ):
            print( 'WARNING: gap value: {:}'.format( gap_value ) )
    else:
        print( 'ERROR: Unexpected edit length, exiting' )
        exit()

    # The first edit entry is for gaps --- discard for plotting.
    edit_values = edit_values[ 1: ]
    for ev in edit_values:
        if( ev < 0 and found_negative == False ):
            print( 'WARNING: Negative edit entry found.' )
```

```
            found_negative = True
        if( ev != ev and found_nan == False ):
            print( 'WARNING: NaN edit entry found.  Setting to 1e308.' )
            found_nan = True
        if( ev < min_nz_val and ev > 0.0 ): min_nz_val = ev
        if( ev < min_val ): min_val = ev
        if( ev > max_val ): max_val = ev

    if( found_nan == True ):
        edit_values = [ 1e308 if ev != ev else ev for ev in edit_values  ]

    print( '    Maximum          value: {:.5e}'.format( max_val ) )
    print( '    Minimum positive value: {:.5e}'.format( min_nz_val ) )
    print( '    Minimum          value: {:.5e}'.format( min_val ) )

    edit_values = [ '{:.5e}'.format( i ) for i in edit_values ]

    return edit_values

# Separate into list and parse into results and relative uncertainties, if
# appropriate.
def get_results( edit_values, edit_number, total_elements ):
    edit_results = []
    tmp = edit_values.split('DATA SETS')[1:]
    edit_sets = [tmp[i] for i in range(len(tmp)) if 'RESULT SQR TIME BIN' not in tmp[i]]
    sublists = [ ['RESULT', 'TIME'] if 'RESULT TIME' in edit_sets[i] else ['REL', 'ERROR', 'TIME'] for i in range(len(edit_sets))]
    count = -1
    for s in sublists:
        count += 1
        edit_data = extract_sublist_entries( s, total_elements + 1 + 26, edit_sets[count].split() )
        # Get supplemental edit-identifying information.
        time_bin   = edit_data[2]
        time_value = edit_data[7]
        erg_bin    = edit_data[16]
        erg_value  = edit_data[21]

        # Construct unique name.
        edit_name = 'EDIT_{:}_{:}_TIME_BIN_{:}_MAX_TIME_{:}_ENERGY_BIN_{:}_MAX_ENERGY_{:}'.format( \
            edit_number, s[-2], time_bin, time_value, erg_bin, erg_value )

        # Extract only edit data values and validate.
        edit_data = edit_data[26:]
        print( '    Processing & Validating {:}...'.format( edit_name ) )
        check_gap = (not 'ERROR' in s ) # Don't check gap for error arrays.
        edit_data = perform_edit_checks( edit_data, total_elements, check_gap )
        edit_results.append( [ edit_name, edit_data ] )
```

```python

    return edit_results

##############################################################################

import __main__ as main
if(__name__ == '__main__' and hasattr(main, '__file__')):

    # Validate command line arguments.
    if( len( sys.argv ) != 2 ):
        print( 'ERROR: Incorrect number of command line arguments provided ('
            + str( len( sys.argv ) ) + '); those provided:' )
        print( sys.argv )
        exit()

    if( not os.path.isfile( sys.argv[ 1 ] ) ):
        print( 'ERROR: MCNP EEOUT file not found.' )
        exit()

    infilename = sys.argv[1]

    print( 'Processing {:}...'.format( infilename ) )

    with open ( infilename, 'r' ) as myfile:
        eeout = myfile.read()

    # Determine number of nodes and cells.
    nodes  = int( re.search( r'NUMBER OF NODES\s*:\s+(\d+)',     eeout ).group( 1 ) )
    tets1  = int( re.search( r'NUMBER OF 1st TETS\s*:\s+(\d+)',  eeout ).group( 1 ) )
    pents1 = int( re.search( r'NUMBER OF 1st PENTS\s*:\s+(\d+)', eeout ).group( 1 ) )
    hexs1  = int( re.search( r'NUMBER OF 1st HEXS\s*:\s+(\d+)',  eeout ).group( 1 ) )
    tets2  = int( re.search( r'NUMBER OF 2nd TETS\s*:\s+(\d+)',  eeout ).group( 1 ) )
    pents2 = int( re.search( r'NUMBER OF 2nd PENTS\s*:\s+(\d+)', eeout ).group( 1 ) )
    hexs2  = int( re.search( r'NUMBER OF 2nd HEXS\s*:\s+(\d+)',  eeout ).group( 1 ) )
    total_elements = tets1 + pents1 + hexs1 + tets2 + pents2 + hexs2

    # Retrieve edit information.
    edit_list = re.findall( r'(DATA OUTPUT PARTICLE.*?)\n', eeout, re.S )

    print( '  Found {:} edit(s).'.format( len( edit_list ) ) )

    # Capture edit data for use later.
    eeout_edits = re.search( r'(DATA OUTPUT.*?)CENTROIDS', eeout, re.S ).group( 1 )

    # Capture header information.
    eeout_header = re.search( r'(.*?)NODES X', eeout, re.S ).group( 1 )
```

```
    eeout_header = re.sub( r'\s+\n', '\n', eeout_header )
    eeout_header = re.sub( r'\n+', '\n', eeout_header )
    eeout_header = re.sub( r'^', '#  ', eeout_header )
    eeout_header = re.sub( r'\n', '\n#  ', eeout_header )

    # Reformat eeout to list to permit easy reading of list data.
    eeout = eeout.replace( '\n', '' )
    eeout = eeout.split( ' ' )
    eeout = list( filter( None, eeout ) )

    # Remove head of file to make matching easier.
    eeout = eeout[ find_sublist( [ 'NODES', 'X', '(cm)' ], eeout )[ 0 ]: ]

    # Find the list positions for the first and last nodes.
    x_coords    = extract_sublist_entries( ['NODES', 'X', '(cm)'], nodes, eeout )
    y_coords    = extract_sublist_entries( ['NODES', 'Y', '(cm)'], nodes, eeout )
    z_coords    = extract_sublist_entries( ['NODES', 'Z', '(cm)'], nodes, eeout )
    e_types     = extract_sublist_entries( ['ELEMENT', 'TYPE'], total_elements, eeout )
    e_materials = extract_sublist_entries( ['ELEMENT', 'MATERIAL'], total_elements, eeout )

    # Process connectivity list.
    connectivity_list_elements = calculate_connectivity_list_length( e_types )
    connectivities = []
    if( '4' in e_types ):
        connectivities += extract_sublist_entries(
            [ 'CONNECTIVITY', 'DATA', '1ST', 'ORDER', 'TETS', 'ELEMENT', 'ORDERED' ],
            4 * e_types.count( '4' ), eeout )
    if( '5' in e_types ):
        connectivities += extract_sublist_entries(
            [ 'CONNECTIVITY', 'DATA', '1ST', 'ORDER', 'PENTS', 'ELEMENT', 'ORDERED' ],
            6 * e_types.count( '5' ), eeout )
    if( '6' in e_types ):
        connectivities += extract_sublist_entries(
            [ 'CONNECTIVITY', 'DATA', '1ST', 'ORDER', 'HEXS', 'ELEMENT', 'ORDERED' ],
            8 * e_types.count( '6' ), eeout )
    if( '14' in e_types ):
        connectivities += extract_sublist_entries(
            [ 'CONNECTIVITY', 'DATA', '2ND', 'ORDER', 'TETS', 'ELEMENT', 'ORDERED' ],
            10 * e_types.count( '14' ), eeout )
    if( '15' in e_types ):
        connectivities += extract_sublist_entries(
            [ 'CONNECTIVITY', 'DATA', '2ND', 'ORDER', 'PENTS', 'ELEMENT', 'ORDERED' ],
            15 * e_types.count( '15' ), eeout )
    if( '16' in e_types ):
        connectivities += extract_sublist_entries(
            [ 'CONNECTIVITY', 'DATA', '2ND', 'ORDER', 'HEXS', 'ELEMENT', 'ORDERED' ],
```

```
            20 * e_types.count( '16' ), eeout )

    densities = extract_sublist_entries(
        [ 'DENSITY', '(gm/cm^3)' ],
        total_elements, eeout )

    volumes = extract_sublist_entries(
        [ 'VOLUMES', '(cm^3)' ],
        total_elements, eeout )

    # Create list of vertices from individual coordinate lists.
    vertices = create_vertices( x_coords, y_coords, z_coords )

    # Subtract one from all vertex IDs in the connectivity list (to make
    # zero-indexed).
    connectivities = [ str( int( x ) - 1 ) for x in connectivities ]

    # Accumulate offset list. Reproduce np.cumsum to avoid NumPy dependency.
    def cumsum( inlist ):
        cumlist = [ 0 ]
        for n,i in enumerate( inlist ):
            cumlist.append( cumlist[n] + inlist[n] )
        return cumlist[1:]
    offsets = cumsum( connectivity_list_elements )
    offsets = [ str( x ) for x in offsets ]

    # Convert eeout element types to VTK element types.
    vtk_e_types = calculate_vtk_e_types( e_types )

    # Open up output vtu (unstructured mesh VTK) file.
    f = open( infilename + '.vtu', 'w' )

    # Write header comments (but a long header does not work), default: off.
    if( False ):
        [ f.write( '<!-- ' + l + ' -->\n' ) for l in eeout_header.split( '\n' ) ]
        f.write( '<!--' + '\n' )
        f.write( 80 * '#' + '\n' )
        f.write( '# EEOUT Header Follows' + '\n' )
        f.write( 80 * '#' + '\n' )
        f.write( eeout_header + '\n' )
        f.write( 80 * '#' + '\n' )
        f.write( '-->' + '\n' )

    f.write( '<VTKFile type="UnstructuredGrid" version="0.1" byte_order="LittleEndian">' + '\n' )
    f.write( '  <UnstructuredGrid>' + '\n' )
    f.write( '    <Piece NumberOfPoints="' + str( nodes ) + '" NumberOfCells="' + str( total_elements ) + '">' + '\n' )
```

```
    f.write( '      <CellData Scalars="scalars">' + '\n' )
    f.write( '        <DataArray type="Int32" Name="material" format="ascii">' + '\n' )
    f.write( pretty_print_list( 10, 10, 5, e_materials ) )
    f.write( '        </DataArray>' + '\n' )
    f.write( '        <DataArray type="Float64" Name="density" format="ascii">' + '\n' )
    f.write( pretty_print_list( 10, 5, 13, densities ) )
    f.write( '        </DataArray>' + '\n' )
    f.write( '        <DataArray type="Float64" Name="volume" format="ascii">' + '\n' )
    f.write( pretty_print_list( 10, 5, 13, volumes ) )
    f.write( '        </DataArray>' + '\n' )

    # Output edit information.  Edits may have corresponding relative
    # uncertainties.  Edits may be binned by energy and/or time.
    if( len( edit_list ) > 0 ):
        for e in edit_list:
            particle_type = re.search( r'PARTICLE : (\d+)', e ).group( 1 )
            edit_type = re.search( r'TYPE : (.*?)$', e ).group( 1 )
            edit_number = re.search( r'TYPE : .*?_(\d+)$', e ).group( 1 )
            print( '  Processing {:} edit...'.format( edit_type ) )
            myregex = '({:}.*?)(:?DATA OUTPUT PARTICLE|$)'.format( e )
            edit_data = re.search( myregex, eeout_edits, re.S ).group( 1 )
            edit_results = get_results( edit_data, edit_number, total_elements )
            for er in edit_results:
                f.write( '        <DataArray type="Float64" Name="' + er[0]+ '" format="ascii">' + '\n' )
                f.write( pretty_print_list( 10, 5, 13, er[1] ) )
                f.write( '        </DataArray>' + '\n' )

    f.write( '      </CellData>' + '\n' )
    f.write( '      <Points>' + '\n' )
    f.write( '        <DataArray type="Float64" NumberOfComponents="3" format="ascii">' + '\n' )
    f.write( pretty_print_list( 10, 3, 13, vertices ) )
    f.write( '        </DataArray>' + '\n' )
    f.write( '      </Points>' + '\n' )
    f.write( '      <Cells>' + '\n' )
    f.write( '        <DataArray type="Int32" Name="connectivity" format="ascii">' + '\n' )
    f.write( pretty_print_list( 10, 8, 5, connectivities ) )
    f.write( '        </DataArray>' + '\n' )
    f.write( '        <DataArray type="Int32" Name="offsets" format="ascii">' + '\n' )
    f.write( pretty_print_list( 10, 10, 5, offsets ) )
    f.write( '        </DataArray>' + '\n' )
    f.write( '        <DataArray type="UInt8" Name="types" format="ascii">' + '\n' )
    f.write( pretty_print_list( 10, 20, 2, vtk_e_types ) )
    f.write( '        </DataArray>' + '\n' )
    f.write( '      </Cells>' + '\n' )
    f.write( '    </Piece>' + '\n' )
    f.write( '  </UnstructuredGrid>' + '\n' )
```

```
    f.write( '</VTKFile>' + '\n' )

    f.close()
```

B Test Case Files

This appendix provides various files associated with the test cases. They are provided as examples and to aide in reproducing some of the verification work herein.

B.1 Test Case Example Files (Only Linear Hexahedral Elements)

An example MCNP input using only linear hexahedral elements is given in Listing 2 and the corresponding Abaqus mesh input file is given in Listing 3.

Listing 2: Test Case MCNP Input File (Only Linear Hexahedral Elements)

```
No description for this unstructured mesh file
c
c  Created from file    : test_case_01_c_c3d8__r_c3d8__s_c3d8__abaq.inp
c  Using data cards file: std_datacards.mcnp.inp
c  Created on           :  1- 9-2019 @  6:42:36
c
c
c PSEUDO CELLS
01       1     -15.1000      0  u=1
02       2     -2.69890      0  u=1
03       3    -0.998207      0  u=1
04       0                   0  u=1
c
c LEGACY CELLS
05       0                 -99  fill=1
06       0                  99

c
c SURFACES
99  sph   2.50000E+00   0.00000E+00   1.25000E+00   8.07775E+00

c
c  DATA CARDS
embed1 meshgeo=abaqus
       mgeoin=test_case_01_c_c3d8__r_c3d8__s_c3d8__abaq.inp
       meeout=test_case_01_c_c3d8__r_c3d8__s_c3d8__abaq.eeout
       length= 1.00000E-01
       background=       4
       matcell=  1  1  2  2  3  3
c
embee4:n embed=1
c
sdef  pos= volumer
c
      erg=d1
sp1 -3 1.18 1.03419
nonu
c
mode n
c
m1       98252     1.0                                                    $ californium-252
c                                                                         $ density: 15.1 g/cc
c                                                                         $ composition assumed; density from crc handbook
c
m2       13027     1.0                                                    $ aluminum
```

```
c                                                                    $ density: 2.6989 g/cc
c                                                                    $ composition & density from pnnl-15870, rev. 1
c
m3        1001     0.666657                                          $ water, liquid @ 23.15 deg-c
          8016     0.333343                                          $ density: 0.998207 g/cc
mt3 lwtr.10                                                          $ composition & density from pnnl-15870, rev. 1
c
imp:n 1 1 1 1 1 0
c
embee14:n embed=1 errors=yes
embee6:n embed=1
embee16:n embed=1 errors=yes
c
rand gen=2 seed=12345
print
nps 1e6
```

Listing 3: Test Case Abaqus Mesh Input File (Only Linear Hexahedral Elements)

```
*Heading
** Job name: test_case_01_c_c3d8__r_c3d8__s_c3d8__abaq Model name: Model-1
** Generated by: Abaqus/CAE 2018
*Preprint, echo=NO, model=NO, history=NO, contact=NO
**
** PARTS
**
*Part, name=Cube
*Node
      1,          25.,          25.,          25.
      2,           0.,          25.,          25.
      3,         -25.,          25.,          25.
      4,          25.,           0.,          25.
      5,           0.,           0.,          25.
      6,         -25.,           0.,          25.
      7,          25.,         -25.,          25.
      8,           0.,         -25.,          25.
      9,         -25.,         -25.,          25.
     10,          25.,          25.,           0.
     11,           0.,          25.,           0.
     12,         -25.,          25.,           0.
     13,          25.,           0.,           0.
     14,           0.,           0.,           0.
     15,         -25.,           0.,           0.
     16,          25.,         -25.,           0.
     17,           0.,         -25.,           0.
     18,         -25.,         -25.,           0.
```

```
     19,          25.,          25.,         -25.
     20,           0.,          25.,         -25.
     21,         -25.,          25.,         -25.
     22,          25.,           0.,         -25.
     23,           0.,           0.,         -25.
     24,         -25.,           0.,         -25.
     25,          25.,         -25.,         -25.
     26,           0.,         -25.,         -25.
     27,         -25.,         -25.,         -25.
*Element, type=C3D8
1,  4,  5,  2,  1, 13, 14, 11, 10
2,  5,  6,  3,  2, 14, 15, 12, 11
3,  7,  8,  5,  4, 16, 17, 14, 13
4,  8,  9,  6,  5, 17, 18, 15, 14
5, 13, 14, 11, 10, 22, 23, 20, 19
6, 14, 15, 12, 11, 23, 24, 21, 20
7, 16, 17, 14, 13, 25, 26, 23, 22
8, 17, 18, 15, 14, 26, 27, 24, 23
*Nset, nset=Set_material_tally_source_1, generate
  1,  27,   1
*Elset, elset=Set_material_tally_source_1, generate
 1,  8,  1
*End Part
**
*Part, name=Right_Triangle
*Node
      1,          75.,         -25.,         -25.
      2,          25.,         -25.,          25.
      3,          25.,         -25.,         -25.
      4,          50.,         -25.,           0.
      5,          25.,         -25.,           0.
      6,          50.,         -25.,         -25.
      7,   42.6746407,         -25.,   -7.3253603
      8,          75.,           0.,         -25.
      9,          25.,           0.,          25.
     10,          25.,           0.,         -25.
     11,          50.,           0.,           0.
     12,          25.,           0.,           0.
     13,          50.,           0.,         -25.
     14,   42.6746407,           0.,   -7.3253603
     15,          75.,          25.,         -25.
     16,          25.,          25.,          25.
     17,          25.,          25.,         -25.
     18,          50.,          25.,           0.
     19,          25.,          25.,           0.
     20,          50.,          25.,         -25.
```

```
     21,   42.6746407,          25.,   -7.3253603
*Element, type=C3D8
1,  5,  7,  6,  3, 12, 14, 13, 10
2,  7,  5,  2,  4, 14, 12,  9, 11
3,  7,  4,  1,  6, 14, 11,  8, 13
4, 12, 14, 13, 10, 19, 21, 20, 17
5, 14, 12,  9, 11, 21, 19, 16, 18
6, 14, 11,  8, 13, 21, 18, 15, 20
*Nset, nset=Set_material_tally_2, generate
  1,  21,   1
*Elset, elset=Set_material_tally_2, generate
 1,  6,  1
*End Part
**
*Part, name=Semicircular_Cap
*Node
      1,          25.,         -25.,          25.
      2,         -25.,         -25.,          25.
      3,   17.6776695,         -25.,   42.6776695
      4,           0.,         -25.,          50.
      5,  -17.6776695,         -25.,   42.6776695
      6,           0.,         -25.,          25.
      7,          25.,           0.,          25.
      8,         -25.,           0.,          25.
      9,   17.6776695,           0.,   42.6776695
     10,           0.,           0.,          50.
     11,  -17.6776695,           0.,   42.6776695
     12,           0.,           0.,          25.
     13,          25.,          25.,          25.
     14,         -25.,          25.,          25.
     15,   17.6776695,          25.,   42.6776695
     16,           0.,          25.,          50.
     17,  -17.6776695,          25.,   42.6776695
     18,           0.,          25.,          25.
*Element, type=C3D8
1,  4,  3,  1,  6, 10,  9,  7, 12
2,  4,  6,  2,  5, 10, 12,  8, 11
3, 10,  9,  7, 12, 16, 15, 13, 18
4, 10, 12,  8, 11, 16, 18, 14, 17
*Nset, nset=Set_material_tally_3, generate
  1,  18,   1
*Elset, elset=Set_material_tally_3, generate
 1,  4,  1
*End Part
**
**
```

```
** ASSEMBLY
**
*Assembly, name=Assembly
**
*Instance, name=Cube-1, part=Cube
*End Instance
**
*Instance, name=Right_Triangle-1, part=Right_Triangle
*End Instance
**
*Instance, name=Semicircular_Cap-1, part=Semicircular_Cap
*End Instance
**
*End Assembly
**
** MATERIALS
**
*Material, name=Material_Al_2
*Density
 -2.6989,
*Material, name=Material_Cf-252_1
*Density
 -15.1,
*Material, name=Material_H2O_3
*Density
 -0.998207,
```

B.2 Test Case Example Files (Mixed Linear and Quadratic Elements)

An example MCNP input using a variety of linear and quadratic hexahedral and pentahedral elements is given in Listing 4 and the corresponding Abaqus mesh input file is given in Listing 5.

Listing 4: Test Case MCNP Input File (Mixed Linear and Quadratic Elements)

```
No description for this unstructured mesh file
c
c  Created from file    : test_case_01_c_c3d8__r_c3d6__s_c3d15_abaq.inp
c  Using data cards file: std_datacards.mcnp.inp
c  Created on           :  1- 9-2019 @  6:42:36
c
c
c PSEUDO CELLS
01       1     -15.1000      0  u=1
02       2     -2.69890      0  u=1
03       3    -0.998207      0  u=1
04       0                   0  u=1
c
c LEGACY CELLS
05       0                 -99  fill=1
06       0                  99

c
c SURFACES
99  sph   2.50000E+00   0.00000E+00   1.25000E+00   8.07775E+00

c
c  DATA CARDS
embed1 meshgeo=abaqus
       mgeoin=test_case_01_c_c3d8__r_c3d6__s_c3d15_abaq.inp
       meeout=test_case_01_c_c3d8__r_c3d6__s_c3d15_abaq.eeout
       length= 1.00000E-01
       background=       4
       matcell=  1  1  2  2  3  3
c
embee4:n embed=1
c
sdef  pos= volumer
c
      erg=d1
sp1 -3 1.18 1.03419
nonu
c
mode n
c
m1       98252     1.0                                                    $ californium-252
c                                                                         $ density: 15.1 g/cc
c                                                                         $ composition assumed; density from crc handbook
c
m2       13027     1.0                                                    $ aluminum
```

```
c                                                                    $ density: 2.6989 g/cc
c                                                                    $ composition & density from pnnl-15870, rev. 1
c
m3        1001     0.666657                                          $ water, liquid @ 23.15 deg-c
          8016     0.333343                                          $ density: 0.998207 g/cc
mt3 lwtr.10                                                          $ composition & density from pnnl-15870, rev. 1
c
imp:n 1 1 1 1 1 0
c
embee14:n embed=1 errors=yes
embee6:n embed=1
embee16:n embed=1 errors=yes
c
rand gen=2 seed=12345
print
nps 1e6
```

Listing 5: Test Case Abaqus Mesh Input File (Mixed Linear and Quadratic Elements)

```
*Heading
** Job name: test_case_01_c_c3d8__r_c3d6__s_c3d15_abaq Model name: Model-1
** Generated by: Abaqus/CAE 2018
*Preprint, echo=NO, model=NO, history=NO, contact=NO
**
** PARTS
**
*Part, name=Cube
*Node
      1,          25.,          25.,          25.
      2,           0.,          25.,          25.
      3,         -25.,          25.,          25.
      4,          25.,           0.,          25.
      5,           0.,           0.,          25.
      6,         -25.,           0.,          25.
      7,          25.,         -25.,          25.
      8,           0.,         -25.,          25.
      9,         -25.,         -25.,          25.
     10,          25.,          25.,           0.
     11,           0.,          25.,           0.
     12,         -25.,          25.,           0.
     13,          25.,           0.,           0.
     14,           0.,           0.,           0.
     15,         -25.,           0.,           0.
     16,          25.,         -25.,           0.
     17,           0.,         -25.,           0.
     18,         -25.,         -25.,           0.
```

```
     19,          25.,          25.,         -25.
     20,           0.,          25.,         -25.
     21,         -25.,          25.,         -25.
     22,          25.,           0.,         -25.
     23,           0.,           0.,         -25.
     24,         -25.,           0.,         -25.
     25,          25.,         -25.,         -25.
     26,           0.,         -25.,         -25.
     27,         -25.,         -25.,         -25.
*Element, type=C3D8
1,  4,  5,  2,  1, 13, 14, 11, 10
2,  5,  6,  3,  2, 14, 15, 12, 11
3,  7,  8,  5,  4, 16, 17, 14, 13
4,  8,  9,  6,  5, 17, 18, 15, 14
5, 13, 14, 11, 10, 22, 23, 20, 19
6, 14, 15, 12, 11, 23, 24, 21, 20
7, 16, 17, 14, 13, 25, 26, 23, 22
8, 17, 18, 15, 14, 26, 27, 24, 23
*Nset, nset=Set_material_tally_source_1, generate
  1,  27,   1
*Elset, elset=Set_material_tally_source_1, generate
 1,  8,  1
*End Part
**
*Part, name=Right_Triangle
*Node
      1,          75.,         -25.,         -25.
      2,          25.,         -25.,          25.
      3,          25.,         -25.,         -25.
      4,          50.,         -25.,           0.
      5,          25.,         -25.,           0.
      6,          50.,         -25.,         -25.
      7,          75.,           0.,         -25.
      8,          25.,           0.,          25.
      9,          25.,           0.,         -25.
     10,          50.,           0.,           0.
     11,          25.,           0.,           0.
     12,          50.,           0.,         -25.
     13,          75.,          25.,         -25.
     14,          25.,          25.,          25.
     15,          25.,          25.,         -25.
     16,          50.,          25.,           0.
     17,          25.,          25.,           0.
     18,          50.,          25.,         -25.
*Element, type=C3D6
1,  3,  4,  6,  9, 10, 12
```

```
2,  5,  2,  4, 11,  8, 10
3,  4,  3,  5, 10,  9, 11
4,  4,  1,  6, 10,  7, 12
5,  9, 10, 12, 15, 16, 18
6, 11,  8, 10, 17, 14, 16
7, 10,  9, 11, 16, 15, 17
8, 10,  7, 12, 16, 13, 18
*Nset, nset=Set_material_tally_2, generate
  1,  18,   1
*Elset, elset=Set_material_tally_2, generate
 1,  8,  1
*End Part
**
*Part, name=Semicircular_Cap
*Node
      1,          25.,         -25.,          25.
      2,         -25.,         -25.,          25.
      3,   17.6776695,         -25.,   42.6776695
      4,           0.,         -25.,          50.
      5,  -17.6776695,         -25.,   42.6776695
      6,           0.,         -25.,          25.
      7,          25.,           0.,          25.
      8,         -25.,           0.,          25.
      9,   17.6776695,           0.,   42.6776695
     10,           0.,           0.,          50.
     11,  -17.6776695,           0.,   42.6776695
     12,           0.,           0.,          25.
     13,          25.,          25.,          25.
     14,         -25.,          25.,          25.
     15,   17.6776695,          25.,   42.6776695
     16,           0.,          25.,          50.
     17,  -17.6776695,          25.,   42.6776695
     18,           0.,          25.,          25.
     19,           0.,         -25.,         37.5
     20,   8.83883476,         -25.,   33.8388367
     21,   9.56708622,         -25.,   48.0969887
     22,   9.56708527,           0.,   48.0969887
     23,   8.83883476,           0.,   33.8388367
     24,           0.,           0.,         37.5
     25,   17.6776695,        -12.5,   42.6776695
     26,           0.,        -12.5,          50.
     27,           0.,        -12.5,          25.
     28,  -8.83883476,         -25.,   33.8388367
     29,  -9.56708622,         -25.,   48.0969887
     30,  -9.56708527,           0.,   48.0969887
     31,  -8.83883476,           0.,   33.8388367
```

```
      32,  -17.6776695,         -12.5,   42.6776695
      33,        -12.5,          -25.,          25.
      34,  -23.0969887,          -25.,   34.5670853
      35,  -23.0969868,            0.,   34.5670891
      36,        -12.5,            0.,          25.
      37,         -25.,         -12.5,          25.
      38,         12.5,          -25.,          25.
      39,   23.0969887,          -25.,   34.5670853
      40,   23.0969868,            0.,   34.5670891
      41,         12.5,            0.,          25.
      42,          25.,         -12.5,          25.
      43,   9.56708622,          25.,   48.0969887
      44,   8.83883476,          25.,   33.8388367
      45,           0.,          25.,         37.5
      46,   17.6776695,         12.5,   42.6776695
      47,           0.,         12.5,          50.
      48,           0.,         12.5,          25.
      49,  -9.56708622,          25.,   48.0969887
      50,  -8.83883476,          25.,   33.8388367
      51,  -17.6776695,         12.5,   42.6776695
      52,  -23.0969887,          25.,   34.5670853
      53,        -12.5,          25.,          25.
      54,         -25.,         12.5,          25.
      55,   23.0969887,          25.,   34.5670853
      56,         12.5,          25.,          25.
      57,          25.,         12.5,          25.
*Element, type=C3D15
1,  4,  3,  6, 10,  9, 12, 21, 20, 19, 22, 23, 24, 26, 25, 27
2,  5,  4,  6, 11, 10, 12, 29, 19, 28, 30, 24, 31, 32, 26, 27
3,  2,  5,  6,  8, 11, 12, 34, 28, 33, 35, 31, 36, 37, 32, 27
4,  3,  1,  6,  9,  7, 12, 39, 38, 20, 40, 41, 23, 25, 42, 27
5, 10,  9, 12, 16, 15, 18, 22, 23, 24, 43, 44, 45, 47, 46, 48
6, 11, 10, 12, 17, 16, 18, 30, 24, 31, 49, 45, 50, 51, 47, 48
7,  8, 11, 12, 14, 17, 18, 35, 31, 36, 52, 50, 53, 54, 51, 48
8,  9,  7, 12, 15, 13, 18, 40, 41, 23, 55, 56, 44, 46, 57, 48
*Nset, nset=Set_material_tally_3, generate
  1,  57,   1
*Elset, elset=Set_material_tally_3, generate
 1,  8,  1
*End Part
**
**
** ASSEMBLY
**
*Assembly, name=Assembly
**
```

```
*Instance, name=Cube-1, part=Cube
*End Instance
**
*Instance, name=Right_Triangle-1, part=Right_Triangle
*End Instance
**
*Instance, name=Semicircular_Cap-1, part=Semicircular_Cap
*End Instance
**
*End Assembly
**
** MATERIALS
**
*Material, name=Material_Al_2
*Density
 -2.6989,
*Material, name=Material_Cf-252_1
*Density
 -15.1,
*Material, name=Material_H2O_3
*Density
 -0.998207,
```

B.3 Test Case Example Files (Energy/Time Binning)

An example MCNP input using time only, energy only and time and energy bins are given in Listing 6 and the corresponding Abaqus mesh input file is given in Listing 7.

Listing 6: Test Case MCNP Input File (Energy/Time Binning)

```
Simple Al Cube
c ----------------------------- Cell Cards ---------------------------------- 80
1     1     -2.7           0                                u=1
2    0      0                                u=1 $ background
18    0      100 -101 102 -103 104 -105   fill=1 $ fill cell
19    0                  (-100:101:-102:103:-104:105)
c -------------------------- End Cell Cards ------------------------------- 80

c ---------------------------- Surface Cards ------------------------------- 80
c
100 px -130
101 px 130
102 py -130
103 py 130
104 pz -130
105 pz 130
c ------------------------- End Surface Cards ----------------------------- 80

c ----------------------------- Data Cards ---------------------------------- 80
c Embedded Geometry Specification
embed1 meshgeo=abaqus mgeoin=Simple_Example.abaq
       meeout=Simple_Example.eeout
       filetype=ascii
       background=2
       matcell= 1 1
c
c Materials
m1 13027 -1.0
mode n
c
c Cell Importances
imp:n 1 1 1 0
c
c Source Definition
sdef pos 12.071 12.071 12.071  erg=d2 wgt=1E12 tme= d1
si1 1.0 1E2
sp1 0 1
si2 1E-3 1E-2 1E-1 1E0
sp2 0 1 2 3
c
nps 1E7
prdmp j 1E8 1 2 1E8
print
c Histories (or Computer Time Cutoff)
embee4:n embed=1
```

```
embtb4 1.0 1E2
embeb4 1E-3 1E-2 1E-1 1E0
c
c --------------------------- End Data Cards ------------------------------- 80
c End MCNP Input
```

Listing 7: Test Case Abaqus Input File (Energy/Time Binning)

```
*Heading
** Job name: Job-1 Model name: Model-1
** Generated by: Abaqus/CAE 2018
*Preprint, echo=NO, model=NO, history=NO, contact=NO
**
** PARTS
**
*Part, name=Simple_Example
*Node
      1,  0.500843108,  0.500843108,          0.5
      2,  0.500843108,           0.,          0.5
      3,  0.500843108, -0.500843108,          0.5
      4,  0.500843108,  0.500843108,           0.
      5,  0.500843108,           0.,           0.
      6,  0.500843108, -0.500843108,           0.
      7,  0.500843108,  0.500843108,         -0.5
      8,  0.500843108,           0.,         -0.5
      9,  0.500843108, -0.500843108,         -0.5
     10,           0.,  0.500843108,          0.5
     11,           0.,           0.,          0.5
     12,           0., -0.500843108,          0.5
     13,           0.,  0.500843108,           0.
     14,           0.,           0.,           0.
     15,           0., -0.500843108,           0.
     16,           0.,  0.500843108,         -0.5
     17,           0.,           0.,         -0.5
     18,           0., -0.500843108,         -0.5
     19, -0.500843108,  0.500843108,          0.5
     20, -0.500843108,           0.,          0.5
     21, -0.500843108, -0.500843108,          0.5
     22, -0.500843108,  0.500843108,           0.
     23, -0.500843108,           0.,           0.
     24, -0.500843108, -0.500843108,           0.
     25, -0.500843108,  0.500843108,         -0.5
     26, -0.500843108,           0.,         -0.5
     27, -0.500843108, -0.500843108,         -0.5
*Element, type=C3D8R
1, 10, 11, 14, 13,  1,  2,  5,  4
```

```
2, 11, 12, 15, 14,  2,  3,  6,  5
3, 13, 14, 17, 16,  4,  5,  8,  7
4, 14, 15, 18, 17,  5,  6,  9,  8
5, 19, 20, 23, 22, 10, 11, 14, 13
6, 20, 21, 24, 23, 11, 12, 15, 14
7, 22, 23, 26, 25, 13, 14, 17, 16
8, 23, 24, 27, 26, 14, 15, 18, 17
*Nset, nset=Simple_Box_material_tally_001, generate
  1,  27,   1
*Elset, elset=Simple_Box_material_tally_001, generate
 1,  8,  1
** Section: Section-1
*Solid Section, elset=Simple_Box_material_tally_001, material=Material-1
*End Part
**
**
** ASSEMBLY
**
*Assembly, name=Assembly
**
*Instance, name=Simple_Example-1, part=Simple_Example
*End Instance
,
**
*End Assembly
**
** MATERIALS
**
*Material, name=Material-1
*Density
-1.,
```

C ParaView Macros

This appendix provides several ParaView macros that the first author has found useful when visualizing MCNP output. All macros have been tested with ParaView version 5.6.0.

Listing 8 provides a macro to adjust the lighting from strictly diffuse (the default) to strictly ambient. This eliminates the ability to observe shadows; however, the resulting color palette is directly correlated to the color bar as demonstrated in Fig. 7.

Listing 9 simply reloads all files in the pipeline.

Listing 10 applies several filters to illustrate material boundaries. This can be particularly useful when drawing multiple datasets with geometry and results overlaid. An example of this macro applied to UM geometry is shown in Fig. 8, which shows neutron flux 3-D contours and associated statistical uncertainties at a particular elevation with the geometry boundaries to orient the viewer [7].

Listing 8: ParaView Macro to Make Lighting Strictly Ambient

```
from paraview.simple import *

# Originally from: https://www.cfd-online.com/Forums/openfoam-paraview/92638-dark-areas-paraview.html
# Updated for Paraview 5.6.0.
asrc = GetActiveSource()
rv = GetActiveViewOrCreate('RenderView')
asrcd = GetDisplayProperties(asrc, view=rv)
asrcd.Ambient = 1.0
asrcd.Diffuse = 0.0
```

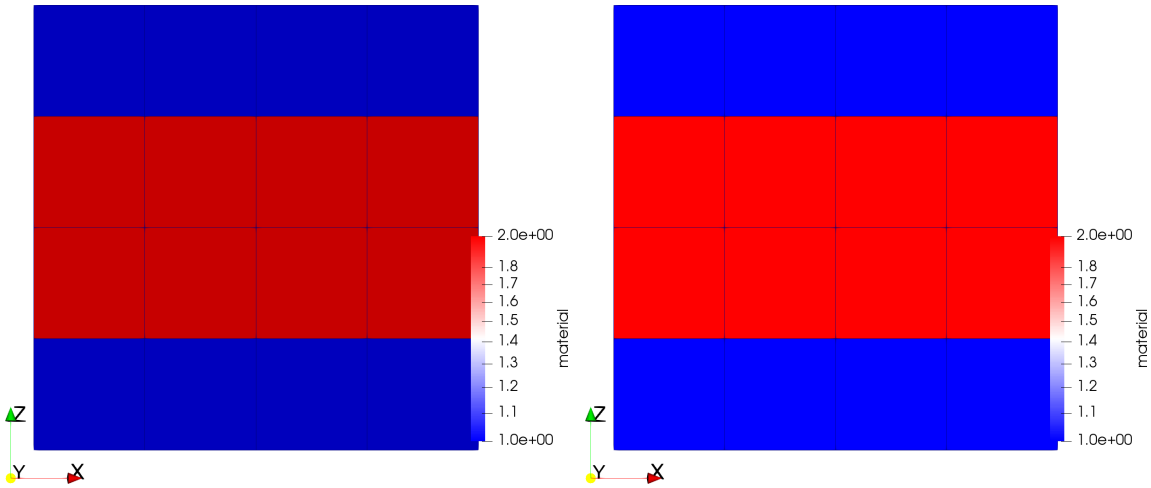
Listing 9: ParaView Macro to Reload all Data Files

```
from paraview.simple import *
for k,v in GetSources().items():
  ReloadFiles(v)
```

Listing 10: ParaView Macro to Draw Material Boundary Lines

```
from paraview.simple import *

t = GetActiveSource()
e = ExtractSurface( Input = t )
f = FeatureEdges( Input = e )
r = GetActiveView()
HideAll(r)
Show(f, r)
r.Update()
```



(a) Strictly Diffuse (b) Strictly Ambient

Figure 7: Diffuse vs. Ambient Lighting Effect on Data Field Visualization vs. Color Bar Representation

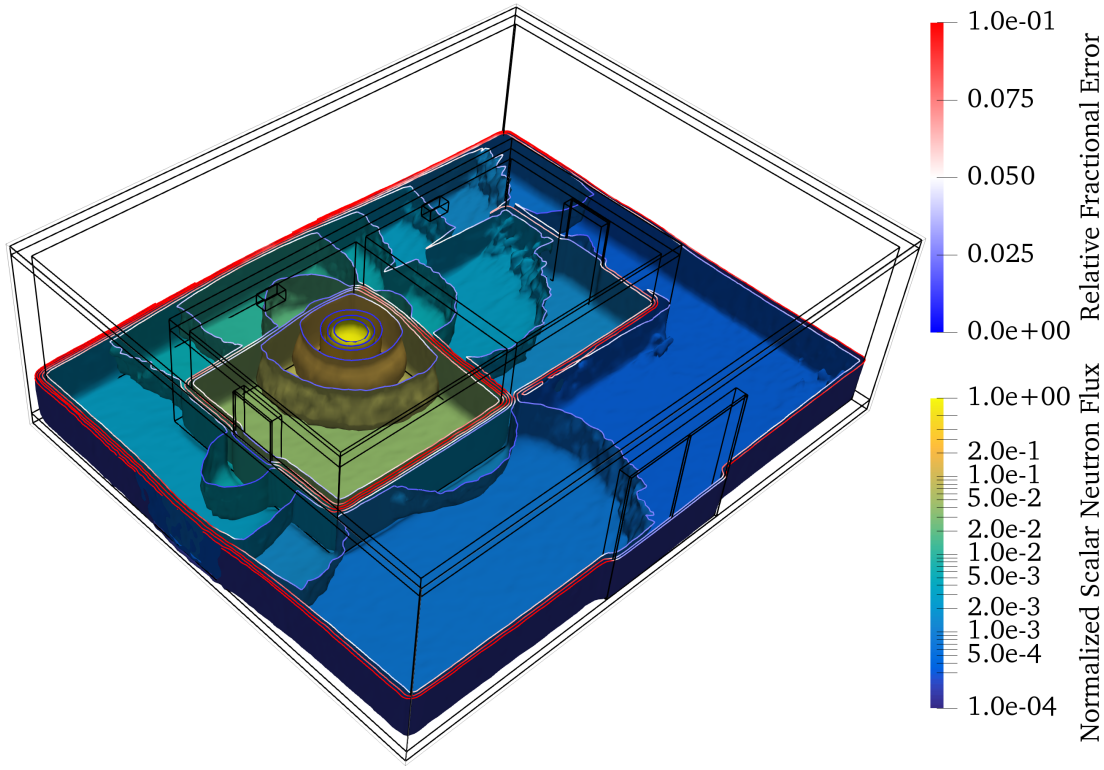


Figure 8: Concurrent Visualization of Geometry, Results, and Associated Statistical Uncertainties