LA-UR-24-29475

Approved for public release; distribution is unlimited.

Title:	A Python Script to Read MCNP6.3 Surface-Source Files
Author(s):	Kulesza, Joel A.
Intended for:	Report
Issued:	2024-09-03

A Python Script to Read MCNP6.3 Surface-Source Files

Joel A. Kulesza

Monte Carlo Codes Group (XCP-3), Los Alamos National Laboratory

1 Introduction

This report provides a Python script to read an MCNP® [1] surface source file created with the `SSW` card with `SYM = 0` (i.e., the default symmetry treatment). For background: the general format of an MCNP surface-source file is described in [2]; however, that document did not provide coding and/or a tool to interrogate such files. The current format will not be given in this document other than through the record-read statements necessary for the script to function.

The reader capability in this report is augmented with the ability to directly write a couple demonstrative outputs:

1. A comma-separated value (CSV) file containing particle phase-space state information and
2. A Matplotlib histogram of the energy distribution of the particles.

This report also describes accompanying verification work that shows the script performing as required with MCNP6.2, MCNP6.3, and (expected) MCNP6.4 surface-source files. However, users of the enclosed script must still verify that the script is behaving correctly for their own work.

2 Script Usage

The enclosed script, named **`Convert_MCNP_ssa.py`** and given in Listing 5 in Appendix A and electronically attached to this PDF, is executed from the command line followed by the filename of the surface-source file as, for example:

```
python3 Convert_MCNP_ssa.py wssa
```

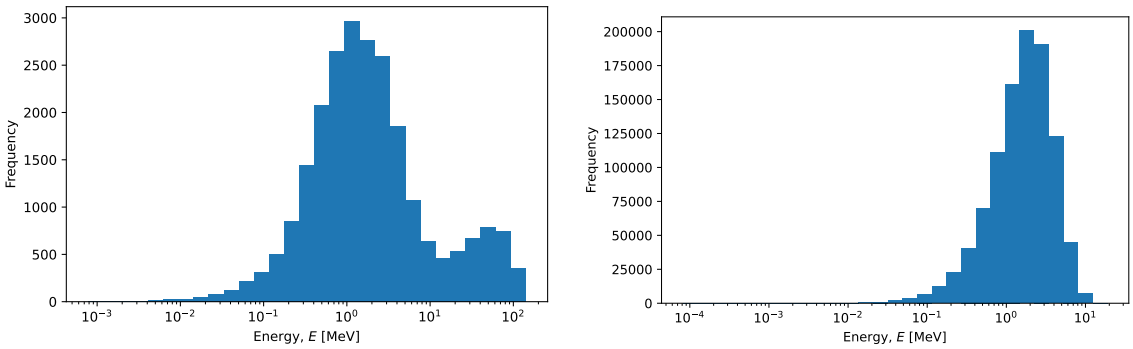
Output to the terminal is of the form:

```
Found 23829 source points in file.
Processing 23829 histories...
```

The reason the script gives both the number of source points found and the number processed is that it processes the lesser of the total number of source points found and the user-defined value with the **`-n`** flag, where `python3 Convert_MCNP_ssa.py wssa -n 10000` would yield

```
Found 23829 source points in file.
Processing 10000 histories...
```

In addition, a **`-csv`** option writes a new CSV file containing particle phase-space state information such that `python3 Convert_MCNP_ssa.py wssa -n 10000 -csv` produces a file named **`wssa.csv`** starting in the manner shown in Listing 2. Finally, a **`-erg`** option writes a new PDF file containing an



(a) Test case 1 energy distribution.

(b) Test case 2 energy distribution.

Figure 1: Example energy distributions.

energy histogram created with Matplotlib such that `python3 Convert_MCNP_ssa.py wssa -n 10000 -erg` produces something like either image in Fig. 1.

3 Functional and Performance Requirements

The functional requirements for this script are:

1. The script shall be able to read MCNP6.3 binary surface-source files for "traditional" surface sources. A "traditional" surface source is composed of particle phase-space states that are recorded as the particles pass through an identified surface.

2. The script shall be able to read MCNP6.3 binary surface-source files for fission-site "surface" sources. A fission-site "surface" source is composed of fission source-site phase-space information from a k-eigenvalue (i.e., `KCODE`) calculation.

3. The script shall demonstrate how to use the information read from the surface-source file.

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

Note that no requirement exists to write surface-source files. As noted previously, no requirement exists to handle symmetric surface sources (i.e., `SYM` $\neq$ `0` on the `SSW` card).

4 Design and Implementation

There is a current hope that MCNP surface-source files will be converted to an HDF5 format, which would render this work obsolete. As a result, speed of development and flexibility were informal design and implementation requirements. Accordingly, the Python language was selected. The script does not significantly leverage object-oriented functionality; however, a **`Particle`** class to collect particle phase-space state information is included that can be extended, as needed. 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 3.9.13 provided by Anaconda[1] on macOS version 14.5). The SciPy package is required to enable reading the

[1] https://www.anaconda.com

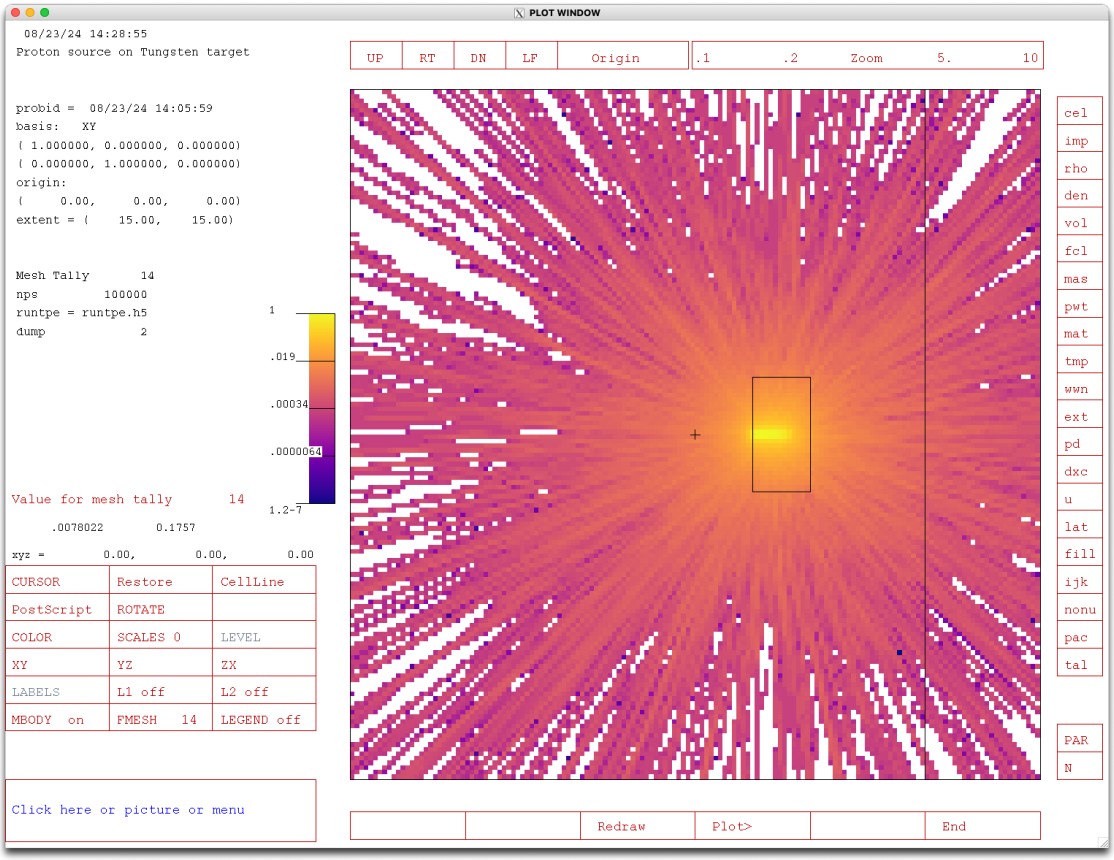


Figure 2: `FMESH`14 neutron distribution.

surface-source file. Further, the NumPy and Matplotlib packages are needed to create an optional energy-histogram plot.

5 Testing

Two test cases are used to demonstrate that functional requirements 1, 2, and 3 are met. Both test cases for this work are taken from [3]. The input files necessary to reproduce these tests are given in Appendix B.

5.1 Test Case 1: "Traditional" Surface Source

A "traditional" surface source test case is performed using the surface-source file created from [Listing 4.7 of 3]. This test case uses a proton beam incident on a tungsten target to create spallation neutrons. The neutrons are recorded crossing a surface to create a neutron surface source suitable for a subsequent set of calculations that avoid the need to routinely simulate the spallation process.

For MCNP6.2, 6.3, and 6.4, all calculations yield an `FMESH`14 result comparable to [page 83 of 3] as shown in Fig. 2 and write a surface-source file containing 23,829 tracks corresponding to 18,569 of the original 100,000 histories.

A quantitative verification of Requirement 1 is performed by executing [Listing 4.9 of 3] modified for spacing and to run 100,000 histories to directly correspond to the surface source written. The first 50 source-particle conditions given in `print table 110` are compared with the **`-csv`** output

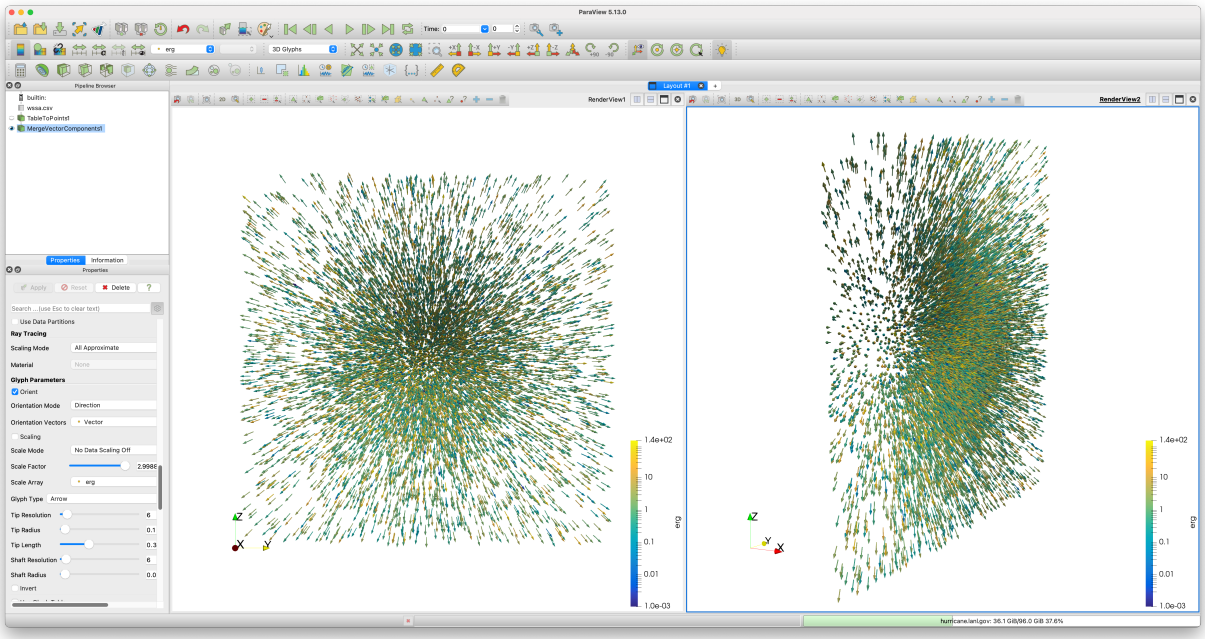


Figure 3: Test case 1 visual verification.

from the script herein (Listings 1 and 2). The values match (expect for the expected −5 cm shift in x that is intentionally demonstrated in [Listing 4.9 of 3]) suggesting that the surface-source file is read correctly. Further, a visual verification of the resulting surface source can be performed using the process given in Appendix C that results in Fig. 3, which matches behavior suggested by Fig. 2.

Listing 1: Test Case 1 MCNP6.3 Print Table 110 Excerpt

```
1starting mcrun.      cp0 =  0.02                                                                print table 110

      Neutron spallation source read


     nps     x          y          z          cell       surf     u          v          w          energy     weight      time

       1  5.000E+00  9.735E+00 -7.351E-01       20         11  5.026E-01  8.630E-01 -5.131E-02  9.256E+00  9.999E-01  3.039E-01
       2  5.000E+00  5.153E-01  3.169E+00       20         11  9.117E-01  6.857E-02  4.051E-01  1.150E+00  1.000E+00  5.477E-01
       3  5.000E+00  2.287E+00 -3.983E+00       20         11  8.989E-01  2.154E-01 -3.815E-01  8.105E-01  9.887E-01  6.796E-01
       4  5.000E+00 -1.564E+00 -2.483E+00       20         11  8.908E-01 -2.406E-01 -3.856E-01  3.582E+01  9.948E-01  1.134E-01
       5  5.000E+00  3.215E-01  5.822E+00       20         11  7.894E-01  3.385E-02  6.130E-01  2.567E+01  1.000E+00  1.548E-01
```

Listing 2: Test Case 1 MCNP6.3 CSV Output Excerpt

```
nps, x, y, z, u, v, w, erg, tme, wgt
4, 1.00000e+01, 9.73518e+00, -7.35106e-01, 5.02555e-01, 8.63021e-01, -5.13092e-02, 9.25630e+00, 3.03915e-01, 9.99860e-01
8, 1.00000e+01, 5.15268e-01, 3.16868e+00, 9.11695e-01, 6.85703e-02, 4.05105e-01, 1.15030e+00, 5.47689e-01, 1.00000e+00
14, 1.00000e+01, 2.28738e+00, -3.98265e+00, 8.98903e-01, 2.15419e-01, -3.81533e-01, 8.10455e-01, 6.79583e-01, 9.88736e-01
15, 1.00000e+01, -1.56429e+00, -2.48265e+00, 8.90754e-01, -2.40614e-01, -3.85569e-01, 3.58215e+01, 1.13408e-01, 9.94802e-01
16, 1.00000e+01, 3.21534e-01, 5.82221e+00, 7.89360e-01, 3.38530e-02, 6.12996e-01, 2.56723e+01, 1.54767e-01, 1.00000e+00
```

5.2 Test Case 2: Fission Source Sites

A fission-source-site "surface" source test case is performed using the surface-source file created from [Listing 4.11 of 3]. This test case models a near-critical assembly within a concrete facility that might be used to assess criticality accident alarm system (CAAS) behavior. The goal of such an analysis is to use the k-eigenvalue (`KCODE`) calculation to calculate a physically reasonable set of neutron source sites within fissile or fissionable material to enable a subsequent fixed-source calculation to transport those neutrons to the (possibly distant) detector(s).

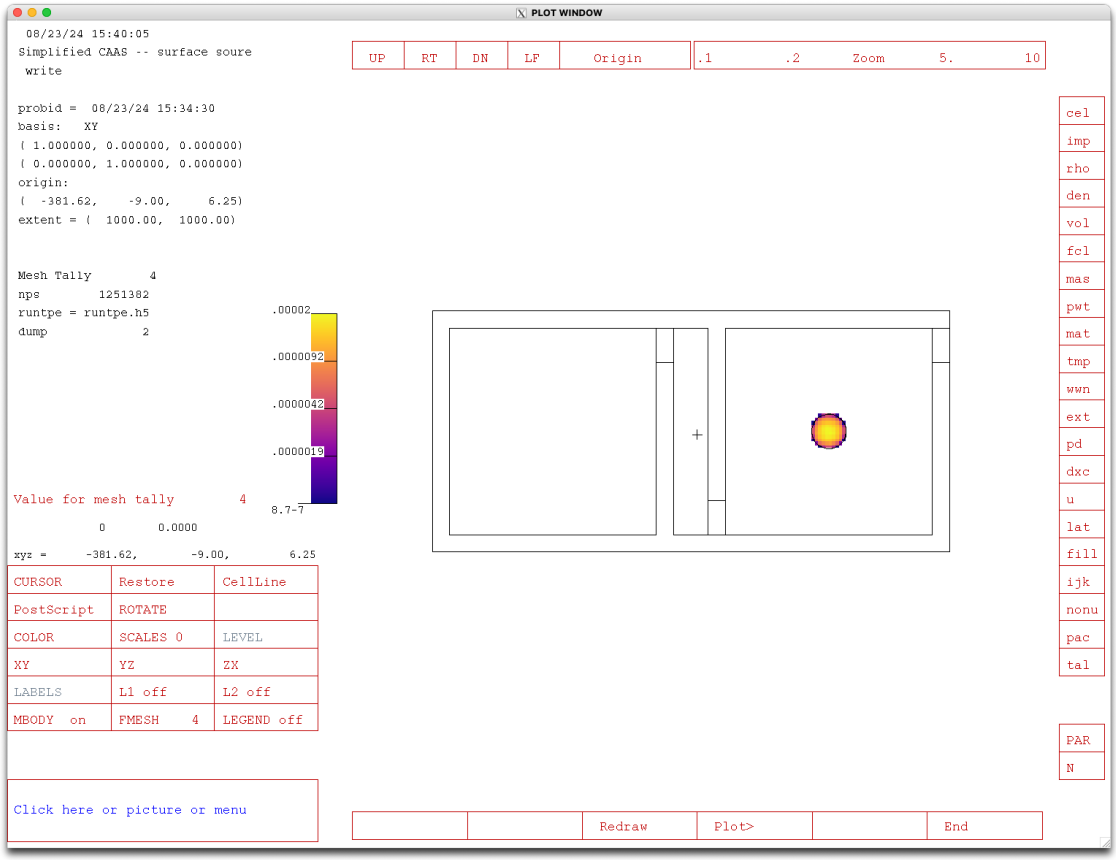


Figure 4: `FMESH4` source distribution.

For MCNP6.2, 6.3, and 6.4, all calculations yield an `FMESH4` result comparable to [page 89 of 3] as shown in Fig. 4 and write a surface-source file containing 1,000,451 tracks corresponding to 1,000,451 histories from 100 active cycles.

A quantitative verification of Requirement 2 is performed by executing [Listing 4.11 of 3] modified for spacing and to run 1,000,451 histories to directly correspond to the surface source written. The first 50 source-particle conditions given in `print table 110` are compared with the `-csv` output from the script herein (Listings 3 and 4). The values match suggesting that the surface-source file is read correctly. Further, a visual verification of the resulting surface source and its energy distribution can be performed using a process similar to the one given in Appendix C that results in Fig. 5, which matches behavior suggested by Figs. 1b and 4.

Furthermore, processing the 1,000,451 histories in this test case takes 18 seconds on the LANL Rocinante high-performance computer with a single processing core on a Intel® Xeon® Platinum 8480+ processor, which is deemed acceptable performance.

Listing 3: Test Case 2 MCNP6.3 Print Table 110 Excerpt

```
1starting mcrun.      cp0 =  0.09                                                                print table 110

      Simplified CAAS -- surface soure read


     nps     x          y          z          cell       surf     u          v          w          energy     weight      time

       1 -1.782E+00  2.552E+01  4.818E+00      100          0 -7.729E-01  4.665E-01  4.302E-01  7.069E-01  9.729E-01  0.000E+00
       2  1.417E+01  1.784E+01  8.727E+00      100          0  6.667E-02  7.865E-01  6.140E-01  2.137E+00  9.729E-01  0.000E+00
       3  1.424E+01  1.832E+01  8.833E+00      100          0  7.910E-01  5.716E-01 -2.182E-01  1.348E+00  9.729E-01  0.000E+00
       4  4.218E+00 -1.877E-01  1.000E+01      100          0  7.245E-01  6.568E-01 -2.093E-01  1.505E+00  9.729E-01  0.000E+00
       5  4.064E+00 -1.767E-01  1.009E+01      100          0  7.291E-01 -4.745E-01  4.932E-01  6.872E-01  9.729E-01  0.000E+00
```

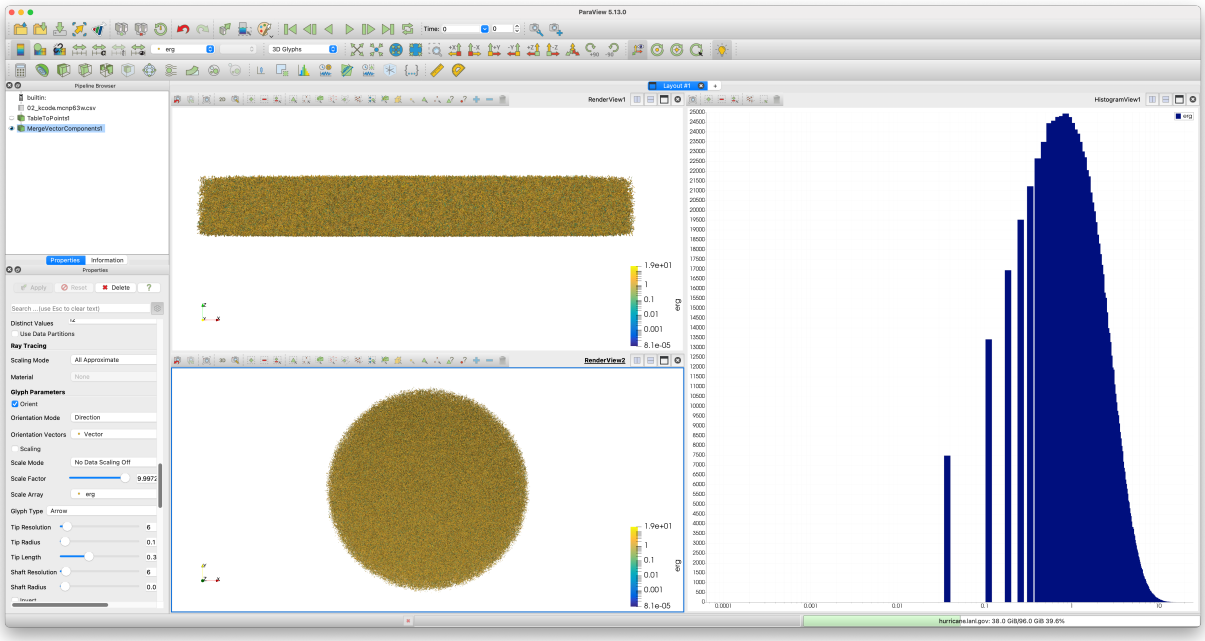


Figure 5: Test case 2 visual verification.

Listing 4: Test Case 2 MCNP6.3 CSV Output Excerpt

```
nps, x, y, z, u, v, w, erg, tme, wgt
1, -1.78210e+00, 2.55236e+01, 4.81845e+00, -7.72853e-01, 4.66460e-01, 4.30248e-01, 7.06888e-01, 0.00000e+00, 9.72857e-01
2, 1.41654e+01, 1.78371e+01, 8.72733e+00, 6.66662e-02, 7.86516e-01, 6.13961e-01, 2.13682e+00, 0.00000e+00, 9.72857e-01
3, 1.42360e+01, 1.83206e+01, 8.83349e+00, 7.90967e-01, 5.71645e-01, -2.18159e-01, 1.34809e+00, 0.00000e+00, 9.72857e-01
4, 4.21775e+00, -1.87744e-01, 1.00026e+01, 7.24470e-01, 6.56752e-01, -2.09333e-01, 1.50487e+00, 0.00000e+00, 9.72857e-01
5, 4.06391e+00, -1.76659e-01, 1.00882e+01, 7.29132e-01, -4.74472e-01, 4.93196e-01, 6.87236e-01, 0.00000e+00, 9.72857e-01
```

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 read MCNP6.2, 6.3, and (planned) 6.4 surface-source files.

References

[Citing pages are listed after each reference.]

1. J. A. Kulesza, T. R. Adams, J. C. Armstrong, S. R. Bolding, F. B. Brown, J. S. Bull, T. P. Burke, A. R. Clark, R. A. Forster, III, J. F. Giron, T. S. Grieve, C. J. Josey, R. L. Martz, G. W. McKinney, E. J. Pearson, M. E. Rising, C. J. Solomon, Jr., S. Swaminarayan, T. J. Trahan, S. C. Wilson, and A. J. Zukaitis, "MCNP® Code Version 6.3.0 Theory & User Manual," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-22-30006, Rev. 1, Sep. 2022. DOI: 10.2172/1889957 [Page 1]

2. T. J. Trahan, "MCNP Surface Source Write/Read File Format Primer," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-16-20109, Jan. 2016. [Page 1]

3. K. L. Currie and M. E. Rising, "MCNP6 Source Primer: Release 1.0," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-18-21377, Rev. 1, Feb. 2020. DOI: 10.2172/1599011 [Pages 3, 4, 5, 15, 16, and 19]

A Script Source Code

The source code for the script described herein is given in Listing 5. 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.

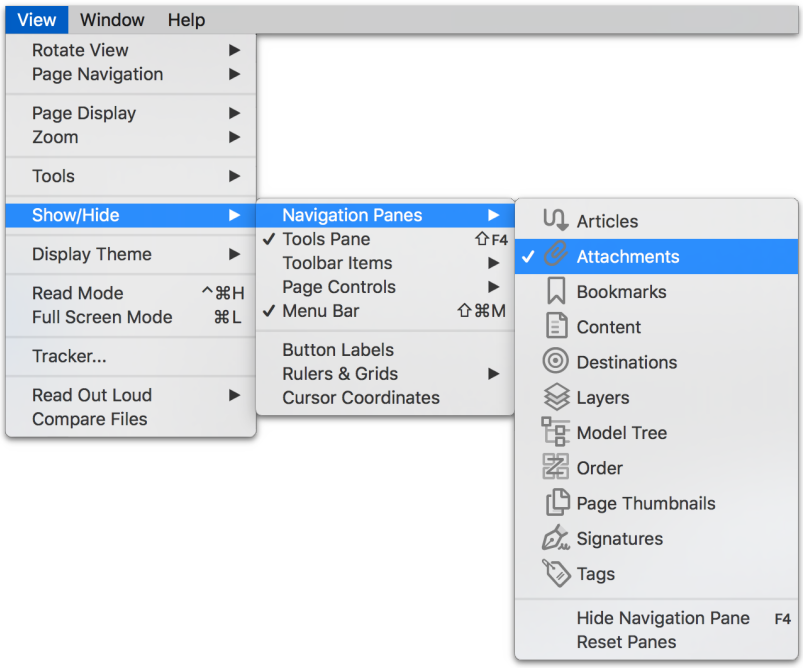


Figure 6: Adobe Acrobat menu path to access PDF attachments.

Listing 5: Script Source Code

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

"""
This module provides a ‘read_ssa_file‘ method that can return a list of particle
objects with phase-space state information as read from an MCNP surface source
file created from either fixed-source or k-eigenvalue calculations.
"""

import argparse
import math
import os
import sys
import textwrap

import __main__ as main
from scipy.io import FortranFile

INT_INF = int(1e20)


class Particle:
    """
```

```
    This simple class, effectively a ‘dataclass‘ holds particle phase-space
    state information that can be extended to enable other capability.
    """

    def __init__(
        self,
        nps,
        erg,
        tme,
        wgt,
        x=None,
        y=None,
        z=None,
        u=None,
        v=None,
        w=None,
        cs=None,
    ):
        self.nps = nps
        self.x = x
        self.y = y
        self.z = z
        self.u = u
        self.v = v
        self.w = w
        self.erg = erg
        self.tme = tme
        self.wgt = wgt
        self.cs = cs

    def __str__(self):
        FMT = ".5e"
        s = f"{self.nps}, "
        s += ", ".join(
            [
                f"{i:{FMT}}"
                for i in (
                    self.x,
                    self.y,
                    self.z,
                    self.u,
                    self.v,
                    self.w,
                    self.erg,
                    self.tme,
                    self.wgt,
                )
            ]
        )
        return s


def read_ssa_file(filename, max_histories=INT_INF):
    """Reads a MCNP surface source file and returns a list of Particle objects.
```

```
    Args:
        filename (str): The path to the SSA file.
        max_histories (int, optional): The maximum number of histories to read
        from the file. Defaults to infinity.

    Returns:
        list[Particle]: A list of Particle objects.
    """

    # Open binary MCNP surface-source file for reading
    f = FortranFile(filename)

    # Read format identifier.
    _ = f.read_record("a8")

    # Read first record: code_name, ver, build_date_code, idtm, probid, aid, knod .
    _ = f.read_record("a8", "a5", "a8", "a19", "a19", "a128", "i4")

    # Read second record: np, nrsw, nd, njss, nqsw
    _, nrsw, _, njss, _ = f.read_record("i8", "i8", "i4", "i4", "i8")
    print(f"Found {nrsw[0]} source points in file.")

    # Read third record.
    nilw, mipt, kq, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _, _ = f.read_record(
        "i4",  # nilw
        "i4",  # mipt
        "i4",  # kq
        "i4",  # 0
        "i4",  # 0
        "i4",  # 0
        "i4",  # 0
        "i4",  # 0
        "i4",  # 0
        "i4",  # 0
        "i4",  # 0
        "i4",  # 0
        "i4",  # 0
        "i4",  # 0
        "i4",  # 0
        "i4",  # 0
        "i4",  # 0
        "i4",  # 0
        "i4",  # 0
        "i4",  # 0
    )

    # Read surface, cell, and summary information.
    if kq[0] == 0:
        for _ in range(njss[0]):
            # Read fourth record: jss, surface_type, n, coeffs.
            _, _, _, _ = f.read_record("i4", "i4", "i4", "f8")
    else:
        for _ in range(njss[0]):
```

```
            # Read fourth record: jss, jsq, surface_type, n, coeffs.
            _, _, _, _, _ = f.read_record("i4", "i4", "i4", "i4", "f8")

    # Read selected surface record: jss.
    for _ in range(nilw[0]):
        _, _, _, _ = f.read_record("i4", "i4", "i4", "f8")

    # Read summary record: nsl.
    FMT = f"({2+4*mipt[0]},{njss[0]+nilw[0]})i4"
    _, _ = f.read_record("f8", FMT)

    # Loop over the lesser of all histories or the user-defined number of
    # histories.
    max_histories = min(nrsw[0], max_histories)
    print(f"Processing {max_histories} histories...")
    particles = [None] * max_histories
    for i in range(max_histories):
        tmp = f.read_record("f8")

        # Symmetric source on spherical surface.
        if len(tmp) == 7:
            a = abs(int(tmp[0]))  # nps
            b = tmp[1]
            wgt = tmp[2]
            erg = tmp[3]
            tme = tmp[4]
            cs = tmp[5]
            _ = tmp[6]  # c, unused

            particles[i] = Particle(a, erg, tme, wgt, cs=cs)

        # General surface source.
        elif len(tmp) == 11:
            a = abs(int(tmp[0]))  # nps
            b = tmp[1]
            wgt = tmp[2]
            erg = tmp[3]
            tme = tmp[4]
            x = tmp[5]
            y = tmp[6]
            z = tmp[7]
            u = tmp[8]
            v = tmp[9]
            _ = tmp[10]  # c, unused

            w = math.copysign(math.sqrt(1.0 - u * u - v * v), b)
            particles[i] = Particle(
                nps=a,
                erg=erg,
                tme=tme,
                wgt=wgt,
                x=x,
                y=y,
                z=z,
```

```
                u=u,
                v=v,
                w=w,
            )

        else:
            print("Error, unknown particle record length.")

    return particles


def write_csv(particle_list, outfilename):
    """
    Writes the phase-space state information for each particle in a given list
    of Particle objects to a CSV file.

    Args:
        particle_list: A list of 'Particle' objects.
        outfilename: The name of the CSV file to write to.
    """

    with open(outfilename, "w", encoding="utf-8") as f:
        f.write("nps, x, y, z, u, v, w, erg, tme, wgt\n")
        for particle in particle_list:
            f.write(str(particle) + "\n")


def write_erg(particle_list, outfilename, nbins=30):
    """
    Generates a PDF plot of a histogram of the energy distribution for a list of
    of particles.

    Args:
        particle_list: A list of 'Particle' objects.
        outfilename: The name of the PDF file to write to.
        nbins: The number of bins to use for energy histogram. Defaults to 30.
    """

    import numpy as np
    from matplotlib import pyplot as plt

    erg = np.array([p.erg for p in particle_list])
    bins = np.logspace(np.log10(min(erg)), np.log10(max(erg)), nbins)

    plt.figure(figsize=(6.5, 6.5 / 1.62))
    plt.hist(
        erg,
        bins=bins,
    )

    plt.xscale("log")

    plt.xlabel("Energy, $E$ [MeV]")
    plt.ylabel("Frequency")
```

```
239
240     plt.savefig(outfilename, bbox_inches="tight")
241
242
243 ###############################################################################
244 # EXECUTE PROGRAM #############################################################
245 ###############################################################################
246
247
248 if __name__ == "__main__" and hasattr(main, "__file__"):
249
250     ############################################################
251     # Command line parsing.
252     ############################################################
253
254     description = textwrap.dedent(
255         """
256     This script is used to read an MCNP6.3-produced surface-source file and
257     create a CSV giving the phase-space information for each of the source
258     points found.
259     """
260     )
261
262     epilog = textwrap.dedent(
263         """
264     Typical command line calls might look like:
265
266     > python """
267         + os.path.basename(__file__)
268         + """ -n 10000 <inputfilename>
269     """
270         + "\u2063"
271     )
272
273     parser = argparse.ArgumentParser(
274         formatter_class=argparse.RawDescriptionHelpFormatter,
275         description=description,
276         epilog=epilog,
277     )
278
279     # Required positional argument(s).
280     parser.add_argument("inp", type=str, help="input surface-source file name")
281
282     # Optional named argument(s).
283     parser.add_argument(
284         "--number_of_histories",
285         "-n",
286         type=int,
287         default=INT_INF,
288         help=f"maximum number of points to process (default: {INT_INF})",
289     )
290     parser.add_argument(
291         "--write_csv",
292         "-csv",
```

```
        action="store_true",
        help="write CSV file of particle phase-space state information (default: False)",
    )
    parser.add_argument(
        "--write_erg",
        "-erg",
        action="store_true",
        help="write PDF histogram of particle energy distribution(default: False)",
    )

    args = parser.parse_args()

    # Basic error checking of command line options.
    if not os.path.isfile(args.inp):
        print(f"Error: input file ({args.inp}) not found.")
        sys.exit(1)

    ##############################################################
    # Program execution.
    ##############################################################

    particles = read_ssa_file(args.inp, max_histories=args.number_of_histories)

    # Create CSV of particle data.
    if args.write_csv:
        print("Writing CSV...")
        write_csv(particles, f"{args.inp}.csv")

    # Write histogram of particle energies.
    if args.write_erg:
        print("Writing energy histogram...")
        write_erg(particles, f"{args.inp}.pdf")
```

B Test Case Files

B.1 "Traditional" Surface Source

The input file for the "traditional" surface source for neutrons is given in Listing 6 and is reproduced from [Listing 4.7 of 3] with only minor spacing changes.

Listing 6: 01a_sswprot.mcnp.inp

```
Proton source on Tungsten target
c
c Cell Cards
c
10 0          +1    -9 IMP:H,N=1
20 0          -1 +2 -9 IMP:H,N=1
30 100 -19.25    -2    IMP:H,N=1
99 0                -9 IMP:H,N=0

c
c Surface Cards
c
1 PX 10
2 RPP 2.5 5 -2.5 2.5 -2.5 2.5
9 RPP -15 15 -15 15 -15 15

c
c Data Cards
c
MODE H N
NPS 1e5
SSW 1 PTY=N
c
PHYS:H 150
SDEF PAR=H ERG=150 VEC=1 0 0 DIR=1
c
c Materials
c
M100 74184 1.0
c
FMESH4:H GEOM=XYZ ORIGIN=-5 -5 -5
         IMESH= 5 IINTS=150
         JMESH= 5 JINTS=150
         KMESH= 5 KINTS=150
         OUT=NONE
c
FMESH14:N GEOM=XYZ ORIGIN=-15 -15 -15
         IMESH=15 IINTS=150
         JMESH=15 JINTS=150
         KMESH=15 KINTS=150
         OUT=NONE
```

To verify behavior of Listing 5, a subsequent calculation is made using the MCNP input file in Listing 7, which is reproduced from [Listing 4.9 of 3] with minor spacing changes and changing the `NPS` card to process 100,000 histories.

Listing 7: 01b_ssrneut.mcnp.inp

```
Neutron spallation source read
c
c Cell Cards
c
10 0          -11              -9 IMP:N=1
20 0          +11 -2           -9 IMP:N=1
30 100 -1.0       +2    +4 -5 -9 IMP:N=1
40 200 -7.874     +2 +3 -4 -5    IMP:N=1
50 0              +2 -3    -5    IMP:N=1
60 0              +5           -9 IMP:N=1
99 0                           +9 IMP:N=0

c
c Surface Cards
c
11 PX 5
2 PX 10
3 CX 2.5
4 CX 7.5
5 PX 60
9 RPP 0 100 -25 25 -25 25

c
c Data Cards
c
MODE N
NPS 100000
SSR NEW 11 TR=1
TR1 -5 0 0
c
c Materials
c
M100 1001 2 8016 1 5010 2
MT100 lwtr
M200 26056 1.0
c
F4:N 60
E4 1E-6 99ilog 30
c
FMESH14:N GEOM=XYZ ORIGIN= 0 -25 -25
          IMESH=100 IINTS=200
          JMESH=25 JINTS=50
          KMESH=25 KINTS=50
          OUT=NONE
c
PRINT
```

B.2 Fission Source Sites

The input file for a fission-source-site "surface" source for neutrons is given in Listing 8 and is reproduced from [Listing 4.11 of 3] with only minor spacing changes.

Listing 8: 02a_kcode.mcnp.inp

```
Simplified CAAS -- surface soure write
c ### cells
c
c >>>>> accident tank
c
100 1 9.9270e-2 -10 -12 imp:n=1
101 3 4.8333e-5 -10 +12 imp:n=1
102 2 8.6360e-2 +10 -11 imp:n=1
c
c >>>>> facility rooms: nw. -> ne., sw. -> se.
c
200 3 4.8333e-5 -20 imp:n=1
210 3 4.8333e-5 -21 +11 imp:n=1
220 3 4.8333e-5 -22 imp:n=1
c
c >>>>> doorways
c
260 3 4.8333e-5 -30 imp:n=1
261 3 4.8333e-5 -31 imp:n=1
262 3 4.8333e-5 -32 imp:n=1
c
c >>>>> facility and rest of world
c
900 4 0.0764 -99 +20 +21 +22 +30 +31 +32 imp:n=1
999 0 +99 imp:n=0

c ### surfaces
c
c >>>>> critical experiment tank
10 rcc 0 0 1 0 0 100 50
11 rcc 0 0 0 0 0 101 50.5
12 pz 13.6
c
c >>>>> rpp's for the empty space in the rooms
20 rpp -1100 -500 -300 300 0 300
21 rpp -300 300 -300 300 0 300
22 rpp -450 -350 -300 300 0 300
c
c >>>>> doorways
30 rpp -350 -300 -300 -200 0 250
31 rpp 300 350 200 300 0 250
32 rpp -500 -450 200 300 0 250
c
c >>>>> building structure
99 rpp -1150 350 -350 350 -50 310

mode n
kcode 10000 1.0 25 125
ksrc 0 0 7
c
ssw cel = 100
c
```

```
fmesh4:n geom=xyz origin=-1150 -350 -50
         imesh=350 iints=150
         jmesh=350 jints=70
         kmesh=310 kints=36
         type=source
c
fmesh14:n geom=xyz origin=-1150 -350 -50
          imesh=350 iints=150
          jmesh=350 jints=70
          kmesh=310 kints=36
c
c ### materials
c plutonium nitrate solution
m1      1001 6.0070e-2
        8016 3.6540e-2
        7014 2.3699e-3
       94239 2.7682e-4
       94240 1.2214e-5
       94241 8.3390e-7
       94242 4.5800e-8
mt1 lwtr
c stainless steel
m2     24050 7.1866e-4 $ Cr-50 4.345%
       24052 1.3859e-2 $ Cr-52 83.789%
       24053 1.5715e-3 $ Cr-53 9.501%
       24054 3.9117e-4 $ Cr-54 2.365%
       26054 3.7005e-3 $ Fe-54 5.845%
       26056 5.8090e-2 $ Fe-56 91.754%
       26057 1.3415e-3 $ Fe-57 2.119%
       26058 1.7853e-4 $ Fe-58 0.282%
       28058 4.4318e-3 $ Ni-58 68.0769%
       28060 1.7071e-3 $ Ni-60 26.2231%
       28061 7.4207e-5 $ Ni-61 1.1399%
       28062 2.3661e-4 $ Ni-62 3.6345%
       28064 6.0256e-5 $ Ni-64 0.9256%
c dry air (typical of American Southwest)
m3      1001 1.7404E-10
        1002 1.3065E-14
        2003 8.3540E-16
        2004 4.5549E-10
        6000 1.11008E-08
        7014 3.8981E-05
        7015 1.3515E-07
        8016 9.1205E-06
        8017 3.4348E-09
       18036 3.0439E-10
       18038 5.3915E-11
       18040 8.0974E-08
       36078 1.7811E-14
       36080 1.1164E-13
       36082 5.6154E-13
       36083 5.49985E-13
       36084 2.69359E-12
       36086 7.98498E-13
```

```
       54124 2.30549E-13
mt3 lwtr
c los alamos concrete
m4      1001 0.00842
        8016 0.04423
       13027 0.00252
       14028 0.014690958
       14029 0.000718176
       14030 0.000460866
       11023 0.00105
       20040 2.84037E-03
       20042 1.89571E-05
       20043 3.95550E-06
       20044 6.11198E-05
       20046 1.17200E-07
       20048 5.47910E-06
       26054 0.000041788
       26056 0.000632003
       26057 0.000014347
       26058 0.000001862
       19039 6.43481E-04
       19040 8.07300E-08
       19041 4.64384E-05
mt4 lwtr
```

To verify behavior of Listing 5, a subsequent calculation is made using the MCNP input file in Listing 9, which is reproduced from [Listing 4.13 of 3] with minor spacing changes and changing the `NPS` card to process 1,000,451 histories.

Listing 9: 02b_fixed_src.mcnp.inp

```
Simplified CAAS -- surface soure read
c ### cells
c
c >>>>> accident tank
c
100 1 9.9270e-2 -10 -12 imp:n=1
101 3 4.8333e-5 -10 +12 imp:n=1
102 2 8.6360e-2 +10 -11 imp:n=1
c
c >>>>> facility rooms: nw. -> ne., sw. -> se.
c
200 3 4.8333e-5 -20 imp:n=1
210 3 4.8333e-5 -21 +11 imp:n=1
220 3 4.8333e-5 -22 imp:n=1
c
c >>>>> doorways
c
260 3 4.8333e-5 -30 imp:n=1
261 3 4.8333e-5 -31 imp:n=1
262 3 4.8333e-5 -32 imp:n=1
c
c >>>>> facility and rest of world
c
900 4 0.0764 -99 +20 +21 +22 +30 +31 +32 imp:n=1
```

```
999 0 +99 imp:n=0

c ### surfaces
c
c >>>>> critical experiment tank
10 rcc 0 0 1 0 0 100 50
11 rcc 0 0 0 0 0 101 50.5
12 pz 13.6
c
c >>>>> rpp's for the empty space in the rooms
20 rpp -1100 -500 -300 300 0 300
21 rpp -300 300 -300 300 0 300
22 rpp -450 -350 -300 300 0 300
c
c >>>>> doorways
30 rpp -350 -300 -300 -200 0 250
31 rpp 300 350 200 300 0 250
32 rpp -500 -450 200 300 0 250
c
c >>>>> building structure
99 rpp -1150 350 -350 350 -50 310

nps 1000451
mode n p
nonu 0 10r
c
ssr cel = 100 psc = 0.5
c
fmesh4:n geom=xyz origin=-1150 -350 -50
         imesh=350 iints=150
         jmesh=350 jints=70
         kmesh=310 kints=36
         type=source
         out=none
c
fmesh14:n geom=xyz origin=-1150 -350 -50
          imesh=350 iints=150
          jmesh=350 jints=70
          kmesh=310 kints=36
          out=none
c
fmesh24:p geom=xyz origin=-1150 -350 -50
          imesh=350 iints=150
          jmesh=350 jints=70
          kmesh=310 kints=36
          out=none
c
print
c ### materials
c plutonium nitrate solution
m1      1001 6.0070e-2
        8016 3.6540e-2
        7014 2.3699e-3
       94239 2.7682e-4
```

```
       94240 1.2214e-5
       94241 8.3390e-7
       94242 4.5800e-8
mt1 lwtr
c stainless steel
m2     24050 7.1866e-4 $ Cr-50 4.345%
       24052 1.3859e-2 $ Cr-52 83.789%
       24053 1.5715e-3 $ Cr-53 9.501%
       24054 3.9117e-4 $ Cr-54 2.365%
       26054 3.7005e-3 $ Fe-54 5.845%
       26056 5.8090e-2 $ Fe-56 91.754%
       26057 1.3415e-3 $ Fe-57 2.119%
       26058 1.7853e-4 $ Fe-58 0.282%
       28058 4.4318e-3 $ Ni-58 68.0769%
       28060 1.7071e-3 $ Ni-60 26.2231%
       28061 7.4207e-5 $ Ni-61 1.1399%
       28062 2.3661e-4 $ Ni-62 3.6345%
       28064 6.0256e-5 $ Ni-64 0.9256%
c dry air (typical of American Southwest)
m3      1001 1.7404E-10
        1002 1.3065E-14
        2003 8.3540E-16
        2004 4.5549E-10
        6000 1.11008E-08
        7014 3.8981E-05
        7015 1.3515E-07
        8016 9.1205E-06
        8017 3.4348E-09
       18036 3.0439E-10
       18038 5.3915E-11
       18040 8.0974E-08
       36078 1.7811E-14
       36080 1.1164E-13
       36082 5.6154E-13
       36083 5.49985E-13
       36084 2.69359E-12
       36086 7.98498E-13
       54124 2.30549E-13
mt3 lwtr
c los alamos concrete
m4      1001 0.00842
        8016 0.04423
       13027 0.00252
       14028 0.014690958
       14029 0.000718176
       14030 0.000460866
       11023 0.00105
       20040 2.84037E-03
       20042 1.89571E-05
       20043 3.95550E-06
       20044 6.11198E-05
       20046 1.17200E-07
       20048 5.47910E-06
       26054 0.000041788
```

```
       26056 0.000632003
       26057 0.000014347
       26058 0.000001862
       19039 6.43481E-04
       19040 8.07300E-08
       19041 4.64384E-05
mt4 lwtr
```

C CSV to ParaView Energy-Colored Vector Plot

This appendix describes a sequence of steps to read a CSV file produced using the script in Listing 5 with the **`-csv`** option to visualize the particle locations and initial trajectories using ParaView arrow glyphs. The steps taken follow with representative screenshots of the process given in Figs. 7–12. The steps are:

1. Launch ParaView (this work uses ParaView 5.13).
2. Select File → Open → choose the desired **`.csv`** file.
3. If prompted, select the "CSV Reader" (see Fig. 7).
4. Ensure "Detect Numeric Columns," "Use String Delimiter," and "Have Headers" are checked and click "Apply" to load the file. A "SpreadSheetView" of the data should appear as in Fig. 8.
5. Click in the left RenderView and then the Filters menu → Miscellaneous → Table to Points, and assign the X Column, Y Column, and Z Column drop-down menus to **`x`**, **`y`**, and **`z`**, respectively as in Fig. 9. Click Apply.
6. Click in the left RenderView and then the Filters menu → Miscellaneous → Merge Vector Components, and assign the X Array, Y Array, and Z Array drop-down menus to **`u`**, **`v`**, and **`w`**, respectively as in Fig. 10. Click Apply.
7. Ensure the "Toggle advanced properties" ( ) button is clicked to provide the ability to control all properties.
8. Update the Properties as follows:
   (a) Change the Representation to "3D Glyphs" and change the Coloring to **`erg`** (Fig. 11).
   (b) Check "Orient" under Glyph Parameters and change Orientation Vectors from None to Vector (Fig. 12).

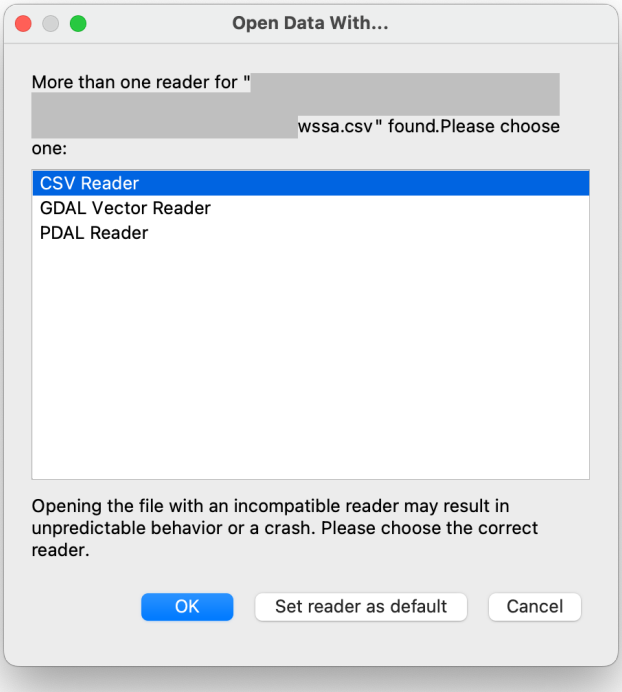


Figure 7: Select CSV reader.

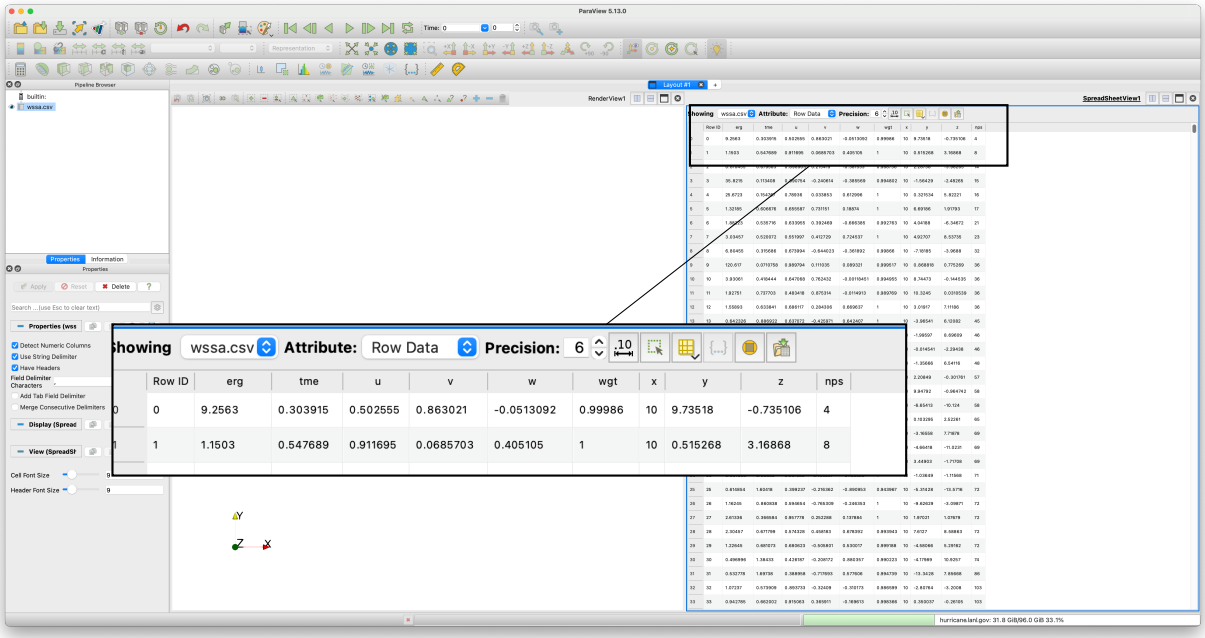


Figure 8: SpreadSheet View.

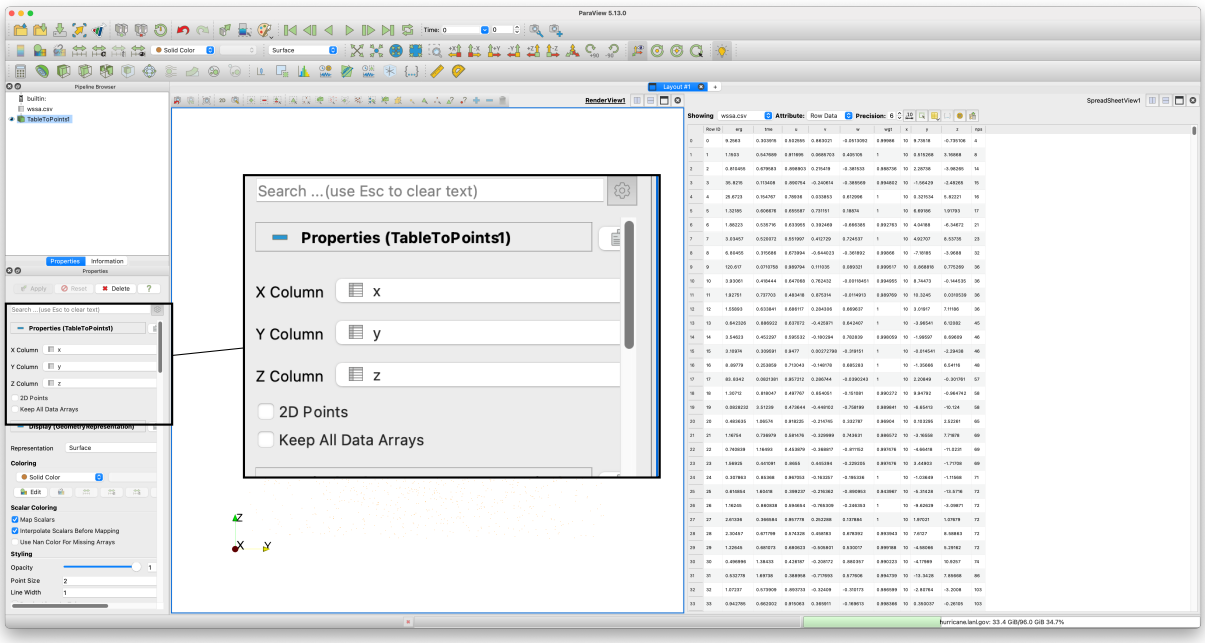


Figure 9: TableToPoints Filter Variable Assignment.

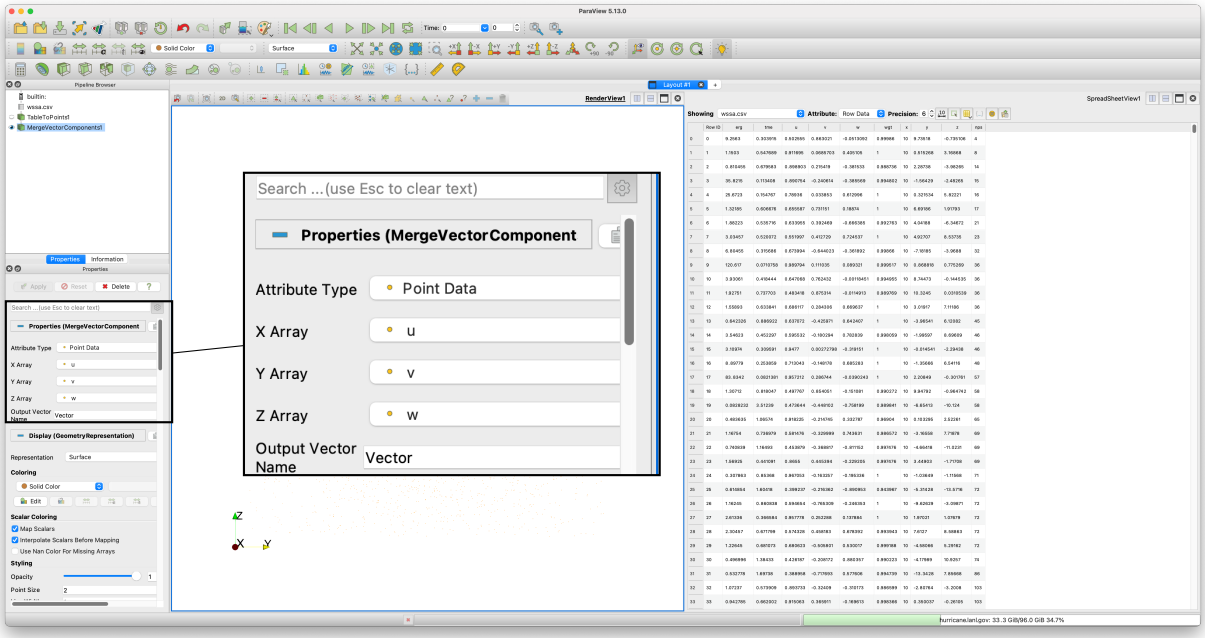


Figure 10: MergeVectorComponents filter variable assignment.

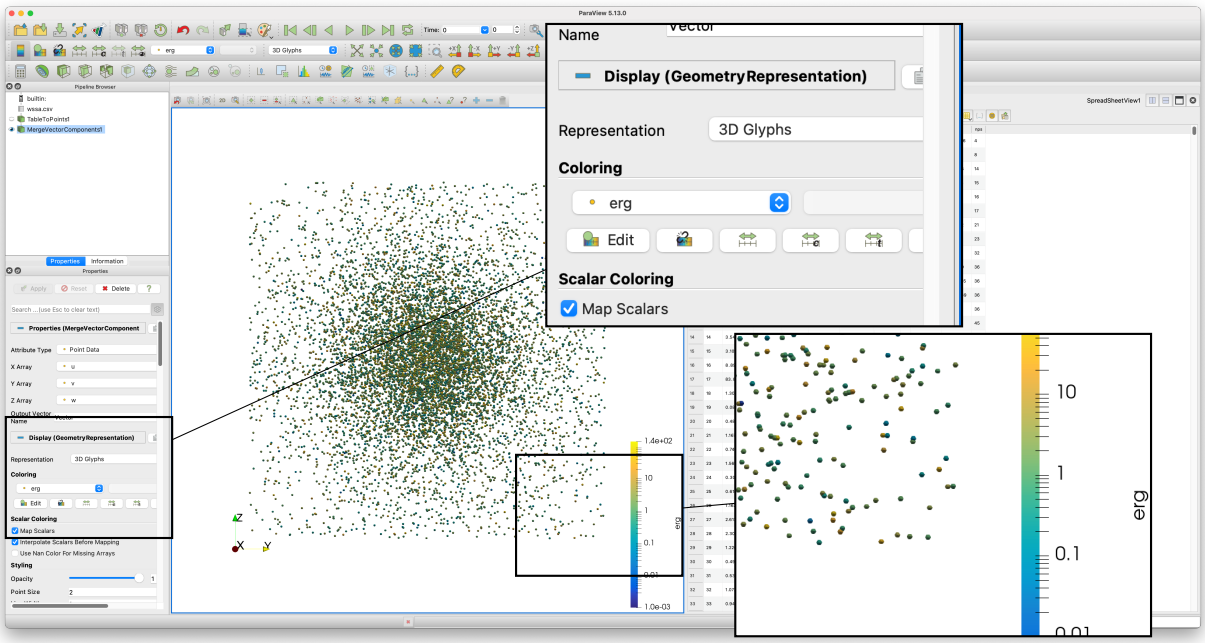


Figure 11: Change representation to "3D Glyphs" and coloring to **erg**.

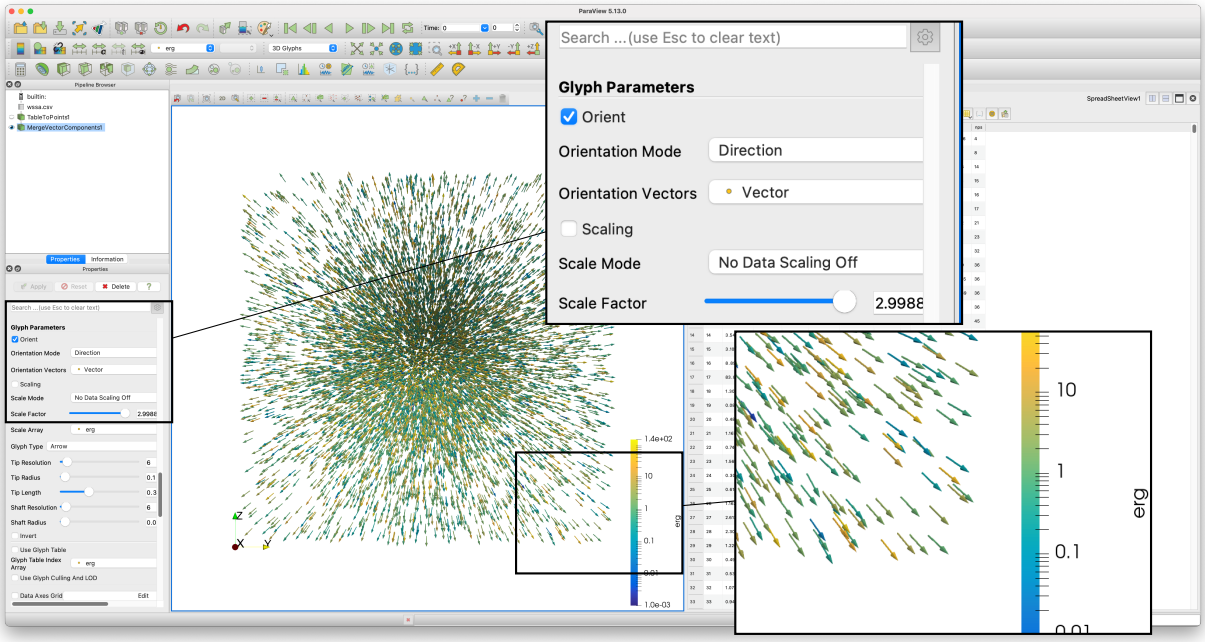


Figure 12: Enable orientation and assign orientation vectors.