

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



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 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:

```
1 python3 Convert_MCNP_ssa.py wssa
```

Output to the terminal is of the form:

```
1 Found 23829 source points in file.  
2 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

```
1 Found 23829 source points in file.  
2 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

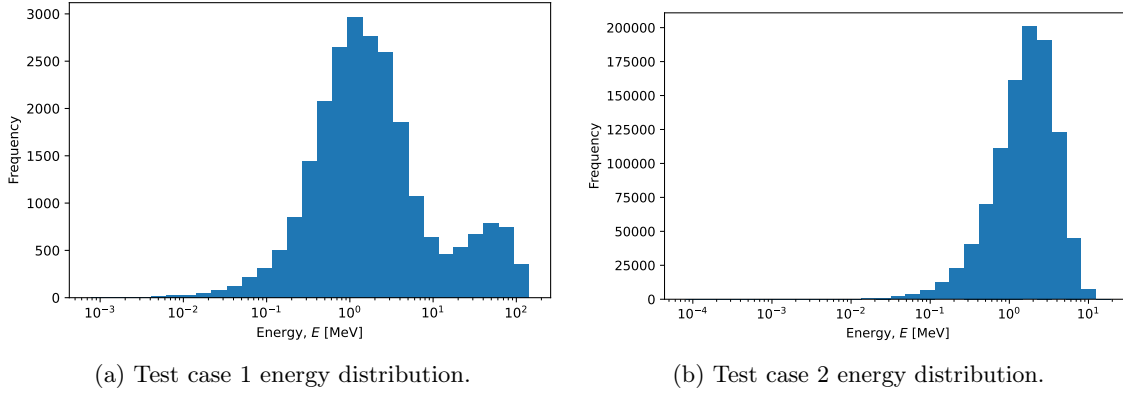


Figure 1: Example energy distributions.

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 Anaconda¹ on macOS version 14.5). The SciPy package is required to enable reading the

¹<https://www.anaconda.com>

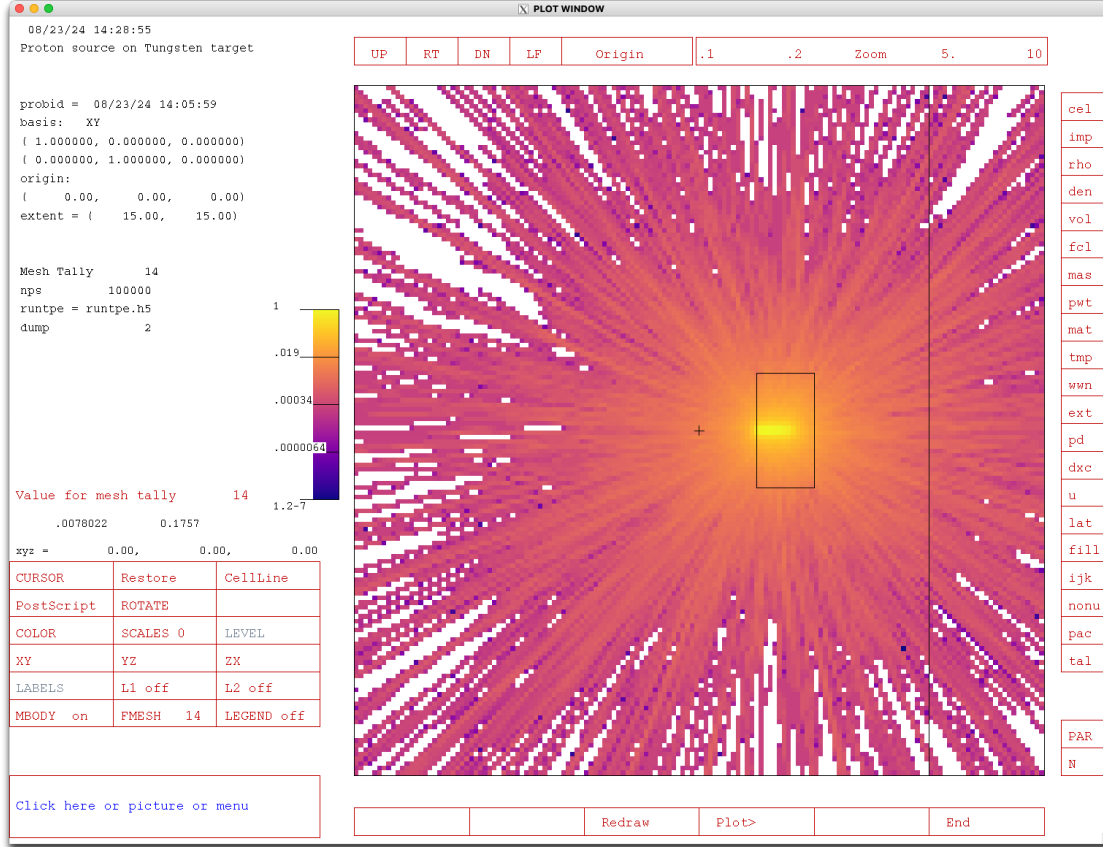


Figure 2: **FMESH14** neutron distribution.

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 **FMESH14** 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

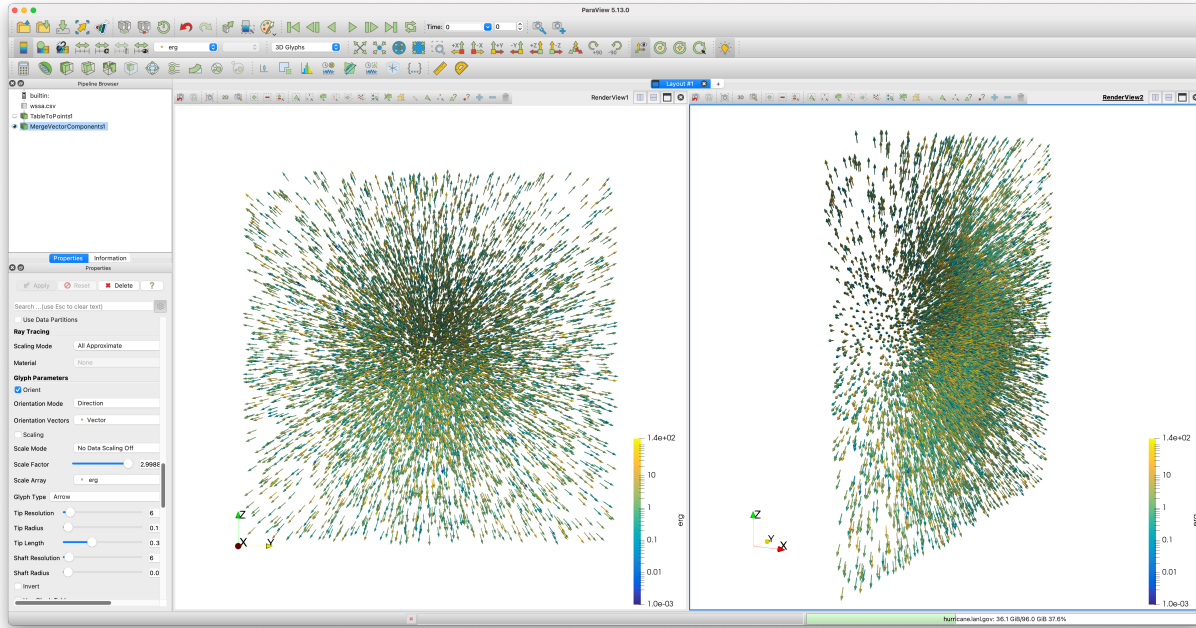


Figure 3: Test case 1 visual verification.

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

1	starting mcrun.					cp0 = 0.02		print table 110					
2													
3	Neutron spallation source read												
4													
5													
6		nps	x	y	z	cell	surf	u	v	w	energy	weight	time
7		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
8		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
9		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
10		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
11		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
12													

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

1	nps, x, y, z, u, v, w, erg, tme, wgt
2	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
3	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
4	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
5	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
6	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).

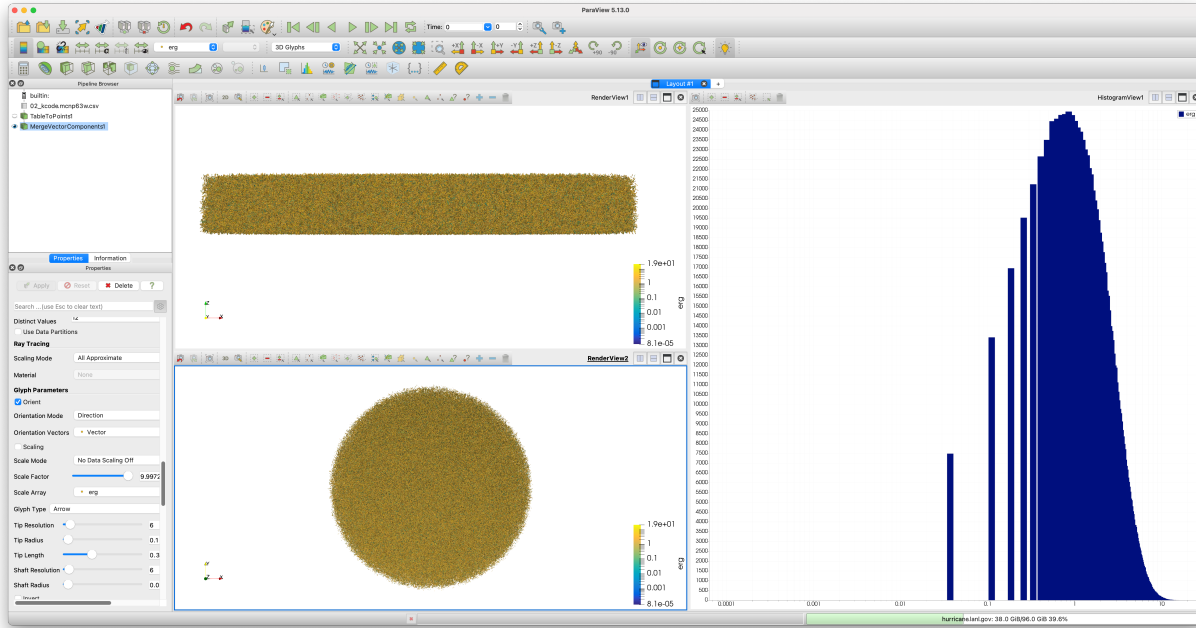


Figure 5: Test case 2 visual verification.

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

```

1 nps, x, y, z, u, v, w, erg, tme, wgt
2 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
3 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
4 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
5 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
6 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](https://doi.org/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](https://doi.org/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.

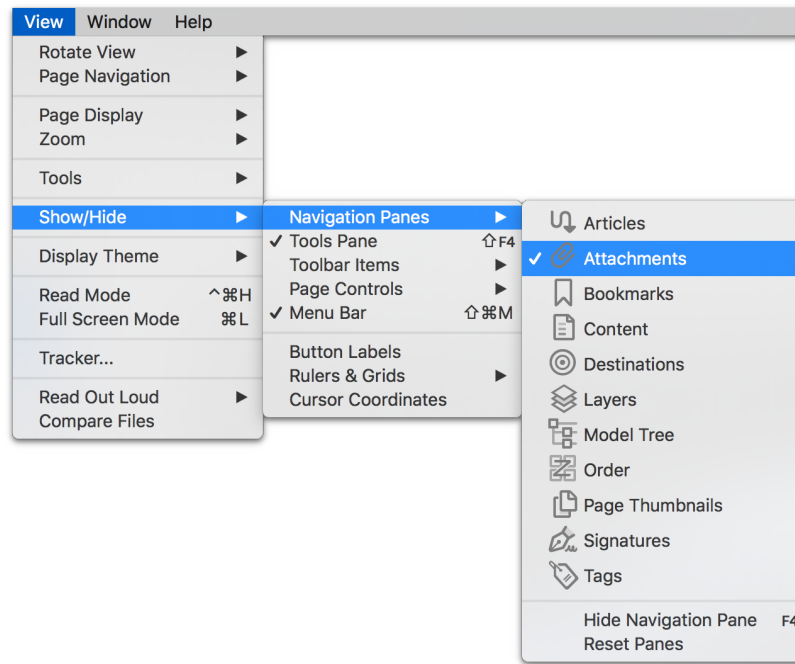


Figure 6: Adobe Acrobat menu path to access PDF attachments.

Listing 5: Script Source Code

```
1 #!/usr/bin/env python
2
3 """
4 This module provides a 'read_ssa_file' method that can return a list of particle
5 objects with phase-space state information as read from an MCNP surface source
6 file created from either fixed-source or k-eigenvalue calculations.
7 """
8
9 import argparse
10 import math
11 import os
12 import sys
13 import textwrap
14
15 import __main__ as main
16 from scipy.io import FortranFile
17
18 INT_INF = int(1e20)
19
20
21 class Particle:
22     """
```

```

23 This simple class, effectively a 'dataclass' holds particle phase-space
24 state information that can be extended to enable other capability.
25 """
26
27 def __init__(
28     self,
29     nps,
30     erg,
31     tme,
32     wgt,
33     x=None,
34     y=None,
35     z=None,
36     u=None,
37     v=None,
38     w=None,
39     cs=None,
40 ):
41     self.nps = nps
42     self.x = x
43     self.y = y
44     self.z = z
45     self.u = u
46     self.v = v
47     self.w = w
48     self.erg = erg
49     self.tme = tme
50     self.wgt = wgt
51     self.cs = cs
52
53 def __str__(self):
54     FMT = ".5e"
55     s = f"{self.nps}, "
56     s += ", ".join(
57         [
58             f"{i:{FMT}}"
59             for i in (
60                 self.x,
61                 self.y,
62                 self.z,
63                 self.u,
64                 self.v,
65                 self.w,
66                 self.erg,
67                 self.tme,
68                 self.wgt,
69             )
70         ]
71     )
72     return s
73
74
75 def read_ssa_file(filename, max_histories=INT_INF):
76     """Reads a MCNP surface source file and returns a list of Particle objects.

```

```

77
78 Args:
79     filename (str): The path to the SSA file.
80     max_histories (int, optional): The maximum number of histories to read
81     from the file. Defaults to infinity.
82
83 Returns:
84     list[Particle]: A list of Particle objects.
85     """
86
87     # Open binary MCNP surface-source file for reading
88     f = FortranFile(filename)
89
90     # Read format identifier.
91     _ = f.read_record("a8")
92
93     # Read first record: code_name, ver, build_date_code, idtm, prohib, aid, knod .
94     _ = f.read_record("a8", "a5", "a8", "a19", "a19", "a128", "i4")
95
96     # Read second record: np, nrsw, nd, njss, nqsw
97     _, nrsw, _, njss, _ = f.read_record("i8", "i8", "i4", "i4", "i8")
98     print(f"Found {nrsw[0]} source points in file.")
99
100    # Read third record.
101    nilw, mipt, kq, _, _, _, _, _, _, _, _, _, _, _, _, _, _ = f.read_record(
102        "i4", # nilw
103        "i4", # mipt
104        "i4", # kq
105        "i4", # 0
106        "i4", # 0
107        "i4", # 0
108        "i4", # 0
109        "i4", # 0
110        "i4", # 0
111        "i4", # 0
112        "i4", # 0
113        "i4", # 0
114        "i4", # 0
115        "i4", # 0
116        "i4", # 0
117        "i4", # 0
118        "i4", # 0
119        "i4", # 0
120        "i4", # 0
121        "i4", # 0
122    )
123
124    # Read surface, cell, and summary information.
125    if kq[0] == 0:
126        for _ in range(njss[0]):
127            # Read fourth record: jss, surface_type, n, coeffs.
128            _, _, _, _ = f.read_record("i4", "i4", "i4", "f8")
129    else:
130        for _ in range(njss[0]):

```

```

131         # Read fourth record: jss, jsq, surface_type, n, coeffs.
132         _, _, _, _, _ = f.read_record("i4", "i4", "i4", "i4", "f8")
133
134     # Read selected surface record: jss.
135     for _ in range(nlw[0]):
136         _, _, _, _ = f.read_record("i4", "i4", "i4", "f8")
137
138     # Read summary record: nsl.
139     FMT = f"({2+4*mip[0]},{njss[0]+nlw[0]})i4"
140     _, _ = f.read_record("f8", FMT)
141
142     # Loop over the lesser of all histories or the user-defined number of
143     # histories.
144     max_histories = min(nrs[0], max_histories)
145     print(f"Processing {max_histories} histories...")
146     particles = [None] * max_histories
147     for i in range(max_histories):
148         tmp = f.read_record("f8")
149
150         # Symmetric source on spherical surface.
151         if len(tmp) == 7:
152             a = abs(int(tmp[0])) # nps
153             b = tmp[1]
154             wgt = tmp[2]
155             erg = tmp[3]
156             tme = tmp[4]
157             cs = tmp[5]
158             _ = tmp[6] # c, unused
159
160             particles[i] = Particle(a, erg, tme, wgt, cs=cs)
161
162         # General surface source.
163         elif len(tmp) == 11:
164             a = abs(int(tmp[0])) # nps
165             b = tmp[1]
166             wgt = tmp[2]
167             erg = tmp[3]
168             tme = tmp[4]
169             x = tmp[5]
170             y = tmp[6]
171             z = tmp[7]
172             u = tmp[8]
173             v = tmp[9]
174             _ = tmp[10] # c, unused
175
176             w = math.copysign(math.sqrt(1.0 - u * u - v * v), b)
177             particles[i] = Particle(
178                 nps=a,
179                 erg=erg,
180                 tme=tme,
181                 wgt=wgt,
182                 x=x,
183                 y=y,
184                 z=z,

```

```

185         u=u,
186         v=v,
187         w=w,
188     )
189
190     else:
191         print("Error, unknown particle record length.")
192
193     return particles
194
195
196 def write_csv(particle_list, outfilename):
197     """
198     Writes the phase-space state information for each particle in a given list
199     of Particle objects to a CSV file.
200
201     Args:
202         particle_list: A list of 'Particle' objects.
203         outfilename: The name of the CSV file to write to.
204     """
205
206     with open(outfilename, "w", encoding="utf-8") as f:
207         f.write("nps, x, y, z, u, v, w, erg, tme, wgt\n")
208         for particle in particle_list:
209             f.write(str(particle) + "\n")
210
211
212 def write_erg(particle_list, outfilename, nbins=30):
213     """
214     Generates a PDF plot of a histogram of the energy distribution for a list of
215     of particles.
216
217     Args:
218         particle_list: A list of 'Particle' objects.
219         outfilename: The name of the PDF file to write to.
220         nbins: The number of bins to use for energy histogram. Defaults to 30.
221     """
222
223     import numpy as np
224     from matplotlib import pyplot as plt
225
226     erg = np.array([p.erg for p in particle_list])
227     bins = np.logspace(np.log10(min(erg)), np.log10(max(erg)), nbins)
228
229     plt.figure(figsize=(6.5, 6.5 / 1.62))
230     plt.hist(
231         erg,
232         bins=bins,
233     )
234
235     plt.xscale("log")
236
237     plt.xlabel("Energy, $E$ [MeV]")
238     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",

```

```

293         action="store_true",
294         help="write CSV file of particle phase-space state information (default: False)",
295     )
296     parser.add_argument(
297         "--write_erg",
298         "-erg",
299         action="store_true",
300         help="write PDF histogram of particle energy distribution(default: False)",
301     )
302
303     args = parser.parse_args()
304
305     # Basic error checking of command line options.
306     if not os.path.isfile(args.inp):
307         print(f"Error: input file ({args.inp}) not found.")
308         sys.exit(1)
309
310     #####
311     # Program execution.
312     #####
313
314     particles = read_ssa_file(args.inp, max_histories=args.number_of_histories)
315
316     # Create CSV of particle data.
317     if args.write_csv:
318         print("Writing CSV...")
319         write_csv(particles, f"{args.inp}.csv")
320
321     # Write histogram of particle energies.
322     if args.write_erg:
323         print("Writing energy histogram...")
324         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

```
1 Proton source on Tungsten target
2 c
3 c Cell Cards
4 c
5 10 0          +1    -9 IMP:H,N=1
6 20 0          -1 +2 -9 IMP:H,N=1
7 30 100 -19.25  -2    IMP:H,N=1
8 99 0          +9    IMP:H,N=0
9
10 c
11 c Surface Cards
12 c
13 1 PX 10
14 2 RPP 2.5 5 -2.5 2.5 -2.5 2.5
15 9 RPP -15 15 -15 15 -15 15
16
17 c
18 c Data Cards
19 c
20 MODE H N
21 NPS 1e5
22 SSW 1 PTY=N
23 c
24 PHYS:H 150
25 SDEF PAR=H ERG=150 VEC=1 0 0 DIR=1
26 c
27 c Materials
28 c
29 M100 74184 1.0
30 c
31 FMESH4:H GEOM=XYZ ORIGIN=-5 -5 -5
32      IMESH= 5 IINTS=150
33      JMESH= 5 JINTS=150
34      KMESH= 5 KINTS=150
35      OUT=NONE
36 c
37 FMESH14:N GEOM=XYZ ORIGIN=-15 -15 -15
38      IMESH=15 IINTS=150
39      JMESH=15 JINTS=150
40      KMESH=15 KINTS=150
41      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

```

1 Neutron spallation source read
2 c
3 c Cell Cards
4 c
5 10 0 -11 -9 IMP:N=1
6 20 0 +11 -2 -9 IMP:N=1
7 30 100 -1.0 +2 +4 -5 -9 IMP:N=1
8 40 200 -7.874 +2 +3 -4 -5 IMP:N=1
9 50 0 +2 -3 -5 IMP:N=1
10 60 0 +5 -9 IMP:N=1
11 99 0 +9 IMP:N=0
12
13 c
14 c Surface Cards
15 c
16 11 PX 5
17 2 PX 10
18 3 CX 2.5
19 4 CX 7.5
20 5 PX 60
21 9 RPP 0 100 -25 25 -25 25
22
23 c
24 c Data Cards
25 c
26 MODE N
27 NPS 100000
28 SSR NEW 11 TR=1
29 TR1 -5 0 0
30 c
31 c Materials
32 c
33 M100 1001 2 8016 1 5010 2
34 MT100 lwtr
35 M200 26056 1.0
36 c
37 F4:N 60
38 E4 1E-6 99ilog 30
39 c
40 FMESH14:N GEOM=XYZ ORIGIN= 0 -25 -25
41 IMESH=100 IINTS=200
42 JMESH=25 JINTS=50
43 KMESH=25 KINTS=50
44 OUT=NONE
45 c
46 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

```

1 Simplified CAAS -- surface source write
2 c ### cells
3 c
4 c >>>> accident tank
5 c
6 100 1 9.9270e-2 -10 -12 imp:n=1
7 101 3 4.8333e-5 -10 +12 imp:n=1
8 102 2 8.6360e-2 +10 -11 imp:n=1
9 c
10 c >>>> facility rooms: nw. -> ne., sw. -> se.
11 c
12 200 3 4.8333e-5 -20 imp:n=1
13 210 3 4.8333e-5 -21 +11 imp:n=1
14 220 3 4.8333e-5 -22 imp:n=1
15 c
16 c >>>> doorways
17 c
18 260 3 4.8333e-5 -30 imp:n=1
19 261 3 4.8333e-5 -31 imp:n=1
20 262 3 4.8333e-5 -32 imp:n=1
21 c
22 c >>>> facility and rest of world
23 c
24 900 4 0.0764 -99 +20 +21 +22 +30 +31 +32 imp:n=1
25 999 0 +99 imp:n=0
26
27 c ### surfaces
28 c
29 c >>>> critical experiment tank
30 10 rcc 0 0 1 0 0 100 50
31 11 rcc 0 0 0 0 0 101 50.5
32 12 pz 13.6
33 c
34 c >>>> rpp's for the empty space in the rooms
35 20 rpp -1100 -500 -300 300 0 300
36 21 rpp -300 300 -300 300 0 300
37 22 rpp -450 -350 -300 300 0 300
38 c
39 c >>>> doorways
40 30 rpp -350 -300 -300 -200 0 250
41 31 rpp 300 350 200 300 0 250
42 32 rpp -500 -450 200 300 0 250
43 c
44 c >>>> building structure
45 99 rpp -1150 350 -350 350 -50 310
46
47 mode n
48 kcode 10000 1.0 25 125
49 ksrc 0 0 7
50 c
51 ssw cel = 100
52 c

```

```

53 fmesh4:n geom=xyz origin=-1150 -350 -50
54     imesh=350 iints=150
55     jmesh=350 jints=70
56     kmesh=310 kints=36
57     type=source
58 c
59 fmesh14:n geom=xyz origin=-1150 -350 -50
60     imesh=350 iints=150
61     jmesh=350 jints=70
62     kmesh=310 kints=36
63 c
64 c ### materials
65 c plutonium nitrate solution
66 m1      1001 6.0070e-2
67          8016 3.6540e-2
68          7014 2.3699e-3
69          94239 2.7682e-4
70          94240 1.2214e-5
71          94241 8.3390e-7
72          94242 4.5800e-8
73 mt1 lwtr
74 c stainless steel
75 m2      24050 7.1866e-4 $ Cr-50 4.345%
76          24052 1.3859e-2 $ Cr-52 83.789%
77          24053 1.5715e-3 $ Cr-53 9.501%
78          24054 3.9117e-4 $ Cr-54 2.365%
79          26054 3.7005e-3 $ Fe-54 5.845%
80          26056 5.8090e-2 $ Fe-56 91.754%
81          26057 1.3415e-3 $ Fe-57 2.119%
82          26058 1.7853e-4 $ Fe-58 0.282%
83          28058 4.4318e-3 $ Ni-58 68.0769%
84          28060 1.7071e-3 $ Ni-60 26.2231%
85          28061 7.4207e-5 $ Ni-61 1.1399%
86          28062 2.3661e-4 $ Ni-62 3.6345%
87          28064 6.0256e-5 $ Ni-64 0.9256%
88 c dry air (typical of American Southwest)
89 m3      1001 1.7404E-10
90          1002 1.3065E-14
91          2003 8.3540E-16
92          2004 4.5549E-10
93          6000 1.11008E-08
94          7014 3.8981E-05
95          7015 1.3515E-07
96          8016 9.1205E-06
97          8017 3.4348E-09
98          18036 3.0439E-10
99          18038 5.3915E-11
100         18040 8.0974E-08
101         36078 1.7811E-14
102         36080 1.1164E-13
103         36082 5.6154E-13
104         36083 5.49985E-13
105         36084 2.69359E-12
106         36086 7.98498E-13

```

```

107      54124 2.30549E-13
108 mt3 lwtr
109 c los alamos concrete
110 m4      1001 0.00842
111      8016 0.04423
112      13027 0.00252
113      14028 0.014690958
114      14029 0.000718176
115      14030 0.000460866
116      11023 0.00105
117      20040 2.84037E-03
118      20042 1.89571E-05
119      20043 3.95550E-06
120      20044 6.11198E-05
121      20046 1.17200E-07
122      20048 5.47910E-06
123      26054 0.000041788
124      26056 0.000632003
125      26057 0.000014347
126      26058 0.000001862
127      19039 6.43481E-04
128      19040 8.07300E-08
129      19041 4.64384E-05
130 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

```

1 Simplified CAAS -- surface source read
2 c ### cells
3 c
4 c >>>> accident tank
5 c
6 100 1 9.9270e-2 -10 -12 imp:n=1
7 101 3 4.8333e-5 -10 +12 imp:n=1
8 102 2 8.6360e-2 +10 -11 imp:n=1
9 c
10 c >>>> facility rooms: nw. -> ne., sw. -> se.
11 c
12 200 3 4.8333e-5 -20 imp:n=1
13 210 3 4.8333e-5 -21 +11 imp:n=1
14 220 3 4.8333e-5 -22 imp:n=1
15 c
16 c >>>> doorways
17 c
18 260 3 4.8333e-5 -30 imp:n=1
19 261 3 4.8333e-5 -31 imp:n=1
20 262 3 4.8333e-5 -32 imp:n=1
21 c
22 c >>>> facility and rest of world
23 c
24 900 4 0.0764 -99 +20 +21 +22 +30 +31 +32 imp:n=1

```

```

25 999 0 +99 imp:n=0
26
27 c ### surfaces
28 c
29 c >>>> critical experiment tank
30 10 rcc 0 0 1 0 0 100 50
31 11 rcc 0 0 0 0 0 101 50.5
32 12 pz 13.6
33 c
34 c >>>> rpp's for the empty space in the rooms
35 20 rpp -1100 -500 -300 300 0 300
36 21 rpp -300 300 -300 300 0 300
37 22 rpp -450 -350 -300 300 0 300
38 c
39 c >>>> doorways
40 30 rpp -350 -300 -300 -200 0 250
41 31 rpp 300 350 200 300 0 250
42 32 rpp -500 -450 200 300 0 250
43 c
44 c >>>> building structure
45 99 rpp -1150 350 -350 350 -50 310
46
47 nps 1000451
48 mode n p
49 nonu 0 10r
50 c
51 ssr cel = 100 psc = 0.5
52 c
53 fmesh4:n geom=xyz origin=-1150 -350 -50
54     imesh=350 iints=150
55     jmesh=350 jints=70
56     kmesh=310 kints=36
57     type=source
58     out=none
59 c
60 fmesh14:n geom=xyz origin=-1150 -350 -50
61     imesh=350 iints=150
62     jmesh=350 jints=70
63     kmesh=310 kints=36
64     out=none
65 c
66 fmesh24:p geom=xyz origin=-1150 -350 -50
67     imesh=350 iints=150
68     jmesh=350 jints=70
69     kmesh=310 kints=36
70     out=none
71 c
72 print
73 c ### materials
74 c plutonium nitrate solution
75 m1      1001 6.0070e-2
76      8016 3.6540e-2
77      7014 2.3699e-3
78      94239 2.7682e-4

```

79	94240	1.2214e-5	
80	94241	8.3390e-7	
81	94242	4.5800e-8	
82	mt1 lwtr		
83	c stainless steel		
84	m2	24050	7.1866e-4 \$ Cr-50 4.345%
85		24052	1.3859e-2 \$ Cr-52 83.789%
86		24053	1.5715e-3 \$ Cr-53 9.501%
87		24054	3.9117e-4 \$ Cr-54 2.365%
88		26054	3.7005e-3 \$ Fe-54 5.845%
89		26056	5.8090e-2 \$ Fe-56 91.754%
90		26057	1.3415e-3 \$ Fe-57 2.119%
91		26058	1.7853e-4 \$ Fe-58 0.282%
92		28058	4.4318e-3 \$ Ni-58 68.0769%
93		28060	1.7071e-3 \$ Ni-60 26.2231%
94		28061	7.4207e-5 \$ Ni-61 1.1399%
95		28062	2.3661e-4 \$ Ni-62 3.6345%
96		28064	6.0256e-5 \$ Ni-64 0.9256%
97	c dry air (typical of American Southwest)		
98	m3	1001	1.7404E-10
99		1002	1.3065E-14
100		2003	8.3540E-16
101		2004	4.5549E-10
102		6000	1.11008E-08
103		7014	3.8981E-05
104		7015	1.3515E-07
105		8016	9.1205E-06
106		8017	3.4348E-09
107		18036	3.0439E-10
108		18038	5.3915E-11
109		18040	8.0974E-08
110		36078	1.7811E-14
111		36080	1.1164E-13
112		36082	5.6154E-13
113		36083	5.49985E-13
114		36084	2.69359E-12
115		36086	7.98498E-13
116		54124	2.30549E-13
117	mt3 lwtr		
118	c los alamos concrete		
119	m4	1001	0.00842
120		8016	0.04423
121		13027	0.00252
122		14028	0.014690958
123		14029	0.000718176
124		14030	0.000460866
125		11023	0.00105
126		20040	2.84037E-03
127		20042	1.89571E-05
128		20043	3.95550E-06
129		20044	6.11198E-05
130		20046	1.17200E-07
131		20048	5.47910E-06
132		26054	0.000041788

133	26056	0.000632003
134	26057	0.000014347
135	26058	0.000001862
136	19039	6.43481E-04
137	19040	8.07300E-08
138	19041	4.64384E-05
139	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).

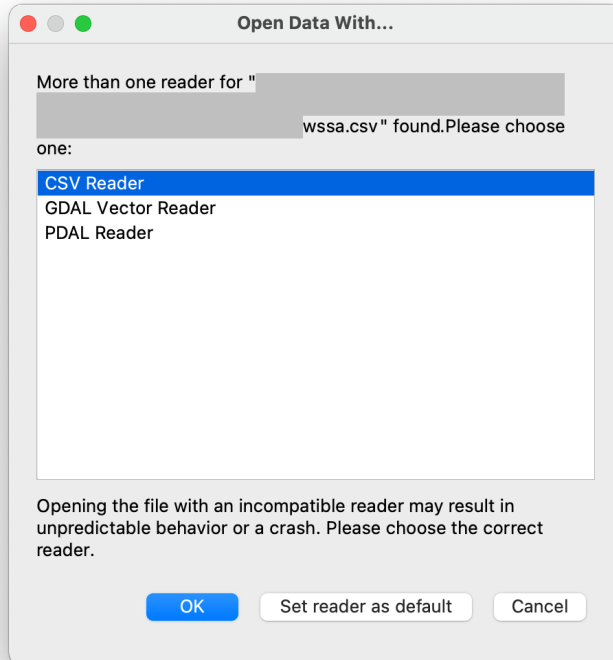


Figure 7: Select CSV reader.

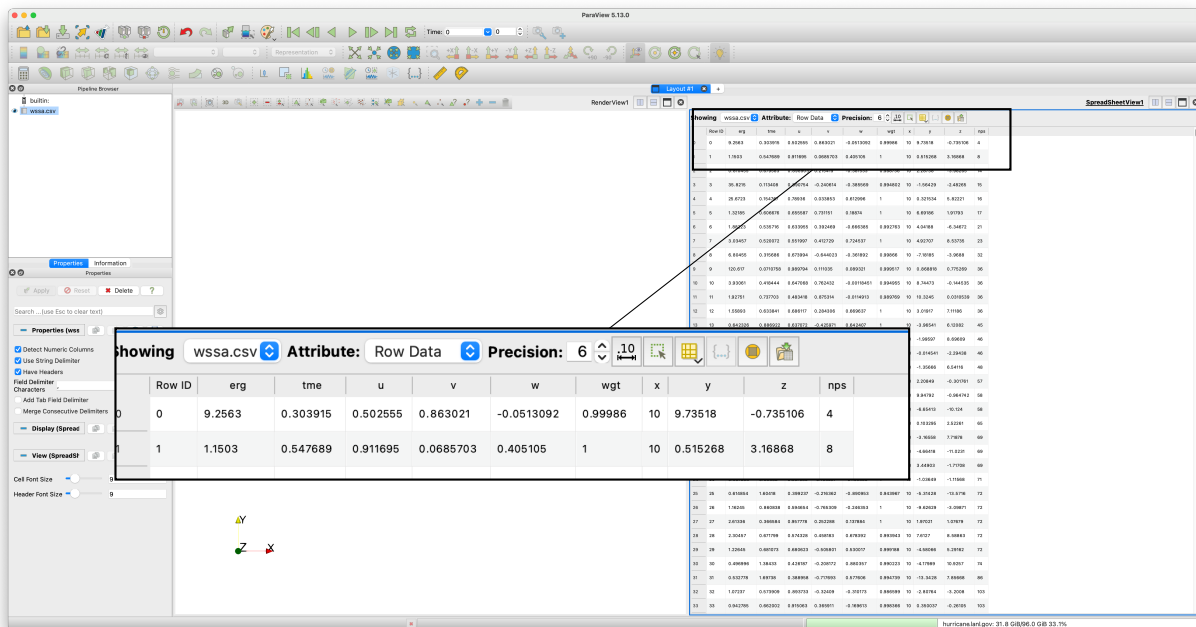


Figure 8: Spreadsheet View.

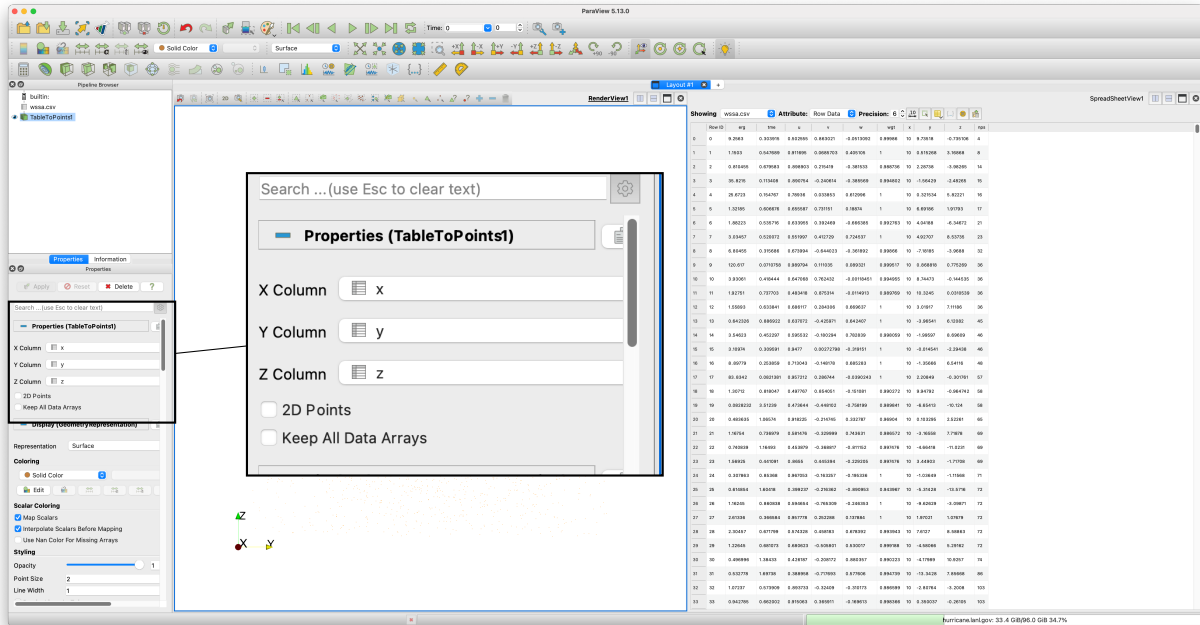


Figure 9: TableToPoints Filter Variable Assignment.

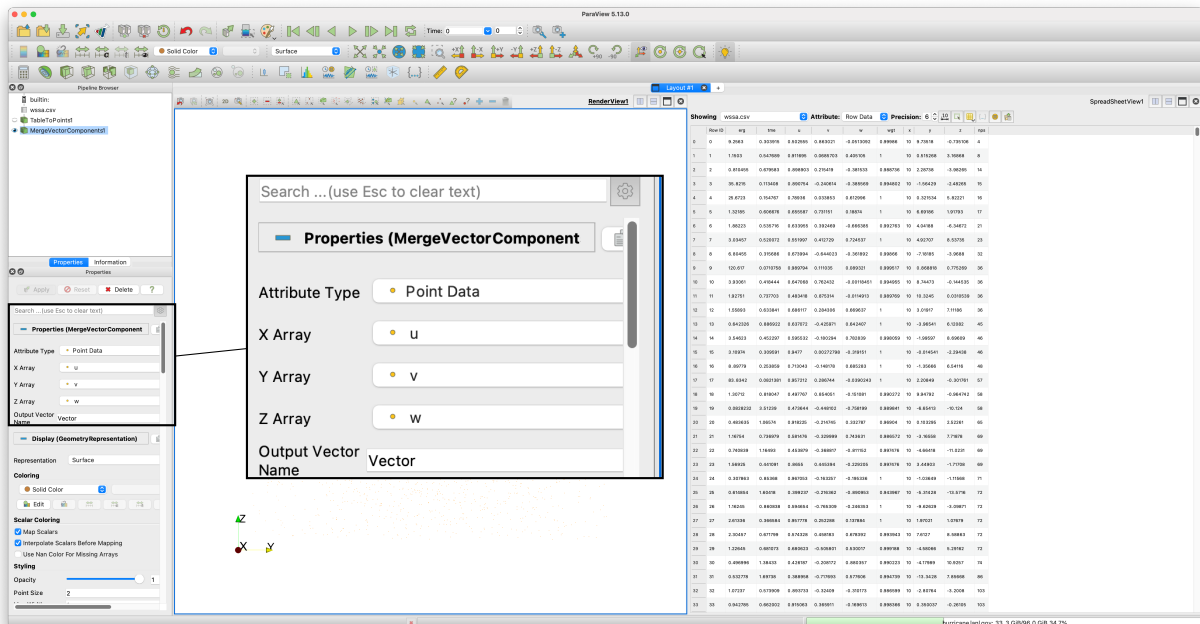


Figure 10: MergeVectorComponents filter variable assignment.

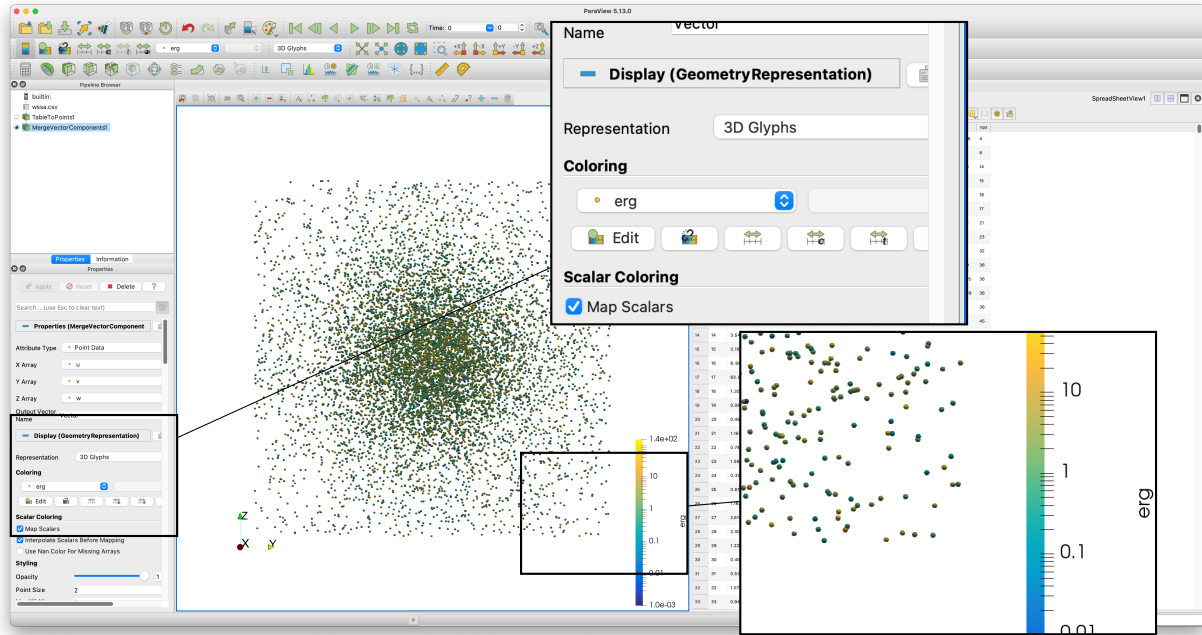


Figure 11: Change representation to “3D Glyphs” and coloring to **erg**.

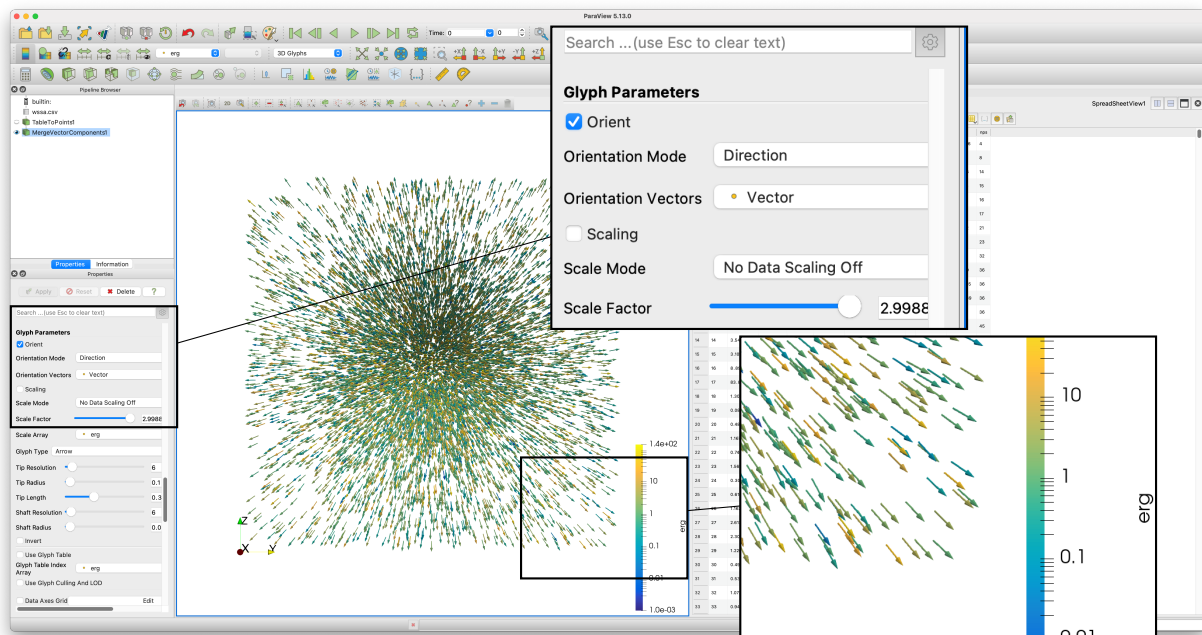


Figure 12: Enable orientation and assign orientation vectors.