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.
MCNP Unstructured Mesh
Overview, Improvement, and
Verification & Validation Testing

Jerawan Armstrong, Joel Kulesza, Colin Josey,
Scott Mosher, Sriram Swaminarayan, Karen Kelley,
Jen Alwin, Josh Spencer, Vedant Mehta

2021 MCNP User Symposium, July 12-16, 2021
Outline

• MCNP Unstructured Mesh Overview
• Code Improvement from 6.2 to 6.3 versions
• Verification & Validation Testing
• MCNP UM Limitations
• Future Work
MCNP Geometry Models

Geometry setup is a crucial step of MCNP simulations!

- **Constructive Solid Geometry (CSG) Model:**
  - Constructed by organizing an arbitrary 3D configuration of materials into geometric cells bounded by surfaces.

- **Hybrid Geometry Model:**
  - Constructed by embedding finite element meshes (structured or unstructured meshes) into CSG cells.
  - Finite element meshes are typically generated by host codes or meshing software packages.
MCNP Unstructured Mesh (UM) Calculations

Mesh Input File Format:
- Abaqus Input [6.0 - 6.3 versions]
- HDF5 [6.3 version]

EEOUT (Element Edit OUTput) File Format:
- Flat ASCII or Binary [6.0 - 6.3 versions]
- HDF5 [6.3 version]

HDF5 EEOOUT = HDF5 Input + Edit Results
MCNP UM Preprocessing & Postprocessing

- An MCNP UM calculation requires two input file types:
  - MCNP input file, &
  - Mesh input file [Abaqus or HDF5]
- An Abaqus input file must have the correct Abaqus syntax rules and meet additional MCNP requirements.
- Any "code" that can export an Abaqus formatted input file may be used to create UM models for MCNP simulations.
Abaqus UM Input Model

Element type: C3D4, C3D6, C3D8, C3D10, C3D15, C3D20, SC8

SC8 is in 6.3 version

elset format/name and material name must meet MCNP requirements
MCNP UM Calculations Using Abaqus

Abaqus Element Types that MCNP 6.0-6.2 versions can process:
- 1\textsuperscript{st} order tet, pent, hex elements
- 2\textsuperscript{nd} order tet, pent, hex elements

- **write\_mcnp\_um\_input.py**
  - performs extensively error checking on an Abaqus input file format.

- **um\_pre\_op**:
  - developed to write an MCNP skeleton input file.

https://www.3ds.com

Abaqus Element Types that MCNP 6.0-6.2 versions can process:
- 1\textsuperscript{st} order tet, pent, hex elements
- 2\textsuperscript{nd} order tet, pent, hex elements

- **write\_mcnp\_um\_input.py**
  - performs extensively error checking on an Abaqus input file format.

- **um\_pre\_op**:
  - developed to write an MCNP skeleton input file.
MCNP UM Calculations Using Attila4MC

SpaceClaim

.sat or .x_t

Attila4MC

Abaqus Input File

MCNP Input File

MCNP Input Setup Using Attila4MC

Attila4MC

https://silverfirsoftware.com

Only 1st order tet elements

“Attila4MC provides an easy-to-use graphical interface, allowing novice and advanced MCNP users to easily set up, run, and visualize MCNP solutions from CAD data.”

silverfirsoftware.com
MCNP UM Calculations Using CUBIT

CUBIT is the Sandia National Laboratory automated mesh generation toolkit

Volume-Material Files

MCNP card files: data_card, surface_card, and/or cell_card

Abaqus Input File

cubit_to_mcnp.py

Abaqus Input File for MCNP

write_mcnp_um_input.py

MCNP Input File

MCNP6

Version 15.6: 1st order tet & hex and 2nd order tet elements.

https://cubit.sandia.gov
MCNP CSG & UM Input files

CSG Input File
1 input file type

UM Input Files
2 input file types

mgeoin = mesh input file
meeout = elemental edit output file
Three Options to Produce EEOUT Files: Option I

• Option 1: create an ASCII EEOUT file

 embed2 meshgeo=abaqus
   mgeoin=example.inp
   meeout=example.eeout
   background= 20
   matcell= 1 10 2 11 3 12 4 13

   Produce an ASCII EEOUT file:
     example.eeout

• Letters in file names on EMBED card must be lowercase letters.
• MCNP overwrites an old ASCII EEOUT file.
Three Options to Produce EEOUT Files: Option II

• Option 2: create a binary HDF5 EEOUT file

```plaintext
eMBED2 meshgeo=abaqus
  mgeoin=example.inp
  hdf5file=example.eeout.h5
  background= 20
  matcell= 1 10 2 11 3 12 4 13
```

Produce two output files:
- `example.eeout.h5` (binary HDF5 file)
- `example.eeout.h5.xdmf` (ASCII XML file)

• Letters in file names on EMBED card must be lowercase letters.
• MCNP overwrites an old HDF5 EEOUT file [ & XML file].

New feature in MCNP 6.3 version
Three Options to Produce EEOUT Files: Option III

- Option 3: create an ASCII EEOUT file & a binary HDF5 EEOUT file

```
embed2 meshgeo=abaqus
mgeoin=example.inp
meeout=example.eeout
hdf5file=example.eeout.h5
background= 20
matcell=  1 10  2 11  3 12  4 13
```

This option is not recommended for a large calculation.

Produce three output files:
- `example.eeout` (ASCII EEOUT file)
- `example.eeout.h5` (binary HDF5 file)
- `example.eeout.h5.xdmf` (ASCII XML file)

- Letters in file names on EMBED card must be lowercase letters.
- MCNP overwrites the old ASCII & HDF5 EEOUT files [ & XML file].
MCNP UM Postprocessing & Visualization

• 2021 MCNP User Symposium [July 14, 10:55-11:15]:

MCNP Unstructured Mesh Visualization & Post-processing Techniques


1Los Alamos National Laboratory, Monte Carlo Codes Group, Los Alamos, NM
2Los Alamos National Laboratory, Applied Computer Science Group, Los Alamos, NM
3Los Alamos National Laboratory, Radiation Transport Applications Group, Los Alamos, NM
4Los Alamos National Laboratory, Advanced Engineering Analysis Group, Los Alamos, NM
5Silver Fir Software, Gig Harbor, WA

Corresponding Author Email: jkulesza@lanl.gov

This talk describes several techniques for post processing MCNP6 ASCII unstructured mesh (UM) elemental-edit output (EEOUT) files as well as HDF5 EEOUT files expected to be present in the upcoming release of the MCNP code, version 6.3.
MCNP UM New Features in 6.3 version

• Mesh Quality Metric Tables
  – “MCNP UM Elemental Quality Assessment” Presentation by Joel Kulesza

• New Element Type: Abaqus SC8
  – pure SC8 in a part
  – mixed SC8 and C3D8 in a part
  – using same tracking algorithm for SC8 and C3D8 elements

• Convert an Abaqus input file to an HDF5 mesh input file
  – run MCNP input option \([\text{mcnp6 \ i \ inp} \text{ or \ mpirun –np \ <n> \ mcnp6.mpi \ i \ inp}]\) using “hdf5file” on EMBED card to create an HDF5 mesh input file

• HDF5 Mesh Input File
• HDF5 EEOUT File
  – restart using HDF5 EEOUT file
HDF5 UM Input/Output

• Why is an HDF5 file chosen?
  − Becomes I/O library of choice for NNSA Labs.
  − Designed to manage large complex data collections.
  − Portable among different computing platforms.
  − Easy to view, edit, and analyze using public available software tools or Python scripts.

• An HDF5 file is a container for an organized collection of objects where each object must have a unique identity within an HDF5 file and can be accessed only by its name within the hierarchy of the file.
  − HDF5 objects: attribute, dataset, group
  − HDF5 link: unstructured_mesh/cell_name

• See MCNP 6.3 User Manual (Chapter 8.10) for an HDF5 file format used to store an unstructured mesh model and element edit outputs.
<filename>.h5: /unstructured_mesh

<table>
<thead>
<tr>
<th>Name</th>
<th>HDF5 Object</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>model_description</td>
<td>attribute</td>
<td>string</td>
</tr>
<tr>
<td>total_cells</td>
<td>attribute</td>
<td>integer</td>
</tr>
<tr>
<td>total_elements</td>
<td>attribute</td>
<td>integer</td>
</tr>
<tr>
<td>total_parts</td>
<td>attribute</td>
<td>integer</td>
</tr>
<tr>
<td>total_part_elements</td>
<td>attribute</td>
<td>integer</td>
</tr>
<tr>
<td>cell_name</td>
<td>dataset</td>
<td>1D string array</td>
</tr>
<tr>
<td>&lt;unique_cell_name&gt;</td>
<td>group</td>
<td>Attributes &amp; groups</td>
</tr>
</tbody>
</table>

groups in each <unique_cell_name>:
- material, mesh, volume, source, edit
- source is an optional group
- edit is a group in an output file
Easy to process HDF5 UM Input/Output File using Python

```python
import h5py

def printall(name, obj):
    print(name, dict(obj.attrs))

name = "nestedcylinder_electrontestv3.eeout.h5"
with h5py.File(name, 'r') as hf:
    hf.visititems(printall)

import numpy as np
filename = "nestedcylinder_electrontestv3.eeout.h5"
cell_data, cell_name = get_mesh_data_HDF5(filename)
for cname in cell_name:
    volume = cell_data.get(cname)[1]
    volume = np.array(volume)
    indx = np.where(volume <= 1.E-6)[0]
    print("{:s}".format(cname))
    for i in indx:
        print("  {:20d} {:20.5e}".format(i, volume[i]))

import h5py

def get_mesh_data_HDF5(eeout_filename):
    f = h5py.File(eeout_filename, 'r')
    um = '/unstructured_mesh'
    cell_label = list(f[um+"/cell_name"])
    path_name = []
    cell_name = []
    for c in cell_label:
        name = c.decode('utf-8').strip()
        path_name.append(um + '/' + name)
        cell_name.append(name)

    cell_data = {}
    for pname, cname in zip(path_name, cell_name):
        k = pname + '/material/mass_density'
        density = list(f[k])
        k = pname + '/volume/element_volume'
        volume = list(f[k])
        cell_data[cname] = [density, volume]

    f.close()
    return cell_data, cell_name
```
Some MCNP UM Code Enhancements in 6.3 Version

• Improved UM Abaqus Input preprocessing
  – reduce memory, faster, & more robust

• Fixed poor code performance
  – significantly faster for large calculations

• Fixed codes so that all UM regression/feature testing problems can be run with the executable build with more restrictive flags:
  – Fortran_FLAGS="-check all, noshape, noarg_temp_created"

• Fixed neutral particle tracking bugs:
  – “collision in void in colidn routine”
  – “photon transport with all-zero photoatomic cross section”

• Other UM fixed bugs and code enhancements will be listed in the release notes of MCNP 6.3 version.

  do not build MCNP with this option for production calculations
MCNP UM Code Verification & Validation

• Motivation:
  − MCNP code V&V gives users confidence in its calculated results. Several MCNP V&V suites are distributed with MCNP code release.
  − Despite the MCNP UM feature being increasingly used for new applications, there is no UM V&V suite distributed with the code.

• Verification:
  − converted Oktavian testing problems in MCNP CSG VERIFICATION_SHLD_SVDM Suite into UM models and verify MCNP CSG & UM results.

• Validation:
  − converted Godiva sphere into 4 models [1\textsuperscript{st} order tet & hex; 2\textsuperscript{nd} order tet & hex] and validate the calculated keff values with experimental value.

MCNP UM Godiva & Oktavian test problems will be released with MCNP 6.3 version (15 MCNP and Abaqus input files).
MCNP UM Limitations

• Limited testing on the following features:
  − Non-void background cell
  − PTRAC
  − SDEF options
  − Average & Entry Overlap model
  − 2\textsuperscript{nd} order tet, pent, hex elements
  − Neutron/Photon/Electron, Photon/Electron, and charged particle transport calculations

• Known issues:
  − Incorrect results for mixed void and non-void pseudo cells
  − Negative energy depositions for electrons

• Should not use for magnetic fields

• No code implementation for forced collision on UM

• No testing on UM Utilities Program

• Surface tallies are not permitted in the background cell and pseudo cells

Currently, UM Capability is not fully integrated with all of the pre-existing MCNP features. See MCNP 6.3 User Manual for other UM Limitations.
Future Work

• Refactor codes to reduce memory & speed up the calculations [continuing work]
  - Replace inefficient data structure
  - Replace inefficient tracking algorithms
  - Replace algorithms used to calculate edits

• HDF5 parallel reading/writing input/output files

• Improvement for Photon/Electron transport calculations

• Remove UM Utilities from MCNP code [Fortran Code]
  - Develop Python scripts to replace UM Utilities
  - Remove MCNPUM format from MCNP code

• MCNP UM V&V
Questions?