Title: Fission Matrix Processing Using the MCNP6.3 HDF5 Restart File

Author(s): Kulesza, Joel A.
Josey, Colin James

Intended for: Reference

Issued: 2022-05-09 (rev.1)

Fission Matrix Processing Using the MCNP6.3 HDF5 Restart File

Joel A. Kulesza and Colin J. Josey

Monte Carlo Codes Group (XCP-3), Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, NM

jkulesza@lanl.gov & cjosey@lanl.gov

INTRODUCTION

This paper describes an approach to interrogating the fission matrix [1, 2] available in the HDF5-formatted [3] restart file (also known as the "runtape" file), which is produced by the upcoming public release of the MCNP® code, version 6.3. The fission matrix, its eigenvalues, and its eigenvectors, are important tools for characterizing fissile systems such as research and power reactors and for accelerating the convergence of the Monte Carlo fission-source site distribution [4, 5].

The MCNP6.3 restart file can be used to store the fission matrix after its calculation during an MCNP *k*-eigenvalue calculation. Because the restart file is HDF5-formatted in MCNP6.3, retrieving values from it is straightforward using a variety a programming languages and/or utilities. While ways of interacting with the HDF5-formatted restart file will be described in the documentation accompanying the upcoming release of the MCNP code, this paper is written to highlight the ability to interact with the fission matrix within the restart file via example.

As such, this paper provides both a brief review of how the fission matrix relates to a familiar form of the Boltzmann transport equation and an example Python processing script that can be used quantitatively and qualitatively understand the matrix. Finally, this paper gives results of using the script on the fission matrix resulting from a high-fidelity MCNP calculation of the Oak Ridge National Laboratory (ORNL) Pool Critical Assembly (PCA) [6, 7].

BRIEF REVIEW OF THEORY

References such as [4, 5] provide a full description of the fission matrix's development and application in the MCNP code. However, a brief review is given here to orient the reader to its origin and to indicate how it is useful. Accordingly, one can begin by writing the Boltzmann neutron transport equation in its *k*-eigenvalue time-independent form as

$$\mathcal{M}\psi(\boldsymbol{x},\boldsymbol{\Omega},E) = \frac{1}{k}\frac{\chi(E)}{4\pi}S(\boldsymbol{x}), \tag{1}$$

where $\mathcal{M}$ is the migration operator containing terms such as streaming, collision, and scattering, ψ is the angular neutron flux, $\boldsymbol{x}$ is position, E is energy, $\boldsymbol{\Omega}$ is the direction of particle travel, and χ is the emission energy spectrum. The function $S(\boldsymbol{x})$ is the fission neutron source,

$$S(\boldsymbol{x}) = \int \mathrm{d}E' \int \mathrm{d}\Omega' \nu\Sigma_{\mathrm{f}}(\boldsymbol{x},E')\psi(\boldsymbol{x},\boldsymbol{\Omega}',E'), \tag{2}$$

where $\nu\Sigma_{\mathrm{f}}$ is the product of neutron multiplicity and macroscopic fission cross section for a given position and energy.

By performing discretization and region-wise integration, one can express the fission source by the eigenvalue-eigenvector equation

$$k_n\boldsymbol{S}_n = \mathbf{F}\boldsymbol{S}_n\ ,\ n = 0, 1, \ldots, N, \tag{3}$$

where k_n is the nth eigenvalue (with $k_0 > |k_1| > |k_2| > \cdots$), $\boldsymbol{S}_n$ is the nth eigenvector, and $\mathbf{F}$ is the fission matrix. The resulting k_n and S_n approximate the terms k and $S(x)$ from Eq. (1), with k_0 and S_0 representing the fundamental. Higher modes can be used to assess characteristics such as system asymmetry. Physically, $\mathbf{F}$ represents the mapping of fission neutrons produced in a given spatial region as a result of a fission neutron starting in a given spatial region (either the same or different).

EXAMPLE PROCESSING SCRIPT

This section provides an example Python script that can be used to process the fission matrix found in an MCNP6.3 restart file. To demonstrate this script, the ORNL PCA is used. The ORNL PCA is characterized by a 5×5 array of fuel assemblies with about 18 curved fuel plates each that contain highly enriched uranium metal with three of the fuel assemblies interrupted by axial control rods and a fourth interrupted by a stainless-steel regulating rod. The core geometry is shown in Fig. 1.

An example script that can be used to process the fission matrix contained in the MCNP6.3 restart file to calculate eigenvalue and eigenvectors, and to produce 2-d slice plots of the eigenvectors, is given in Listing 2.

When the script given in Listing 2 is applied to the ORNL PCA using the input from [8] modified to add the lines shown in Listing 1, Fig. 2 is created. Note that no consideration is given to the effect of statistical noise on Fig. 2, which could be addressed by considering an ensemble of estimated fission matrices through additional independent sampling.

Listing 1. MCNP Input Modifications to Create Fission Matrix

```
kcode 250000 1.000 50 250
kopts fmat=yes
hsrc   40 -20.41    20.25
       40 -19.275   19.275
       75 -36.35375 36.35375
```

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.

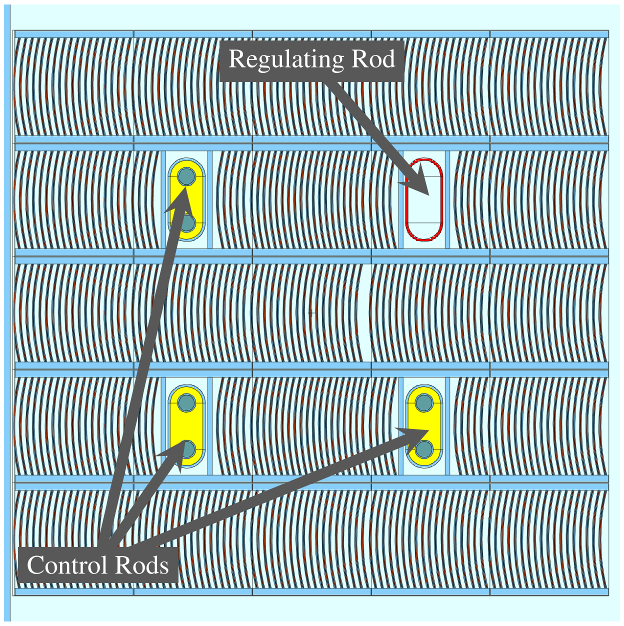

Figure 1. ORNL PCA Core Configuration

CONCLUSIONS & FUTURE WORK

This paper describes an approach that enables quantitative characterization and qualitative visualization of the fission matrix that can be calculated during an MCNP6.3 k-eigenvalue calculation. The approach is demonstrated by processing an easily described but non-trivial fissile system with a designed-in asymmetry. Future work includes extending this work to provide 2-d and 3-d interactive interrogation and visualization of the fission matrix.

ACKNOWLEDGEMENTS

The authors thank Michael E. Rising (LANL) for performing code review of work that relocated the fission matrix to be a component of the MCNP6.3 restart file `results` group. This work is supported by the Department of Energy National Nuclear Security Administration (NNSA) Advanced Simulation and Computing (ASC) Program.

REFERENCES

1. K. W. MORTON, "Criticality Calculations by Monte Carlo Methods," Tech. Rep. AERE-T/R-1903, Great Britain Atomic Energy Research Establishment, Harwell, Berks, England (1956).
2. E. L. KAPLAN, "Monte Carlo Methods for Equilibrium Solutions in Neutron Multiplication," Tech. Rep. UCRL-5275-T, Lawrence Radiation Laboratory, Berkeley, CA, USA (1958).
3. THE HDF GROUP, "Hierarchical Data Format, version 5," Website (1997–2020), last Accessed: Feb. 2020.
4. F. B. BROWN et al., "Fission Matrix Capability for MCNP, Part I - Theory," in "Proceedings of International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)," Sun Valley, ID, USA; May 5–9 (2013), Los Alamos National Laboratory Tech. Rep. LA-UR-13-20429.

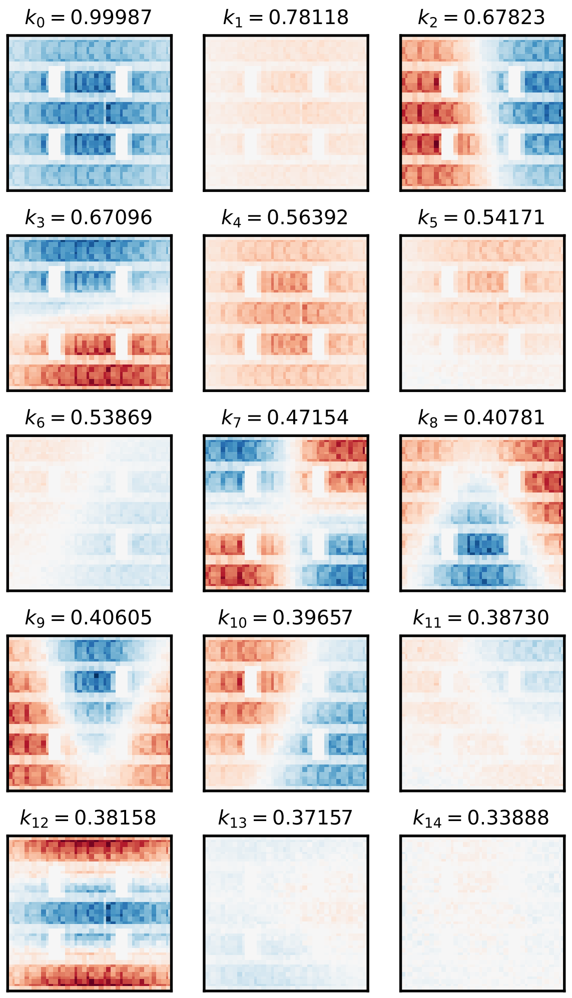

Figure 2. Estimated Eigenvalues & Eigenvectors (2-d Slices)

5. S. E. CARNEY et al., "Fission Matrix Capability for MCNP, Part II - Applications," in "Proceedings of International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)," Sun Valley, ID, USA; May 5–9 (2013), Los Alamos National Laboratory Tech. Rep. LA-UR-13-20454.
6. I. REMEC and F. B. K. KAM, "Pool Critical Assembly Pressure Vessel Facility Benchmark," Tech. Rep. NUREG/CR-6454, Oak Ridge National Laboratory, Oak Ridge, TN, USA (Jul. 1997).
7. J. A. KULESZA and R. L. MARTZ, "Evaluation of the Pool Critical Assembly Benchmark with Explicitly Modeled Geometry Using MCNP6," *Nuclear Technology*, **197**, *3*, 284–295 (Mar. 2017).
8. J. A. KULESZA, "Oak Ridge National Laboratory Pool Critical Assembly MCNP6 Criticality Calculation Input File," Tech. Rep. LA-UR-20-21532, Los Alamos National Laboratory, Los Alamos, NM, USA (Feb. 2020).

Listing 2. Example Fission Matrix Processing Script

```python
#!/usr/bin/env python3

import h5py
import matplotlib.pyplot as plt
import numpy as np
import scipy.sparse as sparse
import scipy.sparse.linalg as sla
from matplotlib.cm import get_cmap

SUPPORTED_RUNTAPE = [1, 0, 0]


def extract_fmat(runtape):
    """Returns the last saved fission matrix as a scipy.sparse.csr_matrix"""
    with h5py.File(runtape, "r") as handle:
        # Check runtape version
        version_file = handle["config_control"].attrs["version_file"]
        if any(SUPPORTED_RUNTAPE != version_file):
            print("Possibly incompatible runtape detected.")

        fmat = handle["results/fission_matrix"]

        n_dim = fmat["n"][()]
        indices = fmat["indices"][:]
        indptr = fmat["indptr"][:]
        data = fmat["data"][:]

        n_xyz = fmat["n_xyz"][:]
        delta_xyz = fmat["delta_xyz"][:]
        origin = fmat["origin"][:]

    return (
        sparse.csr_matrix((data, indices, indptr), shape=(n_dim, n_dim)),
        n_xyz,
        delta_xyz,
        origin,
    )


def plot_eigs(mat, n_xyz, n_tot=6, n_col=2):
    """Retrieve eigenvalues/vectors, sort, reshape to 3-d object, and plot."""
    eigenvalues, eigenvectors = sla.eigs(mat, k=n_tot)

    # Clean up and sort the eigenvectors.
    sorted_eigvals = []
    sorted_eigvecs = []
    for i in np.argsort(-np.abs(eigenvalues)):
        val = eigenvalues[i]
        val = np.real(val) if np.real(val) == val else val
        sorted_eigvals.append(val)
        vec = np.real(eigenvectors[:, i].reshape(n_xyz[::-1]).transpose())
        if len(sorted_eigvals) == 1:
            vec = np.abs(vec)
        sorted_eigvecs.append(vec)

    # Create example plot grid.
    cmap = get_cmap("RdBu")
    fig, ax = plt.subplots(int(n_tot / n_col), n_col, figsize=(3, 3 * 1.75))
    aspect_ratio_z = delta_xyz[1] / delta_xyz[0]
    for i in range(int(n_tot / n_col)):
        for j in range(n_col):
            k = i * (n_col) + j
            data = sorted_eigvecs[k]
            cbar_scaling = max(np.max(data), -np.max(-data))
            ax[i, j].matshow(
                data[:, :, int(n_xyz[2] / 2)].transpose(),
                cmap=cmap,
                vmin=-cbar_scaling,
                vmax=cbar_scaling,
                origin="lower",
                aspect=aspect_ratio_z,
            )
            ax[i, j].set_xticks([])
            ax[i, j].set_yticks([])
            ax[i, j].set_title(
                f"$k_{{{k}}}={sorted_eigvals[k]:.5f}$", y=1.0, pad=3, fontsize=6,
            )

    plt.savefig("eigenvalues.pdf", bbox_inches="tight")


mat, n_xyz, delta_xyz, origin = extract_fmat("pca_kcode.mcnp.inp.txt.rtp.h5")
plot_eigs(mat, n_xyz, 15, 3)
```