LA-UR-13-26449
Approved for public release; distribution is unlimited.

Title: Eigenfunction Decomposition of Reactor Perturbations and Transitions Using MCNP Monte Carlo

Author(s): Josey, Colin
Veit, Max D.

Intended for: MCNP documentation
Report
Web

Issued: 2013-08-15

Eigenfunction Decomposition of Reactor Perturbations and Transitions Using MCNP Monte Carlo

Colin Josey, Max Veit

Computational Physics Workshop
Los Alamos National Laboratory

August 14, 2013

Abstract

Monte Carlo criticality calculations for nuclear reactors have typically only been able to find the primary eigenvalue and eigenmode. With the new fission matrix capability in MCNP6, it becomes possible to solve for higher modes, which are useful in wide variety of analyses. As a test case of this new capability, two reactor models in both a base configuration and a perturbed configuration were simulated in MCNP to generate fission matrices. Tools were written to find a relatively small number of eigenpairs of the resulting matrices, whereupon it was found that the implicitly restarted Arnoldi method outperformed the previously used power iteration as an eigenvalue algorithm by a factor of 1500 for a 3600×3600 sparse matrix. The primary eigenvalue was then compared to the default MCNP result and, although the values showed bias with regards to mesh size, the matrix-derived values had superior statistics. With the eigenvalue verification complete, the primary eigenmode of the base case was then projected onto the perturbed core's eigenspace, where transition coefficients, simplified quasistatic transitions, and projection errors were calculated. The projection error typically dropped off after the first 20 eigenvalues to a value that was stable through the first 100.

1 Introduction

Criticality calculations using the MCNP Monte Carlo code determine the fundamental mode eigenvalue (k_{eff}) and eigenfunction (fission distribution) of a fissile system. These calculations are routinely used in the design and analysis of critical experiments, nuclear reactor cores, and criticality safety applications. Recently developed MCNP capabilities for the fission matrix method permit the calculation of higher-mode eigenvalues and eigenfunctions. With knowledge of the higher modes, transitions from the base state of a system to perturbed states can be analyzed. The state transition parameters characterize changes to the system induced by material substitutions, geometry changes, and feedback, and are important for analyzing potential instabilities.

The fission matrix is essentially a spatially discretized Green's function for the neutron transport equation. It is a matrix $\bar{F}$ whose entries, $\bar{F}_{ij}$, contain the expected number of next generation fission neutrons generated in spatial mesh region i by a neutron born in mesh region j. The eigenvalues and eigenvectors of this matrix will approximate the eigenvalues and eigenvectors of the reactor. These approximations are mesh size dependent, but the size of the matrix goes as the square of the number of cells. For even moderate mesh sizes, the resulting fission matrix will be massive. Thankfully, it is also rather sparse. In this report the methods of solving for the eigenvalues and eigenvectors of a very large sparse matrix are detailed. Tools that analyze the output of the fission matrix from MCNP were implemented in C++, MATLAB and Python.

Demonstrations of the capabilities of the fission matrix were performed on two reactor models, the Advanced Test Reactor, and a 2D PWR model. These models are frequently used in the testing of MCNP. These models were then perturbed and the resulting eigenpairs were analyzed.

Lastly, as it is very difficult to store all of the data for the fission matrix itself, additionally storing the squares of the tallies that generated it to calculate statistics is not implemented as of yet. By using a moderate quantity of MCNP runs with varying starting seeds, the statistics can roughly be approximated. The same two models used for testing MCNP as before were used here as well.

2 Theory

First, the theory and physics underlying the concept of fission matrices is summarized, and the linear algebra behind transition coefficients investigated. The predominant algorithms used for finding eigenpairs of large, sparse, asymmetric matrices are also listed and briefly discussed.

2.1 Fission Matrices

The utility of a fission matrix is rooted in the neutron transport equation. Through no approximation other than a simple spatial discretization, the neutron transport equation

$$M\Psi(\mathbf{r}, E, \hat{\Omega}) = \frac{1}{k}\frac{\chi(E)}{4\pi}S(\mathbf{r}) \tag{2.1}$$

can be integrated over space and energy into the form

$$\mathbf{s} = \frac{1}{k}\bar{F}\mathbf{s}, \tag{2.2}$$

where $\mathbf{s}$ is the source distribution vector and $\bar{F}$ is the fission matrix as defined before [1]. As shown, it is clear that this is an eigenvalue problem, with k as the eigenvalue and $\mathbf{s}$ as the eigenfunction. As all components are matrices, vectors, or scalars, this equation lends itself well to a linear algebra solution.

The solutions of this eigenvalue problem are especially useful for a specific type of analysis, namely, computing transition coefficients for reactor perturbations. First, assume the reactor configuration instantaneously changes, through e.g. a geometry or a material property change. In this situation, one would like to know how the reactor's initial fission source distribution can be represented in terms of eigenmodes of the fission matrix of the new configuration. In particular, one would like to be able to express any of the initial configuration's eigenmodes in terms of this new partial basis.

Let $\{\mathbf{u}_i\}$ be the set of eigenmodes of the fission matrix of the initial configuration, and $\{\mathbf{v}_i\}$ be the set of eigenmodes in the final configuration. Assuming the fission matrix for the final state has at least m linearly independent eigenmodes $\{\mathbf{v}_i\}_{i=1}^{m}$, then any source distribution $\mathbf{s}$ can be approximated by its projection onto the space spanned by these m modes. This approximation can be considered optimal (in

the sense that it minimizes the norm of the residual) if the set of right eigenmodes is orthogonal.

Applying the above expansion to the i-th eigenmode of the initial fission matrix, $\mathbf{u}_i$, gives:

$$\mathbf{u}_i \approx \mathbf{u}_i^m = \sum_{j=1}^{m-1} C_{ij}\mathbf{v}_j. \tag{2.3}$$

The j-th expansion coefficient C_{ij} can be extracted by exploiting the fact that the system of forward and adjoint modes $\left\{\left(\mathbf{s}_k, \mathbf{s}_k^\dagger\right)\right\}_{k=0}^d$ of a diagonalizable matrix forms a biorthogonal system; i.e., with the appropriate normalization,

$$\left\langle \mathbf{s}_l, \mathbf{s}_m^\dagger \right\rangle = \delta_{lm}. \tag{2.4}$$

The notation $\langle \mathbf{a}, \mathbf{b} \rangle$ denotes the inner product on $\mathbb{C}^n$, which for the purposes of this report is defined as $\mathbf{b}^*\mathbf{a}$. This product is linear in the first argument and conjugate-linear in the second, a property called 'sesquilinearity'.

The adjoint modes are the left eigenvectors of the fission matrix. In practice, they can be computed by taking the eigenvectors of its transpose. In the general case, the resulting vectors are the complex conjugates of the adjoint modes. This is usually not a problem, since it is usually assumed that the eigenmodes of a fission matrix will be real. Although this fact has not been theoretically proven, numerical evidence bears it out within the limits of statstical variation [1].

Relation (2.4) holds, albeit in a restricted sense, even if a complete system of eigenvectors does not exist for a given fission matrix; see Appendix A. Using this fact, one can write, using the linearity of the dot product in its first argument,

$$C_{ij} = \left\langle \mathbf{u}_i, \mathbf{v}_j^\dagger \right\rangle = \sum_{k=1}^{N} C_{ik} \left\langle \mathbf{v}_k, \mathbf{v}_j^\dagger \right\rangle = \sum_{k=1}^{N} C_{ik}\delta_{kj}, \tag{2.5}$$

assuming the final eigenmodes are normalized such that Relation (2.4) holds.

The approximation $\mathbf{u}_i^m$ can then be constructed using the transition coefficients $\left\{C_{ij}\right\}_{j=1}^m$ and the eigenvectors $\left\{\mathbf{v}_j\right\}_{j=1}^m$.

Once these transition coefficients are known, a quasistatic model of the transition from the base to the perturbed state is:

$$\mathbf{s}_{\text{current}} = \sum_{j=1}^{N} C_{1,j} \left(\frac{k_j}{k_1} \right)^n \mathbf{v}_j, \tag{2.6}$$

where n is the current neutron generation and N is the total number of eigenvalues.

2.2 Eigenvalue Solvers

The fission matrix $\bar{F}$ is nonsymmetric and tends to be a large, sparse matrix. As such, it is very difficult to store without using a sparse storage scheme. This severely constrains the choice of eligible algorithms. For example, the implicit QR algorithm, the eigenpair solver of choice for dense matrices, requires a transformation into upper Hessenberg form [2]. Such transformations tend to cause fill-in of the formerly empty elements, causing what was once a sparse matrix to occupy more than half of the space of a full version of the matrix. Further, the resulting matrix of eigenvectors will be nearly full. Most computers cannot handle matrices so large.

An alternative comes in the form of Krylov subspace solvers. The primary advantage of such solvers is the iterative mode of calculation that performs only matrix-vector math with the sparse fission matrix, so the problem of fill-in is avoided. The most simple Krylov subspace solver is the power iteration method. This simple algorithm is robust and effective, but has two major flaws. For one, to get more than the first eigenpair requires some sort of deflation, such as Hotelling deflation [3, pp. 85-96]. Secondly, the convergence of the power method is linear and governed by the factor $|\lambda_1/\lambda_2|$, also known as the dominance ratio [4]. When this ratio is close to one, the convergence of this method can be extremely slow.

More advanced Krylov subspace solvers such as implicitly restarted Arnoldi method (IRAM) address both issues. IRAM can solve for more than one eigenpair at a time and it has (usually) superlinear convergence [5]. All math done with regards to the fission matrix is in matrix-vector form. Two matrices are stored separately from the fission matrix. Matrix V has columns of the same length as the fission matrix, but a number of rows equal to m, a variable that is between 2-3x as large as the total number of eigenpairs needed. The columns

are, once sorted, the resulting eigenvectors of the problem. Matrix H is an upper Hessenberg matrix of size m by m whose eigenvalues are the eigenvalues of the fission matrix [3, pp. 128-136].

3 Methodology

Several steps were involved in the investigation of the fission matrix capability. First, in order to study transitions and transition coefficients, reactor models were needed in both an initial and final state. Then, a tool was needed to find the eigenpairs of both systems efficiently. These tools were finally expanded to perform numerous data manipulations and plot the results.

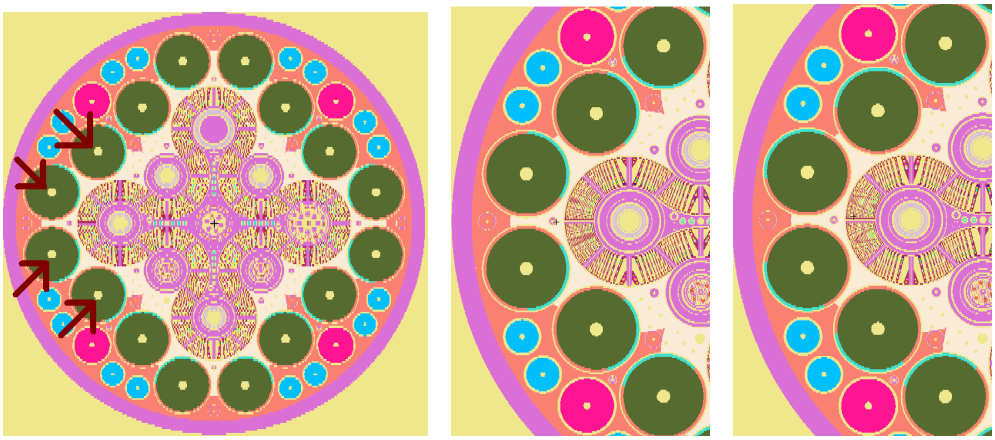
3.1 Reactor Model Modification

A main goal of the project was to test the fission matrix capability on a reasonably complex analysis problem. The problem of finding transition coefficients between two reactor models was chosen for this purpose. This analysis requires perturbed versions of base-case reactor models, as well as the base models themselves. If the two models were too different, the transition coefficients would be essentially meaningless, so the alterations were done in such a way to significantly alter the flux while at the same time not significantly altering the geometry. For the ATR case, four of the control drums S3, S4, W1, and W2 were rotated 50°, moving beryllium closer and hafnium away from the core. The core model, before and after, is shown in Figure 1.

For the 2-D PWR core, control rods consisting of type 304 stainless steel were inserted in each assembly in the upper right quadrant of the core. A comparison is also shown in Figure 2. Note that the region chosen is asymmetric in that it does not represent a true rotationally symmetric quarter of the core; this fact has significant consequences for the eigenmodes of the perturbed case as well as the transition coefficients in between the states.

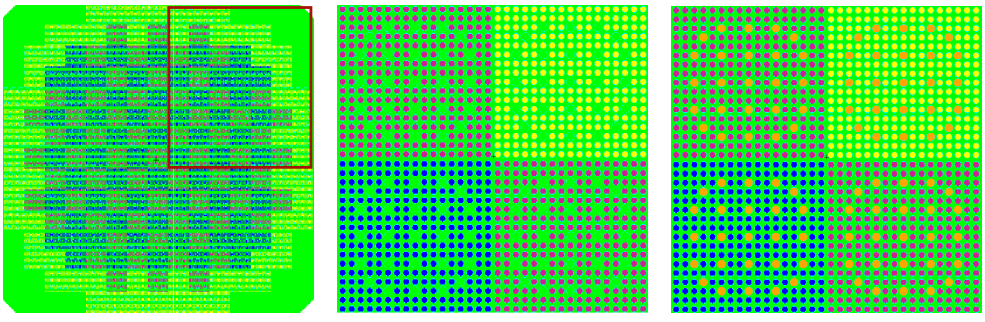
3.2 Fission Matrix Generation

Currently, as of August 2013, published versions of MCNP6 do not contain the full fission matrix capability. The commands described



(a) ATR Full Core (b) ATR Original (c) ATR Modified

Figure 1: ATR Core Modifications



(a) PWR Full Core (b) PWR Original (c) PWR Modified

Figure 2: PWR Core Modifications

below will work, but no fission matrix will be output. Further, as time goes on, the method of generating fission matrices as well as their output format will likely change as they are not precisely user friendly at the moment. This part of the report will detail how current versions of MCNP6 internal to LANL operate with regards to fission matrices.

In order to enable fission matrices, two cards must be present in the input deck. First, the mesh extent from the `HSRC` card is used as the extent for the fission matrix. As such, the `HSRC` card must be included. The number of cells along each axis will not be used, however, as the fission matrix usually needs to be finer than the Shannon entropy mesh. The second is the `KOPTS` card. The `KOPTS` card defines the options for `KCODE`, and contains the settings for the fission matrix. `KOPTS` has the following form:

```
KOPTS   ...several-unrelated-options...
        fmat= (yes/no)
        fmataccel= (yes/no)
        fmatskip= n
        fmatncyc= n
        fmatnbr=  n
        fmatnbrx= n
        fmatnbry= n
        fmatnbrz= n
```

The meaning of these options is summarized in Table 1.

As long as both commands are present, MCNP will keep the fission matrix tallies internally. If `fmataccel = yes`, the resulting primary eigenfunction from solving the fission matrix will be used to split or roulette `KCODE` source neutrons in each cell to closer match the source distribution. This has some advantages, as the fission matrix primary eigenmode will be more accurate than the initial guess or any unaltered `KCODE` results prior to convergence. Current versions of MCNP do not save the fission matrix, but in-development versions do. The resulting file is `fmat_file`. The structure of this file as currrently implemented is discussed in the following sections.

3.3 Eigenvalue Tool Exploration

By default, the current development versions of MCNP6 output the fission matrix as an unformatted binary file as written by a Fortran-

command	Description
fmat= yes	Enables fission matrix
fmataccel= yes	Enables using the fission matrix primary eigenmode to accelerate the convergence of KCODE
fmatskip= n	Skips n cycles before tallying fission matrix
fmatncyc= n	Solves for k_{eff} and dominance ratio every n cycles
fmatnbr= n	Total number of entries available for the sparse matrix.
fmatnbrx= n	Sets number of cells in the x-axis to n
fmatnbry= n	Sets number of cells in the y-axis to n
fmatnbrz= n	Sets number of cells in the z-axis to n

Table 1: KOPTS Options

based code. These files are not human-readable, and their structure is machine and compiler dependent. A Fortran-based tool already existed that would read the fission matrix, perform power iteration, and output several text and PostScript files containing results and plots. This worked well on smaller fission matrices on the order of $N = 10000$ rows, but beyond that, the calculation time was unreasonably high. Two different approaches (with a third later) were undertaken to expedite the calculation process. The first approach used Python interfaced with ARPACK [6], the second was a custom C++ implementation of Arnoldi iteration, and the third used MATLAB, again interfaced with ARPACK.

Each toolset had in common the same general steps necessary to perform the analysis: First, it read in the initial and final (perturbed) fission matrices from their native binary formats. Second, the tool normalized each matrix's rows against the corresponding fission source tally, since the matrices were saved in the form of raw tallies. Third, it extracted the forward and adjoint eigenvalues of the resulting sparse matrix. Finally, it obtained transition coefficients by taking dot products between the initial forward and final adjoint eigenmodes. These results were then visualized and interpreted.

3.3.1 Python

One approach to extracting the eigenmodes from the fission matrices used an interactive Python analysis system[1] equipped with several useful libraries. The system used the NumPy package for numerical computing, as well as the SciPy package's sparse-matrix capabilities, which included an interface to the freely available ARPACK sparse-matrix eigenvalue solver [6]. Plots and visualizations were generated using the Matplotlib visualization package.

The system proved effective at reading in the generated fission matrices in Fortran unformatted-binary sparse storage format and finding a small (80 or fewer) number of eigenvectors and -values. As an illustrative example, the eigensolver routine took 163 seconds to solve for the 80 forward modes of the unperturbed case with $N = 57600$; the adjoint case usually took longer (212 seconds in this case)

[1]Technical details: Enthought Python Distribution 7.3-2 (64-bit) (`https://www.enthought.com/`) with Python 2.7.3, SciPy 0.10.1, NumPy 1.6.1, Matplotlib 1.1.0, and IPython 0.12.1 for interactive use.

because of the additional cost of multiplying by the transposed fission matrix; see Section 4.1.

The prime disadvantage of the Python system is that it did not have parallel processing capability in the standard configuration. A parallel implementation of ARPACK is available [6]; however, the SciPy package apparently only links to the serial version. Moreover, the sparse matrix-vector products themselves must be performed by SciPy; this procedure is apparently also implemented serially in standard configuration.

3.3.2 C++

Initially, the C++ code, called `eigtest`, was simply a matrix math library that would implement just enough matrix math to do power iteration so that the person writing it could relearn the programming language. After finding out about the massive speedup possible converting from power iteration to the implicitly restarted Arnoldi method, the tool was converted to mimic a reference implementation written in MATLAB [7]. Portions of the original F90 code were merged in to provide for reading the fission matrix files. After a few weeks, however, the implementation was not converging correctly, and due to limited time, it was scrapped. The ability for the tool to read and write the matrix in a few different formats proved useful later on in the MATLAB implementation.

3.3.3 MATLAB

Although it may seem a bit strange to implement this code in both Python and MATLAB, during the C++ implementation, a large amount of MATLAB backend was written before the Python implementation was complete. In general, the MATLAB capability is very similar to the Python one and was used for the core step-by-step transition calculations and the statistics[2].

3.3.4 Fortran ARPACK Interface

Near the end of the project, another interface was developed for ARPACK, written in Fortran 90 and using parallel sparse-matrix vector

[2]Technical details: Matlab version R2013a

products. By that time, however, most of the work requiring eigenvalue solvers was completed, so it was not extensively tested.

3.4 Verification

It is not clear to what extent the algorithms and implementation of ARPACK have been verified. Therefore, two simple independent checks were undertaken to verify that ARPACK was indeed returning results with the advertised properties.

The nature of the eigenvalue problem admits a straightforward method of verification: Given a computed solution $(\mathbf{v}_C, \lambda_C)$ to the problem $A\mathbf{v} = \lambda\mathbf{v}$, the residual $r = A\mathbf{v}_C - \lambda_C\mathbf{v}_C$ can be computed, and its properties (e.g. norm) investigated to assess whether the computed answer solves the problem to the desired tolerance. A residual was considered acceptable if its l_2 norm did not considerably exceed the number of mesh cells used times the machine epsilon. This was indeed found to be the case for a set of 80 eigenvalues extracted both from the PWR and the ATR cases. We suspect this check is already done internally in ARPACK, but since no concrete evidence of such a check could be found, an independent verification was seen as justified. This independent check also guarded against any possible failure conditions in ARPACK that would not have been reported back to the user.

Another important criterion on the set of eigenvectors returned by ARPACK is that they be linearly independent; this property can be checked using a singular value decomposition of the matrix of eigenvectors. For both the PWR and the ATR test cases, it was indeed the case that all 80 singular values of the matrix of eigenvectors were clearly nonzero, i.e. many orders of magnitude larger than machine precision.

4 Results

The two different test problems, PWR and ATR, were run in MCNP with different parameters. A summary of the parameters used is in Table 2. These two runs generated fission matrices that were 4.8 GB and 32 MB in size for the PWR and ATR respectively. Both runs had enough cycles discarded to be converged for `KCODE`. The choice of fission matrix mesh size depends on the reactor being studied as well

Reactor	2D PWR	ATR
Cycles	300	500
`KCODE` Active Cycles	200	400
`fmatskip`	3	3
`fmatnbrx`	480	100
`fmatnbry`	480	100
`fmatnbrz`	1	1
Neutrons/cycle	4 million	100 thousand

Table 2: MCNP6 run details

Matrix Size	IRAM (ARPACK)	Power Iteration
3600×3600	3.09 s	4878 s
900×900	0.234 s	353 s
225×225	0.0337 s	30.6 s

Table 3: IRAM vs. Power Iteration for Various Matrix Sizes

as computational resources. For example, the PWR has low connectivity between distal regions of the core, owing to its great size. This is reflected in the fission matrix as a very high sparsity and thus, low memory usage. The opposite case is true for the ATR. As such, the ATR was typically run with a smaller mesh. A smaller mesh also requires fewer neutrons for statistical reasons. However, it is always beneficial to calculate with as large a mesh as possible, because the fission matrix can be aggregated into a smaller one as needed.

In practice, the fission matrix used for the PWR was aggregated by a factor of two, reducing it in size from 230400 to 57600 rows. This made the eigenvalue solve times much more tractable and improved the statistical properties of the resulting eigenmodes.

4.1 Solver Timings

A quick speed comparison was done between the two algoritms available. The runs were done on 1 CPU, and the first 16 eigenmodes and 16 adjoint eigenmodes were solved for. These runs are summarized in Table 3. As shown, a speedup of approximately 1500-fold was ex-

Number of	Unperturbed		Perturbed	
Modes	Forward	Adjoint	Forward	Adjoint
20	95 s	110 s	120 s	149 s
40	129 s	170 s	105 s	143 s
80	163 s	212 s	148 s	196 s

Table 4: Timings of the IRAM eigenvalue algorithm, as implemented by SciPy (see Section 3.3.1), on the PWR problem with $N = 57600$ rows.

perienced in switching algorithms on the 3600×3600 matrix.

Additionally, a preliminary investigation was done to investigate how the time taken by Arnoldi iteration scales with mesh size and the number of modes requested. The results are summarized in Table 4, from the PWR problem with $N = 57600$ rows. The times were computed both for the forward and adjoint mode solves. The given times are best out of 3 runs on a fairly capable multi-user machine. An important caveat on these timings is that a surprisingly large amount of variation was observed in the eigenvalue-solve timings, particularly on a spatial mesh with $N = 230400$.

Since not enough time was available to study the statistical properties of these timings in more detail, they are presented here only in order to illustrate some interesting general properties of the algorithm's runtime. First, note that the algorithm took consistently more time solving the adjoint problem than the forward problem; this is likely due to the overhead involved in transposing a CSR-stored sparse matrix, or computing matrix-vector products with its transpose. Second, the scaling as a function of number of eigenmodes requested is much weaker than one would expect given the known complexities of the individual components of the algorithm. This fact hints at the nontrivial convergence properties of the implicitly restarted Arnoldi method, as does a surprising result where it took longer to solve for 40 modes than for 80 modes of the matrix of the PWR problem with $N = 230400$ rows (1130 seconds versus 793 seconds, best of 6 runs each).

4.2 Eigenmodes

Using the techniques described earlier, the eigenpairs of the resulting matrices were solved for and plotted. Figure 3 shows the first 4 eigenmodes and adjoint eigenmodes of the initial and final configuration of the core. The same was done for the ATR in Figure 4. Comparing the original to the modified for the PWR, it becomes clear that the insertion of the control rods has significantly depressed a quadrant of the fundamental mode. The slight asymmetry in the perturbation also becomes noticeable in the higher modes. As for the ATR, the lobe that is surrounded by the moved control barrels is more active than the rest of the core. This is essentially as expected.

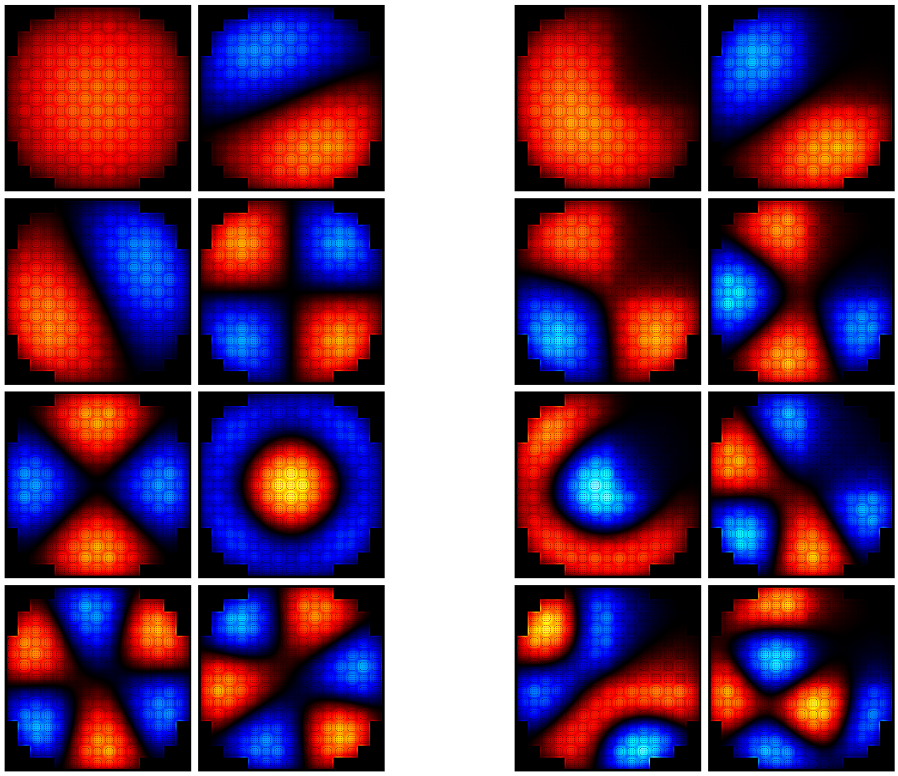
4.3 Transition Coefficients

Using a set of 40 eigenmodes calculated from each fission matrix, the transition coefficients were calculated and mapped into a grid. The PWR and ATR transition coefficients are plotted in Figure 5. The PWR transition coefficients give hints as to the more intricate transition occuring. The insertion of control rods in an asymmetric region of the core is not nearly as uniform an alteration as moving four adjacent control barrels by the same amount, and has a very strong spatial dependence.

4.4 Transitions

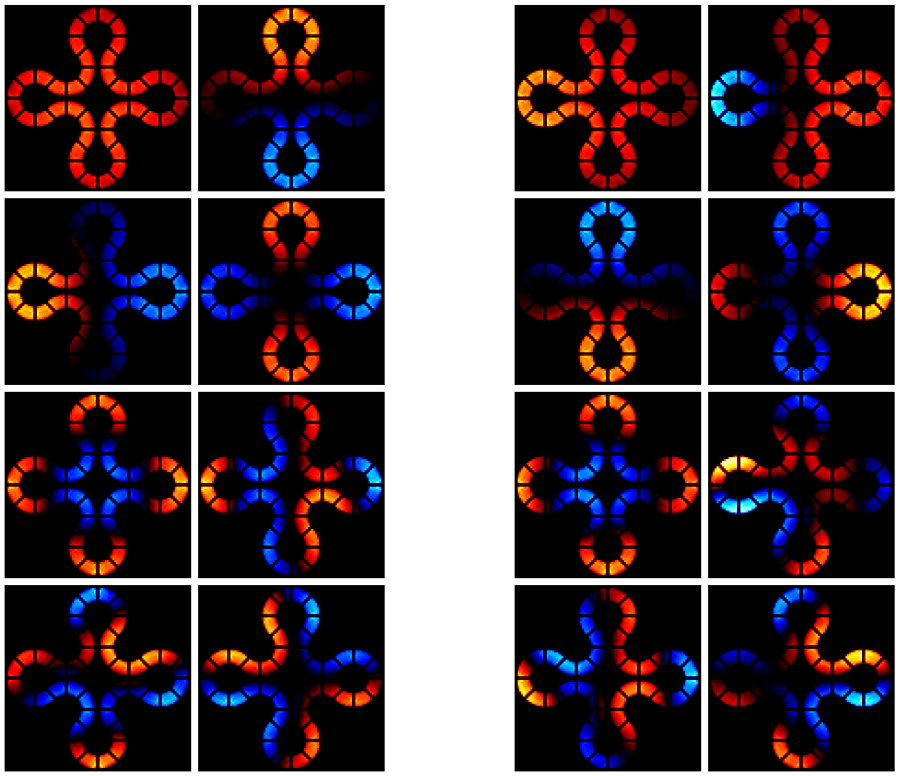
For the ATR, these coefficients were used to reconstruct a very basic stepwise transition. This is shown in Figure 6, along with what should be the original and final source distributions. Very slight differences are evident between the original and the generation 0 result, hinting that the reconstruction is not exact due to the limited number of eigenmodes used. Had the entire set of eigenvalues been calculated, this reconstruction would not be as imprecise.

The transition is found to be relatively smooth with the majority of sources propagating into the lobe of the core with the rotated control drums. The transition is mostly completed after 15 neutron generations, which, ignoring delayed neutrons and other reactions such as temperature dependence that will alter the transition, is a very short time. These plots can be seen as an approximation of the prompt jump at the beginning of a reactor configuration change.



(a) PWR Original (b) PWR Modified

Figure 3: Eigenvectors of the 2D PWR



(a) ATR Original (b) ATR Modified

Figure 4: Eigenvectors of the ATR

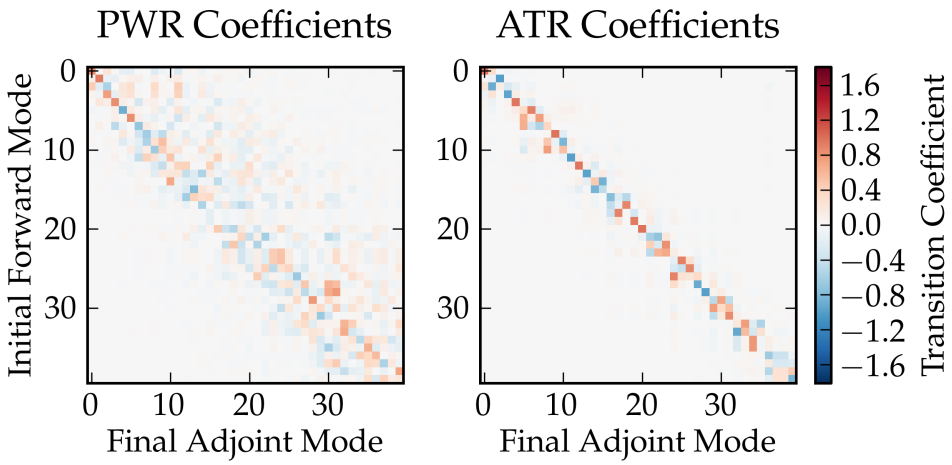


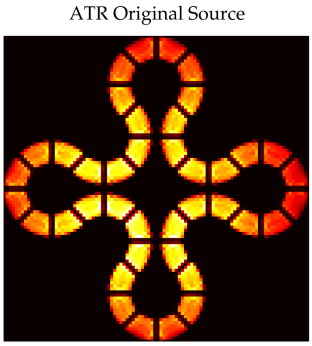
Figure 5: Transition Coefficients

4.5 Reconstruction Error

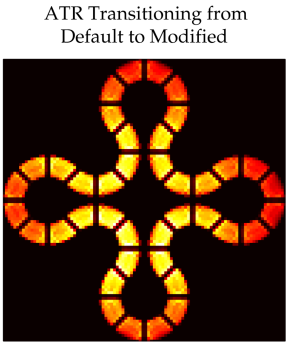
The error between the fundamental mode and its reconstruction in the space spanned by the perturbed eigenmodes was also studied. The initial fundamental mode was reconstructed with various quantities of eigenmodes in the final configuration and the l_2 norm of the difference between the two was calculated. The differences for both reactors with 40 eigenmodes are plotted in Figure 7, with the eigenmode count dependence plotted in Figure 8. It is clear that the more symmetric perturbation in the ATR core has made the transition coefficients rather simple. It takes relatively few transition coefficients to properly reconstruct the initial modes. The rather complex perturbation done to the PWR causes a more spread-out set of transition coefficients, indicating the larger number of perturbed modes necessary to reconstruct any given initial mode.

4.6 Statistics

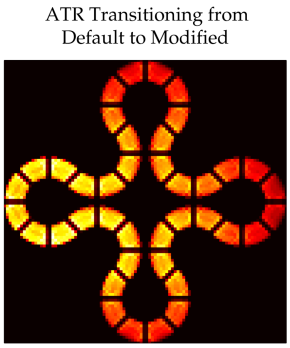
One primary drawback to the fission matrix method is that it is already quite difficult to store even rather small fission matrices. As



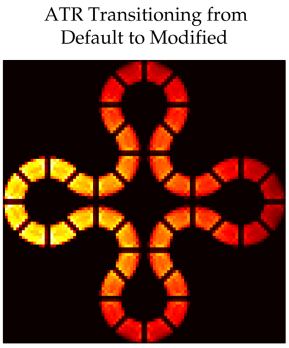
(a) ATR Original



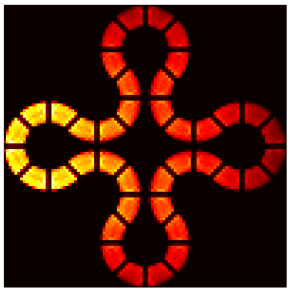
(b) ATR Generation 0



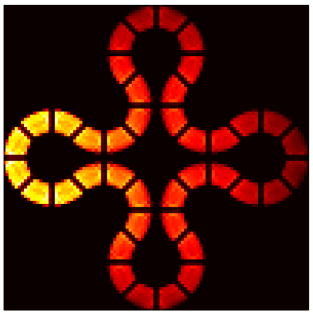
(c) ATR Generation 5



(d) ATR Generation 10

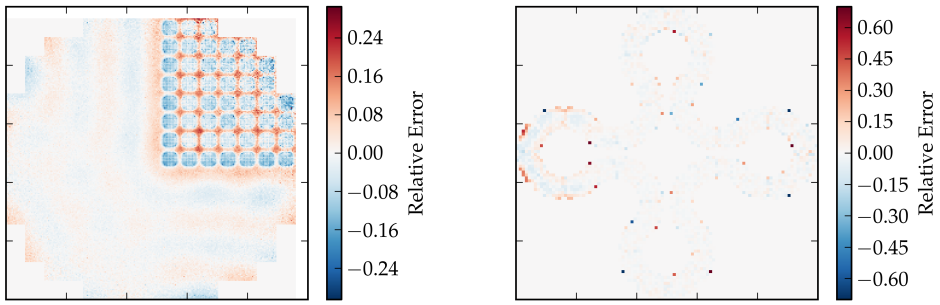


(e) ATR Generation 15



(f) ATR Final

Figure 6: ATR Transition from Start to Final



(a) PWR (b) ATR

Figure 7: Reconstruction Error

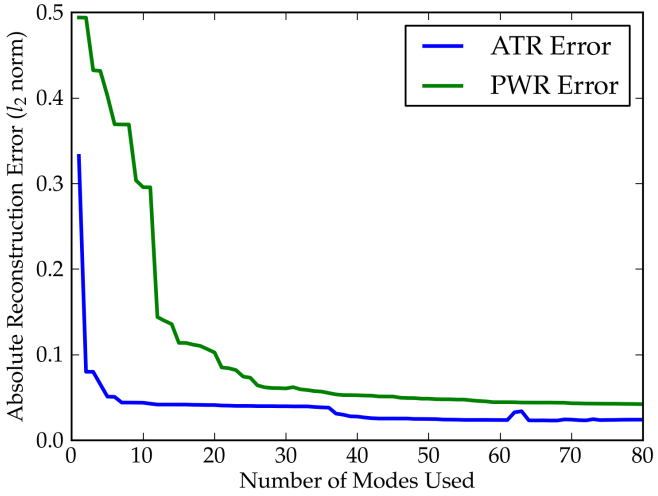


Figure 8: Reconstruction Error by Eigenmode Count

	$\bar{k}_1$	$\sigma_{\bar{k}_1}$	$\bar{k}_2$	$\sigma_{\bar{k}_2}$
KCODE	0.995077	0.000023	N/A	N/A
$50 \times 50 \times 50$ mesh	0.995017	0.000021	0.900928	0.000033
$25 \times 25 \times 25$ mesh	0.995011	0.000021	0.898198	0.000033
$10 \times 10 \times 10$ mesh	0.994977	0.000021	0.879747	0.000036
$5 \times 5 \times 5$ mesh	0.994924	0.000021	0.831998	0.000042

Table 5: Statistics of Runs

such, storing the squares of the tallies necessary to do error propagation in MCNP is not implemented. As such, the only way to measure statistics is to repeatedly run the same problem over and over again with different starting seeds until there are enough datapoints to calculate the variance of the sample.

To see if resolution has any impact on statistics, the ATR run was modified to use a $50 \times 50 \times 50$ mesh for the fission matrix, generating a 125000×125000 fission matrix. 25 runs were done and the mesh was aggregated into smaller meshes of size $25 \times 25 \times 25$, $10 \times 10 \times 10$, and $5 \times 5 \times 5$. The average of the resulting values along with the standard deviation of those values is summarized in Table 5.

It is worth noting that the statistics do get worse with decreasing mesh size, but, for example, the $5 \times 5 \times 5$ k_2 is not equal to the $50 \times 50 \times 50$ k_2, even within statistical variation. It appears that there is a significant bias effect for the non-fundamental modes at play with regards to mesh size. The same can be said of the $50 \times 50 \times 50$ k_1 value as compared to the KCODE k_1, but not to the same extreme. σ_{k_1} for all fission matrix runs was smaller than KCODE's, likely owing to the greater number of cycles contributing to the final result.

Another interesting test was to find the statistics on the first 100 eigenvalues for the $50 \times 50 \times 50$ run. This turned up a surprising result shown in Figure 9. Further exploration showed that mode 27 from run 20 had some regions inside of it with abnormally large values, as shown in Figure 10. The average mode is the one with run 20 removed but the other 24 combined. Removing all values above a certain threshold, the run 20 mode 27 is once again similar to the average, as shown in Figure 11, implying that the small number of very large values that caused the shift are likely statistical noise. Further, removing mode 20 completely from all calculations yields the statistics shown in Figure 12.

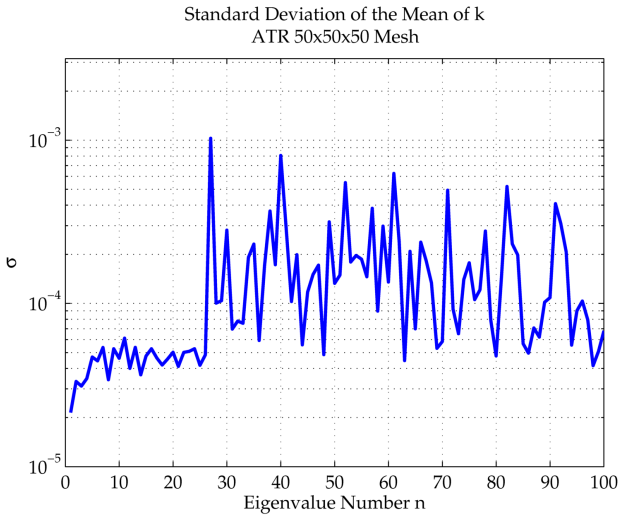


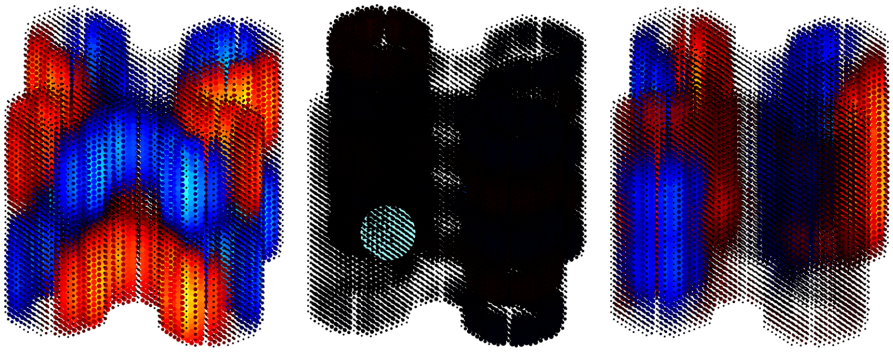
Figure 9: Standard Deviation by Eigenmode

5 Future Work

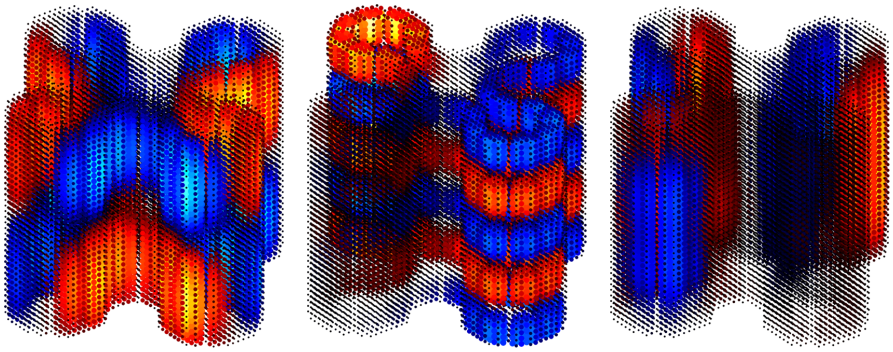
During the course of this work, many avenues for improvement and future investigation were uncovered. Many of these avenues were not pursued due to time constraints, so they are presented here as recommendations for continuations of this work by future groups.

First, improvements are possible in the algorithms themselves. The algorithms in ARPACK as it is typically implemented are not parallel. Since most of the time is spent doing matrix-vector math, parallelizing at least that component of the calculation is in principle fairly straightforward. This would be a benefit especially for extremely large matrices, where it can still take a large amount of time to solve for any useful number of eigenvectors.

Another ongoing point of concern is the existance of imaginary components that occasionally appear in the eigenvalues of the fission matrix. This phenomenon was observed during the course of this project, but not systematically investigated. It is investigated in [1], which makes the preliminary conclusion that the imaginary components appear to be solely due to statistical variation. However, a more thorough investigation of these components with a larger parameter space would be necessary in order to gather convincing evidence of this suspicion.

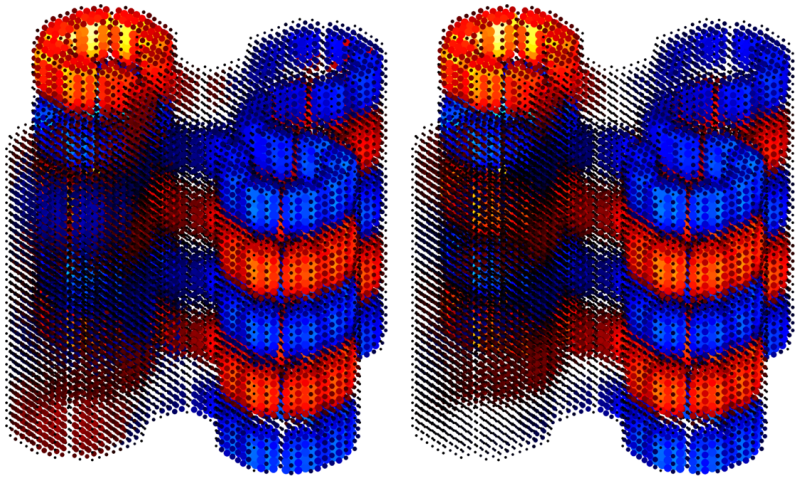


(a) Run 20 - Mode 26 (b) Run 20 - Mode 27 (c) Run 20 - Mode 28



(d) Average - Mode 26 (e) Average - Mode 27 (f) Average - Mode 28

Figure 10: Mode Comparison in 3D, Run 20 vs. Average



(a) Run 20 - Mode 27 - Modified (b) Average - Mode 27

Figure 11: Mode Comparison in 3D, Run 20 vs. Average, With Abnormal Values Removed

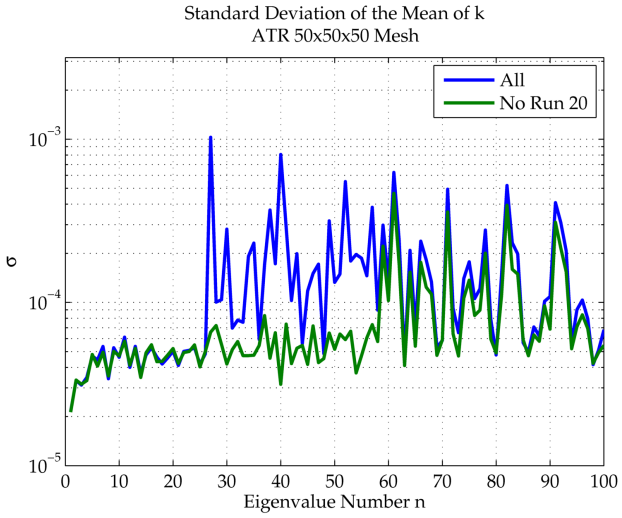


Figure 12: Standard Deviation With and Without Run 20

Finally, only a limited set of perturbations were explored in this project, and more interesting ones may be possible. For example, one might investigate the transitions from a core in cold-clean conditions to hot-clean conditions. It might also be interesting to investigate other temperature effects, void effects, or xenon oscillations. In principle, any type of transition can be investigated providing MCNP has the capability to simulate the perturbed case.

6 Conclusion

Fundamentally, most of the groundwork for using the fission matrices that MCNP produces has been completed. It is relatively cost-effective to generate fission matrices with meshes of acceptably fine resolution, with the primary limiters being memory and storage space. Large matrices provide more accurate results as long as the statistics are sufficient, and if they are not, the matrix can be reduced in size until they are sufficient. By switching from the power method to the implicitly restarted Arnoldi method, speedups on the order of 1500 fold were attained, proving this an effective algorithm for handling these large matrices. The calculated eigenvalues converged towards the `KCODE` result with expanding matrix size, and also appeared to have superior statistical properties due to the reduced number of discarded cycles, with the notable caveat of a bias appearing for insufficiently fine mesh resolutions. Lastly, the fundamental mode can be reconstructed in a perturbed space with relatively small errors with few eigenmodes used.

7 Acknowledgements

We would like to gratefully acknowledge the guidance of our mentor, Forrest Brown, and everyone else at Los Alamos National Laboratory who made the Computational Physics Worshop and this project possible, especially Scott Runnels, the organizer of the workshop.

References

[1] Forrest B. Brown, Sean E. Carney, Brian C. Kiedrowski, and William R. Martin. Fission matrix capability for MCNP, part I -

Theory. Technical Report LA-UR-13-20429, Los Alamos National Laboratory, May 2013. URL https://laws.lanl.gov/vhosts/mcnp.lanl.gov/pdf_files/la-ur-13-20429.pdf. Intended for Mathematics & Computation 2013, Sun Valley, ID, USA.

[2] David S Watkins. Understanding the QR algorithm, part II. URL http://www.researchgate.net/publication/228966308_Understanding_the_QR_algorithm_Part_II/file/79e4150e24b0e6b291.pdf.

[3] Y. Saad. *Numerical Methods for Large Eigenvalue Problems-classics edition*. SIAM, Philadelpha, PA, 2011. doi: 10.1137/1.9781611970739. URL http://dx.doi.org/10.1137/1.9781611970739.

[4] James S Warsa, Todd A Wareing, Jim E Morel, John M McGhee, and Richard B Lehoucq. Krylov subspace iterations for deterministic k-eigenvalue calculations. *Nuclear Science and Engineering*, 147(1):26–42, 2004.

[5] Bernhard Beckermann and Stefan Güttel. Superlinear convergence of the rational Arnoldi method for the approximation of matrix functions. *Numerische Mathematik*, 121(2):205–236, 2012.

[6] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users' Guide*, October 1997. URL http://www.caam.rice.edu/software/ARPACK/.

[7] Danny Sorensen. Caam 551: Advanced numerical linear algebra, matlab code, 2011. URL http://www.caam.rice.edu/~caam551/MatlabCode/matlabcode.html.

A Biorthogonality Relation

For an $N \times N$ matrix A with N distinct eigenvalues and a set of right eigenvectors $\{\mathbf{r}_i\}$ and associated left eigenvectors $\{\mathbf{l}_i\}$ (here written as column vectors), the following relation holds:

$$\begin{aligned} \langle A\mathbf{r}_j, \mathbf{l}_i \rangle &= \langle \lambda_j \mathbf{r}_j, \mathbf{l}_i \rangle \\ = \langle \mathbf{r}_j, A^* \mathbf{l}_i \rangle &= \langle \mathbf{r}_j, \lambda_i^* \mathbf{l}_i \rangle \end{aligned} \tag{A.1}$$

so

$$\left\langle \mathbf{r}_j, \mathbf{l}_i \right\rangle (\lambda_i - \lambda_j) = 0 \tag{A.2}$$

by sesquilinearity of the dot product defined in Section 2.1. This means that for $\lambda_i \neq \lambda_j$, the product $\langle \mathbf{r}_j, \mathbf{l}_i \rangle = 0$. Therefore, since the matrix A and its complex conjugate transpose A^* have sets of eigenvalues that are complex conjugates of each other, the sets of left and right eigenvectors $\{\mathbf{l}_i\}$ and $\{\mathbf{r}_j\}$ form a biorthogonal system.

In the more general case in which A has repeated eigenvalues, which is often observed to be the case with fission matrices, a more general relation is needed which makes use of the Jordan normal form. Any square matrix can be decomposed in the form $A = RJR^{-1}$, where J is a block-diagonal matrix where the individual blocks are Jordan blocks, which have an eigenvalue on the diagonal and ones on the superdiagonal. The number of Jordan blocks corresponding to an eigenvalue λ_i is equal to its geometric multiplicity, or the number of linearly independent eigenvectors corresponding to that eigenvalue ($= \dim \text{Null}(A - \lambda_i I)$). All semi-simple eigenvalues, i.e. those that have a geometric multiplicity equal to their algebraic multiplicty (their multiplicity as roots of the characteristic polynomial of A), have Jordan blocks of size 1. For more details, see [3, pp. 14-15] or any standard textbook on linear algebra. For the purposes of illustration, a Jordan block of the eigenvalue λ_i has the form:

$$\begin{pmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{pmatrix}$$

Writing the above decomposition in the form $AR = RJ$, it can be seen that R contains all the right eigenvectors of A. The set of indices of the columns in R that comprise the eigenvectors of the matrix A is the set of (column) indices in J where each new Jordan block starts; this is the set of indices at which a column vector of R is scaled by the corresponding λ_i through the matrix multiplication.

Alternatively, the decomposition can be written $R^{-1}A = JR^{-1}$, or, letting $L = R^{-1}$, $LA = JL$. This form reveals that L contains all the *left* eigenvectors of A in its rows. This time, however, the eigenvectors occur at a different set of (row) indices: By inspecting the properties

of left-multiplication of a matrix by a Jordan block, one can see that the true eigenvectors occur at the set of indices where each Jordan block *ends*; these are the indices at which a row vector in L is scaled by the corresponding λ_i in the matrix multiplication.

Since $LR = I$, the identity matrix, the rows $\{\mathbf{l}_i\}_{i=1}^N$ and the columns $\{\mathbf{r}_i\}_{i=1}^N$ form a biorthogonal basis of $\mathbb{C}^N$. From this fact, it is possible to make a restricted conclusion regarding the biorthogonality of eigenvectors of a matrix: The sets of left and right eigenvectors of a matrix, with those vectors deleted that do not correspond to semi-simple eigenvalues, form a biorthogonal system. This is because, for semi-simple eigenvalues, the corresponding left and right eigenvectors have the same row indices in L as column indices in R, since Jordan blocks corresponding to semi-simple eigenvalues always have size 1.

This manipulation of the Jordan normal form offers another insight in the case where the fission matrix $\bar{F}$ does not have a complete, linearly independent set of eigenvectors: One can augment the existing eigenvectors with principal vectors from the Jordan normal form, forming two complete biorthogonal bases of $\mathbb{C}^N$ in which any arbitrary source distribution can be written as $\mathbf{s} = \sum_{i=1}^N a_i \mathbf{l}_i$, for example (see Section 2.1). More care is required, however, in extracting the j-th transition coefficient via $a_j = \langle \mathbf{s}, \mathbf{r}_j \rangle$. The index j is an index into a column of R, which includes principal vectors from the Jordan normal form. Taking into account the above-mentioned indexing irregularities, this means that while $\mathbf{l}_j$ may be an eigenvector, this does not imply that $\mathbf{r}_j$ is as well, and vice versa.