CP 2: Report

An Investigation into Neutron Balance and Criticality of Two and Eight Neutron Group Nuclear Systems

Specht, Joseph
NPRE 247
5/5/23

Purpose

The purpose of this analysis is to figure out the required flux (φ) and criticality values (k) required to maintain constant power output. To do this, we determine the eigenvalues and eigenvectors across multiple neutron energy group systems. Each set of eigenvalues and eigenvectors is important for operation of nuclear systems as the eigenvalues are the criticality value (k) and the eigenvectors are the flux vectors (φ).

These quantities are useful when designing reactors that are safe and functional as they allow mathematical exploration without the need for physical testing. Problems like this allow a deeper understanding of how reactors work without needing to be close to a reactor. This small evaluation will hardly be a substitute for hands on nuclear reactor testing, but, if it was scaled to be more accurate, it could be far more useful.

Part 1: Theory

This section is an explanation of the theory behind the tests that will occur later.

Description

We will be analyzing an infinite reactor with neutrons that have multiple energy groups. Doing this will help achieve a more accurate result as the more groups there are, the closer the mathematical predictions will be to reality. If we intend to scale this to higher numbers of groups, which we do, we should pick the group the neutrons are born from fission in to be Group 1 as it would be unwise to change their group every time a new group is added. To start, we will assume two energy groups: fast neutrons (Group 1) and thermal neutrons (Group 2). The equations of for balance being:

$$\Sigma_{a1}\varphi_1 + \Sigma_{1\to2}\varphi_1 = \frac{\chi_1}{k}\left(\nu\Sigma_{f1}\varphi_1 + \nu\Sigma_{f2}\varphi_2\right) + \Sigma_{2\to1}\varphi_2 \quad (1)$$

$$\Sigma_{a2}\varphi_2 + \Sigma_{2\to1}\varphi_2 = \frac{\chi_2}{k}\left(\nu\Sigma_{f2}\varphi_2 + \nu\Sigma_{f1}\varphi_1\right) + \Sigma_{1\to2}\varphi_1 \quad (2)$$

Where each Eq 1 & Eq 2 is the balance of neutrons lost on left side and gained on the right side, which should be equal as we intend to operate in equilibrium conditions. To do anything further, we convert these equations into matrices as follows:

$$\begin{bmatrix}\Sigma_{a1} & 0\\ 0 & \Sigma_{a2}\end{bmatrix}\begin{bmatrix}\varphi_1\\ \varphi_2\end{bmatrix} + \begin{bmatrix}\Sigma_{1\to2} & 0\\ 0 & \Sigma_{2\to1}\end{bmatrix}\begin{bmatrix}\varphi_1\\ \varphi_2\end{bmatrix} \quad (3)$$

$$= \frac{1}{k}\begin{bmatrix}\chi_1\nu\Sigma_{f1} & \chi_1\nu\Sigma_{f2}\\ \chi_2\nu\Sigma_{f2} & \chi_2\nu\Sigma_{f2}\end{bmatrix}\begin{bmatrix}\varphi_1\\ \varphi_2\end{bmatrix} + \begin{bmatrix}0 & \Sigma_{2\to1}\\ \Sigma_{1\to2} & 0\end{bmatrix}\begin{bmatrix}\varphi_1\\ \varphi_2\end{bmatrix}$$

Derivation

I will define the following matrices below to simplify the equations as these will be substituted into Eq 3.

$$\text{Absorption Matrix: } A = \begin{bmatrix} \Sigma_{a1} & 0 \\ 0 & \Sigma_{a2} \end{bmatrix} \tag{4}$$

$$\text{Outscattering Matrix: } S_{out} = \begin{bmatrix} \Sigma_{1\to2} & 0 \\ 0 & \Sigma_{1\to2} \end{bmatrix} \tag{5}$$

$$\text{Inscattering Matrix: } S_{in} = \begin{bmatrix} 0 & \Sigma_{2\to1} \\ \Sigma_{1\to2} & 0 \end{bmatrix} \tag{6}$$

$$\text{Fission Matrix: } F = \begin{bmatrix} \chi_1 \nu\Sigma_{f1} & \chi_1 \nu\Sigma_{f2} \\ \chi_2 \nu\Sigma_{f2} & \chi_2 \nu\Sigma_{f2} \end{bmatrix} \tag{7}$$

Substituting into Eq 3 then gives Eq 8 which can be rearranged to Eq 9:

$$A\begin{bmatrix}\varphi_1\\ \varphi_2\end{bmatrix} + S_{out}\begin{bmatrix}\varphi_1\\ \varphi_2\end{bmatrix} = \frac{1}{k}F\begin{bmatrix}\varphi_1\\ \varphi_2\end{bmatrix} + S_{in}\begin{bmatrix}\varphi_1\\ \varphi_2\end{bmatrix} \tag{8}$$

$$k\begin{bmatrix}\varphi_1\\ \varphi_2\end{bmatrix} = (A + S_{out} - S_{in})^{-1}F\begin{bmatrix}\varphi_1\\ \varphi_2\end{bmatrix} \tag{9}$$

Below we set another matrix for ease and observe and inspect Eq 11 below.

$$\text{Migration Matrix: } M = (A + S_{out} - S_{in}) \tag{10}$$

$$k\begin{bmatrix}\varphi_1\\ \varphi_2\end{bmatrix} = M^{-1}F\begin{bmatrix}\varphi_1\\ \varphi_2\end{bmatrix} \tag{11}$$

Eq 11 is in the form of an eigenvalue and eigenvector problem where k is the eigenvalue, flux is the eigenvector, and $M^{-1}F$ is the characteristic matrix. It should also be noted that the reason M is such because it is how matrix that tracks how neutrons are transferring from one group to another, while the fission matrix shows which energy groups and how many neutrons are coming from fission.

Part 2: Two-Group Analysis

This section is about applying Eq 11 for a two-neutron energy group system.

To begin solving Eq 11 for the two-group system, we will start by formulating the Migration Matrix (M) and the Fission Matrix (F) for the two-group scenario. The data to generate these matrices have been taken tables A-1 and A-2 in the appendix where the sources for the data are also found. If we interpret the data, we can substitute it directly into the Eq 4-7 and get the following matrices for the two-group system.

$$\text{Absorption Matrix: } A = \begin{bmatrix} 0.0092 & 0 \\ 0 & 0.0932 \end{bmatrix} \tag{12}$$

$$\text{Outscattering Matrix: } S_{out} = \begin{bmatrix} 0.0202 & 0 \\ 0 & 0 \end{bmatrix} \tag{13}$$

$$\text{Inscattering Matrix: } S_{in} = \begin{bmatrix} 0 & 0 \\ 0.0202 & 0 \end{bmatrix} \tag{14}$$

$$\text{Fission Matrix: } F = \begin{bmatrix} 0.0046 & 0.1139 \\ 0 & 0 \end{bmatrix} \tag{15}$$

All these matrices (Eq 12-15) will be used in three different methods to calculate the eigenvalues and eigenvectors of the two-group characteristic matrix $M^{-1}F$. These methods being a by hand analytical solution, computer software from a python package called SymPy, and power iteration.

Method 1: Analytical

To solve Eq 11 analytically, we start by defining the characteristic matrix.

$$\text{Characteristic Matrix: } A = (M^{-1}F) \tag{16}$$

We can then take this matrix and apply the equation below to begin finding the eigenvalues and eigen vectors.

$$\det(A - \lambda \mathrm{I}) = 0 \tag{17}$$

Solving Eq 17 gives a second order polynomial expression that can be solved to find the two eigenvalues (λ). Once we have the two eigenvalues, we solve the following equations to find their corresponding eigenvectors.

$$(A - \lambda_i \mathrm{I}) \begin{bmatrix} \varphi_1 \\ \varphi_2 \end{bmatrix} = 0 \tag{18}$$

$$\text{Assuming: } A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{19}$$

$$\begin{bmatrix} (a - \lambda_i) & b \\ c & (d - \lambda_i) \end{bmatrix} \begin{bmatrix} \varphi_1 \\ \varphi_2 \end{bmatrix} = 0 \tag{20}$$

$$\begin{bmatrix} (a - \lambda_i)\varphi_1 + b\varphi_2 \\ c\varphi_1 + (d - \lambda_i)\varphi_2 \end{bmatrix} = 0 \tag{21}$$

We can then solve each of the systems of equations generated by the matrix by plugging in the two eigenvalues (k) to find the eigen vectors (φ).

$$k_1 = 0 \qquad k_2 = .99614 \tag{22}$$

$$\varphi_1 = \begin{bmatrix} 24.7609 \\ 1 \end{bmatrix} \qquad \varphi_2 = \begin{bmatrix} 4.61386 \\ 1 \end{bmatrix} \tag{23}$$

These Eq 20 & Eq 21 show the results of solving the two-group reactor problem analytically by hand. By inspection, we see $k_2 > k_1$.

Method 2: Python Code

There are many packages available that allow somebody to calculate the eigenvectors and eigenvalues of matrices in Python. The package I used was SymPy, but others such as NumPy's linalg that also work. This method is the fastest and simplest to solve as the only requirement to solve is migrating the data into python. This method also generates all eigenvectors and eigenvalues, so if we are looking for the max eigenvalue, this method will get increasingly tedious as more groups are added, but inspection is enough to find the largest eigenvalue for now. The results of the findings are below.

$$k_1 = -2.129 * 10^{17} \qquad k_2 = .0.99614 \tag{24}$$

$$\varphi_1 = \begin{bmatrix} 0.99919 \\ -0.04035 \end{bmatrix} \qquad \varphi_2 = \begin{bmatrix} 1.00965 \\ 0.21883 \end{bmatrix} \tag{25}$$

It is interesting to note that one of the eigenvalues found to be 0 in the analytical solution turned out not to be 0. Another thing that is interesting is that Eq 25 has the same eigenvectors as Eq 23, but simply scaled by a real number.

Method 3: Power Iteration

The final method that will be looked at is power iteration, which involves generating a random vector in the same vector space as the eigenvector and continually multiplying it by the characteristic matrix A and then normalizing by the magnitude of that quantity. Then, this new vector has the same process done to it until it stops changing, which is when it is the eigenvector closest to the starting point. This same iterative method can be applied to the eigenvalue k with a different equation. In equation form, these look like the following.

$$\varphi_{i+1} = \frac{A\varphi_i}{||A\varphi_i||} \tag{26}$$

$$k_{i+1} = \frac{(A\varphi_{i+1})^T A\varphi_{i+1}}{(\varphi_{i+1})^T \varphi_{i+1}} \tag{27}$$

Generally, any random value for the first eigenvector and eigenvalue guesses are acceptable, but the following are what I used for my initial values.

$$\varphi_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{28}$$

$$k_0 = 1 \tag{29}$$

With Eq 26-28, we can now solve for the eigenvector and eigenvalue after 2 iterations.

$$\varphi_2 = \begin{bmatrix} 0.97354 \\ 0.21100 \end{bmatrix} \tag{30}$$

$$k_2 = 0.99614 \tag{31}$$

Another interesting observation is that Eq 30 is almost the same as Eq 23, just scaled by a real number, which makes sense as they should be the same eigenvector.

Conclusion

Comparing all the results, they are all relatively similar, however, I suspect the eigenvectors and eigenvalues found by the power iteration are the closest to the true values because they can be iterated to a higher step to improve accuracy where the other two methods have no way to improve accuracy without taking more decimal places.

Part 3: Eight-Group Analysis

This section is about applying Eq 11 for a two-neutron energy group system.

In much the same way as the two-group system, an eight-group system follows the same general form as Eq 11. For the two-group system, the flux vector was 2x1 and the characteristic polynomial was 2x2. However, in the 8-group system, the flux vector is 8x1 and the characteristic matrix is 8x8. Even though the dimensions of the vectors change, the equation stays the same.

To get started, we need to generate the migration matrix (M) and fission matrix (F). The data for these can be found in tables A-3 and A-4 which are in the appendix. These matrices are as follows.

$$M = \begin{bmatrix} .088 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -.053 & .1193 & 0 & 0 & 0 & 0 & 0 & 0 \\ -.0301 & -.1159 & 0 & 0 & 0 & 0 & 0 & 0 \\ -.0001 & -.0005 & .2152 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -.1961 & .2529 & 0 & 0 & 0 & 0 \\ 0 & 0 & -.005 & -.1737 & .3437 & -.0023 & -.0023 & 0 \\ 0 & 0 & -.0007 & -.0246 & -.2707 & .417 & .417 & -.0275 \\ 0 & 0 & -.0001 & -.0073 & -.055 & -.055 & -.3589 & .2073 \end{bmatrix} \quad (32)$$

$$F = \begin{bmatrix} .00470 & .00196 & .000386 & .00235 & .00772 & .00772 & .00772 & .00779 \\ .0055 & .00230 & .00275 & .00275 & .00903 & .00903 & .00903 & .00911 \\ .00312 & .00134 & .00160 & .00160 & .00525 & .00525 & .00525 & .00510 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (33)$$

Now that we have these, we can apply the same methods as before with two-group.

Method 1: Python Code

Using the same code as before with the two-group but replacing the old matrices with the migration and fission matrices for the eight-group. This substitution generates the following eigenvectors and eigenvalues.

$$k_1 = 3.026e - 50 \tag{34}$$

$$k_2 = -7.500 * 10^{-18} - 4.128i * 10^{-50} \tag{35}$$

$$k_3 = -6.339 * 10^{-19} - 5.945i * 10^{-18} \tag{36}$$

$$k_4 = -6.339 * 10^{-19} + 5.945i * 10^{-18} \tag{37}$$

$$k_5 = -5.806 * 10^{-19} + 1.256i * 10^{-50} \tag{38}$$

$$k_6 = 8.971 * 10^{-18} \tag{39}$$

$$k_7 = 1.259 * 10^{-17} \tag{40}$$

$$k_8 = 1.090 \tag{41}$$

Above are the eigenvalues for the eight-group system.

$$\varphi_1 = \begin{bmatrix} 0.265 + 0.360i \\ 1.961 * 10^{-32} + 2.665i * 10^{-32} \\ -1.281 * 10^{-31} - 1.734i \\ 2.598 * 10^{-33} + 3.531i * 10^{-33} \\ -5.310 * 10^{-33} - 7.316i * 10^{-33} \\ 4.356 * 10^{-33} + 5.922i * 10^{-33} \\ 4.358 * 10^{-33} - 1.424i * 10^{-33} \\ -1.048 * 10^{-33} - 4.128i * 10^{-50} \end{bmatrix} \tag{42}$$

$$\varphi_2 = \begin{bmatrix} -0.0869 + 0.0954i \\ -0.492 + 0.540i \\ -0.414 + 0.454i \\ -0.233 + 0.256i \\ 0.0700 - 0.0768i \\ 0.0374 - 0.0410i \\ -0.00664 + 0.00723i \\ 0.0195 - 0.0213i \end{bmatrix} \tag{43}$$

$$\varphi_3 = \begin{bmatrix} -0.0742 + 0.0929i \\ -0.213 - 0.0207i \\ -1.317 - 0.153i \\ 0.537 - 0.145i \\ 0.00383 + 0.114i \\ -0.0161 + 0.0600i \\ -0.0175 - 0.0852i \\ 0.00878 + 0.0178 \end{bmatrix} \tag{44}$$

$$\varphi_4 = \begin{bmatrix} 0.00822 - 0.124i \\ -0.180 - 0.135i \\ -1.128 - 0.812i \\ 0.314 + 0.492i \\ 0.0834 - 0.0855i \\ 0.0299 - 0.0579i \\ -0.0738 + 0.0536i \\ 0.0194 - 0.00760i \end{bmatrix} \tag{45}$$

$$\varphi_5 = \begin{bmatrix} -0.297 - 0.264 \\ 0.0560 + 0.0498i \\ -0.777 - 0.691i \\ 0.687 + 0.611i \\ 0.0268 + 0.0238i \\ 0.0107 + 0.00956i \\ 0.00374 + 0.00332i \\ -0.00581 - 0.00516i \end{bmatrix} \tag{46}$$

$$\varphi_6 = \begin{bmatrix} 0.0879 + 9.815i * 10^{-33} \\ 0.0588 + 7.939i * 10^{-33} \\ -0.559 - 5.798i * 10^{-32} \\ -0.470 + 4.866i * 10^{-33} \\ -0.0166 - 9.885i * 10^{-33} \\ -0.139 - 7.661i * 10^{-33} \\ -0.0471 + 7.674i * 10^{-34} \\ 0.0464 - 1 * 10^{-33} \end{bmatrix} \tag{47}$$

$$\varphi_7 = \begin{bmatrix} -0.0733 - 1.522i * 10^{-32} \\ -0.0337 - 3.708i * 10^{-33} \\ 1.434 + 1.210i * 10^{-31} \\ 0.823 - 1.971i * 10^{-32} \\ 0.157 + 2.876i * 10^{-32} \\ 0.264 + 2.022i * 10^{-32} \\ 0.124 - 5.770i * 10^{-33} \\ -0.123 + 2.633i * 10^{-33} \end{bmatrix} \tag{48}$$

$$\varphi_8 = \begin{bmatrix} -0.102 - 5.733i * 10^{-33} \\ -0.135 - 1.372i * 10^{-33} \\ -0.306 + 2.576i * 10^{-33} \\ -0.110 - 1.614i * 10^{-33} \\ -0.0873 - 4.435i * 10^{-33} \\ -0.0460 - 1.351i * 10^{-33} \\ -0.0409 + 2.521i * 10^{-33} \\ -0.0861 - 7.457i * 10^{-34} \end{bmatrix} \quad (49)$$

Above are the eigenvectors corresponding to the eigenvalues from Eq 34-41. It is curious that the eigenvalues and eigenvectors have imaginary components. The only hypothesis I have is that some eigenvalues are truly imaginary, but the computer program failed somehow with the other eigenvalues and eigenvectors and added an imaginary component not truly there. I believe this because the magnitudes of quite a few imaginary parts in Eq 42-49 have magnitudes on the order of 10^{-33}, which is effectively 0 in any calculation.

Method 2: Power Iteration

To find the eigenvalue closest to 1 from power iteration, we start with k=1 and the flux vector being 1 in all 8 places. This plugged into Eq 26-27, and applied for 2 iterations, generates the following results.

$$k_2 = 1.09003 \tag{50}$$

$$\varphi_8 = \begin{bmatrix} 0.28514 \\ 0.37511 \\ 0.85238 \\ 0.85238 \\ 0.24556 \\ 0.12820 \\ 0.11392 \\ 0,,23400 \end{bmatrix} \tag{51}$$

It is concerning that the k value for after this power iteration is greater than 1 as this indicates somebody operating the reactor at an initial criticality of 1 would have a supercritical reactor soon after starting, which can runaway too quickly for a human to solve.

Conclusion

It seems that the method of power iteration is the most accurate as it generates a sensible value that does not include imaginary numbers like the python code. Imaginary numbers are not a cause for concern in general, but in this instance, it would not be appropriate as the criticality should be a real number and the flux should be a real vector as you cannot have imaginary flow of neutrons.

Appendix

Group	Σ_a	$\nu\Sigma_f$	χ
1	0.0092	.0046	1.0000
2	0.0932	0.1139	0

Table A-1: Two-group data from NEACRP L336, assumes homogeneous UO2 composition.

To (row) ↓	**From (col) →**	1	2
1		1.0000	0
2		0.0202	2.0000

Table A-2: Two-group scattering cross sections $\Sigma_{col \rightarrow row}$.

Group	Σ_a	$\nu\Sigma_f$	χ
1	0.0056	0.0134	0.3507
2	0.0029	0.0056	0.4105
3	0.0025	0.0011	0.2388
4	0.0133	0.0067	0
5	0.0473	0.0220	0
6	0.0180	0.0222	0
7	0.0558	0.0897	0
8	0.1798	0.2141	0

Table A-3: Eight-group data from VENUS-2, UO2 3.3% composition, xs_{pin2}-8g-LF

To (row) ↓	**From (col) →**	1	2	3	4	5	6	7	8
1		0.1179	0	0	0	0	0	0	0
2		0.0530	0.1949	0	0	0	0	0	0
3		0.0301	0.1159	0.5868	0	0	0	0	0
4		0.0001	0.0005	0.0769	0.8234	0	0	0	0
5		0	0	0.0019	0.1961	0.8180	0	0	0
6		0	0	0	0.0050	0.1737	.6902	0.6902	0
7		0	0	0	0.0007	0.0246	0.2707	0.8626	0.0275
8		0	0	0	0.0001	0.0073	0.0550	0.3589	1.9761

Table A-4: Eight-group scattering cross sections $\Sigma_{col \rightarrow row}$.