NPRE 247 Computational Project 2

Arnav Goyal

May 2, 2024

1 Introduction

This report aims to solve the two (and eight) group neutron balance equations using both the computer and through the power iteration method. The neutron balance equation is based on cross-section data: absorption, fission, and scattering cross-sections are all taken into account. The neutron balance equation is based on the idea that there is an equivalence between the loss and gain for each of the two energy groups

$$\begin{cases} Loss = Gain \\ Absorption + Outscattering = Fission + Inscattering \\ \Sigma_{ag}\phi_g + \Sigma_{g\to g'}\phi_g = \frac{\chi_g}{k}(\nu\Sigma_{fg}\phi_g + \nu\Sigma_{fg'}\phi_{g'}) + \Sigma_{g'\to g}\phi_{g'} \end{cases} \tag{1}$$

Of note in the final equation from Equation 1 is the way the specific values are calculated. Absorption is based on the macroscopic absorption cross-section of an energy group multiplied by its corresponding flux. The out-scattering for the energy group is the scattering cross section into any other energy group present, for the two energy group neutron balance equation, there is only one other energy group to scatter into from the original energy group. The fission part of the equation is based on the chi value of the original group multiplied by the fission values for both the original energy group and all of the other energy groups available. The final piece, the in-scattering part, is for every energy group that goes into the energy group at hand, multiplied by the flux of the energy group the neutron is scattering in from.

$$\begin{cases} \Sigma_{a1}\phi_1 + \Sigma_{1\to 2}\phi_1 = \frac{\chi_1}{k}(\nu\Sigma_{f1}\phi_1 + \nu\Sigma_{f2}\phi_2) + \Sigma_{2\to 1}\phi_2 \\ \Sigma_{a2}\phi_2 + \Sigma_{2\to 1}\phi_2 = \frac{\chi_2}{k}(\nu\Sigma_{f2}\phi_2 + \nu\Sigma_{f1}\phi_1) + \Sigma_{1\to 2}\phi_1 \end{cases} \tag{2}$$

For the two-group neutron balance equation, we get two balance equations as seen in Equations 2 These two equations can then be rearranged into matrix form as seen in Equation 3.

$$\begin{cases} \begin{bmatrix} \Sigma_{a1} & 0 \\ 0 & \Sigma_{a2} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} + \begin{bmatrix} \Sigma_{1\to 2} & 0 \\ 0 & \Sigma_{2\to 1} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} - \begin{bmatrix} 0 & \Sigma_{2\to 1} \\ \Sigma_{1\to 2} & 0 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} = \frac{1}{k} \begin{bmatrix} \chi_1\nu\Sigma_{f1} & \chi_1\nu\Sigma_{f2} \\ \chi_2\nu\Sigma_{f1} & \chi_2\nu\Sigma_{f2} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} \\ \begin{bmatrix} A \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} + \begin{bmatrix} S_{out} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} - \begin{bmatrix} S_{in} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} = \frac{1}{k} \begin{bmatrix} F \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} \end{cases} \tag{3}$$

The absorption matrix is given first, which is added to the out-scattering matrix and then the in-scattering matrix is subtracted. This is equivalent to the fission matrix times the inverse of the value for k. The migration matrix can now be defined as everything on the left-hand side that is being multiplied by the flux, specifically as seen in Equation 4

$$Migration = \begin{bmatrix} A + S_{out} - S_{in} \end{bmatrix} = \begin{bmatrix} \Sigma_{a1} + \Sigma_{1\to 2} & -\Sigma_{2\to 1} \\ -\Sigma_{1\to 2} & \Sigma_{a2} + \Sigma_{2\to 1} \end{bmatrix} \tag{4}$$

The fission matrix is calculated using the chi (χ) values given and the fission cross sections multiplied by the average number of neutrons produced per fission (also given); the chi values are the same for every row and the $\nu\Sigma_f$ values are the same for every column, giving Equation 5.

$$Fission = \begin{bmatrix} \chi_1\nu\Sigma_{f1} & \chi_1\nu\Sigma_{f2} \\ \chi_2\nu\Sigma_{f1} & \chi_2\nu\Sigma_{f2} \end{bmatrix} \tag{5}$$

If this migration matrix is called M and the fission matrix is given by F, the problem in Equation 3 becomes a simplified eigenvalue problem as seen in Equation 6 once rearranged.

$$k \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} = \big[M\big]^{-1} \big[F\big] \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} \tag{6}$$

Equation 6 can be solved through a few methods (Python's *numpy.linalg* package and power iteration) which will be delved into in the next section, an extremely important missing component is the data required to create the migration and fission matrices.

2 Two-Group Eigenvalue/Eigenvector Solution

The input data for the two energy group neutron balance equations is given in Table 1 and Table 2.

Table 1: Two-group data, assuming homogeneous UO_2 composition

Group	Σ_a	$\nu\Sigma_f$	χ
1	0.0092	0.0046	1.0000
2	0.0932	0.1139	0.0000

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

To(row)$\downarrow$ From(col)$\rightarrow$	1	2
1	1.0000	0.0000
2	0.0202	2.0000

Both of these tables of data allow us to form the absorption, and scattering (in and out) matrices using Equation 3. Plugging these matrices into Equation 4, for the migration matrix, yields the matrices seen in Equation 7.

$$\begin{cases} \Big[A\Big] = \begin{bmatrix} 0.0092 & 0 \\ 0 & 0.0932 \end{bmatrix} \\ \Big[S_{out}\Big] = \begin{bmatrix} 0.0202 & 0 \\ 0 & 0 \end{bmatrix} \\ \Big[S_{in}\Big] = \begin{bmatrix} 0 & 0 \\ 0.0202 & 0 \end{bmatrix} \end{cases} \tag{7}$$

These matrices can be used to calculate the migration and fission matrix and since $[\,B\,] = [\,M\,]^{-1}[\,F\,]$ the matrix for B, as seen in Equation 8

$$\begin{cases} \Big[M\Big] = \begin{bmatrix} 0.0294 & 0 \\ -0.0202 & 0.0932 \end{bmatrix} \\ \Big[F\Big] = \begin{bmatrix} 0.0046 & 0.1139 \\ 0 & 0 \end{bmatrix} \\ \Big[B\Big] = \begin{bmatrix} 0.15646259 & 3.87414966 \\ 0.03391142 & 0.83967621 \end{bmatrix} \end{cases} \tag{8}$$

Placing B into Equation 6 yields Equation 18

$$k \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} = \begin{bmatrix} 0.15646259 & 3.87414966 \\ 0.03391142 & 0.83967621 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} \tag{9}$$

This equation can then be solved in many ways, this report will walk through the solution on its own, display all of the eigenvalues\eigenvectors that the program calculated using the *numpy.linalg* package, and display the largest eigenvalue and corresponding eigenvector as calculated using power iteration.

2.1 By-Hand

To solve for the eigenvalues of B, as seen in Equation 8, one can use Equation 10.

$$det(B - \lambda I) = 0 \tag{10}$$

The resulting matrix, which should have a determinant that equals 0, is $\left[\begin{smallmatrix} 0.1565-\lambda & 3.8741 \\ 0.0339 & 0.8396-\lambda \end{smallmatrix}\right]$. The resulting equation solving for the lambdas is Equation 11

$$0.13137791457 - 0.9961388\lambda + \lambda^2 - 0.13137791626 = 0 \tag{11}$$

Simplifying this equation gives the easily solvable equation, Equation 12.

$$\lambda(\lambda - 0.9961388) = 0 \tag{12}$$

The resulting λ values, which are the eigenvalues for the original B matrix, are 0 and 0.9961388. Since we are concerned with the non-zero eigenvalues, specifically 0.9961388. To calculate the eigenvector associated with this eigenvalue, x_1 and x_2 in Equation 13 needs to be solved, where $\lambda = k = 0.9961388$.

$$\left[B - \lambda I\right] \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} = 0 \tag{13}$$

This results in the two equations seen in Equation 14 when the numbers and the chosen eigenvalue.

$$\begin{cases} -0.8396\phi_1 + 3.8741\phi_2 = 0 \\ 0.0339\phi_1 + -0.1565\phi_2 = 0 \end{cases} \tag{14}$$

The resulting vector for $\left[\begin{smallmatrix} \phi_1 \\ \phi_2 \end{smallmatrix}\right]$ is $\left[\begin{smallmatrix} 4.6142 \\ 1 \end{smallmatrix}\right]$ which is the eigenvector for the k value 0.9961.

2.2 Numpy.Linalg solutions

The results from the linalg package in numpy in Python were simply calculated from just $[\,B\,]$. The eigenvalues were 0.0 and 0.9961388. The first eigenvector is $\begin{bmatrix} -0.9991854 \\ 0.0403534 \end{bmatrix}$ and the second eigenvector is $\begin{bmatrix} -0.9773087 \\ -0.2118201 \end{bmatrix}$ which becomes $\begin{bmatrix} 0.9773087 \\ 0.2118201 \end{bmatrix}$ when made positive. This second eigenvector corresponds to an eigenvalue of 0.9961388.

2.3 Power Iteration

The power iteration method used an initial k value of 1 and flux matrix of $\begin{bmatrix}1\\1\end{bmatrix}$. The next flux values are calculated based on Equation 15. The denominator of this equation is the Euclidean norm of the resulting vector when multiplying B with ϕ_i. These new flux values are then used to calculate a new eigenvalue (k) using Equation 16. This continues happening until the values stop changing by a lot, for this report, the code was designed to complete 2 iterations.

$$\phi_{i+1} = \frac{B\phi_i}{\|B\phi_i\|^2} \tag{15}$$

$$k_{i+1} = \frac{(B\phi_{i+1})^T \phi_{i+1}}{(\phi_{i+1}^T \phi_{i+1})} \tag{16}$$

The results from the power iteration for the two-group neutron balance equation after 2 loops through these equations according to the program are $k = 0.9961388$ and the flux eigenvector is $\begin{bmatrix}0.9773087\\0.2118201\end{bmatrix}$

3 Conclusion

The three methods of calculating eigenvalues and eigenvectors for the matrix $\left[B\right]$ which is equivalent to $\left[\,M\,\right]^{-1}\left[\,F\,\right]$, where $\left[M\right] = \left[A + S_{out} - S_{in}\right] = \left[\begin{smallmatrix} \Sigma_{a1}+\Sigma_{1\to 2} & -\Sigma_{2\to 1} \\ -\Sigma_{1\to 2} & \Sigma_{a2}+\Sigma_{2\to 1} \end{smallmatrix}\right]$ and $\left[\,F\,\right] = \left[\begin{smallmatrix} \chi_1\nu\Sigma_{f1} & \chi_1\nu\Sigma_{f2} \\ \chi_2\nu\Sigma_{f1} & \chi_2\nu\Sigma_{f2} \end{smallmatrix}\right]$, all yielded equivalent numbers for two group neturon balance problem. The consensus, from these three methods, was that the eigenvalue k is 0.9961 while the eigenvector, $\left[\begin{smallmatrix} \phi_1 \\ \phi_2 \end{smallmatrix}\right]$, points in the direction of $\left[\begin{smallmatrix} 0.9773 \\ 0.2118 \end{smallmatrix}\right]$.

4 Extra-Credit Eight Group Eigenvalue/Eigenvector Solution

Table 3: Eight-group data, assuming $UO_2$3.3% composition

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	0.0000
5	0.0473	0.0220	0.0000
6	0.0180	0.0222	0.0000
7	0.0558	0.0897	0.0000
8	0.1798	0.2141	0.0000

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

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	0.6902	0.0023	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

The tables of data, Table 3 and Table 4, for the eight-group neutron balance equation, allow us to form the absorption, and scattering (in and out) matrices. These matrices can be used to calculate the migration and fission matrix and since $[\,B\,] = [\,M\,]^{-1}[\,F\,]$ the matrix for B, as seen in Equation 17 (some of the values have been rounded for presentation in this equation). The general concepts are the same, the migration matrix is still calculated as explained in Equation 4 (absorption + outscattering - inscattering), but the matrices are $8X8$s now. The fission matrix is also an $8X8$ matrix and follows the same general pattern, the chi values are based on the row index of the number in the matrix, and the other term (average neutron produced per fission multiplied by macroscopic fission cross section in a particular energy group) is based on the column index of the number in the matrix.

$$\begin{cases} \Big[M\Big] = \begin{bmatrix} 0.0888 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0530 & .1193 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0301 & -0.1159 & 0.0813 & 0 & 0 & 0 & 0 & 0 \\ -0.0001 & -0.0005 & -0.0769 & 0.2152 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.0019 & -0.1961 & 0 & 0.2529 & 0 & 0 \\ 0 & 0 & 0 & -0.0050 & -0.1737 & 0.3473 & -0.0023 & 0 \\ 0 & 0 & 0 & -0.0007 & -0.0246 & -0.2707 & 0.4170 & -0.0275 \\ 0 & 0 & 0 & -0.0001 & -0.0073 & -0.0550 & -0.3589 & 0.2073 \end{bmatrix} \\ \Big[F\Big] = \begin{bmatrix} 0.0047 & 0.0020 & 0.0004 & 0.0023 & 0.0077 & 0.0078 & 0.0315 & 0.0751 \\ 0.0055 & 0.0023 & 0.0005 & 0.0028 & 0.0090 & 0.0091 & 0.0368 & 0.0879 \\ 0.0032 & 0.0013 & 0.0003 & 0.0016 & 0.0053 & 0.0053 & 0.0214 & 0.0511 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ \Big[B\Big] = \begin{bmatrix} 0.0529 & 0.0221 & 0.0043 & 0.0265 & 0.0869 & 0.0877 & 0.3542 & 0.8456 \\ 0.0696 & 0.0291 & 0.0057 & 0.0348 & 0.1143 & 0.1153 & 0.4660 & 1.1123 \\ 0.1582 & 0.0661 & 0.0130 & 0.0791 & 0.2597 & 0.2621 & 1.0590 & 2.5277 \\ 0.0567 & 0.0237 & 0.0047 & 0.0284 & 0.0931 & 0.0940 & 0.3797 & 0.9062 \\ 0.0452 & 0.0189 & 0.0037 & 0.0226 & 0.0742 & 0.0748 & 0.3024 & 0.7217 \\ 0.0238 & 0.0099 & 0.0020 & 0.0119 & 0.0391 & 0.0394 & 0.1593 & 0.3802 \\ 0.0211 & 0.0088 & 0.0017 & 0.0106 & 0.0347 & 0.0350 & 0.1415 & 0.3378 \\ 0.0445 & 0.0186 & 0.0037 & 0.0223 & 0.0731 & 0.0738 & 0.2981 & 0.7116 \end{bmatrix} \end{cases} \tag{17}$$

Placing B into Equation 6 yields Equation 18

$$k \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \\ \phi_7 \\ \phi_8 \end{bmatrix} = \begin{bmatrix} 0.0529 & 0.0221 & 0.0043 & 0.0265 & 0.0869 & 0.0877 & 0.3542 & 0.8456 \\ 0.0696 & 0.0291 & 0.0057 & 0.0348 & 0.1143 & 0.1153 & 0.4660 & 1.1123 \\ 0.1582 & 0.0661 & 0.0130 & 0.0791 & 0.2597 & 0.2621 & 1.0590 & 2.5277 \\ 0.0567 & 0.0237 & 0.0047 & 0.0284 & 0.0931 & 0.0940 & 0.3797 & 0.9062 \\ 0.0452 & 0.0189 & 0.0037 & 0.0226 & 0.0742 & 0.0748 & 0.3024 & 0.7217 \\ 0.0238 & 0.0099 & 0.0020 & 0.0119 & 0.0391 & 0.0394 & 0.1593 & 0.3802 \\ 0.0211 & 0.0088 & 0.0017 & 0.0106 & 0.0347 & 0.0350 & 0.1415 & 0.3378 \\ 0.0445 & 0.0186 & 0.0037 & 0.0223 & 0.0731 & 0.0738 & 0.2981 & 0.7116 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \\ \phi_7 \\ \phi_8 \end{bmatrix} \tag{18}$$

This equation can then be solved in many ways, this report will display all of the eigenvalues\eigenvectors that the program calculated using the *numpy.linalg*

package, and display the largest eigenvalue and corresponding eigenvector as calculated using power iteration.

4.1 Numpy.Linalg solutions

The *numpy.linalg* package, when applied to $[B]$, gives both complex and real eigenvalues and eigenvectors. For the sake of simplicity, this report only focuses on the real solutions, where both the eigenvalues and eigenvectors are both real. These eigenvalues, according to the numpy package, are 0, and 1.0900. There are other real eigenvalues but they are all on the order of 10^{-17} and 10^{-18}. The eigenvector for 0 is $[\,0.9827\ 0.0660\ 0.1499\ 0.0537\ 0.0428\ 0.0225\ 0.0200\ 0.0422\,]$, while the eigenvalue for 1.0900 is eigenvector is
$[\,-0.2616\ -0.3441\ -0.7820\ -0.2804\ -0.2233\ -0.1176\ -0.1045\ -0.2201\,]$.

4.2 Power Iteration

To calculate the largest eigenvalue and eigenvector through power iteration for the eight-group neutron balance equation, the program followed the same process described in Section 2.3. Equation 15 and Equation 16 are used for 2 iterations on the $[B]$ to calculate the eigenvalue and eigenvector in question. The final k is 1.0900 and the final flux vector, the eigenvector is
$[\,0.2616\ 0.3441\ 0.7820\ 0.2804\ 0.2233\ 0.1176\ 0.1045\ 0.2201\,]$.