Computer Project 2

November 14, 2020

Andrew Liu
University of Illinois at Urbana-Champaign
NPRE 247

Theory

In order to model a steady-state neutron balance problem in infinite space with 2 and 8 discrete neutron energy groups, Python code was used to represent the problem in matrix form and solve for the eigenvalues and eigenvectors of neutron flux using both pre-coded (numpy.linalg.eig) as well as a power iteration algorithm (for dominant eigenvalue-eigenvector, only). Hand-calculated solutions were also found for the 2-group model.

General formulas:

Following the statement that, for a steady-state neutron balance of a point-like fission reaction into the surroundings and assuming that the only reactions are scattering, absorption, and fission, the rate of loss must be equal to the rate of gain for each discrete neutron energy level, we obtain the following:

$$loss\ rate = gain\ rate$$
$$Outscattering\ rate + absorption\ rate = inscattering\ rate + fission\ rate \tag{1}$$

Thus, for G discrete neutron energies, with *1* having the highest energy and *G* having the lowest energy, the neutron balance expressions for energy *h* will be as follows:

$$Scattering\ cross\ section\ from\ energy\ g\ to\ h = \Sigma_{h \leftarrow g}$$
$$Absorption\ cross\ section\ at\ energy\ h = \Sigma_{a,h}$$
$$Fission\ cross\ section\ at\ energy\ h = \Sigma_{f,h}$$
$$Flux\ of\ neutrons\ at\ energy\ h = \Phi_h$$
$$Criticality\ constant = k$$
$$Average\ number\ of\ neutrons\ produced\ by\ fission = v$$
$$Fission\ probability\ distribution\ as\ a\ function\ of\ incident\ neutron\ energy\ h = X_h$$

$$Outscattering\ rate\ = \left[\Sigma_{1\leftarrow h}\Phi_h + \Sigma_{2\leftarrow h}\Phi_h + \ldots + \Sigma_{G\leftarrow h}\Phi_h\right] \tag{2}$$

$$Absorption\ rate\ = \left[\Sigma_{a,h}\Phi_h\right] \tag{3}$$

$$Inscattering\ rate\ = \left[\Sigma_{h\leftarrow 1}\Phi_1 + \Sigma_{h\leftarrow 2}\Phi_2 + \ldots + \Sigma_{h\leftarrow G}\Phi_G\right] \tag{4}$$

$$Fission\ rate\ = \frac{1}{k}X_h\left[v\Sigma_{f,1}\phi_1 + v\Sigma_{f,2}\phi_2 + \ldots + v\Sigma_{f,G}\phi_G\right] \tag{5}$$

With flux as the unknown variable, equations (2) through (5) are arranged into matrices:

$$Outscattering\ matrix\ \left[S_{out}\right] = \begin{bmatrix} \sum_{g\neq 1}\Sigma_{g\leftarrow 1} & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & \sum_{g\neq h}\Sigma_{g\leftarrow h} & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \sum_{g\neq G}\Sigma_{g\leftarrow G} \end{bmatrix} \tag{6}$$

$$Absorption\ matrix\ [A] = \begin{bmatrix} \Sigma_{a,1} & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & \Sigma_{a,h} & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \Sigma_{a,G} \end{bmatrix} \tag{7}$$

$$Inscattering\ matrix\ \left[S_{in}\right] = \begin{bmatrix} 0 & \Sigma_{1\leftarrow 2}\ \cdots & \Sigma_{1\leftarrow h} & \Sigma_{1\leftarrow h+1}\ \cdots & \Sigma_{1\leftarrow G} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \Sigma_{h\leftarrow 1} & \Sigma_{h\leftarrow 2}\ \ldots & 0 & \Sigma_{h\leftarrow h+1}\ \cdots & \Sigma_{h\leftarrow G} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \Sigma_{G\leftarrow 1} & \Sigma_{G\leftarrow 2}\ \cdots & \Sigma_{G\leftarrow h} & \Sigma_{G\leftarrow h+1}\ \cdots & 0 \end{bmatrix} \tag{8}$$

$$Fission\ matrix\ [F] = \begin{bmatrix} X_1\nu\Sigma_{f,1} & \cdots & X_1\nu\Sigma_{f,h} & \cdots & X_1\nu\Sigma_{f,G} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ X_h\nu\Sigma_{f,1} & \cdots & X_h\nu\Sigma_{f,h} & \cdots & X_h\nu\Sigma_{f,G} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ X_G\nu\Sigma_{f,1} & \cdots & X_G\nu\Sigma_{f,h} & \cdots & X_G\nu\Sigma_{f,G} \end{bmatrix} \tag{9}$$

$$Flux\ matrix\ [\Phi] = \begin{bmatrix} \Phi_1 \\ \vdots \\ \Phi_h \\ \vdots \\ \Phi_G \end{bmatrix} \tag{10}$$

Next, using equation (1), the following compact form is obtained. Matrix operations are used to further simplify the compact form into an eigenvalue-eigenvector form:

$$[A][\Phi] + [S_{out}][\Phi] = \frac{1}{k}[F][\Phi] + [S_{in}][\Phi] \tag{11}$$

$$\left([A] + [S_{out}] - [S_{in}] \right)[\Phi] = \frac{1}{k}[F][\Phi]$$
$$Migration\ matrix\ [M] = \left([A] + [S_{out}] - [S_{in}] \right)$$
$$Problem\ matrix\ [P] = [M]^{-1}[F]$$
$$k[\Phi] = [M]^{-1}[F][\Phi] = [P][\Phi] \tag{12}$$

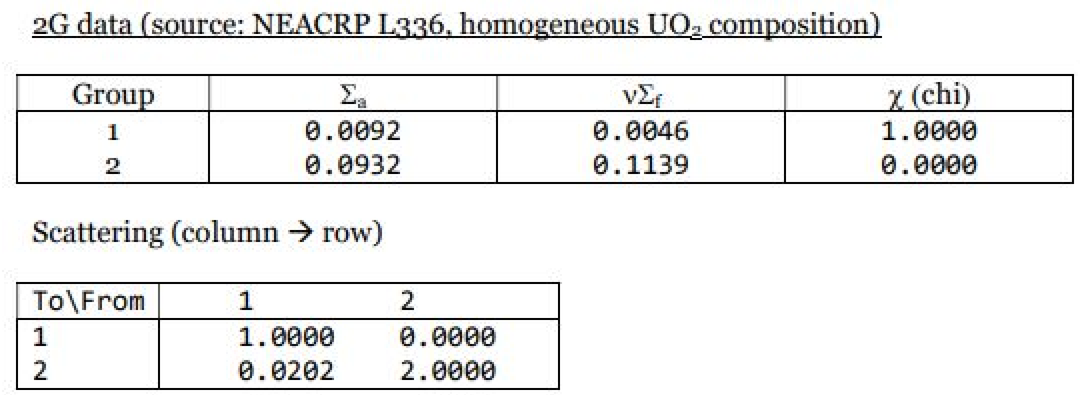
Next, to be used in problems 1 and 2, the Power Iteration method is shown. Its final iteration (denoted by i) returns the dominant eigenvalue-eigenvector pair.

$$\Phi_{i+1} = \frac{[P][\Phi_i]}{\|[P][\Phi_i]\|_2} \tag{13}$$

$$k_{i+1} = \frac{\left([P][\Phi_{i+1}] \right)^T [\Phi_{i+1}]}{[\Phi_{i+1}]^T [\Phi_{i+1}]} \tag{14}$$

2G Eigenvalue-Eigenvector Problem

Both a hand-solution and Power Iteration method are applied to find the eigenvalues and eigenvectors for a 2-group neutron balance problem using the following data:



2G data (source: NEACRP L336, homogeneous UO_2 composition)

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

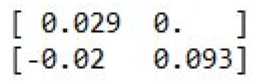
Scattering (column → row)

To\From	1	2
1	1.0000	0.0000
2	0.0202	2.0000

Figure 1

Given data to be applied to 2G matrices

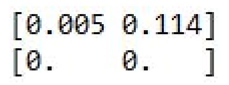
Part 1: Migration and Fission matrices



```
[ 0.029  0.   ]
[-0.02   0.093]
```

Figure 2

Migration Matrix



```
[0.005 0.114]
[0.    0.   ]
```

Figure 3

Fission Matrix

Part 2: Hand-calculated Eigenvalues and Eigenvectors

After applying the data from Figure 1 to Equations (6), (7), (8), (9), (11), and (12), the following form is obtained, with variables a, b, c, and d written to avoid unnecessary mess. The eigenvalues can be solved for as such:

$$k\Phi = P\Phi$$

$$P = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$det(P - kI) = (a - k)(d - k) - (bc) = k^2 - (a + d)k + ad - bc = 0$$

$$k = \frac{(a + d) \pm \sqrt{(a + d)^2 - 4(ad - bc)}}{2}$$

$$k = x,\ k = y \tag{15}$$

After finding the eigenvalues in Equation (15), the following methods, along with basic algebra, are applied to solve for their corresponding eigenvectors in Equations (16) and (17). Normalization is also used in order to match the numpy outputs obtained from the code.

$$x\begin{bmatrix} \Phi_1 \\ \Phi_2 \end{bmatrix}_x = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} \Phi_1 \\ \Phi_2 \end{bmatrix}_x$$

$$(a - x)\Phi_1 + b\Phi_2 = 0$$

$$c\Phi_1 + (d - x)\Phi_2 = 0$$

$$\frac{\begin{bmatrix} \Phi_1 \\ \Phi_2 \end{bmatrix}_x}{\left\| \begin{matrix} \Phi_1 \\ \Phi_2 \end{matrix} \right\|_x} = \begin{bmatrix} \Phi_1 \\ \Phi_2 \end{bmatrix}_{x,\, numpy} \tag{16}$$

$$y\begin{bmatrix}\Phi_1\\ \Phi_2\end{bmatrix}_y = \begin{bmatrix}a & b\\ c & d\end{bmatrix}\begin{bmatrix}\Phi_1\\ \Phi_2\end{bmatrix}_y$$

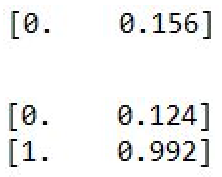
$$(a-y)\,\Phi_1 + b\Phi_2 = 0$$

$$c\Phi_1 + (d-y)\,\Phi_2 = 0$$

$$\frac{\begin{bmatrix}\Phi_1\\ \Phi_2\end{bmatrix}_y}{\left\|\begin{matrix}\Phi_1\\ \Phi_2\end{matrix}\right\|_y} = \begin{bmatrix}\Phi_1\\ \Phi_2\end{bmatrix}_{y,\,numpy} \tag{17}$$

Plugging in the values ends up with the exact same numbers as calculated by numpy in the following part. This indicates that the hand-method is correct.

Part 3: Numpy calculated Eigenvalues and Eigenvectors



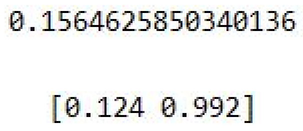
```
[0.    0.156]

[0.    0.124]
[1.    0.992]
```

Figure 4

Eigenvalues (top) and corresponding Eigenvectors (bottom) found with numpy

Part 4: Power Iteration calculated Dominant Eigenvalues and Eigenvectors



```
0.1564625850340136

[0.124 0.992]
```

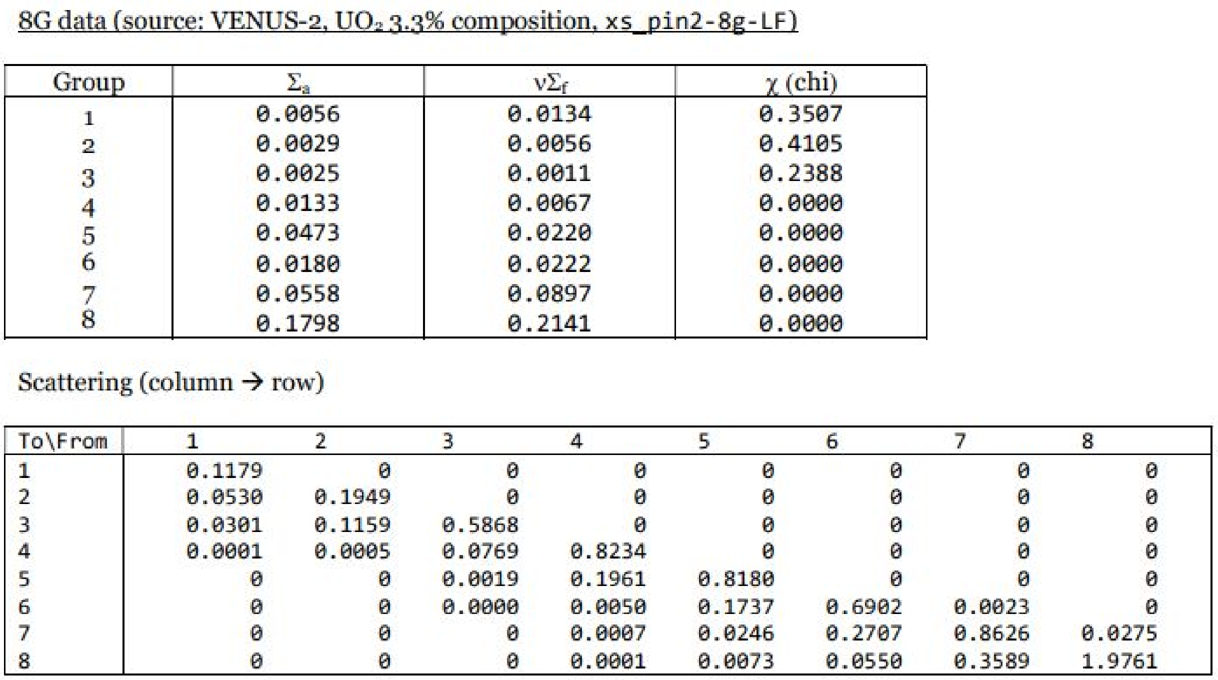
Figure 5

Dominant Eigenvalue (top) and Dominant Eigenvector (bottom) found with Power Iteration

Since the eigenvalue-eigenvector pair in Figure 5 matches the dominant eigenvalue-eigenvector pair in Figure 4 (column 2), the Power Iteration method was successful.

8G Eigenvalue-Eigenvector Problem

A Power Iteration method is applied to find the eigenvalues and eigenvectors for an 8-group neutron balance problem using the following data:



8G data (source: VENUS-2, UO_2 3.3% composition, xs_pin2-8g-LF)

Group	Σ_a	$\nu\Sigma_f$	χ (chi)
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

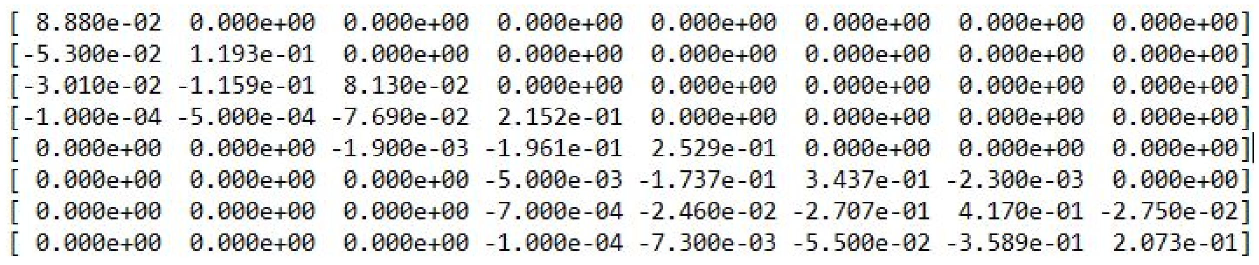
Scattering (column → row)

To\From	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	0.0000	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

Figure 6

Given data to be applied to 8G matrices

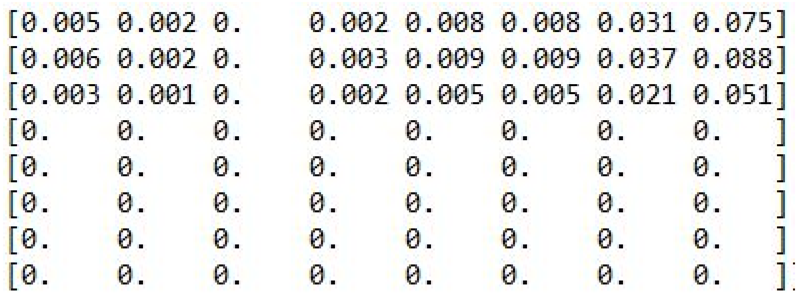
Part 1: Migration and Fission matrices



```
[ 8.880e-02  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00]
[-5.300e-02  1.193e-01  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00]
[-3.010e-02 -1.159e-01  8.130e-02  0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00]
[-1.000e-04 -5.000e-04 -7.690e-02  2.152e-01  0.000e+00  0.000e+00  0.000e+00  0.000e+00]
[ 0.000e+00  0.000e+00 -1.900e-03 -1.961e-01  2.529e-01  0.000e+00  0.000e+00  0.000e+00]
[ 0.000e+00  0.000e+00  0.000e+00 -5.000e-03 -1.737e-01  3.437e-01 -2.300e-03  0.000e+00]
[ 0.000e+00  0.000e+00  0.000e+00 -7.000e-04 -2.460e-02 -2.707e-01  4.170e-01 -2.750e-02]
[ 0.000e+00  0.000e+00  0.000e+00 -1.000e-04 -7.300e-03 -5.500e-02 -3.589e-01  2.073e-01]
```

Figure 7

Migration Matrix

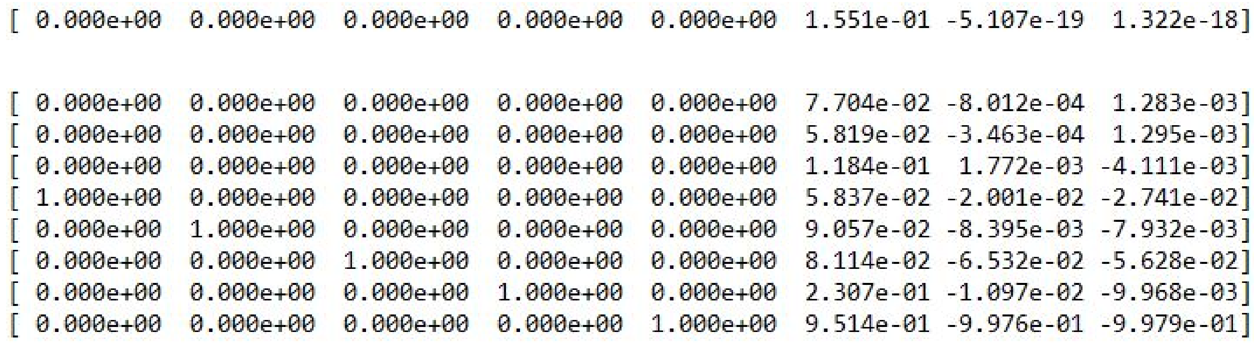


```
[0.005 0.002 0.    0.002 0.008 0.008 0.031 0.075]
[0.006 0.002 0.    0.003 0.009 0.009 0.037 0.088]
[0.003 0.001 0.    0.002 0.005 0.005 0.021 0.051]
[0.    0.    0.    0.    0.    0.    0.    0.   ]
[0.    0.    0.    0.    0.    0.    0.    0.   ]
[0.    0.    0.    0.    0.    0.    0.    0.   ]
[0.    0.    0.    0.    0.    0.    0.    0.   ]
[0.    0.    0.    0.    0.    0.    0.    0.   ]
```

Figure 8

Fission Matrix

Part 3: Numpy calculated Eigenvalues and Eigenvectors



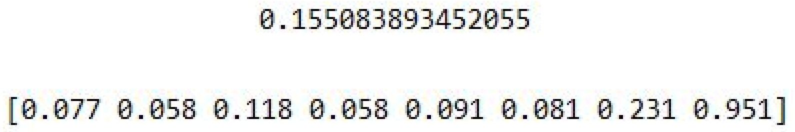
```
[ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  1.551e-01 -5.107e-19  1.322e-18]

[ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  7.704e-02 -8.012e-04  1.283e-03]
[ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  5.819e-02 -3.463e-04  1.295e-03]
[ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  1.184e-01  1.772e-03 -4.111e-03]
[ 1.000e+00  0.000e+00  0.000e+00  0.000e+00  0.000e+00  5.837e-02 -2.001e-02 -2.741e-02]
[ 0.000e+00  1.000e+00  0.000e+00  0.000e+00  0.000e+00  9.057e-02 -8.395e-03 -7.932e-03]
[ 0.000e+00  0.000e+00  1.000e+00  0.000e+00  0.000e+00  8.114e-02 -6.532e-02 -5.628e-02]
[ 0.000e+00  0.000e+00  0.000e+00  1.000e+00  0.000e+00  2.307e-01 -1.097e-02 -9.968e-03]
[ 0.000e+00  0.000e+00  0.000e+00  0.000e+00  1.000e+00  9.514e-01 -9.976e-01 -9.979e-01]
```

Figure 9

Eigenvalues (top) and corresponding Eigenvectors (bottom) found with numpy

Part 4: Power Iteration calculated Dominant Eigenvalues and Eigenvectors



```
0.155083893452055

[0.077 0.058 0.118 0.058 0.091 0.081 0.231 0.951]
```

Figure 10

Dominant Eigenvalue (top) and Dominant Eigenvector (bottom) found with Power Iteration

Since the eigenvalue-eigenvector pair in Figure 10 matches the dominant eigenvalue-eigenvector pair in Figure 9 (column 6), the Power Iteration method was successful.

Appendix

Code, input, and output are all located at https://github.com/ahliu3/Neutron-Balance