NE 255 - Homework 5

University of California, Berkeley
Department of Nuclear Engineering

Daniel Hellfeld
dhellfeld@berkeley.edu

Problem 1
With the operator form of the Transport Equations:

$$\mathbf{L}\psi = \mathbf{MS}\phi + \mathbf{M}q_e \tag{1}$$

$$\phi = \mathbf{D}\psi \tag{2}$$

and given the following discretizations:

- 3 groups
- P_2 (number of moments is $(N+1)^2$)
- S_2 (number of angles is $N(N+2)$, with the N being from S_N rather than P_N above)
- $4 \times 4 \times 4$ mesh
- Diamond Difference

(a) Indicate the dimensions of each matrix in Eq. (1), using real numbers for what we did generically in class.

$\Rightarrow$ The fully discretized transport equation (with implicit spatial discretization in $\vec{r}$) is given by

$$\hat{\Omega}_a \cdot \nabla \psi_a^g(\vec{r}) + \Sigma_t^g(\vec{r})\psi_a^g(\vec{r}) = \sum_{g'=1}^{G}\sum_{l=0}^{N}\Sigma_{sl}^{g'\rightarrow g}(\vec{r})\Big[Y_{l0}^e(\hat{\Omega}_a)\phi_{l0}^{g'}(\vec{r}) + \sum_{m=1}^{l}(Y_{lm}^e(\hat{\Omega}_a)\phi_{lm}^{g'}(\vec{r}) + Y_{lm}^o(\hat{\Omega}_a)\vartheta_{lm}^{g'}(\vec{r}))\Big] + q_{a,e}^g(\vec{r})\,,$$

$$\phi_{lm}^g = \int_{4\pi} Y_{lm}^e(\hat{\Omega})\psi^g(\hat{\Omega})d\hat{\Omega}\,, \quad m \geq 0\,,$$

$$\vartheta_{lm}^g = \int_{4\pi} Y_{lm}^o(\hat{\Omega})\psi^g(\hat{\Omega})d\hat{\Omega}\,, \quad m > 0\,,$$

where ψ_a^g is the angular flux discretized in angle (a), energy (g), and implicitly in space ($\vec{r}$), $\hat{\Omega}_a$ is the unit vector along angle a, $\Sigma_t^g(\vec{r})$ is the total cross-section for group g, Y_{lm}^e and Y_{lm}^o are the even and odd spherical harmonics, respectively, ϕ_{lm}^g and ϑ_{lm}^g are the even and odd flux moments in group g, respectively, and $q_{a,e}^g(\vec{r})$ is the external source in group g and angle a. To simplify these equations, we can put them into operator form as shown in Eq. (1-2) above, where $\mathbf{L}$ is the transport operator ($= \hat{\Omega}_a \cdot \nabla + \Sigma_t^g(\vec{r})$), $\mathbf{M}$ converts the harmoic moments into discrete angles, $\mathbf{S}$ is the scattering matrix, $\mathbf{D}$ is the discrete-to-moment operator ($= \mathbf{M^T W} = \sum_{a=1}^{n} Y_{lm}^{e/o} w_a$), q_e contains the external source, and ϕ are the flux moments.

Now, we first adopt the following terminology

$$\begin{aligned}
G &= \#\text{ of groups} = 3\\
N &= \#\text{ of moments} = (2+1)^2 = 9\\
n &= \#\text{ of angular unknowns} = 2(2+2) = 8\\
c &= \#\text{ of spatial cells} = 4\times 4\times 4 = 64\\
u &= \#\text{ of unknowns per cell} = 1\text{ (Unknown in 3D Diamond Difference} \rightarrow \text{center flux, } \psi_{ijk})\\
\alpha &= G\times n\times c\times u = 3\times 8\times 64\times 1 = 1536\\
f &= G\times N\times c\times u = 3\times 9\times 64\times 1 = 1728
\end{aligned}$$

We begin by noting that ψ is a column vector containing the angular flux values for all groups (G), at all angles (n), in all spatial cells (c), and all the solutions we wish to solve for (u). Therefore its size is $\alpha \times 1$. The transport

operator, $\mathbf{L}$ gets applied to the angular flux column vector and we want the operation to return a vector of the same size as ψ, therefore it will have a size of $\alpha \times \alpha$. The flux moments will be similar to the angular flux, but instead of angular moments, we have expansion moments. Thus ϕ will be a column vector containing the flux values for all groups (G), for all expansion moments (N), in all spatial cells (c) and all solution points (u). Therefore its size is $f \times 1$. Like the transport operator on ψ, the scattering matrix $\mathbf{S}$ is applied to the flux moments and should return a vector of the same size as ϕ. Therefore the size of $\mathbf{S}$ is $f \times f$. Now the left hand side of the Eq. (1) has a size of $\alpha \times 1$, and thus the right side should as well. So to get the $\mathbf{MS}\phi$ term to have a size of $\alpha \times 1$, the size of $\mathbf{M}$ must be $\alpha \times f$. With the same idea, if $\mathbf{M}$ has a size of $\alpha \times f$, then the size of q_e must be $f \times 1$ to get a column vector of size $\alpha \times 1$. Finally, for Eq. (2), if we know that ϕ has size $f \times 1$ and ψ has size $\alpha \times 1$, thus in order for the left and right sides to equal, $\mathbf{D}$ must have a size of $f \times \alpha$. Putting this all together, we can write (where again $\alpha = 1536$ and $f = 1728$)

$$\overbrace{\underbrace{\mathbf{L}}_{(\alpha\times\alpha)}\underbrace{\psi}_{(\alpha\times 1)}}^{(\alpha\times 1)} = \overbrace{\underbrace{\mathbf{M}}_{(\alpha\times f)}\underbrace{\mathbf{S}}_{(f\times f)}\underbrace{\phi}_{(f\times 1)}}^{(\alpha\times 1)} + \overbrace{\underbrace{\mathbf{M}}_{(\alpha\times f)}\underbrace{q_e}_{(f\times 1)}}^{(\alpha\times 1)}$$

$$\underbrace{\phi}_{(f\times 1)} = \overbrace{\underbrace{\mathbf{D}}_{(f\times\alpha)}\underbrace{\psi}_{(\alpha\times 1)}}^{(f\times 1)}$$

(b) Write out the matrices $[\mathbf{M}]_{gg}$, $\mathbf{S}$, and $[\mathbf{S}]_{21}$ as well as the vectors ψ, $[\psi]_1$, and $[\phi]_1$ to make sure you know what values match with what.

$\Rightarrow$ Let's begin with the flux vectors. The angular flux will again be a set of values for all groups, angles, cells, and unknowns. We can first write

$$\psi = \begin{bmatrix}[\psi]_1 & [\psi]_2 & \dots & [\psi]_g & \dots & [\psi]_G\end{bmatrix}^T,$$

where $[\psi]_g$ represents the column vector for all angles, cells, and unknowns within group g

$$[\psi]_g = \begin{bmatrix}\psi_1^g & \psi_2^g & \dots & \psi_a^g & \dots & \psi_n^g\end{bmatrix}^T,$$

where ψ_a^g is again another column vector containing all the values in all cells and unknowns (length $= c \times u = 64 \times 1 = 64$). We can write ψ and $[\psi]_1$ as an example using the values at hand (remembering that each ψ_a^g term includes values for all cells and unknowns)

$$\psi = \begin{bmatrix}[\psi]_1 & [\psi]_2 & [\psi]_3\end{bmatrix}^T,$$

$$[\psi]_1 = \begin{bmatrix}\psi_1^1 & \psi_2^1 & \psi_3^1 & \psi_4^1 & \psi_5^1 & \psi_6^1 & \psi_7^1 & \psi_8^1\end{bmatrix}^T.$$

We see that the size of ψ will indeed be $Gncu \times 1$ or $\alpha \times 1$. Similarly for the flux moments, we can write

$$\phi = \begin{bmatrix}[\phi]_1 & [\phi]_2 & \dots & [\phi]_g & \dots & [\phi]_G\end{bmatrix}^T,$$

where $[\phi]_g$ represents the column vector for all $P_{N'}$ expansions, cells, and unknowns within group g

$$[\phi]_g = \begin{bmatrix}\phi_{00}^g & \phi_{10}^g & \vartheta_{11}^g & \phi_{11}^g & \phi_{20}^g & \vartheta_{21}^g & \phi_{21}^g & \dots & \vartheta_{N'N'}^g & \phi_{N'N'}^g\end{bmatrix}^T,$$

where $(\phi/\vartheta)_{lm}^g$ are again column vectors containing all the values in all cells and unknowns (length $= c \times u = 64 \times 1 = 64$). As an example, using the values at hand, we can write

$$[\phi]_1 = \begin{bmatrix}\phi_{00}^1 & \phi_{10}^1 & \vartheta_{11}^1 & \phi_{11}^1 & \phi_{20}^1 & \vartheta_{21}^1 & \phi_{21}^1 & \vartheta_{22}^1 & \phi_{22}^1\end{bmatrix}^T.$$

Again going through the math, we see that ϕ will indeed have a size of $GNcu \times 1$ or $f \times 1$.

Now, the $\mathbf{M}$ matrix can be written as

$$\mathbf{M} = \begin{bmatrix} [\mathbf{M}]_{11} & 0 & \dots & 0 \\ 0 & [\mathbf{M}]_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & [\mathbf{M}]_{GG} \end{bmatrix} = \begin{bmatrix} [\mathbf{M}] & 0 & \dots & 0 \\ 0 & [\mathbf{M}] & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & [\mathbf{M}] \end{bmatrix},$$

and $[\mathbf{M}]_{gg} = [\mathbf{M}]$ because it does not actually depend on the group. Also note that the zeros in the matrix represent matrices of zeros with the same size of $[\mathbf{M}]$. Now, $[\mathbf{M}]$ is given by

$$[\mathbf{M}] = \begin{bmatrix} Y_{00}^e(\hat{\Omega}_1) & Y_{10}^e(\hat{\Omega}_1) & Y_{11}^o(\hat{\Omega}_1) & Y_{11}^e(\hat{\Omega}_1) & Y_{20}^e(\hat{\Omega}_1) & \dots & Y_{N'N'}^o(\hat{\Omega}_1) & Y_{N'N'}^e(\hat{\Omega}_1) \\ Y_{00}^e(\hat{\Omega}_2) & Y_{10}^e(\hat{\Omega}_2) & Y_{11}^o(\hat{\Omega}_2) & Y_{11}^e(\hat{\Omega}_2) & Y_{20}^e(\hat{\Omega}_2) & \dots & Y_{N'N'}^o(\hat{\Omega}_2) & Y_{N'N'}^e(\hat{\Omega}_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ Y_{00}^e(\hat{\Omega}_n) & Y_{10}^e(\hat{\Omega}_n) & Y_{11}^o(\hat{\Omega}_n) & Y_{11}^e(\hat{\Omega}_n) & Y_{20}^e(\hat{\Omega}_n) & \dots & Y_{N'N'}^o(\hat{\Omega}_n) & Y_{N'N'}^e(\hat{\Omega}_n) \end{bmatrix},$$

where N' is equal to the order of the $P_{N'}$ expansion, n is the total number of angles, and each term in the matrix is a constant square matrix of size $cu \times cu$ (to account for the spatial cells and unknowns). For example

$$Y_{00}^e(\hat{\Omega}_1) = \begin{bmatrix} Y_{00}^e(\hat{\Omega}_1) & Y_{00}^e(\hat{\Omega}_1) & \dots & Y_{00}^e(\hat{\Omega}_1) \\ Y_{00}^e(\hat{\Omega}_1) & Y_{00}^e(\hat{\Omega}_1) & \dots & Y_{00}^e(\hat{\Omega}_1) \\ \vdots & \vdots & \ddots & \vdots \\ Y_{00}^e(\hat{\Omega}_1) & Y_{00}^e(\hat{\Omega}_1) & \dots & Y_{00}^e(\hat{\Omega}_1) \end{bmatrix}.$$

Putting it all together for $N' = 2$ and $n = 8$, we get

$$[\mathbf{M}] = \begin{bmatrix} Y_{00}^e(\hat{\Omega}_1) & Y_{10}^e(\hat{\Omega}_1) & Y_{11}^o(\hat{\Omega}_1) & Y_{11}^e(\hat{\Omega}_1) & Y_{20}^e(\hat{\Omega}_1) & Y_{21}^o(\hat{\Omega}_1) & Y_{21}^e(\hat{\Omega}_1) & Y_{22}^o(\hat{\Omega}_1) & Y_{22}^e(\hat{\Omega}_1) \\ Y_{00}^e(\hat{\Omega}_2) & Y_{10}^e(\hat{\Omega}_2) & Y_{11}^o(\hat{\Omega}_2) & Y_{11}^e(\hat{\Omega}_2) & Y_{20}^e(\hat{\Omega}_2) & Y_{21}^o(\hat{\Omega}_2) & Y_{21}^e(\hat{\Omega}_2) & Y_{22}^o(\hat{\Omega}_2) & Y_{22}^e(\hat{\Omega}_2) \\ Y_{00}^e(\hat{\Omega}_3) & Y_{10}^e(\hat{\Omega}_3) & Y_{11}^o(\hat{\Omega}_3) & Y_{11}^e(\hat{\Omega}_3) & Y_{20}^e(\hat{\Omega}_3) & Y_{21}^o(\hat{\Omega}_3) & Y_{21}^e(\hat{\Omega}_3) & Y_{22}^o(\hat{\Omega}_3) & Y_{22}^e(\hat{\Omega}_3) \\ Y_{00}^e(\hat{\Omega}_4) & Y_{10}^e(\hat{\Omega}_4) & Y_{11}^o(\hat{\Omega}_4) & Y_{11}^e(\hat{\Omega}_4) & Y_{20}^e(\hat{\Omega}_4) & Y_{21}^o(\hat{\Omega}_4) & Y_{21}^e(\hat{\Omega}_4) & Y_{22}^o(\hat{\Omega}_4) & Y_{22}^e(\hat{\Omega}_4) \\ Y_{00}^e(\hat{\Omega}_5) & Y_{10}^e(\hat{\Omega}_5) & Y_{11}^o(\hat{\Omega}_5) & Y_{11}^e(\hat{\Omega}_5) & Y_{20}^e(\hat{\Omega}_5) & Y_{21}^o(\hat{\Omega}_5) & Y_{21}^e(\hat{\Omega}_5) & Y_{22}^o(\hat{\Omega}_5) & Y_{22}^e(\hat{\Omega}_5) \\ Y_{00}^e(\hat{\Omega}_6) & Y_{10}^e(\hat{\Omega}_6) & Y_{11}^o(\hat{\Omega}_6) & Y_{11}^e(\hat{\Omega}_6) & Y_{20}^e(\hat{\Omega}_6) & Y_{21}^o(\hat{\Omega}_6) & Y_{21}^e(\hat{\Omega}_6) & Y_{22}^o(\hat{\Omega}_6) & Y_{22}^e(\hat{\Omega}_6) \\ Y_{00}^e(\hat{\Omega}_7) & Y_{10}^e(\hat{\Omega}_7) & Y_{11}^o(\hat{\Omega}_7) & Y_{11}^e(\hat{\Omega}_7) & Y_{20}^e(\hat{\Omega}_7) & Y_{21}^o(\hat{\Omega}_7) & Y_{21}^e(\hat{\Omega}_7) & Y_{22}^o(\hat{\Omega}_7) & Y_{22}^e(\hat{\Omega}_7) \\ Y_{00}^e(\hat{\Omega}_8) & Y_{10}^e(\hat{\Omega}_8) & Y_{11}^o(\hat{\Omega}_8) & Y_{11}^e(\hat{\Omega}_8) & Y_{20}^e(\hat{\Omega}_8) & Y_{21}^o(\hat{\Omega}_8) & Y_{21}^e(\hat{\Omega}_8) & Y_{22}^o(\hat{\Omega}_8) & Y_{22}^e(\hat{\Omega}_8) \end{bmatrix}.$$

And since we only have 3 energy groups, we can write $\mathbf{M}$ as

$$\mathbf{M} = \begin{bmatrix} [\mathbf{M}] & 0 & 0 \\ 0 & [\mathbf{M}] & 0 \\ 0 & 0 & [\mathbf{M}] \end{bmatrix}.$$

Each matrix in the $[\mathbf{M}]$ matrix has size $cu \times cu$, $[\mathbf{M}]$ has size $Ncu \times ncu$, and thus $\mathbf{M}$ has size $GNcu \times Gncu$ or $f \times \alpha$ (as noted above). Now, the $\mathbf{S}$ matrix is given by

$$\mathbf{S} = \begin{bmatrix} [\mathbf{S}]_{11} & [\mathbf{S}]_{12} & \dots & [\mathbf{S}]_{1G} \\ [\mathbf{S}]_{21} & [\mathbf{S}]_{22} & \dots & [\mathbf{S}]_{2G} \\ \vdots & \vdots & \ddots & \vdots \\ [\mathbf{S}]_{G1} & [\mathbf{S}]_{G2} & \dots & [\mathbf{S}]_{GG} \end{bmatrix},$$

where $[\mathbf{S}]_{gg'}$ represents scattering from group g' to group g. In this problem, $G = 3$, so we have

$$\mathbf{S} = \begin{bmatrix} [\mathbf{S}]_{11} & [\mathbf{S}]_{12} & [\mathbf{S}]_{13} \\ [\mathbf{S}]_{21} & [\mathbf{S}]_{22} & [\mathbf{S}]_{23} \\ [\mathbf{S}]_{31} & [\mathbf{S}]_{32} & [\mathbf{S}]_{33} \end{bmatrix}.$$

Now, each $[\mathbf{S}]_{gg'}$ is given by

$$[\mathbf{S}]_{gg'} = \begin{bmatrix} \Sigma_{s0}^{g'\to g} & 0 & \dots & 0 \\ 0 & \Sigma_{s1}^{g'\to g} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \Sigma_{sN}^{g'\to g} \end{bmatrix}.$$

As an example, we can show $[\mathbf{S}]_{21}$. First remember that $\mathbf{S}$ is a square matrix, and thus each $[\mathbf{S}]_{gg'}$ must also be a square matrix. Now referring back to the fully discretized form of the transport equation, we see that the scattering cross-section for a given group-to-group scatter ($g' \to g$) and P_N expansion (l) is multiplied by the spherical harmonics for all possible values of m given l ($m = 0, 1, 2, \dots, l$). Note that when $m \geq 1$ that it is multiplied by both the even and odd harmonics. Therefore, $[\mathbf{S}]_{gg'}$ should be written a bit more explicitly as

$$[\mathbf{S}]_{gg'} = \begin{bmatrix} \Sigma_{s0,m=0}^{g'\to g} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & \Sigma_{s1,m=0}^{g'\to g} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & \Sigma_{s1,m=1}^{g'\to g} & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & \Sigma_{s1,m=1}^{g'\to g} & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \Sigma_{s2,m=0}^{g'\to g} & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & \Sigma_{s2,m=1}^{g'\to g} & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \Sigma_{s2,m=1}^{g'\to g} & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Sigma_{s2,m=2}^{g'\to g} & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Sigma_{s2,m=2}^{g'\to g} & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

and like for the $[\mathbf{M}]$ matrix, each entry in this matrix is a constant square matrix of size $cu \times cu$ to account for all spatial cells and unknowns. Also, remember the scattering cross-section does not depend on m ($\Sigma_{sl,m}^{g'\to g} = \Sigma_{sl}^{g'\to g}$). Now for this problem, we are given the P_2 expansion, and thus the $[\mathbf{S}]_{21}$ matrix will be given as

$$[\mathbf{S}]_{21} = \begin{bmatrix} \Sigma_{s0}^{2\to 1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \Sigma_{s1}^{2\to 1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \Sigma_{s1}^{2\to 1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \Sigma_{s1}^{2\to 1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \Sigma_{s2}^{2\to 1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \Sigma_{s2}^{2\to 1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \Sigma_{s2}^{2\to 1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Sigma_{s2}^{2\to 1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Sigma_{s2}^{2\to 1} \end{bmatrix}.$$

Each element is this matrix has size $cu \times cu$ so the size of $[\mathbf{S}]_{gg'}$ is $Ncu \times Ncu$ and thus the size of $\mathbf{S}$ is $GNcu \times GNcu$ or $f \times f$ (which agrees with what was stated above).

(c) Write what the $\mathbf{D}$ matrix would be.

$\Rightarrow$ As stated above, the $\mathbf{D}$ matrix is the discrete-to-moment operator and is given by $\mathbf{M^T W}$ where $\mathbf{W}$ is a diagonal matrix of diagonal matrices of quadrature weights ($\mathbf{D} = \mathbf{M^T W} = \sum_{a=1}^{n} Y_{lm}^{e/o} w_a$). In general we can write

$$\mathbf{D} = \mathbf{M}^T\mathbf{W} = \begin{bmatrix} [\mathbf{M}]_{11} & 0 & \dots & 0 \\ 0 & [\mathbf{M}]_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & [\mathbf{M}]_{GG} \end{bmatrix}^T \begin{bmatrix} [\mathbf{W}]_{11} & 0 & \dots & 0 \\ 0 & [\mathbf{W}]_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & [\mathbf{W}]_{GG} \end{bmatrix},$$

and dropping the group dependencies

$$\mathbf{D} = \mathbf{M}^T\mathbf{W} = \begin{bmatrix} [\mathbf{M}] & 0 & \dots & 0 \\ 0 & [\mathbf{M}] & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & [\mathbf{M}] \end{bmatrix}^T \begin{bmatrix} [\mathbf{W}] & 0 & \dots & 0 \\ 0 & [\mathbf{W}] & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & [\mathbf{W}] \end{bmatrix},$$

where each $[\mathbf{W}]$ is given by

$$[\mathbf{W}] = \begin{bmatrix} w_1 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \dots & \vdots \\ 0 & \dots & w_a & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & w_n \end{bmatrix},$$

where again each element is another matrix to account for the values for all spatial cells and unknowns (size $cu \times cu$). Therefore the size of $[\mathbf{W}]$ is $ncu \times ncu$ and the size of $\mathbf{W}$ is $Gncu \times Gncu$ or $\alpha \times \alpha$. In our case, $[\mathbf{W}]$ becomes

$$[\mathbf{W}] = \begin{bmatrix} w_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & w_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & w_7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_8 \end{bmatrix},$$

and $\mathbf{D}$ becomes

$$\begin{aligned} \mathbf{D} &= \begin{bmatrix} [\mathbf{M}] & 0 & 0 \\ 0 & [\mathbf{M}] & 0 \\ 0 & 0 & [\mathbf{M}] \end{bmatrix}^T \begin{bmatrix} [\mathbf{W}] & 0 & 0 \\ 0 & [\mathbf{W}] & 0 \\ 0 & 0 & [\mathbf{W}] \end{bmatrix}, \\ &= \begin{bmatrix} \square & 0 & 0 \\ 0 & \square & 0 \\ 0 & 0 & \square \end{bmatrix}, \end{aligned}$$

where $\square$ is defined as

$$\square = \begin{bmatrix} w_1 Y_{00}^e(\hat{\Omega}_1) & w_2 Y_{00}^e(\hat{\Omega}_2) & w_3 Y_{00}^o(\hat{\Omega}_3) & w_4 Y_{00}^e(\hat{\Omega}_4) & w_5 Y_{00}^e(\hat{\Omega}_5) & w_6 Y_{00}^o(\hat{\Omega}_6) & w_7 Y_{00}^e(\hat{\Omega}_7) & w_8 Y_{00}^o(\hat{\Omega}_8) \\ w_1 Y_{10}^e(\hat{\Omega}_1) & w_2 Y_{10}^e(\hat{\Omega}_2) & w_3 Y_{10}^o(\hat{\Omega}_3) & w_4 Y_{10}^e(\hat{\Omega}_4) & w_5 Y_{10}^e(\hat{\Omega}_5) & w_6 Y_{10}^o(\hat{\Omega}_6) & w_7 Y_{10}^e(\hat{\Omega}_7) & w_8 Y_{10}^o(\hat{\Omega}_8) \\ w_1 Y_{11}^e(\hat{\Omega}_1) & w_2 Y_{11}^e(\hat{\Omega}_2) & w_3 Y_{11}^o(\hat{\Omega}_3) & w_4 Y_{11}^e(\hat{\Omega}_4) & w_5 Y_{11}^e(\hat{\Omega}_5) & w_6 Y_{11}^o(\hat{\Omega}_6) & w_7 Y_{11}^e(\hat{\Omega}_7) & w_8 Y_{11}^o(\hat{\Omega}_8) \\ w_1 Y_{11}^e(\hat{\Omega}_1) & w_2 Y_{11}^e(\hat{\Omega}_2) & w_3 Y_{11}^o(\hat{\Omega}_3) & w_4 Y_{11}^e(\hat{\Omega}_4) & w_5 Y_{11}^e(\hat{\Omega}_5) & w_6 Y_{11}^o(\hat{\Omega}_6) & w_7 Y_{11}^e(\hat{\Omega}_7) & w_8 Y_{11}^o(\hat{\Omega}_8) \\ w_1 Y_{20}^e(\hat{\Omega}_1) & w_2 Y_{20}^e(\hat{\Omega}_2) & w_3 Y_{20}^o(\hat{\Omega}_3) & w_4 Y_{20}^e(\hat{\Omega}_4) & w_5 Y_{20}^e(\hat{\Omega}_5) & w_6 Y_{20}^o(\hat{\Omega}_6) & w_7 Y_{20}^e(\hat{\Omega}_7) & w_8 Y_{20}^o(\hat{\Omega}_8) \\ w_1 Y_{21}^e(\hat{\Omega}_1) & w_2 Y_{21}^e(\hat{\Omega}_2) & w_3 Y_{21}^o(\hat{\Omega}_3) & w_4 Y_{21}^e(\hat{\Omega}_4) & w_5 Y_{21}^e(\hat{\Omega}_5) & w_6 Y_{21}^o(\hat{\Omega}_6) & w_7 Y_{21}^e(\hat{\Omega}_7) & w_8 Y_{21}^o(\hat{\Omega}_8) \\ w_1 Y_{21}^e(\hat{\Omega}_1) & w_2 Y_{21}^e(\hat{\Omega}_2) & w_3 Y_{21}^o(\hat{\Omega}_3) & w_4 Y_{21}^e(\hat{\Omega}_4) & w_5 Y_{21}^e(\hat{\Omega}_5) & w_6 Y_{21}^o(\hat{\Omega}_6) & w_7 Y_{21}^e(\hat{\Omega}_7) & w_8 Y_{21}^o(\hat{\Omega}_8) \\ w_1 Y_{22}^e(\hat{\Omega}_1) & w_2 Y_{22}^e(\hat{\Omega}_2) & w_3 Y_{22}^o(\hat{\Omega}_3) & w_4 Y_{22}^e(\hat{\Omega}_4) & w_5 Y_{22}^e(\hat{\Omega}_5) & w_6 Y_{22}^o(\hat{\Omega}_6) & w_7 Y_{22}^e(\hat{\Omega}_7) & w_8 Y_{22}^o(\hat{\Omega}_8) \\ w_1 Y_{22}^e(\hat{\Omega}_1) & w_2 Y_{22}^e(\hat{\Omega}_2) & w_3 Y_{22}^o(\hat{\Omega}_3) & w_4 Y_{22}^e(\hat{\Omega}_4) & w_5 Y_{22}^e(\hat{\Omega}_5) & w_6 Y_{22}^o(\hat{\Omega}_6) & w_7 Y_{22}^e(\hat{\Omega}_7) & w_8 Y_{22}^o(\hat{\Omega}_8) \end{bmatrix}.$$

Accounting for each element in $\square$ being a matrix of size $cu \times cu$, the $\mathbf{D}$ matrix has a size of $GNcu \times Gncu$ or $f \times \alpha$ (which agrees with what was stated above).

(d) Why don't we form an $\mathbf{L}$ matrix?

$\Rightarrow$ In the nuclear field, the $\mathbf{L}$ matrix is typically not formed because it will govern how we can implement our solvers (except with SP_N, which does/can form $\mathbf{L}$). Instead, we essentially find $\mathbf{L}^{-1}$ and use it implicitly to solve for ψ (this is what we are doing in our sweep). The $\mathbf{L}$ matrix is also very large in most cases (using fine discretizations and high order expansions) and will be nonuniformly sparse (as opposed to other large, but only diagonal, matrices). Therefore we significantly reduce the amount of unnecessary memory used in the calculation by not forming $\mathbf{L}$.

(e) Combine Eqs. (1) and (2) to get a system that looks like $\mathbf{A}x = b$, writing out the steps.

$\Rightarrow$ Starting with

$$\mathbf{L}\psi = \mathbf{MS}\phi + \mathbf{M}q_e \,,$$

we first invert $\mathbf{L}$ and multiply it to both sides of the equation

$$\begin{aligned} \mathbf{L}^{-1}\mathbf{L}\psi &= \mathbf{L}^{-1}\mathbf{MS}\phi + \mathbf{L}^{-1}\mathbf{M}q_e \,, \\ \psi &= \mathbf{L}^{-1}\mathbf{MS}\phi + \mathbf{L}^{-1}\mathbf{M}q_e \,. \end{aligned}$$

Now multiply by $\mathbf{D}$ on both sides

$$\mathbf{D}\psi = \mathbf{DL}^{-1}\mathbf{MS}\phi + \mathbf{DL}^{-1}\mathbf{M}q_e \,.$$

Substituting in Eq. (2) for the left hand side, and using $Q = \mathbf{DL}^{-1}\mathbf{M}q_e$, we can do

$$\begin{aligned} \phi &= \mathbf{DL}^{-1}\mathbf{MS}\phi + Q \,, \\ \phi - \mathbf{DL}^{-1}\mathbf{MS}\phi &= Q \,, \\ (\mathbf{I} - \mathbf{DL}^{-1}\mathbf{MS})\phi &= Q \,, \end{aligned}$$

where $\mathbf{I}$ is the identity matrix. Now we can substitute in $\mathbf{H} = \mathbf{I} - \mathbf{DL}^{-1}\mathbf{MS}$ to get

$$\mathbf{H}\phi = Q \,,$$

which looks like $\mathbf{A}x = b$.

Problem 2
Implement a Jacobi multigroup solver for the 1D, steady state transport equations with isotropic scattering and an isotropic external source. Use the weighted diamond difference solver you wrote for the previous homework to solve the within group equations (if you are unsure if yours worked let me know) (note: you functionally should have written source iteration). Use the following values and three energy groups:

- $x_0 = 0.0,\ x_1 = 2.0,\ h = 0.1$
- $\alpha = 0.5$
- $\mu_a = \pm[0.2, 0.5, 0.7]$
- $\Sigma_{t,1} = 0.5,\ \Sigma_{t,2} = 0.8,\ \Sigma_{t,3} = 1.0$
- $\Sigma_s^{g' \to g}$ values are given by Table 1.
- $q_{e,1} = 1.5,\ q_{e,2} = 0.0,\ q_{e,3} = 0.2$
- left boundary condition is 0.5 incoming in group 1, zero otherwise

g \ g'	1	2	3
1	0.1	0.0	0.0
2	0.3	0.1	0.1
3	0.1	0.3	0.3

Table 1. Scattering (from group g' to group g) cross-section values.

Plot the resulting scalar flux in each energy group as a function of x. Use a convergence tolerance for the multigroup iteration and the scattering iteration of at least 1×10^{-4}.

$\Rightarrow$ The code from the previous homework was used to perform the inner (within group) iterations for all angles, but now the scattering source includes the scattering from all groups into the current group

$$s_a^{g,(k)} = \sum_{g'=1}^{G} \sum_{\mu} w_a \Sigma_{sl}^{g' \to g}(x) \psi_\mu^{g'}(x) + q_a^g(x)\,.$$

Looking at Table 1, we see there is only in-group scattering in group 1 ($1 \to 1$). In group 2, we have downscattering from group 1 ($1 \to 2$), in-group scattering ($2 \to 2$), and upscattering from group 3 ($3 \to 2$). In group 3, we have downscattering from group 1 ($1 \to 3$), downscattering from group 2 ($2 \to 3$), and in-group scattering ($3 \to 3$). We begin in group 1, and iterate on the scattering source until convergence (l_2 norm of the difference of the current scalar flux and previous scalar flux). This is the same procedure as the previous homework. The Jacobi method is order independent, which means that the results of one within group iteration does not affect any other within group iteration (for a given outer iteration level). This means that we begin was some initial guesses for the fluxes in group 1, 2, and 3 and then perform within group iterations for all three groups with these initial guesses. Once we have done all three groups, we then update the group fluxes and redo the inner iterations. This is in contrast to Gauss-Seidel, where we do the inner iterations sequentially and update the group fluxes after each within group iteration.

To define outer iteration convergence, we sum the group scattering source for each group, in each cell,

$$\sum_{g=1}^{G} \left[\sum_{g'=1}^{G} \sum_{\mu} w_a \Sigma_{sl}^{g' \to g}(x) \psi_\mu^{g'}(x) + q_a^g(x) \right]\,.$$

and then compute the l_2 norm of the difference between the current iteration and the previous iteration. Since group 2 and 3 have no influence on group 1, it will not need more than one outer iteration. Since both group 2 and 3 have influence from other groups, we will need to perform multiple outer iterations until convergence.

The code is below and the results (center group scalar fluxes) are plotted in Fig. 1. We see that group 1 has the largest amplitude, followed by group 2 and then group 3. This agrees with intuition because there is a large

source into group 1 and the total cross-section is small compared to the other cells. In group 2, there is no external source and a larger total cross-section. There is a small source of particles into group 3 and particles from group 1 and 2 can scatter down into group 3, but we see that the large total cross section and ability to upscatter into 2 significantly limits the flux in group 3. The absolute amplitudes of the fluxes are somewhat questionable because the problem statement was unclear on how the boundary condition flux and the external source are distributed among angle and whether or not we include a factor of 1/2 in the calculation of the scalar flux. Also, note that group 1 indeed only needs 1 inner iteration following the first outer group iteration.

```
# Imports
import numpy as np
import matplotlib.pyplot as plt

# Within group iteration procedure (weighted diamond difference)
def InnerIteration(a, mu, sig_t, sig_s, q_e, N, psi_group1, psi_group2, psi_group3, group, converge):

    # Initialize new center fluxes arrays for the curret group
    psi_new = np.zeros((N,6))
    phi_new = np.zeros(N)

    # Calculate starting scalar flux (Sum 6 directions with equal weight)
    if   (group == 1): phi = (2./np.size(mu)) * psi_group1.sum(axis=1)
    elif (group == 2): phi = (2./np.size(mu)) * psi_group2.sum(axis=1)
    elif (group == 3): phi = (2./np.size(mu)) * psi_group3.sum(axis=1)

    # Start iterating
    innerconverge = False
    inneritr = 0
    while (innerconverge == False):

        # Boundary condition
        if (group == 0): psi_in = np.array([0.5,0.5,0.5])
        else           : psi_in = np.array([0.0,0.0,0.0])

        # Sweep in mu > 0 (right)
        for i in range(int(N)):
            # Calculate source
            # Weight is normalized to two, and there are 18 terms (6 directions in 3 groups)
            s = q_e[group-1] + (2./(np.size(mu)*np.size(q_e))) * ((sig_s[group-1,0]*psi_group1[i,:].sum()) +
                 (sig_s[group-1,1]*psi_group2[i,:].sum()) + (sig_s[group-1,2]*psi_group3[i,:].sum()))

            # Calculate center flux
            psi_new[i,0:3] = (s + (2. * np.fabs(mu[0:3]) / (d * (1.+a))) * psi_in) / (sig_t[group-1] + (2. *
                 np.fabs(mu[0:3]) / (d*(1.+a))))

            # Calcualte outgoing flux (set it equal to incoming flux for next cell)
            psi_in = ((2. / (1.+a)) * psi_new[i,0:3]) - ((1.-a)/(1.+a))*psi_in

        # Sweep in mu < 0 (left)
        for i in range(int(N)):
            s = q_e[group-1] + (2./(np.size(mu)*np.size(q_e))) * ((sig_s[group-1,0]*psi_group1[N-i-1,:].sum()) +
                 (sig_s[group-1,1]*psi_group2[N-i-1,:].sum()) + (sig_s[group-1,2]*psi_group3[N-i-1,:].sum()))

            psi_new[N-i-1,3:6] = (s + (2. * np.fabs(mu[3:6]) / (d * (1.+a))) * psi_in) / (sig_t[group-1] + (2. *
                 np.fabs(mu[3:6]) / (d*(1.+a))))

            psi_in = ((2. / (1.+a)) * psi_new[N-i-1,3:6]) - ((1.-a)/(1.+a))*psi_in

        # Calculate scalar flux from angular flux ((1/2)*SUM(w*psi))
        # Sum 6 directions with equal weight
        phi_new = (2./np.size(mu)) * psi_new.sum(axis=1)

        # Calculate convergence criterion (l2 norm of differences)
        innercrit = np.sqrt(np.sum((phi_new - phi)**2))

        # Check convergence
        if (innercrit < converge): innerconverge = True

        # Update anglular fluxes
        phi= np.copy(phi_new)
        if   (group == 1): psi_group1 = np.copy(psi_new)
        elif (group == 2): psi_group2 = np.copy(psi_new)
        elif (group == 3): psi_group3 = np.copy(psi_new)

        # Increment inner iteration number
        inneritr += 1

    # How many inner iterations did we do?
    print '(Group = %i) Number of iterations = %i' % (group,inneritr)
```

```
67
68      # Return angular and scalar flux for group
69      return psi_new, phi
70
71  # ------------------------------------------------
72
73
74  # Define variables
75  a     = 0.5
76  mu    = np.array([0.2,0.5,0.7,-0.2,-0.5,-0.7])
77  sig_t = np.array([0.5,0.8,1.0])
78  sig_s = np.array([[0.1,0.0,0.0],[0.3,0.1,0.1],[0.1,0.3,0.3]])
79  q_e   = np.array([1.5,0.0,0.2])
80  d     = 0.1
81  x0    = 0.0
82  x1    = 2.0
83  N     = (x1 - x0) / d
84  x     = np.linspace((d/2.), 2.-(d/2.),N) # center points for plotting
85
86  # Group fluxes (inital guesses)
87  psi_group1 = psi_group2 = psi_group3 = np.zeros((N,6))
88  phi_group1 = phi_group2 = phi_group3 = np.zeros(N)
89
90  # Choose solver method
91  #method = 'GaussSeidel'
92  method = 'Jacobi'
93
94  # Convergence value
95  converge = 1.0e-4
96
97  # Perform outer iteration over energy groups
98  outerconverge = False
99  outeritr = 0
100 while (outerconverge == False):
101
102     # Calculate current group source values
103     groupsource = np.zeros(N)
104     for group in range(3):
105         groupsource += q_e[group] + (2./(np.size(mu)*np.size(q_e))) * (sig_s[group,0]*psi_group1.sum(axis=1)) +
                (sig_s[group,1]*psi_group2.sum(axis=1)) + (sig_s[group,2]*psi_group3.sum(axis=1))
106
107     if (method == 'Jacobi'):
108         # Do within group iterations (all with inital guesses)
109         # Store in intermediate variable (as to not update between groups, which is Gauss-Seidel)
110         psi_group1_, phi_group1_ = InnerIteration(a, mu, sig_t, sig_s, q_e, N, psi_group1, psi_group2, psi_group3, 1,
                converge)
111         psi_group2_, phi_group2_ = InnerIteration(a, mu, sig_t, sig_s, q_e, N, psi_group1, psi_group2, psi_group3, 2,
                converge)
112         psi_group3_, phi_group3_ = InnerIteration(a, mu, sig_t, sig_s, q_e, N, psi_group1, psi_group2, psi_group3, 3,
                converge)
113
114         # Update fluxes
115         psi_group1 = np.copy(psi_group1_); phi_group1 = np.copy(phi_group1_)
116         psi_group2 = np.copy(psi_group2_); phi_group2 = np.copy(phi_group2_)
117         psi_group3 = np.copy(psi_group3_); phi_group3 = np.copy(phi_group3_)
118
119     elif (method == 'GaussSeidel'):
120         # Do within group iterations, updating after each group
121         psi_group1, phi_group1 = InnerIteration(a, mu, sig_t, sig_s, q_e, N, psi_group1, psi_group2, psi_group3, 1,
                converge)
122         psi_group2, phi_group2 = InnerIteration(a, mu, sig_t, sig_s, q_e, N, psi_group1, psi_group2, psi_group3, 2,
                converge)
123         psi_group3, phi_group3 = InnerIteration(a, mu, sig_t, sig_s, q_e, N, psi_group1, psi_group2, psi_group3, 3,
                converge)
124
125     # Calculate new group source values
126     groupsource_new = np.zeros(N)
127     for group in range(3):
128         groupsource_new += q_e[group] + (2./(np.size(mu)*np.size(q_e))) * (sig_s[group,0]*psi_group1.sum(axis=1)) +
                (sig_s[group,1]*psi_group2.sum(axis=1)) + (sig_s[group,2]*psi_group3.sum(axis=1))
129
130     # Calculate convergence cirterion
131     outercrit = np.sqrt(np.sum((groupsource_new - groupsource)**2))
132
133     # Check convergence
134     if (outercrit < converge): outerconverge = True
135
136     # Increment outer iteration number
137     outeritr += 1
138
139     print 'Outer group iteration = %i' % outeritr
140
```

```
141
142  # Plot scalar fluxes
143  plt.figure()
144  plt.plot(x,phi_group1, marker='s', color='c',linestyle='none', label='Group 1')
145  plt.plot(x,phi_group2, marker='s', color='b',linestyle='none', label='Group 2')
146  plt.plot(x,phi_group3, marker='s', color='r',linestyle='none', label='Group 3')
147  plt.xlim(-0.1,2.1)
148  plt.ylim(0,8)
149  plt.xlabel('x'), plt.ylabel('Center $\phi$')
150  plt.title('Cell Center Scalar Flux Profile, $\\alpha$=%.2f' % a)
151  plt.legend(numpoints=1,fontsize=10)
152
153  # Render plots
154  plt.show()
```

Output:

```
(Group = 1) Number of iterations = 7
(Group = 2) Number of iterations = 1
(Group = 3) Number of iterations = 7
Outer group iteration = 1
(Group = 1) Number of iterations = 1
(Group = 2) Number of iterations = 6
(Group = 3) Number of iterations = 7
Outer group iteration = 2
(Group = 1) Number of iterations = 1
(Group = 2) Number of iterations = 4
(Group = 3) Number of iterations = 7
Outer group iteration = 3
(Group = 1) Number of iterations = 1
(Group = 2) Number of iterations = 4
(Group = 3) Number of iterations = 5
Outer group iteration = 4
(Group = 1) Number of iterations = 1
(Group = 2) Number of iterations = 3
(Group = 3) Number of iterations = 5
Outer group iteration = 5
(Group = 1) Number of iterations = 1
(Group = 2) Number of iterations = 3
(Group = 3) Number of iterations = 2
Outer group iteration = 6
(Group = 1) Number of iterations = 1
(Group = 2) Number of iterations = 1
(Group = 3) Number of iterations = 2
Outer group iteration = 7
(Group = 1) Number of iterations = 1
(Group = 2) Number of iterations = 1
(Group = 3) Number of iterations = 1
Outer group iteration = 8
```

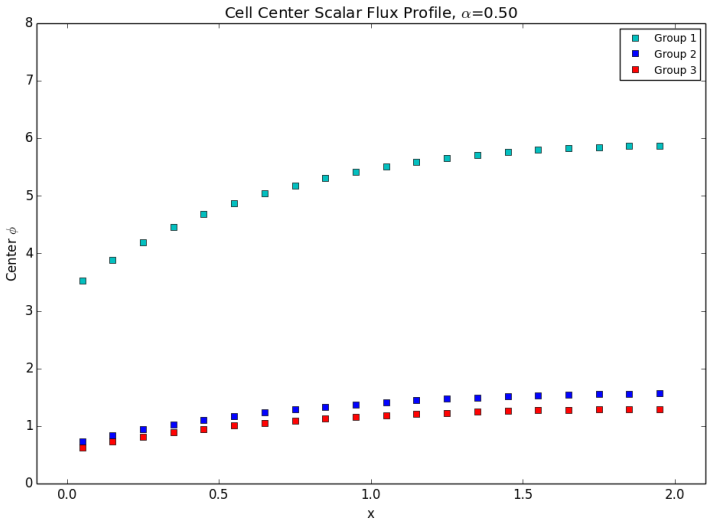

Fig. 1. Cell centered scalar flux values for groups 1, 2, and 3.

Aside: If we reduce the total cross-sections in group 2 and 3 (to 0.5 in each group), we see that the group 3 flux surpasses the group 2 flux (Fig. 2). This agrees with intuition, as now particles in group 2 will have a higher likelihood to scatter into group 3 and once in group 3, have a smaller likelihood to get absorbed. We also see an increase in amplitude for both group 2 and group 3, because the absorption cross-section is now lower. And again, since groups 2 and 3 have no influence on group 1, the group 1 flux is unchanged.

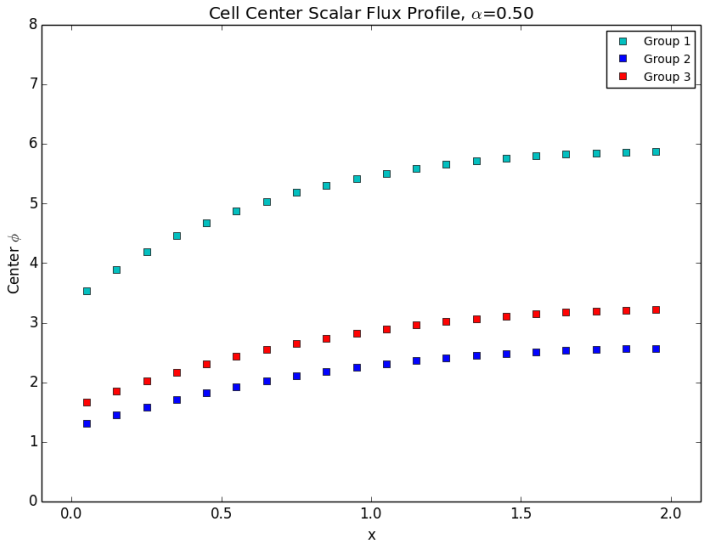

Fig. 2. Cell centered scalar flux values for groups 1, 2, and 3 using $\Sigma_t = 0.5$ in all groups.

Another aside: The code also implements the Gauss-Seidel method, which computes the within group iterations sequentially (group 1 then group 2 then group 3) and updates the fluxes between each group. This means the resultant flux we get from the inner iterations in group 1 will be used in the calculation of the source in group 2. And similarly for group 3. This means the outer iterations should converge faster. Rerunning the code but using the Gauss-Seidel method (line 91), we get the same results but only needed 5 outer group iterations (see below). We also notice it took 6 inner iterations for group 2 in the first outer iteration (as opposed to 1 in the Jacobi method) - this is because in the Jacobi method, group 2 has no information in the first outer iteration (no source, all other fluxes are zero) so the flux converges to 0. However in the Gauss-Seidel method, after we perform the inner iteration in group 1, we have a new group 1 flux that is used in the group 2 inner iteration and thus it has to iterate a few times before convergence.

Output:

```
1   (Group = 1) Number of iterations = 7
2   (Group = 2) Number of iterations = 6
3   (Group = 3) Number of iterations = 8
4   Outer group iteration = 1
5   (Group = 1) Number of iterations = 1
6   (Group = 2) Number of iterations = 5
7   (Group = 3) Number of iterations = 5
8   Outer group iteration = 2
9   (Group = 1) Number of iterations = 1
10  (Group = 2) Number of iterations = 3
11  (Group = 3) Number of iterations = 3
12  Outer group iteration = 3
13  (Group = 1) Number of iterations = 1
14  (Group = 2) Number of iterations = 2
15  (Group = 3) Number of iterations = 1
16  Outer group iteration = 4
17  (Group = 1) Number of iterations = 1
18  (Group = 2) Number of iterations = 1
19  (Group = 3) Number of iterations = 1
20  Outer group iteration = 5
```