NE 255 - Homework 4

University of California, Berkeley
Department of Nuclear Engineering

Daniel Hellfeld
dhellfeld@berkeley.edu

Problem 1
Describe the sweeping process in one dimension (marching through space) along one angle using the diamond difference scheme.

⇒ The diamond difference (DD) scheme is a finite volume method which uses a cell-average value solution. As a cell-balance method, this appraoch is essentially a statement of conservation of particles in a mesh cell of interest. For a given time step, energy group, angle, and neutron source, the neutron transport equation can be written as:

$$\hat{\Omega} \cdot \nabla \psi(\vec{r}) + \Sigma_t(\vec{r})\psi(\vec{r}) = s(\vec{r})$$

To arrive at the cell balance equation, we integrate this over the mesh cell ($dV = dxdydz$), depicted in Fig. 1, and divide by the total volume.

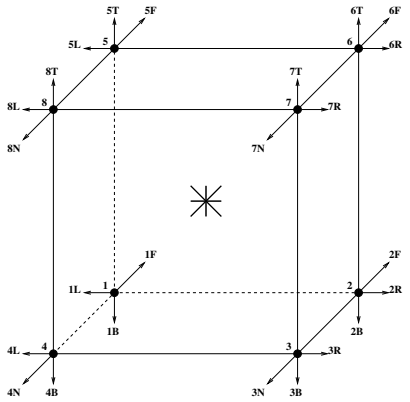

Fig. 1. General mesh cell used to discretize spatial equations.

First we state a few useful definitions:

$$\hat{\Omega} = \mu\hat{x} + \eta\hat{y} + \xi\hat{z}$$

$$\iiint = \int_{z_{k-1/2}}^{z_{k+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} \int_{x_{i-1/2}}^{x_{i+1/2}}$$

$$dV = dxdydz$$

$$\Delta_i = x_{i+1/2} - x_{i-1/2} =$$

$$\Delta_j = y_{j+1/2} - y_{j-1/2}$$

$$\Delta_k = z_{k+1/2} - z_{k-1/2}$$

Now, we integrate over the mesh cell and divide by the volume of the mesh cell ($V_c = \Delta_i\Delta_j\Delta_k$)

$$\frac{1}{V_c}\iiint \left(\mu\frac{\partial}{\partial x}\psi(\vec{r}) + \eta\frac{\partial}{\partial x}\psi(\vec{r}) + \xi\frac{\partial}{\partial x}\psi(\vec{r}) + \Sigma_t(\vec{r})\psi(\vec{r})\right) dV = \frac{1}{V_c}\iiint s(\vec{r})dV$$

$$\frac{\Delta_j\Delta_k}{V_c}\mu\int_{x_{i-1/2}}^{x_{i+1/2}} \partial\psi + \frac{\Delta_i\Delta_k}{V_c}\eta\int_{y_{j-1/2}}^{y_{j+1/2}} \partial\psi + \frac{\Delta_i\Delta_j}{V_c}\xi\int_{z_{k-1/2}}^{z_{k+1/2}} \partial\psi + \Sigma_{t,i,j,k}\psi_{ijk} = s_{ijk}$$

$$\frac{\mu}{\Delta_i}(\psi_{i+1/2} - \psi_{i-1/2}) + \frac{\eta}{\Delta_j}(\psi_{j+1/2} - \psi_{j-1/2}) + \frac{\xi}{\Delta_k}(\psi_{k+1/2} - \psi_{k-1/2}) + \Sigma_{t,i,j,k}\psi_{ijk} = s_{ijk}$$

Now we relate the center fluxes (ψ_i) to the edge fluxes ($\psi_{i\pm1/2}$) and close the above equation with a weighted average of the edge fluxes. We show this in just one dimension (x) here for brevity.

$$\begin{aligned}
\psi_i &= \frac{1}{2}\big((1+\alpha_i)\psi_{i+1/2} + (1-\alpha_i)\psi_{i-1/2}\big) \\
\psi_{i+1/2} &= \frac{2}{(1+\alpha_i)}\psi_{ijk} - \frac{(1-\alpha_i)}{(1+\alpha_i)}\bar{\psi}_{i-1/2}\,, \quad \mu > 0 \ (\psi_{i-1/2} \text{ is incoming}) \\
\psi_{i-1/2} &= \frac{2}{(1-\alpha_i)}\psi_{ijk} - \frac{(1+\alpha_i)}{(1-\alpha_i)}\bar{\psi}_{i+1/2}\,, \quad \mu < 0 \ (\psi_{i+1/2} \text{ is incoming})
\end{aligned}$$

where $-1 \leq \alpha \leq 1$ and $\bar{\psi}$ indicate the incoming flux (which will depend on μ). The α's are weighting factors that are set to zero in the classic DD scheme ($\alpha = \pm 1$ is the step-difference scheme):

$$\begin{aligned}
\psi_i &= \frac{1}{2}\big(\psi_{i+1/2} + \psi_{i-1/2}\big) \\
\psi_{i+1/2} &= 2\psi_{ijk} - \bar{\psi}_{i-1/2}\,, \quad \mu > 0 \ (\psi_{i-1/2} \text{ is incoming}) \\
\psi_{i-1/2} &= 2\psi_{ijk} - \bar{\psi}_{i+1/2}\,, \quad \mu < 0 \ (\psi_{i+1/2} \text{ is incoming})
\end{aligned}$$

We substitute this into the cell balance transport equation (for all three dimensions) and rearrange to arrive at the following set of equations:

$$\begin{aligned}
\psi_{ijk} &= \frac{s_{ijk} + \frac{2|\mu|}{\Delta i}\bar{\psi}_{i\mp1/2} + \frac{2|\eta|}{\Delta j}\bar{\psi}_{j\mp1/2} + \frac{2|\xi|}{\Delta k}\bar{\psi}_{k\mp1/2}}{\Sigma_{t,ijk} + \frac{2|\mu|}{\Delta i} + \frac{2|\eta|}{\Delta j} + \frac{2|\xi|}{\Delta k}} \\
\psi_{i\pm1/2} &= 2\psi_{ijk} - \bar{\psi}_{i\mp1/2} \\
\psi_{j\pm1/2} &= 2\psi_{ijk} - \bar{\psi}_{j\mp1/2} \\
\psi_{k\pm1/2} &= 2\psi_{ijk} - \bar{\psi}_{k\mp1/2}
\end{aligned}$$

for $\mu \gtrless 0\,,\ \eta \gtrless 0\,,\ \xi \gtrless 0$. Now, in one dimension this reduces to, for $\mu \gtrless 0$:

$$\begin{aligned}
\psi_i &= \frac{s_i + \frac{2|\mu|}{\Delta i}\bar{\psi}_{i\mp1/2}}{\Sigma_{t,i} + \frac{2|\mu|}{\Delta i}} \\
\psi_{i\pm1/2} &= 2\psi_i - \bar{\psi}_{i\mp1/2}
\end{aligned}$$

(a) for $\mu > 0$ (or moving from left to right)

In this case, our equations become

$$\begin{aligned}
\psi_i &= \frac{s_i + \frac{2\mu}{\Delta i}\bar{\psi}_{i-1/2}}{\Sigma_{t,i} + \frac{2\mu}{\Delta i}} \\
\psi_{i+1/2} &= 2\psi_i - \bar{\psi}_{i-1/2}
\end{aligned}$$

Now to march through the sweeping process for one angle using the DD scheme we would first need to assume some value for the incoming flux on the left most face of our geometry. Let's number our geometry such that the first cell is $i = 0$, and the next one to the right is $i = 1$, and so on. So we would need to know (or be given) the incoming flux at cell 0, $\bar{\psi}_{-1/2}$. With this we can then solve for ψ_0 using the equation above, and then also solve for $\psi_{1/2}$ using $\bar{\psi}_{-1/2}$ and ψ_0. With $\psi_{1/2}$ in hand, we could then move to the next cell, $i = 1$, set $\bar{\psi}_{1/2} = \psi_{1/2}$ and then solve for ψ_1, and then $\psi_{3/2}$. We would then repeat the process of using the left face flux to solve for the center flux, and then using the left and center fluxes together to solve for the right face flux for each cell. We do this in order, moving from the left to the right (in the direction of particle flow). Also, in each cell we are computing the source s_i which very well may depend on the flux. Therefore after the sweep, we would need to update s_i before starting another sweep. We continue the sweeps until we have satisfied some convergence criterion.

(b) for $\mu < 0$ (or moving from right to left)

In this case, our equations become

$$\psi_i = \frac{s_i + \frac{2|\mu|}{\Delta i}\bar{\psi}_{i+1/2}}{\Sigma_{t,i} + \frac{2|\mu|}{\Delta i}}$$
$$\psi_{i-1/2} = 2\psi_i - \bar{\psi}_{i+1/2}$$

The only difference now is that in order to calculate the center flux, we need to use the right face flux. Once we have the right and center fluxes, we can then solve for the left face flux. To start the sweep, we would need to assume (or be given) some value for the right-most right face flux. If we have a total of N cells, then we would need to know $\bar{\psi}_{N+1/2}$. With this we would solve for ψ_N, and then for $\psi_{N-1/2}$ (the left face). We would then continue the process like above, but now in the opposite direction. Starting from the right most right face flux, solving for the center and left face fluxes, then moving to the next cell to the left and using the left face flux from the previous cell as the incoming flux. Again, after the sweep we update s_i, and then keep performing sweeps until convergence.

(c) At a reflecting boundary on the right edge, including how to transition from $\mu > 0$ to $\mu < 0$.

If there was a reflecting boundary on the right most edge of the geometry, then the right moving ($\mu > 0$) face flux in the right most cell will act as the left incoming flux when we switch to the $\mu < 0$ case. Using the same indexing scheme as above, the right most cell is $i = N$. We would sweep in the $\mu > 0$ direction just as we did in part A (needing to be given the incoming flux on the left, $\bar{\psi}_{-1/2}$). Once we reach cell N, we would finish by calculating $\psi_{N+1/2}$. This flux would then be used as $\bar{\psi}_{N+1/2}$ (notice the bar) when we move into the $\mu < 0$ direction and begin by solving for ψ_N. Once we have swept to the right AND left, we then update s_i and repeat until convergence.

What if we have multiple angles in each direction? The above case would only work if we performed our sweep for each angle individually and the reflecting boundary reflected the particles from μ_1 to $-\mu_1$. This is not always the case (some reflectors can exhibit an isotropic reflection, μ_1 can reflect to $-\mu_1$, $-\mu_2$, etc.)

(d) If using the angular flux to generate flux moments during the solution process, what data do you need to store in the sweeping process?

For each cell, we need to know the incoming flux and the source to calculate the center flux and then subsequently the outgoing flux. So we will need to store the moments of the incoming flux in order to calculate the source moments for that cell. With the incoming flux moments and source moments stored, we can calculate the center flux moments. We can then discard the source moments and use the center flux moments and incoming flux moments to calculate the outgoing flux moments. Once we have the outgoing flux moments stored, we can discard the incoming and center flux moments of that cell and then move on to the next cell, using the previous outgoing flux moments now as the incoming flux moments.

In each cell we need to perform three calculations - the source term, the center flux, and the outgoing flux. We do these in order and we only need the incoming flux moments to determine all three. Therefore, for each cell we only need to store the flux moments of the incoming flux into that cell. Once we move to the next cell, we can overwrite those fluxes with the incoming flux in the next cell.

Problem 2

Let's look at truncation error in the diamond difference method by examining the 1-D case. Consider uncollided neutrons with a zero group source moving along angle a:

$$\mu_a \frac{d\psi_a}{dx} + \Sigma_t \psi_a(x) = 0 \,,$$

where the cross section is taken as constant.

(a) For neutrons moving $\mu_a > 0$, write an expression for the flux at some location x' in terms of the flux at location x (you should have an exponential).

The differential equation above is simple enough to solve by hand. First, we rearrange:

$$\frac{d\psi_a(x)}{dx} = -\frac{\Sigma_t}{\mu_a}\psi_a(x)$$
$$\frac{d\psi_a(x)}{\psi_a(x)} = -\frac{\Sigma_t}{\mu_a}dx$$

We now integrate both sides:

$$\int \frac{d\psi_a(x)}{\psi_a(x)} = -\frac{\Sigma_t}{\mu_a}\int dx$$
$$\ln(\psi_a(x)) = -\frac{\Sigma_t}{\mu_a}x + C$$
$$\psi_a(x) = e^{-\frac{\Sigma_t}{\mu_a}x+C}$$
$$= e^{-\frac{\Sigma_t}{\mu_a}x}e^{C}$$
$$= Ce^{-\frac{\Sigma_t}{\mu_a}x}$$

Now we can write an expression for the flux at x' to the flux at x:

$$\frac{\psi_a(x')}{\psi_a(x)} = \frac{e^{-\frac{\Sigma_t}{\mu_a}x'}}{e^{-\frac{\Sigma_t}{\mu_a}x}}$$
$$= e^{-\frac{\Sigma_t}{\mu_a}x'}e^{\frac{\Sigma_t}{\mu_a}x}$$
$$= e^{-\frac{\Sigma_t}{\mu_a}(x'-x)}$$
$$\psi_a(x') = \psi_a(x)e^{-\frac{\Sigma_t}{\mu_a}(x'-x)}$$

The agrees with intuition (exponential falloff of flux into an absorbing material).

(b) Let's say we impose a Cartesian grid with mesh index i. What is the expression for $\psi_{a,i+1/2}$ in terms of $\psi_{a,i-1/2}$? Use mesh spacing $\Delta_i = x_{i+1/2} - x_{i-1/2}$ and the definition $h \equiv \frac{\Sigma_t \Delta_i}{2|\mu_a|}$.

To gain more intuition on this problem and to relate it to part A, lets rewrite the given fluxes:

$$\psi_{a,i+1/2} = \psi_a(x_{i+1/2})$$
$$\psi_{a,i-1/2} = \psi_a(x_{i-1/2})$$

Now we see it is easy to substitute into our expression in part A:

$$\psi_a(x_{i+1/2}) = \psi_a(x_{i-1/2})e^{-\frac{\Sigma_t}{\mu_a}(x_{i+1/2}-x_{i-1/2})}$$
$$\Rightarrow \quad \psi_{a,i+1/2} = \psi_{a,i-1/2}\, e^{-\frac{\Sigma_t}{\mu_a}\Delta_i}$$
$$= \psi_{a,i-1/2}\, e^{-\frac{\Sigma_t}{\mu_a}\Delta_i}$$
$$= \psi_{a,i-1/2}\, e^{-2h}$$

(c) Plug the relationship you just found into the 1D diamond difference equations ($\alpha = 0$). Manipulate those to get another expression for $\psi_{a,i+1/2}$ in terms of $\psi_{a,i-1/2}$ and h.

The 1D DD equations are as follows:

$$\mu_a(\psi_{a,i+1/2} - \psi_{a,i-1/2}) + \Sigma_t \Delta_i \psi_i = 0$$
$$\psi_i = \frac{1}{2}(\psi_{a,i+1/2} + \psi_{a,i-1/2})$$

Plugging in the bottom equation into the top equation we get

$$\psi_{a,i+1/2} - \psi_{a,i-1/2} = -\frac{\Sigma_t \Delta_i}{2\mu_a}(\psi_{a,i+1/2} + \psi_{a,i-1/2})$$
$$\psi_{a,i+1/2} - \psi_{a,i-1/2} = -h(\psi_{a,i+1/2} + \psi_{a,i-1/2})$$
$$\psi_{a,i+1/2}(1+h) = \psi_{a,i-1/2}(1-h)$$
$$\Rightarrow \quad \psi_{a,i+1/2} = \left(\frac{1-h}{1+h}\right)\psi_{a,i-1/2}$$

We did not need to plug in anything from part B... I am not sure why that was suggested.

(d) Look again at your solution from part b. Expand the exponential in a power series and expand your part c expression in a power series through the h^2 terms and show they are the same. What does that mean about the accuracy of the relationship?

Expanding the exponential term found in part B up to the h^2 term, we get:

$$e^{-2h} \approx \frac{e^{-2h}|_{h=0}}{0!}(h-0)^0 + \frac{-2e^{-2h}|_{h=0}}{1!}(h-0)^1 + \frac{4e^{-2h}|_{h=0}}{2!}(h-0)^2$$
$$\approx 1 - 2h + 2h^2$$

And expanding the $\frac{1-h}{1+h}$ term from part C, we get:

$$\frac{1-h}{1+h} \approx \frac{\frac{1-h}{1+h}|_{h=0}}{0!}(h-0)^0 + \frac{-2(1+h)^{-2}|_{h=0}}{1!}(h-0)^1 + \frac{4(1+h)^{-3}|_{h=0}}{2!}(h-0)^2$$
$$\approx 1 - 2h + 2h^2$$

We see that they are the same up to $O(h^2)$ in the power series expansions. Therefore the DD scheme is a second order accurate, or $O(h^2)$, method. If we use h, we will get some truncation error, ϵ. The second order accuracy tells us that if we use $h/2$ instead of h, the truncation error will be $\epsilon/4$. If we use $h/4$, the truncation error will be $\epsilon/16$, and so on.

(e) Look carefully at the expression for $\psi_{a,i+1/2}$. What is a condition on h that would guarantee that the flux would be positive? What does that mean about mesh spacing given the smallest μ_a in a set and a specific Σ_t?

Inspecting the expression for $\psi_{a,i+1/2}$ in part C, we see that the resulting flux will be negative if $h > 1$. Therefore to guarantee the flux to be positive, we must enforce that $h \leq 1$ when using the DD scheme. It also does not make physical sense to have a negative value for h (by inspection of its definition), so to be explicit, h must be defined on the following domain: $0 \leq h \leq 1$. Given the smallest μ_a and a specific Σ_t, the mesh spacing, Δ_i, must defined to accommodate the requirement on h.

$$h = \frac{\Sigma_t \Delta_i}{2|\mu_a|} \quad \Rightarrow \quad \Delta_i = \frac{2|\mu_a|h}{\Sigma_t} = \frac{2|\mu_a|[0 \to 1]}{\Sigma_t} = [0 \to \frac{2|\mu_a|}{\Sigma_t}] = [0 \to 2|\mu_a|\lambda]$$

where λ is the mean free path in the material. Therefore we have $0 \leq \Delta_i \leq 2|\mu_a|\lambda$. Which can be very small in highly absorbing media (small mean free path) and with highly anisotropic fluxes where high order S_N is needed (small minimum $|\mu_a|$ values)! With a small Δ_i (or small h), we have many mesh cells and thus the computation time will increase!

Problem 3
Write a piece of code that implements the 1-D, one-speed, steady state, weighted diamond difference equations; include scattering and an external source. Use $\psi(0) = 2.0$ for $\mu > 0$; non-reentrant boundary condition at $x = 0.0$ and a reflecting boundary at $x = 2.0$. For this case assume isotropic scattering.

$\Rightarrow$ The 1-D, one-speed, steady-state, weighted DD equations are given by:

$$\psi_i = \frac{s_i + \frac{2}{(1\pm\alpha_i)}\frac{|\mu|}{\Delta_i}\bar{\psi}_{i\mp1/2}}{\Sigma_{t,i} + \frac{2}{(1\pm\alpha_i)}\frac{|\mu|}{\Delta_i}}$$
$$\psi_{i\pm1/2} = \frac{2}{(1\pm\alpha_i)}\psi_{ijk} - \frac{(1\mp\alpha_i)}{(1\pm\alpha_i)}\bar{\psi}_{i\mp1/2}$$

for $\mu \gtrless 0$ and where s_i includes the scattering and external source, given by:

$$\begin{aligned} s_i &= q_{e,i} + \int_{-1}^{1} d\mu' \Sigma_s(\mu' \to \mu)\psi_i(\mu') \\ &\approx q_{e,i} + \sum_{\mu'} w_{\mu'}\Sigma_s\psi_i(\mu') \end{aligned}$$

where $q_{e,i}$ is the external source and w_μ is the quadrature weight for direction μ. We are ultimately interested in the scalar flux, which is given by:

$$\begin{aligned} \phi &= \int_{-1}^{1} d\mu \psi_i(\mu) \\ &\approx \sum_{\mu} w_\mu \psi_i(\mu) \end{aligned}$$

(a) Explore negative flux: use the following values

- $\alpha = 0$
- $\mu_a = \pm 0.1$
- $\Sigma_t = 1.0$
- $\Sigma_s = 0$
- $q_e(x) = 0$
- mesh spacings: $\Delta_i = [0.08, 0.1, 0.125, 0.2, 0.4]$.

Plot the cell-centered scalar flux that results from each mesh spacing. What do you notice? How does that compare with your conclusion from the previous problem?

$\Rightarrow$ In this case, the problem is greatly simplified. We do not need to concern ourselves with the source at all. Therefore we only need to perform one sweep (to the right and then left). Beginning with the boundary condition of $\psi(0) = 2$, we can solve for the center flux of the first mesh cell, and then subsequently the outgoing flux from that cell (all with $\mu > 0$). We then set the outgoing flux as the incoming flux of the next cell and repeat for all cells until we reach the reflecting boundary at $x = 2$. At this point, the outgoing flux in the $\mu > 0$ direction becomes the incoming flux in the $\mu < 0$ direction. We then solve for the center and outgoing flux and march through the mesh cells in the left direction. After the sweep is done, we will have an array of center flux values (angular flux). We then use our quadrature rule to integrate over direction to get the scalar flux. In this case we use an equiprobable quadrature set that is normalized to 2 (as suggested by Prof. Slaybaugh, this will lead to cancelations with all the 2π's that results from the integration over $d\phi$ - I will thus neglect these integrations here). We only have two directions in this case, so each weight, w_μ is equal to 1.

The above method was implemented in Python for each mesh spacing and the code is shown below. The results of the center angular flux and center scalar flux in each mesh cell are shown in Fig. 2. The code is able to handle non-zero values of α, therefore this same code was used for part B below.

```
# NE 255 - Homework 4, Problem 3a-b
# Daniel Hellfeld
# 11/8/16

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

# Define variables
a = 0.0    # -0.9, -0.5, 0, 0.25, 0.5, 0.9
mu = np.asarray([0.1,-0.1])
sig_t = 1.0
d_array = np.asarray([0.4,0.2,0.125,0.1,0.08])
weight = np.asarray([1.,1.]) # (made up quaderature set, equiprobable, normalized to 2)

# Loop through mesh spacings
loopnum = 1  # for subplots
for d in d_array:

    # Define more variables
    N = 2. / d
    x = np.linspace((d/2.), 2.-(d/2.),N) # get center points for plotting

    # Initialize center fluxes arrays
    psi_cent_right = np.zeros(N)
    psi_cent_left = np.zeros(N)

    # Boundary condition
    psi_in = 2.

    # Sweep in mu > 0 (right)
    for i in range(int(N)):
        psi_cent_right[i] = ((2. * np.abs(mu[0]) / (d * (1.+a))) * psi_in) / (sig_t + (2. * np.abs(mu[0]) / (d*(1.+a))))
        psi_out = ((2. / (1.+a)) * psi_cent_right[i]) - ((1.-a)/(1.+a))*psi_in
        psi_in = psi_out

    # Sweep in mu < 0 (left)
    for i in range(int(N)):
        psi_cent_left[N-i-1] = ((2. * np.abs(mu[1]) / (d*(1.-a))) * psi_in) / (sig_t + (2. * np.abs(mu[1]) / (d*(1.-a))))
        psi_out = ((2./(1.-a)) * psi_cent_left[N-i-1]) - ((1.+a)/(1.-a))*psi_in
        psi_in = psi_out

    # Calculate scalar flux from angular flux (quaderature)
    phi = weight[0]*psi_cent_left + weight[1]*psi_cent_right

    # Plot angular flux
    plt.subplot(5,2,2*loopnum-1)
    plt.subplots_adjust(hspace = 0.0, wspace=0.2)
    plt.plot(x,psi_cent_right, marker="o", linestyle='none',label='$\mu>0$')
    plt.plot(x,psi_cent_left, marker="o",color='r', linestyle='none', label='$\mu<0$')
    plt.xlim(-0.1,2.1)
    plt.ylim(np.min(phi)-0.5*np.max(phi),np.max(psi_cent_right)+0.3*np.max(psi_cent_right))
    plt.xlabel('x'), plt.ylabel('Center $\psi$, ($\Delta_i$=%.2f)' % d)
    if (loopnum == 1): plt.title('Cell Center Angular Flux Profile, $\\alpha$=%0.2f' % a)
    plt.legend(numpoints=1,fontsize=10)

    # Plot scalar flux
    plt.subplot(5,2,2*loopnum)
    plt.plot(x,phi, marker='s', color='c',linestyle='none')
    plt.xlim(-0.1,2.1)
    plt.ylim(np.min(phi)-0.3*np.max(phi),np.max(phi)+0.3*np.max(phi))
    plt.xlabel('x'), plt.ylabel('Center $\phi$, ($\Delta_i$=%.2f)' % d)
    if (loopnum == 1): plt.title('Cell Center Scalar Flux Profile, $\\alpha$=%.2f' % a)

    # Increment loopnum
    loopnum += 1

# Render plots
plt.show()
```

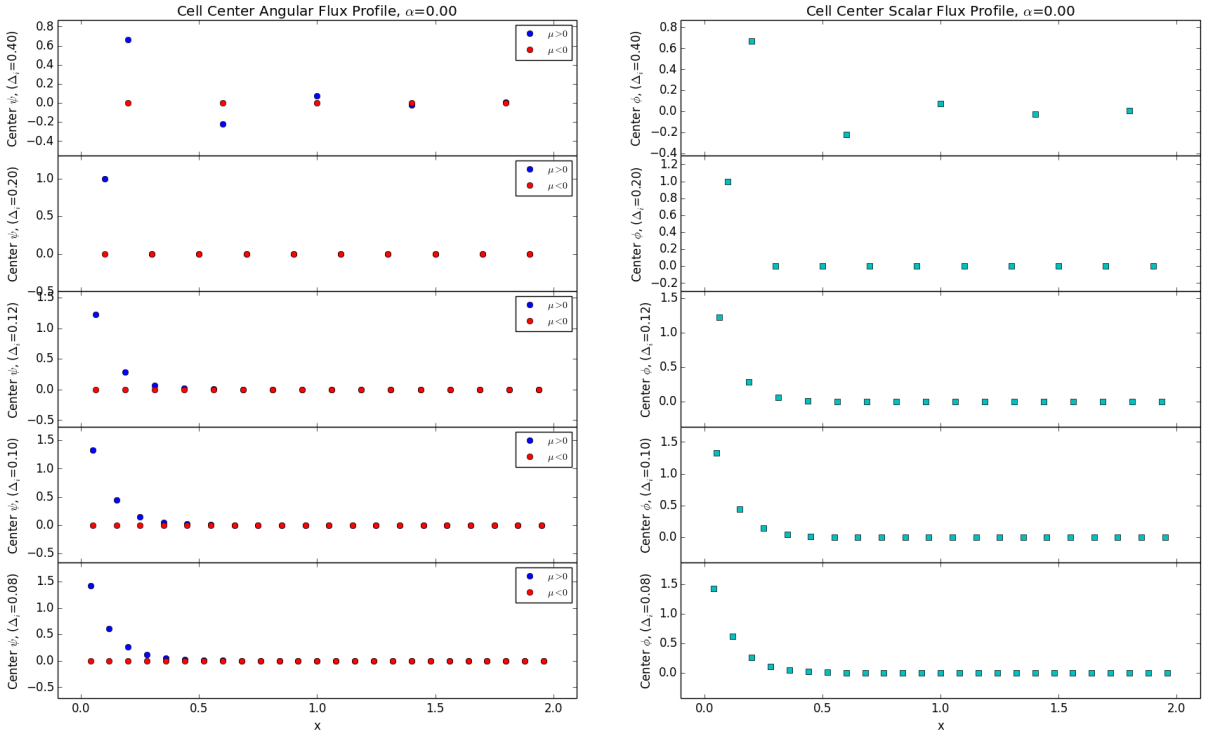

Fig. 2. Center angular and scalar flux profiles for varying mesh spacings with $\alpha = 0$, and no scattering or external source.

We see that the right moving angular flux has an exponential fall off as expected. The flux is quite small by the time it reaches the reflecting boundary, so the left moving angular flux is essentially zero in all cells. In the scalar flux case, we simply add the two angular flux values together (their weights are equal to 1), and since the left moving angular flux is essentially zero, we see the scalar flux is about the same as the right moving angular flux. Now the most important thing to notice is that the flux (in both angular and scalar) goes negative in the $\Delta_i = 0.4$ case. In the previous problem, we derived that Δ_i must be less than or equal to $\frac{2|\mu|}{\Sigma_t}$ to avoid negative flux, which in this case is equal to 0.2. This is violated when $\Delta_i = 0.4$, and thus our calculation indeed results in a negative flux value. This agrees with our conclusion from the previous problem.

(b) Impact of α: try $\alpha = [-0.9, -0.5, 0.25, 0.5, 0.9]$. What happens?

We use the same code as above, but we now vary α. The results are shown in Figs. 3-7.

For $\alpha = -0.9$, we observe positive and negative fluxes for mesh spacings greater than 0.1. For a mesh spacing of 0.1, the flux seems to drop off rather quickly near the left boundary. The smallest mesh spacing (0.08) seems to produce a reasonable looking flux. For $\alpha = -0.5$, we again observe positive and negative fluxes, but now only for mesh spacings greater than 0.12. The flux also exhibits the quick exponential fall-off for mesh spacing smaller than 0.12, but it begins to look more reasonable as the mesh spacing is decreased further. For $\alpha = 0.25$, only a mesh spacing of 0.4 produces negative flux values. And again, the flux begins to look more reasonable as the mesh spacing is decreased further. For $\alpha = 0.5$, we no longer have any negative fluxes and the flux looks more reasonable as the mesh spacing is decreased (only have a quick fall off for a mesh spacing of 0.4). For $\alpha = 0.9$, we no longer have any negative fluxes and the flux looks more reasonable as the mesh spacing is decreased (it also looks smoother than $\alpha = 0.5$ for all mesh spacings).

It seems that as we increase the value of α, we start to improve the behavior for large mesh spacings. This is good because in the instance where large meshing is needed (to reduce computational time), we can introduce

a large positive value of α to assist in better reconstructing the flux (at least for the parameters we are using here).

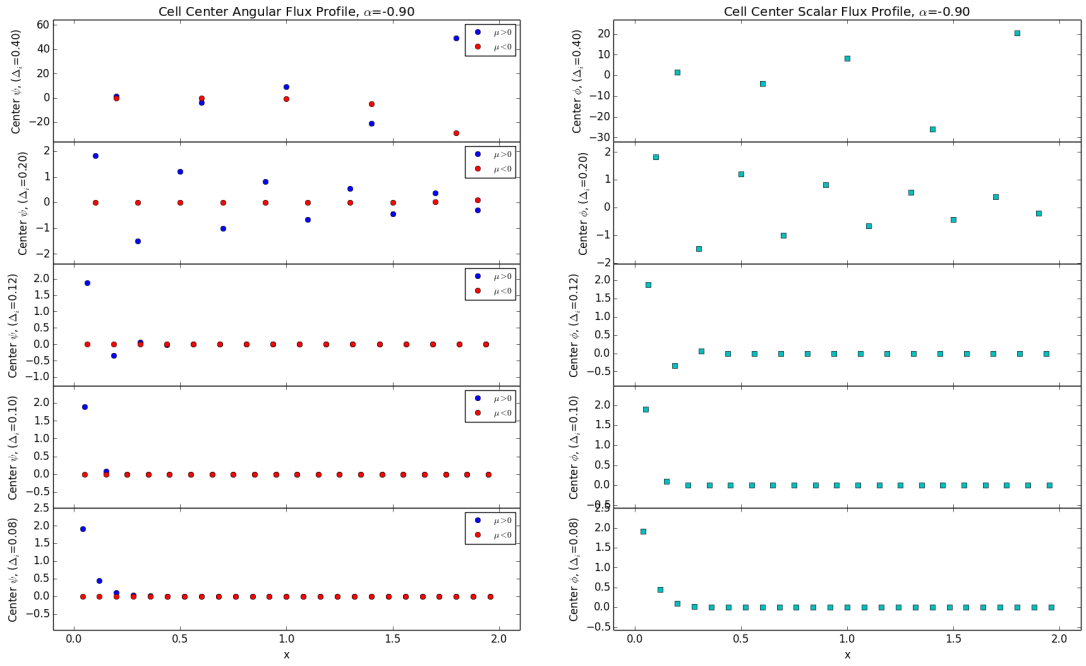

Fig. 3. Center angular and scalar flux profiles for varying mesh spacings with $\alpha = -0.9$, and no scattering or external source.

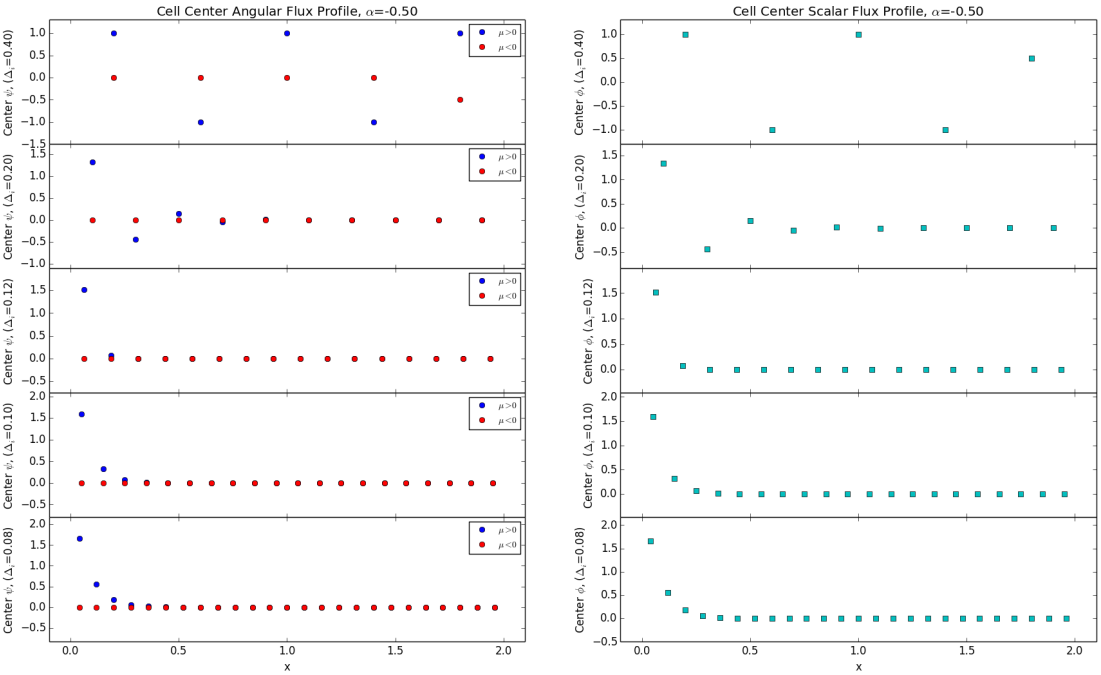

Fig. 4. Center angular and scalar flux profiles for varying mesh spacings with $\alpha = -0.5$, and no scattering or external source.

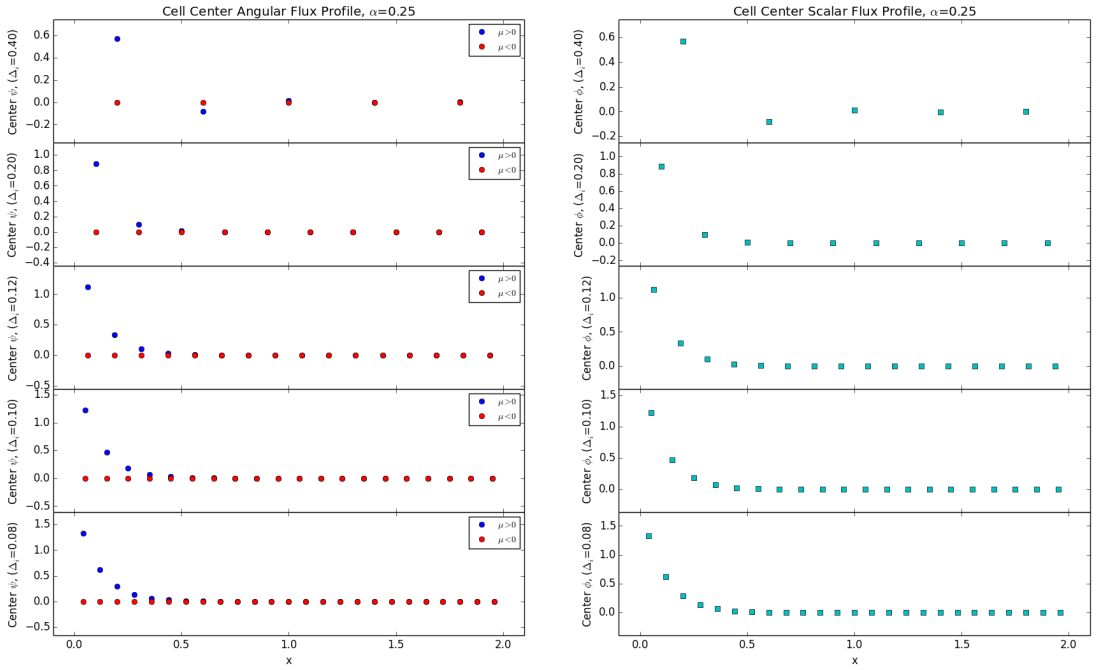

Fig. 5. Center angular and scalar flux profiles for varying mesh spacings with $\alpha = 0.25$, and no scattering or external source.

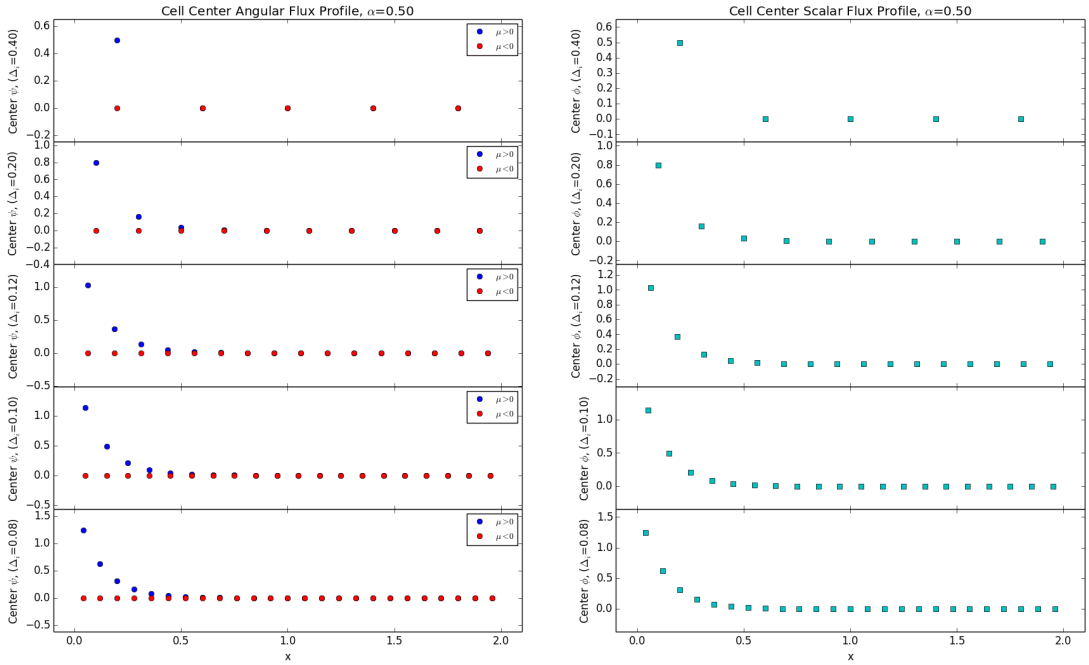

Fig. 6. Center angular and scalar flux profiles for varying mesh spacings with $\alpha = 0.5$, and no scattering or external source.

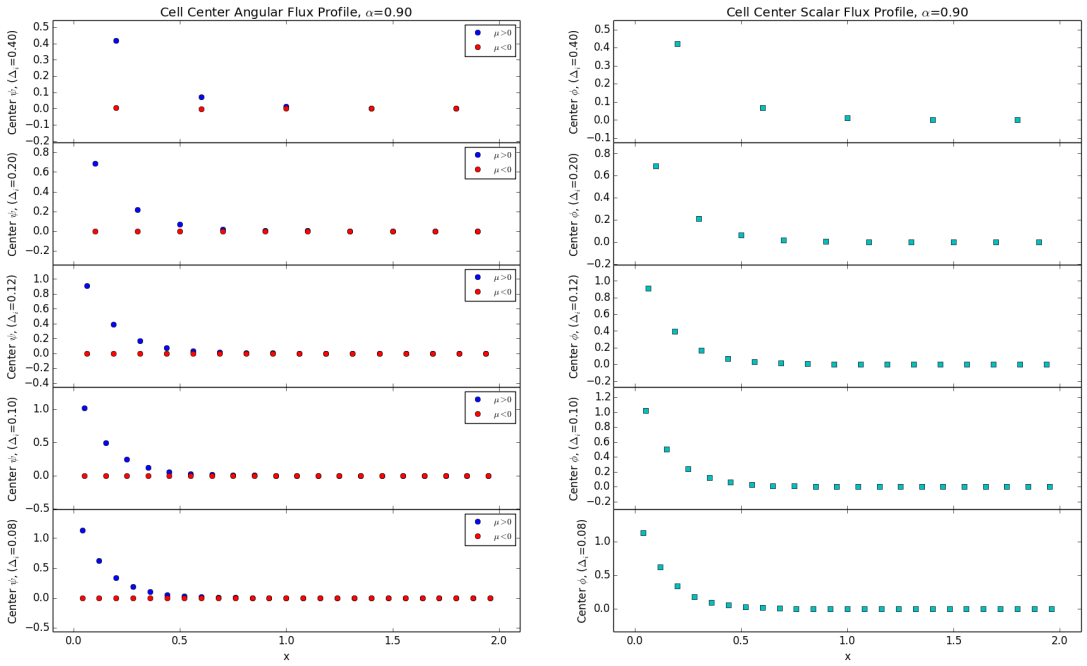

Fig. 7. Center angular and scalar flux profiles for varying mesh spacings with $\alpha = 0.9$, and no scattering or external source.

(c) Real results? Now try adding a source

- $\alpha = [-0.5, 0, 0.5]$ (feel free to try others)
- $\mu = \pm[0.2,\ 0.7]$
- $\Sigma_t = 1.0$
- $\Sigma_s = 0.5$
- $q_e(x) = 1.0$

Report the results.

Now we wish to include scattering, an external source, and more angles. Let's first consider scattering. We can use the quadrature to evaluate the integral over angle, but now we have 4 angles and thus to keep the same normalization as before (= 2), each equiprobable weight will now be 0.5. Using equiprobable weights for each angle is equivalent to isotropic scattering. Therefore our scattering source in each mesh cell i will be given by

$$\sum_{\mu} w_\mu \Sigma_s \psi_i(\mu) = \frac{\Sigma_s}{2}\left[\psi_i(-0.7) + \psi_i(-0.2) + \psi_i(0.2) + \psi_i(0.7)\right]$$

In each sweep we now calculate s_i and we also now calculate the angular flux for both angles (in both directions). We will need to assume some initial values of the angular flux when we first calculate the scattering source (I will assume a value of 1 in all cells and all directions). Once we do a full sweep (right and left), we then update the source term with the newly calculated angular flux values. After each sweep (right and left) we also calculate the scalar flux with our new quadrature weights. As we iterate through the sweeps and update the flux, we need to define a convergence criterion to eventually stop our calculation. My first attempt is to use the l_2 norm of the difference between the current scalar flux and the previous scalar flux (one could also use the difference in the scattering source, but since this is defined using the fluxes they should produce similar results). Once we have satisfied:

$$||\phi^k - \phi^{k-1}||_{l_2} < 10^{-4}$$

we stop the calculation and plot the results. The modified code is shown below, and the results using the three different values of α are shown in Figs. 8-10.

```
# NE 255 - Homework 4, Problem 3c-d
# Daniel Hellfeld
# 11/8/16

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

# Define variables
a = 0.5   # -0.5, 0.0, 0.5
mu = np.asarray([0.2,0.7,-0.2,-0.7])
sig_t = 1.0
sig_s = 0.5
q = 1.0
d_array = np.asarray([0.4,0.2,0.125,0.1,0.08])
weight = np.asarray([0.5,0.5,0.5,0.5]) # (made up quaderature set, equiprobable, normalized to 2)

# Loop through mesh spacings
loopnum = 1  # for subplots
for d in d_array:

    # Define more variables
    N = 2. / d
    x = np.linspace((d/2.), 2.-(d/2.),N) # get center points for plotting

    # Initialize center fluxes arrays (2D now b/c two angles)
    psi_cent_right = np.ones((N,2))
    psi_cent_left = np.ones((N,2))
    phi = np.zeros(N)
    psi_cent_right_new = np.zeros((N,2))
    psi_cent_left_new = np.zeros((N,2))
    phi_new = np.zeros(N)

    # Iterate
    converge = False
    itr = 0
    while (converge == False):

        # Boundary condition
        psi_in = np.asarray([2.,2.])
        psi_out = np.asarray([1.,1.])

        # Sweep in mu > 0 (right)
        for i in range(int(N)):
            s = q + sig_s * (weight[0]*psi_cent_right[i,0] + weight[1]*psi_cent_right[i,1] + weight[2]*psi_cent_left[i,0]
                 + weight[3]*psi_cent_left[i,1])

            psi_cent_right_new[i,0] = (s + (2. * np.fabs(mu[0]) / (d * (1.+a))) * psi_in[0]) / (sig_t + (2. *
                 np.fabs(mu[0]) / (d*(1.+a))))
            psi_cent_right_new[i,1] = (s + (2. * np.fabs(mu[1]) / (d * (1.+a))) * psi_in[1]) / (sig_t + (2. *
                 np.fabs(mu[1]) / (d*(1.+a))))

            psi_out[0] = ((2. / (1.+a)) * psi_cent_right[i,0]) - ((1.-a)/(1.+a))*psi_in[0]
            psi_out[1] = ((2. / (1.+a)) * psi_cent_right[i,1]) - ((1.-a)/(1.+a))*psi_in[1]

            psi_in = psi_out

        # Sweep in mu < 0 (left)
        for i in range(int(N)):
            s = q + sig_s * (weight[0]*psi_cent_right[N-i-1,0] + weight[1]*psi_cent_right[N-i-1,1] +
                 weight[2]*psi_cent_left[N-i-1,0] + weight[3]*psi_cent_left[N-i-1,1])

            psi_cent_left_new[N-i-1,0] = (s + (2. * np.fabs(mu[2]) / (d*(1.-a))) * psi_in[0]) / (sig_t + (2. *
                 np.fabs(mu[2]) / (d*(1.-a))))
            psi_cent_left_new[N-i-1,1] = (s + (2. * np.fabs(mu[3]) / (d*(1.-a))) * psi_in[1]) / (sig_t + (2. *
                 np.fabs(mu[3]) / (d*(1.-a))))

            psi_out[0] = ((2./(1.-a)) * psi_cent_left[N-i-1,0]) - ((1.+a)/(1.-a))*psi_in[0]
            psi_out[1] = ((2./(1.-a)) * psi_cent_left[N-i-1,1]) - ((1.+a)/(1.-a))*psi_in[1]

            psi_in = psi_out

        # Calculate scalar flux from angular flux (quaderature)
        phi_new = weight[0]*psi_cent_right_new[:,0] + weight[1]*psi_cent_right_new[:,1] + weight[2]*psi_cent_left_new[:,0]
             + weight[3]*psi_cent_left_new[:,1]

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

        # Check convergence
        if (crit < 0.0001):
            converge = True
```

```
76
77          # Update fluxes
78          psi_cent_right = psi_cent_right_new
79          psi_cent_left = psi_cent_left_new
80          phi = phi_new
81
82          # Increment iteration number
83          itr += 1
84
85      # How many iterations did we do?
86      print '(mesh spacing = %.3f) Number of iterations = %i' % (d,itr)
87
88      # Plot angular flux
89      plt.subplot(5,2,2*loopnum-1)
90      plt.subplots_adjust(hspace = 0.0, wspace=0.2)
91      plt.plot(x,psi_cent_right[:,0], marker="o", linestyle='none',label='$\mu=0.2$')
92      plt.plot(x,psi_cent_right[:,1], marker="o", linestyle='none', label='$\mu=0.7$')
93      plt.plot(x,psi_cent_left[:,0], marker="o", linestyle='none',label='$\mu=-0.2$')
94      plt.plot(x,psi_cent_left[:,1], marker="o", linestyle='none', label='$\mu=-0.7$')
95      plt.xlim(-0.1,2.4)
96      plt.ylim(np.min(psi_cent_left)-0.5*np.max(psi_cent_left),np.max(psi_cent_right)+0.3*np.max(psi_cent_right))
97      plt.xlabel('x'), plt.ylabel('Center $\psi$, ($\Delta_i$=%.2f)\n(Iterations = %i)' % (d,itr))
98      if (loopnum == 1): plt.title('Cell Center Angular Flux Profile, $\\alpha$=%0.2f' % a)
99      plt.legend(numpoints=1,fontsize=10)
100
101     # Plot scalar flux
102     plt.subplot(5,2,2*loopnum)
103     plt.plot(x,phi, marker='s', color='c',linestyle='none')
104     plt.xlim(-0.1,2.1)
105     plt.ylim(np.min(phi)-0.3*np.max(phi),np.max(phi)+0.3*np.max(phi))
106     plt.xlabel('x'), plt.ylabel('Center $\phi$, ($\Delta_i$=%.2f)\n(Iterations = %i)' % (d,itr))
107     if (loopnum == 1): plt.title('Cell Center Scalar Flux Profile, $\\alpha$=%.2f' % a)
108
109     # Increment loopnum
110     loopnum += 1
111
112 # Render plots
113 plt.show()
```

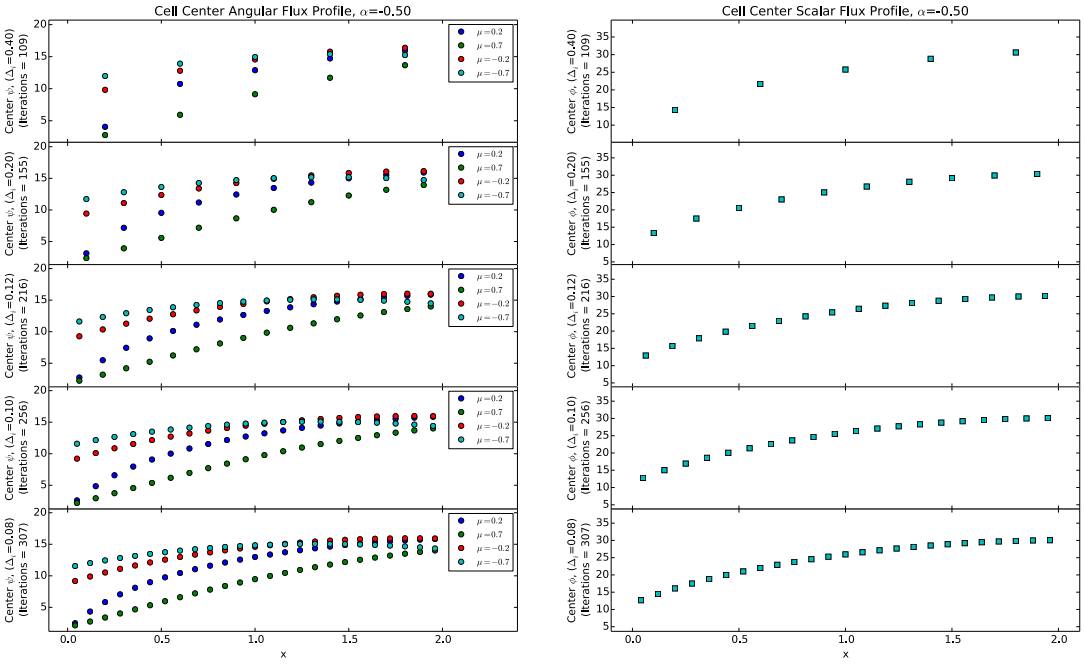

Fig. 8. Center angular and scalar flux profiles for varying mesh spacings with $\alpha = -0.5$, with scattering ($\Sigma_s = 0.5$) and external source ($q_e = 1$).

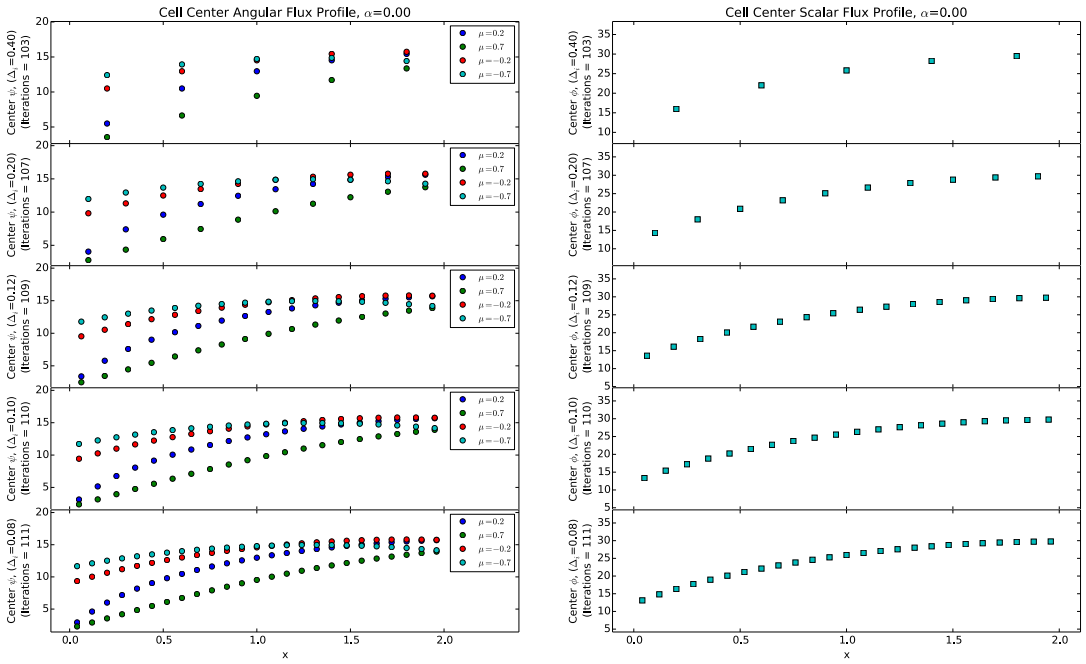

Fig. 9. Center angular and scalar flux profiles for varying mesh spacings with $\alpha = 0$, with scattering ($\Sigma_s = 0.5$) and external source ($q_e = 1$).

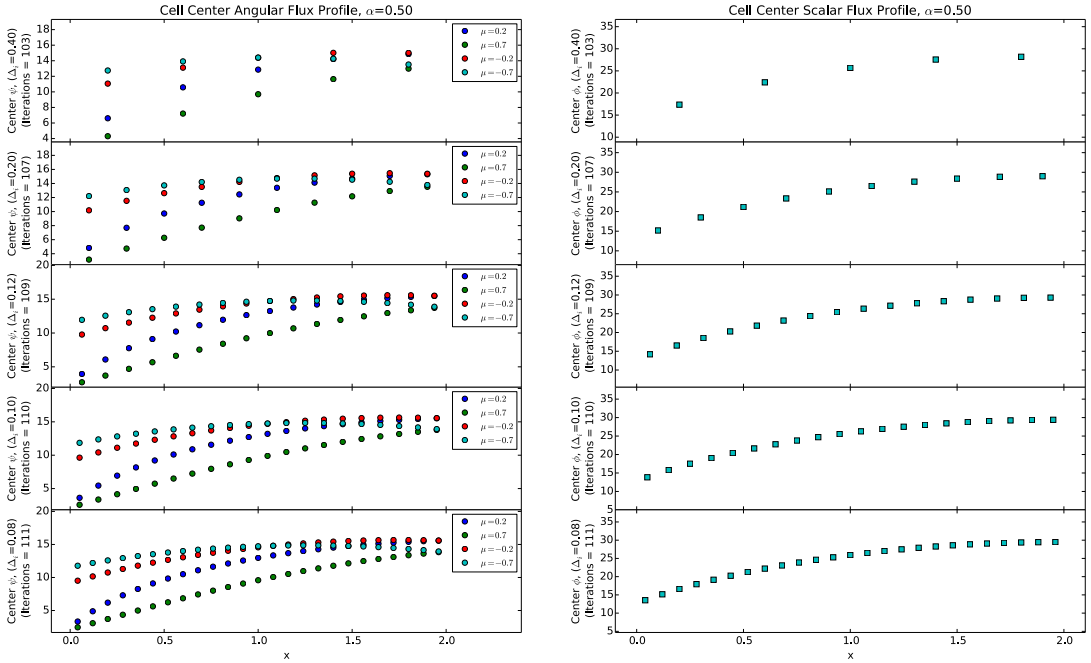

Fig. 10. Center angular and scalar flux profiles for varying mesh spacings with $\alpha = 0.5$, with scattering ($\Sigma_s = 0.5$) and external source ($q_e = 1$).

We see that the value of α did not make much of a difference in the results. But it did however have an effect on the required number of iterations before convergence (at least moving away from $\alpha = -0.5$) (see the y-axes in the plots for the number of iterations). We also notice that the shape of the flux is dramatically different than in part A and B. With scattering and a constant source included, the particles make it much further into the material and have a peak flux near the reflecting boundary. We then see a drop off (still nonzero though) near the vacuum boundary due to neutrons leaving the system and not returning (we do have a source of right-moving neutrons at the vacuum boundary, but its value is much lower than the flux in the other cells).

(d) What happens with $\alpha = 0$ and $\Sigma_s = 0.9$?

If we rerun the above code with these inputs, Python indicates that an overflow has occurred (for all mesh spacings). If we profile the code a bit, we see that the flux keeps increasing with every iteration. The values eventually get larger than the declared variable type can handle, thus it reports an overflow. With scattering essentially being the only interaction (Σ_s close to Σ_t), it seems our simple model is unable to converge on a flux solution. Perhaps we need a more sophisticated quadrature set, include higher scattering moments, or use a finer mesh spacing?

Problem 4
Starting from the following general system of equations

$$\frac{\mu_a}{h_i}(\psi^g_{a,i+1/2} - \psi^g_{a,i-1/2}) + \Sigma^g_{t,i}\psi^g_{a,i} = 2\pi \sum_{a=1}^{N} w_a \sum_{g'=1}^{G} \Sigma^{gg'}_{s,i}(a' \to a)\psi^{g'}_{a',i} + \frac{\chi_g}{2} \sum_{g'=1}^{G} \nu_{g'}\Sigma^{g'}_{f,i}\phi^{g'}_i + \frac{1}{2}Q^g_i \,,$$

where ϕ *is* scalar flux, write a set of five coupled equations for a five group problem. Assume neutrons can only downscatter from fast groups (1 and 2) to thermal groups (3, 4, and 5). Assume that thermal groups can upscatter into other thermal groups and can downscatter. Assume there is an external source and a fission source.

$\Rightarrow$ From the wording of the question, we assume the following about scattering: Neutrons in group 1 can downscatter to groups 3, 4, and 5 (it says fast groups can only downscatter to thermal groups, so group 1 cannot scatter to group 2). Neutrons in group 2 can downscatter into groups 3, 4, and 5. Neutrons in group 3 can downscatter into groups 4 and 5. Neutrons in group 4 can downscatter into group 5 and upscatter into group 3. Neutrons in group 5 can upscatter into groups 3 and 4. Neutrons in each group can also participate in in-group scattering ($1 \to 1$, $3 \to 3$, $5 \to 5$, etc.). We also assume the following about fission: Neutrons can undergo fission in any group (which is a bit unphysical), and fission neutrons can be born into any group (which is quite unphysical). Finally, we assume there is an external neutron source in every group. Expanding out the equation above for each group (not expanding over angles), we arrive at the following

g = 1:

$$\frac{\mu_a}{h_i}(\psi^1_{a,i+1/2} - \psi^1_{a,i-1/2}) + \Sigma^1_{t,i}\psi^1_{a,i} = 2\pi \sum_{a=1}^{N} w_a \left[\Sigma^{1\to1}_{s,i}(a' \to a)\psi^1_{a',i}\right] + \frac{\chi_1}{2}[\nu_1\Sigma^1_{f,i}\phi^1_i + \nu_2\Sigma^2_{f,i}\phi^2_i + \nu_3\Sigma^3_{f,i}\phi^3_i + \nu_4\Sigma^4_{f,i}\phi^4_i + \nu_5\Sigma^5_{f,i}\phi^5_i] + \frac{1}{2}Q^1_i$$

g = 2:

$$\frac{\mu_a}{h_i}(\psi^2_{a,i+1/2} - \psi^2_{a,i-1/2}) + \Sigma^2_{t,i}\psi^2_{a,i} = 2\pi \sum_{a=1}^{N} w_a \left[\Sigma^{2\to2}_{s,i}(a' \to a)\psi^2_{a',i}\right] + \frac{\chi_2}{2}[\nu_1\Sigma^1_{f,i}\phi^1_i + \nu_2\Sigma^2_{f,i}\phi^2_i + \nu_3\Sigma^3_{f,i}\phi^3_i + \nu_4\Sigma^4_{f,i}\phi^4_i + \nu_5\Sigma^5_{f,i}\phi^5_i] + \frac{1}{2}Q^2_i$$

g = 3:

$$\frac{\mu_a}{h_i}(\psi^3_{a,i+1/2} - \psi^3_{a,i-1/2}) + \Sigma^3_{t,i}\psi^3_{a,i} = 2\pi \sum_{a=1}^{N} w_a[\Sigma^{1\to3}_{s,i}(a' \to a)\psi^1_{a',i} + \Sigma^{2\to3}_{s,i}(a' \to a)\psi^2_{a',i} + \Sigma^{3\to3}_{s,i}(a' \to a)\psi^3_{a',i} + \Sigma^{4\to3}_{s,i}(a' \to a)\psi^4_{a',i} + \Sigma^{5\to3}_{s,i}(a' \to a)\psi^5_{a',i}] + \frac{\chi_3}{2}[\nu_1\Sigma^1_{f,i}\phi^1_i + \nu_2\Sigma^2_{f,i}\phi^2_i + \nu_3\Sigma^3_{f,i}\phi^3_i + \nu_4\Sigma^4_{f,i}\phi^4_i + \nu_5\Sigma^5_{f,i}\phi^5_i] + \frac{1}{2}Q^3_i$$

g = 4:

$$\frac{\mu_a}{h_i}(\psi^4_{a,i+1/2} - \psi^4_{a,i-1/2}) + \Sigma^4_{t,i}\psi^4_{a,i} = 2\pi \sum_{a=1}^{N} w_a[\Sigma^{1\to4}_{s,i}(a' \to a)\psi^1_{a',i} + \Sigma^{2\to4}_{s,i}(a' \to a)\psi^2_{a',i} + \Sigma^{3\to4}_{s,i}(a' \to a)\psi^3_{a',i} + \Sigma^{4\to4}_{s,i}(a' \to a)\psi^4_{a',i} + \Sigma^{5\to4}_{s,i}(a' \to a)\psi^5_{a',i}] + \frac{\chi_4}{2}[\nu_1\Sigma^1_{f,i}\phi^1_i + \nu_2\Sigma^2_{f,i}\phi^2_i + \nu_3\Sigma^3_{f,i}\phi^3_i + \nu_4\Sigma^4_{f,i}\phi^4_i + \nu_5\Sigma^5_{f,i}\phi^5_i] + \frac{1}{2}Q^4_i$$

g = 5:

$$\frac{\mu_a}{h_i}(\psi^5_{a,i+1/2} - \psi^5_{a,i-1/2}) + \Sigma^5_{t,i}\psi^5_{a,i} = 2\pi \sum_{a=1}^{N} w_a[\Sigma^{1\to5}_{s,i}(a' \to a)\psi^1_{a',i} + \Sigma^{2\to5}_{s,i}(a' \to a)\psi^2_{a',i} + \Sigma^{3\to5}_{s,i}(a' \to a)\psi^3_{a',i}$$
$$+\Sigma^{4\to5}_{s,i}(a' \to a)\psi^4_{a',i} + \Sigma^{5\to5}_{s,i}(a' \to a)\psi^5_{a',i}] + \frac{\chi_5}{2}[\nu_1\Sigma^1_{f,i}\phi^1_i + \nu_2\Sigma^2_{f,i}\phi^2_i + \nu_3\Sigma^3_{f,i}\phi^3_i$$
$$+\nu_4\Sigma^4_{f,i}\phi^4_i + \nu_5\Sigma^5_{f,i}\phi^5_i] + \frac{1}{2}Q^5_i$$