NE155 Homework 3 - Munis Thahir

In [ ]:
```
import numpy as np
import math
import matplotlib as plt
```

Question 1

Part A

```
We imagine the mesh cube centered on the point (i,j,k). When analyzing the flux
through this cube, a positive mu value indicates that the flux from the
negative half-interval -- that is, psi(i-1/2, j, k) -- is incoming, while the
flux from the positive half-interval is outgoing.
```

Part B

```
We imagine the mesh cube centered on the point (i,j,k). When analyzing the flux
through this cube, a negative mu value indicates that the flux from the
positive half-interval -- that is, psi(i+1/2, j, k) -- is incoming, while the
flux from the negative half-interval is outgoing.
```

Part C

```
When the outgoing flux is less than zero, we set the face-edge flux (in this
case, the positive half-interval on one axis) to zero. From here, we
recalculate the flux [psi(i,j,k)], and the new edge fluxes. We repeat the
process until all the outward fluxes are greater than or equal to zero.
```

Question 2

(Note: I was unable to debug my code in time, but I followed the algorithm in the 10/25 Lecture as best as I could in the hopes I could earn partial credit.)

Part A, mu = .1

In [ ]:
```
###Initialize Values
xarray = np.linspace(0,2,5)
xlen = len(xarray)
mu = .1
flux0 = 2
Sigt = 1.0                                          #Sigma-t value
Sigs = 0                                            #Sigma-s value
Qext = 0                                            #Source Value
h = [.08, .1, .125, .2, .4]                         #mesh spacing
###
c = 0                                               #Not Converged. When c = 1, we hav
CenterFlux0 = flux0
while c == 0:                                       #Operate until convergence
    for i in range(xlen):
            Qt = Qext + (Sigs*CenterFlux0)      #Sigs = Q = 0, Qtotal = Qext + Si
            CenterFlux1 = (Q + 2*(mu/h)*CenterFlux0)/(Sigt +(2*(mu/h)))
            PosHalfFlux = 2*CenterFlux1 - CenterFlux0
#Convergence Check
    PHI = np.sqrt(1/(1-xarray[i]^2))*CenterFlux1
    p = np.martix.max(np.linalg.eig(PHI,0))     #returns highest-value eigenvalue
    if p < 1:
        c = 1 #convergent!
    else:
        c = 0

plt.plot(h, CenterFlux1)
```

Part A, mu = -.1

In [ ]:
```
###Initialize Values
xarray = np.linspace(0,2,5)
xlen = len(xarray)
mu = -.1
flux0 = -2
Sigt = 1.0                                          #Sigma-t value
Sigs = 0                                            #Sigma-s value
Qext = 0                                            #Source Value
h = [.08, .1, .125, .2, .4]                         #mesh spacing
###
c = 0                                               #Not Converged. When c = 1, we hav
CenterFlux0 = flux0
while c == 0:                                       #Operate until convergence
    for i in range(xlen):
            Qt = Qext + (Sigs*CenterFlux0)      #Sigs = Q = 0, Qtotal = Qext + Si
            CenterFlux1 = (Q + 2*(mu/h)*CenterFlux0)/(Sigt +(2*(mu/h)))
            PosHalfFlux = 2*CenterFlux1 - CenterFlux0
#Convergence Check
    PHI = np.sqrt(1/(1-xarray[i]^2))*CenterFlux1
    p = np.martix.max(np.linalg.eig(PHI,0))     #returns highest-value eigenvalue
    if p < 1:
        c = 1 #convergent!
    else:
        c = 0

plt.plot(h, CenterFlux1)
```

```
Part B, mu = .1
```

In [ ]:
```
###Initialize Values
xarray = np.linspace(0,2,5)
xlen = len(xarray)
mu = .1
flux0 = 2
Sigt = 1.0                                          #Sigma-t value
Sigs = 0                                            #Sigma-s value
Qext = 0                                            #Source Value
h = [.08, .1, .125, .2, .4]                         #mesh spacing
a = [-.9, -.5, .25, .5, .9]
###
c = 0                                               #Not Converged. When c = 1, we hav
CenterFlux0 = flux0
while c == 0:                                       #Operate until convergence
    for i in range(xlen):
        if a[i] > 0:
            Qt = Qext + (Sigs*CenterFlux0)      #Sigs = Q = 0, Qtotal = Qext + Si
            CenterFlux1 = (Q + (2/(1+a[i])*(mu/h)*CenterFlux0)/(Sigt +((2/(1+a[i]
            PosHalfFlux = (2/(1+a[i]))*CenterFlux1 - ((1-a[i])/(1+a[i]))*CenterFl
        if a[i] < 0:
            Qt = Qext + (Sigs*CenterFlux0)      #Sigs = Q = 0, Qtotal = Qext + Si
            CenterFlux1 = (Q + (2/(1+a[i])*(mu/h)*CenterFlux0)/(Sigt +((2/(1+a[i]
            PosHalfFlux = (2/(1+a[i]))*CenterFlux1 - ((1-a[i])/(1+a[i]))*CenterFl
#Convergence Check
    PHI = np.sqrt(1/(1-xarray[i]^2))*CenterFlux1
    p = np.martix.max(np.linalg.eig(PHI,0))     #returns highest-value eigenvalue
    if p < 1:
        c = 1 #convergent!
    else:
        c = 0

plt.plot(h, CenterFlux1)
```

Part B, mu = -.1

In [ ]:
```
###Initialize Values
xarray = np.linspace(0,2,5)
xlen = len(xarray)
mu = -.1
flux0 = -2
Sigt = 1.0                                          #Sigma-t value
Sigs = 0                                            #Sigma-s value
Qext = 0                                            #Source Value
h = [.08, .1, .125, .2, .4]                         #mesh spacing
a = [-.9, -.5, .25, .5, .9]
###
c = 0                                               #Not Converged. When c = 1, we hav
CenterFlux0 = flux0
while c == 0:                                       #Operate until convergence
    for i in range(xlen):
        if a[i] > 0:
            Qt = Qext + (Sigs*CenterFlux0)      #Sigs = Q = 0, Qtotal = Qext + Si
            CenterFlux1 = (Q + (2/(1+a[i])*(mu/h)*CenterFlux0)/(Sigt +((2/(1+a[i]
            PosHalfFlux = (2/(1+a[i]))*CenterFlux1 - ((1-a[i])/(1+a[i]))*CenterFl
        if a[i] < 0:
            Qt = Qext + (Sigs*CenterFlux0)      #Sigs = Q = 0, Qtotal = Qext + Si
            CenterFlux1 = (Q + (2/(1+a[i])*(mu/h)*CenterFlux0)/(Sigt +((2/(1+a[i]
            PosHalfFlux = (2/(1+a[i]))*CenterFlux1 - ((1-a[i])/(1+a[i]))*CenterFl
#Convergence Check
    PHI = np.sqrt(1/(1-xarray[i]^2))*CenterFlux1
    p = np.martix.max(np.linalg.eig(PHI,0))     #returns highest-value eigenvalue
    if p < 1:
        c = 1 #convergent!
    else:
        c = 0

plt.plot(h, CenterFlux1)
```

Part C, mu = .2

In [ ]:
```
###Initialize Values
xarray = np.linspace(0,2,5)
xlen = len(xarray)
mu = .2
flux0 = 2
Sigt = 1.0                                          #Sigma-t value
Sigs = 0.5                                          #Sigma-s value
Qext = 1                                            #Source Value
h = [.08, .1, .125, .2, .4]                         #mesh spacing
a = [-.5, 0, .5]

###
c = 0                                               #Not Converged. When c = 1, we hav
CenterFlux0 = flux0
while c == 0:                                       #Operate until convergence
    for i in range(len(a)):
        if a[i] > 0:
            Qt = Qext + (Sigs*CenterFlux0)      #Sigs = Q = 0, Qtotal = Qext + Si
            CenterFlux1 = (Q + (2/(1+a[i])*(mu/h)*CenterFlux0)/(Sigt +((2/(1+a[i]
            PosHalfFlux = (2/(1+a[i]))*CenterFlux1 - ((1-a[i])/(1+a[i]))*CenterFl
        if a[i] < 0:
            Qt = Qext + (Sigs*CenterFlux0)      #Sigs = Q = 0, Qtotal = Qext + Si
            CenterFlux1 = (Q + (2/(1+a[i])*(mu/h)*CenterFlux0)/(Sigt +((2/(1+a[i]
            PosHalfFlux = (2/(1+a[i]))*CenterFlux1 - ((1-a[i])/(1+a[i]))*CenterFl

#Convergence Check
    PHI = np.sqrt(1/(1-xarray[i]^2))*CenterFlux1
    p = np.martix.max(np.linalg.eig(PHI,0))     #returns highest-value eigenvalue
    if p < 1:
        c = 1 #convergent!
    else:
        c = 0

plt.plot(h, CenterFlux1)
```

Part C, mu = -.2

In [ ]:
```
###Initialize Values
xarray = np.linspace(0,2,5)
xlen = len(xarray)
mu = -.2
flux0 = -2
Sigt = 1.0                                          #Sigma-t value
Sigs = 0.5                                          #Sigma-s value
Qext = 1                                            #Source Value
h = [.08, .1, .125, .2, .4]                         #mesh spacing
a = [-.5, 0, .5]

###
c = 0                                               #Not Converged. When c = 1, we hav
CenterFlux0 = flux0
while c == 0:                                       #Operate until convergence
    for i in range(len(a)):
        if a[i] > 0:
            Qt = Qext + (Sigs*CenterFlux0)      #Sigs = Q = 0, Qtotal = Qext + Si
            CenterFlux1 = (Q + (2/(1+a[i])*(mu/h)*CenterFlux0)/(Sigt +((2/(1+a[i]
            PosHalfFlux = (2/(1+a[i]))*CenterFlux1 - ((1-a[i])/(1+a[i]))*CenterFl
        if a[i] < 0:
            Qt = Qext + (Sigs*CenterFlux0)      #Sigs = Q = 0, Qtotal = Qext + Si
            CenterFlux1 = (Q + (2/(1+a[i])*(mu/h)*CenterFlux0)/(Sigt +((2/(1+a[i]
            PosHalfFlux = (2/(1+a[i]))*CenterFlux1 - ((1-a[i])/(1+a[i]))*CenterFl

#Convergence Check
    PHI = np.sqrt(1/(1-xarray[i]^2))*CenterFlux1
    p = np.martix.max(np.linalg.eig(PHI,0))     #returns highest-value eigenvalue
    if p < 1:
        c = 1 #convergent!
    else:
        c = 0

plt.plot(h, CenterFlux1)
```

Part C, mu = .7

In [ ]:
```
###Initialize Values
xarray = np.linspace(0,2,5)
xlen = len(xarray)
mu = .7
flux0 = 2
Sigt = 1.0                                          #Sigma-t value
Sigs = 0.5                                          #Sigma-s value
Qext = 1                                            #Source Value
h = [.08, .1, .125, .2, .4]                         #mesh spacing
a = [-.5, 0, .5]

###
c = 0                                               #Not Converged. When c = 1, we hav
CenterFlux0 = flux0
while c == 0:                                       #Operate until convergence
    for i in range(len(a)):
        if a[i] > 0:
            Qt = Qext + (Sigs*CenterFlux0)      #Sigs = Q = 0, Qtotal = Qext + Si
            CenterFlux1 = (Q + (2/(1+a[i])*(mu/h)*CenterFlux0)/(Sigt +((2/(1+a[i]
            PosHalfFlux = (2/(1+a[i]))*CenterFlux1 - ((1-a[i])/(1+a[i]))*CenterFl
        if a[i] < 0:
            Qt = Qext + (Sigs*CenterFlux0)      #Sigs = Q = 0, Qtotal = Qext + Si
            CenterFlux1 = (Q + (2/(1+a[i])*(mu/h)*CenterFlux0)/(Sigt +((2/(1+a[i]
            PosHalfFlux = (2/(1+a[i]))*CenterFlux1 - ((1-a[i])/(1+a[i]))*CenterFl

#Convergence Check
    PHI = np.sqrt(1/(1-xarray[i]^2))*CenterFlux1
    p = np.martix.max(np.linalg.eig(PHI,0))     #returns highest-value eigenvalue
    if p < 1:
        c = 1 #convergent!
    else:
        c = 0

plt.plot(h, CenterFlux1)
```

In [ ]:
```
Part C, mu = -.7
```

In [ ]:
```
###Initialize Values
xarray = np.linspace(0,2,5)
xlen = len(xarray)
mu = -.7
flux0 = 2
Sigt = 1.0                                          #Sigma-t value
Sigs = 0.5                                          #Sigma-s value
Qext = 1                                            #Source Value
h = [.08, .1, .125, .2, .4]                         #mesh spacing
a = [-.5, 0, .5]

###
c = 0                                               #Not Converged. When c = 1, we hav
CenterFlux0 = flux0
while c == 0:                                       #Operate until convergence
    for i in range(len(a)):
        if a[i] > 0:
            Qt = Qext + (Sigs*CenterFlux0)      #Sigs = Q = 0, Qtotal = Qext + Si
            CenterFlux1 = (Q + (2/(1+a[i])*(mu/h)*CenterFlux0)/(Sigt +((2/(1+a[i]
            PosHalfFlux = (2/(1+a[i]))*CenterFlux1 - ((1-a[i])/(1+a[i]))*CenterFl
        if a[i] < 0:
            Qt = Qext + (Sigs*CenterFlux0)      #Sigs = Q = 0, Qtotal = Qext + Si
            CenterFlux1 = (Q + (2/(1+a[i])*(mu/h)*CenterFlux0)/(Sigt +((2/(1+a[i]
            PosHalfFlux = (2/(1+a[i]))*CenterFlux1 - ((1-a[i])/(1+a[i]))*CenterFl

#Convergence Check
    PHI = np.sqrt(1/(1-xarray[i]^2))*CenterFlux1
    p = np.martix.max(np.linalg.eig(PHI,0))     #returns highest-value eigenvalue
    if p < 1:
        c = 1 #convergent!
    else:
        c = 0

plt.plot(h, CenterFlux1)
```

Question 3

See Following Document

NE155 - Munis Thakur

Homework 5

3a) $\underline{\underline{L}}\psi + \underline{\underline{M}}\underline{\underline{S}}\phi + \underline{\underline{M}}Q \Leftrightarrow (\alpha \times \alpha)\psi = (\alpha \times \beta)(\beta \times \beta)\phi + (\alpha \times \beta)Q$

α = Number of Energy Groups + # of angular unknowns + # cells + # $\frac{\text{unknowns}}{\text{cell}}$

β = # Energy Groups + # moments + # of cells + # $\frac{\text{unknowns}}{\text{cell}}$

$\hookrightarrow \alpha = 3 + 8 + 64 + 10 = 85$

$\beta = 3 + 9 + 64 + 10 = 86$

} The 10 unknowns due to WDD are $\Sigma t, n, m; \alpha, \Delta i, \Delta j, \Delta k$.

$\therefore$ L is (85x85), M is (85x86), and S is (86x86).

3b) $\begin{bmatrix} M_{00} & 0 & 0 \\ 0 & M_{11} & 0 \\ 0 & 0 & M_{22} \end{bmatrix} = M; \begin{bmatrix} [S_{0\to 0}] & [S_{1\to 0}] & [S_{2\to 0}] \\ [S_{0\to 1}] & [S_{1\to 1}] & [S_{2\to 1}] \\ [S_{0\to 2}] & [S_{1\to 2}] & [S_{2\to 2}] \end{bmatrix} = S$

$S_{1\to 2} = \begin{bmatrix} \Sigma_{s_0}^{1\to 2} & 0 & \cdots & 0 \\ 0 & \Sigma_{s_1}^{1\to 2} & & \vdots \\ 0 & & \ddots & \vdots \\ \vdots & 0 & \cdots & \Sigma_q^{1\to 2} \end{bmatrix}$ ← # Moments

$\psi = \begin{bmatrix} [\psi_0] \\ [\psi_1] \\ [\psi_2] \end{bmatrix}, \quad [\psi]_1 = \begin{bmatrix} \psi_{1,3} \\ \psi_{2,3} \\ \vdots \\ \psi_{9,3} \end{bmatrix}$

$[\phi]_1 = [\phi_{00}^1 \ \phi_{10}^1 \ v_{11}^1 \ \phi_{11} \ \phi_{20}^1 \ \phi_{21}^1 \ v_{22}^1 \ \phi_{22} \cdots v_{99}^1 \ \phi_{99}^1]^T$

3c) $D = M^T W = \sum_{a=1}^{n} Y_{lm}^{e/o} w_a = \sum_{a=1}^{8} Y_{lm} w_a$

3d) We don't form an L matrix because we prefer to view the indices of L as single-group equations that are only a function of space and angle.

3e) 1: Combine eqs. 1 and 2 $\to \underline{\underline{L}}\psi = \underline{\underline{M}}\underline{\underline{S}}[D\psi] + MQ$

2: Left-multiply L^{-1} to isolate ψ $\to \psi = L^{-1}MSD\psi + L^{-1}MQ$

3: Subtract left term from RHS $\to \psi - L^{-1}MSD\psi = L^{-1}MQ$

4: Factor ψ $\to (I - L^{-1}MSD)\psi = L^{-1}MQ$

5: Expand $Q = (I - DL^{-1}MS)\phi$ $\to (I - L^{-1}MSD)\psi = L^{-1}M(I - DL^{-1}MS)\phi$ [Eq. 4 from 10/23]

6: Divide by $(I - DL^{-1}MS)$ $\to \psi = L^{-1}MQ$

$\therefore$ This is now of the form $Ax = b$ where $A = L^{-1}M$, $x = \phi$, and $b = \psi$.