Homework 2

Simon Bolding
NUEN 629

October 8, 2015

NUEN 629, Homework 2

Due Date Oct. 6

1 Childs

(35 points) Compute three group cross-sections for a homogeneous mixture of graphite and natural uranium where the ratio of graphite to uranium is 150:1. You can assume the Watt-fission spectrum, and that the group bounds are $\{0, 1\,\text{eV}, 100\,\text{keV}, 20\,\text{MeV}\}$.

2 Franklin

(40 points) The enclosed file gives the microscopic cross-sections for ^{1}H in units of barns for 5 groups as calculated by the code `NJOY`. Imagine we have a large, nearly infinite tank of high-pressure hydrogen at 30 atm next to a bare sphere of ^{235}U. Compute the scalar flux ϕ_g and the current $\vec{J}_g$ in the hydrogen using the separable, P1 equivalent, and extended Legendre approximations. Compare your solutions graphically.

3 Geer

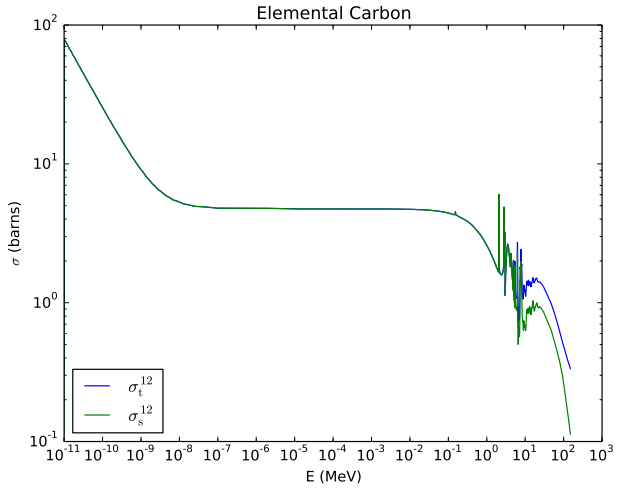
(25 points) Find the solution to the diffusion equation for 1-group, slab geometry with a uniform source, where the material is a pure scatter and the slab width is X under the following conditions

1. Vacuum Marshak conditions
2. Vacuum Mark conditions
3. Vacuum Dirichlet conditions
4. Vacuum Dirichlet condition on the left and albedo on the right at $X/2$, and
5. Vacuum Dirichlet condition on the left and reflecting on the right at $X/2$.

Compare the solutions and comment on the similarities and differences.

Solution 1:

Several approximations were made to simplify the process. First, graphite is approximated as elemental Carbon, with molar mass 12.0107 (g/mol). This was done because there is not a human friendly form of the graphite cross sections available on NNDC and elemental Carbon will have similar scattering properties, except for at low energies where diffraction is possible. A plot of the elastic and total cross sections for elemental Carbon from NNDC are given below.



The ratio of 150:1 for graphite to natural uranium is assumed to be an of atomic ratio. Natural uranium is taken to be 0.72% ^{235}U and the remainder ^{238}U, by atom percentage. The total cross section is assumed to only consist of elastic scattering, fission, and removal events. Since this system is scattering dominated, it is probably more accurate to use a Maxwellian distribution for the flux at thermal energies.

We follow a similar procedure to the one in lab. For an infinite medium, with fine-group cross sections, the balance equation becomes

$$N^U \sum_j \gamma_j \sigma_t^j(E)\psi(\mu,E) = N^U \sum_j \gamma_j \frac{1}{2}\int_0^\infty \mathrm{d}E' \sigma_s^j(E'\to E)\phi(E') + N^U \frac{\chi(E)}{2k}\int_0^\infty \mathrm{d}E' \left(\gamma_{238}\overline{\nu}\sigma_f^{238} + \gamma_{235}\overline{\nu}\sigma_f^{235}\right)\phi(E'), \quad (1)$$

where j indicates the j-th isotope, N^U is the atom density of natural uranium in the system, and γ_j is 150/151, 0.9928/151, and 0.0072/151 for Carbon, ^{238}U, and ^{235}U, respectively. It is assumed $\chi(E)$ is the same for ^{238}U and ^{235}U, given by the Watt spectrum from class

$$\chi(E) = 0.4865\sinh(\sqrt{2E})e^{-E}. \quad (2)$$

We now simplify by normalizing such that the energy integrated fission source has a magnitude of 1. We also assume all scattering events result in the average scattering energy loss,

which, assuming isotropic scattering in the center of mass frame, gives an average outgoing energy of

$$\langle E \rangle = \frac{A^2+1}{(A+1)^2} E' \tag{3}$$

in the lab frame. With this simplificiation, only a particular E' governed by the above equation can scatter into E, so the the elastic scattering source for the j-th term in the summation can be simplified as

$$\int_0^\infty \mathrm{d}E' \sigma_s^j(E') P(E' \to E) \phi(E') = \int_0^\infty \mathrm{d}E' \sigma_s^j(E') \delta \left(E' - E \frac{E'}{\langle E \rangle} \right) \phi(E') \tag{4}$$

$$= \sigma_s^j \left(\frac{(A+1)^2}{A^2+1} E \right) \phi \left(\frac{(A+1)^2}{A^2+1} E \right) \tag{5}$$

$$\tag{6}$$

where A is the atomic mass number for the j-th isotope, approximated as 12.0107 for elemental carbon. Substituting back into the original equation and integrating over angle gives the final equation for the scalar flux as

$$\sum_j \gamma_j \sigma_t^j(E) \phi(E) = \sum_j \gamma_j \sigma_s^j \left(\frac{(A+1)^2}{A^2+1} E \right) \phi \left(\frac{(A+1)^2}{A^2+1} E \right) + \chi(E). \tag{7}$$

We solve this equation with the Jacobi iteration

$$\sum_j \gamma_j \sigma_t^j(E) \phi^{(k)}(E) = \sum_j \gamma_j \sigma_s^j \left(\frac{(A+1)^2}{A^2+1} E \right) \phi^{(k-1)} \left(\frac{(A+1)^2}{A^2+1} E \right) + \chi(E). \tag{8}$$

with an initial guess of $\phi^{(0)}(E) = 0$. To approximate the continuous energy cross sections and $\phi(E)$ we simply evaluate the above iteration at each of the energy points of the fine group cross sections. The points are defined using the union of the total cross section energy grids of all isotopes. A linear interpolation (python interp1D default interpolation) is used between energy points when one cross section is coarser than others. For evaluation above the maximum energy for a given cross section, the value of the cross section at the maximum energy is used. All values of cross sections and flux above 20 MeV were ignored.

Once $\phi(E)$ is obtained, the collapsed cross sections are computed as

$$\sigma_{n,g} = \frac{\int_{E_g}^{E_{g-1}} \mathrm{d}E \sigma_n^j(E) \sum \phi(E)}{\int_{E_g}^{E_{g-1}} \mathrm{d}E \phi(E)} \tag{9}$$

for the g-th group, j-th isotope, and n-th reaction type. The integral is approximated with midpoint quadrature.

In Fig. 1 the obtained solution for the fully converged spectrum is plotted against the initial uncollided spectrum, where both are normalized relative to the integral of the converged $\phi(E)$. As shown, the spectrum demonstrates significant moderation to lower energies due to the large amount of graphite present. The microscopic collapsed group cross sections are given in Table 1. For comparison, I also computed a total thermal group collapsed

cross section using a Maxwellian distribution. The Maxwellian collapsed cross sections give $\Sigma_t = 5.04$ b and $\Sigma_t = 4.99$ b, indicating the flux is larger at lower energies for the reactor than the Watt-spectrum indicates, likely a result of the scattering treatment not allowing for upscattering at thermal energies. The python script used to compute the answers is given at the end of the assignment.

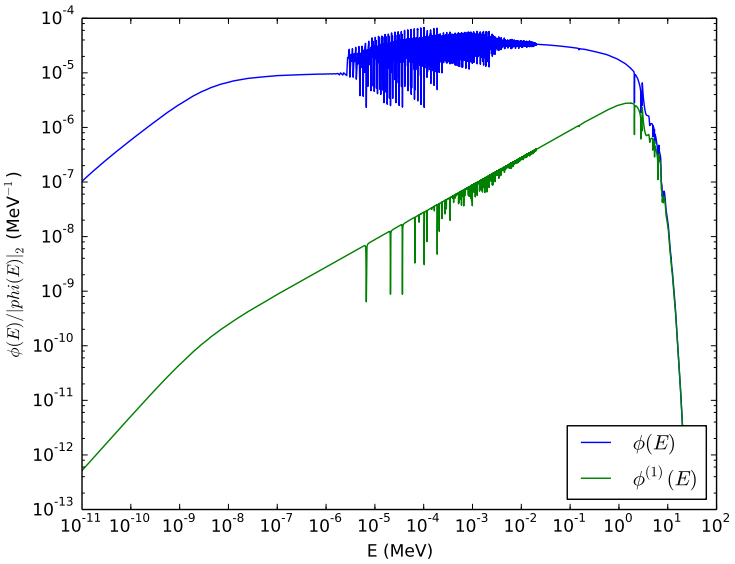


Figure 1: Comparison of scattering corrected and uncollided spectrum for infinite-medium mix of natural Uranium and elemental Carbon

Table 1: 3 Group cross section for 150:1 Carbon Uranium mix

Group	$\sigma_{t,g}$ (b)	$\sigma_{s,g}$ (b)
Carbon		
0	2.581	2.575
1	4.558	4.554
2	4.810	4.803
^{235}U		
0	7.755	4.543
1	13.68	10.60
2	171.8	13.96
^{238}U		
0	7.927	5.177
1	13.41	12.79
2	10.10	9.189
150:1 C to U mix		
0	2.617	2.592
1	4.617	4.609
2	4.853	4.832

Solution 2:

First, we need an atom density of the hydrogen. Based on ideal gas law, the density scales proportional to tempurature. At 1 atm, the density of hydrogen at room temperature is roughly 9.E-04 g cm^{-3}. Scaling by 30, gives a density of 0.0025 g cm^{-3}. This is in agreement to 1 significant digit with an online calculator that takes into account compressibility of hydrogen at 30 atm and room temperature. This gives a corresponding atom density of hydrogen as

$$N^H = 0.0025 \frac{\text{g}}{\text{cm}^3} \frac{1 \text{ mol } \text{H}_2}{2.02\text{g } \text{H}_2} \frac{2H}{1H_2} \frac{0.60221 \text{ atoms cm}^2}{\text{b}} \approx 0.00149 \frac{\text{H atoms}}{\text{b–cm}} \tag{10}$$

As an initial approximation, I assumed that within the semi-infinite Hydrogen medium, several mfp away from the source, the sphere of ^{235}U will appear as an isotropic boundary source to a 1D, semi-infinite medium. Following the notes, the general multigroup 1D transport equation we will be solving, in the Hydrogen, will have the form

$$\mu \frac{\partial \psi_g(z,\mu)}{\partial z} + \hat{\Sigma}_{tg} \psi_g = \sum_{l=0}^{\infty} \frac{2l+1}{2} P_l(\mu) \sum_{g'=0}^{G-1} \left[\Sigma_{slg' \to g} + \delta_{gg'} \left(\hat{\Sigma}_{tg'} - \Sigma_{tlg'} \right) \right] \phi_{lg'}(z) + q(\mu, z), \tag{11}$$

where $\hat{\Sigma}_{tg}$ is yet to be defined. Also, it is assumed that the spectrum of energies leaving the sphere of ^{235}U is well approximated by the fission emission energy spectrum $\chi(E)$ (i.e., only consider the uncollided energy spectrum). The group integrated $\chi(E)$ for ^{235}U is computed. All groups have a high scattering ratio of ≈ 0.9999. Thus, a few MFP away from the boundary source, we expect diffusion theory to be applicable, with essentially a pure scatter. Consider the 1-speed diffusion theory equation for a pure scatterer, in a semi-infinite medium, with no internal source

$$-D \frac{\text{d}^2 \phi}{\text{d}x^2} = 0. \tag{12}$$

The requirement that the flux be bounded at infinity requires a constant spatial solution, so we thus can neglect spatial gradients in Eq. (11). We choose to normalize the effective source of neutrons from the sphere such that the magnitude of the energy integrated source is 1. Thus, our transport equation to be solved becomes

$$\hat{\Sigma}_{t,g} \psi_g(\mu) = \sum_{l=0}^{\infty} \frac{2l+1}{2} P_l(\mu) \sum_{g'=0}^{G-1} \left[\Sigma^*_{slg' \to g} + \delta_{gg'} \left(\hat{\Sigma}_{tg'} - \Sigma_{tlg'} \right) \right] \phi_{lg'} + \frac{\chi_g}{2} \tag{13}$$

where the Σ^*_s will depend on the method. Taking the zeroth moment gives the equation for the scalar flux in each group as

$$\hat{\Sigma}_{t,g} \phi_g = \sum_{g'=0}^{G-1} \left[\Sigma_{s0g' \to g} + \delta_{gg'} \left(\hat{\Sigma}_{tg'} - \Sigma_{t0g'} \right) \right] \phi_{g'} + \chi_g \tag{14}$$

The first moment gives the equation for the current in each group as

$$\hat{\Sigma}_{t,g} J_g = \sum_{g'=0}^{G-1} \left[\Sigma_{s1g' \to g} + \delta_{gg'} \left(\hat{\Sigma}_{tg'} - \Sigma_{t1g} \right) \right] J'_g. \tag{15}$$

The only solution to the above equation is 0. So, based on our assumptions of an isotropic source, for all methods and all groups, $\boxed{J_g = 0}$.

The equation for the scalar flux can be written as a matrix equation as

$$\mathbf{S}\boldsymbol{\Phi} = \boldsymbol{\chi} \tag{16}$$

with matrix elements defined as

$$\chi_i = \chi_{g_i}, \quad g_i = 0, 1, \ldots, G-1 \tag{17}$$

$$\phi_i = \phi_{g_i}, \quad g_i = 0, 1, \ldots, G-1 \tag{18}$$

and

$$S_{ij} = \begin{cases} \hat{\Sigma}_{t,g_i} - \Sigma^*_{s0g_i \to g_i} & i = j \\ -\Sigma^*_{s0g_i \to g_j} & i \neq j \end{cases} \tag{19}$$

For the approximation of separable in energy, the effective cross sections become $\hat{\Sigma}_{t,g} = \Sigma_{t,g}$ and $\Sigma^*_{s0,g' \to g} = \Sigma_{s0,g' \to g}$. For the P_1 consistent form of the equation, we set $\hat{\Sigma}_{t,g} = \Sigma_{t0,g}$, which gives the same equations for scalar flux since we are only interested in the 0-th moment cross sections. For the extended Legendre expansion we want

$$\hat{\Sigma}_{tg} = \Sigma_{t,L+1,g} - \sum_{g'=0}^{G-1} \Sigma_{s,L+1,g \to g'} \tag{20}$$

where since we are interested in the scalar flux we will choose $L = 0$, giving

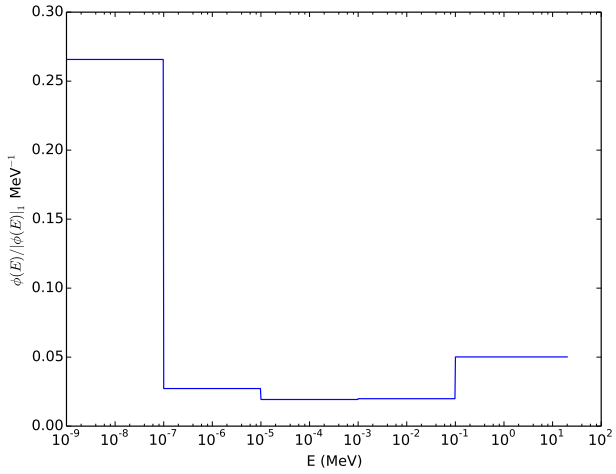
$$\hat{\Sigma}_{tg} = \Sigma_{t1,g} - \sum_{g'=0}^{G-1} \Sigma_{s1,g \to g'} \tag{21}$$

If we make the approximation

$$\Sigma_{t1,g} \approx \Sigma_{s1,g} := \sum_{g'=0}^{G-1} \Sigma_{s1,g \to g'} \tag{22}$$

then we get $\hat{\Sigma}_{tg} = 0$. This leads to an identical set of equations as the other two methods.

Solving the matrix system gives a solution for ϕ_g (the same for all three methods). A plot of the solutions in each group are compared below. The code is given at the end of the assignment


0.30
0.25
0.20
0.15
0.10
0.05
0.00
$\phi(E)/|\phi(E)|_1$ MeV^{-1}
10^{-9} 10^{-8} 10^{-7} 10^{-6} 10^{-5} 10^{-4} 10^{-3} 10^{-2} 10^{-1} 10^{0} 10^{1} 10^{2}
E (MeV)

#3 Solution

- Since a pure scatterer, D.E. reduces to (for 1speed, slabs)

$$-D\frac{d^2\phi}{dx^2} = Q \qquad x \in \left[\frac{-X}{2}, \frac{X}{2}\right] \qquad (1)$$

- w/ the general Brunner B.C.

$$A\phi + BD(\hat{n}\cdot\hat{i})\frac{\partial\phi}{\partial x} = C, \qquad x \in \partial V$$

- Solving for $\phi(x)$ on interior from (1):

$$\int -D\frac{d^2\phi}{dx^2}dx = \int Q\,dx$$

$$\frac{d\phi}{dx} = -\frac{Qx}{D} + K_1$$

$$\phi(x) = -\frac{Qx^2}{2D} + K_1 x + K_2 \qquad (2)$$

- For all B.C. of interest, in Brunner form, $C=0$:

$$A_L\phi\left(\frac{-X}{2}\right) - B_L D\frac{d\phi}{dx}\Big|_{x=\frac{-X}{2}} = 0 \qquad (3)$$

$$A_R\phi\left(\frac{X}{2}\right) + B_R D\frac{d\phi}{dx}\Big|_{x=\frac{X}{2}} = 0$$

- (2)→(3)

$$A_L\left(\frac{-QX^2}{8D} - \frac{K_1 X}{2} + K_2\right) + B_L\left(\frac{-QX}{2} - K_1 D\right) = 0$$

$$A_R\left(\frac{-QX^2}{8D} + \frac{K_1 X}{2} + K_2\right) + B_R\left(\frac{-QX}{2} + K_1 D\right) = 0$$

or: $$\left(\frac{A_L X}{2} + B_L D\right) K_1 - (A_L K_2) = -\left(\frac{A_L Q X^2}{8D} + \frac{B_L Q X}{2}\right)$$

$$\left(\frac{A_R X}{2} + B_R D\right) K_1 + (A_R K_2) = \frac{A_R Q X^2}{8D} + \frac{B_R Q X}{2}$$

- Solving algebraically

$$K_1 = \frac{-\left(\frac{A_L Q X^2}{8D} + \frac{B_L Q X}{2}\right) + \left(\frac{A_L}{A_R}\right)\left(\frac{A_R Q X^2}{8D} + \frac{B_R Q X}{2}\right)}{\left(\frac{A_L X}{2} + B_L D\right) + \left(\frac{A_L}{A_R}\right)\left(\frac{A_R X}{2} + B_R D\right)}$$

$$K_2 = \frac{\frac{A_L Q X^2}{8D} + \frac{B_L Q X}{2} + \left(\frac{\frac{A_L X}{2} + B_L D}{\frac{A_R X}{2} + B_R D}\right)\left(\frac{A_R Q X^2}{8D} + \frac{B_R Q X}{2}\right)}{A_L + \left(\frac{\frac{A_L X}{2} + B_L D}{\frac{A_R X}{2} + B_R D}\right) A_R}$$

- The above equations are evaluated and simplified for each case to get K_1 and K_2.

A summary of the solutions obtained for each of the boundary conditions is given in the table below. The figures below compares plots for the case of $Q = 1$, $D = 0.5$ and $X = 10$. The first figure demonstrates that Dirichlet (without use of the extrapolated BC) can be very inaccurate. The solution with Mark boundary conditions is lower in magnitude than for the Marshak case, which is expected due to symmetry and that the Mark boundary condition has a shorter extrapolation distance of $\sqrt{3}D$, compared to $2D$ for Marshak. For the chosen parameters there is noticeable variation between the different choices of BC. A smaller value of D or increased X will result in more consistent solutions, particularly on the interior of the domain. The second plot shows that albedo varies between a Marshak and Reflective condition. As expected, reflective-type BC's on the right side of the domain results in a much larger magnitude in the solution as leakage is reduced but the source strength is the same as the vacuum cases.

Table 2: Solutions with different boundary conditions for a pure scatter for slab of width X centered at $x = 0$.

Left BC	Right BC	$\phi(x)$
Vac. Marshak	Vacuum Marshak	$\phi(x) = Q\left(\frac{X^2}{8D} + X - \frac{x^2}{2D}\right)$
Vac. Mark	Vacuum Marshak	$\phi(x) = Q\left(\frac{X^2}{8D} + \frac{X\sqrt{3}}{2} - \frac{x^2}{2D}\right)$
Vac. Dirichlet	Vacuum Dirichlet	$\phi(x) = \frac{Q}{2D}\left(\frac{X^2}{4} - x^2\right)$
Vac. Dirichlet	Albedo	$\phi(x) = -\frac{Qx^2}{2D} + QxX\left(\frac{1+\frac{(1-\alpha)}{2(1+\alpha)}\frac{X}{2D}}{\frac{(1-\alpha)}{2(1+\alpha)}X+D} - \frac{1}{2D}\right) + Q\frac{X^2}{2}\left(\frac{1+\frac{(1-\alpha)}{2(1+\alpha)}\frac{X}{2D}}{\frac{(1-\alpha)}{2(1+\alpha)}X+D} - \frac{1}{4D}\right)$
Vac. Dirichlet	Reflecting	$\phi(x) = \frac{Q}{2D}\left(\frac{3X^2}{4} + xX - x^2\right)$

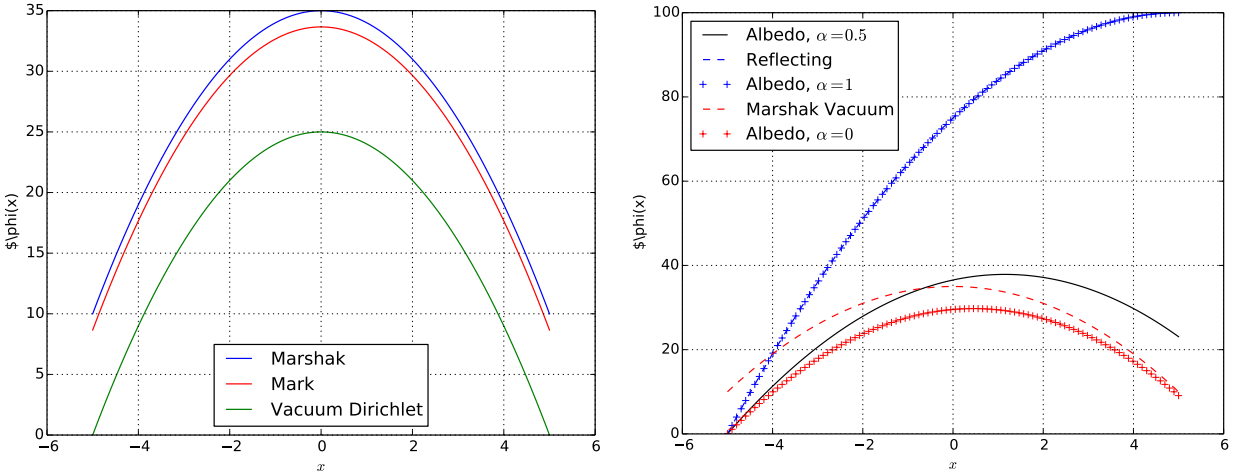


Figure 2: Comparison of Diffusion solutions for various boundary conditions with Q=1, D=0.5, X=10.

Code for Problem 1

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import matplotlib
import math
import matplotlib.font_manager as fm
import matplotlib.ticker as mtick
from scipy import interpolate
from scipy.integrate import quadrature
from copy import deepcopy
from math import exp

#create the fission spectrum
chi = lambda E:  0.4865*np.sinh(np.sqrt(2*E))*np.exp(-E)

#read in cross sections
sigma_t_235 = np.genfromtxt('u235_total.csv', delimiter=",",skip_header=1)
sigma_s_235 = np.genfromtxt('u235_elastic.csv', delimiter=",", skip_header=1)
sigma_t_12  = np.genfromtxt('carbon_total.csv', delimiter=",", skip_header=1)
sigma_s_12  = np.genfromtxt('carbon_elastic.csv', delimiter=",", skip_header=1)
#read in 238-U data
#open total cross-section
sigma_t_238 = np.genfromtxt('u238_total.csv', delimiter=",",skip_header=1)
#open total cross-section
sigma_s_238 = np.genfromtxt('u238_elastic.csv', delimiter=",",skip_header=1)

#Convert energies to MeV and apply a fixup
def fix_energies(cx_2d):
    for i in xrange(len(cx_2d)):
        cx_2d[i,0] *= 1.E-6
        if cx_2d[i,0] == cx_2d[i-1,0]:
           cx_2d[i,0] *= 1.0000001

#apply to all energies
fix_energies(sigma_t_235)
fix_energies(sigma_s_235)
fix_energies(sigma_t_238)
fix_energies(sigma_s_238)
fix_energies(sigma_t_12)
fix_energies(sigma_s_12)

#make interpolation functions
sig_t_235_interp = interpolate.interp1d(sigma_t_235[:,0],
        sigma_t_235[:,1],bounds_error=False, fill_value=sigma_t_235[-1,1])
sig_s_235_interp = interpolate.interp1d(sigma_s_235[:,0],
        sigma_s_235[:,1],bounds_error=False, fill_value=sigma_s_235[-1,1])
sig_t_238_interp = interpolate.interp1d(sigma_t_238[:,0],
        sigma_t_238[:,1],bounds_error=False, fill_value=sigma_t_238[-1,1])
sig_s_238_interp = interpolate.interp1d(sigma_s_238[:,0],
        sigma_s_238[:,1],bounds_error=False, fill_value=sigma_s_238[-1,1])
sig_t_12_interp = interpolate.interp1d(sigma_t_12[:,0],
        sigma_t_12[:,1],bounds_error=False, fill_value=sigma_t_12[-1,1])
sig_s_12_interp = interpolate.interp1d(sigma_s_12[:,0],
        sigma_s_12[:,1],bounds_error=False, fill_value=sigma_s_12[-1,1])

energies = np.union1d(sigma_t_235[:,0], sigma_t_238[:,0])
energies = np.union1d(energies,sigma_t_12[:,0])


#let's make some plots
fig = plt.figure()
plt.loglog(energies, sig_t_238_interp(energies),label=r"$\sigma^{238}_t$")
plt.loglog(energies, sig_s_238_interp(energies),label=r"$\sigma^{238}_s")
plt.loglog(energies, sig_t_12_interp(energies), label="$\sigma^{12}_\mathrm{t}$")
plt.loglog(energies, sig_s_12_interp(energies), label="$\sigma^{12}_\mathrm{s}$")
```

```
plt.legend(loc=3) #bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.ylabel("$\sigma$ (barns)")
plt.xlabel("E (MeV)")
plt.title("Elemental Carbon")
plt.savefig("carb_cx.pdf",bbox_inches='tight')

#atom ratios, keys are atomic mass numbers
gam = dict()
tot = 150 + 0.0072 + 0.9928
gam[235] = 0.0072/tot
gam[238] = 0.9928/tot
gam[12.0107]  = 150./tot
isotopes = gam.keys()

#energy change factors
exc_func = lambda A: (A+1.)**2/(A*A + 1.)

#Put cross sections in a dict
sig_t = dict()
sig_s = dict()
sig_t[12.0107] = sig_t_12_interp
sig_t[235] = sig_t_235_interp
sig_t[238] = sig_t_238_interp
sig_s[12.0107] = sig_s_12_interp
sig_s[235] = sig_s_235_interp
sig_s[238] = sig_s_238_interp

#Initialize phi to 0
phi_prev = interpolate.interp1d(energies,np.zeros(len(energies)),fill_value=0,bounds_error=False)
phi1 = None

converged = 0
tolerance = 1.0e-6
iteration = 0
max_iterations = 1000


#Function for evaluating a new phi
phi_new = lambda E: E*0.0
while not(converged):

    phi_prev = interpolate.interp1d(energies,phi_new(energies),fill_value=0,bounds_error=False)

    #make some lambdas to simplify things
    scat_src_k = lambda E: sum([gam[i]*sig_s[i](exc_func(i)*E)*phi_prev(exc_func(i)*E) for i in isotopes])
    sig_t_k = lambda E: sum([gam[i]*sig_t[i](E) for i in isotopes])
    phi_new = lambda E: (chi(E) + scat_src_k(E))/sig_t_k(E)

    rel_err = np.linalg.norm(phi_prev(energies) - phi_new(energies))/ \
                np.linalg.norm(phi_new(energies))
    converged = rel_err < tolerance or (iteration >= max_iterations)
    iteration += 1

    #if first iteration save it for plotting
    if iteration == 2:
        phi1 = deepcopy(phi_prev)

    print "Completed iteration:", iteration, "Relative change:", rel_err


#plot the first iteration and last iteration, normalized to have an integral of 1
plt.figure()
plt.loglog(energies,phi_new(energies)/np.sum(phi_new(energies)),label=r"$\phi(E)$")
plt.loglog(energies,phi1(energies)/np.sum(phi_new(energies)),label=r"$\phi^{(1)}(E)$")

#plt.loglog(energies,phi(energies)/np.linalg.norm(phi(energies)),label="U metal")
plt.xlabel("E (MeV)")
plt.ylabel("$\phi(E)/\|phi(E)\|_2$ (MeV$^{-1}$)")
```

```
plt.legend(loc='best')
plt.savefig("../uc_spect.pdf",bbox_inches='tight')


#Collapse the cross sections for each material
final_scat_cx = dict()
final_tot_cx = dict()
for iso in isotopes:

    cx_t = []
    cx_s = []
    int_phi_sig_t = 0.0
    int_phi_sig_s = 0.0
    int_phi = 0.0
    bounds = [1.E-06, 0.1, 19.999999999]
    count = 0

    for Ei in xrange(len(energies)-1):

        E = (energies[Ei]+energies[Ei+1])/2.
        dE = energies[Ei+1] - energies[Ei]

        #get cross sections at this energy
        sig_t_tot = sig_t[iso](E)
        sig_s_tot = sig_s[iso](E)
        phi_i = phi_new(E)

        #Use left point quadrature at this energy
        int_phi_sig_t += phi_i*sig_t_tot*dE
        int_phi_sig_s += phi_i*sig_s_tot*dE
        int_phi += phi_i*dE

        #check if hit bound, make CX
        if E > bounds[count] or Ei == len(energies)-2:
            print "Done with group", len(bounds)-count-1
            cx_t.append(int_phi_sig_t/int_phi)
            cx_s.append(int_phi_sig_s/int_phi)
            int_phi_sig_t = 0.0
            int_phi_sig_s = 0.0
            int_phi = 0.0
            count += 1

    print "Final cross sections for: ", iso
    print "Scattering: ", [j for j in reversed(cx_s)]
    print "Total: ", [j for j in reversed(cx_t)]
    final_scat_cx[iso] = [j for j in reversed(cx_s)]
    final_tot_cx[iso]  = [j for j in reversed(cx_t)]

#Total cross sections Collapse
cx_t = []
cx_s = []
int_phi_sig_t = 0.0
int_phi_sig_s = 0.0
int_phi = 0.0
bounds = [1.E-06, 0.1, 19.999999999]
count = 0

#collapse by hand
scat = []
tot  = []
for g in range(len(bounds)):
    scat.append(sum([gam[i]*final_scat_cx[i][g] for i in isotopes]))
    tot.append(sum([gam[i]*final_tot_cx[i][g] for i in isotopes]))

print "Scattering collapsed: ", scat
print "Total collapsed: ", tot


```

```
#See what the thermal cross section would look like with a maxwelliana
k = 8.6173303E-11 #MeV/K
T = 293 #K

maxwell = lambda E: E*exp(-E/(k*T))
int_phi_sig_t = 0.0
int_phi_sig_s = 0.0
int_phi = 0.0
bound   = 1.E-06
for Ei in xrange(len(energies)-1):

    E = (energies[Ei]+energies[Ei+1])/2.
    dE = energies[Ei+1] - energies[Ei]

    #get cross sections at this energy
    sig_t_tot = sum([gam[iso]*sig_t[iso](E) for iso in isotopes])
    sig_s_tot = sum([gam[iso]*sig_s[iso](E) for iso in isotopes])
    phi_i = maxwell(E)

    #Use left point quadrature at this energy
    int_phi_sig_t += phi_i*sig_t_tot*dE
    int_phi_sig_s += phi_i*sig_s_tot*dE
    int_phi += phi_i*dE

    #check if hit bound, make CX
    if E > bound:
        print "Maxwellian collapsed Sigma_t, Sigma_s:",int_phi_sig_t/int_phi,\
                int_phi_sig_s/int_phi
        break

plt.show()
```

```
import numpy as np
import re
import matplotlib
import matplotlib.pyplot as plt
import matplotlib
import math
import matplotlib.font_manager as fm
import matplotlib.ticker as mtick
from scipy import interpolate
from copy import deepcopy
from scipy.integrate import quadrature


def main():

    group_edges, sig_t, sig_el, scat_mat = proc_cx_file('hydrogen.cx')
    group_dE = [group_edges[i] - group_edges[i+1] for i in range(len(group_edges)-1)]

    #convert all cross sections to
    atom_dens = 0.00149

    sig_t = [i*atom_dens for i in sig_t]
    for key in scat_mat.keys():
        scat_mat[key] = [[i*atom_dens for i in x] for x in scat_mat[key]]
    sig_el = [i*atom_dens for i in sig_el]

    #Check
    print "These two numbers should be equal", sum([scat_mat[0][i][4] for i in range(5)]), sig_el[4]

    #create the fission spectrum for U235
    chi = lambda E:  0.4865*np.sinh(np.sqrt(2.*E))*np.exp(-E)

    #Find group averaged chi's
    chi_groups = []
    for i in range(len(group_edges)-1):
        chi_g = quadrature(chi,group_edges[i+1],group_edges[i])[0]
```

```
37          chi_groups.append(chi_g)
38      chi_groups = np.array(chi_groups)
39
40      #Build a matrix
41      A = np.zeros((len(chi_groups),len(chi_groups)))
42
43      #Same matrix for all methods
44      S = scat_mat[0]
45      for i in range(len(A)):
46          A[i,i] = sig_t[i]
47          for j in range(len(A[i])):
48              A[i,j] -= S[i][j]
49
50      #Solve system
51      phi = np.linalg.solve(A,chi_groups)
52
53      #normalize
54      intphi = sum([group_dE[i]*phi[i] for i in range(len(phi))])
55      phi = [i/intphi for i in phi]
56
57      #let's make some plots
58      fig = plt.figure()
59      log_x = np.linspace(math.log10(group_edges[0]-0.000001),math.log10(group_edges[-1]),num=1000)
60      x = [10.**i for i in log_x]
61      y = []
62      for i in x:
63          for g in range(len(phi)):
64              if i <= group_edges[g]:
65                  if i >= group_edges[g+1]:
66                      y.append(phi[g])
67      print len(x), len(y)
68      plt.semilogx(x,y)
69      plt.legend(loc=3) #bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
70      plt.ylabel("$\phi(E)/|\phi(E)|_1$ MeV$^{-1}$")
71      plt.xlabel("E (MeV)")
72      plt.savefig("../method_compare.pdf",bbox_inches='tight')
73
74  def proc_cx_file(fname):
75
76      groups = []
77      sig_t = None
78      sig_el = None
79      scat_mat = dict()
80
81      with open(fname) as f:
82          lines = f.readlines()
83          for line in lines:
84
85              if re.search("Group boun", line):
86                  idx = lines.index(line)
87                  while True:
88                      if re.search("^\s*$",lines[idx]):
89                          break
90                      else:
91                          idx+=1
92                          groups += [float(i)*1.E-06 for i in
93                              lines[idx].split()]
94
95              elif re.search("MT 1\s*$", line):
96                  sig_t = [float(i) for i in lines[lines.index(line)+1].split()]
97              elif re.search("MT 2\s*$",line):
98                  sig_el =[float(i) for i in lines[lines.index(line)+1].split()]
99              elif re.search(", Moment \d+",line):
100                 mom = float(re.search(", Moment (\d+)",line).group(1))
101                 idx = lines.index(line)+1
102                 mat = dict()
103
104                 #Get all the scatering moments til we are done
```

```
105                 while True:
106                     if idx >= len(lines) or re.search(", Moment \d+",lines[idx]) or re.search("^\s+$",lines
107                         scat_mat[mom] = mat
108                         break
109                     else:
110                         (sink,first,last) = lines[idx].split()[3:6]
111                         idx += 1
112                         mat[int(sink)] = [float(i) for i in lines[idx].split()]
113                         idx += 1
114
115     #Convert scat matrix to an actual matrix
116     for mom in scat_mat.keys():
117         dic = scat_mat[mom]
118         rows = max(dic.keys())+1
119         mat = [None for j in range(rows)]
120         for i in range(rows):
121             new_row = [0.0 for j in range(rows)]
122             for j in range(len(dic[i])):
123                 new_row[j] = dic[i][j]
124             mat[i] = new_row
125
126         scat_mat[mom] = np.array(mat)
127
128
129     return groups, sig_t, sig_el, scat_mat
130
131
132 if __name__ == "__main__":
133     main()
```