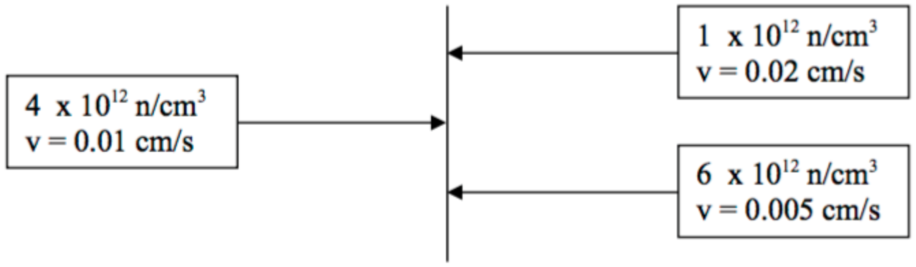
Problem 1.

Consider a thin slab of ^{235}U with the incident thermal neutron beams shown below:



4×10^{12} n/cm^3
v = 0.01 cm/s
1×10^{12} n/cm^3
v = 0.02 cm/s
6×10^{12} n/cm^3
v = 0.005 cm/s

Assuming the beam intensities are constant throughout the entire slab, compute:

(a) the neutron flux,

(b) the current density,

(c) the fission rate density.

Solution

Part (a)

The neutron flux is defined as $\phi(\vec{r},t) = vN(\vec{r},t)$.

$$\begin{aligned}\phi(\vec{r},t) &= vN(\vec{r},t)\\ &= \sum_v vN(\vec{r},t)\\ &= 0.01\,\mathrm{cm\,s^{-1}} \times 4 \times 10^{12}\,\mathrm{n/cm^3} + 0.02\,\mathrm{cm\,s^{-1}} \times 1 \times 10^{12}\,\mathrm{n/cm^3} + 0.005\,\mathrm{cm\,s^{-1}} \times 6 \times 10^{12}\,\mathrm{n/cm^3}\\ &= 9 \times 10^{10}\,\mathrm{neutron/cm^2s}\end{aligned}$$

Part (b)

Since these neutrons are passing through the slab in different directions, we must compute this as a vector quantity:

$$\begin{aligned}\vec{J} &= J_+ - J_-\\ &= 0.01\,\mathrm{cm\,s^{-1}} \times 4 \times 10^{12}\,\mathrm{n/cm^3} - (0.02\,\mathrm{cm\,s^{-1}} \times 1 \times 10^{12}\,\mathrm{n/cm^3} + 0.005\,\mathrm{cm\,s^{-1}} \times 6 \times 10^{12}\,\mathrm{n/cm^3})\\ &= -1 \times 10^{10}\,\mathrm{neutron/cm^2s}\end{aligned}$$

Part (c)

$$\begin{aligned}
F(\vec{r},t) &= v\Sigma_f N(\vec{r},t) \\
&= \Sigma_f \phi(\vec{r},t) \\
&= \frac{N_A\rho}{M_m}\sigma_f\phi(\vec{r},t) \\
&= 28.7346\,\mathrm{cm}^{-1} \times 9 \times 10^{10}\,/\mathrm{cm}^2\mathrm{s} \\
&= 2.586 \times 10^{12}\,\mathrm{fission}/\mathrm{cm}^3\mathrm{s}
\end{aligned}$$

Problem 2. Duderstadt & Hamilton 4-4

In a spherical thermal reactor of radius R, it is found that the angular neutron flux can be roughly described by:

$$\phi\left(\mathbf{r}, E, \hat{\Omega}\right) = \frac{\phi_0}{4\pi} E \exp\left(-\frac{E}{kT}\right) \frac{\sin(\pi r/R)}{r}$$

Compute the total number of neutrons in the reactor.

Solution

We know that $\phi(r,t) = vN(r,t)$, $v = \sqrt{2Em_n}$ for non-relativistic velocities, and that the number of neutrons in $\mathrm{d}^3 r$ is $N(\mathbf{r},t)\mathrm{d}^3 r$. The number of neutrons in the reactor is therefore:

$$\iiint_R \phi/v \mathrm{d}^3 r$$

$$\begin{aligned}
\int N \,\mathrm{d}^3 r &= \iiint_R \frac{1}{v}\frac{\phi_0}{4\pi} E \exp\left(-\frac{E}{kT}\right) \frac{\sin(\pi r/R)}{r} \mathrm{d}^3 r \\
&= \int_0^R \int_0^\pi \int_0^{2\pi} \frac{1}{v}\frac{\phi_0}{4\pi} E \exp\left(-\frac{E}{kT}\right) \frac{\sin(\pi r/R)}{r} r^2 \sin(\phi)\,\mathrm{d}\theta\,\mathrm{d}\phi\,\mathrm{d}r \\
&= \int_0^R \int_0^\pi \int_0^{2\pi} \frac{\phi_0}{8\pi} m_n v \exp\left(-\frac{m_n v^2}{2kT}\right) r \sin(\pi r/R) \sin(\phi)\,\mathrm{d}\theta\,\mathrm{d}\phi\,\mathrm{d}r \\
&= \frac{\phi_0}{4} m_n v \exp\left(-\frac{m_n v^2}{2kT}\right) \int_0^R \int_0^\pi r \sin\left(\frac{\pi r}{R}\right) \sin\phi \,\mathrm{d}\phi\,\mathrm{d}r \\
&= \frac{\phi_0}{\pi} m_n v \exp\left(-\frac{E}{kT}\right) R^2
\end{aligned}$$

Problem 3.

Consider the differential equation analytical and numerical solution presented in class `NE470_2012_02_15` at 35 minutes. Reproduce both of the solutions on your own using Fortran, C/C++, or other computer language you use for the project.

Extra Credit (20% bonus added to total grade): Modify your program to solve this problem for a VARIABLE NUMBER OF NODES (input to the program). Then evaluate the impact of increasing the number of nodes upon the error of your numerical solution. How many nodes are required to match within 4 or 5 significant figures?

Solution

See attached code. The maximum disagreement between the discrete value and the analytic solution at any one node dropped below 10^{-4} when the number of nodes was 370.

Problem 4. Duderstadt & Hamilton 4-12

Use Simpson's rule to write a numerical quadrature formula for the angular integral $\int_{-1}^{+1} \mathrm{d}\mu\phi(x,\mu)$ for N equal mesh intervals.

Solution

Simpson's rule:

$$\int_{x_0}^{x_2} f(x)\mathrm{d}x \approx \Delta x \frac{f(0)+4f(1)+f(2)}{3}, \Delta x = x_2 - x_1 = x_1 - x_0$$

$$\int_{-1}^{1} \phi(x,\mu)\mathrm{d}\mu \approx \frac{\Delta x}{3}\left[\phi_{-1}(x) + 4\phi_0(x) + \phi_1(x)\right]$$

We know that

$$\int_a^b f = \int_a^{a+h} f + \int_{a+h}^{a+2h} f + \cdots + \int_{b-2h}^{b+h} f + \int_{b-h}^{b} f$$

Therefore we can select $h = \frac{b-a}{n}$ and say:

$$\int_{-1}^{1} \phi(x,\mu)\,\mathrm{d}\mu = \frac{h}{3}\sum_{j=1}^{n}\left[\left(\phi_{-1+(j-1)h}(x) + 4\phi_{-1+jh/2}(x) + \phi_{-1+jh}\right)\right]$$

Problem 5.

Compute the thermal neutron diffusion coefficients characterizing light water, heavy water, graphite, and natural uranium. Then compute the extrapolation length z_0 characterizing these materials.

Solution

Assuming plane geometries, *Duderstadt & Hamilton* equation 4-180 gives the extrapolation length z_0 as

$$z_0 = 0.7104\lambda_{tr}$$

$$\lambda_{tr} = \Sigma_{tr}^{-1}$$
$$\Sigma_{tr} = \Sigma_t - \bar{\mu_0}\Sigma_s$$

Material	Σ_t (cm^{-1})	$1 - \bar{\mu_0}$	Σ_s (cm^{-1})
H_2O	3.45	0.676	3.45
D_2O	0.449	0.884	0.449
Graphite	0.385	0.9444	0.385
U	0.765	0.9972	0.397

Calculations are performed in the attached code. Results:

Material	D (cm)	z_0 (cm)
H_2O	0.143	0.305
D_2O	0.840	1.79
Graphite	0.917	1.95
U	0.436	0.930

NE470 Homework 4 Addendum

October 24, 2016

1 NE470 Homework No. 4

1.1 Problem 3

This is an attempt to solve the for the function $f(x)$ by discretizing it into N nodes such that

$$f_n \approx f(n * \Delta x), \Delta x = W/n, n \in \{0, 1, \ldots, N-1\}$$

For this problem, $W = 4$. Our boundary conditions are set as

$$f(0) = 2$$

and

$$f(4) = 54.61647$$

```
In [2]: # numpy contains useful array/matrix tools
        import numpy as np
```

The equation we are trying to solve is:

$$f'' - f = 0$$

Using the following approximation of the second derivative:

$$f'' \approx \frac{f_{n-1} - 2f_n + f_{n+1}}{\Delta x^2}$$

we arrive at an operator that approximately solves for f:

$$\left(\frac{1}{\Delta x^2}\right) f_{i-1} + \left(\frac{-2}{\Delta x^2} - 1\right) f_i + \left(\frac{1}{\Delta x^2}\right) f_{i+1} = b_i$$

Now we solve for

$$\mathbf{A} f = b$$

```
In [3]: def solution(N):
            # define our interval length
            deltaX = 4.0 / (N-1)
            deltaXsquared = deltaX ** 2

            # define our matrix
```

```
        diag = -2.0 / deltaXsquared - 1.0
        offdiag = 1.0 / deltaXsquared

        # instantiate matrix and fill diagonal values
        A = np.eye(N-2) * diag
        A = A + offdiag * np.diagflat(np.resize(np.array([1.0]), (N-3)), 1)
        A = A + offdiag * np.diagflat(np.resize(np.array([1.0]), (N-3)), -1)

        # define our output vector, b
        b = np.zeros(N-2)
        b[0] = -2.0 / deltaXsquared
        b[N-3] = -54.61647 / deltaXsquared

        # compute solution
        f = np.linalg.solve(A, b)

        return f
```

```
In [4]: def analytic(N):
            """ The analytic solution to the equation """
            x = np.linspace(0, 4, num=N)
            x = x[1:-1]
            return np.exp(x) + np.exp(-x)
```

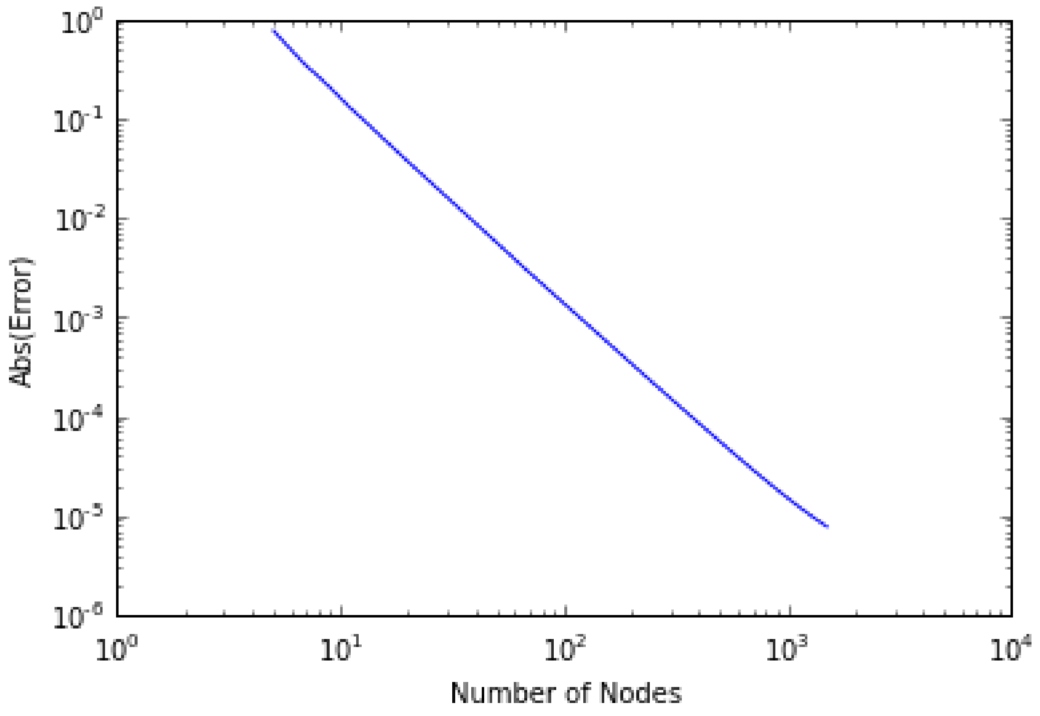
```
In [5]: def error(n):
            return np.amax(np.abs(analytic(n) - solution(n)))
```

```
In [6]: %%timeit
        e_min = 10**-4
        e=1.0

        n_max = 10**5
        n = 5

        while e>e_min and n<n_max:
            n += 1
            e = error(n)
        print "{0}\t\t{1}".format(n, e)
```

```
370                9.97073790181e-05
370                9.97073790181e-05
370                9.97073790181e-05
370                9.97073790181e-05
1 loop, best of 3: 605 ms per loop
```

```
In [9]: %matplotlib inline
        import matplotlib.pyplot as plt
        n = np.arange(5, 1500)
```

```
plt.plot(n, [error(i) for i in n])
plt.yscale('log')
plt.xscale('log')
plt.xlabel('Number of Nodes')
plt.ylabel('Abs(Error)')
plt.show()
```



1.2 Problem 5

```
In [10]: def sigma_tr(sigma_t, mu_0, sigma_s):
             # mu_0 is actually 1-mu_0 in my table
             mu_0 = 1 - mu_0

             return sigma_t - mu_0 * sigma_s
```

```
In [22]: def diffusion_coefficient(sigma_t, mu_0, sigma_s):
             return (1.0 / 3.0) / sigma_tr(sigma_t, mu_0, sigma_s)
```

```
In [11]: def z_0(sigma_t, mu_0, sigma_s):
             return 0.7104 / sigma_tr(sigma_t, mu_0, sigma_s)
```

Light water

```
In [12]: sigma_tr(3.45, 0.676, 3.45)
```

```
Out[12]: 2.3322000000000003

In [23]: diffusion_coefficient(3.45, 0.676, 3.45)

Out[23]: 0.1429265643312466

In [13]: z_0(3.45, 0.676, 3.45)

Out[13]: 0.30460509390275275
```

Heavy water

```
In [14]: sigma_tr(0.449, 0.884, 0.449)

Out[14]: 0.396916

In [25]: diffusion_coefficient(0.449, 0.884, 0.449)

Out[25]: 0.8398082549792231

In [15]: z_0(0.449, 0.884, 0.449)

Out[15]: 1.7897993530117204
```

Graphite

```
In [16]: sigma_tr(0.385, 0.9444, 0.385)

Out[16]: 0.36359400000000003

In [24]: diffusion_coefficient(0.385, 0.9444, 0.385)

Out[24]: 0.9167734707760119

In [17]: z_0(0.385, 0.9444, 0.385)

Out[17]: 1.953827620917837
```

Uranium

```
In [18]: sigma_tr(0.765, 0.9972, 0.397)

Out[18]: 0.7638884

In [26]: diffusion_coefficient(0.765, 0.9972, 0.397)

Out[26]: 0.4363639156365423

In [19]: z_0(0.765, 0.9972, 0.397)

Out[19]: 0.9299787770045991

In [ ]:
```