hw2

October 4, 2016

1 NE255 Homework 2

1.1 Milos Atz

1.2 Due 2016-10-4

1.2.1 1) Determine the macroscopic scattering cross section of UO_2 as a function of a generic enrichment factor $\gamma = N_{U-235}/N_{U-238}$ where N is the atom density. Find its value assuming a density of 10 g/cm^3, $\sigma_s^U = 8.9$ b and $\sigma_s^O = 3.75$ b, and 5% weight enrichment.

$$\Sigma_s^{UO2} = N_{UO2}\sigma_s^{UO2} \qquad = N_{UO2}\big(\sigma_s^U + 2\sigma_s^O\big)$$

We can use γ to find N_{UO2}. Since the density is given, γ is used to find the molar mass of uranium depending on the enrichment. Based on that, the molar mass can be calculated.

$$MM_U = 235\frac{N_{U235}}{N_{U235}+N_{U238}} + 238\frac{N_{U238}}{N_{U235}+N_{U238}}$$

$$= 235\left[\frac{N_{U235}+N_{U238}}{N_{U235}}\right]^{-1} + 238\left[\frac{N_{U235}+N_{U238}}{N_{U238}}\right]^{-1}$$

$$= 235\left[\frac{\gamma+1}{\gamma}\right]^{-1} + 238\left[1+\gamma\right]^{-1}$$

$$= 235\frac{\gamma}{\gamma+1} + 238\frac{1}{\gamma+1}$$

$$= \frac{235\gamma+238}{\gamma+1}$$

Now that we have $MM_U(\gamma)$, we can solve for N_{UO2} in terms of γ.

$$N_{UO2} = N_A\frac{\rho_{UO2}}{MM_{UO2}}$$

…where N_A is Avogadro's number. To get MM_{UO2}, we simply sum the molar masses of uranium and oxygen, weighting by their atomic ratios.

$$MM_{UO2}(\gamma) = MM_U(\gamma) + 2MM_O$$

Thus, the formulation of the macroscopic scattering cross section for UO_2 is given as the following.

$$\Sigma_s^{UO2}(\gamma) = N_A \frac{\rho_{UO2}}{MM_U(\gamma) + 2MM_O}(\sigma_s^U + 2\sigma_s^O)$$

We can calculate the scattering cross section for the given inputs in python.

```
In [2]: # First, need to convert enrichment by mass into gamma, which is an atomic
        def det_gamma(enrichment):
            mol_u235 = enrichment/235
            mol_u238 = (1-enrichment)/238
            return(mol_u235/mol_u238)

        def det_scattering_xs(gamma, sxs_u = 8.9, sxs_o = 3.75, density = 10):
            mm_u = (235*gamma+238)/(gamma+1)
            mm_uo2 = mm_u+(2*16)
            n_uo2 = 6.022141e23*density/(mm_uo2)
            return(n_uo2*(sxs_u+2*sxs_o)*1e-24)

        e = 0.05
        # print(det_gamma(e))
        print(str(det_scattering_xs(det_gamma(e)))+' cm^-1')
```

```
0.365995100148 cm^-1
```

1.2.2 2) A major challenge in scientific computing is navigating the disparate architectures that comprise new supercomputers.

Rank	Computer	Year	Peak Speed (Rmax)	Architecture
1	Sunway TaihuLight	2016	93.01 PFLOPS	Uses a total of 40,960 Chinese-designed SW26010 manycore 64-bit RISC processors (CPUs). Each processor chip contains 256 processing cores, and an additional four auxiliary cores for system management, for a total of 10,649,600 CPU cores across the entire system.
2	NUDT Tianhe-2	2013	33.86 PFLOPS	With 16,000 computer nodes, each comprising two Intel Ivy Bridge Xeon processors and three Xeon Phi coprocessor chips (MICs), it represented the world's largest installation of Ivy Bridge and Xeon Phi chips, counting a total of 3,120,000 cores.

Rank	Computer	Year	Peak Speed (Rmax)	Architecture
3	Cray Titan	2012	17.59 PFLOPS	Titan has 18,688 nodes, each containing a 16-core AMD Opteron 6274 CPU with 32 GB of DDR3 ECC memory and an Nvidia Tesla K20X GPU with 6 GB GDDR5 ECC memory. There are a total of 299,008 processor cores and a total of 693.6 TiB of CPU and GPU RAM.

(a) Look up the Top 10 supercomputing list and briefly describe the archi- tecture of the top three machines. List the number of machines of each type on the top 10 (e.g. X are GPU-accelerated, Y contain MICs, etc.) The architecture specifications for each computer are obtained from the Wikipedia page associated with that computer.

(b) Describe the main characteristics of GPUs, MICs (multi integrated cores), and CPUs–memory, clock speed, structure, etc. CPU: A central processing unit (CPU) is the electronic circuit in most computeres that carries out operations. Most CPUs are microprocessors in that they are contained on a single (or at most, a few) integrated circuit chips. Synchronous microprocessors have clock rates ranging from tens of MHz to several GHz. Associated RAM ranges from very little (2 GB) up to lots (16 GB).

GPU: A graphics processing unit (GPU) is a specialized electronic circuit. It is made of thousands of very small cores and because of this highly parallel structure, it is more efficient than conventional CPUs at performing operations where processing of large blocks of data is done in parallel. GP (general purpose) GPUs are often used to "boost" computing power by doing CPU work; they have fast access to expensive internal memory which allows them to do operations faster than CPUs (which have to talk back and forth with the associated RAM). Clock speeds in the range of hundred MHz to a few GHz are possible; the GPUs in Titan have a base clock speed of about 732 MHz and 6144 MB internal memory.

MICs: A many/multi-integrated core (MIC) is a processor sold by Intel that combines many CPU cores onto a single chip (rather than 2 to 4 in microprocessors, MICs have 50+). Doing this results in increased processing power relative to CPUs because the MICs are designed to handle many simple tasks in parallel, whereas CPU parallel performance is not as good. The Intel Xeon Phi has 64-72 cores, a clock speed around 1.3-1.5 GHz and DDR4-2400 (16 GB) RAM.

(c) Based on what we've talked about so far, postulate challenges of solving the neutron transport equation in a way that would work on all of these architectures Solving the NTE in a way that would work on all of these architectures would require massive parallelization. If one wanted to incorporate the use of all of these architectures, they would have to allot different tasks to different machines. Breaking up those tasks in the best way possible is a challenge in and of itself. One must determine how many tasks they'd like to break the problem into - more tasks means that you can utilize more cores/chips, but also means that you need more information transfer between them, which takes time. Optimizing this over a massive machine (i.e. a supercomputer) is a type of science in and of itself!

1.2.3 3) We often measure convergence by comparing one iteration to the previous iteration (rather than the solution, since we presumably don't know what it is). Imagine that you have software that gives the following solution vectors

$$\mathbf{x_{n-1}} = \begin{pmatrix} 0.45 \\ 0.95 \\ 0.20 \\ -0.05 \\ 0.60 \end{pmatrix}, \mathbf{x_n} = \begin{pmatrix} 0.50 \\ 0.90 \\ 0.30 \\ -0.10 \\ 0.50 \end{pmatrix}$$

1.2.4 Calculate:

(a) The absolute and relative error using the 1 norm

```
In [1]: # First, have to set up these two vectors
        import numpy as np
        x_old = np.array([0.45, 0.95, 0.20, -0.05, 0.60])
        x_new = np.array([0.50, 0.90, 0.30, -0.10, 0.50])
```

The norms can be used to find the absolute and relative error as follows:

$$e_{abs} = \|x_n - x_{n-1}\|$$

$$e_{rel} = \frac{\|x_n - x_{n-1}\|}{\|x_n\|}$$

For this problem, I write out how to compute the norms and errors, calculate them manually, and check the result utilizing the built-in numpy functions.

The 1-norm is the taxicab norm, taken as the sum of the absolute values of the vector. Thus, the absolute error with the 1-norm is found as follows:

$$e_{abs} = \|x_n - x_{n-1}\|_1$$

$$\mathbf{x_n} - \mathbf{x_{n-1}} = \begin{pmatrix} 0.50 - 0.45 \\ 0.90 - 0.95 \\ 0.30 - 0.20 \\ -0.10 - -0.05 \\ 0.50 - 0.60 \end{pmatrix} = \begin{pmatrix} 0.05 \\ -0.05 \\ 0.10 \\ -0.05 \\ -0.1 \end{pmatrix}$$

$$e_{abs} = \|x_n - x_{n-1}\|_1 = \sum_i |x_{n,i} - x_{n-1,i}| = |0.05| + |-0.05| + |0.10| + |-0.05| + |-0.1| = 0.35$$

The relative error is just the absolute value of the absolute error divided by the norm of the current solution vector. This gives the following:

$$e_{rel} = \frac{\|x_n - x_{n-1}\|_1}{\|x_n\|_1}$$

$$e_{rel} = \frac{0.35}{\sum_i |x_{n,i}|} = \frac{0.35}{|0.50| + |0.90| + |0.30| + |-0.10| + |0.50|} = 0.15$$

```
In [31]: # The 1-norm is the taxicab norm, taken as the sum of the absolute values
         # Absolute error
         print('Absolute error = '+str(np.linalg.norm(x_new-x_old, 1)))

         # Relative error
         print('Relative error = '+str(np.linalg.norm(x_new-x_old,1)/np.linalg.norm

Absolute error = 0.35
Relative error = 0.152173913043
```

(b) The absolute and relative error using the 2 norm The same as above can be done here. The 2-norm is the Euclidian norm, calculated as the square root of the sum of the values squared.

$$\mathbf{x_n} - \mathbf{x_{n-1}} = \begin{pmatrix} 0.05 \\ -0.05 \\ 0.10 \\ -0.05 \\ -0.1 \end{pmatrix}$$

$$e_{abs} = \|x_n - x_{n-1}\|_2 = \sqrt{\sum_i (x_{n,i} - x_{n-1,i})^2}$$

$$e_{abs} = \sqrt{(0.05)^2 + (-0.05)^2 + (0.10)^2 + (-0.05)^2 + (-0.1)^2} = 0.166$$

$$e_{rel} = \frac{\|x_n - x_{n-1}\|_2}{\|x_n\|_2}$$

$$e_{rel} = \frac{0.166}{\sqrt{\sum_i (x_{n,i})^2}} = \frac{0.166}{\sqrt{(0.50)^2 + (0.90)^2 + (0.30)^2 + (-0.10)^2 + (0.50)^2}} = 0.140$$

```
In [33]: # The 2-norm is the Euclidian norm, calculated as the square root of the s
         # Absolute error
         print('Absolute error = '+str(np.linalg.norm(x_new-x_old)))

         # Relative error
         print('Relative error = '+str(np.linalg.norm(x_new-x_old)/np.linalg.norm(x

Absolute error = 0.165831239518
Relative error = 0.139655096933
```

(c) The absolute and relative error using the infinity norm The infinity norm is simply the maximum value in the vector. Thus, the error terms can be calculated as follows:

$$\mathbf{x_n} - \mathbf{x_{n-1}} = \begin{pmatrix} 0.05 \\ -0.05 \\ 0.10 \\ -0.05 \\ -0.1 \end{pmatrix}$$

$$e_{abs} = \|x_n - x_{n-1}\|_\infty = \max_i (x_{n,i} - x_{n-1,i}) = 0.10$$

$$e_{rel} = \frac{\|x_n - x_{n-1}\|_\infty}{\|x_n\|_\infty}$$

$$e_{rel} = \frac{0.10}{\max_i x_{n,i}} = \frac{0.10}{0.90} = 0.11$$

```
In [34]: # The infinity norm is simply the maximum value in the vector.
         # Absolute error
         print('Absolute error = '+str(np.linalg.norm(x_new-x_old, np.inf)))

         # Relative error
         print('Relative error = '+str(np.linalg.norm(x_new-x_old, np.inf)/np.linal
```

```
Absolute error = 0.1
Relative error = 0.111111111111
```

What is most restrictive (that is, what would cause the code to converge first)? The one that is the most restrictive (i.e. causes the code to converge first) is the one that yields the lowest error; that norm will most quickly cause the error vector to meet the convergence criterion. Judging by this example, the lowest value occurs for the **infinity norm**, because it has the lowest error for both absolute and relative error.

Imagine that now $x_{n-1} = (0.49, 0.92, 0.4, -0.09, 0.51)^T$. Recalculate the convergence values. What do you observe? What does that mean about how you might select convergence criteria? Here, I do this calculation only using numpy, having demonstrated the calculations by-hand above.

```
In [35]: x_old = np.array([0.49, 0.92, 0.40, -0.09, 0.51])
```

```
In [36]: # 1-norm
         print('1-norm')
         # Absolute error
         print('Absolute error = '+str(np.linalg.norm(x_new-x_old, 1)))
         # Relative error
         print('Relative error = '+str(np.linalg.norm(x_new-x_old,1)/np.linalg.norm

         # 2-norm
         print('\n')
         print('2-norm')
         # Absolute error
         print('Absolute error = '+str(np.linalg.norm(x_new-x_old)))
         # Relative error
         print('Relative error = '+str(np.linalg.norm(x_new-x_old)/np.linalg.norm(x
         # Infinity norm
         print('\n')
         print('infinity-norm')
         # Absolute error
         print('Absolute error = '+str(np.linalg.norm(x_new-x_old, np.inf)))
         # Relative error
         print('Relative error = '+str(np.linalg.norm(x_new-x_old, np.inf)/np.linal
```

```
1-norm
Absolute error = 0.15
Relative error = 0.0652173913043


2-norm
Absolute error = 0.103440804328
Relative error = 0.0871128720814


infinity-norm
Absolute error = 0.1
Relative error = 0.111111111111
```

The absolute and relative errors using the infinity norm do not change. The infinity norm still produces the smallest absolute error. However, the absolute errors calculated using the 1- and 2-norms have significantly decreased and the relative errors from both are now less than those produced using the infinity-norm. This indicates that the infinity norm might be good to get coarse convergence, but for more precise solution, it might be best to use a different norm.

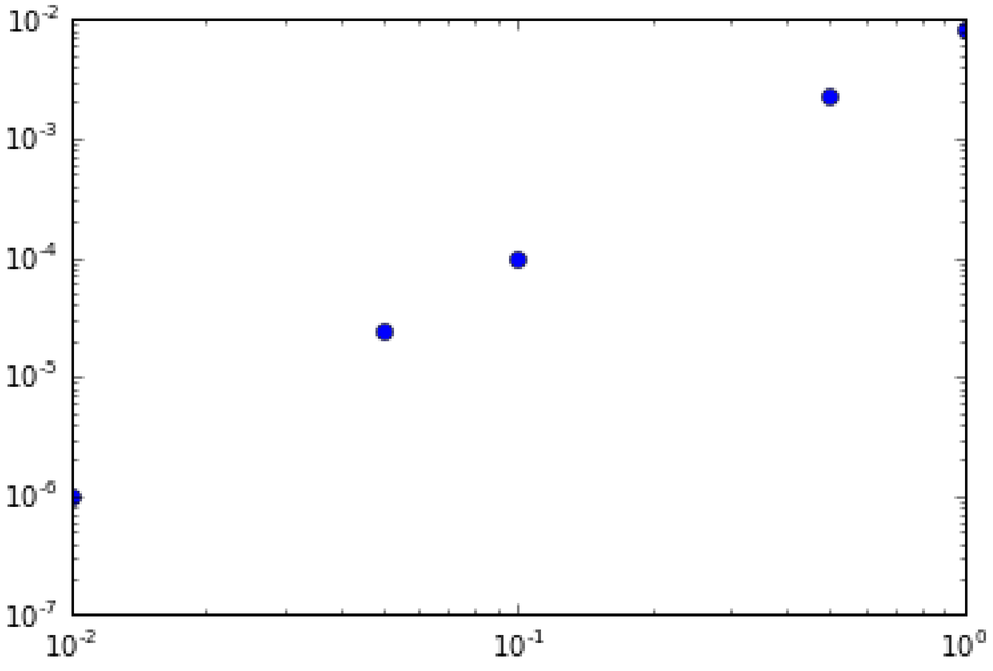
1.2.5 4) You have a piece of software that takes mesh spacing as an input variable. Imagine that you have the following relative error values for each mesh spacing (h) / number of mesh cells (N cells):

h	N cells	rel err
1.00	8	8.44660179e-03
0.50	16	2.30286448e-03
0.10	80	9.84273963e-05
0.05	160	2.48043656e-05
0.01	800	9.98488163e-07

1.2.6 One of the ways we characterize method performance is the order of convergence. We'd like to know how the error changes as we change the resolution of our discretization, in this case, mesh spacing. Using a log-log plot, plot relative error as a function of:

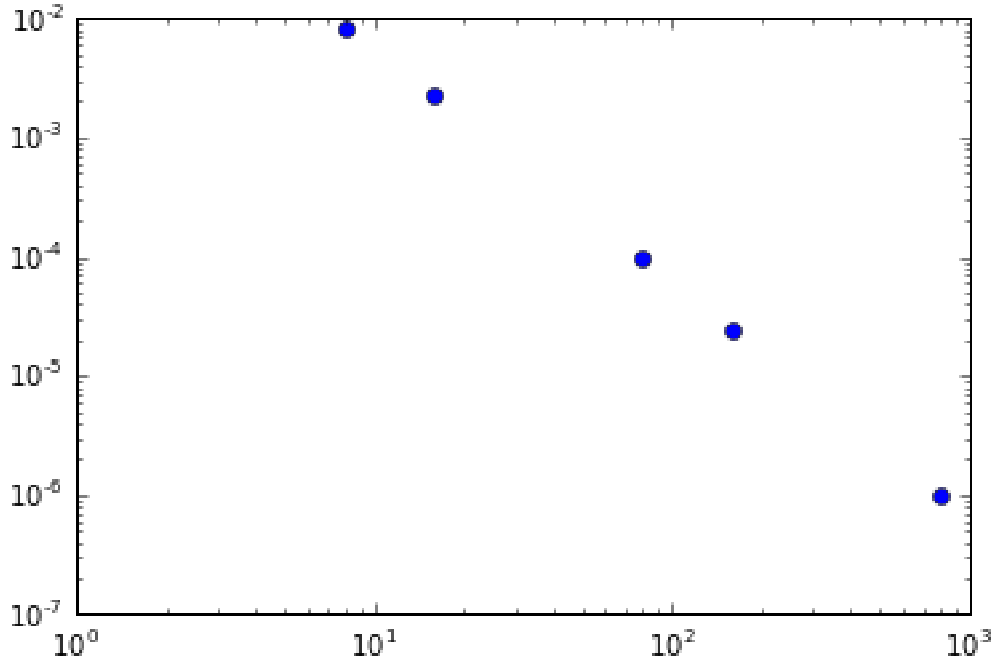
(a) mesh spacing

```
In [10]: %matplotlib inline
         # Mesh spacing
         import matplotlib.pyplot as plt
         relerr = np.array([8.44660179e-03, 2.30286448e-03, 9.84273963e-05, 2.48043
         h = np.array([1, 0.5, 0.1, 0.05, 0.01])
         plt.loglog(h, relerr, 'o')

Out[10]: [<matplotlib.lines.Line2D at 0x115958750>]
```



b) cell count

```
In [12]: # Cell count
         cellcount = np.array([8, 16, 80, 160, 800])
         plt.loglog(cellcount, relerr, 'o')

Out[12]: [<matplotlib.lines.Line2D at 0x115ac8710>]
```



Both of the relationships appear linear on the log-log scale, which indicates that the relationships between relative error and mesh spacing and cell count are monomial, i.e. of the form $e = ax^k$, where e is the error, a is the intercept, and k is the slope.

As mesh spacing increases, the relative error increases. The mesh spacing is related to the cell count because a tighter mesh means more cells. As the number of cells increases, the relative error decreases.

1.2.7 5) What are six underlying assumptions in the neutron transport equation? Write at least one sentence explaining what each assumption means and why we need or want to make it.

- Particles are point objects. This implies that their de Broglie wavelength $\lambda = \frac{h}{mv}$ is small compared to atomic diameter. Their state is fully described by location, velocity vector, and a given time. Rotation and quantum effects are ignored. This set of assumptions is significant because it lays the basis for the flux dependencies of the transport equation and eliminates less significant aspects of the physics that would complicate the equation.

- Neutral particles travel in straight lines between collisions. This is a reflection of the underlying physics (neutrality means no attraction) but also implies no gravitational effects, which avoids complication of the equation.

- Particle-particle interactions are negligible. By not considering self-interaction, the tranport equation becomes linear and much easier to solve.

- Material properties are isotropic. This is important because while material properties should depend on the region of interest, to have them depend on direction would make the equation much more difficult to solve. In general, this assumption is true unless particle velocities are really low.

- Material composition is time-independent. This is generally valid over short time scales and eliminates a dependency, making solution easier. To take into account changes in material properties (i.e. consumption of fissile material), one can alternate flux calculations with solution of Bateman equations.

- Quantities (ψ, χ, ν, etc) are expected values. In most cases, this assumption is valid, but for material properties in regions with very low densities, fluctuations about the mean are not accounted for and can produce high variance.

1.2.8 6) Consider the transport equation:

$$\underbrace{\frac{1}{v}\frac{\partial\psi}{\partial t}(\vec{r},E,\hat{\Omega},t)}_{\text{A}}+\underbrace{\hat{\Omega}\cdot\nabla\psi(\vec{r},E,\hat{\Omega},t)}_{\text{B}}+\underbrace{\Sigma_t(\vec{r},E)\psi(\vec{r},E,\hat{\Omega},t)}_{\text{C}}=\underbrace{S(\vec{r},E,\hat{\Omega},t)}_{\text{D}}+\underbrace{\int_0^\infty dE'\int_{4\pi}d\hat{\Omega}'\Sigma_s(\vec{r},E'\rightarrow E,\hat{\Omega}'\cdot\hat{\Omega})\psi(}_{\text{E}}$$

(a) Briefly describe what each term in the Transport Equation physically represents. A: Time rate of change of neutrons at position $\vec{r}$ with energy E, direction $\hat{\Omega}$ at time t

B: Streaming loss rate - how many neutrons are entering vs. exiting the differential volume at position $\vec{r}$ with energy E, direction $\hat{\Omega}$ at time t

C: Total interaction loss rate - how many neutrons at position $\vec{r}$ with energy E, direction $\hat{\Omega}$ at time t are reacting or disappearing due to interactions.

D: External source rate - some addition of neutrons with position $\vec{r}$ with energy E, direction $\hat{\Omega}$ at time t.

E: In-scattering source rate - how many neutrons with position $\vec{r}$ with energy E', direction $\hat{\Omega}'$ at time t are scattering into the direction and energy ranges of interest, $\hat{\Omega}$ and E.

F: Fission source rate - how many neutrons with position $\vec{r}$ with energy E, direction $\hat{\Omega}$ at time t are formed from fission caused by neutrons with direction and energy $\hat{\Omega}'$ and E'.

(b) Rewrite the time independent form of the equation to include azimuthal symmetry. Show the steps needed to get there. The time independent form of the transport equation can be written as follows:

$$\hat{\Omega}\cdot\nabla\psi(\vec{r},E,\hat{\Omega})+\Sigma_t(\vec{r},E)\psi(\vec{r},E,\hat{\Omega})=S(\vec{r},E,\hat{\Omega})\qquad+\int_0^\infty dE'\int_{4\pi}d\hat{\Omega}'\Sigma_s(\vec{r},E'\rightarrow E,\hat{\Omega}'\cdot\hat{\Omega})\psi(\vec{r},E',\hat{\Omega}',t)$$

At this point, the time dependences have been removed and the first term (partial with respect to time) has been deleted. To apply azimuthal symmetry to the TE, we must deal with the integration over $d\hat{\Omega}$:

$$d\hat{\Omega}=sin(\theta)d\theta d\phi$$

Azimuthal symmetry allows us to evaluate the ϕ portion of $d\hat{\Omega}$, and so each integral over angle becomes an integral over μ from -1 to 1.

$$\int_{4\pi}d\hat{\Omega}=\int_0^{2\pi}d\phi\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}sin(\theta)d\theta=\int_0^{2\pi}d\phi\int_{-1}^{1}d\mu=4\pi$$

Thus, the integration of ψ over angle becomes:

$$\int_{4\pi} d\hat{\Omega}\psi(\vec{r}, E, \hat{\Omega}) = \int_0^{2\pi} d\phi \int_{-1}^{1} d\mu\psi(\vec{r}, E, \mu) = 2\pi \int_{-1}^{1} d\mu\psi(\vec{r}, E, \mu)$$

Applying this to the transport equation, we treat the terms in different ways. Most are straightforward, but the scattering term is rewritten with macroscopic scattering cross section $\Sigma_s(\vec{r}, \mu)$; when written this way, azimuthal symmetry is implied because $\hat{\Omega}' \cdot \hat{\Omega} = \mu$.

Thus, we arrive at the following:

$$\mu \cdot \nabla\psi(\vec{r}, E, \mu) + \Sigma_t(\vec{r}, E)\psi(\vec{r}, E, \mu) = S(\vec{r}, E, \mu) \qquad + 2\pi \int_0^{\infty} dE' \int_{-1}^{1} d\mu' \Sigma_s(\vec{r}, E' \to E, \mu)\psi(\vec{r}, E', \mu', t) \qquad + \frac{\chi(}{}$$

The 2π from the fission term canceled with the 4π in the denominator below the fission energy spectrum.

1.2.9 7) Solve the following differential equation by hand:

$$\frac{d^2y}{dx^2} + 3y(x) = sin(x) \qquad x \in [0, 1]$$

where $y(0) = 1$, $y(1) = 3$. The above equation has a solution of the form:

$$y = y_h + y_p$$

where y_h is the solution to the corresponding homogeneous equation and y_p is the solution that satisfies the nonhomegenous equation. The corresponding homogeneous equation has 0 on the RHS instead of $sin(x)$.

$$\frac{d^2y}{dx^2} + 3y(x) = 0 \qquad x \in [0, 1]$$

The solution to the homogeneous equation is always of the form:

$$y_h = C_1 e^{r_1 x} + C_2 e^{r_2 x}$$

To find this solution, we solve the characteristic equation, which in this case is $r^2 - 3 = 0$. This has the roots $\pm\sqrt{3}$. Therefore,

$$y_h = C_1 e^{\sqrt{3}x} + C_2 e^{-\sqrt{3}x}$$

To find the particular solution, we can perform the "method of undetermined coefficients". We know the relationship of sines and cosines under integration and differentiation. Therefore, we should choose a function of the form $y_p = Acos(x) + Bsin(x)$. Then, we know that:

$$y_p'' = -Acos(x) - Bsin(x)$$

Plugging into our equation, we get:

$$-Acos(x) - Bsin(x) + 3Acos(x) + 3Bsin(x) = sin(x)$$

We can see right away that $A = 0$. To find B is simple algebra.

$$-Bsin(x) + 3Bsin(x) = sin(x)$$

$$2B = 1$$

So $B = \frac{1}{2}$. The solution to the differential equation is:

$$y = C_1 e^{\sqrt{3}x} + C_2 e^{-\sqrt{3}x} + \frac{1}{2} sin(x)$$

To find C_1 and C_2, we plug in the known values, namely that $y(0) = 1$ and $y(1) = 3$.

$$y(x = 0) = 1 = C_1 + C_2$$

$$y(x = 1) = 3 = C_1 e^{\sqrt{3}} + C_2 e^{-\sqrt{3}} + \frac{1}{2} sin(1)$$

$$3 = C_1 e^{\sqrt{3}} + (1 - C_1) e^{-\sqrt{3}} + \frac{1}{2} sin(1)$$

$$C_1 \left[e^{\sqrt{3}} - e^{-\sqrt{3}} \right] = 3 - \frac{1}{2} sin(1) - e^{-\sqrt{3}}$$

$$C_1 = \frac{3 - \frac{1}{2} sin(1) - e^{-\sqrt{3}}}{e^{\sqrt{3}} - e^{-\sqrt{3}}}$$

$$C_2 = 1 - C_1 = 1 - \frac{3 - \frac{1}{2} sin(1) - e^{-\sqrt{3}}}{e^{\sqrt{3}} - e^{-\sqrt{3}}}$$

With our coefficients determined, the final solution is given as:

$$y(x) = \left[\frac{3 - \frac{1}{2} sin(1) - e^{-\sqrt{3}}}{e^{\sqrt{3}} - e^{-\sqrt{3}}} \right] e^{\sqrt{3}x} + \left[1 - \frac{3 - \frac{1}{2} sin(1) - e^{-\sqrt{3}}}{e^{\sqrt{3}} - e^{-\sqrt{3}}} \right] e^{-\sqrt{3}x} + \frac{1}{2} sin(x)$$

What simplifications to the transport equation would have been required to get an equation form that looks like this? To get the transport equation into a form like this one, a number of simplifications and approximations must be made. Note that there is only dependence on one variable. This means reducing the TE phase space by assuming one dimension (x), monoenergetic, and time-independent. Further, the angular dependence should be reduced by making the P_1 approximation on the TE with an isotropic source; this implies linearly anisotropic flux. Ultimately, this results in the diffusion equation.

$$\frac{-1}{3(\Sigma_t - \Sigma_{s,1})} \frac{d^2 \phi_0}{dx^2} + \Sigma_a \phi_0(x) = S_0(x)$$

1.2.10 8) At what energy is the lowest isolated resonance of U-235, U-238, Pu-239, Pu-240, Pu-241, and Pu-242? Why do we care about that?

Considering the (n,tot) microscopic cross section; read off graphs from ENDF (ENDF/B-VII.1):

- U-235: ~0.28 eV; ~255 b
- U-238: ~6.8 ev; ~7300 b
- Pu-239: ~0.30 eV; ~5500 b
- Pu-240: ~1.06 eV; ~126460 b
- Pu-241: ~0.256 eV ; ~2435 b
- Pu-242: ~2.68 eV; 34700 b

We should care about these isolated resonances in thermal reactors as neutrons slow from fast energies to thermal ones. In particular, these resonance represent huge changes in cross section over very narrow energy ranges for very important species with respect to neutronics. In addition, these low-energy resonances are in an energy region that encompasses a potentially high number of neutrons. Correctly incorporating these resonances in thermal reactor calculations should be very important to getting a result that appropriately models the reactor physics.

```
In [ ]:
```