NE155 Final Report

Steven Arroyo
Department of Nuclear Engineering
University of California Berkeley

Introduction

In this project, I am building a two-dimensional (2D) neutron diffusion solver to simulate how neutrons move and interact in a slab reactor model. This type of reactor model represents a horizontal cross-section of a nuclear reactor core and can help us understand how neutrons behave in different materials like the fuel, moderator, and reflector regions. The solver uses our standard NE155 diffusion equation, which is a simplified but powerful way to predict neutron behavior when many collisions occur. It will calculate the neutron flux across a rectangular grid, taking into account material properties and different boundary conditions; specifically, vacuum boundaries on the bottom and left sides, and reflecting boundaries on the top and right. Being able to solve neutron diffusion problems like this is important because it helps nuclear engineers analyze reactor performance, improve safety, and design better systems without needing to perform a full-scale experiment or more.

Mathematics

Once again, the goal of this project is to solve the steady-state neutron diffusion equation in two dimensions. This equation models how neutrons spread out (or simply "diffuse") through the nuclear reactor in question and how they interact with different materials like fuel and moderator. The diffusion equation is a simplified form of the full neutron transport equation and is used when we assume that neutrons undergo many collisions and move in random directions, which makes their behavior smoother and easier to approximate.

Our equation for steady-state neutron diffusion in 2D is:

$$-\nabla \cdot (D(\vec{r})\nabla\phi(r)) + \Sigma_a(\vec{r})\phi(r) = S(\vec{r})$$

In this equation:

$\phi(\vec{r}) = is\ the\ neutron\ scalar\ flux\ at\ position\ \vec{r} = (x, y)$

$D(\vec{r}) = is\ the\ diffusion\ coefficient,\ which\ depends\ on\ the\ material$

$\Sigma_a(\vec{r}) = is\ the\ macroscopic\ absorption\ cross\ section$

$S(\vec{r}) = is\ the\ source\ term$

$\nabla \cdot\ and\ \nabla = divergence\ and\ gradient\ operators,\ respectively$

To solve this equation, I will be using the finite difference method. I can discretize the domain into a grid of square cells and approximate the Laplacian operator $\nabla^2\phi$ using central difference formulas. For a uniform mesh with spacing I'll denote as h, the second derivative in each direction is approximated as:

$$\frac{\partial^2\phi}{\partial x^2} = \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{h^2},\ and, \frac{\partial^2\phi}{\partial y^2} = \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{h^2}$$

Combining each of these dimensions, the full discretized diffusion equation becomes:

$$- D(\frac{\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i+1,j} + \phi_{i-1,j} - 4\phi_{i,j}}{h^2}) + \Sigma_a \phi_{i,j} = S_{i,j}$$

This equation can be applied at each point in the grid I make, leading to a good system of linear equations. I can represent the solution as a vector Φ and the system becomes:

$$A\Phi = b$$

In this equation:

$A = a\ sparse\ matrix\ including\ boundary\ conditions$

$\Phi = vector\ containing\ the\ unknown\ flux\ values\ at\ each\ grid\ point$

$b = source\ vector$

And for my boundary conditions:

Vacuum Boundary Conditions: $\phi = 0$ on the left and bottom edges

Reflecting Boundary Conditions: $\frac{\partial\phi}{\partial n} = 0$ on the right and top edges

The vacuum condition models a surface where neutrons freely escape, while the reflecting condition models a symmetry or material barrier that reflects neutrons back into the domain. Once the matrix equation is constructed, I can use a solver to compute the solution for Φ, giving me the neutron flux across the 2D slab reactor model.

Algorithms

Once again, the goal of this project is to solve the steady-state neutron diffusion equation in two dimensions. This equation models how neutrons spread out (or simply "diffuse") through the nuclear reactor in question and how they interact with different materials like fuel and moderator. The diffusion equation is a simplified form of the full neutron transport equation and is used when we assume that neutrons undergo many collisions and move in random directions, which makes their behavior smoother and easier to approximate.

To solve the 2D neutron diffusion equation numerically, I'll use the **finite difference method** to discretize the domain and construct a system of linear equations. This process involves several algorithmic steps, each of which is implemented in Python to build a complete solver. The main goal is to approximate the behavior of neutrons in a rectangular slab reactor model, accounting for material properties and boundary conditions.

1. **Grid Construction**
   The 2D reactor domain can be divided into a uniform grid of square cells. Each grid cell represents a small region of the reactor and holds properties such as diffusion coefficient D, absorption cross section Σ_a, and source term S. Each cell is identified by its index (i, j), which I'll later map into a 1D index to build the matrix system.
2. **Material Mapping**
   We define different regions of the slab reactor, such as fuel, moderator, and reflector, by assigning specific values of D, Σ_a, and S to different parts of the grid. These values will be stored in 2D arrays for easy lookup during matrix assembly.
3. **Matrix Assembly**
   For each interior grid point, I will apply the discretized diffusion equation using the central difference formula. This produces a five-point stencil that connects each point to its immediate neighbors (left, right, top, bottom). These relationships are translated into a sparse matrix AA using standard numerical indexing schemes, where each row corresponds to one equation in the system.
4. **Boundary Conditions**
   Briefly introduced in the mathematics portion. These will be established as our conditions that must be adhered to in the simulation.
5. **Linear Solver**
   After the matrix A and source vector b are constructed, I can solve the linear system $A\Phi=b$ using a sparse matrix solver like scipy.sparse.linalg.spsolve. This gives us the neutron flux values at each grid point.
6. **Postprocessing**
   The solution vector Φ is reshaped back into a 2D array that matches the spatial grid. I'll visualize the result using matplotlib.pyplot.imshow() or contourf() to show how the neutron flux varies across the reactor. This helps identify high-flux (hot) and low-flux (cold) regions.

Plan for Completion (From Interim Report)

I've started this project, but completion is still a little away. I plan on following a step-by-step coding and testing process to ensure each part of the solver functions correctly before moving on to the next. First, I will set up the basic 2D grid and indexing system using a Python script in Jupyterlab, which forms the foundation for the solver. Then, I will define the material map to assign different diffusion coefficients D, absorption cross sections, and source terms S across the grid. This will allow me to represent a realistic 2D slab reactor, with distinct regions such as fuel in the center and reflectors or moderators on the edges. I will also build functions to assemble the sparse coefficient matrix and right-hand-side source vector based on the finite difference discretization.

Once the matrix system is fully constructed, I will apply the correct boundary conditions: vacuum on the left and bottom edges, and reflecting on the right and top edges. Afterward, I will solve the matrix equation using the sparse linear solver code and visualize the flux distribution using plots in the Jupyterlab notebook. Finally, I will validate the behavior of the solver under different mesh sizes and material configurations. The report will be updated throughout the project to include final results, visualizations, and any improvements or challenges encountered during development. I plan to complete all coding and testing within one week, leaving additional time for refining the report and ensuring clean documentation.

Code Use

The Python code for this project builds a complete 2D neutron diffusion solver using the finite difference method. I'll reiterate, this code calculates how neutrons move through a rectangular slab reactor with different materials and boundary conditions. The code is organized into steps that create the grid, assign material properties, build the diffusion matrix, apply boundary conditions, solve the system of equations, and finally visualize the results (six different steps, with an added seventh for testing other sources or metrics).

To use the code, you first define the size of the reactor domain by setting Lx and Ly, which are the lengths in centimeters (cm) in the x and y directions. Then, you choose how many grid points to use in each direction by setting Nx and Ny. The more points you use, the more accurate the solution will be, but it may take longer to compute (for me it took less than a minute). You also define the material properties for each part of the grid. For example, you set the diffusion coefficient (D), the absorption cross section (Sigma_a), and the external neutron source (S) for every point in the mesh. These are stored in 2D arrays called D_map, Sigma_a_map, and S_map.

Once the setup is complete, the code assembles the matrix A and the vector b, which represent the system of equations that needs to be solved. It uses vacuum boundary conditions (Dirichlet) on the bottom and left edges, where the neutron flux is set to zero. It also uses reflecting boundary conditions (Neumann) on the top and right edges, which simulate symmetry or reflective walls (these were the conditions mentioned in the problem statement). The spsolve() function from SciPy is then used to solve the matrix equation A * phi = b, where phi is the neutron flux at every grid point. The final result is reshaped back into a 2D array and displayed using color plots.

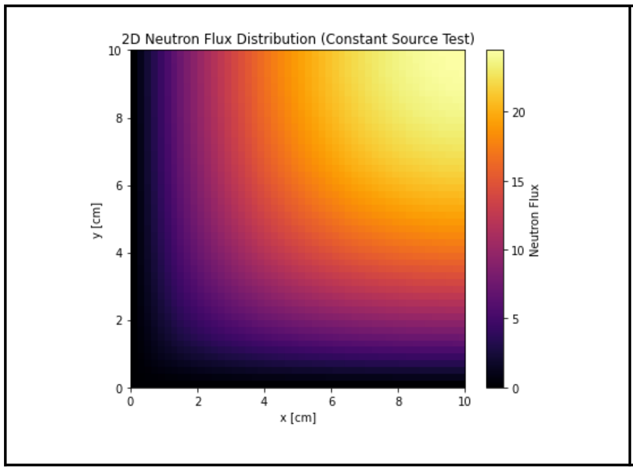
My final output of the code includes a heatmap of the neutron flux and a cross-section plot that shows how the flux changes across the reactor. I made these to help visualize how neutrons act in the system and to confirm whether my model was working correctly, which I was able to prove it does. The code can be easily changed to model different materials or source conditions for 2d slab reactors.

Test Problems & Results

So, to make sure my solver was working correctly, I ran a test using a reactor made of only one material with a constant neutron source across the entire grid. This is one of the easiest test cases because it has a clear expected outcome. In this kind of setup, the neutron flux should be highest in the center of the reactor and decrease smoothly toward the boundaries. The vacuum boundaries on the bottom and left edges allow neutrons to escape, so the flux near those sides should drop to zero. The reflecting boundaries on the top and right edges act like mirrors, so the flux should stay strong near those edges. This is what the problem statement asked for, and my tests confirmed it to be present.

My code ran this test by assigning the same diffusion coefficient, absorption cross section, and source value to every cell in the grid. After solving the system, I created two plots to show the results. The first was a heatmap that showed the full 2D neutron flux across the reactor. This plot looked exactly as I hoped, the flux was highest in the center of the domain and fell off near the vacuum edges. The reflecting boundaries correctly allowed the flux to bounce back, keeping the values high near the top and right sides.

The second plot I made showed a cross-section of the flux along the middle of the reactor, from left to right. My goal here was to show how the flux changed in one direction. It confirmed that the flux steadily increased from the vacuum boundary on the left, peaked near the center, and stayed strong toward the reflecting boundary on the right. This smooth shape matched the physical behavior we would expect from diffusion in a uniform medium with a constant source. With both of these tests together, I think I have strong evidence that the solver was accurate and the boundary conditions were implemented correctly. The results were realistic, stable, and matched the expected physical patterns of neutron behavior in a slab reactor.



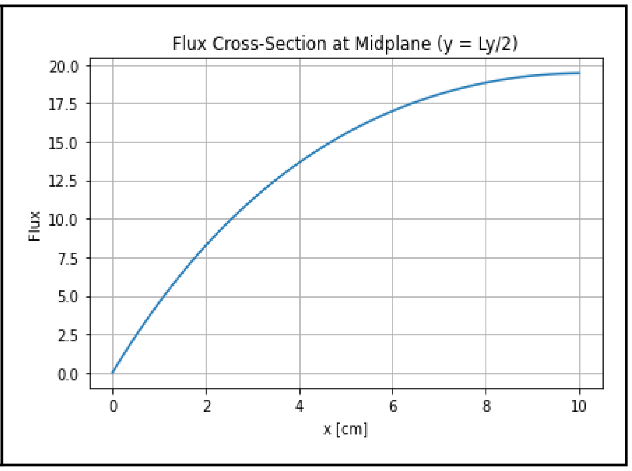


Figure 1 shows the two-dimensional neutron flux distribution across the slab reactor, with higher flux concentrated in the central region and lower values near vacuum boundaries. Figure 2 provides a flux cross-section along the horizontal midplane, clearly illustrating the spatial variation and symmetry of the neutron field.

Conclusion

In conclusion, my project successfully created a 2D neutron diffusion solver that models how neutrons move through a rectangular slab reactor using the finite difference method and applying vacuum and reflecting boundary conditions. The solver was definitely able to simulate realistic neutron behavior across the domain I established. The python code was tested using a simple and uniform reactor setup with a constant source, and the results showed an accurate flux pattern that matched physical expectations. The solver can be adjusted easily for different materials or geometries, making it a flexible tool for future use. Overall, this project helped me understand how numerical methods and reactor physics work together to solve criticality questions

References

1. D. Siefman, *Lecture 14: 2D Diffusion Equation*, NE 155 Course Notes, University of California, Berkeley, Spring 2025.
2. D. Siefman, *Lecture 6: Diffusion Equation*, NE 155 Course Notes, University of California, Berkeley, Spring 2025.
3. https://matplotlib.org/stable/api/matplotlib_configuration_api.html (Matplotlib Guide)
4. W. F. Ames, *Numerical Methods for Partial Differential Equations*, 3rd Edition, Academic Press, 1992.