Capstone Project Report

Introduction

The aim of this project was to effectively model a fission nuclear reactor using python. The general structure is as follows. Beginning with a set number of neutrons, a Monte Carlo simulation would be run to determine the number of fission, scattering, absorption, and escaped neutron events. Determining which events happen utilizes the cross-sections of the different materials in the reactor. A cross-section is a measure of area and can be used to determine the mean free path or the probability of an interaction happening, dependent on the particular elements involved and the energy of a given neutron. Using the cross-sections, cumulative density functions can be found which are used in the simulation.

With the number of different events, several interesting physical properties of the reactor can be calculated. Primarily, whether the reactor produces as many neutrons as it consumes; i.e. is the reactor critical. Other results include (but are not limited to) the number of antineutrinos the reactor produces, how much power the reactor produces, or how the criticality changes as the concentrations of materials are changed in the reactor.

Implementation

Our Process

Reaching our final desing took several iterations, as multiple hurdles had to be overcome before we reached a workable model. This process saw us first simulating a single neutron, turning this model into a crude Monte Carlo. As we worked towards a more polished model, we split our code into several interacting modules. These modules would first spawn a set number of neutrons into the simulation, process their interactions based on their cross-sections (probabilities), and produce counts of each event which occurred. These three modules, in order, may be referred to as the 'neutrons' modeule (initialize_neutrons), the 'density' module (density_functions), and the 'escape' module (check_escape). Finally a fourth 'master' file would be used to bring each of these pieces together by initializing the reactor size, number of neutrons, the particular elements involved, and then running each of the other modules. This final file was labeled the 'reactor' module (reactor_function) in the final iteration.

The second significant update put in much of the work to create cohesion between the modules. It updated the escape module so that event counters could be reset within the module itself, and created a consistently functional reactor module that, with the number of neutrons and mixture inputs alone, could output event counters by running through the entire simulation. A visualization module (reactor_visualizations) module was also added, allowing for several graphical displays of input-output interactions. This ultimately helped us verify how realistic our reactor was, allowing us to visualize several known aspects of nuclear reactors as well as indicating places where there was room to improve our simulation. Many of the graphics from this visualization module will be shown in the results section below. Finally, upon noticing in our visualizations that our base reactor wasn't nearing criticality, we added a variation of our escape module that allowed for partial reflection (rather than eescape), providing more realism as well as demonstrating criticality.

Code Walkthrough

Neutrons Module

The neutron module creates a ‘neutrons’ function which takes as inputs the number of neutrons (count), the reactor radius, and the energy of each neutron (array-like). Firstly, a (3, 3, count) array is initialized with random values between 0 and 1. Each neutron was represented by a slice along axis 2 and needed to represent an (x, y, z) position, direction, and an energy. The first row’s random (0, 1) was used to give each neutron a randomized spherical coordinate which was then converted to cartesian. The second row indicated the direction (x, y, z) the particle would travel, and was converted to unit length. The neutron energies (an input to the original function) would be placed in the position (2,0), with the final 2 positions in the array unused.

Mixtures Module

By unpacking and processing a csv file, we receive the cross-sectional information on uranium-238, uranium-235, boron-10, and heavy water. When initiating a ‘mixtures’ function, we inputted percentage values of the total reactor volume to be taken up by each material. These weighted densities allow us to show the probability of each event (fission, absorption, scattering) for both high- and low-energy neutrons. These new weighted cross-sections can then be used as inputs to our probability density function.

Density Module

The density function was somewhat simpler. Firstly, the probability density function takes the number of particles and total cross section as inputs, outputting a function of ‘x’. The cumulative distribution function integrates over the pdf. By taking the inverse of the cdf and mapping onto it a random value (0, 1), we find the distance each particle will travel before experienceing an interaction.

Escape Module (No Reflection)

The escape module creates a 1d array which, using the inverse cdf function, determines the distance each neutron will travel before an interaction. By moving the particle’s position in the specified direction, it will receive a final position. If this position is outside the reactor, the neutron will have successfully escaped; else it will perform a different event based on the probabilities taken from the mixtures module. In the case that the neutron scatters, its energy may be decreased and it will be run through the simulation recursively.

Reflective Shell

A variation of the escape module allowed neutrons to reflect off a wall rather than simply escaping. This was done probabilistically depending on a certain reflectivity which could be changed. By making this change, our reactor was much more real-to-life and could reach criticality.

Figure 1: caption

References

[1] “Delayed neutrons.” [Online]. Available: https://ec.europa.eu/programmes/erasmus-plus/project-result-content/fbcfee8c-7b46-4e39-a5eb-129ea94d51c7/CTU1_Delayed_neutrons_part_2_Experiment_procedure_for_students.pdf

[2] J. A. E. Agency. [Online]. Available: https://wwwndc.jaea.go.jp/cgi-bin/Tab80WWW.cgi?/data/JENDL/JENDL-4-prc/intern/B010.intern

[3] OpenMC. [Online]. Available: https://docs.openmc.org/en/latest/methods/neutron_physics.html

[4] O. Andersen, “Uranium 238.” [Online]. Available: https://www.sciencedirect.com/topics/pharmacology-toxicology-and-pharmaceutical-science/uranium-238#:~:text=Uranium%20(U%2C%20atomic%20number%2092%2C%20in%20group%205%20of,is%20a%20naturally%20occurring%20actinide.

[5] D. Bogart, “Boron cross sections as - a source of discrepancy.” [Online]. Available: https://ntrs.nasa.gov/api/citations/19660018445/downloads/19660018445.pdf

[6] foxneSsfoxneSs, “Local (?) variable referenced before assignment.” [Online]. Available: https://stackoverflow.com/questions/11904981/local-variable-referenced-before-assignment

[7] H. Kim, “Neutron cross section evaluation of u-238.” [Online]. Available: https://t2.lanl.gov/fiesta2017/Talks/Kim.pdf

[8] W. McDowell. [Online]. Available: https://slideplayer.com/slide/8428135/

[9] username4567, “Calculate inverse of a function–library.” [Online]. Available: https://stackoverflow.com/questions/15200560/calculate-inverse-of-a-function-library

[10] Sep 2024. [Online]. Available: https://en.wikipedia.org/wiki/Neutron_transport