1 What is Our Project?

1. Simulating a nuclear reactor using monte-carlo

   Repeating random results to find a statistical output

   → No time dependence: actions happen in 'ticks'

1.1 Assumptions

1. Reactive elements are in a homogenous mixture
2. Spherical reactor
3. Fissions do not release new neutrons

   For simplicity: This event would cause a never-ending cycle of reactions because the fuel is assumed to be homogenous and unchanging in quantity

1.2 Inputs

1. Reactor Size
2. Neutrons

   Position, Velocity, Energy (?)

3. Cross-sections

1.2.1 Cross-Sections

A varying 'size' dependent on neutron energy and reactive element

→ Determines probability of an event occurring

Cross-sections are different for each event

1.2.2 Possible Events (And How They Are Processed)

Fission: The neutron collides with an atom with enough energy to create fission

→ The fission event is counted and the neutron is removed from the simulation

Escape: The netron escapes the reactor

→ The escape event is counted and the neutron is removed from the simulation

Absorption (Inelastic Collision): The neutron collides with an atom and sticks

→ The absorption event is counted and the neutron is removed from the simulation

Scattering (Elastic Collision): The neutron bounces off an atom (and loses energy?)

→ The scattering event is counted and the neutron is run through the simulation recursively until escape (with lower energy?)

1.3 Outputs

Number of each event

1. Fissions: Energy released, productivity of reactor
2. Escapes
3. Absorptions
4. Scattering

2 Project Flowchart

Initial conditions entered:

1. Reactor Radius
2. Cross Sections
3. Number of Neutrons

Neutrons 'spawned' with a position, direction, and energy

Time 'ticked,' moving neutrons by an amount dictated by density function in the initialized direction

Check what neutron does and perform recursion as necessary

3 Our Process

3.1 Starting

We began with simulating a single neutron to test procedures

Issues:

1. Position and direction each had 3 separate variables
2. When neutron was randomly spawned, it was not always inside the reactor

3.2 First Monte-Carlo

We attempted to loop the single example 'n' times but ran into an issue with creating many variables

This was solved by turning the positions and velocities into a (2, 3) array

3.2.1 Issues

1. The neutrons still did not always spawn inside the reactor
2. Probability density functions were not equipped to be used as a random input to another function
3. Loops were too slow for large 'n'

4. Escape tests extremely slow

3.3 Solving Issues

Speed: Neutrons were processed in a 3d array, removing the need for most loops

Probability density functions tweaked to allow mapping into other variables (screenshot pls this was such an annoying function to write)

Position: By taking the random variable (0, 1) as inputs and converting to spherical coordinates, the radius could always be kept within a sphere of 1 (screenshot this too)

Escape: Testing final position (radius) sped up significantly using Frobenius norm $\sqrt{x^2 + y^2 + z^2}$

4 Extra Additions