Bailey Oteri

May 5, 2025

CSCI 575 – Final Project

Multigroup Optimization with DQN (MGO-DQN):

1.) Abstract

This project investigates the use of reinforcement learning—specifically Deep Q-Learning (DQL) and Proximal Policy Optimization (PPO)—to generate an optimal 5-group energy structure from an initial 2000-group representation, using the BEAVRS benchmark. While most work completed with a PyTorch-based DQN model faced hardware and time limitations, a Stable Baselines3 PPO implementation successfully ran, though full model convergence was not achieved in time. Preliminary results show that reinforcement learning has strong potential for adaptive, data-driven group structure optimization, offering a promising path forward beyond the current standard cross section multigroup structures.

2.) Introduction

Optimizing energy group structures for multigroup neutron transport simulations is an unsolved problem in nuclear engineering. Many existing group structures were developed decades ago, with limited documentation on their original design rationale. Current approaches to optimizing group structures, such as simulated annealing, exploring the solution space effectively but struggle with the sheer complexity and inconsistency of the problem. Machine learning techniques, such as random forests, have demonstrated potential in selecting effective group structures, but they rely on predefined datasets rather than adaptive learning strategies. In this work, I propose the use of a Reinforcement Learning (RL) approach, specifically a Deep Q-Learning (DQL) algorithm, to dynamically generate and optimize a reduced 5-group structure from an initial 2000-group structure, using the BEAVRS benchmark reactor (1) as the starting point. Figure 1 below shows the general goal of the project, where continuous energy cross-section data is condensed to 2000 groups, then the optimal 5 group structure.

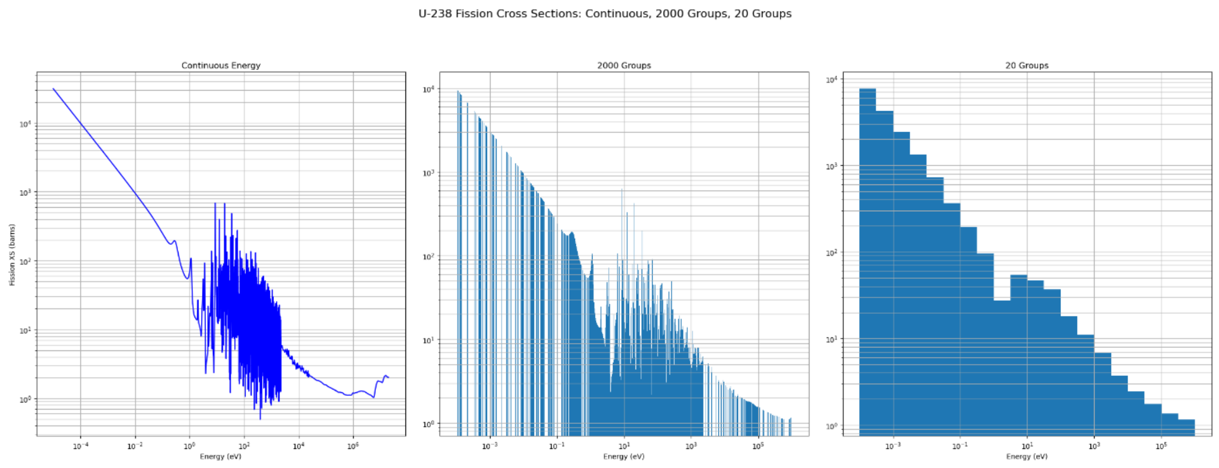

Figure 1: Fission cross-section collapse of U-235 (ENDFvii.1).

3.) Methodology

3.1) Multi-group Cross-Sections (MGXS):

Cross-sections are a way of defining the probability of a neutron having a specific kind of interaction with the area of a specific isotope [1]. These values are a function of the neutrons energy and range from approximately 1E-6 to 1E5 eV. Cross-sections are expressed in the units of barns [b], where 1 barn is equal to $1\text{x}10^{-24}$ cm^2. While this data is found experimentally, it is compiled into nuclear data libraries and used in nuclear reactor simulations industry wide. The problem with these libraries, however, is that they take up a lot of space in memory, and simulations that use the full library over all energy are very computationally expensive.

Because of this, deterministic codes exist, where the cross-section data is summed up to preserve the reaction rates into a specified number of groups. By using multigroup neutron transport models, the simulations can run quickly without overwhelming available memory while also accurately capturing the underlying physics of the neutron reactions. Deterministic codes also have their downsides, as they are not as accurate as their continuous energy counterparts, self-shielding of the cross sections must be included in the calculations, and the optimal group structures have not yet been found. Another layer of complexity is added when you change anything about the simulation being run, as the optimal cross-section group structure for one reactor might be completely different than another as things like materials and geometry definitions can easily drastically change the shape of the cross sections over energy.

3.2) Benchmark

For this project, I used the BEAVRS benchmark, a 4-loop Westinghouse plant simulation, licensed by MIT. BEAVRS is a Pressurized Water Reactor (PWR), that is composed of 193 fuel assemblies, which each have a 17 x 17 grid of fuel pins at three different enrichment levels (Figure 2). For this project, due to the time constraints, I worked with just one assembly that had 1.6% enriched fuel, with an instrument tube, and no burnable absorbers, shown in figure 3. For the assembly run of BEAVRS, 10,000 particles were simulated in 400 batches, with 200 being inactive. This resulted in a k_{eff} (effective multiplication factor) value of 0.992739 +/- 0.0005. For Monte Carlo simulations like OpenMC, the more particles simulated, the longer each run takes, but the lower the uncertainty on the k_{eff} value. Having 4 significant figures of uncertainty was a good balance between a relatively quick OpenMC run (a few min per run), while having accurate results. Because of this, our goal for our Δk_{eff} between our 5-energy group run and the continuous energy run had to be less than 0.0001.

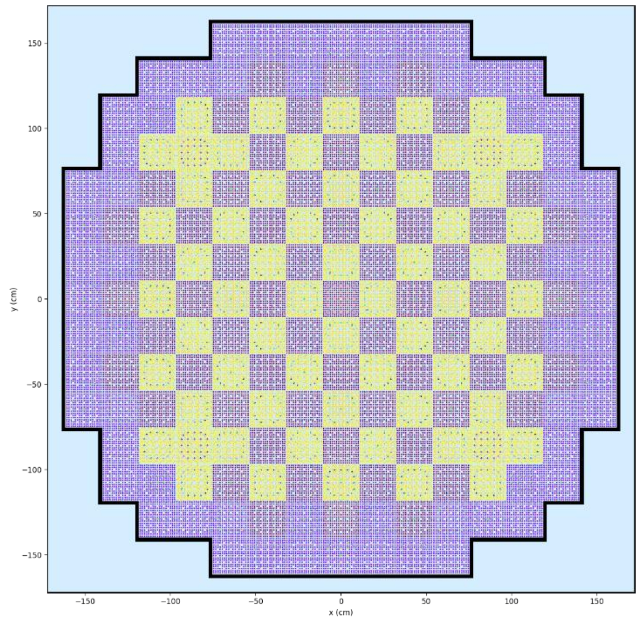

Figure 2: BEAVRS Reactor - Full Core

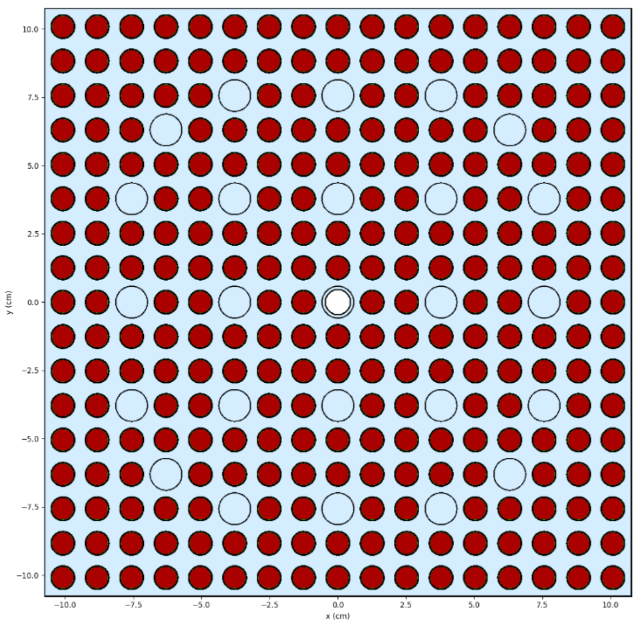

Figure 3: BEAVRS reactor - Assembly, 1.6 % enrichment

3.3) Pytorch DQN Approach – Original Plan

For this project, I decided to use PyTorch to create a DQN model that would try to determine the best group structure possible. This is the type of problem that can very quickly become overcomplicated, so for this first installation of this model I tried to go for as bare minimum as possible.
DQN Environment:

- **State Representation:** Defining the problem state based the materials, geometry, and settings in the BEAVRS model.
- **Action Space:** Modifying energy group boundaries and adjusting group widths dynamically to find an optimal 5-group structure from the initial 2000-group representation.
- **Reward Function:** Evaluating group structures based on their ability to minimize the difference in k_{eff} (effective multiplication factor) and reaction rates from the 2000 group structure.

3.4.) Stable Baselines3 PPO – Final Implementation

After a few weeks of continuous headaches using Pytorch's DQN algorithm, I decided to try Stable Baseline3's library instead to see if I could resolve the CUDA GPU errors I was receiving. I created a new environment and script to run the model, and it worked great, the only problem being that there was not enough time to sufficiently train a full model for my use case. When the complexity of OpenMC runs is added, a few episodes alone took over 24 hours, and it will take a lot more than a few episodes to create a fully trained model.

The environment was defined in a very similar manner to the DQN approach, the only difference being the reward function being purely binary, a 1 or a 0, and unless the Δk_{eff} was less then 0.0000, the model received a 0 for a reward for that episode. With the PPO algorithm, none of the parameters like the epsilon greedy function or learning rate had to be adjusted, which adds a lot of simplicity to the environment and runme scripts.

3.5.) Toy Problem

Unfortunately, due to the limited time to complete this project, a full model was not able to be made on time. However, a simplified version was created where OpenMC does not need to be run at every step, and the Δk_{eff} decreased with every step. While this is not the best for getting the model to learn, it did provide a starting point to build off to incorporate OpenMC and a more complicated problem.

4.) Results

As stated before, a full model using Stable Baseline3's PPO algorithm was not possible, however some very promising data was collected from the DQN runs (before a CUDA error would occur). In these results, depending on the how variables like the epsilon decay and learning rate were set, the DQN algorithm regularly finds group boundaries that would result in a Δ k_{eff} less than 0.0000. Figure 4 below shows some of the data from the DQN run that ran for 25 episodes, and while the k_{eff} does jump around a lot, figure 5 shows just how close most of the results to the goal k_{eff}, and most are within the uncertainty bounds.

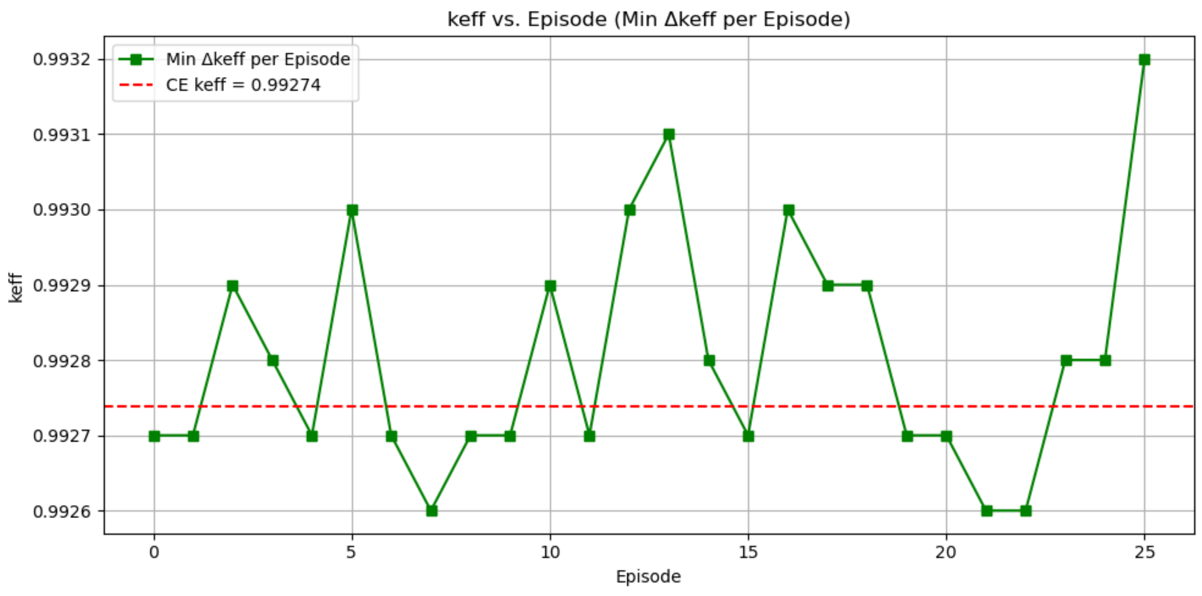

Figure 4: k_{eff} values correlating to lowest Δ k_{eff} in each episode.

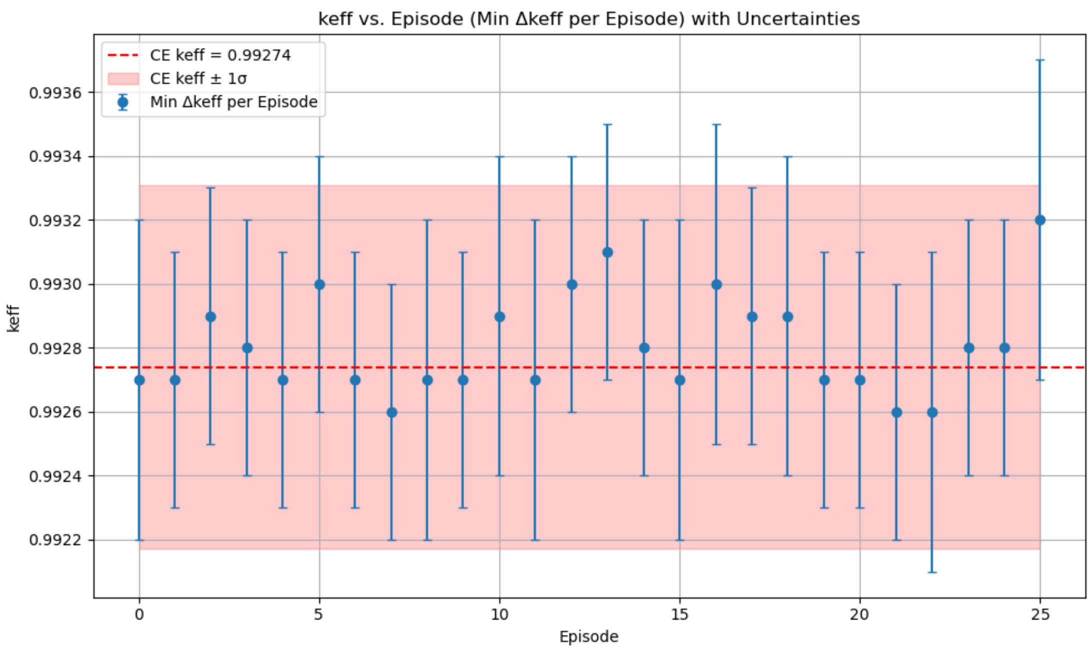

Figure 5: k_{eff} values from lowest Δ k_{eff} in each episode with uncertainty of continuous energy k_{eff} and uncertainty of each episode.

5.) Future Work

While the results of this project are promising, there is still significant work needed to complete a full model and demonstrate its effectiveness. The immediate next step will be to run the script to train a full model using the PPO algorithm on the BEAVRS assembly, likely using HPC resources. This process may take several days to complete. Once the model is trained, optimization strategies can be explored—such as implementing multiple fully observable agents, with one agent controlling each movable boundary, instead of relying on a single agent. In addition to model optimization, future work could involve applying this framework to other reactor types (e.g., fast reactors), which may not favor the equal lethargy group structure and could yield different Δ k_{eff} values when compared to continuous energy runs.

6.) References

1.) https://courses.washington.edu/mengr430/au07/handouts/cross_section.pdf

2.) N. Horelik, B. Herman, B. Forget, and K. Smith. Benchmark for Evaluation and Validation of Reactor Simulations (BEAVRS), v1.0.1. Proc. Int. Conf. Mathematics and Computational Methods Applied to Nuc. Sci. & Eng., 2013. Sun Valley, Idaho