Project 3 Editor Letter

December 16, 2025

Dear Allen Zhao,

Thank you for your feedback on my Project 3 submission. Here are my responses. I hope that they are sufficient.

> This is quite a deviation from the array-based Monte Carlo framework we introduced in class. If you are intent on using this, I would like to see a straightforward comparison with the method we've laid out for you which makes it clear that your implementation is equivalent and robust enough for the purposes of this project.

I spoke to Joss about this. I understand your concern, but I'm completely unsure what you mean by the "array-based ... method we've laid out for you," as I never followed any sort of tutorial or outline for performing this project. The starter code for this project (such as Homework 17) doesn't include a full process for this and Homework 16 isn't immediately applicable for these kinds of trees, so I'm not sure what you want me to compare against (as I've been using tree generation from the beginning of Homework 17). With that, Joss simply told me to ignore this.

However, despite that, I would still like to provide some justification for my method to qualitatively defend it regardless. I actually made a major mistake in my prior assumption that using the exponentially-growing tree would result in modeling being effectively impossible for generational depths that are too high. While it is true that a generational depth of, say, 20 could result in $2^{20} \approx 1.05 \times 10^6$ neutrons that have to be simultaneously modeled, this assumes that *no neutrons are lost*, and this is completely unrealistic for the simulation. Even the most stable simulations I'm modelling will lose large amounts of neutrons, so these systems certainly aren't going to be reaching such large numbers of objects being handled at the same time. In particular, the effective multiplication rate for stable systems has experimentally trended towards being just above $k = 1$ (reactor criticality), so the number of objects being managed is generally growing, but not by a considerable amount each generation.

> One major concern I have is that you are currently only modeling 3–4 generations of neutrons, which might not be enough to give statistically sound results (you need to verify this). If you end up needing to iterate over many more generations, I am not sure how performant an exponentially-growing tree of Python objects will be.

Honestly, you're right. I spoke to Joss about this as well, and the solution I've come to is that I could refocus my project to model long-run stable k values and produce a singular contour, and this would require determining the maximum generation depth required for a system to be stable across all of the systems.

> You have very well laid-out unit tests. This is great practice, but for the purposes of this project you can depend on "trusting the math" for simple expressions.

To keep this simple (and because of Joss' advice on my Project 2), I'll move all of the tests under validation to declutter the main section of the project.

> While your printed trees are a great debug tool, they're a bit cluttered to be included in their entirety in your writeup. They're definitely closer to debugging text, and it isn't very useful for the reader to interpret the larger ones. Maybe leave a small example at most.

Yeah, this is probably a good idea. I'll remove the longer ones, and just keep the example.

> Not sure how I feel about using ChatGPT to write unit tests...

I'm only using it for tests where I need to verify very basic things (like the shape of output vectors and whatnot), so the code for doing it is tedious rather than complex. If it works, it works.

> While your code and description of its implementation is great, this currently almost feels like the main purpose of your writeup rather than the investigation you should be carrying out for Project 3.

Honestly, you are right, and I feel as though the best way to rectify this is to apply the additions to the introduction that you request in your next point alongside some deeper consideration sprinkled throughout the document. This message is a bit vague (in comparison to your next point about the introduction) so I'm going to take the liberty of just assuming that the most critical areas to increase the level of specificity will still remain at the investigative sections (so the strictly technical sections, such as function definitions, will remain mostly just technical descriptions).

> Your project almost immediately launches into a technical review of code... without giving the reader any of the necessary context (what is a monte carlo simulation? what is the physical scenario you are investigating? how are you modeling your system?) to properly follow it. Imagine your writeup as a lab report. Prior to making any nontrivial calculations in your "Methods" section, you would want to first introduce the relevant formalism and equations you are working with.

To begin, I'm going to address this point by considering each of the questions that you're explicitly asking here one-by-one, and then I'll use my answers to each of those questions to produce the basis statement that you're asking for as one succinct description.

> What is a Monte Carlo Simulation?

"A Monte Carlo simulation is a general class of stochastic (random) methods for performing computational calculations. In this case, we're calculating a value through performing a high volume of random simulations to produce a distribution of values (multiplication rates), then selecting the mean and standard error of that distribution to act as the final calculation of the value."

> What is the physical scenario you are investigating? / How are you modelling your system?

"**General Background Information**

Consider the core of a nuclear reactor. neutrons are, obviously, much smaller than the gaps between atoms and their neutral charge causes them to not have attractive force to free electrons or any repellant force, meaning that they can fairly easily escape the metallic bounds of reactor cores once atoms are split through fission. Therefore, it's fairly accurate to model the neutrons in a reactor as

having no physical bounds whatsoever (in the form of a potential field blocking them from leaving), and this simulation seeks to investigate how often neutrons continue to replicate within this reactor.

As a nuclear reaction progresses, neutrons "replicate" into several more by triggering further fission reactions in other atoms, and the amount of free neutrons that are "produced" from the triggered fission by the previous neutron are considered to be successfully replicated. However, as the neutrons leak out of the reactor, they're unlikely to return within the core to continue replications, and it's also unlikely for them to replicate outside of the core, so for the purposes of this reactor, we're only interested in the neutrons that replicate within the core, and any neutrons that leak out of the reactor can be said to be effectively lost to the reactor altogether.

Therefore, we can model this system using a non-physical bound for the reactor core and a growing tree of randomly-generated neutrons for the replications: - The reactor core is like a "line in the sand" rather than a wall: it has no power to stop neutrons from exiting, and they will do so randomly, but once they cross the boundary, further progress is entirely not considered. - The neutrons are assumed to always induce a chain reaction that produces exactly two further free neutrons every time, but each neutron that is produced is then sent off in a randomly-determined direction and distance from the reaction that produced it, and if it ends up outside of the reactor core, then that neutron is considered "lost."

This, of course, requires an important distinction: each neutron is implicitly defined to replicate two further neutrons each time, but the effective replication rate is only considering the neutrons that remain in the reactor, so for example, a neutron could induce a reaction to produce two free neutrons, but if one neutron exits the reactor and the other ends within the reactor, then that neutron is said to have only replicated a single neutron, and the single neutron that remained in the reactor will induce another reaction.

Research Question

With all of the above information in mind, for a given core shape, the reactor core is going to eventually reach a specific replication rate. Unstable reactor geometries would have replication rates approach zero due to too much leakage, and stable reactor geometries will have replication rates that stay roughly constant at some non-zero value. The focus of this simulation, in particular, is to see how this long-run replication rate changes as the geometry of the core changes.

To do this, I've defined the reactor core's geometry as a rectangular prism, so for simplicity, we can call it a "box." This means that the box's geometry has three parameters: a length ℓ, width w, and height h. To keep this simulation simple, we'll make the length ℓ a function of the width and the height:

$$\ell = \frac{1}{2}(w + h),$$

Then, we can just vary the volume V and the aspect ratio R with each being explicit functions of only width and height:

$$R = \frac{w}{h} \tag{1}$$

$$V = w\ell h = wh\frac{1}{2}(w + h) \tag{2}$$

then each of the parameters are only a function of R and V:

$$\begin{cases} w &= R\left(\frac{2V}{R(R+1)}\right)^{\frac{1}{3}} \\ h &= \left(\frac{2V}{R(R+1)}\right)^{\frac{1}{3}} \\ \ell &= \frac{1}{2}(R+1)\left(\frac{2V}{R(R+1)}\right)^{\frac{1}{3}} \end{cases}.$$

Now we have a two-dimensional phase space for the simulation, aspect ratio R and volume V, and thus, each box geometry can be defined as a phase coordinate (R, V). For each geometry, we can perform a singular simulation with a random starting position, then record the replication rate k_i (for an individual simulation i) after a certain number of generations, g. (It's incredibly important that we pick g correctly to best characterize our system, but for now, we'll just vaguely state that it's a large enough number to characterize the long-term behavior of the system.) We can then perform this individual simulation a large number of times with more random starting positions, making the overall simulation f a Monte-Carlo simulation that estimates the long-term multiplication rate as $\mathbf{k}$ based on the mean and standard error, $\overline{k} + \delta k$. Therefore, thus far, we're producing a mapping using our simulation f:

$$f(R, V) \mapsto \mathbf{k} = \overline{k} + \delta k.$$

Thus, simply put, the overarching goal is to calculate this mapping for each (R, V) over some two-dimensional phase domain D

$$D = \{R_{\min} \leq R \leq R_{\max}\} \otimes \{V_{\min} \leq V \leq V_{\max}\}$$

and then analyze the results we observe.

As for the hypothesis, one can safely assume from basic intuition that smaller volumes should be less likely to retain neutrons, as at an infinite volume of the reactor core (and also assuming the entirety of all space is filled with fuel in the realistic case), the reaction will proceed forever and grow over all space. However, the aspect ratio is much more interesting. We can reasonably assume that there should be the greatest preference to more even aspect ratios (closer to 1:1), because if the width, height, or length of the box is approximately zero, any movement of a neutron along the approximately zero axis in three-dimensional space will result in ejection."

> Fig2/3: There seems to be a misconception of what k is. You should treat k as a macroscopic constant intrinsic to a given test structure - not neutrons. By performing this Monte Carlo simulation, you are ultimately trying to compute the best possible "measurement" of k by averaging out the statistically-distributed results of many "experimental" simulations.

I understand what you mean. I think the major error in my work is in Figure 2 with the line "The first generation can only ever replicate 0, 1, or 2 neutrons exactly, so an individual neutron i will have exactly $k_i = 0, 1, 2$: a discrete range. So, in short, because the range of possible k values for a given neutron is so limited, the negative contribution from low k values will be much higher." and in figure 3, with the line "The first generation will usually have a replication rate of $k = 2$ or $k = 1$". I think the best remedy for this is just switching the entire project to use long-run k-values, as stated previously.

> If your k values appear to still change significantly across generations (e.g. in Fig2), what does this tell you about the accuracy of your current measurement? How can you ensure this value of k has stabilized before taking a measurement? Try running your simulation for additional generations and plotting k vs generation number. Consider the relative uncertainty of your average k measurements in Figures 3–4. Does this change with volume and aspect ratio?

Yet again, there is a lot to be addressed in this point, so I'm going to answer each of your individual questions, but the overall remark is that my simulation, as it stands, isn't stabilized for the number of generations that I'm considering.

> If your k values appear to still change significantly across generations (e.g. in Fig2), what does this tell you about the accuracy of your current measurement?

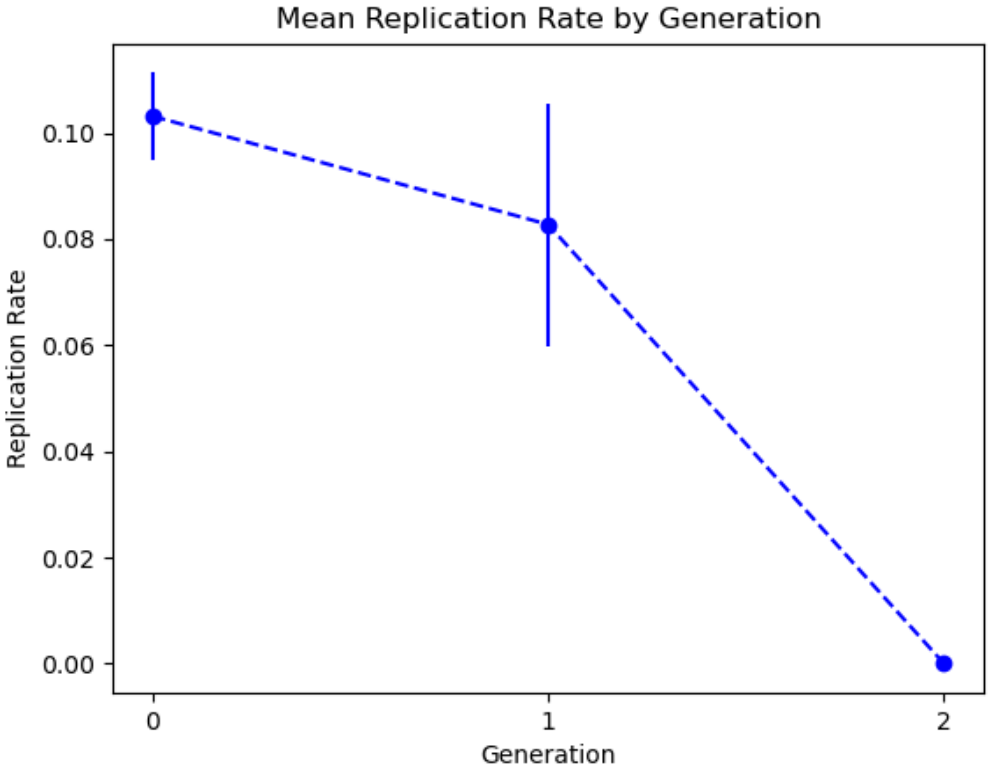
Very simple: my simulations are just not stabilized. High relative error on k is an extremely good indicator that the simulation is not yet long-run stable, so I'll need to increase the number of generations significantly.

> How can you ensure this value of k has stabilized before taking a measurement?

I discussed this with Joss, and the solution I've reached is to pick a minimum-stable-generation g, justify this based on empirically showing which reactor geometries require longer generational depths to stabilize, then simply using that generation depth for all of the future Monte Carlo simulations.

> Consider the relative uncertainty of your average k measurements in Figures 3–4. Does this change with volume and aspect ratio?

This is a difficult question to answer, because the uncertainty in the k value is not something that is consistent from generation to generation. For example, in Figure 6,



the uncertainty spikes just before all systems completely destabilize and both the multiplication rate and the uncertainty go to zero. This is interesting behavior for sure, and it's insightful for understanding the oddly random trends seen in the final contour, but it's not something that's particularly consistent enough or really representative to make a greater overall claim about how the uncertainty in all of the k measurements is a function of the geometry of the box. I'll make sure to discuss the role that uncertainty is playing here in interpreting random behavior from the simulation, but fundamentally, the goal with the uncertainty through my investigation is simply as a guidepost for understanding what parameters need to be given to minimize this uncertainty enough that our results are accurate.

> You need a much more thorough discussion of the uncertainties in your data and their implications on how you can interpret your results. Is 20–40 replications enough to average out statistical differences in this case?

This is a good point. I should take proper care in picking a standard replication rate alongside the generational depth to minimize standard error as well, so after determining g, I'll also determine a good minimal replication rate for producing a relative error for in the long-run multiplication rate less than some very small maximal threshold.

I've added this section as "Determining the Minimal Replication Rate for Producing Accurate Aggregate Multiplication Rates."

> You are also making a lot of assertions about global minima and maxima in your case studies without actually modeling the data you are working with.

I'm slightly confused by what you mean by my not "actually modeling the data [I'm] working with," but I assume that the problem is my referring to these points as "global maxima" or "global minima" without explicitly finding the global maxima and minima directly from my calculated data. In any case, it's probably more accurate to examine these points in a much broader sense, considering the point $(R, V) = (1, 200)$ as reflective of the broader class of even aspect ratios and large volumes and the point $(R, V) = (10, 1)$ as reflective of the broader class of skewed aspect ratios and small volumes, and I can safely make much more general statements about how this relates to the overall trend of reactor stabilities without decreasing the quality of my conclusions/reflections.

> Beyond your heatmaps, the majority of your project focuses more on the properties of the process we are modeling rather than a study of system behavior vs your independent variables. You should perform a thorough and ideally quantitative analysis of how k scales with volume and aspect ratio. One way to do this is to fit cross-sections of your heat map data to a model of your devising.

I see what you mean, and your idea is actually perfect as-is. I'll just do exactly that, and add a dedicated section for it. It's quite long, so I won't be adding it to this document, but it can be found under "Phase Exploration: Cross-Sectional Analysis."

> Appendix: While it's great that you discuss the various validation and unit-testing techniques used to develop your project, you should also include some concrete and followable instances of code validation. An example of something you could check here is whether or not your neutron population behaves as expected for a given multiplication factor.

I understand what you're looking for here. Now, I'm a bit confused how I would implement your given example as the multiplication rate is what we calculate rather than the parameter I'm using. What would realistically make far more sense here is a situational validation similar to the kind I performed on project 2, where I begin by considering trivial cases where their behavior should be known and expected then see how neutrons develop from there. I did a very minor example of this through the ending of the phase investigation, but what in this case, I can consider points far outside of my phase investigation to produce truly trivial situations (like the aforementioned "infinite box" with a size much greater than the mean-free-path or a box with a volume approaching zero).

> I'm unsure if your current experiment and its results generalize well enough to assert that a sphere would be ideal. What additional testing could you do to solidify this claim?

You're absolutely correct. I definitely don't have enough to assert that a sphere would be ideal, and I understand that it would be possible to directly verify this claim through directly modelling it at the end of my simulation, but I actually no longer believe that it is helpful to my investigation whatsoever to add this claim. It makes much more sense to only directly state exactly what was proven through my investigation under my claims then move this section of the discussion about non-rectangular geometry as a future hypothesis/research question under "future research" with the basis that this study serves as a good starting-point for trying to undergo this research.

I hope that this is good enough! Thanks again for the notes, and thank you for the great term.

Regards,

Mufaro Machaya

[ ]: