Introduction

This project is about the shield of a neutron reactor. This system contains three parts: the reactor, the shield and the outside. A neutron will move towards the shield from the reactor. When the neutron is randomly moving inside the shield, it can move to 4 different directions. It will lose energy as it moves, which cause three different destiny of this neutron. The neutron has a probability P1 to move back to the reactor, a probability P2 to move inside the shield (being captured by this shield), and a probability P3 to go through the shield to go outside. The nature of this system is simply about the locations of neutrons after their energy runs out.

Theory and method

In this work, I use the location of neutrons when their energy runs out to find out the probabilities. I start with 10,000 neutrons, they all have the same first step, which is to go into the shield, the shield locates between x=0 and x=X, where X is in the unit of steps.

Assuming neutrons move 1 step each time and lost 1 point of energy after the moving. If the neutron moves one step along the x direction to x=X, then its distance to x=0 along the x direction will be +1. If the neutron moves one step along the x direction to x=0, then its distance to x=0 along the x direction will be -1. If the neutron moves along the direction perpendicular to x direction, then its distance to x=0 along the x direction doesn't change. After adding up all the steps and moves, the neutrons location along x direction can be easily found. If its distance is inside the range between 0 and X, then the neutron is captured by this shield. If its distance is larger than X, then the neutron has left the shield. If the distance is a negtive value, then it means the neutron has returned the reactor.

I didn't use any equations in this program. It is just simple Monte-Carlo calcutions. Use random number to show the direction of the neutron's motion. If I have to say something about limitations, it will be the probabilities of the neutron's motion. We need to know the probability of this neutron to move toward different directions.

In this project, I use 4 different values to represent 4 different direction. I use random.random() to obtain random value in the range of 0~1 to represent the probabilities to find out which direction the neutron is moving to.

The first step of neutron is already known for sure, the partlce will move into the shild in its first step, so its distance to x=0 will be x=+1. Which means the neutron moves toward right at its first step.

We considered about the first trial conditions at the beginning. In this case, the neutron cannot step back at its next step, and the probability to go forward is twice more than changing a direction. Its energy can support it to move 100 steps. So the neutron has three choices for the second step, it has 2/3 chance to continue moving toward right, 1/6 chance to move up, and 1/6 chance to move down. Which means the neutron has 2/3 chance to obtain a distance x=+1+1, and 1/3 chance to obtain a distance x=+1+0.

In the third step, the neutron has more options. If it move towards right in second step, it will be in the same situation as in step 2. But if it move upward, then it will have 2/3 chance to continue moving to up, and 1/6 chance to move to right (to get another +1 in its distance), and 1/6 chance to move to left (which will obtain a -1 in its distance). Things are similar if the neutron move downwards in its second step.

In the forth step, we only have one new situation. If the neutron move to left in its third step, then it has 2/3 chance to continue moving back to the reactor, only 1/3 chance to move toward up or down.

After finding out the rule of the neutron's motion, I use a "for loop" and "if function" to simulate the motion of neutrons. I use a sample with 10,000 neutrons, and after running the code, I obtain the location of all those samples.

Neutrons with a location smaller than 0 are already back to the reactor. The number of negtive values divided by the total number of neutrons will be the P1 (the probability for neutrons to move back to the reactor). I run this code for different X value (I create an array of X, it starts at 1, ends at 100, and with a interval of 1), then I found the P1 for different X.

Similarly, I found the P2 (the probability for a neutron to be captured by the shield), its location is between the range of 0 and X. Also, the P3 is obtained. P3 is the probability for a neutron to go through the shield, those neutrons have a distance greater than X.

Results and verification

For the first trial conditions, the number of neutrons in different situation can be found in the plot as below.

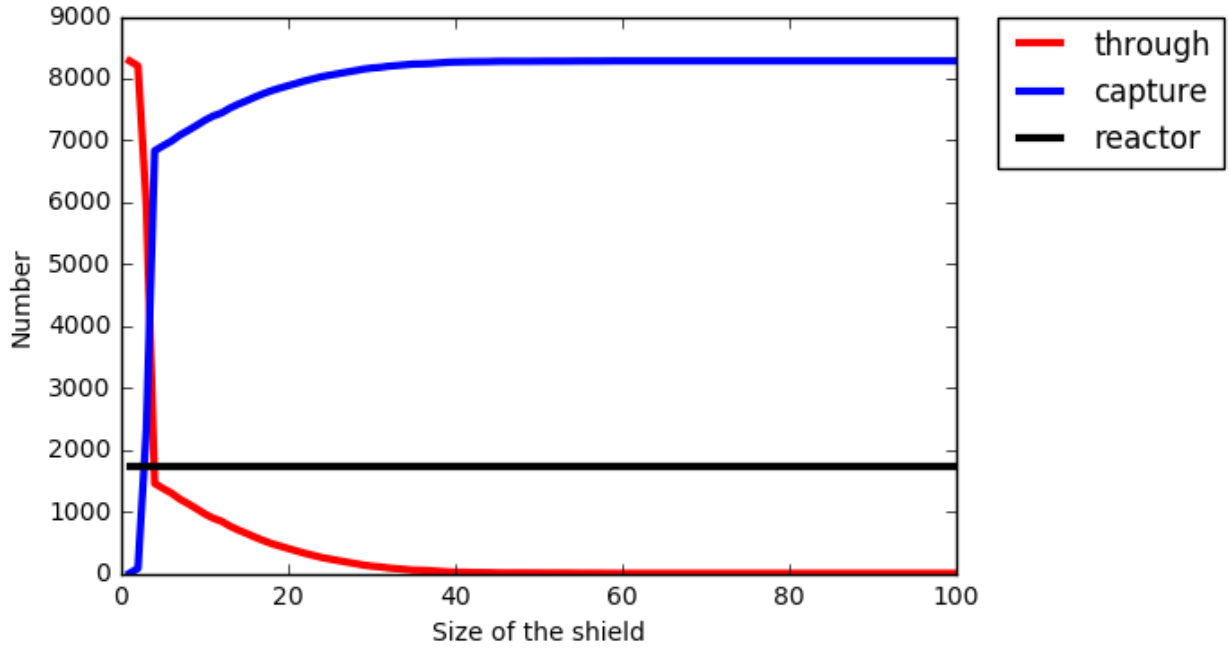

Figure 1

From this graph, we can tell that the probability of through should be in the form of P3 ~ e^{-ax}, and the probability of capture should be in the form of P2 ~ e^{bx}, and the probability of reactor should be in the form of P1 ~ c. In which, a, b and c are all positive variables.

After applying polyfit function, we found the relations:

$$\ln(P1) \cong -1.8073$$

$$\ln(P2) \sim 0.0139x$$

$$\ln(P3) \sim -0.1947x$$

So, the constant a mentioned above should be 0.1947.

After this, I changed the probability for neutrons to go forward as three times more than changing a direction. So at this time, the neutron has ¾ chance to go forward, and ¼ chance to change a direction. After similar calculations, we have the graph as below:

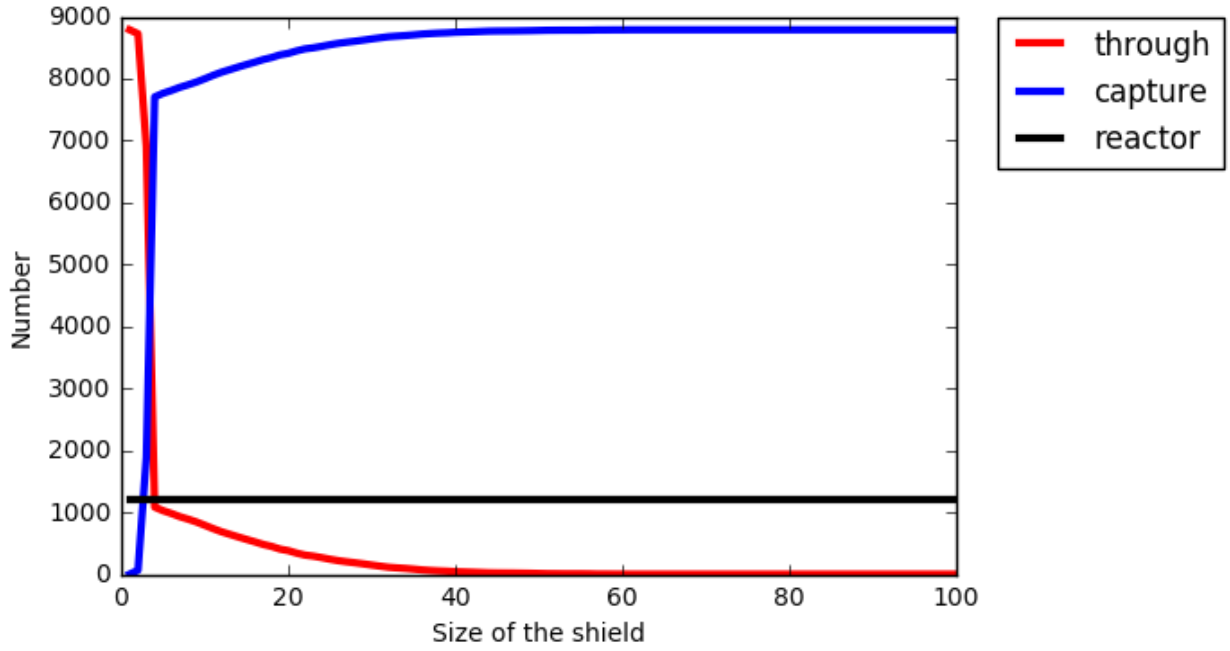

Figure 2

After applying polyfit function, we found the relations:

$$\ln(P1) \cong -2.1108$$

$$\ln(P2) \sim 0.0141x$$

$$\ln(P3) \sim -0.1903x$$

After these, I start to consider about the normal distribution of neutron's energy. Assming the energy distribution has a mean value of 100 steps, and a standard derivation of 20 steps. In this case, the energy changes, I use random.random() to obtain values between 0 and 1 to represent the probabilty of a certain energy. Other parts of the calculation is all the same.

This time, I consider two different situation. Since we have a normal distribution of energy here, so I make two calculations, one is about the energy in the range of 0~100 steps, another one is about the energy in the range of 100~200 steps.

For energy in the range of 0~100 steps, we have similart plot as below:

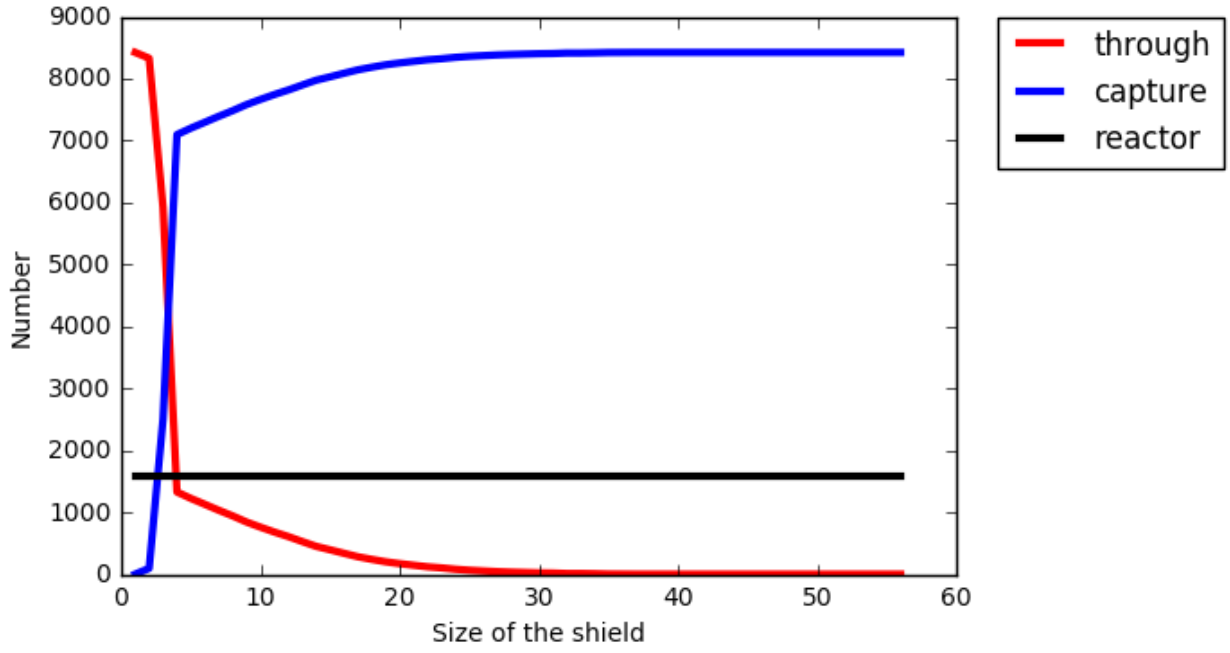

Figure 3

While for energy in the range of 100~200 steps, we have another plot:

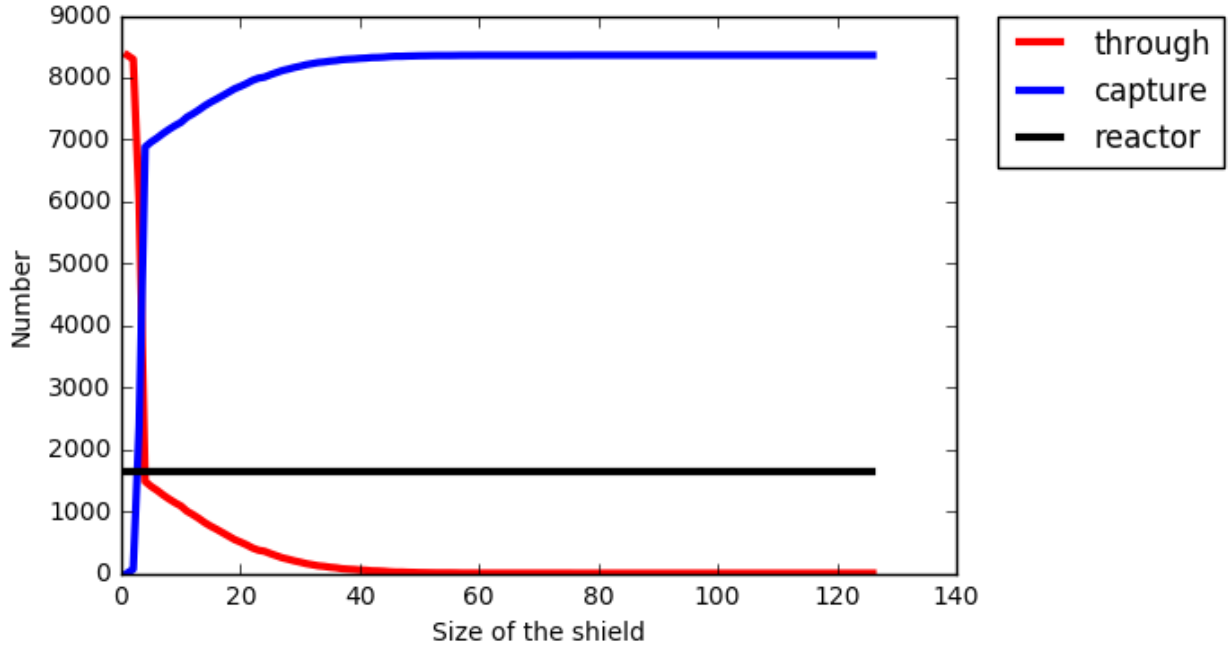

Figure 4

Still, we consider different probability for neutron to go forward as well. Those plots are shown as below:

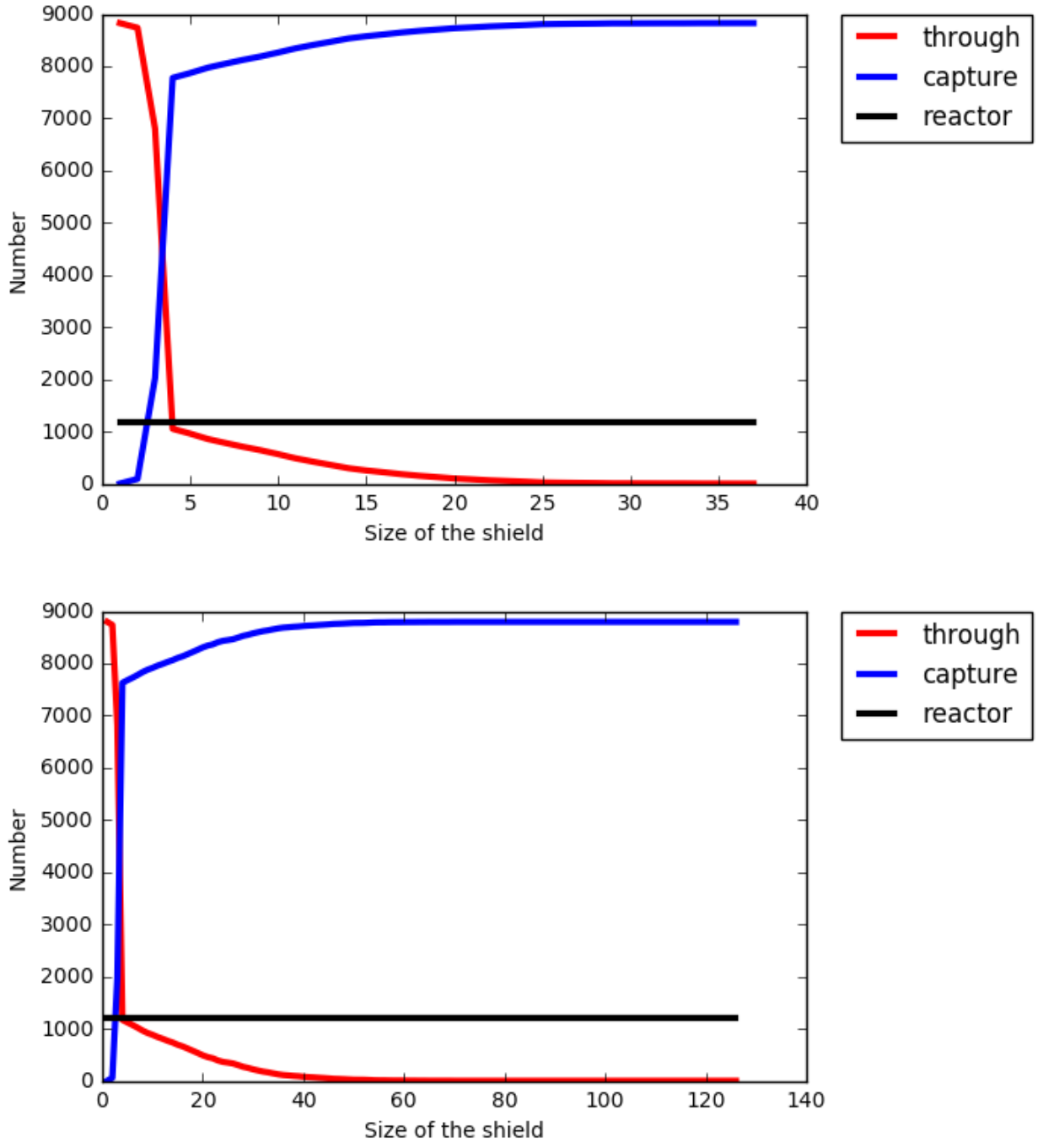

Figure 5

Since we are using Monte-Carlo calculation methods in this project, it is necessary to verify our work. I use random values to represent the probabilities to choose direction and also the energy, if the program is working properly, all the results should have something in common, the a, b and c values shouldn't be

to much different in different simulations. So I run the codes for a few times, and make three plots as below: (for capture, through, and reactor, respectively):

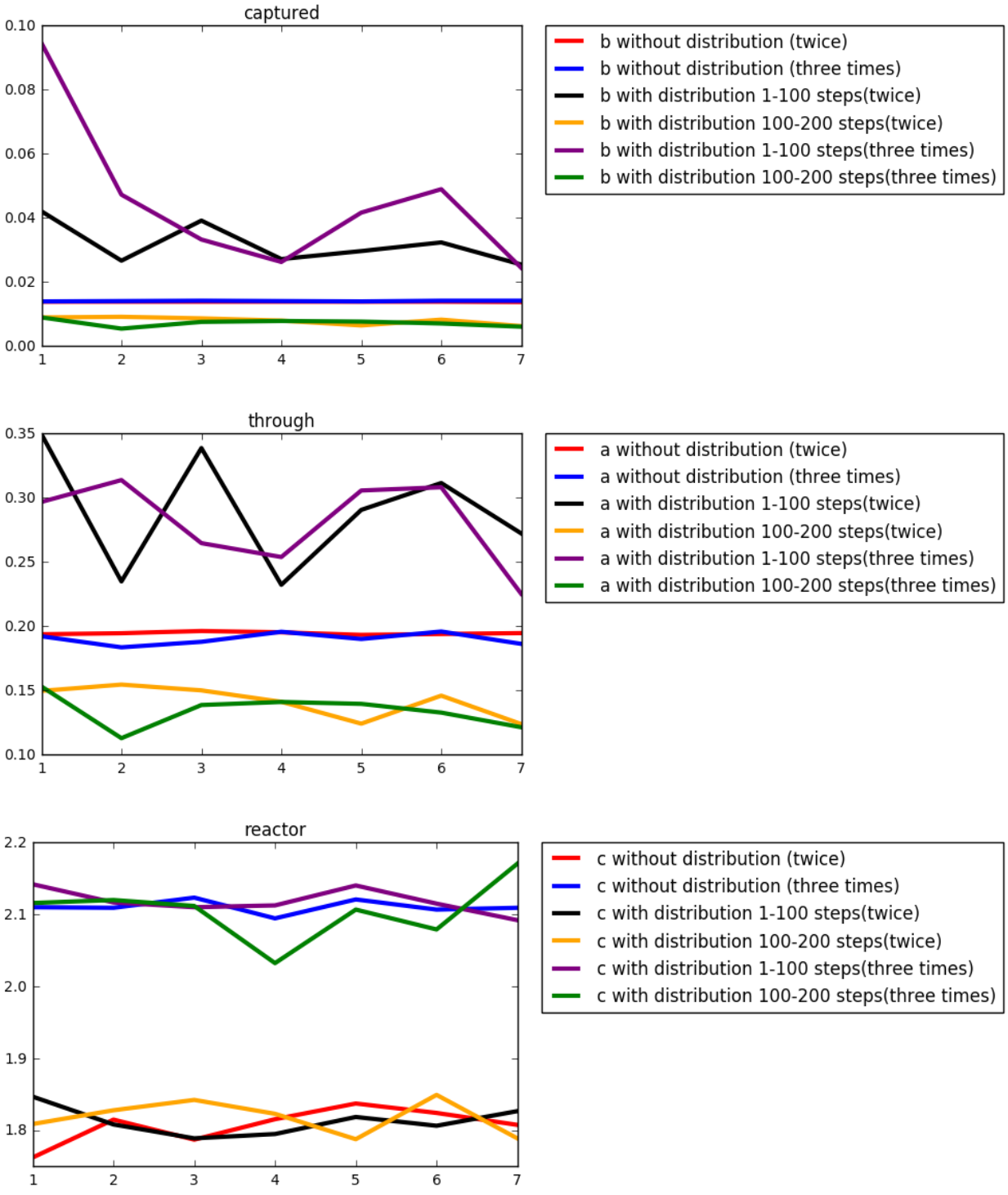

Figure 6

Based on these plots, we can see that for neutrons under same situation, their a,b, and c values doesn't change that much. After consdering the normal distribution of energy, a, b and c changes a lot, but that is caused by the change of energy. Since we considered a normal distribution, our energy is not the same one for every time. And energy matters a lot in the motion of neutrons. And also, for the curves with different probability to go forward, the trends are quite similar, which also proves energy affect our probabilities a lot. Even though it is not that perfect, but I think I can say my calculation is correct.

Analysis

Based on my results, it is easily to find the difference of the three probabilities for neutrons after a random walk.

A very interesting thing can be seen from those plots, which is how the size of shield affect those probabilities. As the size increasing, more and more neutrons will be captured inside, and only very few neutrons can run out. It is certain true that the larger shield works better, but we don't really need a shield larger than 40 steps. From those plots, we can find that most of neutrons are captured when the shield has a size of 40 steps. I think this is very important for the application of reactor shield in reality.

The probability of neutrons go back to reactor seemes to be a constant as the size changes. From figure 6, we can see that when the probability for neutron to step forward is the same, their probability to go back to the reactor is almost the same. So I assume this probability is related with the probability of motion, and doesn't related to the size of shield or the energy of neutrons.

Also, it can be found from the through plot in figure 6, neutrons which have more energy have smaller a values. Recall we have the probability of neutrons to go through the shield as $P \sim e^{-ax}$, so higher energy neutrons have higher probability to run out from the shield. Which can also be proved by the capture plot, since the lower energy neutrons have larger b values, which means they have higher probabilities to be captured.

Critique

I learned a lot python skills during these lectures.

First thing is about some basic programming. The array and list is something very new to me, I didn't really think too much about them when I was using IDL earlier. I feel list is really convenient and useful when I was running a for-function.

The plot function in python is really nice. It makes life much easier compared with IDL. Also I learned how to make simple animation, that can be very useful later.

The Monte-Carlo method is not really new to me, but it is really fun to use them in python.

The most important concepts for me is the integral and differential equations, they were really a pain to me, and I feel they are still a pain now. That part of lectures contains too many methods and equations, I feel things will be better if we can have more class time on that part. Those integral and differential equations is important in reality, but I never thought they can be that hard in programing. I guess more practice on that part may be a little helpful, espectially like how to choose a method to do the calculation.

Appendix

All the program and png files of output have been uploaded to github (my user name is veoj). Which includes:

1— The program for first trial conditions.
   a. The name of code file is JingSun_project_shield_first, it has a jupyter notebook file and also a .py file.
   b. The plot is shown in part_a.png, which is the relation between number of particles and size of the shield.

2— The program for second consider with normal distribution
   a. The name of code file is JingSun_project_shield_second_1-100steps and JingSun_project_shield_second_100-200steps, because I separate the energy range as 0~100 and 100~200 to obtain lower-energy neutrons and higher-energy neutrons. These codes has a jupyter notebook file and also a .py file.
   b. The plot of 1~100 steps is shown in part_b1.png, the plot of 100~200 steps is shown in part_b100.png. They are also show the relations between number of particles and size of shield.

3— The program for a different probability to go forward.
   a. The file with a name of JingSun_project_shield_third (jupyter notebook and also .py) is the one in the first trial conditions, only the probabilty to step forward is changed.
   b. The file with a name of JingSun_project_shield_third_1-100steps (jupyter notebook and also .py) is the one in normal distribution and in the range of 1~100 steps, similar with the one in second consideration, only the probabilty to step forward is changed.
   c. The file with a name of JingSun_project_shield_third_100-200steps (jupyter notebook and also .py) is the one in normal distribution and in the range of 100~200 steps, similar with the one in second consideration, only the probabilty to step forward is changed.
   d. The plot is shown in part_c.png, part_c1.png and part_c100.png ,respectively.

4— A file for varification.
   a. The file with a name of compare (jupyter notebook and the .py file) shows the verification of these program. I run each code several times to obtain some a, b and c values, to make the figure 6.
   b. The plot is shown in captured.png, reactor.png, and through.png