CENG 222

Statistical Methods for Computer Engineering
Homework 4 Report

Alperen Oğuz Çakmak
2237162

While conducting the monte carlo simulations, homework text states that each option should be calculated with 0.98 probability that the numbers do not differ from the true value more than 0.03 which leads me to calculating N with the biggest value that $p * (1 - p)$ can get and that is 0.25. Calculating the number of calculations for the desired accuracy for the simulation we use the 'Size of a Monte Carlo Study' formula in page 116 of the textbook which is $N \geq 0.25(\frac{z_{\alpha/2}}{\varepsilon})^2 = 0.25(\frac{z_{.0.01}^2}{(0.03)^2}) = 1502.85$ for $\alpha = 0.01$ *and* $z_{\alpha/2} = 2.326$ so that N will be at least 1503. In other words, we need to conduct simulations at least 1503 times if we want to differ from true values no more than 0.03 with the probability of 0.98.

Part A)

In my matlab code function part_a() represents the first simulation which is done for the purpose of answering part A. It basically sets corresponding parameters of the Monte Carlo study such as N, p, lambda, some matrices and so on. After that function goes into a loop which will be iterated N = 1509 times. In this loop, at every iteration, I created a new poisson random variable with $\lambda = 160$ to find the number of items and also created an adjacency matrix with the size as this number for representing the compatibility of the items with each other. I filled the matrix with binomial random variables. Since p=0.012, if the corresponding matrix element is lower than 0.012, its corresponding row and column items are compatible and we write 1 to the new matrix, otherwise 0.Then I cleared the diagonal elements and also copied the lower triangular part to the upper triangular part. Result matrix is an adjacency matrix indicating each item pairs' compatibility. After that I calculated the total number of possible triplets with the trace formula given in the homework text and stored the result in the result matrix which is

1xN because there will be N numbers of outcomes. Lastly I transformed this matrix into another matrix which holds 1 if the number of possible triplets is no more than 1 and 0 otherwise. Last matrix is the desired for this question. Calculating the mean of that matrix gives us the result probability is approximately 0.68.

Part B)

For this part, I have written a function which is part_b(). The structure of this function is the same but p is different. This time p=0.79 indicating a good chance of 2 items being compatible. After constructing the adjacency matrix, the number of possible triplets is again calculated with trace formula from the textbook but instead of storing that value in the result matrix I stored the ratio of this number to the number of all triplets as the question implies. Lastly I transformed this matrix into a matrix that holds 1 if the ratio is more than 0.5 and zero otherwise so that mean of the resultant matrix is the answer to this part which is 0.15.

Part C)

I implemented calculations of part C answers into the functions that calculate part A and part B.

For the X values of simulations, the first simulation's X value can be calculated easily since part_a() function already stores each number of possible triplets and stores them in 1 x N matrix. Mean of this matrix gives us the result for the first simulation's estimated X value. The second simulation's estimated value X can be calculated by holding the number of possible triplets in another matrix (since we already calculate that) and getting the mean of that matrix is the value for estimated X.

For the Y values, An extra matrix can be implemented which holds the Y ratio for all iterations. Mean of this matrix gives us the estimated value Y for the first simulation. For the second simulation, part_b() function already calculates this ratio and stores it in a matrix. Mean of this matrix is the estimated value Y for the second simulation.

	X	Y
First simulation	1.20	$1.7* 10^{-7}$
Second Simulation	336000	0.49

These values are approximated, more accurate results can be shown from the code.

Part D)

As I did in part C, this part is also calculated in part_a() and part_b(). Since both functions have matrices which hold N number of different X and Y values and part C is calculated by getting mean of them, instead of mean we can calculate standard deviation of these matrices and that's the answer.

	Std(X)	Std(Y)
First simulation	1.20	$1.7* 10^{-7}$
Second Simulation	81000	0.006

These values are approximated, more accurate results can be shown from the code.

As one can see that standard deviations of X and Y for the first simulation are so close to the estimated X and Y which means that among 1503 sample X and Y values the variables have a good chance of being distant to the estimated mean. We also know that we calculated the probability that at most 1 choice is available and X values are the number of possible choices of triplets. X = 1.20 is so close to 1 and Std(X) = 1.20 means that at least half of the variables will be less than 1. That explains why the result is 0.68. Y is also the ratio of X values to the number of distinct triplets so that one is accurate too.

For the second simulation we want the mean of Y values to be at least 0.5 and our estimated Y is also 0.49 but Std(Y) = 0.006 is so low that it implies that a high number of variables are around estimated Y value. This explains why the probability of getting the ratio at least 0.5 is a low probability (0.15). Since Y values are calculated by dividing X values to the number of distinct triplets and Std(X) is low too compared to estimated X value, the same comment could be done for that estimators accuracy too.