448 Project Report

Mingfu Liang, Student ID:3146919, NetID:MLQ4767

February 2019

1 Problem 1 & 2

Please check for my handwritten version.

2 Problem 3

2.1 Problem 3(a)

First I plot the x_n as a function of n as Figure (1) and we can see that the Linear congruent random number generator seems good when $a = 1229, c = 1, n = 1000$.

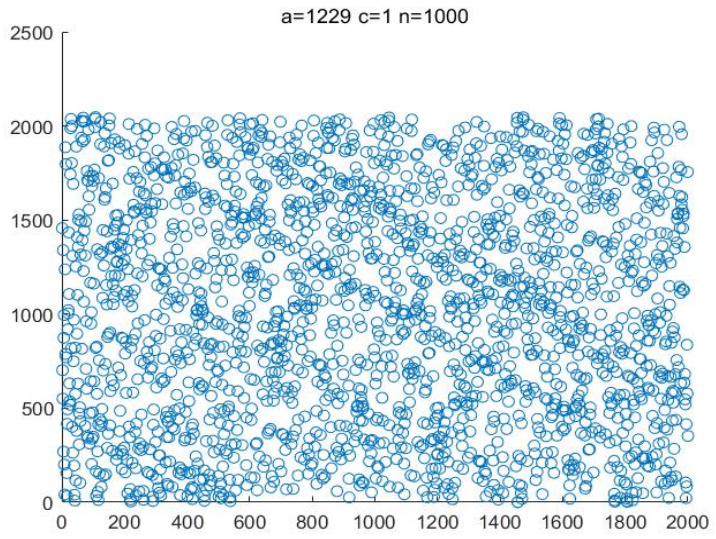

Figure 1: Plot x_n as a function of n.

However, when we plot the scatter plot of x_{n+1} vs. x_n as shown in Figure (2), it is obvious that x_{n+1} and x_n have kind of correlationship, which we will evaluate on the next sub-problem.

What is more, I also do same experiments on $a = 4, 61, 1891$, I see that if the a is not relatively prime with M, which is 2048, then it fails to generate random number by using Linear congruent random number generator. The different choose of a that is relatively prime with M will also have different performance in the scatter plot of x_{n+1} vs. x_n.

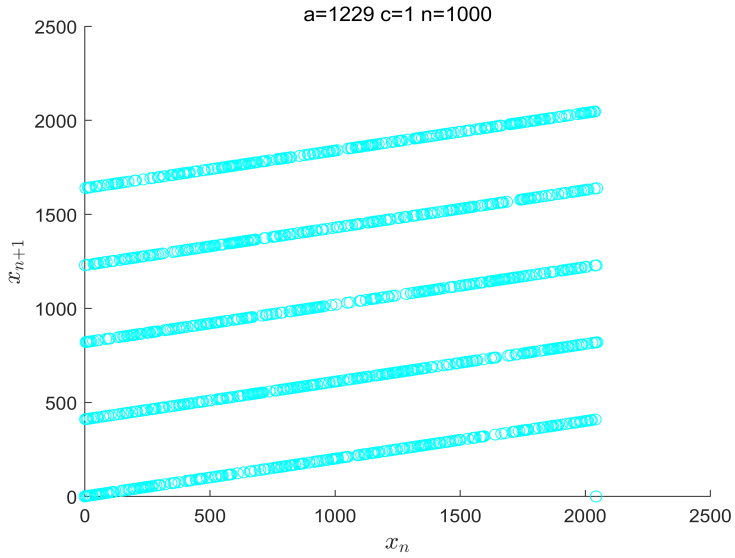

Figure 2: scatter plot of x_{n+1} vs. x_n.

2.2 Problem 3(b)

Figure (3) and Figure (4) are the histograms of covariance of $X = x_{2n-1}$ and $Y = x_{2n}$ for 50 different values of seed x_0. Figure (4) is the normalized of Figure (3), which are the same. For these two Figures, the $x - axis$ are covariance values and the $y - axis$ is the frequency after using 50 different values of seed x_0 in generating random number with Linear congruent random number generator. Here I use *randperm* function in $MATLAB$ to generate different random integer x_0 since it can generate non-repeat random integer numbers in $MATLAB$. We can see that the covariance are so big that the range is between $6.4 * 10^4$ to $7.4 * 10^4$.

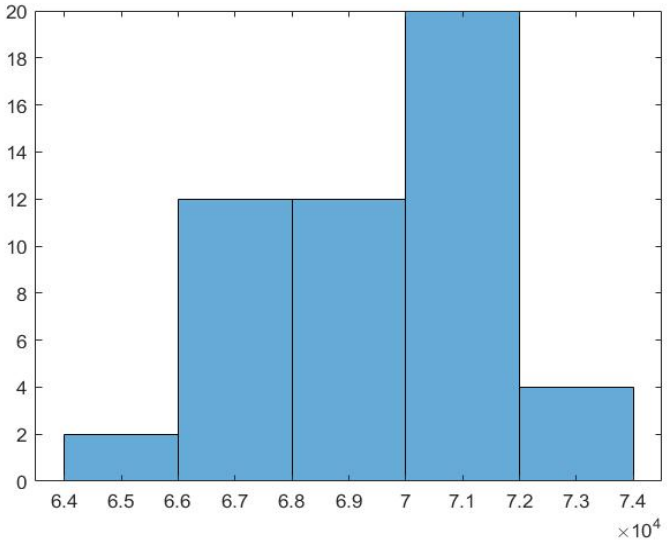

Figure 3: Frequency histogram of covariance of $X = x_{2n-1}$ and $Y = x_{2n}$ for 50 different values of seed x_0.

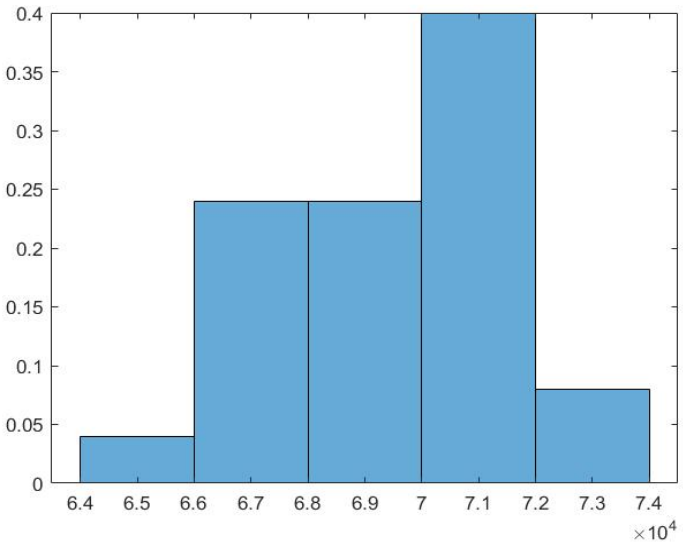

Figure 4: Normalized Frequency histogram of covariance of $X = x_{2n-1}$ and $Y = x_{2n}$ for 50 different values of seed x_0.

2.3 Problem 3(c)

Repeat parts (a) and (b) using Matlab's *rand*. First let's see the plot of the x_n as a function of n using *rand* in $MATLAB$.

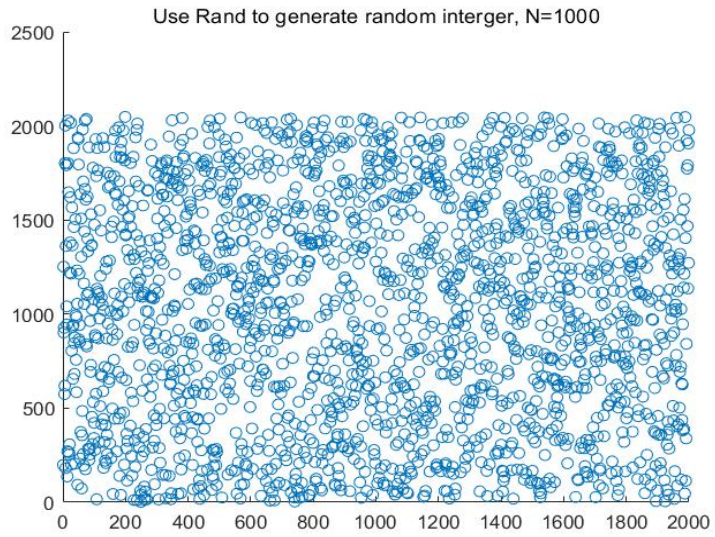

Figure 5: Plot x_n as a function of n using *rand* in $MATLAB$.

From Figure (5) we see that we can not find any difference between using Linear congruent random number generator and *rand* in $MATLAB$. So let's look at scatter plot of x_{n+1} vs. x_n generating by *rand* in $MATLAB$. From Figure (6) we can see that there is no obvious pattern or relationship between x_{n+1} and x_n generating by *rand* in $MATLAB$, which is different from Linear congruent random number generator. Now let's check the covariance histogram.

In Figure (7), the $x - axis$ are covariance values and the $y - axis$ is the frequency after using 50 different values of seed x_0 to generate random number with *rand* in $MATLAB$. We can see that the covariance of $X = x_{2n-1}$ and $Y = x_{2n}$ for 50 different values of seed x_0 using *rand* in $MATLAB$ is in the range from $-1.5*10^4$ to $1.5*10^4$, and most of them are concentrating at 0, which means that the covariance of $X = x_{2n-1}$ and $Y = x_{2n}$ is not determined or highly equal to zero and they are more randomly.

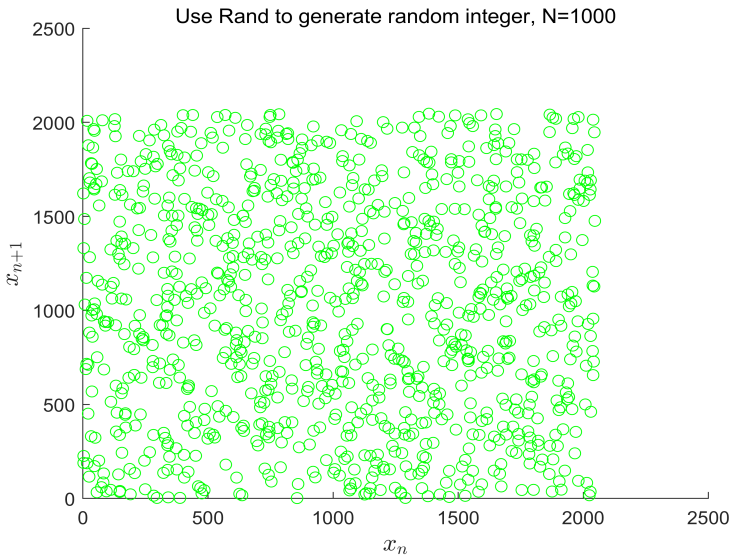

Figure 6: Scatter plot of x_{n+1} vs. x_n generating by *rand* in $MATLAB$.

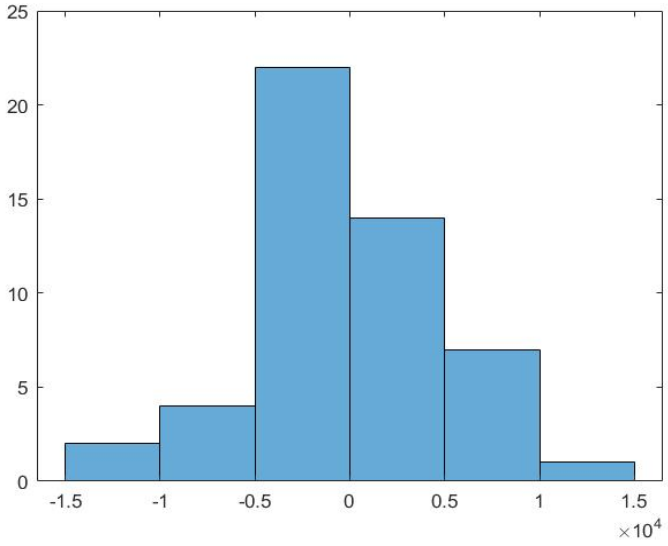

Figure 7: Frequency histogram of covariance of $X = x_{2n-1}$ and $Y = x_{2n}$ for 50 different values of seed x_0 using *rand* in $MATLAB$.

2.4 Problem 3(d)

From the discussion above, we can see that the Linear congruent random number generator is worse than the *rand* in $MATLAB$. The random number generated by Linear congruent random number generator has higher covariance which means that the method is not random enough.

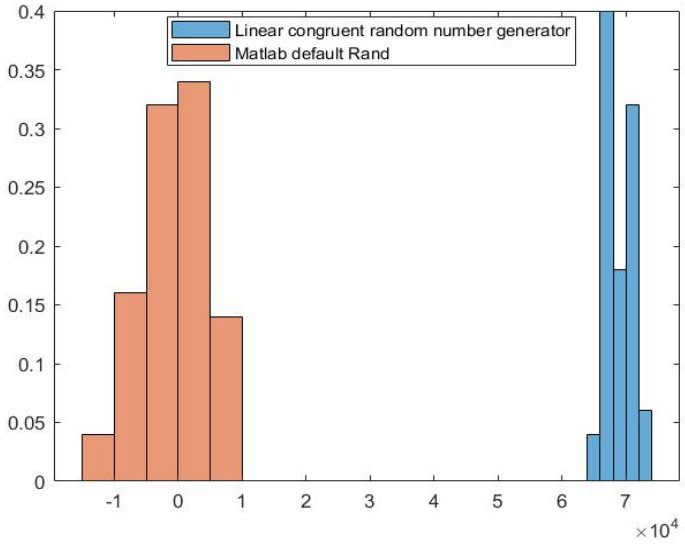

Figure 8: compare the histograms of covariance of $X = x_{2n-1}$ and $Y = x_{2n}$ for 50 different values of seed x_0 using Linear congruent random number generator and *rand* in $MATLAB$.

3 Problem 4

First I demonstrate how I find out the a_0, c_0 and x_0. In this problem, the method of rejection sampling is to using the Lorentzian distribution $f(x)$ function to generate random number that are distributed according to the Gamma distribution $p_n(x)$. To make the sampling sufficiently, we need to make sure that the Lorentzian distribution $f(x)$ function is just above the Gamma distribution $p_n(x)$ since if the $f(x)$ is too far from the $p_n(x)$, it is not sufficient in doing rejection sampling. To make sure that $f(x)$ is above $p_n(x)$, what I do is that first I choose a Cauchy probability (the alternative name of Lorentzian distribution) density function $\hat{f}(x)$, which means that the integral of $\hat{f}(x)$ is 1, then I find k such that the $k * \hat{f}(x) >= p_n(x)$, and

let $f(x) = k * \hat{f}(x)$. To find k, since $k >= p_n(x)/\hat{f}(x)$, which is equally to $k >= max(p_n(x)/\hat{f}(x))$, so I first using $MATLAB$ numerically calculate the $max(p_n(x)/\hat{f}(x))$ and then I prove it analytically that this k is sufficient for $f(x) >= p_n(x)$. Therefore we can find a $f(x)$ with appropriate a_0, c_0 and x_0, and we can easily get the normalized $f(x)$ by just divide k. To find appropriate a_0, c_0 and x_0, I use $MATLAB$ and plot $f(x)$, $\hat{f}(x)$ with different a_0, c_0 and x_0 and compare with $p_n(x)$, like Figure (9) where I find out a good $f(x)$ which have a similar form of $p_n(x)$ and above from $p_n(x)$ for all x. For example x_0 control the position of the $f(x)$, and a_0 control whether the Lorentzian Distribution is flat or steep around the highest values and I visualized them and choose the a_0, c_0 and x_0 which make the Lorentzian distribution have the similar shape of the Gamma distribution.

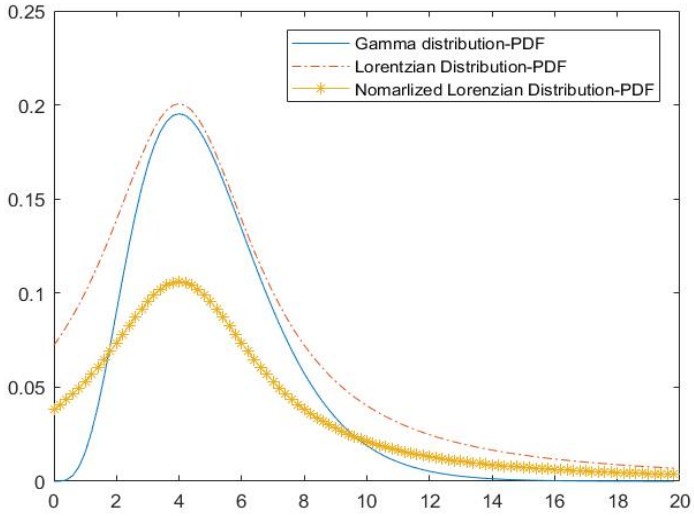

Figure 9: Visualization of different a_0, c_0 and x_0

3.1 Problem 4(a)

3.1.1 Problem 4(a) (i)

To aid in debugging your code as you develop it, plot $f(x)$ (appropriately normalized) together with a histogram of the generated auxiliary variables $\hat{x}$. From Figure (10) we can see that the $\hat{x}$ is definitely generated from $f(x)$ (appropriately normalized).

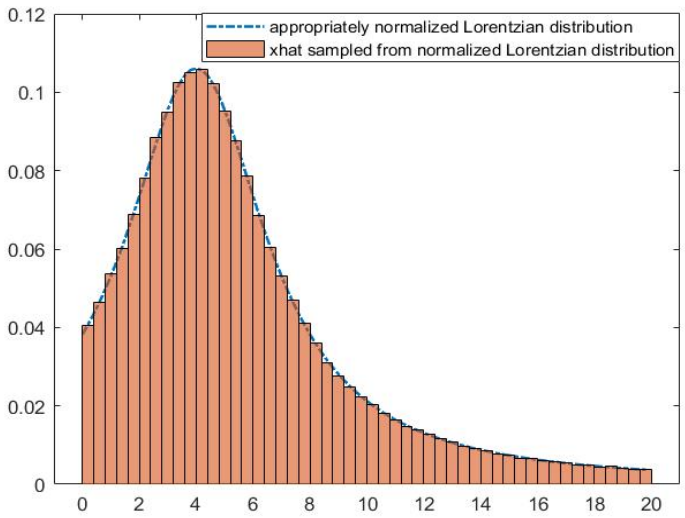

Figure 10: plot of $f(x)$(appropriately normalized) together with a histogram of the generated auxiliary variables $\hat{x}$ when $N = 10^6$ and 50 bins

3.1.2 Problem 4(a) (ii)

I calculate my acceptance rate of your rejection step by acceptance rate is equal to the number of acceptance sample from $\hat{x}$ divided by the number of $\hat{x}$, and the result of it is 0.5463. By repeating many times, my acceptance rate is in the range from 0.54 to 0.58.

3.1.3 Problem 4(a) (iii)

From Figure (11) we can see that the sample from Gamma distribution, which is blue histogram, correctly fit to the Gamma distribution $p_n(x)$, which is red dot line. The purple line is the Lorentz distribution density function $f(x)$ and it's above the Gamma distribution $p_n(x)$, which satisfies the requirement. Here $a_0 = 3$, $c_0 = 0.19537$ and $x_0 = 4$. I use $N = 10^6$ and 50 bins to generate the sample and histogram.

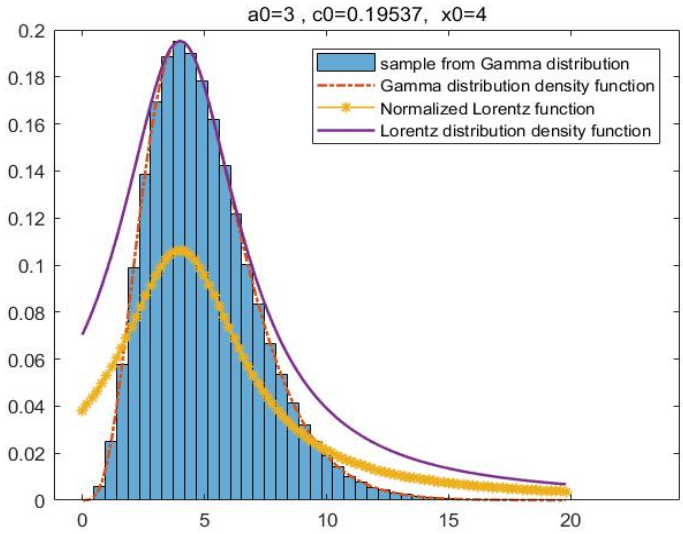

Figure 11: plot $pn(x)$ together with the histogram of y

3.1.4 Problem 4(a) (iv)

In $MATLAB$ I use vectorization method to generate $N = 10^6$ random number and the require time is 0.921 seconds, which is much more less than 10 seconds.

3.2 Problem 4(b)

3.2.1 Problem 4(a) (i)

Here the degrees of freedom in this problem is $k = 10$ since $nbins = 10$. For $\alpha = 0.05$, chi-square statistics should be larger than 18.31 to reject the null-hypothesis. For $\alpha = 0.01$, chi-square statistics should be larger than reject the null-hypothesis 23.21.

3.2.2 Problem 4(a) (ii)

For $N = 100$, $N = 1000$ and $N = 5000$ I perform different chi-square testing for $H(n_0)$ where $n_0 = 5$, $n_0 = 4.7$ and $n_0 = 5.3$ and I repeat for 100 times separately.

From Figure(12), where the $x - axis$ is chi-square statistics values and $y - axis$ is the frequency after 100 times chi-square testing, we can see that

for $H(n_0)$ where $n_0 = 4.7$, when $N = 5000$ I can reject reliably the null-hypotheses at the significance level 0.05.

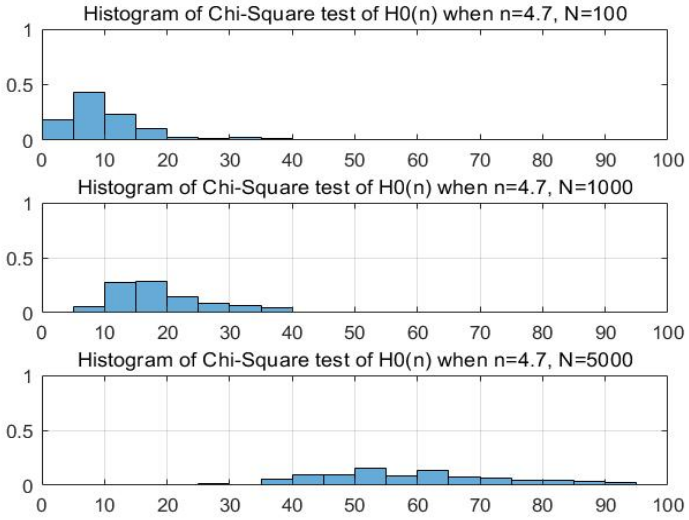

Figure 12: chi-square testing for $H(n_0)$ where $n_0 = 4.7$ and different N

From Figure(13),where the $x - axis$ is chi-square statistics values and $y - axis$ is the frequency after 100 times chi-square testing, we can see that for $H(n_0)$ where $n_0 = 5.3$, when $N = 5000$ I can reject reliably the null-hypotheses at the significance level 0.05.

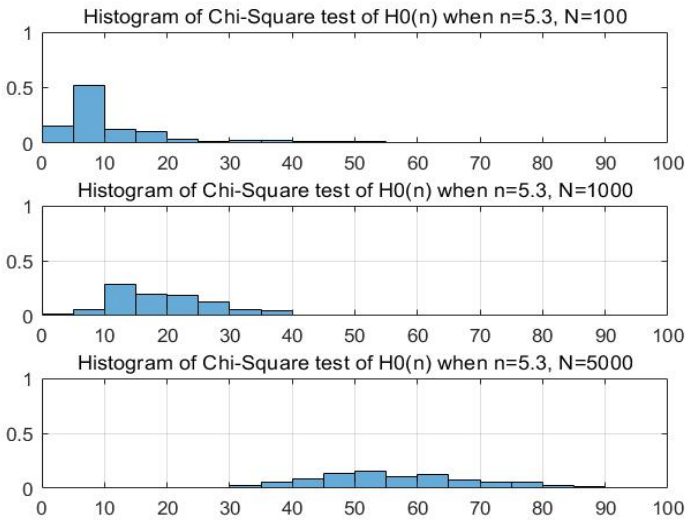

Figure 13: chi-square testing for $H(n_0)$ where $n_0 = 5.3$ and different N

Therefore for $H(n_0)$ where $n_0 = 5.3$ and $n_0 = 4.7$, when N is not big enough, we can not reject reliably the null-hypotheses at the significance level 0.05.

From Figure(14), where the $x - axis$ is chi-square statistics values and $y - axis$ is the frequency after 100 times chi-square testing, we can see that for $H(n_0)$ where $n_0 = 5$, no matter how big the N is, we can not reliably reject the null hypothesis, which means that the sample should be generated from Gamma function $p_n(x)$ with high probability.

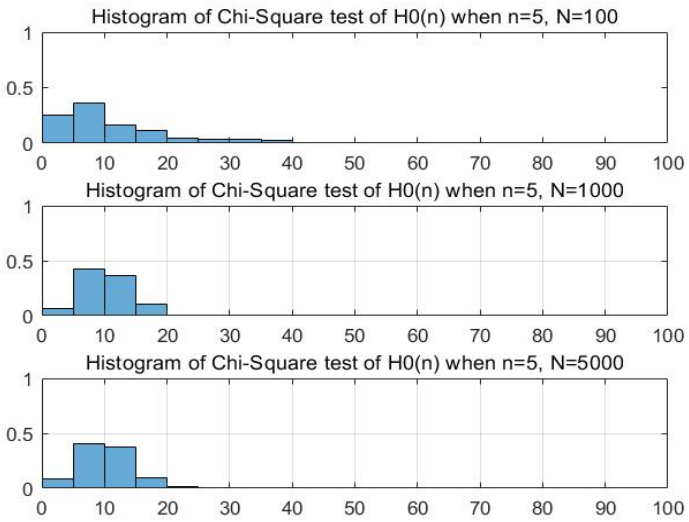

Figure 14: chi-square testing for $H(n_0)$ where $n_0 = 5$ and different N

Those experiments tell us that when we want to perform rejection sampling and want to do the chi-square testing, we should generate big enough samples since if the number of samples is not big enough, we may randomly reject or accept our null hypothesis, for example we may sometimes reject the null hypothesis though it's right or accept the null hypothesis though it's wrong.