Fall 2022: Monte Carlo Methods
Homework 1

NAME: Utkarsh Khandelwal
Net Id: uk2051

Exercise 13.
I wrote a subroutine that takes an input N to generate a sample point $\overline{x}_N$ by using the below expression.

$$\overline{x}_N = \sum_{i=1}^{N} \frac{x_i}{N}$$

Here, x_i is a random number generated from the exponenial distribution, $x_i \sim Exp(\lambda)$ with rate parameter $\lambda = 1$. Below is the code snippet of the same

```
def GenerateOneSamplePoint(N):
  generatedRands = np.random.exponential(size = N)
  sampleMean = np.mean(generatedRands)
  return sampleMean
```

Then this subroutine was used to produce M copies of $\overline{x}_N$ using code in the below snippet.

```
def GetMSamplePoints(M, N):
  currSamples = []
  for i in range(M):
    currSamples.append(GenerateOneSamplePoint(N))
  return np.array(currSamples)
```

These M copies were used to produce the Histogram of $Z_N = \sqrt{N}(\overline{x}_N$ - $\pi[x])$.
Here, $\pi[x]$ is the mean or expectation of x_i.

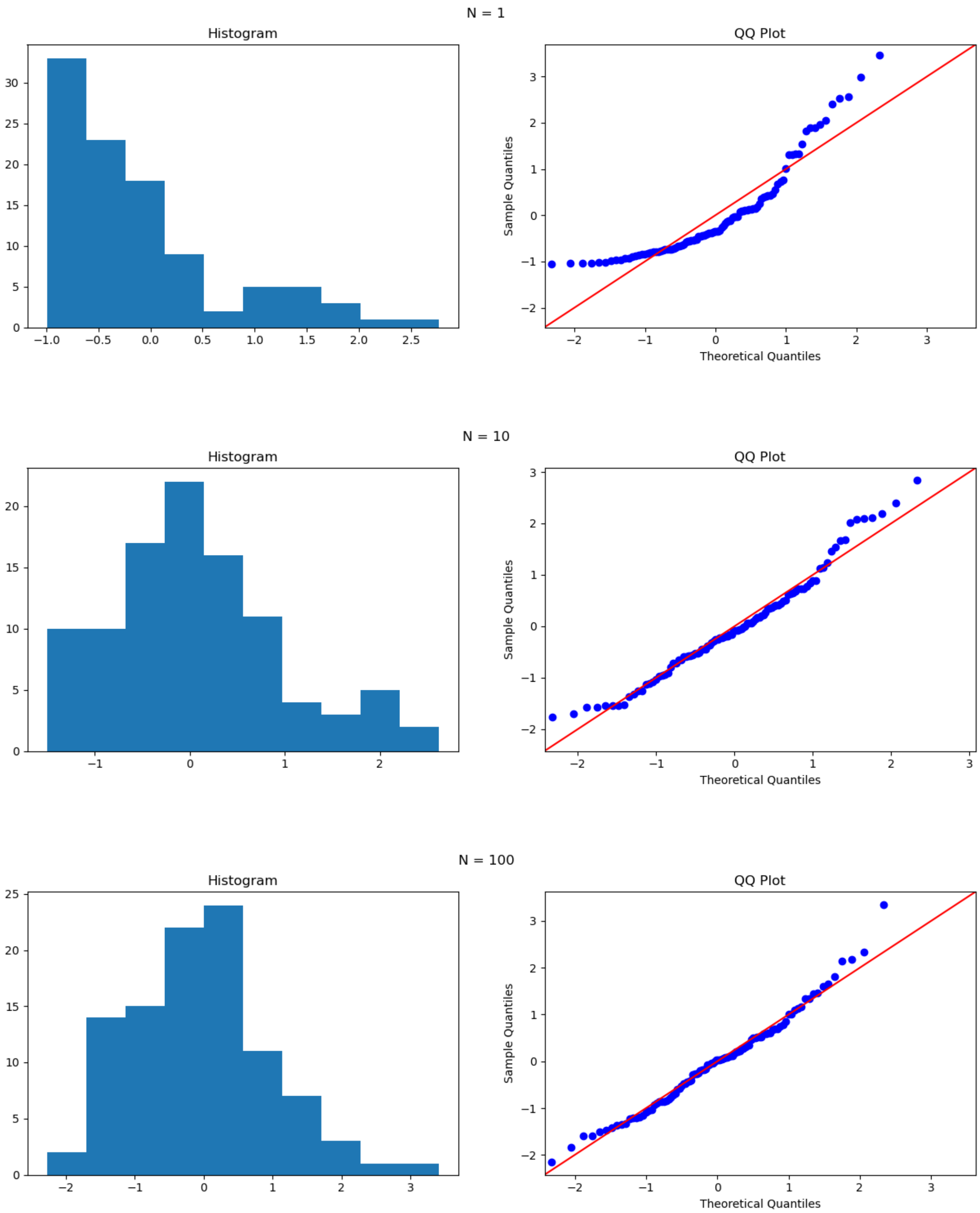
$$\pi[x] = E[x_i] = \frac{1}{\lambda} = 1$$

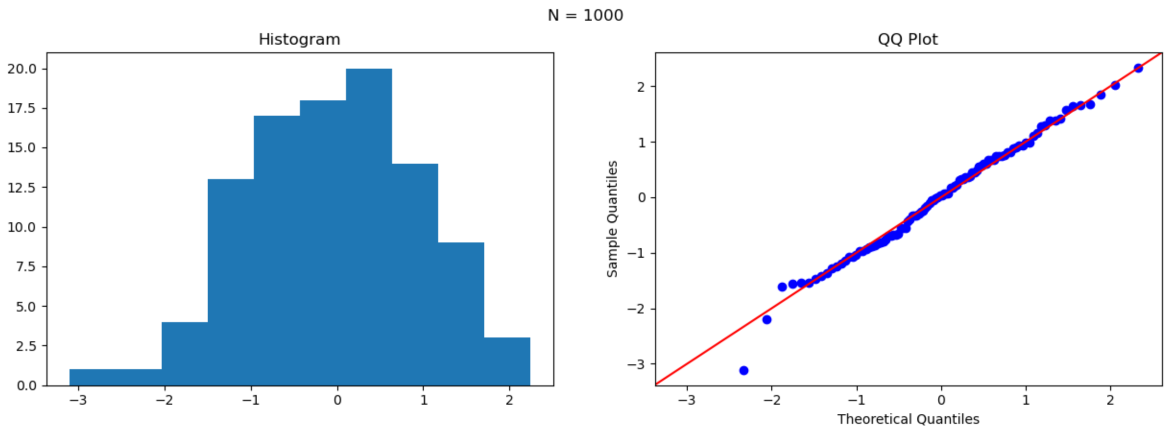
We know that from the Central Limit Theorem (CLT) that if $X_1, X_2, X_3, X_4, \ldots, X_N$ are sequence of iid then Y_N is the Gaussian Distribution $N(0,1)$ when N tends to ∞. Here, $Y_N = \frac{\overline{x}_N - \pi[x]}{\sigma/\sqrt{N}}$

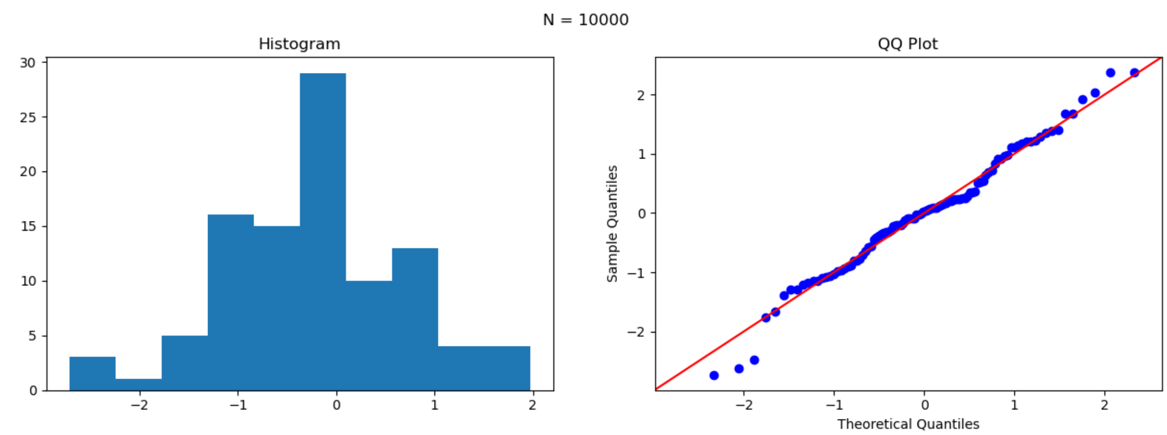
$$\lim_{N\to\infty} Y_N = \lim_{N\to\infty} \frac{\overline{x}_N - \pi[x]}{\sigma/\sqrt{N}} = \lim_{N\to\infty} \frac{\sqrt{N}\,(\overline{x}_N - \pi[x])}{\sigma} \sim N(0,1)$$

$$Y_N = \frac{Z_N}{\sigma}$$

Therefore distribution of Z_N is $N(0, \sigma^2)$. This was checked numerically by simulating these probabilities and plotting the histograms and QQ Plots. I observed that as N increases the histogram will be resembling the shape closer to Gaussian density shape and the QQ Plots will have points closer to $y = x$ line. Below are the QQPlots and Histograms for the for various value of N.


N = 1
Histogram
QQ Plot
Sample Quantiles
Theoretical Quantiles
N = 10
Histogram
QQ Plot
Sample Quantiles
Theoretical Quantiles
N = 100
Histogram
QQ Plot
Sample Quantiles
Theoretical Quantiles





Then I developed a simulator for computing probabilities. I estimated Q_N, the probability $p_N = P[\overline{x}_N - 1 > 0.1]$ using simulation. This expression can be simplified as $p_N = P[\overline{x}_N > 1.1]$.
The algorithm used to compute this probability is simple. It is using the expectation of the indicator random variable. Let A be an event when $\overline{x}_N > 1.1$.
Therefore

$$P[\overline{x}_N - 1 > 0.1] = P[\overline{x}_N > 1.1] = E[\mathbb{1}_A]$$

I computed this expectation Q_N using below mentioned expression. So,

$$Q_N = E[\mathbb{1}_A] = \sum_{i=1}^{M} \frac{\mathbb{1}_A}{M}$$

The logic follows that create M samples of $\overline{x}_N$ and count all of those that are greater than 1.1 and then divide by total number of generated samples M. Just to make computation faster, a sample point $\overline{x}_N$ was generated using the Gamma Distribution. Since,

$$\overline{x}_N = \sum_{i=1}^{N} \frac{x_i}{N}$$

where each x_i is exponentially distributed. As Exponential Distribution $Exp(\lambda)$ can also be weritten as $Gamma(1, \lambda)$. And sum of N iid Gamma Distributed variables is an $Gamma(N, \lambda)$. So

$$\overline{x}_N = \frac{Y_N}{N}$$

Here, Y_N is sampled from $Gamma(N, \lambda)$. Below is the routine that computes this estimate Q_N of p_N

```
def GenerateOneSamplePointFromGammaSum(N):
  return (np.random.gamma(N,1)/N)

def GetManySamplePoints(M, N):
  currSamples = []
  for i in range(M):
    currSamples.append(GenerateOneSamplePointFromGammaSum(N))
  df = pd.DataFrame(currSamples)
  return df

def ComputeGivenProbability(M,N):
  pointsFrame = GetManySamplePoints(M, N)
  col = pointsFrame[0]
  greaterCount = col[col > 1.1].count()
  probab = greaterCount/M
  return probab
```

Next step is finding the decay rate of

$$D = \frac{1}{N} P[\overline{x}_N - 1 > 0.1]$$

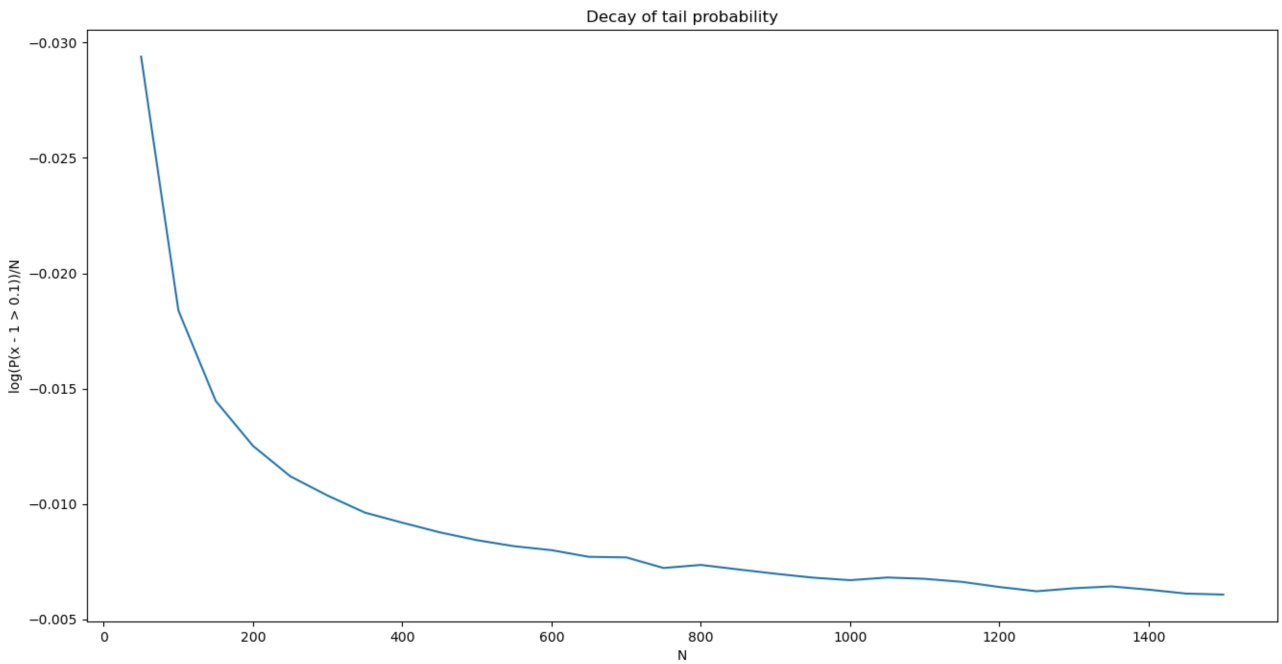
both theoretically and experimentally. It has been showed in Example 3, that this rate of decay of

$$\lim_{N \to \infty} \frac{1}{N} P[\overline{x}_N - 1 > \epsilon]$$

is $-\frac{\epsilon^2}{2}$ using Central Limit Theorem (CLT) and $-\epsilon + \log(1 + \epsilon)$ using the Large Deviations Principle (LDP). So, in our case $\epsilon = 0.1$ and therefore theoretical rate of convergence using CLT is -0.005 (which is flawed) and using LDP is -0.0046898. And using simulation I am getting:

N	D
300	-0.0103317948075986
600	-0.00792665512620446
900	-0.006997766341512492
1200	-0.006541136515700485
1500	-0.006076686794781239

Below is the image that captures the rate of decay as well



Last part of the exercise requires me to compare the decay of standard deviation of Q_N with the decay of probability p_N. Let us analytically calculate the relationship between the standard deviation of Q_N and p_N.

$$Var(Q_N) = E[Q_N^2] - (E[Q_N])^2$$

$$E[Q_N] = E\left[\sum_{i=1}^{M} \frac{\mathbb{1}_A}{M}\right]$$

Using linearity of expectation we can write

$$E[Q_N] = E\left[\sum_{i=1}^{M} \frac{\mathbb{1}_A}{M}\right] = \sum_{i=1}^{M} \frac{E[\mathbb{1}_A]}{M}$$

And since it is a indicator random variable, we know that $E[\mathbb{1}_A] = p_N$ Therefore,

$$E[Q_N] = \sum_{i=1}^{M} \frac{p_N}{M} = p_N$$

Now, computing the second moment

$$E[Q_N^2] = E\left[\left(\sum_{i=1}^{M} \frac{\mathbb{1}_A}{M}\right)^2\right]$$

Now, since these are iid so the covariance between different random variable would be zero, and linearity of expectation, this can be simplified to

$$E[Q_N^2] = E\left[\sum_{i=1}^{M}\left(\frac{\mathbb{1}_A}{M}\right)^2\right] = E\left[\sum_{i=1}^{M}\frac{\mathbb{1}_A}{M^2}\right] = \sum_{i=1}^{M}\frac{E[\mathbb{1}_A]}{M^2} = \frac{Mp_N}{M^2} = \frac{p_N}{M}$$

Hence, variance can be written as:

$$Var(Q_N) = E[Q_N^2] - (E[Q_N])^2 = \frac{p_N}{M} - (p_N)^2 = p_N^2\left(\frac{1}{p_N M} - 1\right)$$

So, the standard deviation would be:

$$sd(Q_N) = \sqrt{var(Q_N)} = \sqrt{p_N^2\left(\frac{1}{p_N M} - 1\right)} = p_N\sqrt{\frac{1}{p_N M} - 1}$$

Here, M are the count of samples generated for estimating Q_N which I have considered constant while varying N.

Rate of decay of $sd(Q_N)$ could be compared to the p_N by the following expression:

$$\frac{sd(Q_N)}{p_N} = \sqrt{\frac{1}{p_N * M} - 1}$$

Now I present a mathematical argument for comparing the decay rates. From the above attached graph it is evident that p_N is decreasing with the increase in the value of N and therefore $\frac{1}{p_N}$ will keep on increasing implying that the whole expression $\sqrt{\frac{1}{p_N * M} - 1}$ will increase with the value of N. So the rate of decay of standard deviation of Q_N is definately more than the rate of decay of p_N.