EE 511 Homework 3

Yijing Jiang
yijingji@usc.edu

Feburary 8, 2020

1 Question 1

1.1 Part a

Explanation:
Simulate the probability of rejection. As we can know from the question, there are 6 defective units in 125 microchips. Choose 5 from 125 microchips without replacement. If any defective unit is chosen, then all will be rejected. First, generate a random sequence in the range of [1,125), get the first six numbers. Then create an empty array with value 0, set six elements whose index are those six numbers as 1. Second, generate a random sequence in range [1,125), and get the first five numbers as the simulation of randomly choosing five to test. Find if there is any defective units in these five elements. If exists, add the number of rejecction. Third, repeat the second step for 10000 times, calculate the rejected events, and conpute the rejection rate.

Code:

```
import numpy as np
import random

# defective components' number
defective_random = np.random.permutation(125)
defective_comp = np.array(defective_random[0:6])
#sequence of 125 components with 6 defective
components = np.empty((125,))
for i in defective_comp:
    components[defective_comp] = 1
#sample 5 components
reject = 0
for j in range(10000):
    sample_random = np.random.permutation(125)
    sample_comp = np.array(sample_random[0:5])
    #check if it has defective components
    defective = 0
    for n in sample_comp:
        if components[n] == 1:
```

```
            defective += 1
    #record reject cases
    if defective != 0:
        reject += 1
rate = reject/10000
print(rate)
```

Results:
By simulating, when distributor chooses five components to test, then the probability of rejection is 0.2281.
Theoretically, this is a combination problem. The probability of rejecting is equal to one minus the probability of accepting. Accepting means all of five units are good. So the theoretical probability of rejection is

$$\begin{aligned} P(rejction) &= \frac{total - accept}{total} && (1) \\ &= \frac{\binom{125}{5} - \binom{125-6}{5}}{\binom{125}{5}} && (2) \\ &= 0.221266874 && (3) \end{aligned}$$

The simulation result is close to the theoretical one, which can verify that our simulation is correct.

1.2 Part b

Explanation:
Find the fewest number of testing samples, which reaches 95% rate of rejection. Start from sample equals to five, and add one each time when the rate of rejection in 10000 times is less than 0.95. Once the rate is larger or equal to 0.95, stop the loop. The current sample is just the fewest number.

Code:

```
import numpy as np
import random

#6 defective components' number
defective_random = np.random.permutation(125)
defective_comp = np.array(defective_random[0:6])
#sequence of 125 components with 6 defective
components = np.empty((125,))
for i in defective_comp:
    components[defective_comp] = 1
#sample 5 components
sample = 5
rate = 0
while rate < 0.95:
    sample += 1
```

```
        reject = 0
        for j in range(10000):
            sample_random = np.random.permutation(125)
            sample_comp = np.array(sample_random[0:sample])
            #check if it has defective components
            defective = 0
            for n in sample_comp:
                if components[n] == 1:
                    defective += 1
            #record reject cases
            if defective != 0:
                reject += 1
        rate = reject/10000
        print(rate)
    print(sample)
```

Results:

The fewest number of testing is 49. It means if the ditributor test 49 or more units each time, then the probability of get the defective units is 95%.

2 Question 2

In this question, we use two algorithems to simulate the poisson distribution. One is simulated from a real world situation, the other is simulated by inverse transfrom method.

2.1 Part a

Explanation:
Poisson distribution can be converged by a binomial distribution with extremlly large N and extremlly small p. The parameter λ will be equal to N times p.
Here we have the problem of waiting cars on freeway. Set the length of the interval as 0.1 second, compute the probability of occurance of one car in this small interval:

$$\lambda = 120 cars/hour \tag{4}$$
$$N = 36000 \tag{5}$$
$$p = \frac{\lambda}{N} \tag{6}$$

Every samll interval can be seen as a bernulli distribution with probability p. Generate a random value μ uniformlly in range [0,1]. If μ is less than p, one car comes. Otherwise, no car comes. Do above steps in 36000 intervals (= one hour), count the number of cars appear. Repeat the experement for 50000 times.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chisquare
from scipy.stats import chi2

#p is the prob of a car in 0.1s
lmbda = 120
N = 36000 #0.1s: 1*60*60*10
p = lmbda/N

#repeat trials times
trials = 50000
results = np.empty((trials,))

#actual number of cars in trials times
for i in range (trials):
    u = np.random.rand(N,1)
    bernoulliTrials = u<p #true(u<p) or false(u>p)
    x = np.sum(bernoulliTrials) # sum the boolean value is true
    results[i] = x

mean = np.mean(results)
var = np.var(results)
print(mean, var)
```

```
#see the probability in 0 to 250 cars
bins = np.empty((152,))
for j in range (152):
    bins[j] = j+50
prob = plt.hist(results, bins, align = "left", edgecolor = "black",
    density = True)

#show theoretical distribution of pdf
theoretic = np.empty((200,))
for i in range (200):
    if i == 0:
        p = np.exp(-lmbda)
        theoretic[i] = p
    else:
        p = p * lmbda / i
        theoretic[i] = p
part = np.array(theoretic[50:201]) # theoretical value = [50,51,...,200]
plt.plot(bins[0:150], part, 'r')
plt.grid()

#show if this fits the poisson distribution
chisq, p_value = chisquare(prob[0][0:150], part)
ci = chi2.ppf(0.95, 150)
print(chisq,ci)
if chisq <= ci :
    print("These data fits ")
else:
    print("These data dose not fit ")

plt.show()
```

Results:

- Histogram:
  - The x-axis of the histogram is the number of arrivals per hour, the y-axis is the probability of the number of arrivals per hour in 50000 experements. The peak of the probability is around 120, and slowly decrease to both side. Also roughly, the probability of less than 80 arrivals or greater than 160 arrivals is almost 0.
  - Futhermore, compute the exact p.m.f of Poisson distribution with $\lambda = 120$ on the range of [50,200]. Then plot these points and connect them in red line. It shows that this line envelops the histogram well, in other words, the distribution of this actual problem can be seen as a Poisson distribution with $\lambda = 120$.

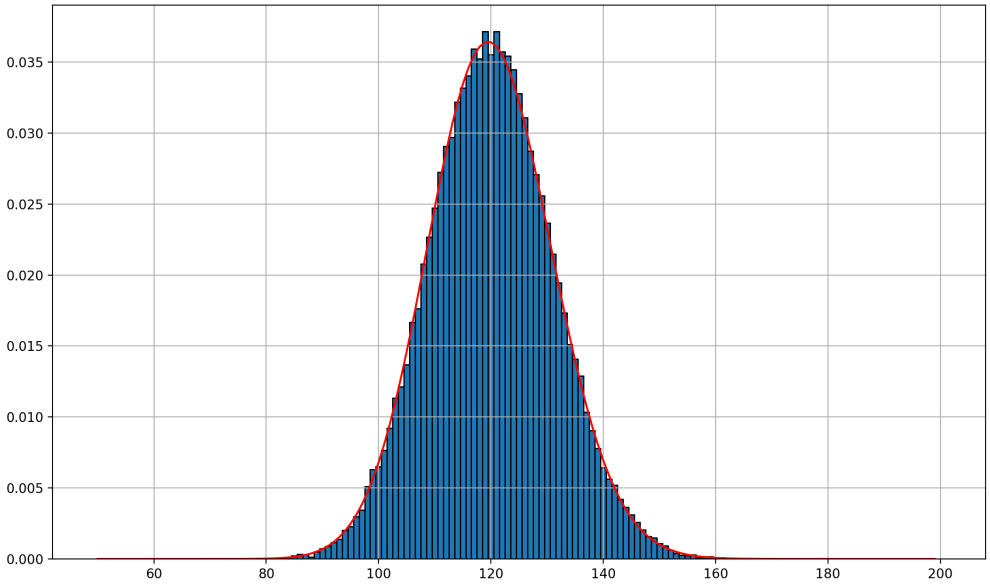

Figure 1: simulation of number of arrivals per hour($\lambda = 120$, experements = 50000)

- Sample mean and sample variance:

  - For 50000 trials, get the sample mean and sample variance as bollow:

$$SampleMean = 119.98 \tag{7}$$
$$SampleVariance = 120.15 \tag{8}$$

  - This experement is a simulation of Binomial distribution. For Binomial distribution B $\sim$ (N,p), have the theoretical mean and variance as bellow:

$$E_b = N * p = 120 \tag{9}$$
$$Var_b = N * p * (1 - p) = 119.6 \tag{10}$$

    We can find that the sample mean and sample variance are both close to the theoretical mean and variance of the Binomial distribution. It shows that the simulation is correct.

  - For a Poisson distribution, its mean and variance are equal:

$$E_p = Var_p = \lambda = 120 \tag{11}$$

    In the actual situation of arriving cars, the sample mean (7) and sample variance (8) of the simulation are really close to each other, thus the distribution is also similar to Poisson distribution. This also confirms the property that a Poisson distribution can be seen as a huge sequence of Binomial distribution with extremlly small p.

- Goodness of fit:

  - To show whether the simulation can be seen as a sample from Poisson distribution, compute the chisquare value between the histogram and the theoretical p.m.f of Poisson distribution. Then, figure out whether it is in the 95% confidence level. Get the following results:

$$ChisquareTest = 0.0023 \tag{12}$$
$$95\%ofChisquare = 179.58 \tag{13}$$
$$\tag{14}$$

    Thus,

$$ChisquareTest < 95\%ofChisquare \tag{15}$$

    we can come to the conclusion that this simulation actually fits the Poisson distribution.

2.2 Part b

Explanation:
For Poisson distribution, it's p.d.f is:

$$P(X = k) = \frac{e^{-\lambda}\lambda^k}{k!} \tag{16}$$

where

$$P(X = k + 1) = \frac{\lambda}{k+1}P(X = k) \tag{17}$$

In inverse transform method, a random μ in range [0,1] will be counted to X=k if it satiesfies:

$$F_k \leq \mu < F_{k+1} \tag{18}$$

To simulating Poisson distribution, first generate μ . Then start from k = 1. If μ is not in the current interval, then move to next interval by doing these two operations:

$$F_{k+1} = F_k + P_{k+1} \tag{19}$$
$$k = k + 1 \tag{20}$$

Finally we can get the interval of μ, record this interval. Repeat this process for 50000 times, we can get a histogram of Poisson distribution.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chisquare
from scipy.stats import chi2

lmbda = 120

#repeat times
trials = 50000
results = np.empty((trials,))

#sample from theoretical pdf by inverse trasnform method
for i in range(trials):
    u = np.random.random_sample()
    p = np.exp(-lmbda)
    F = p
    x = 0
    while u >= F:
        p = lmbda/(x+1)*p
        F = F + p
        x += 1
    results[i] = x

mean = np.mean(results)
var = np.var(results)
print(mean, var)

#see the probability in 50 to 200 cars
bins = np.empty((152,))
for j in range (152): # j = [0,1,...,151]
    bins[j] = j+50 # bins = [50,51,....201]
prob = plt.hist(results, bins, align = "left", edgecolor = "black",
    density = True)

#show theoretical distribution of pdf
theoretic = np.empty((200,))
for i in range (200):
    if i == 0:
        p = np.exp(-lmbda)
        theoretic[i] = p
    else:
        p = p * lmbda / i
        theoretic[i] = p
part = np.array(theoretic[50:201]) # theoretical value = [50,51,...,200]
plt.plot(bins[0:150], part, 'r')
plt.grid()

#show if this fits the poisson distribution
```

```
chisq, p_value = chisquare(prob[0][0:150], part)
ci = chi2.ppf(0.95, 150)
print(chisq,ci)
if chisq <= ci :
    print("These data fits ")
else:
    print("These data dose not fit ")

plt.show()
```

Results:

- Histogram:

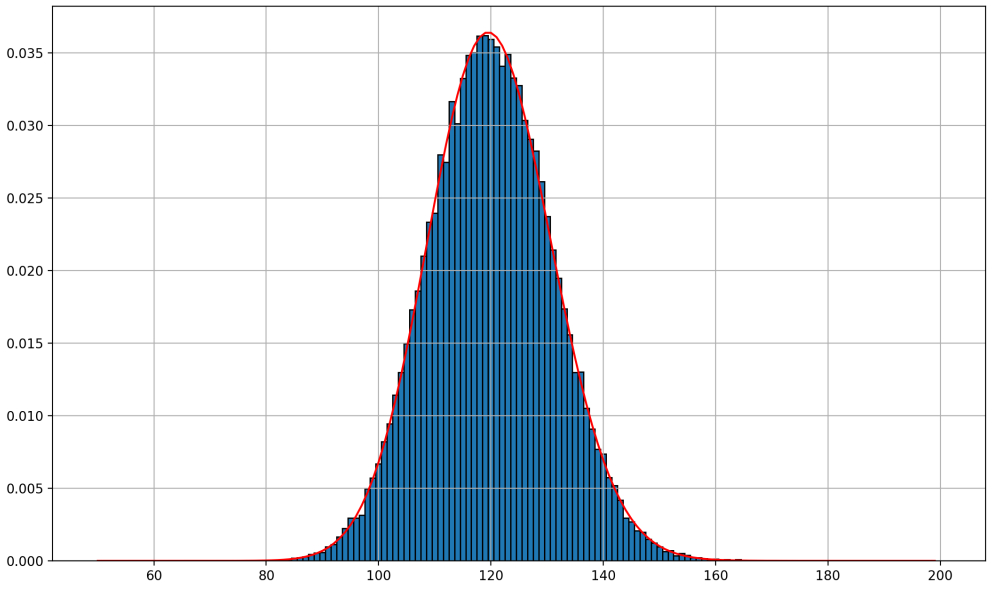

Figure 2: sampling from Poisoon distribuion by inverse transform method($\lambda = 120$, experements = 50000)

  - The x-axis of the histogram is the index k, the y-axis is the probability of the appearance of index k in 50000 experements. Here, we also generate a theoretical p.m.f of Poisson distribution (red line). It shows that this line also envelops the histogram well. It shows that the inverse transform method works here.

- Sample mean and sample variance:

- For 50000 trials, get the sample mean and sample variance as bollow:

$$SampleMean = 119.91 \tag{21}$$
$$SampleVariance = 120.36 \tag{22}$$

- For Poisson distribution here, its mean and variance are:

$$E_p = Var_p = \lambda = 120 \tag{23}$$

The sample values are both close to the theoretical values.

- Goodness of fit:

  - Compute the chisquare value between the histogram and the theoretical p.m.f of Poisson distribution. Get the following results:

$$ChisquareTest = 0.0019 \tag{24}$$
$$95\%ofChisquare = 179.58 \tag{25}$$
$$\tag{26}$$

Thus,

$$ChisquareTest < 95\%ofChisquare \tag{27}$$

we can come to the conclusion that simulating by inverse transform method fits the Poisson distribution.

3 Question 3

Explanation:
To get each N, uniformlly generate random μ in range [0,1] one by one, and record their sum. Once the sum is greater than 4, stop generating. The current numbers of μ is just the minimum index N. Repeat steps above for 100 or 1000 or 10000 times, we can get many N. Then put these values into a histogram.

Code:

```
import numpy as np
import matplotlib.pyplot as plt

for i in range(1,4):
    # 100 or 1000 or 10000 times
    N = 10**(i+1)
    N_samples = np.empty((N,))
    for n in range (N):
        xi = 0 # record index
        sum = 0 # record sum
        # stop when sum > 4
        while(sum <= 4):
            # generate random u in [0,1]
            u = np.random.random_sample()
            sum = sum + u
            xi = xi + 1
        N_samples[n] = xi
    # compute mean
    mean = np.mean(N_samples)
    # show the histogram
    plt.subplot(1,3,i)
    bins = np.array(range(25))
    plt.hist(N_samples, bins, align = "left", edgecolor = "black")
    plt.title("trials = "+str(N)+", E[N] = "+str(mean))
plt.show()
```

Results:

- Histogram and sample mean:

  - The x-axis of the histogram is the minimum number of samples, the y-axis is the counting number of the minimum sample number. We can see that for different sample times on N, the distributions are similar. The minimum number of samples whose sum will be greater than 4 is mostly around 8. And it fits the results of sample mean, which is 8.64, 8.717, 8.6628 for 100, 1000, 10000 trials.
  - We choose the expected value of 10000 trials as the final expected value of the

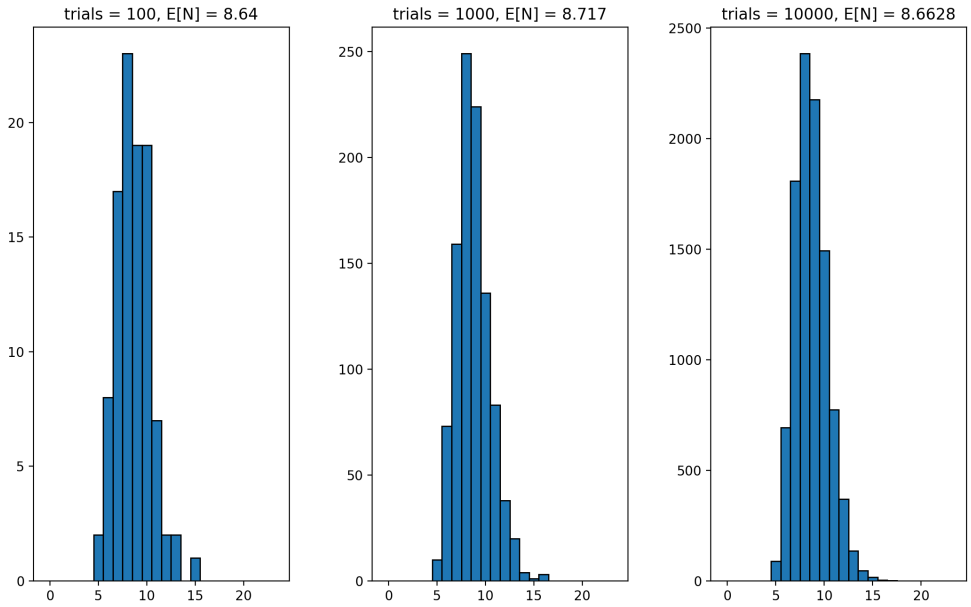

Figure 3: Histograms of sampling 100, 1000, 10000 times for N

minimum number whose sum will be greater than 4:

$$E[N_{10000}] = 8.6628 \tag{28}$$

- Uniform sum distribution(Irwin–Hall distribution):

  – Sum of a number of independent random variables with uniform distribution is called uniform sum distribution or Irwin–Hall distribution. For the sum of n numbers, the variable is:

$$X = \sum_{k=1}^{n} U_k \tag{29}$$

$$U_k \sim U(0,1) \tag{30}$$

The p.d.f is given by:

$$f_X(x;n) = \frac{1}{2(n-1)!}\sum_{k=0}^{n}(-1)^k\binom{n}{k}(x-k)^{n-1} \tag{31}$$

In this distribution, the expected value of picks when the sum is first to the interger s is given by:

$$E[N_s] = \frac{1}{n!}\sum_{k=0}^{n}\frac{-1^k n!(n-k+1)^k}{k!}e^{n-k+1}, (n=s-1) \tag{32}$$

For some s and $E[N_s]$:

s	$E[N_s]$	approximate
1	e	2.718
2	$e^2 - 1$	4.670
3	$\frac{1}{2}(2e^3 - 4e^2 + e)$	6.666
4	$\frac{1}{6}(6e^4 - 18e^3 + 12e^2 - e)$	8.666
5	$\frac{1}{24}(24e^5 - 96e^4 + 108e^3 - 32e^2 + e)$	10.666

– Here, for s = 4, the result we get of 8.6628 is similar to the approximate theoretical value of 8.666.

– Also try on s = 5, the sample mean of experement also fits the approximate theoretical value of 10.666:

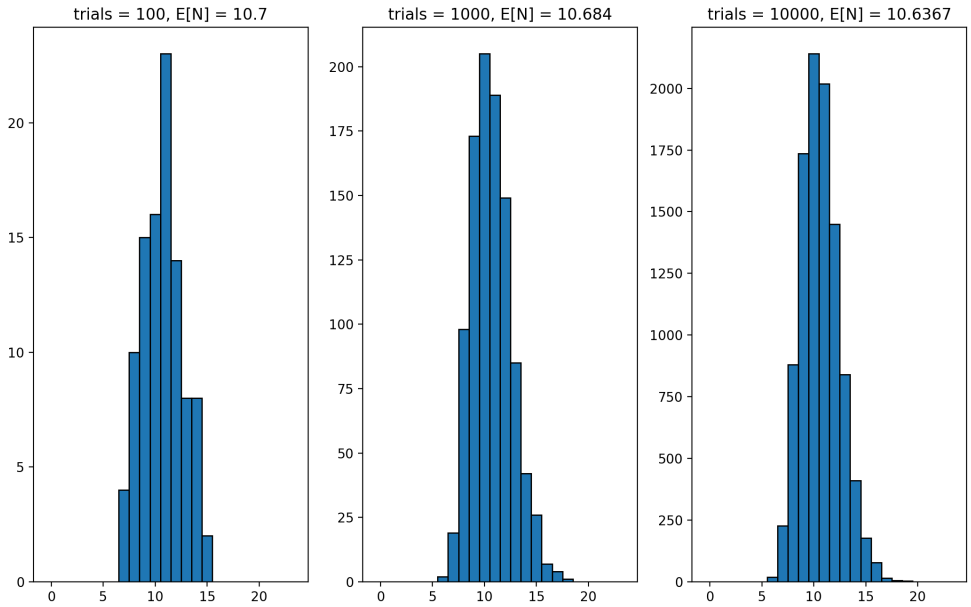

Figure 4: Histograms of the minimum numbers that first reaches s = 5

4 Question 4

Explanation:
Simulate the distribution of the minimum number that first stop at minute 60. First, find the constant p. For the sum of $p_j = \frac{p}{j}$, $(j = 1, 2, ..., 60)$ must equal to 1:

$$\sum_{j=1}^{60} p_j = 1 \tag{33}$$

$$\tag{34}$$

So the constant p is:

$$p = \frac{1}{\sum_{j=1}^{60} \frac{1}{j}} \tag{35}$$

$$= 0.21368113 \tag{36}$$

the probability of stop on minute 60 is:

$$p_{60} = \frac{p}{60} \tag{37}$$

$$= 0.003561352 \tag{38}$$

Second, generate random varaibles μ in range [0,1] uniformly. If μ is larger than p_{60}, regard as not stop on minute 60 and keep on sampling. Otherwise, regard as reaching minute 60, stop sampling and record the current number of samples as the minimum number N_{60} in this one experement. Repeat this experement for 10000 trials.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chisquare
from scipy.stats import chi2

sum_j = 0
for j in range(1,61):
    sum_j += 1/j
p = 1/sum_j
p_60 = p/60

#get the samplings of N60
N = 10000
X = np.empty((N,))
for i in range(N):
    u = 1
    k = 0
    #u>p_60: not stop at 60
    while(u > p_60):
        u = np.random.sample()
```

```
            k = k + 1
        X[i] = k

#compute the sample mean and sample value
mean = np.mean(X)
var = np.var(X)
print(mean, var)

#show the histogram
prob = plt.hist(X, bins = 2500, edgecolor = "black", density = True)

#theoretical line
theoretic = np.empty((2500,))
p = p_60
for i in range (2500):
    p = p_60 * ((1-p_60)**i)
    theoretic[i] = p
bins = range(1, 2501)
plt.plot(bins, theoretic, 'r')

#show if this fits the geometric distribution
chisq, p_value = chisquare(prob[0][0:2500], theoretic)
ci = chi2.ppf(0.95, 2499)
print(chisq,ci)
if chisq <= ci :
    print("These data fits ")
else:
    print("These data dose not fit ")

plt.show()
```

Results:

- Histogram, sample mean and sample variance:
  - The x-axis of the histogram is the minimum number of first stop at minute 60, the y-axis is the probability of the minimum number. We can see that the probability of minimum number is gradually decreasing from the beginning.
  - The results of sample mean and sample varianve:

$$E[N_{60}]_{sample} = 279.526 \tag{39}$$
$$Var[N_{60}]_{sample} = 77695.533324 \tag{40}$$

- Theoretical line, mean and variance:
  - This question can be seen as geometric distribution with probability p_{60}. N_{60} is the first one that stop at minute 60, so all the events before are not at minute

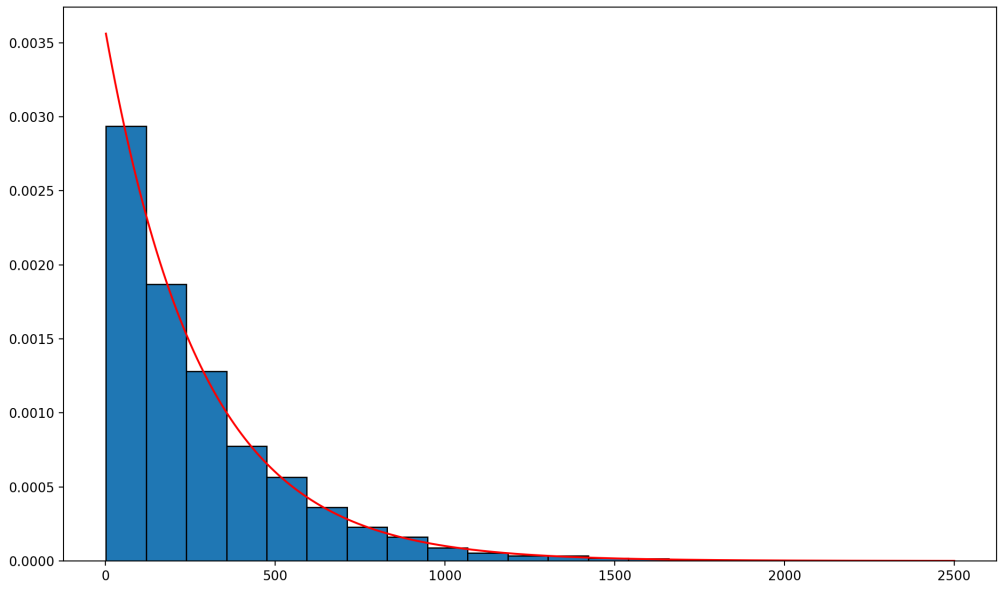

Figure 5: histogram of sampling from N_{60} for 10000 times

60. So the theoretical p.m.f of this distribution is:

$$P[N_{60} = k] = (1 - p_{60})^{k-1} p_{60} \tag{41}$$

with the mean and the variance:

$$E[N_{60}]_{theoretic} = \frac{1}{p_{60}} = 280.7922385 \tag{42}$$

$$Var[N_{60}]_{theoretic} = \frac{1 - p_{60}}{{p_{60}}^2} = 78563.48894 \tag{43}$$

The estimated mean (39) and variance (40) are both really close to the theoretical mean and variance.

- Goodness of fit:

  – Compute the chisquare value between the histogram and the theoretical p.m.f of Geometric distribution. Get the following results:

$$ChisquareTest = 0.23 \tag{44}$$

$$95\%ofChisquare = 2616.41 \tag{45}$$

$$\tag{46}$$

Thus,

$$ChisquareTest < 95\%ofChisquare \tag{47}$$

we can come to the conclusion that the sampling on N_{60} fits the Geometric distribution.

5 Question 5

Explanation:
Use accept-reject method to sample from p_j base on q_j. First, choose a constant c:

$$c = \frac{max(p_j)}{max(q_j)} \tag{48}$$
$$= \frac{0.15}{0.05} \tag{49}$$
$$= 3 \tag{50}$$

Second, uniformly generate a random varaible μ from [0,1] and a random integer j from [1,10]. If $\mu \leq \frac{p_j}{cq_j}$, then this μ is under the point (j,p_j), so we accept it, record the number j and record the number of being accepted. Otherwise, μ is higher than the point(j,p_j), so we reject it and count the number of being rejected. Repeat sampling for 10000 trials.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
#create p, q, c, and the ratio of p/(c*q)
p =
    np.array([0.06,0.06,0.06,0.06,0.06,0.15,0.13,0.14,0.15,0.13,0,0,0,0,0,0,0,0,0,0])
q = np.full((20,),0.05)
c = 4
ratio = np.empty((20,))
for r in range (20):
    ratio[r] = p[r] / (c * q[r])

sequence = np.empty((10000,))
accept = 0
reject = 0
#10000 trials
for i in range(10000):
    #generate a point (j,u)
    u = np.random.random_sample()
    j = np.random.randint(20)
    # if u < ratio, point is lower than p, accept
    if u <= ratio[j]:
        sequence[accept] = j + 1
        accept += 1
    #else, point is higher than p,reject
    else:
        reject += 1

#bins of histogram
bins = np.array(range(1,22))
plt.hist(sequence[0:accept], bins, align = 'left', edgecolor = "black",
    density = True)
```

```
#bins of theoretical distribution
bins = np.array(range(1,21))
plt.plot(bins, p, 'r', marker = '.')
plt.show()

mean = np.mean(sequence[0:accept])
var = np.var(sequence[0:accept])
print(mean, var)
print(accept)
print(reject)
```

Results:

- Histogram, sample mean, sample variance:

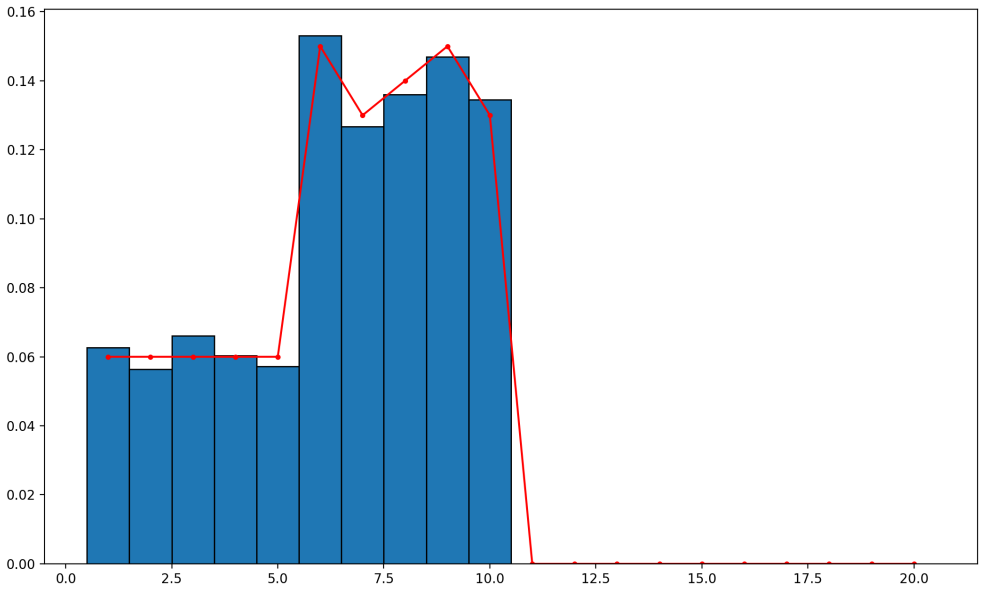

Figure 6: histogram of sampling 10000 times on p_j and the target distribution of p_j(red line)

  - The x-axis of the histogram is the index of p_j and q_j from 1 to 20, the y-axis is the probability of each index. We can see that the distribution of histogram is roughly follow the target distribution line.
  - The results of sample mean and sample varianve:

$$E[p]_{sample} = 6.4627 \tag{51}$$
$$Var[p]_{sample} = 7.2952 \tag{52}$$

– The theoretical mean and variance:

$$E[p]_{theoretic} = \sum_{j=1}^{10} j * p_j \tag{53}$$
$$= 6.48 \tag{54}$$
$$Var[p]_{theoretic} = \sum_{j=1}^{10} p_j (j - E[p]_{theoretic})^2 \tag{55}$$
$$= 7.1896 \tag{56}$$

Sample valus are both close to the theoretical valus.

- Efficiency:

– The estimation of efficiency after sampling 10000 times with c = 3 is:

$$Efficiency_{sample}(c = 3) = 0.33 \tag{57}$$

The theoretical efficiency when c = 3 is:

$$Efficiency_{theoretic}(c = 3) = \frac{\sum_{j=1}^{20} p_j}{\sum_{j=1}^{20} c * q_j} \tag{58}$$
$$= \frac{1}{20 * 3 * 0.05} \tag{59}$$
$$= \frac{1}{3} = 0.3333 \tag{60}$$

It shows that the efficiency of the estimated and the theoretical are similar.

– As an auxiliary distribution q_j, $c * q_j$ must always larger or equal to p_j. So for the constant c here, $c = 3$ gives the best efficiency on estimation. To confirm this, try $c = 4$ and get the estimated efficiency:

$$Efficiency_{sample}(c = 4) = 0.2424 \tag{61}$$

Theoretically, when $c = 4$, the efficiency is:

$$Efficiency_{theoretic}(c = 4) = \frac{\sum_{j=1}^{20} p_j}{\sum_{j=1}^{20} c * q_j} \tag{62}$$
$$= \frac{1}{20 * 4 * 0.05} \tag{63}$$
$$= \frac{1}{4} = 0.25 \tag{64}$$

Both estimated and theoretical efficiency on $c = 4$ is lower than $c = 3$. For further imporvement of the efficiency, we can either change the distribution of q_j or the value of c.

6 Reference

- http://mathworld.wolfram.com/UniformSumDistribution.html
- http://cs.brown.edu/about/system/managed/latex/doc/symbols.pdf