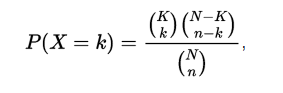
Project #3 – Samples and statistics

1. *A components manufacturer delivers a batch of 125 microchips to a parts distributor. The distributor checks for lot conformance by counting the number of defective chips in a random sampling (without replacement) of the lot. If the distributor finds any defective chips in the sample it rejects the entire lot. Suppose that there are six defective units in the lot of 125 microchips.*
2. *Simulate the lot sampling to estimate the probability that the distributor will reject the lot if it tests five microchips.*
3. *What is the fewest number of microchips that the distributor should test to reject this lot 95% of the time?*

**Concept behind this problem:**

The question requires the knowledge of *Hyper Geometric Random Variables*. Hypergeometric distribution is a discrete probability distribution that describes the probability of k successes in n draws, without replacement, from a finite population of size N that contains exactly K successes, wherein each draw is either a success or a failure.

In contrast, the binomial distribution describes the probability of k successes in n draws with replacement.



In out Question, Lot size given N=125, Defective items (no: of success in lot) k=6, Sample Size,n= 5 and x=1(rejection of lot).

By solving this, we get the **theoretical probability as 0.2212.**

**Code & Code Description:**

* Initialize variables and mark numbers 1-6 as defective in a lot containing 125 numbers.
* The function expects ‘sample\_lot’ as input from the user
* In the below code, we perform two things

1. Probability that the distributor will reject the lot if it tests 5 microchips.
   * Based on the user input of sample\_lot, we run this piece of code sample\_lot times and reject if the randsample(5 samples) has a number from 1 to 6.
   * Probability = (num of rejects)/total sample size

function [ ] = prob\_reject\_lot(sample\_lot)

%Initialize

reject = 0;

N\_microchips = 125;

defective = [1 2 3 4 5 6];

prob\_rejection =0.95;

max\_limit = 1000;

%Probability to reject the lot if it tests 5 microchips.

for iter=1:sample\_lot

sample = randsample(N\_microchips,5);

check\_def = ismember(sample,defective);

if(sum(check\_def) >0)

reject = reject +1;

end

end

disp(['Among ',num2str(sample\_lot),' sample lot, ',num2str(reject),' samples got rejected']);

disp(reject);

disp(['Probability that the distributor will reject the lot if it tests 5 microchips ', num2str(reject/sample\_lot)]);

1. Fewest number of microchips that the distributor should test to reject a lot 95% of the time
   * Here we have two while loops.
   * Outer while loop to keep track of the increments for the number of microchips to be selects to attain the 95% rejection ration
   * Inner while loop to keep track of the reject count (95%).
   * Once we attain the 95% reject ratio, we showcase the number of microchips to be selected to attain such a high rejection ratio.

%Fewest number of microchips to reject a lot 95% of the time

count\_out=1;

flag\_out =1;

while (flag\_out == 1)

count\_out = count\_out+1;

reject = 0;

count\_in = 0;

flag\_in = 1;

while(flag\_in == 1)

count\_in = count\_in+1;

sample = randsample(N\_microchips,count\_out);

check\_def = ismember(sample,defective);

if(sum(check\_def) >0)

reject = reject +1;

end

if (reject == prob\_rejection \* count\_in)

disp(['Total runs of experiment ',num2str(count\_in),' of which ',num2str(reject),' lots are rejected ']);

disp(['Fewest number of microchips to test for rejection of lot 95% of the time is ',num2str(count\_out)]);

flag\_in = 0;

flag\_out = 0;

end

if count\_in == max\_limit

flag\_in = 0;

end

end

end

**Results:**

* Among 10000 sample lot, 2262 samples got rejected 2262
* Probability that the distributor will reject the lot if it tests 5 microchips **0.2262**
* Total runs of experiment 20 of which 19 lots are rejected
* Fewest number of microchips to test for rejection of lot 95% of the time is **37**

**Observations:**

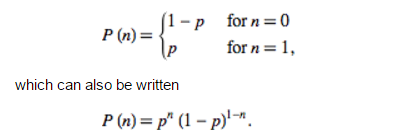
* Theoretical Probability for the distributor will reject the lot if it tests five microchips (0.2212) is closer to the calculated probability (0.2262)
* Rejection is 95% if the sample taken is in a range of 32-40.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. *Suppose that 120 cars arrive at a freeway onramp per hour on average. Simulate one hour of arrivals to the freeway onramp:* 
   1. *subdivide the hour into small time intervals (< 1 second) and then*
   2. *perform a Bernoulli trial to indicate a car arrival within each small time interval. Generate a histogram for the number of arrivals per hour.*
   3. *Repeat the counting experiment by sampling directly from an equivalent Poisson distribution by using the inverse transform method (described in class). Generate a histogram for the number of arrivals per hour using this method.*
   4. *Overlay the theoretical p.m.f. on both histograms. Comment on the results.*

**Concept behind this problem:**

*Bernoulli Distribution* is the probability distribution of a random variable which takes the value 1 with success(n=1) probability of p and the value 0 with failure(n=0) probability of q=1-p



*Inverse Transform Method* uses random number generator that samples from a uniform distribution on [0, 1] to sample from another distribution. P{X=xj} = pj .

We first generate a uniform random number(U) and set X=x0 if U<p0; X=x1 if U<p0+p1 and so on.

**Code & Code Description:**

* Initialize variables and marked Lambda as 120.
* Bernoulli Trail:
  1. P\_value = lambda/sub\_interval(user input-5000)
  2. Generate random variables and compare it with P\_value.
  3. Approximate a sample of Poisson random variable by summing up the successes in the series of Bernoulli trial.

%Initialize

lambda = 120;

Y = 1:1000;

%Bernoulli Method

for iter1=1:1000

p\_value = lambda/sub\_intervals;

r\_number = rand(sub\_intervals,1);

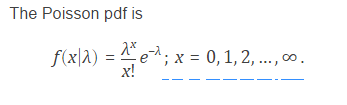
Bernoulli\_trails = r\_number < p\_value;

vec\_bernoulli(iter1) = sum(Bernoulli\_trails);

end

hist(vec\_bernoulli);

* Inverse Transform Method:
  1. Computes the Poisson pdf at each of the values in X using mean parameters in lambda using ***poisspdf*** function of matlab -



* 1. Generate a random variable u –[0,1]
  2. If u<Po, X=Xostop.
  3. Otherwise u<Po+P1, X=X1. Stop
  4. Repeat the experiment 1000 times and store value of counter.

%Inverse Transform Method

for iter2 = 1:1000

p\_value = poisspdf(0,lambda);

f\_value = p\_value; %p0 value

index = 0;

r\_number2 = rand();

while(r\_number2 > f\_value) %p0+p1+... until r\_number2<summation

p\_value = poisspdf(index+1,lambda);

f\_value = f\_value+p\_value;

index = index+1;

end

vec\_inv\_tranfm(iter2) = index;

end

* Theoretical Method:
  1. Calculate distribution of poisson PDF for x=1-1000.

%Theoritical Value

for iter3=1:1000

vec\_theo(iter3) = poisspdf(iter3,lambda);

end

**Histograms & Plots:**

Figure 1-Bernoulli Trail Histogram:Car arrivals per hour(5000) intervals

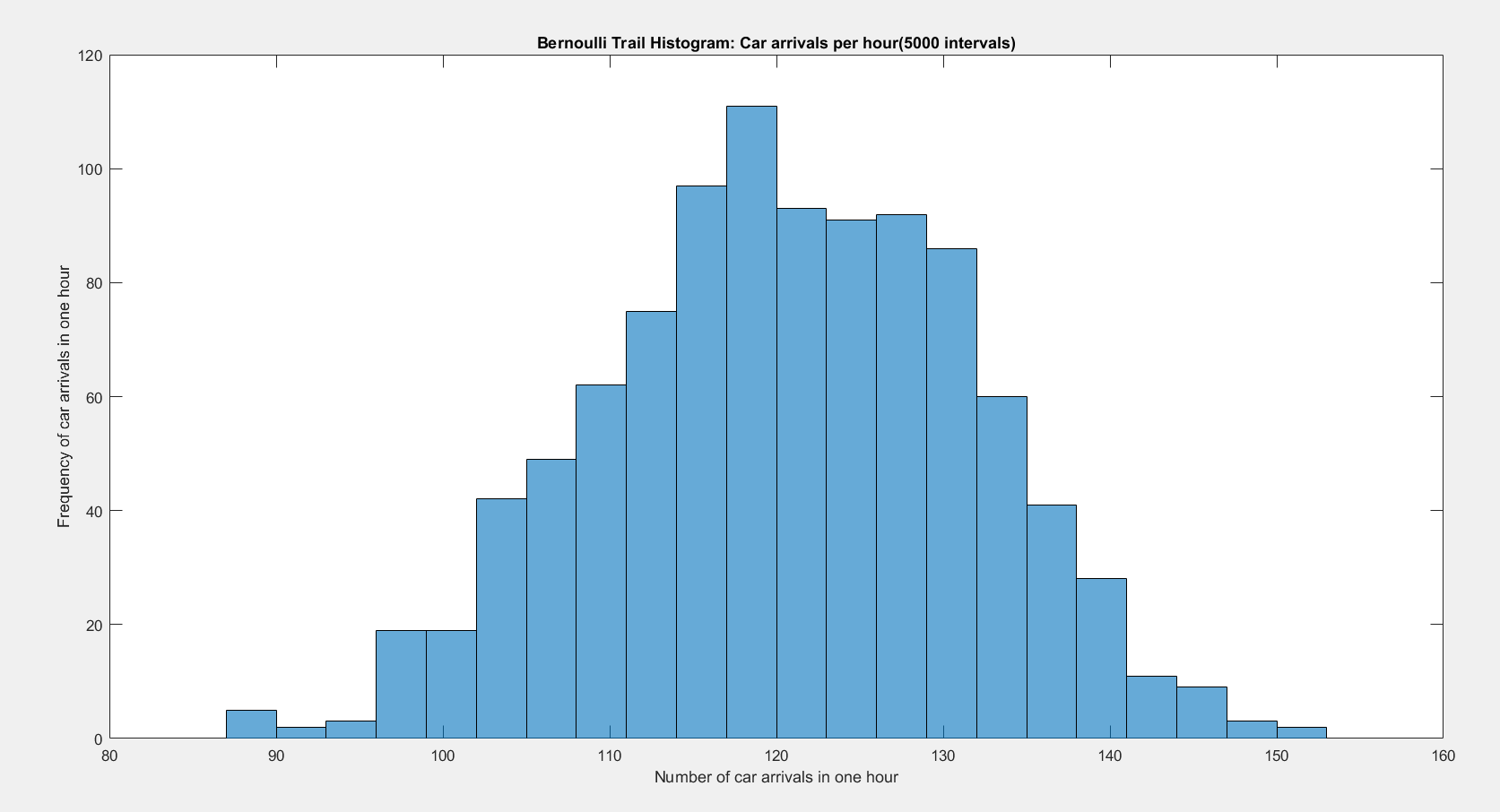


Figure 2- Bernoulli Trail overlayed on Theoritical PMF

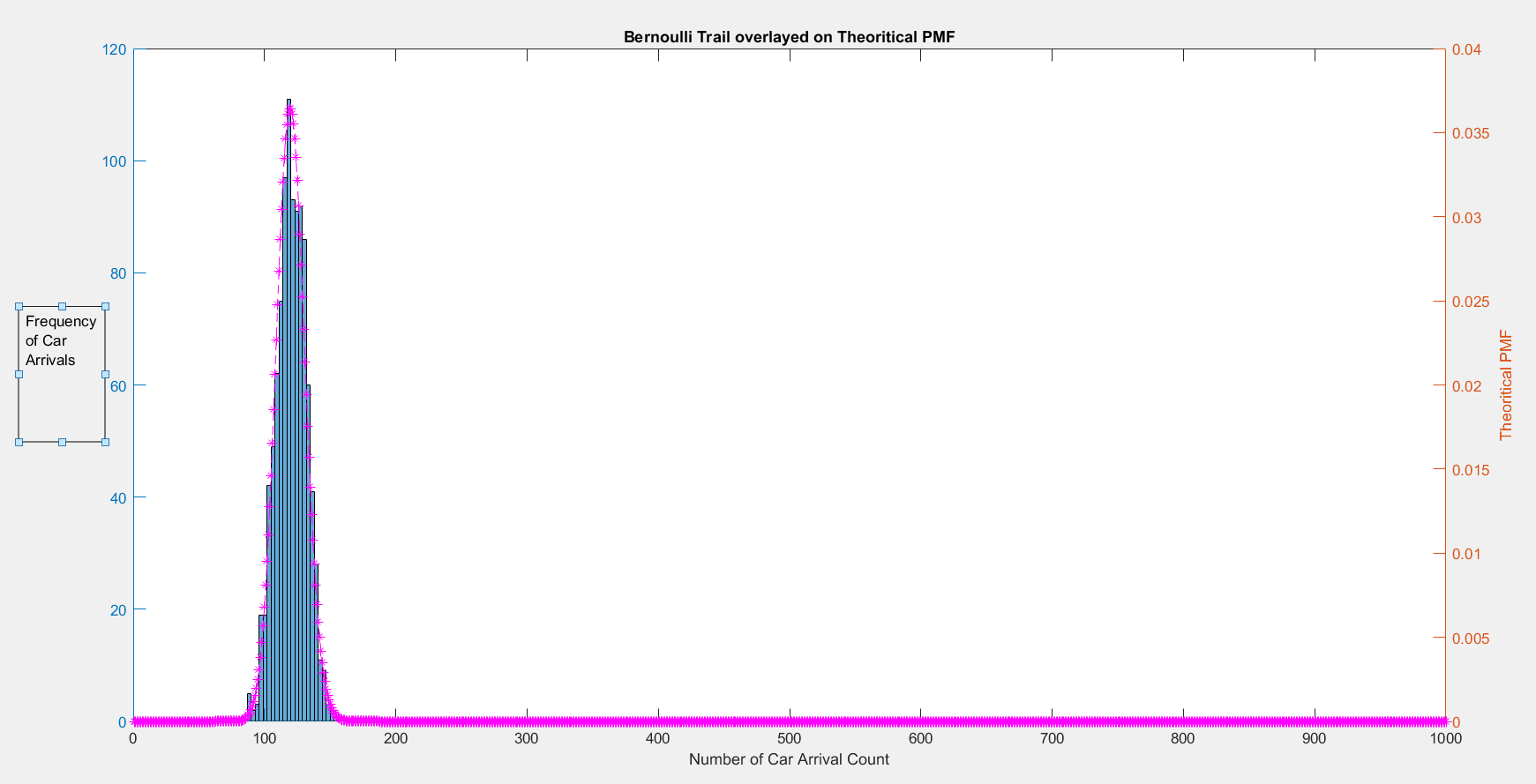
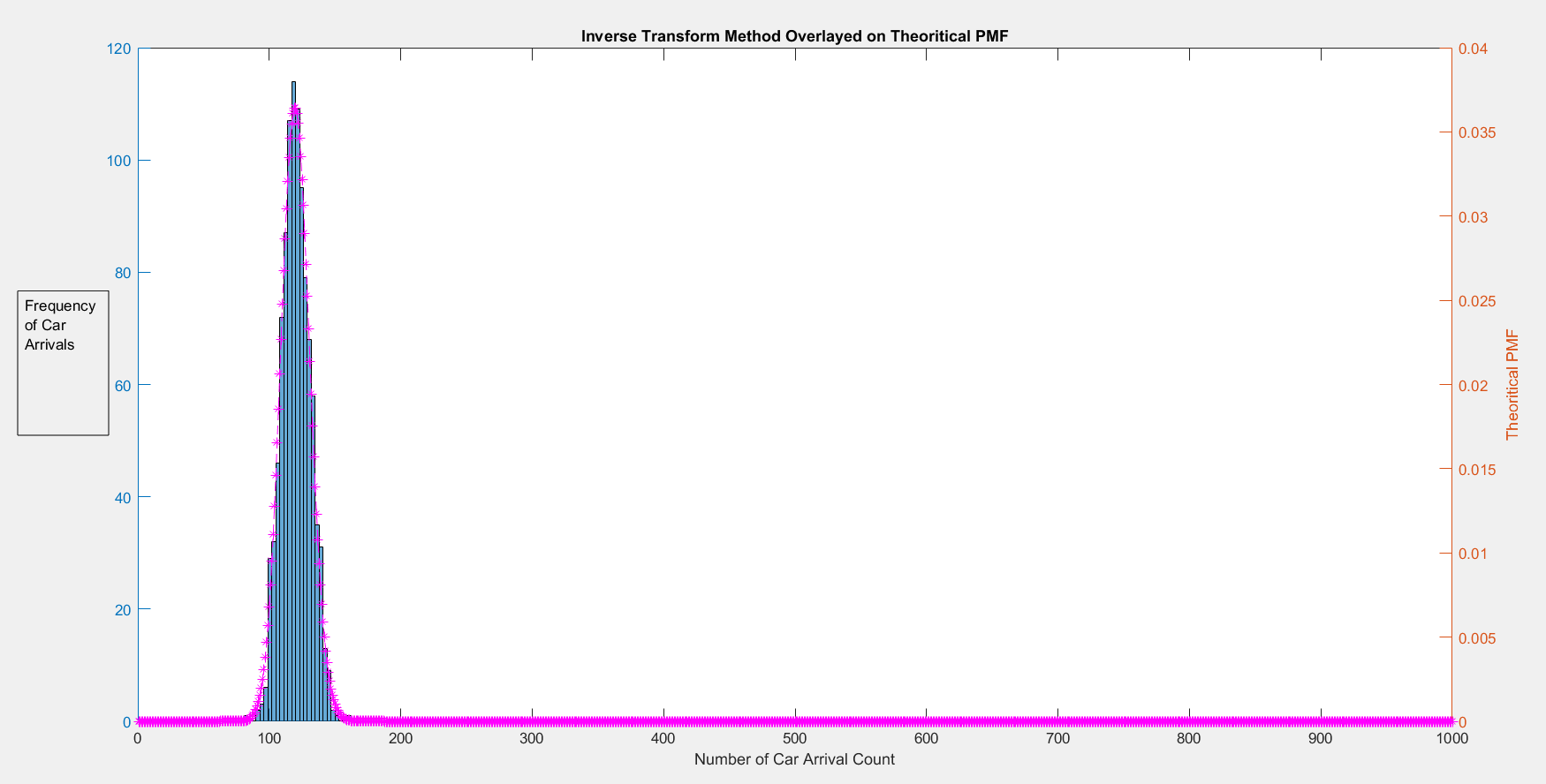


Figure 3- Transform Method Overlayed on Theoritical PMF



**Results:**

* Around car arrival rate of 120, we observe maximum PMF

**Observations:**

* Bernoulli approximation for Poisson random variable gets finer and finer as we make the subinterval smaller.
* In Poisson random variable models, the number of occurrences of an event in a time interval where the event may occur at any point of time equally likely, we can characterize it by cutting the whole interval into a large number of small subintervals and there is a Bernoulli trial in each subinterval.
* Bernoulli or Poisson distribution calculated almost lies within the theoretical PMF laid over which means that the random variables fit the Poisson and Bernoulli distribution.
* Maximum PMF is around 120 car arrival rate.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3) *Define the random variable N =min{n: Σ Xi> 4} for i=1-n as the smallest number of uniform random samples whose sum is greater than four. Generate a histogram using 100, 1000, and 10000 samples for 􀜰. Comment on E[N]*

**Concept behind this problem:**

|  |
| --- |
| The problem requires the concept of Wald’s Equation:  Let X_1, ..., X_N be a sequence of independent observations of a random variable X, and let the number of observations N itself be chosen at random. Then Wald's equation states that the expectation value of the sum X_1+...+X_N is equal to the expectation value of X times the expectation value of N,  <X_1+...+X_N>=<X><N> |

**Code & Code Description:**

* Initialize variables and the function expects ' no\_of\_runs’ as input from user
* Generate random samples (upto 1000) in the interval [0 1]
* Keep adding the consecutive numbers until the sum of all the random numbers is equal to or greater than 4.
* Every note of counter and repeat the experiment the number specified by user input.
* Get the mean(expected value) of the samples and plot histogram for 100, 1000, 10000 intervals.

%Initialize

index=1;

for iter = 1:no\_of\_runs

count = 0;

sum\_of\_Rand\_samples = 0;

r\_number = rand(20,1);

while(sum\_of\_Rand\_samples<=4)

count = count + 1;

sum\_of\_Rand\_samples = sum\_of\_Rand\_samples + r\_number(count);

end

no\_of\_samples(index) = count;

index = index+1;

end

disp(['E[N] of ',num2str(no\_of\_runs),' samples of N is',num2str(mean(no\_of\_samples))]);

xlabel('min numbers exceeding sum-4');

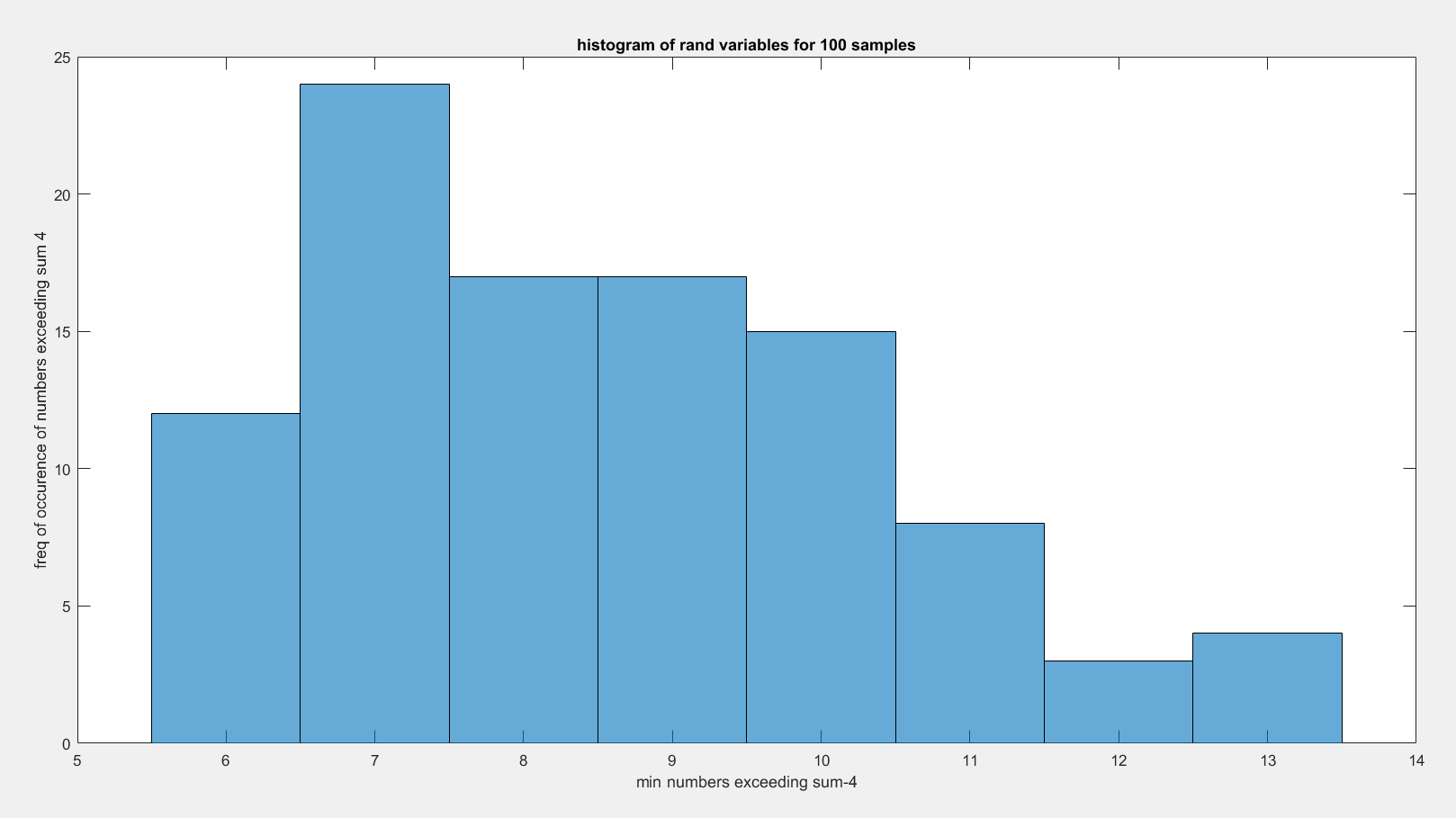
ylabel('freq of occurence of numbers exceeding sum 4');

histogram(no\_of\_samples);

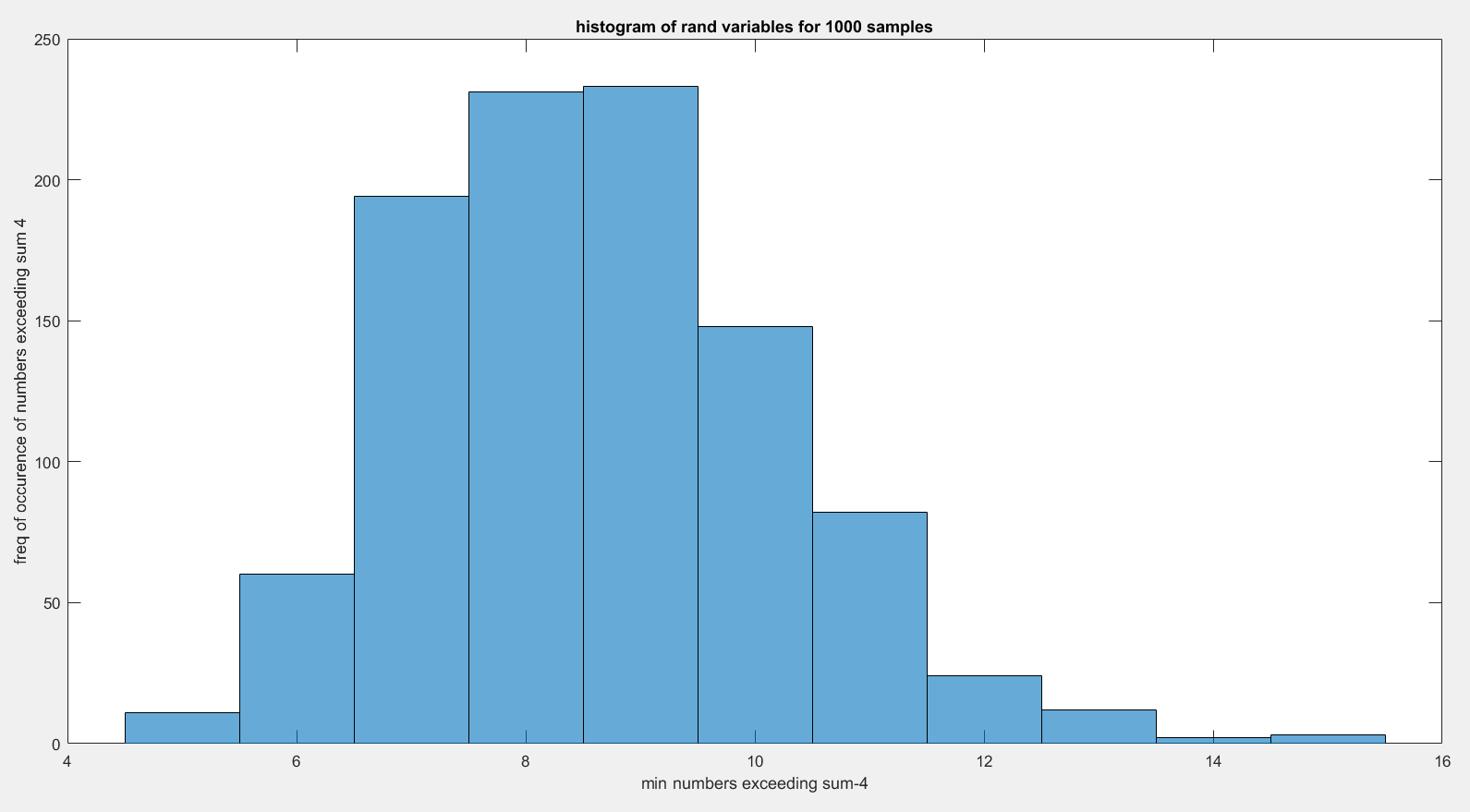
title(['histogram of rand variables for ',num2str(no\_of\_runs),' samples']); end

**Histograms & Plots:**

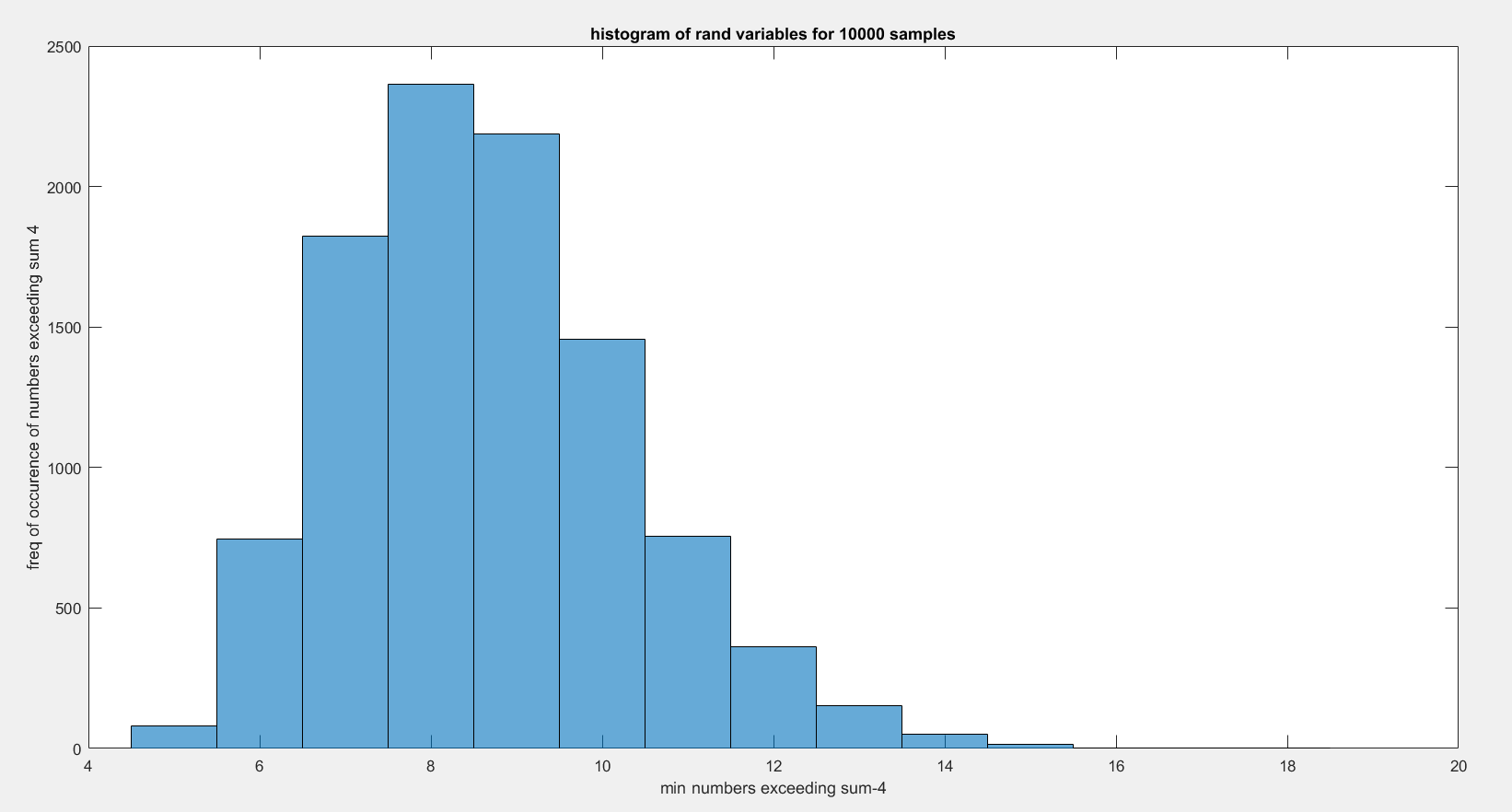
E[N] of 100 samples:



E[N] of 1000 samples



E[N] of 10000 samples



**Results and observations:**

* ee511\_p3\_q3(100) : E[N] of 100 samples of N is8.46
* ee511\_p3\_q3(1000) : E[N] of 1000 samples of N is8.656
* ee511\_p3\_q3(10000): E[N] of 10000 samples of N is8.6578
* 0.5(mean of numbers to reach sum<4 ) \* 8 is 4 which means that the mean should lie in range of 8-9

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

5*) Use the accept‐reject method to sample from the following distribution Pj by sampling from the (non‐optimal) uniform auxiliary distribution (Qj = 0.05 for j= 1, … , 20):*

1. *Generate a histogram and overlay the target distribution Pj*
2. *Compute the sample mean and sample variance and compare these values to the theoretical values. Estimate the efficiency of your sampler with the following ratio:*
3. *Compare your estimate of the efficiency to the theoretical efficiency given your choice for the constant c.*

**Concept behind this problem:**

Acceptance-rejection methods begin with uniform random numbers, but require an additional random number generator. Acceptance-rejection methods also work for continuous or discrete distributions. It is much quicker because it neither requires CDF nor the inverse to be calculated.  In this case of discrete distribution, the goal is to generate random numbers from a distribution with probability mass Pp(X = i) = pi. Use samples from auxillary distribution (Qj) that dominates Pj. Pj/Qj <= C where C is constant.

* Generate Y=qj
* Generate X=U[0,1]
* If U<Pj/(c x Qj) set X=Y and stop. Else repeat step 1

**Code & Code Description:**

* In the question, we are given Qj(j=1 to 20) and Pj(1-20). To calculate C, we use max(p/q) and we get C = 0.15/0.05 = 3
* Number of samples I have taken = 100,000
* Generate a uniform random variable and check if u<p/(cq).
* If it fits, accept otherwise reject. Store the value of counter and uniform variable.
* Now plot graph of target distribution of accepted values taking given PMF on one Y axis and Normalized values of accepted fit on a scale of 20(to fit into the graph).
* Then calculate the Theoretical and calculated values of mean, variance and efficiency.

function [ ] = accept\_reject(sub\_intervals)

p\_value = [6 6 6 6 6 15 13 14 15 13 0 0 0 0 0 0 0 0 0 0]/100;

No\_of\_samples = 100000;

seq\_nums = 1:length(p\_value);

c=0.15/0.05;%Max of p/q

for iter = 1:No\_of\_samples

count = 0;

while 1,

count = count + 1;

uniform\_j = 1 + floor(20\*rand); % Get Uniform j

if ((c\*rand) < p\_value(uniform\_j)/0.05) % Accept p(j) if U<p(j)/c, q(j)= 0.1

accept(iter) = uniform\_j;

theo\_constant\_C(iter) = count; break; end; end; end;

**Results of Theoretical and Calculated Values:**

Sample Mean: Calculated = 6.4972 Theoretical = 6.48

%Calculated Mean and Theoritical Mean

theo\_mean = sum(p\_value.\*[1:20]);

calc\_mean = mean(accept);

disp(['Sample Mean: Calculated = ',num2str(calc\_mean),'

Theoretical = ',num2str(theo\_mean)]);

Sample Variance: Calculated = 7.1475 Theoretical = 7.1896

%Calculated Var and Theoritical Var

theo\_var =sum( p\_value.\*([1:20]-theo\_mean).^2);

calc\_var=var(accept);

disp(['Sample Variance: Calculated = ',num2str(calc\_var),' Theoretical = ',num2str(theo\_var)]);

Efficiency: Calculated = 33.3747 Theoretical = 33.3747

theo\_efficiency = 100/mean(theo\_constant\_C);

calc\_efficiency = 100\*100000/(sum(theo\_constant\_C));

disp(['Efficiency: Calculated = ',num2str(calc\_efficiency),' Theoretical = ',num2str(theo\_efficiency)]);

**Histograms & Plots:**

yyaxis left;

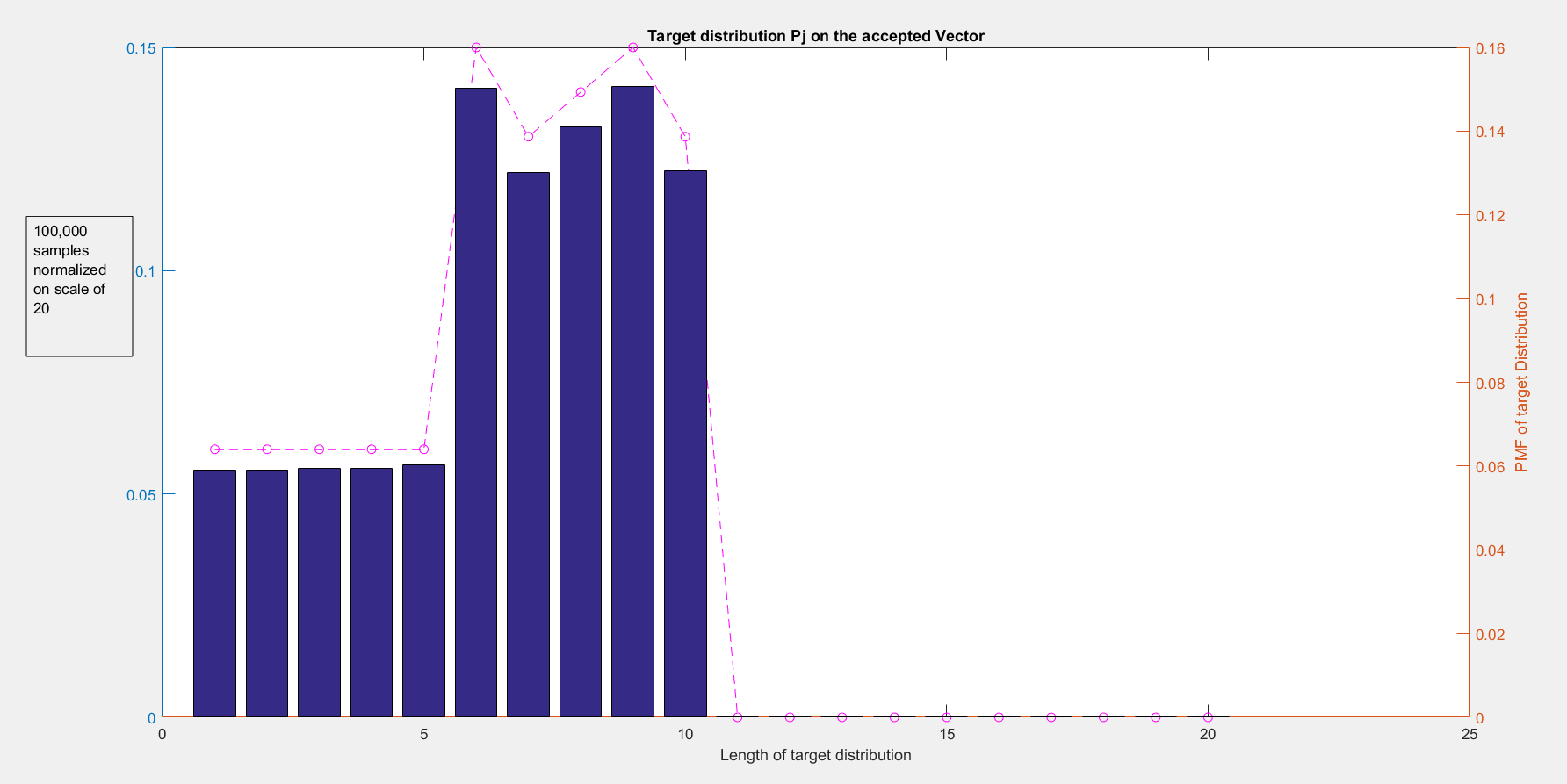
plot(p\_value,'m--o');

yyaxis right;

counts = histc(accept,seq\_nums);

%Normalizing the graph from 100,000 samples to 20 count

bar(seq\_nums,counts/sum(counts));



**Observations:**

* Mean, Variance and Efficiency of Calculated value is almost matching with Theoretical values for 100000 samples.

4) Produce a sequence {Xk} where Pj = p/j for j= 1, 2, ⋯ , 60. [This is equivalent to spinning the minute hand on a clock and observing the stopping position If P[stop on minute] = p/j ]. Generate a histogram. Define the random variable Nj = min{k:Xk=j} Simulate sampling from N60. Estimate E[N60] and Var[N60]. Compare you estimates with the theoretical values.

**Concept behind this problem:**

The *Geometric Probability Distribution* represents the number of failures before you get a success in a series of Bernoulli trials. This discrete probability distribution is represented by the probability density function:

f(x) = (1 − p)^x − 1p , E[X] = 1/p, Var[X] = (1-p)/p^2

*Negative Binomial Experiment:* A statistical experiment that has the following properties:

* The experiment consists of x repeated trials.
* Each trial can result in just two possible outcomes. We call one of these outcomes a success and the other, a failure.
* The probability of success, denoted by p, is the same on every trial.
* The trials are independent; that is, the outcome on one trial does not affect the outcome on other trials.
* The experiment continues until r successes are observed, where r is specified in advance

**Code & Code Description:**

* First calculate P by means of P[1+1/2+1/3,….1/60] = 60 with number of samples = 10000
* Then calculate p/60 as well as generate a sequence of p/j(j from 1 to 60).
* Here we use inverse transform to get the random variable where P is fixed and u is varying. i.e.
  1. Generate a uniform distribution of random sample(using rand)
  2. Now using inv transform check each element of rand(u) is fitting in which prob distribution.
  3. We take one Pi and match with 1000 samples and calculate how many of them are within the given threshold

%Initialize

seq\_60 =(1:60);

p\_value = harmmean(seq\_60)/60;

prob\_j = p\_value./seq\_60;

prob\_60 = p\_value/60;

normalization = 1/p\_value;

Random\_variables=[]

for iter1 = 1:60

count = 0; sum1 = 0; sum2=0;

sum1=prob\_j(iter1)+sum1;

while seq\_60<=No\_of\_samples

if (Samples(seq\_60)>=sum2)&(Samples(seq\_60)<sum1)

count=count+1;

end

seq\_60=seq\_60+1; end

Random\_variables(iter1)=count;

sum2=sum1; end

* For part-b, pick a random number and keep iterating it in a loop until we get the number p/60. Keep iterating the same experiment and keep note of the count it takes to reach 60 th position(p/60)

%To keep track of the number of count it took to hit N60

for iter=1:1:No\_of\_samples

count=0;

while(rand >=prob\_60)

count=count+1;

end

tracking\_matrix(iter)=count;

end

**Results of Theoretical and Calculated Values:**

Mean: Calculated = 279.506 Theoretical = 280.7922

calc\_mean=mean(tracking\_matrix);

theo\_mean=1/(prob\_60);

disp(['Mean: Calculated = ',num2str(calc\_mean),' Theoretical = ',num2str(theo\_mean)]);

Variance: Calculated = 79809.4713 Theoretical = 78563.4813

theo\_var=(1-prob\_60)/(power(prob\_60,2));

calc\_var=var(tracking\_matrix);

disp(['Variance: Calculated = ',num2str(calc\_var),' Theoretical = ',num2str(theo\_var)]);

**Plots and Histograms:**

figure(1)

bar(1:60,Random\_variables)

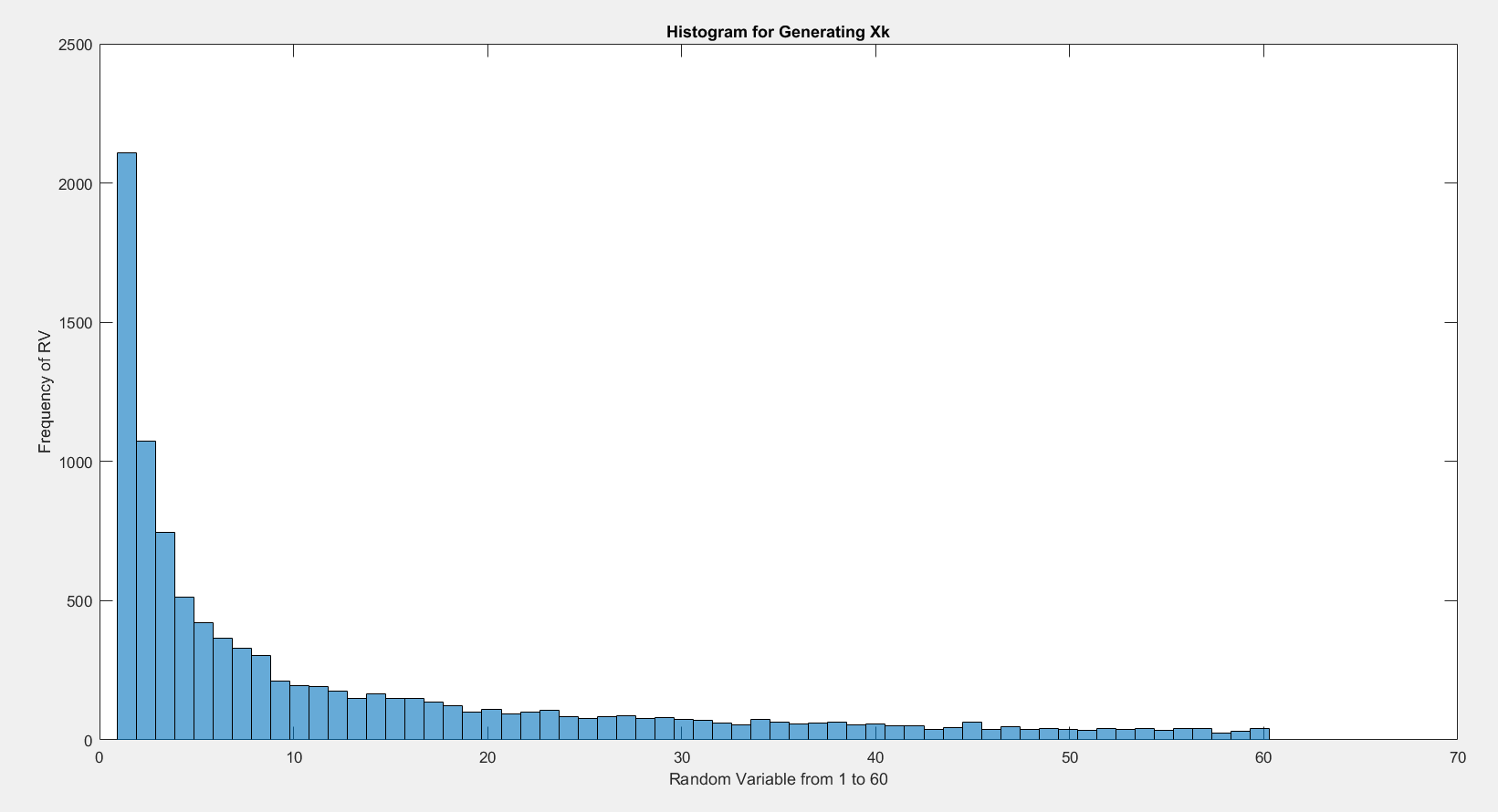
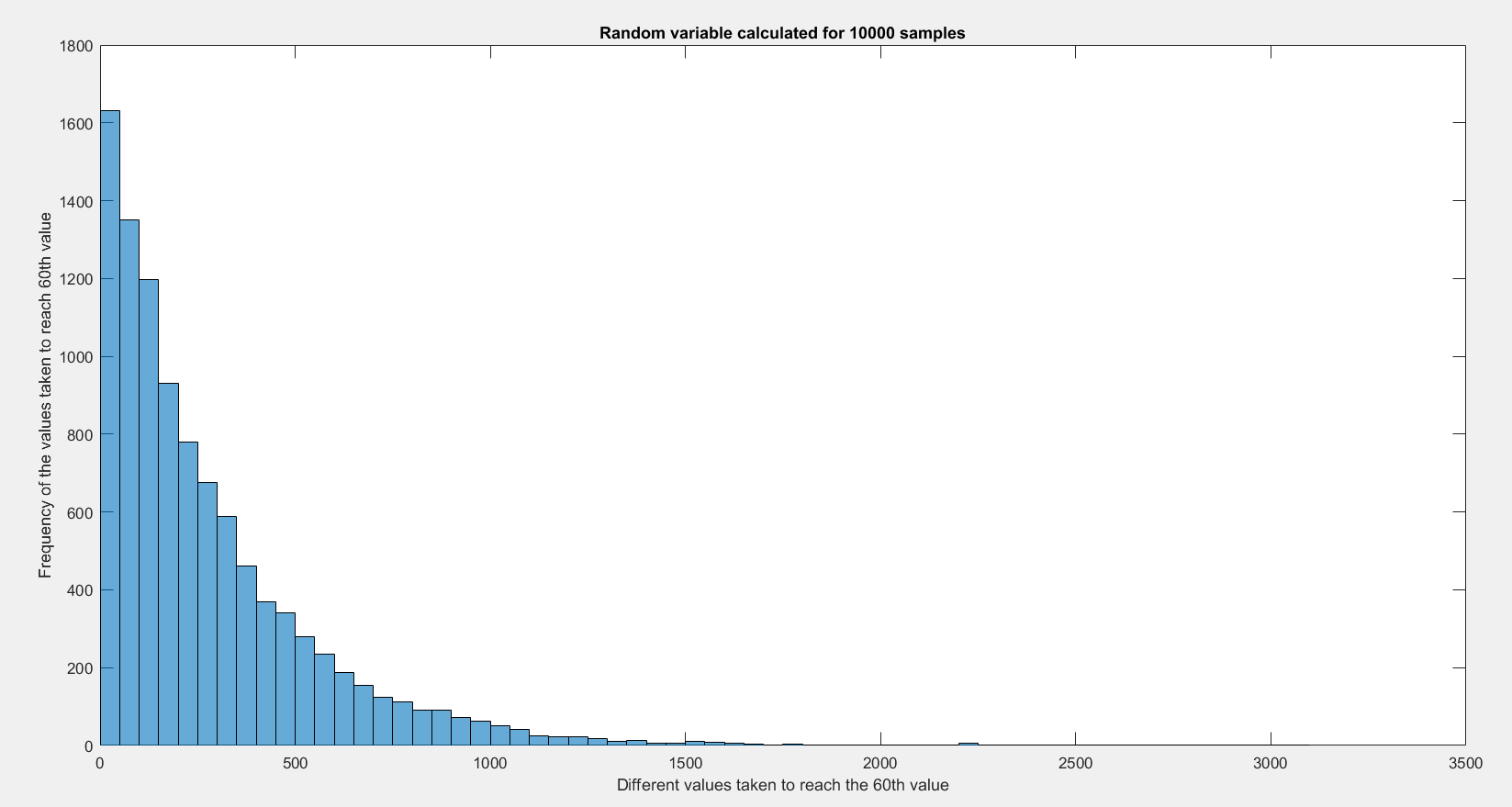


Figure-2 'Random variable calculated for ',num2str(No\_of\_samples),' samples

histogram(tracking\_matrix);

xlabel('Different values taken to reach the 60th value');

ylabel('Frequency of the values taken to reach 60th value');



**Observations:**

* Mean and Variance of Calculated value is almost matching with Theoretical values for 10000 samples.
* In the graph for generating Xk, P(1) ~= 2P(2) and hence from the graph we can see that the random number of outcomes for X1 is almost twice of X2. From this we can imply that the sample is fitting into the given distribution.