Omer Solakli

2471677864

**EE 511: Simulation of Stochastic Processes**

**Spring 2018**

**Project#2**

**[Networking Again]**

n=50; %number of people

p=0.02; %probability of connection

u=zeros(n,n); %to generate the random numbers

A=zeros(n,n); %generating nxn matrix to select node i j if there is any connection between people

for i=1:1:n %iterating over rows

for j=1:1:n %iterating over columns

if i==j

A(i,j)=0; %to make sure people no self-connectio

end

u(i,j)=rand; %random sampling

if u(i,j)<p

A(i,j)=1;

A(j,i)=1; %ensuring if 2 nodes connected they should be connected in both ways since it is an undirected graph.

else

A(i,j)=0;

A(j,i)=0;

end

end

end

G=graph(A); %putting the matrix values in a graph

plot(G); %plotting the graph

****

Figure Graph with p=0.02

****

Figure Graph with p=0.09



Figure Graph with p=0.12

n=50; %number of people

p=0.02; %probability of connection

u=zeros(n,n); %to generate the random numbers

A=zeros(n,n); %generating nxn matrix to select node i j if there is any connection between people

d=zeros(1,100); %generating a sparse array for the histogram bins, 100 bins seemed adequate for me

for k=1:1:100

for i=1:1:n %iterating over rows

for j=1:1:n %iterating over columns

if i==j

A(i,j)=0; %to make sure people no self-connectio

end

u(i,j)=rand; %random sampling

if u(i,j)<p

A(i,j)=1;

A(j,i)=1; %ensuring if 2 nodes connectied they should be connected in both ways since it is an undirected graph.

else

A(i,j)=0;

A(j,i)=0;

end

end

G=graph(A); %putting the matrix values in a graph

d(k)=numedges(G); %counting the degree of a graph for each sample k

end

end

histogram(d);



Figure Histogram for p=0.02



Figure Histogram for p=0.09



Figure Histogram for p=0.12

n=100; %number of people

p=0.06; %probability of connection

u=zeros(n,n); %to generate the random numbers

A=zeros(n,n); %generating nxn matrix to select node i j if there is any connection between people

d=zeros(1,100); %generating a sparse array for the histogram bins, 100 bins seemed adequate for me

for k=1:1:100

for i=1:1:n %iterating over rows

for j=1:1:n %iterating over columns

if i==j

A(i,j)=0; %to make sure people no self-connection

end

u(i,j)=rand; %random sampling

if u(i,j)<p

A(i,j)=1;

A(j,i)=1; %ensuring if 2 nodes connected they should be connected in both ways since it is an undirected graph.

else

A(i,j)=0;

A(j,i)=0;

end

end

G=graph(A); %putting the matrix values in a graph

d(k)=numedges(G); %counting the degree of a graph for each sample k

end

end

histogram(d);



Figure Histogram for p=0.06 n=100

We can easily see that the number of connections, or vertices, increase with increasing probability of connection among people. From the graphs, we see that with p=0.02 some nodes are left unconnected since the expected number of connections (24.5) are less than the number of people (50).

We can see that the there is no isolated vertex for the 2nd and the 3rd graphs, that is because the expected number of connections for those graphs are 110.25 and 147 respectively.

When we see the histograms for the degree of the graphs. We see that that they are centered around the expected number of connections. As the probability of a connection increases the degree of the graph is also increasing. Using law of large numbers and convergence in probability, it can be said that as number of samples goes to infinity the histograms will converge to Gaussian distribution with mean being the expected degrees respectively.

Figure 7 shows that for n=100 p=0.06 we have 100x99/2=4950 possible number of edges. When we distribute this with p=0.06, if we assume the distribution is binomial we see that we expect the degree of the graph to be 4950x0.06=297. And fortunately figure 7 agrees with this fact. When we sample 100 times, 45 of the samples had the degree between 290-310. Also, there is no sample having degree more than 360 or less than 260.

It can be seen that all of the graphs meet with binomial distribution assumptions such as each connection modeled as a success or a fail whether a connection exist or not. There is only 2 outcomes for each vertex and this is true for all vertices independently.

**[Waiting]**

nbins=30; %number of bins

t=0.2; %interarrival times

r=(1/t); %arrival rate

n=1000; %no of samples

u=zeros(1,n); %no of uniforms

x=zeros(1,n); %vector to store the samples

F=1-exp(-r\*x); %cdf of exponential

expected=zeros(1,nbins); %expected number of observations

p=zeros(1,nbins); %probability of falling into the bin

for i=1:1:n

u(i)=rand; %generating samples

x(i)=-log(u(i))\*t; %inverse CDF method to generate the observed numbers

end

[O,edges] = histcounts(x,nbins); %sampled exponential interarrival times are put into a histogram with 30 bins with each bin having uniform width determined by upper and lower bound of vector x and N as number of samples in each bin

for k=2:1:nbins+1

p(k-1)=(1-exp(-r\*edges(k)))-(1-exp(-r\*edges(k-1))); %probability of falling into each bin

end

expected=n\*p; %expected number of samples in each bin

test\_stat=0;

for m=1:1:nbins %iterating over bins

test\_stat=test\_stat+(((O(m)-expected(m))^2)/(expected(m))); %summing over each bin to find the test statistic

end

Above code samples the exponential inter-arrival times using inverse-cdf method and put them in a histogram with 30 bins with each having uniform width determined by upper and lower bounds of the observations. The ith column of matrix O represents the observed number of arrival in ith bin. The first for loops iterates through number of bins to calculate the probability of observations in each bin by subtracting the CDF values of each adjacent bins. The expected number of observations in each bin are found multiplying this probability with number of samples.

Then the 2nd for loop iterates through bins to calculate our test\_statistic variable which will be used for goodness of fit test. We know that since we have 30 bins the degree of freedom is 29 since we have not estimated any parameters. Chi-square value for 29 degrees of freedom and 0.05 level of significance is 42.557. When I run the code above, the test statistic turned out to be is 30.45. Thus, we cannot reject that the distribution coming from an exponential distribution. This means our RNG works fairly well.

t=0.2; %interarrival times

r=(1/t); %arrival rate

n=1000; %no of time units

x=0; %time variable starting from zero

no\_of\_events=zeros(1,n); %vector for counting the number of events within each unit time

while x<1000 %while time is smaller than 1000

u=rand; %generating samples

x=x+(-log(u)\*t); %in each arrival we update x to model the arrival times

for i=1:1:1000

if x <=i && x>i-1

no\_of\_events(i)=no\_of\_events(i)+1;

end

end

end

H = histcounts(no\_of\_events); %sampled exponential interarrival times are put into a histogram as H(i) represents observed i number of arrivals

p=zeros(1,length(H));

expected=zeros(1,length(H));

test\_statistic=0;

for k=0:1:length(H)-1

p(k+1)=(r^k\*exp(-r))/factorial(k); %poission distribution for arrivals per time unit. k is number of arrivals in each time unit

expected(k+1)=n\*p(k+1); %expected number of arrivals for each bin

test\_statistic=test\_statistic+((H(k+1)-expected(k+1))^2/(expected(k+1)));

end

First, the time variable x is formed to simulate the waiting-time as exponential distribution using random number generation. Then no\_of\_events basically counts how many arrivals occurred in each time unit which represents the arrival rate of the system. For each time unit between 0 and 1000 the number of arrivals are checked by increasing the ith element of no\_of\_events variable by one whenever arrival occurs within that time. We assume that the number of events are distributed according poission random variable with interarrival rate being 5. We expect to have 5 arrivals per time because we know that each arrival occurs in 0.2 time units.

The observed number of arrivals and the expected number of arrivals are calculated next and the test\_statistic again is found using chi-square test.

When I ran the above code. I get maximum number of observed arrivals as 14, thus I used 14 bins for goodness of fit test. Chi-square value for 13 degrees of freedom with 0.05 level of significance is 22.362. My test\_statistic variable is 14.80 which tells us not to reject that the arrival rate per time unit is a Poisson distribution.

**[Double Rejection]**

a=0; %bound from left

b=6; %bound from right,

c=1.5;% upper bound for pdf. 0.5◊Beta(8,5) ,0<x?1 since Beta(8,5) cannot achieve a value more than 3.

pd = makedist('Beta','a',5,'b',8); %pdf of beta distribution

rejection=0; %initialize variables

sample=0;

count=0;

while sample<1000 %we need to make sure we have generated 1000 accepted samples

v=c\*rand;%generating the y axis

u=3\*rand; %it is sufficient to generate numbers btw (0,3) since numbers generated btw 1,4 not used. we can manipulate pdf to use the numbers we generated

if u>0 && u<=1 %if x is between

y=0.5\*pdf(pd,u); %y value for the x

if v>y %rejection condition

rejection=rejection+1;

else

sample=sample+1; %else we accept

end

else

if u>1 && u<=2 %if u is between 1,2 we use it as it is between 4,5

y=0.5\*(u-1);

if v>y

rejection=rejection+1;

else

sample=sample+1;

end

else

if u>2 && u<=3 %we use u as it is between 5,6

y=-0.5\*(u-3);

if v>y

rejection=rejection+1;

else

sample=sample+1;

end

end

end

end

count=count+1; %we are counting each random number generation to track rejection rate

end

rejection\_rate=rejection/count;

Using above RNG, my rejection\_rate was 0.76. The reason behind this high rejection is because I set the max value of y as 1.5 where as if x is between (4,6) this max value is 0.5. Thus, there is an unnecessary bound for x∈(4,6). One can manipulate this piece-wise region to decrease the rejection rate of their random number generators to achieve less computation time.

When I solved this question I realized that there is no definite sample number to generate 1000 realizations of the distribution. So that, we should use while rather than using for loops. Also, it is sufficient to generate 3 random variables for the x-component. Since there is a gap in the region x∈(1,4). So, one can generate random numbers between 0 and 3 and manipulate the pdf accordingly to use less computation.