# EE511 Project 2 Summary

**SilongHu email:silonghu@usc.edu**

**This summary is divided into 3 parts: Implementation, Conclusion and Source Code**

## I. Implementation

### Problem 1. [Waiting]

(The Inverse Transform Method) Let F(x), $x \in R$, denote any cumulative distribution function (cdf) (continuous or not). Let $F^{-1}(y)$, $y \in [0, 1]$ denote the inverse function defined in $F^{-1}(y) = \min\{x : F(x) \geq y\}, y \in [0,1]$ . Define X = $F^{-1}$ (U), where U has the continuous uniform distribution over the interval (0, 1). Then X is distributed as F, that is, $P(X \leq x) = F(x)$, $x \in IR$.

The inverse transform method can be used in practice as long as we are able to get an explicit formula for $F^{-1}(y)$ in closed form. We illustrate with some examples. We use the notation $U \sim unif(0,1)$ to denote that U is a random variable with the continuous uniform distribution over the interval (0, 1).

Exponential distribution: $F(x) = 1 - e^{-\lambda x}, x \geq 0$, where $\lambda > 0$ is a constant. Solving the equation $y = 1 - e^{-\lambda x}$ for x in terms of $y \in (0, 1)$ yields $x = F^{-1}(y) = -\frac{1}{\lambda}\ln(1-y)$. This yields $X = -\frac{1}{\lambda}\ln(1-U)$. But (as is easily checked) $1 - U \sim unif(0,1)$ since $U \sim unif(0,1)$, and thus we can simplify the algorithm by replacing 1 − U by U: Algorithm for generating an exponential random variable at rate λ:

i Generate $U \sim unif(0,1)$.

ii Set $X = -\frac{1}{\lambda}\ln(U)$

In this case, $\lambda = 5$. To Evaluate the quality of my generator with goodness of fit tests. Using the "chi2gof" in MATLAB, and setting the significant level 'alpha' 0.05. Executing once, it turns out:

```
H =

     0
```

,which means the hypothesis is accepted. If we try 1000 times of this simulation, and count the number of rejection cases, it turns out:

```
The number of rejection cases is 52
```

.

Also, we could use "kstest" to evaluate the quality of my generator, and it also turns out:

```
>> waiting2
The number of rejection cases is 47
```

The number of rejection cases is close to alpha times trial numbers, which means the rest of tests are accepted.

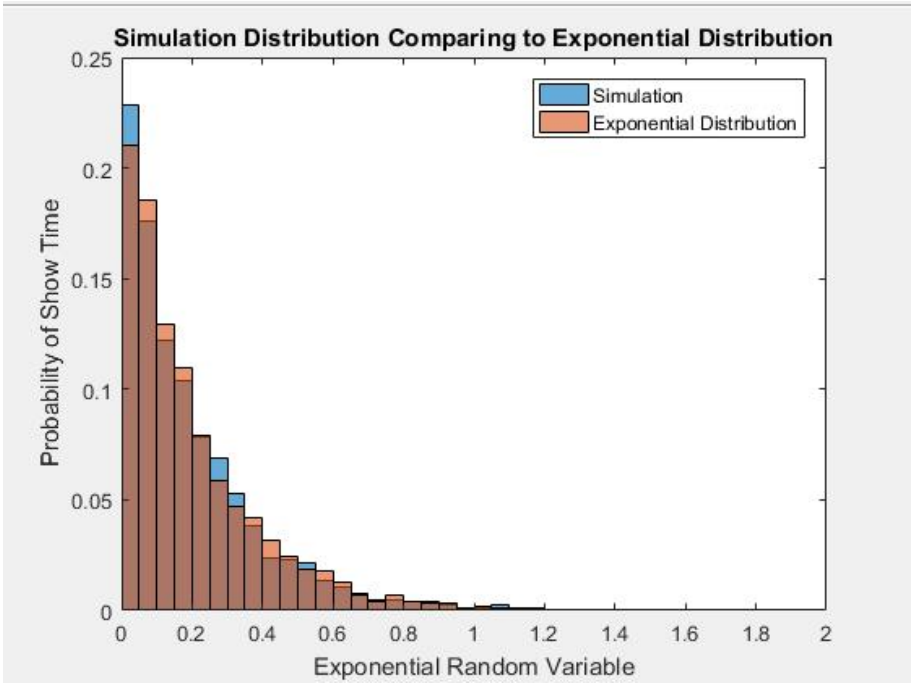


Figure 1. Histogram of Simulation Distribution and Exponential Distribution

**Problem 2. [Counting]**

We use while loop to calculate the number of events that occur in 1 unit of time.

```
for i = 1:1000
summ = 0; %The unit interval
time = 0; % The counter
    while summ < 1
        summ = summ + res(k);
        k = k + 1;
        if (k > 5000) %if exceed the boundary, start again.
            k = 1;
        end
        time = time + 1;
    end
    count(i) = time - 1; %Minus 1 because adding last one is larger than 1
end
```

Figure 2. While Loop Source Code

And we hist that count distribution, it looks like below.

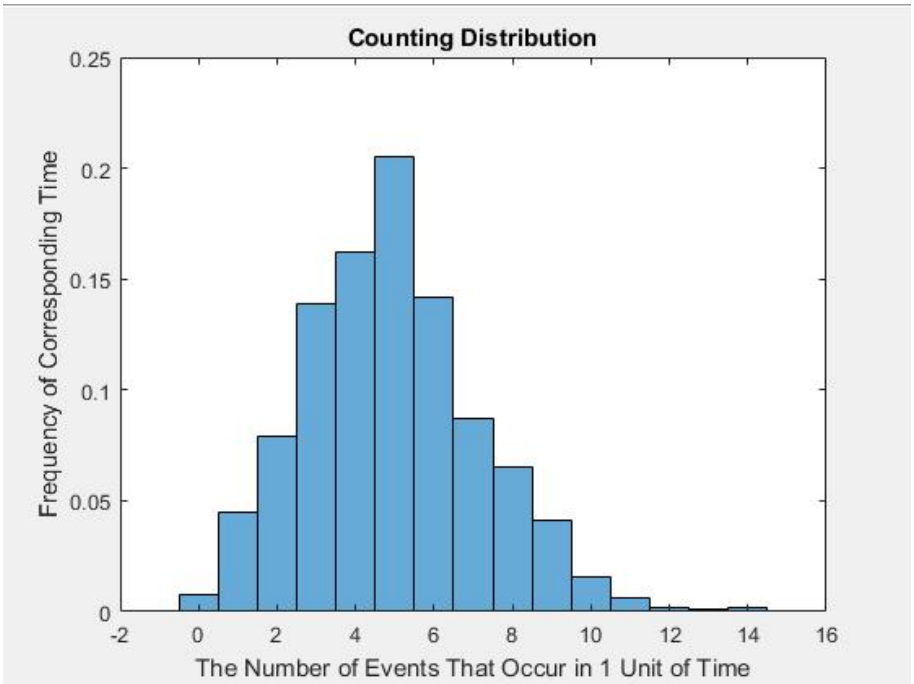


Figure 3. Counting Distribution

The counting distribution looks like Poisson Distribution, and we use MATLAB function 'chi2gof' to judge that. Due to the 'nparams' rules in the 'chi2gof' function, there are 2 methods using for judge the hypothesis shown in the figure 5.

```
If your 'cdf' or 'expected' input depends on estimated parameters, you
should use the 'nparams' parameter to ensure that the degrees of freedom
for the test is correct.  Otherwise the default 'nparams' value is

  'cdf' is a ProbabilityDistribution:  number of estimated parameters
  'cdf' is a function:                 0
  'cdf' is a cell array:               number of parameters in the array
  'expected' is specified:             0

The following options control other aspects of the test.
```

Figure 4. 'nparam' value rules

```
%Test for a Possion Distribution using chi2gof
%Method 1:
[N] = histcounts(count);
bins = min(count):max(count);
n = sum(N);
pd = fitdist(bins','Poisson','Frequency',N');
%lambdaHat = sum(bins.*N)/n;
%expCounts = n * poisspdf(bins,lambdaHat);
expCounts = n * pdf(pd,bins);
[H P STAT] = chi2gof(bins,'Ctrs',bins,...
        'Frequency',N,'Expected',expCounts,'nparam',0,'Alpha',0.05);
%Method 2
%X = count';
%[H P STATS] = chi2gof(X,'cdf',@(z)poisscdf(z,mean(X)),'Alpha',0.05,'nparam',1);
```

Figure 5. 'chi2gof' methods

And the result shows that the hypothesis is accepted. Finally plotting the 2 distribution into one graph.

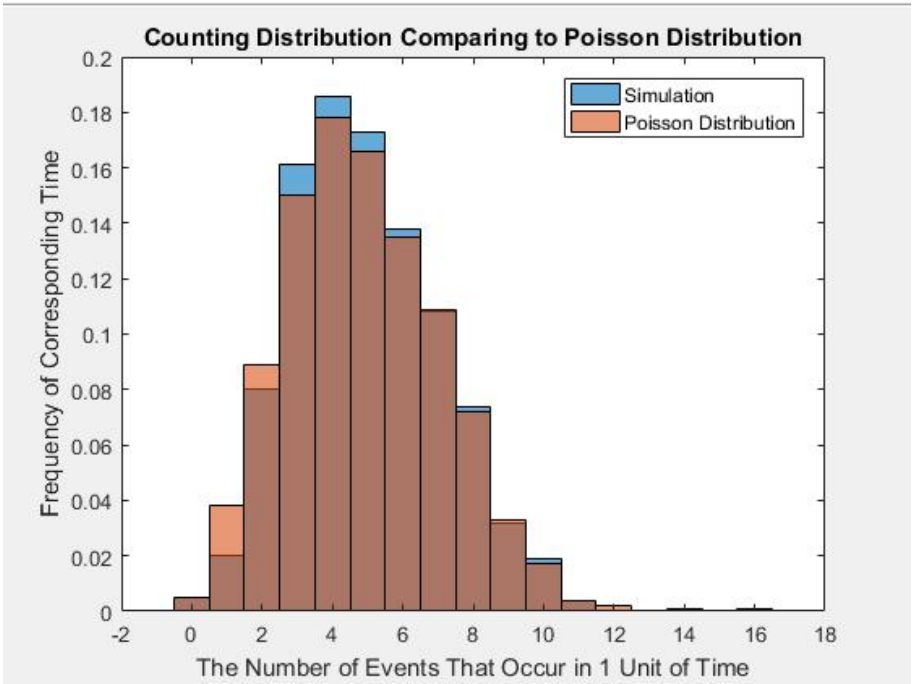


Figure 6. Counting Distribution Comparing to Poisson Distribution

**Problem 3 [Networking]**

Generate and plot sample networks for each value of p={0.02, 0.09}. When p equals to 0.02, the graph has not too many connections. Even some nodes are isolate. The whole network structure is sparse.

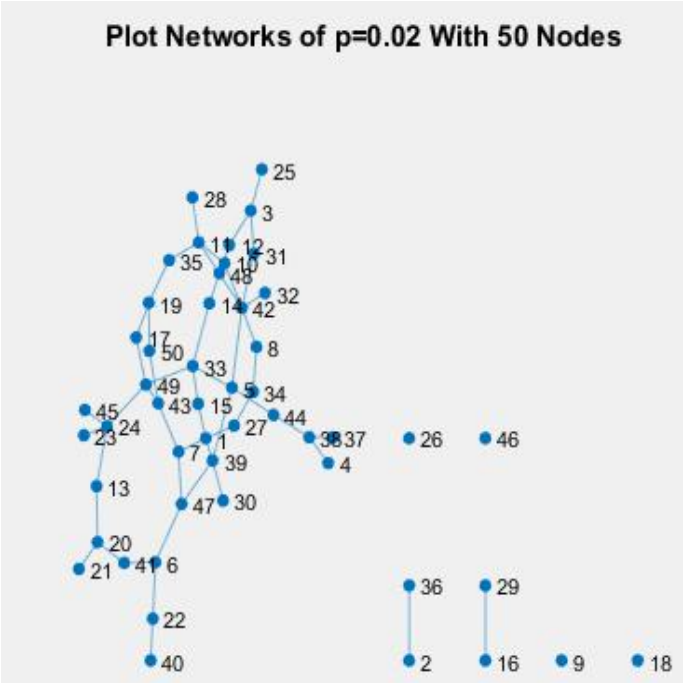


Figure 7. Network for p = 0.02

As for p equals to 0.09, which is large than threshold log(n)/n = 0.0782. Thus, the network structure is more compact and has more connections than p = 0.02.

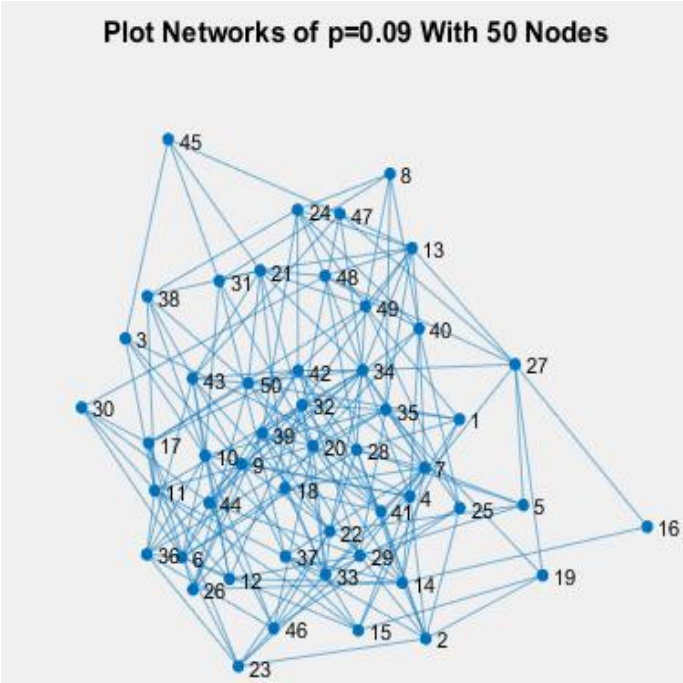


Figure 7. Network for p = 0.09

**Problem 4 [Networking at larger scale]**

When we get the connection of matrix in Problem 3, it is easy to count the degree of each node just counting the '1' in corresponding line.

Plot a histogram of vertex degrees for p = 0.02.

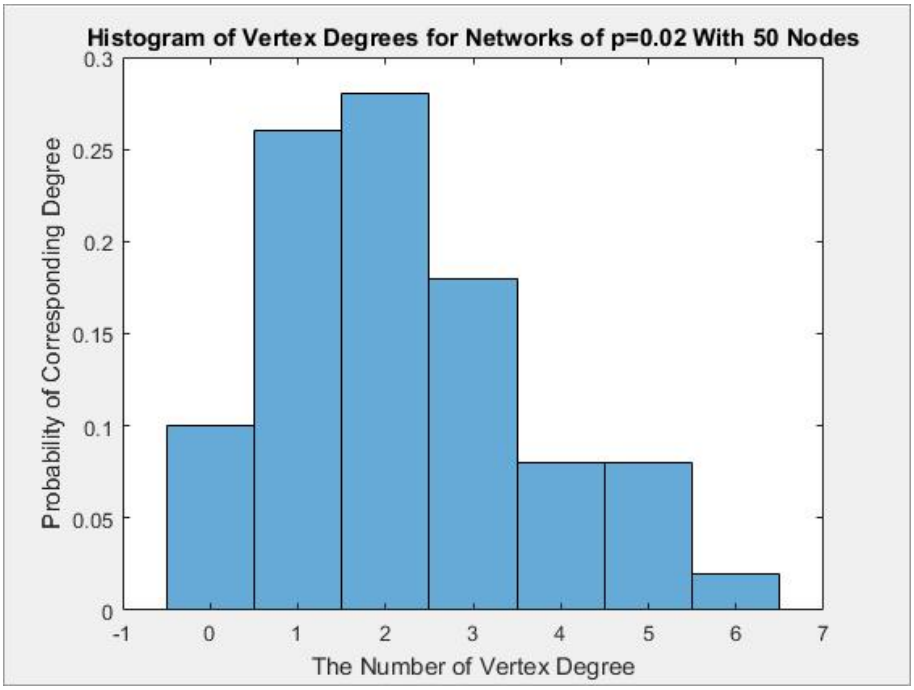


Figure 8. Histogram of Vertex Degrees for Networks of p = 0.02 With 50 Nodes

Plot a histogram of vertex degrees for p = 0.09.

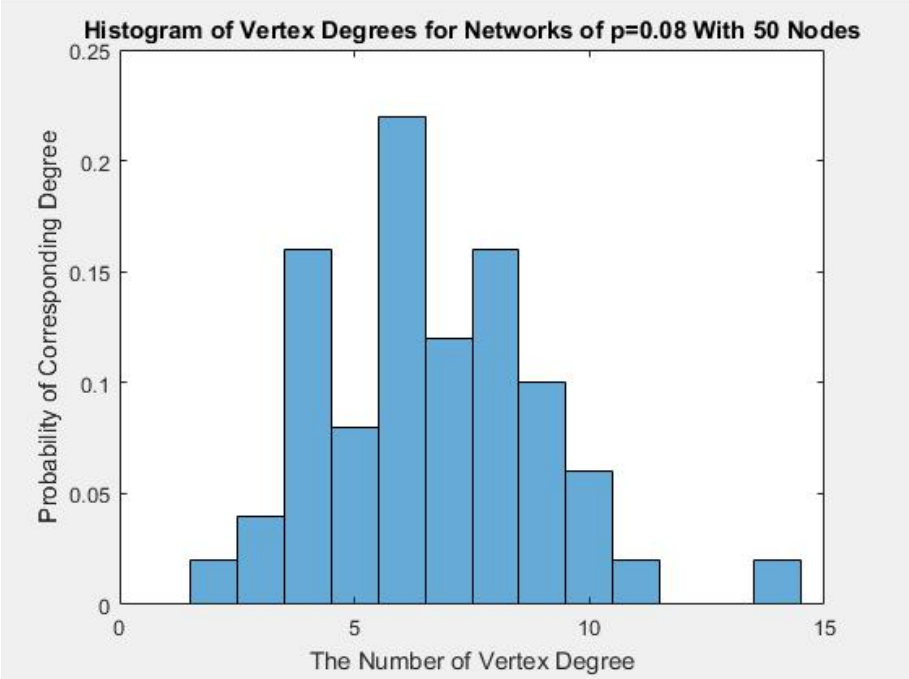


Figure 8. Histogram of Vertex Degrees for Networks of p = 0.09 With 50 Nodes

Generate a network with (n, p) = (250, 0.08) and plot the histogram of its vertex degrees.

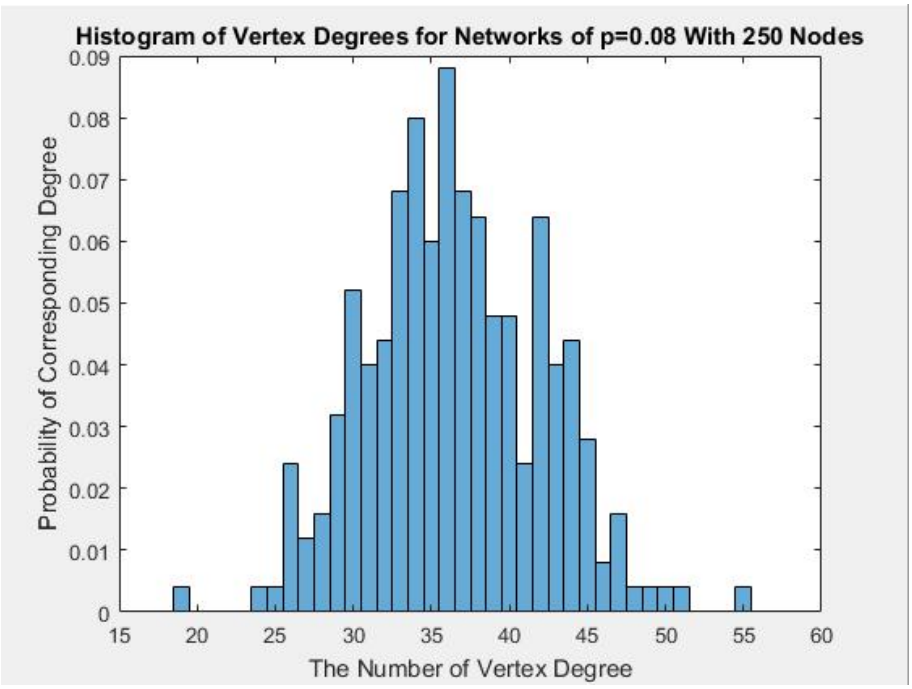


Figure 9. Histogram of Vertex Degrees for Networks of p = 0.08 With 250 Nodes

Vertex degree is a binomially distributed statistic for small networks. When n grows and p decrease, the distribution of Vertex degree is close to Poisson Distribution.

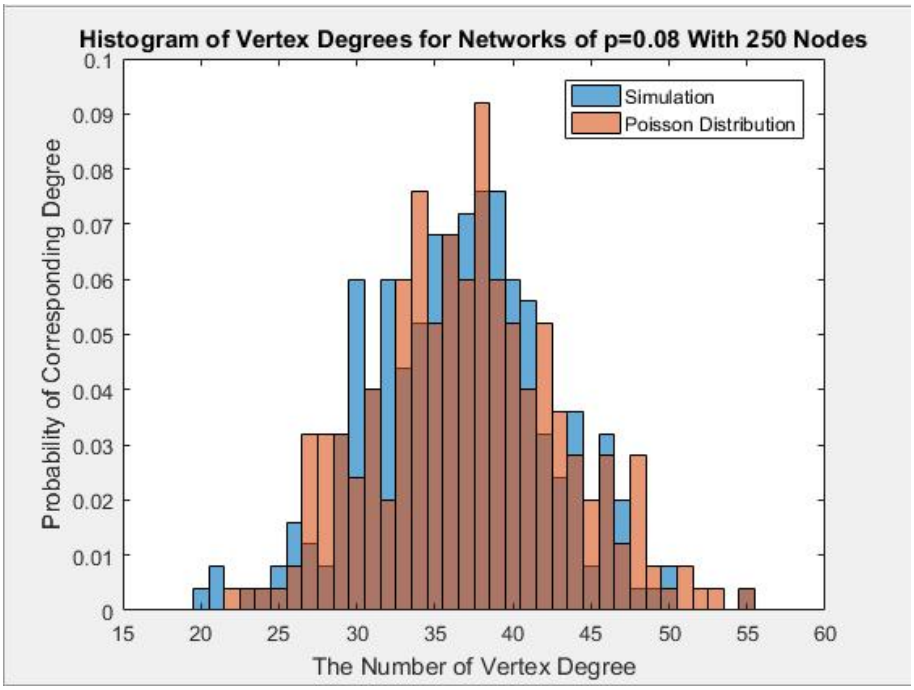


Figure 10. Histogram of Vertex Degrees for Networks of p = 0.08 With 250 Nodes Compared to Poisson Distribution

And the result shows the hypothesis is accepted. The Poisson distribution is actually a limiting case of a Binomial distribution when the number of trials, n, gets very large and p, the probability of success, is small.

Proof:

Consider a binomial process where the number of trials tends to infinity, and the probability of success at the same time tends to zero, with the constraint that the mean of the Binomial distribution *np* remains finitely large. The probability mass function of the Binomial distribution is:

$$p(x) = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x} \quad (1)$$

So, in the example above, x would be the number of bacteria I consume in n units of water, and p is the probability that a random unit of water contains a bacterium.

We'll replace p with the Poisson intensity $l$ = bacteria/ml, and the number of trials n with the amount of water consumed t ml. Note that $l$ and t must have matching units, so that: $\lambda t = np$

Putting into Equation 1. gives:

$$p(x) = \frac{n!}{x!(n-x)!} (\frac{\lambda t}{n})^x (1-\frac{\lambda t}{n})^{n-x} \quad (2) = \frac{n(n-1)...(n-x+1)}{n^x} \frac{(\lambda t)^x}{x!} \frac{(1-\frac{\lambda t}{n})^n}{(1-\frac{\lambda t}{n})^x}$$

For $n$ large and $p$ small:

$$(1-\frac{\lambda t}{n})^n \approx e^{-\lambda t}\ \frac{n(n-1)...(n-x+1)}{n^x} \approx 1*(1-\frac{\lambda t}{n})^x \approx 1$$

which simplifies Equation 2. to:

$$p(X=x) \approx \frac{e^{-\lambda t}(\lambda t)^x}{x!}$$

This is the probability mass function for the Poisson( $\lambda t$ ) distribution.

Number of events $a$ in time $t$ = Poisson( $\lambda t$ )

when the average number of events that will occur in a unit interval of exposure is known to be $\lambda$ .

## II. Conclusion

Evaluate the quality of the generator with the MATLAB function 'chi2gof', concerning the 'nparam' value corresponding to different situation. If the result is 0, then the hypothesis is accepted, otherwise rejected.
Poisson distribution is a discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time and/or space if these events occur with a known average rate and independently of the time since the last event. So the answer of problem2 is Poisson distribution.
At the time of generating network, when the number of node is fixed, the larger p, the more compact of network. And for the vertex degree of each node in the network, its histogram is closer to Poisson distribution due to the Poisson distribution is the limit to Binomial distribution when n closer to infinite and p closer to 0.

## III. Source Code

### 1. waiting.m

```
%Use the inverse CDF method to generate independent samples,
%Xi, of the exponential random variable with average waiting time of 0.2
time units.
%Evaluate the quality of your generator with goodness of fit tests.
clear; clc;
for sim = 1:1000
    X = -log(rand(3000,1))/5;%The CDF inverse exp distribution
    [H(sim) P(sim) STATS] =
chi2gof(X,'cdf',@(z)expcdf(z,mean(X)),'Alpha',0.05,'nparam',1);
end
```

```
str = sprintf('The number of rejection cases is %d',nnz(H));
disp(str)
```

### 2. waiting2.m

```
clear; clc;
for sim =1:1000
    X = -log(rand(100,1))/5;%The CDF inverse exp distribution
    test_cdf = makedist('exp', 'mu', 0.2);
    [h(sim), p(sim)] = kstest(X, 'CDF', test_cdf,'Alpha',0.05);
end
str = sprintf('The number of rejection cases is %d',nnz(h));
disp(str)
```

### 3. waiting_draw.m

```
X = -log(rand(3000,1))/5;
H1 = histogram(X,'Normalization','probability');

R = exprnd(0.2,[3000,1]);
hold on;
H2 = histogram(R,'Normalization','probability');
legend({'Simulation','Exponential Distribution'});
xlabel('Exponential Random Variable');
ylabel('Probability of Show Time');
title('Simulation Distribution Comparing to Exponential Distribution');
```

### 4. Counting.m

```
%Each exponential random sample represents the waiting time until an event
occurs.
%Implement a routine to count the number of events that occur in 1 unit
of time.
%Generate such counts for 1000 separate unit time intervals.
%How are these counts distributed? Justify your answer
clear; clc;

res = -log(rand(1,5000))/5; %The CDF inverse exp distribution
count = zeros(1,1000); %Record the number of events
% that occur in 1 unit of time for 1000 separate intervals.
k = 1;
for i = 1:1000
summ = 0; %The unit interval
time = 0; % The counter
    while summ < 1
        summ = summ + res(k);
        k = k + 1;
        if (k > 5000) %if exceed the boundary, start again.
```

```
            k = 1;
        end
        time = time + 1;
    end
    count(i) = time - 1; %Minus 1 because adding last one is larger than
1
end

%Hist the counting distribution
H1 =
histogram(count,'Normalization','probability','BinMethod','auto');
%{
xlabel('The Number of Events That Occur in 1 Unit of Time');
ylabel('Probability of Corresponding Time');
title('Counting Distribution');
%}
hold on;
%Compare it to the Poisson Distribution
R = poissrnd(5,[1,1000]);
H2 = histogram(R,'Normalization','probability','BinMethod','auto');

legend({'Simulation','Poisson Distribution'});
xlabel('The Number of Events That Occur in 1 Unit of Time');
ylabel('Probability of Corresponding Time');
title('Counting Distribution Comparing to Poisson Distribution');

%Test for a Possion Distribution using chi2gof
%Method 1:
[N] = histcounts(count);
bins = min(count):max(count);
n = sum(N);
pd = fitdist(bins','Poisson','Frequency',N');
%lambdaHat = sum(bins.*N)/n;
%expCounts = n * poisspdf(bins,lambdaHat);
expCounts = n * pdf(pd,bins);
[H P STAT] = chi2gof(bins,'Ctrs',bins,...
       'Frequency',N,'Expected',expCounts,'nparam',0,'Alpha',0.05);
%Method 2
%X = count';
%[H P STATS] =
chi2gof(X,'cdf',@(z)poisscdf(z,mean(X)),'Alpha',0.05,'nparam',1);
```

## 5. PlotNetwork.m

```
clear,clc;
```

```
%function edge = PlotNetwork(p,n)
p = 0.09;
n = 50;
matrix = Plot(p,n);

function edge = Plot(p,n)
    %Create a random edge matrix with n nodes
    %simulate bernoulli trail
    edge=random('Binomial',1,p,n,n);
    edge(edge==0)=-1;
    edge=abs(edge'+edge);
    edge(edge==0)=1;
    figure;
    G=graph(edge==1);
    plot(G);
    axis off;
    str = sprintf('Plot Networks of p=%.2f With %d Nodes',p,n);
    title(str);
end
```

### 6. largeScaleNetwork.m

```
clear,clc;

p = 0.08;% Change the P value
n = 250; % Change the Node number
A = VD(p,n);
H1 = histogram(A,'Normalization','probability');

%Compare to poisson distribution
hold on;
pd = fitdist(A','poisson');
R = poissrnd(pd.lambda,[1,length(A)]);
H2 = histogram(R,'Normalization','probability');
legend({'Simulation','Poisson Distribution'});

xlabel('The Number of Vertex Degree');
ylabel('Probability of Corresponding Degree');
str = sprintf('Histogram of Vertex Degrees for Networks of p=%.2f With %d
Nodes',p,n);
title(str);


[N] = histcounts(A);
bins = min(A):max(A);
```

```
n = sum(N);
pd = fitdist(bins','Poisson','Frequency',N');
expCounts = n * pdf(pd,bins);
[H P STAT] = chi2gof(bins,'Ctrs',bins,...
       'Frequency',N,'Expected',expCounts,'nparam',0,'Alpha',0.05);

function VertexDegree = VD(p,n)
    edge=random('Binomial',1,p,n,n);
    edge(edge==0)=-1;
    edge=abs(edge'+edge);
    edge(edge==0)=1;
    edge(edge~=1)=0;
    VertexDegree = zeros(1,n);
    for i = 1:n
        VertexDegree(i)=nnz(edge(i,:));
    end
end
```