University of Southern California EE511

Markov chains and discrete events

Project #5

Name: Wu Jiawei
USCID: 9600392575

Abstract

In the project of Markov chains and discrete events, three experiments are conducted using Matlab. The core method of the project is to simulate the non-homogeneous Poisson process according to the certain distribution and simulate the process of Markov of chain with the transition matrix. The theories of discrete time stochastic processes, Markov chain system, HOL blocking switch and are applied comprehensively in the lab. The experiments are repeated multiple times and the outcomes are shown in diagrams and calculations.

Introduction

Three experiments are conducted in the lab. All the samples are generated by random selection and the experiments are repeated multiple times. The goal of the first trial is to follow the single server queue model to finish discrete event simulation and obtain the total break time. The aim of the second experiment is to find the mean of packets quantity in the buffer or switching process of a N*N HOL blocking switch. The objective of the third experiment is running simulation to choose different initial allele distribution and find the steady-state genetic composition according to the Markov chain theory.

Methodology & Results

Experiment No.1

Description of Algorithm:

1.The theory of nonhomogeneous Poisson process is utilized in this experiment.

Suppose that $\lambda(t)$ is the arrival function for a non-homogenous Poisson process, and T_s is the time of the first arrival after time s.

Let λ be such that $\lambda(t) \leq \lambda$ for all t.

1). Let $t = s$

2). Generate $U_1 \sim U[0,1]$

3). Let $t = t - \frac{1}{\lambda} * \log U_1$

4). Generate $U_2 \sim U[0,1]$

5). If $U_2 \leq \frac{\lambda(t)}{\lambda}$, set $T_s = t$ and stop

6). Goto step 2.

2.The service time follows the exponential distribution

$T\_service \sim Exp(25)$

3.The break time follows the uniformly distribution $T\_break \sim Uniform(0, 0.3)$

Description of Method:

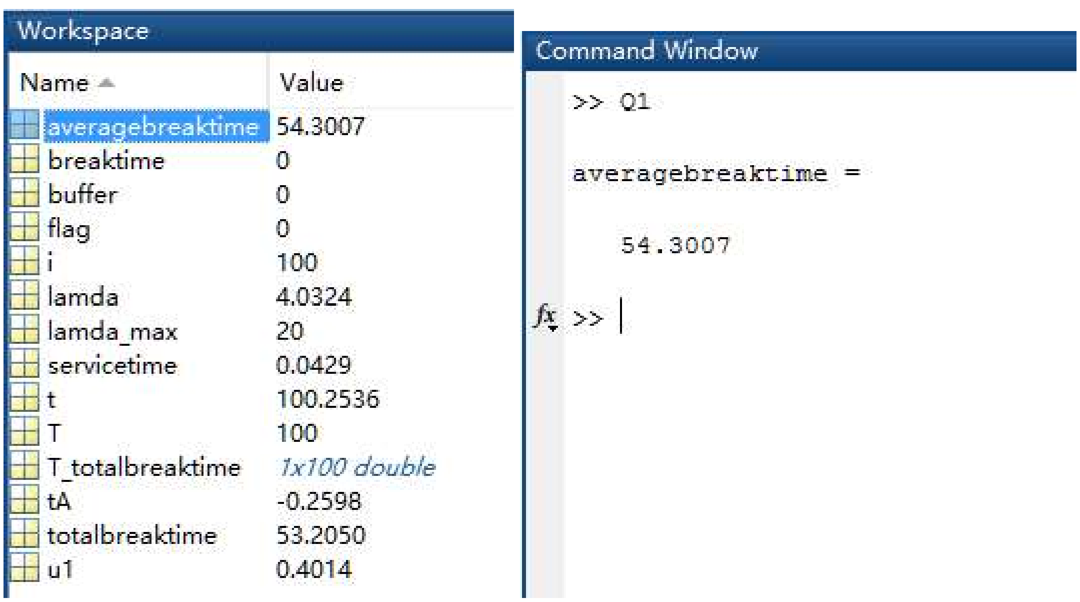
The simulation starts at the time when the first job arrives. The exponential distribution is utilized to estimate when the job will finish. Then the nonhomogeneous Poisson process is simulated to compute the time that next job will arrive. If the next job arrives after the current one finished, the server can take a break. Different cases of single server queueing system are discussed in the experiment based on the relationship between the arrival time and departure time of jobs. The trials are repeated multiple times and the amount of break time of the server is calculated to finish the simulation.

Code:

```
T_totalbreaktime=zeros(1,100);
for i=1:100
    t=0;
    T=100;
    lamda_max=20;%Choose lamda such that lamda(t)<lamda for all t
    buffer=0;
    tA=0;
    breaktime=0;
    totalbreaktime=0;
    servicetime=0;

    while t<T
        if breaktime == 0;
            breaktime=0.3*rand(1);
        end
        if servicetime==0
            servicetime=exprnd(1/25);
        end%Choose breaktime and servicetime by different distributions
        while flag==1
            u1=rand;
            tA=tA-(1/lamda_max)*log(u1);
            if mod(t,10)<5
                lamda=4+3*mod(t,10);
            elseif mod(t,10)>=5
                lamda=34-3*mod(t,10);
            end
            if rand()<=lamda/lamda_max
                flag=0;
            end
        end

        if tA <= 0
            buffer=buffer+1;
            flag=1;
        elseif buffer==0
            t=t+breaktime;
            totalbreaktime = totalbreaktime + breaktime;
            tA=tA-breaktime;
            breaktime=0;
        else
            if servicetime > tA
                buffer=buffer+1;
                t=t+tA;
```

```
                servicetime=servicetime-tA;
                tA=0;
                flag=1;
            elseif servicetime < tA
                buffer=buffer-1;
                t=t+servicetime;
                tA=tA-servicetime;
                servicetime=0;
            else
                t=t+tA;
                tA=0;
                servicetime=0;
                flag=1;%Different cases of single server queueing system
            end
        end
    end
    T_totalbreaktime(i)=totalbreaktime;%Record the total amount of breaktime
end
averagebreaktime=mean(T_totalbreaktime)
```

Simulation Result:



Finding:

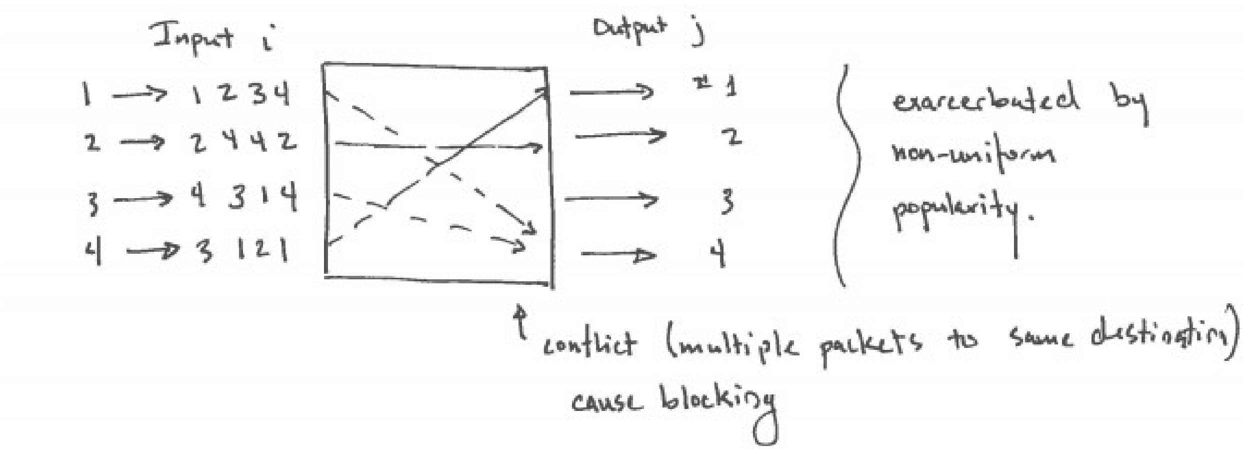
The experiment is repeated 100 times to obtain the result 54.3007.

In the first 100 hours of operation, the expected amount to time that the server is on break is nearly 54.3007 hours.

Experiment No.2

Description of Algorithm:

In the heavy loaded Head of line blocking switch, only one packet delivered at a time to a particular output and every input always has a packet for transmission.



The system state is (x_1, x_2) , $x_i \in \{0,1\}$

The possible states (0,0) and (1,1) allow only a single packet to transmit.

The possible states (0,1) and (1,0) allow both packets to transmit.

Description of Method:

The aim of simulation is to find the distribution of mean of number of packets in the input1 and input2 of the buffer as a function of arrival probability. Two cases are taken into consideration.

Case 1: r_{ij}=0.5

Case 2: r_1=0.75 , : r_2=0.25

The experiment generates 100000 samples and the probability (from 0 to 1) is divided into 100 parts to represent the arrival probability. In the simulation of the working process of 2*2 HOL blocking switch, the conditions are set to determine which input the packets come in and which output the packets departure. When the packets are same the condition is set to determine which packet to process first. The mean of switch efficiency is computed and the 95% confidence interval for the overall efficiency of the switch is estimated.

Code:

```
function f=ho(N,p)
arr_buffer1=zeros(1,101);
arr_buffer2=zeros(1,101);
arr_process=zeros(1,101);
arr_efficiency=zeros(1,101);
j=0;
for pi=0:0.01:1
    j = j+1;
    n1=0;
    n2=0;
    temp1=0;
    temp2=0;
    buffer1=zeros(1,N);
    buffer2=zeros(1,N);
    process=zeros(1,N);
```

```
    for i=1:N
        P=rand();
        if P<pi
            n1=n1+1;
        end

        P1=rand();
        if P1<pi
            n2=n2+1;
        end
 %Determine the packets going to the input1 or input2 of the buffers
        if n1>0 && temp1==0
            r1=rand();
            if r1<p
                temp1=1;
            else
                temp1=2;
            end
        end
 %Deternime the output of packets from input 1
        if n2>0 && temp2==0
            r2=rand();
            if r2<p
                temp2=1;
            else
                temp2=2;
            end
        end
%Determine the output of packets from input 2
        if temp1>0 && temp2>0
            if temp1==temp2
                rnd=rand();
                if rnd<0.5
                    temp1=0;
                    n1=n1-1;
                else
                    temp2=0;
                    n2=n2-1;
                end
                process(i)=1;
            else
                temp1=0;
                temp2=0;
                n1=n1-1;
                n2=n2-1;
                process(i)=2;
            end
        elseif temp1 == 0 && temp2 > 0
            n2=n2-1;
            temp2 = 0;
            process(i)=1;
        elseif temp2 == 0 && temp1 > 0
            n1=n1-1;
            temp1 = 0;
            process(i)=1;
        else % temp1 == 0 && temp2 ==0
        end
```

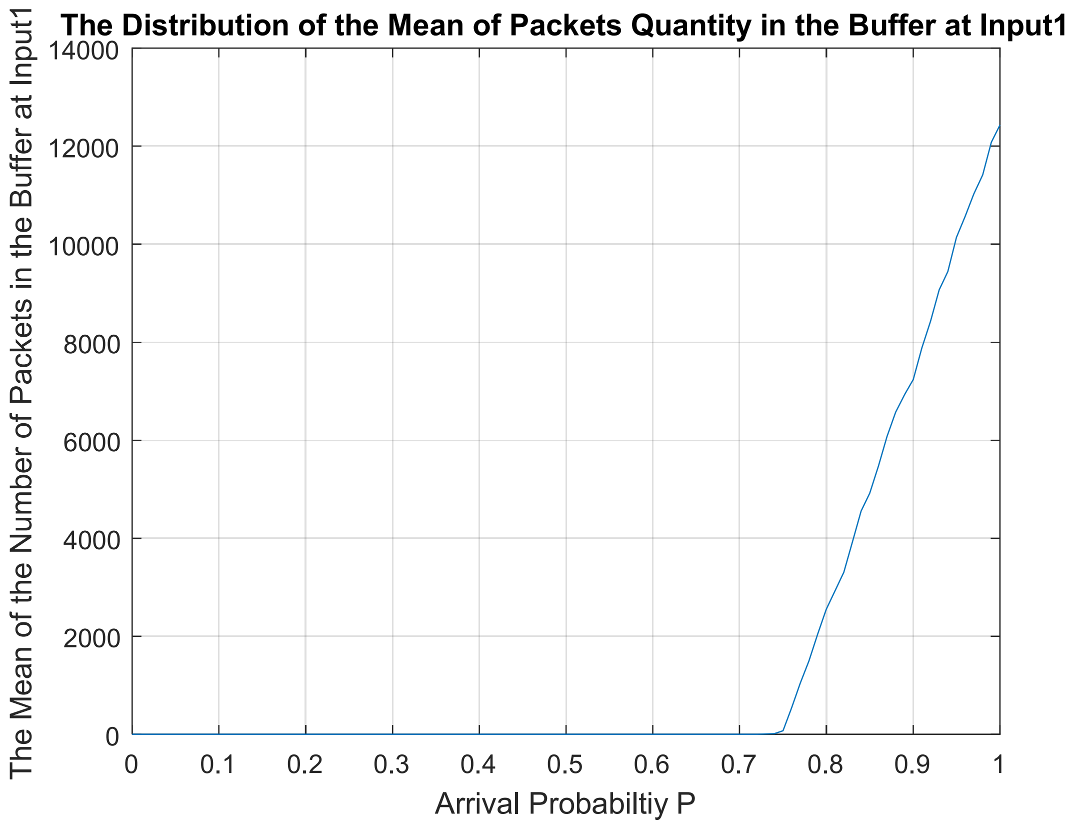
```
 %Determine the number of packets in process
        buffer1(i)=n1;
        buffer2(i) = n2;
    end
    arr_buffer1(j)=mean(buffer1);
    arr_buffer2(j)=mean(buffer2);
    arr_process(j)=mean(process);
    arr_efficiency(j)=sum(process)/(2*N);
end
sorted_efficiency=sort(arr_efficiency);
Mean_efficiency=mean(sorted_efficiency)
Std=std(sorted_efficiency);
z=norminv(0.975,0,1);
lowervalue_efficiency=Mean_efficiency-(Std/sqrt(101))*z
uppervalue_efficiency=Mean_efficiency+(Std/sqrt(101))*z
width_efficiency=uppervalue_efficiency-lowervalue_efficiency
figure;
plot(0:0.01:1,arr_buffer1);
title('The Distribution of the Mean of Packets Quantity in the Buffer at
Input1');
xlabel('Arrival Probabiltiy P');
ylabel('The Mean of the Number of Packets in the Buffer at Input1');
grid on;
figure;
plot(0:0.01:1,arr_buffer2);
title('The Distribution of the Mean of Packets Quantity in the Buffer at
Input2');
xlabel('Arrival Probability P');
ylabel('The Mean of the Number of Packets in the Buffer at Input2');
grid on;
figure;
plot(0:0.01:1,arr_process);
title('The Distribution of the Mean of Packets Quantity in the Process');
xlabel('Arrival Probability P');
ylabel('The Mean of the Number of Packets in Process');
grid on;
```

Simulation Result:

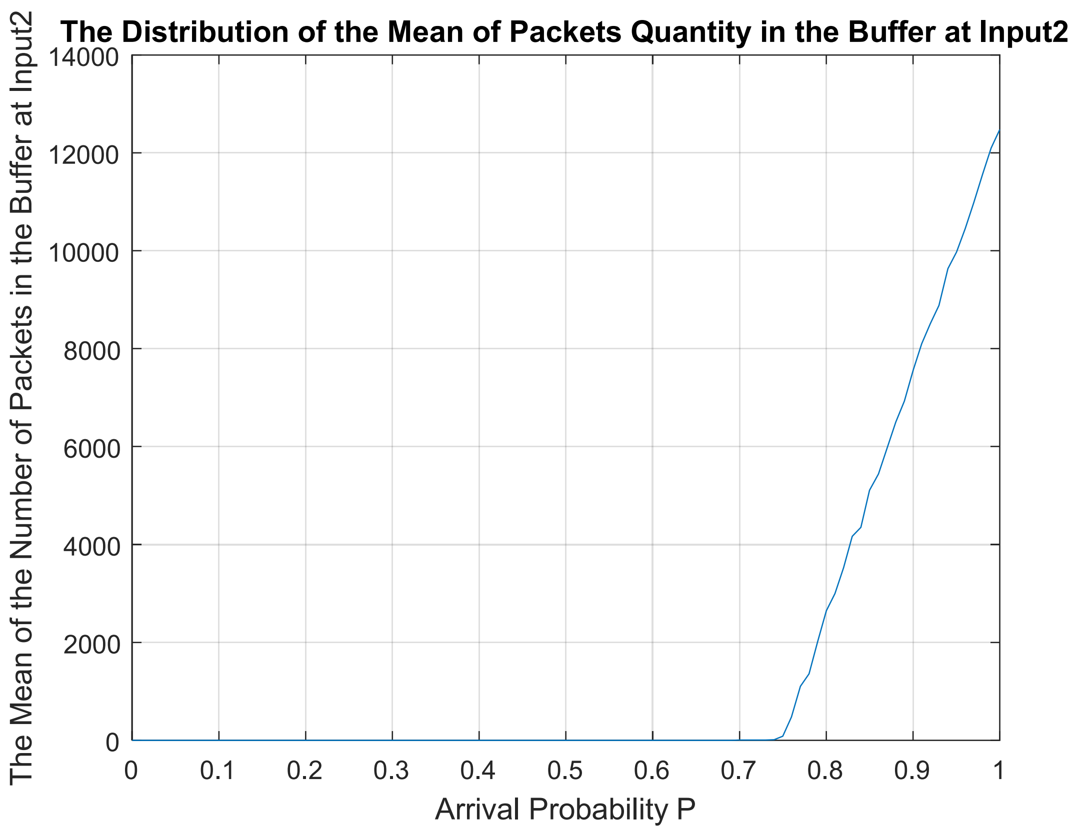
1). >> ho(100000,0.5)

Command Window

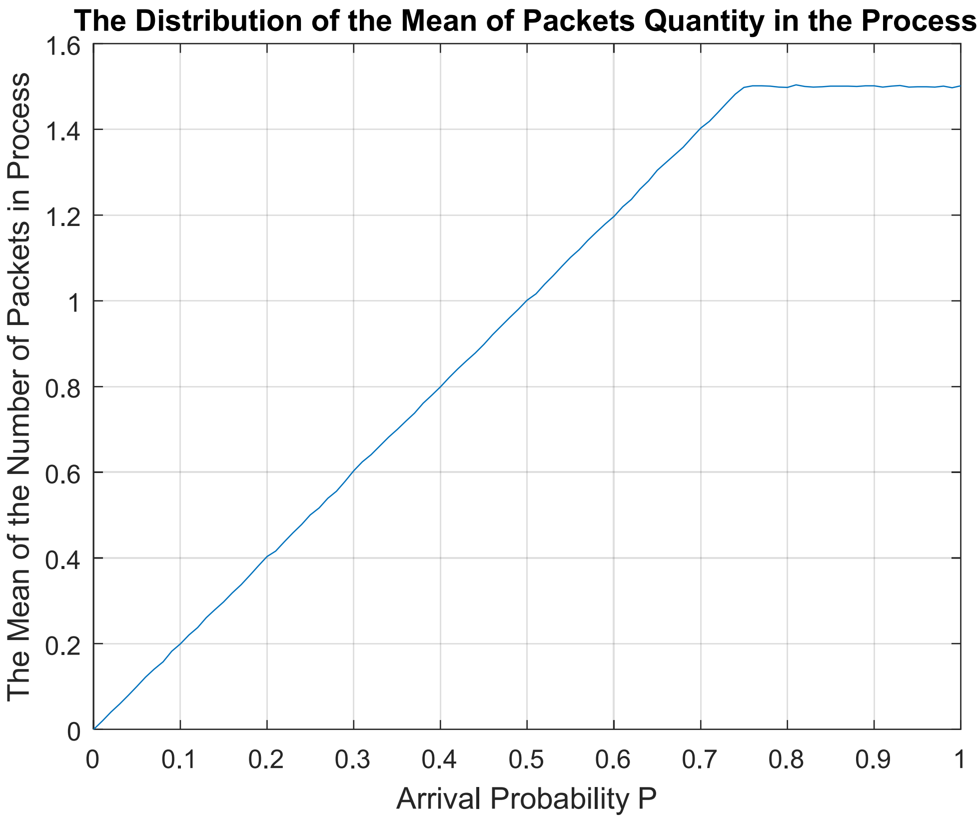
```
>> ho(100000,0.5)

Mean_efficiency =

    0.4679


lowervalue_efficiency =

    0.4189


uppervalue_efficiency =

    0.5169


width_efficiency =

    0.0980
```



The Distribution of the Mean of Packets Quantity in the Buffer at Input1
The Mean of the Number of Packets in the Buffer at Input1
Arrival Probabiltiy P
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
2000
4000
6000
8000
10000
12000
14000



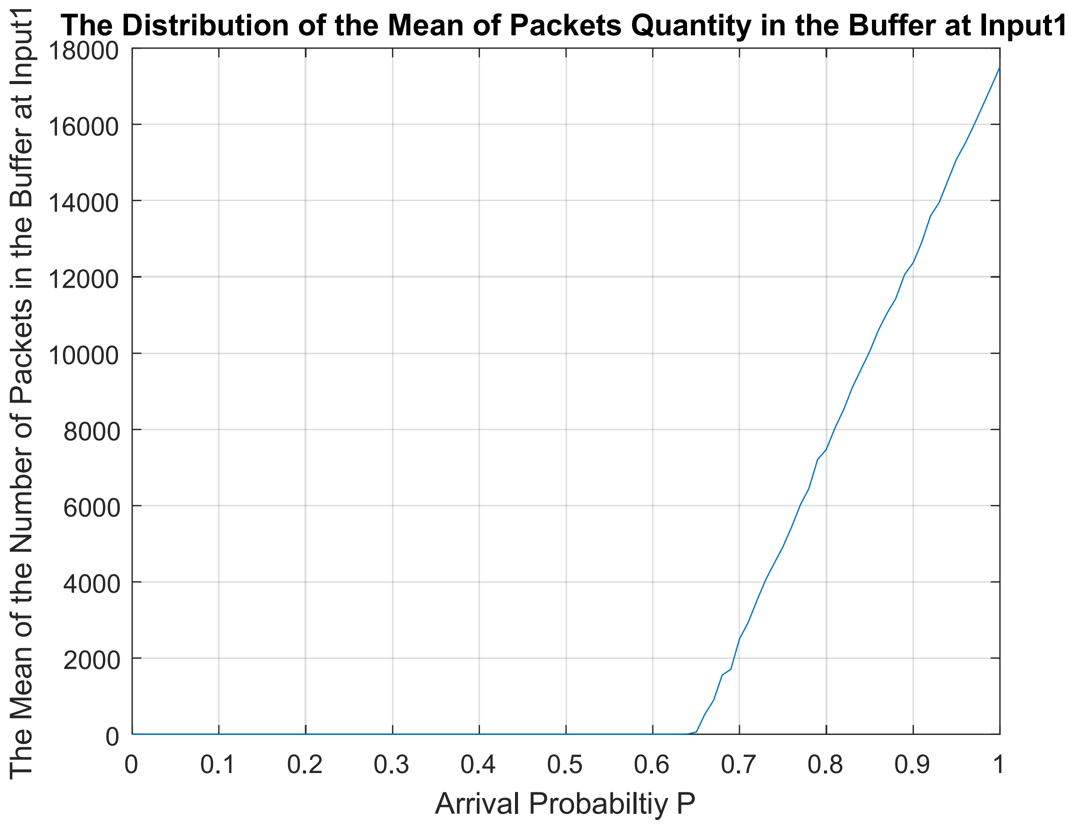
The Distribution of the Mean of Packets Quantity in the Buffer at Input2
The Mean of the Number of Packets in the Buffer at Input2
Arrival Probability P
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
2000
4000
6000
8000
10000
12000
14000



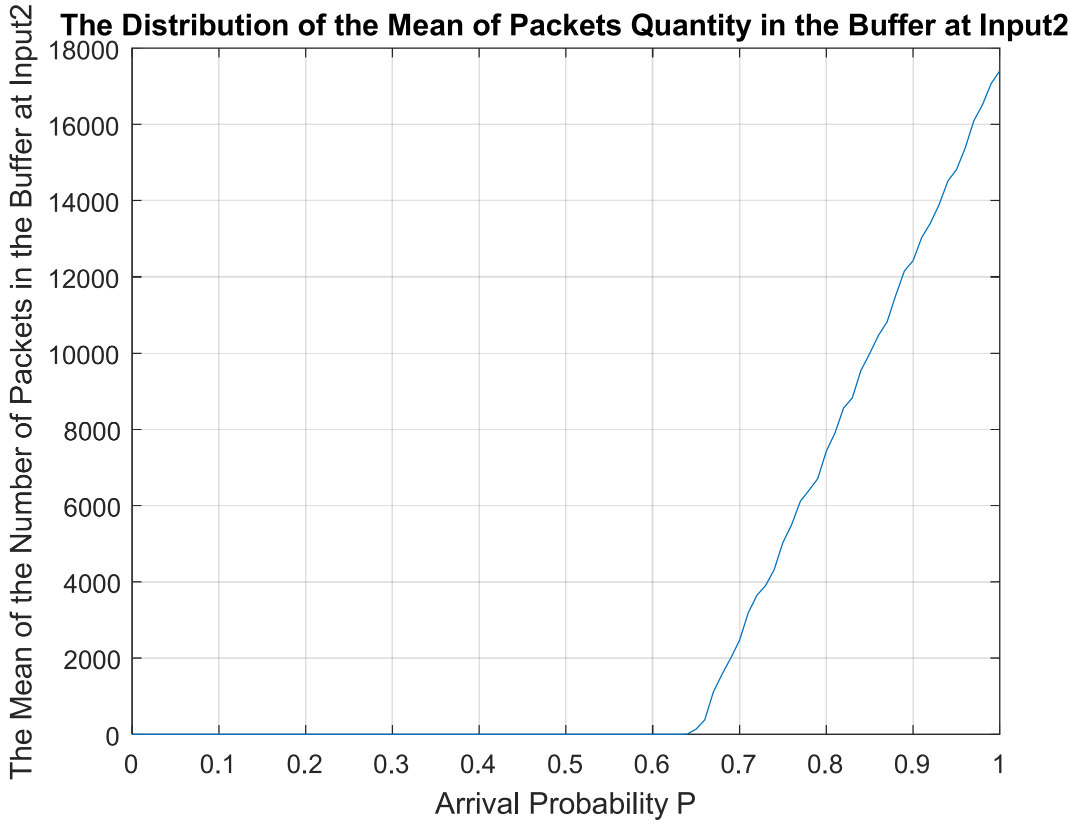
2). >> ho(100000,0.75)

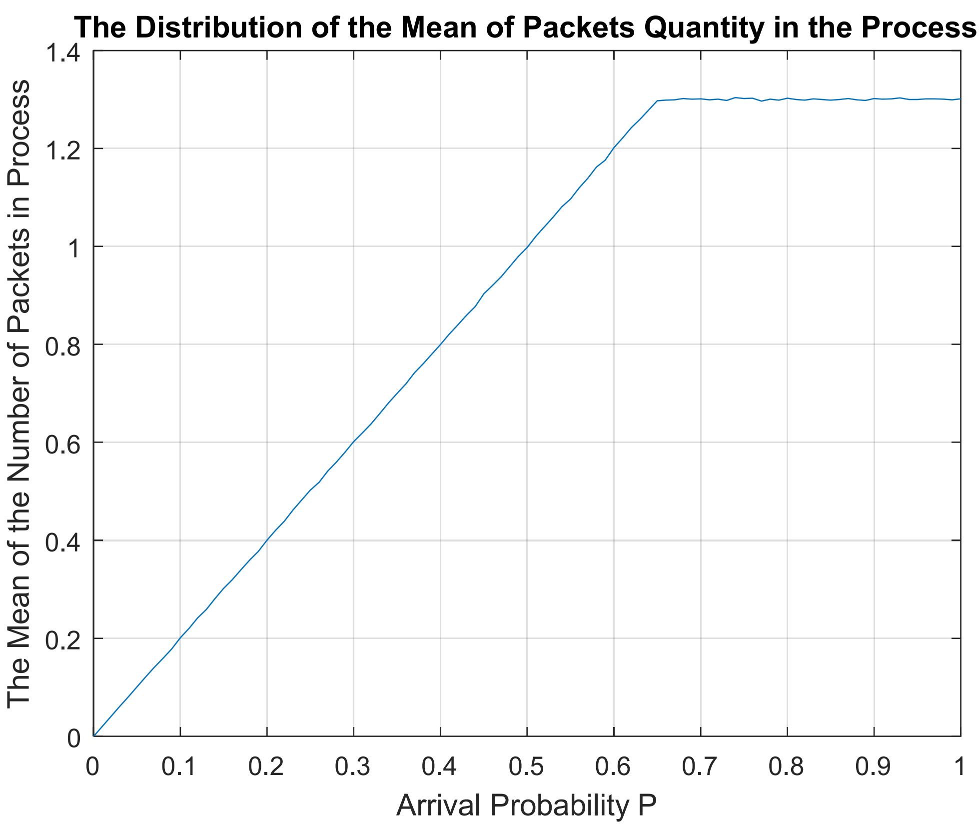
Command Window

```
>> ho(100000,0.75)

Mean_efficiency =

    0.4377


lowervalue_efficiency =

    0.3949


uppervalue_efficiency =

    0.4805


width_efficiency =

    0.0856
```



The Distribution of the Mean of Packets Quantity in the Buffer at Input1
The Mean of the Number of Packets in the Buffer at Input1
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Arrival Probabiltiy P


The Distribution of the Mean of Packets Quantity in the Buffer at Input2
The Mean of the Number of Packets in the Buffer at Input2
Arrival Probability P
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


The Distribution of the Mean of Packets Quantity in the Process
The Mean of the Number of Packets in Process
Arrival Probability P
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Finding:

1). For $r_{ij} = 0.5$,

When the arrival probability is less than 0.75, the mean of the number of packets at input1 and input2 of the buffer is 0. The mean of the number of packets in process is increasing from 0 to1.5. The HOL switch does not have a blocking problem when arrival probability is less than 0.75.

When the arrival probability is more than 0.75, the mean of the number of packets at input1 and input2 of the buffer start increasing. The mean of the number of packets in process remains 1.5.

The 95% confidence interval for the overall efficiency of the switch is [0.4189, 0.5169], the width of the confidence interval is 0.098

2). For $r_1 = 0.75$ and $r_2 = 0.25$,

When the arrival probability is less than 0.65, the mean of the number of packets at input1 and input2 of the buffer is 0. The mean of the number of packets in process is increasing from 0 to1.3, The HOL switch does not have a blocking problem when arrival probability is less than 0.65

When the arrival probability is more than 0.65, the mean of the number of packets at input1 and input2 of the buffer start increasing. The mean of the number of packets in process remains 1.3

The 95% confidence interval for the overall efficiency of the switch is [0.3949, 0.4805], the width of the confidence interval is 0.0856.

3). The value of r_{ij} influence the number of packets in the input 1 and input 2 of the buffer and influence the efficiency of the switch. As is shown is the result, the overall efficiency decreases when r_1 and r_2 are different.

Experiment No.3

Description of Algorithm:

The Markov Chain is a system whose next state depends only on its current state.

P(Xn = xn|Xn-1 = xn-1, Xn-2 = xn-2,..., X0=x0) = P(Xn = xn|Xn-1 = xn-1).

The transition matrix for a markov chain is an arrangement of the individual $P_r[X(t+1) = j|X(t) = i_r]$ values such that

$$P(t) = \{P_{ij}(t)\} = P_r[X(t+1) = j|X(t) = i_t]$$

Often P does not depend on time so P(t)=P.

The Markov Chain Ergodic theorem, for every nonnegative vector $x \in R^N$ then if P satisfies the requirement for Perron-Frobenius.

$$\lim_{n \to \infty} x * P^n = \pi$$

The Perron-Frobenius, Let P be a regular stochastic matrix. Suppose P is irreducible and aperiodic. Then exists a unique positive π with $\pi_i > 0$ for $1 \le i \le N$

$\pi P = \pi$

A Markov of chain is irreducible if and only if $P_{ij}^{n} > 0$ for all I and j and some $n \geq 1$, which means every state communicates with every other state.

Description of Method:

The experiment utilizes the Markov of chain theory to simulate the stochastic genotypic drift during successive generations in the Wright-Fisher model. The simulation first generates N pairs of parents with 2N copies of genes since diploid individuals has 2 copies of the genes. Then each pair produces a single offspring with its genotype inherited by selecting one from each parent. The density evolves according to a binomial density.

$$P[x(t+1) = j | x(t) = i] = \mathrm{Bin}\left(j, 2N, \frac{i}{2N}\right)$$

The Markov of chain transition matrix is utilized as

$$P_{i,j} = \binom{2N}{j}\left(\frac{i}{2N}\right)^{j}\left(1 - \frac{i}{2N}\right)^{2N-j}$$

The lab is repeated 100 times and commented on the steady-state genetic composition. Then the experiment is repeated larger times with different initial allele distributions.

Repeat time n=100; Number of A1=A2=100

Code:

```
N=101;% X(100)=P[100 copies of A1, 100 copies of A2]
input=zeros(1,201);
input(N)=1;%Set the initial allele distribution
N = 100;%Set N=100 diploid heterozygous individuals

% transition matrix
P=zeros(2*N+1,2*N+1);
for i = 1:2*N+1
    for j = 1:2*N+1
        P(i,j) = nchoosek(2*N,j-1)*((i-1)/(2*N))^(j-1)*(1-(i-1)/(2*N))^(2*N-
j+1);
    end
end
n=100; % number of time steps to take
output=zeros(n+1,2*N+1); % clear out any old values

output(1,:)=input; % generate first output value
for i=1:n,
    output(i+1,:) = output(i,:)*P;
    %a tolerance check to  automatically stop the simulation when the density
is close to its steady-state
    LIT = ismembertol(output(i+1,:),output(i,:));
    if all(LIT == 1)
        break;
```

```
    end
end
```

Simulation results:

output

101x201 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
85	0.0450	0.0025	0.0028	0.0029	0.0030	0.0031	0.0031	0.0032	0.0032	0.0033	0.0033	0.0034	0.0034	0.00
86	0.0465	0.0025	0.0028	0.0029	0.0030	0.0031	0.0031	0.0032	0.0032	0.0033	0.0033	0.0034	0.0034	0.00
87	0.0480	0.0025	0.0028	0.0030	0.0030	0.0031	0.0032	0.0032	0.0033	0.0033	0.0034	0.0034	0.0035	0.00
88	0.0496	0.0025	0.0029	0.0030	0.0031	0.0031	0.0032	0.0032	0.0033	0.0033	0.0034	0.0034	0.0035	0.00
89	0.0511	0.0025	0.0029	0.0030	0.0031	0.0032	0.0032	0.0033	0.0033	0.0034	0.0034	0.0034	0.0035	0.00
90	0.0527	0.0026	0.0029	0.0030	0.0031	0.0032	0.0032	0.0033	0.0033	0.0034	0.0034	0.0035	0.0035	0.00
91	0.0542	0.0026	0.0029	0.0031	0.0031	0.0032	0.0032	0.0033	0.0033	0.0034	0.0034	0.0035	0.0035	0.00
92	0.0558	0.0026	0.0029	0.0031	0.0031	0.0032	0.0033	0.0033	0.0034	0.0034	0.0034	0.0035	0.0035	0.00
93	0.0574	0.0026	0.0030	0.0031	0.0032	0.0032	0.0033	0.0033	0.0034	0.0034	0.0035	0.0035	0.0035	0.00
94	0.0590	0.0026	0.0030	0.0031	0.0032	0.0032	0.0033	0.0033	0.0034	0.0034	0.0035	0.0035	0.0035	0.00
95	0.0606	0.0027	0.0030	0.0031	0.0032	0.0033	0.0033	0.0034	0.0034	0.0034	0.0035	0.0035	0.0035	0.00
96	0.0622	0.0027	0.0030	0.0031	0.0032	0.0033	0.0033	0.0034	0.0034	0.0034	0.0035	0.0035	0.0036	0.00
97	0.0638	0.0027	0.0030	0.0032	0.0032	0.0033	0.0033	0.0034	0.0034	0.0035	0.0035	0.0035	0.0036	0.00
98	0.0655	0.0027	0.0030	0.0032	0.0032	0.0033	0.0033	0.0034	0.0034	0.0035	0.0035	0.0035	0.0036	0.00
99	0.0671	0.0027	0.0031	0.0032	0.0033	0.0033	0.0034	0.0034	0.0034	0.0035	0.0035	0.0035	0.0036	0.00
100	0.0688	0.0027	0.0031	0.0032	0.0033	0.0033	0.0034	0.0034	0.0034	0.0035	0.0035	0.0036	0.0036	0.00
101	0.0704	0.0027	0.0031	0.0032	0.0033	0.0033	0.0034	0.0034	0.0035	0.0035	0.0035	0.0036	0.0036	0.00
102														

output

101x201 double

	188	189	190	191	192	193	194	195	196	197	198	199	200	201
85	0.0035	0.0034	0.0034	0.0033	0.0033	0.0032	0.0032	0.0031	0.0031	0.0030	0.0029	0.0028	0.0025	0.0450
86	0.0035	0.0034	0.0034	0.0033	0.0033	0.0032	0.0032	0.0031	0.0031	0.0030	0.0029	0.0028	0.0025	0.0465
87	0.0035	0.0035	0.0034	0.0034	0.0033	0.0033	0.0032	0.0032	0.0031	0.0030	0.0030	0.0028	0.0025	0.0480
88	0.0035	0.0035	0.0034	0.0034	0.0033	0.0033	0.0032	0.0032	0.0031	0.0031	0.0030	0.0029	0.0025	0.0496
89	0.0035	0.0035	0.0034	0.0034	0.0034	0.0033	0.0033	0.0032	0.0032	0.0031	0.0030	0.0029	0.0025	0.0511
90	0.0035	0.0035	0.0035	0.0034	0.0034	0.0033	0.0033	0.0032	0.0032	0.0031	0.0030	0.0029	0.0026	0.0527
91	0.0035	0.0035	0.0035	0.0034	0.0034	0.0033	0.0033	0.0032	0.0032	0.0031	0.0031	0.0029	0.0026	0.0542
92	0.0036	0.0035	0.0035	0.0034	0.0034	0.0034	0.0033	0.0033	0.0032	0.0031	0.0031	0.0029	0.0026	0.0558
93	0.0036	0.0035	0.0035	0.0035	0.0034	0.0034	0.0033	0.0033	0.0032	0.0032	0.0031	0.0030	0.0026	0.0574
94	0.0036	0.0035	0.0035	0.0035	0.0034	0.0034	0.0033	0.0033	0.0032	0.0032	0.0031	0.0030	0.0026	0.0590
95	0.0036	0.0035	0.0035	0.0035	0.0034	0.0034	0.0034	0.0033	0.0033	0.0032	0.0031	0.0030	0.0027	0.0606
96	0.0036	0.0036	0.0035	0.0035	0.0034	0.0034	0.0034	0.0033	0.0033	0.0032	0.0031	0.0030	0.0027	0.0622
97	0.0036	0.0036	0.0035	0.0035	0.0035	0.0034	0.0034	0.0033	0.0033	0.0032	0.0032	0.0030	0.0027	0.0638
98	0.0036	0.0036	0.0035	0.0035	0.0035	0.0034	0.0034	0.0033	0.0033	0.0032	0.0032	0.0030	0.0027	0.0655
99	0.0036	0.0036	0.0035	0.0035	0.0035	0.0034	0.0034	0.0034	0.0033	0.0033	0.0032	0.0031	0.0027	0.0671
100	0.0036	0.0036	0.0036	0.0035	0.0035	0.0034	0.0034	0.0034	0.0033	0.0033	0.0032	0.0031	0.0027	0.0688
101	0.0036	0.0036	0.0036	0.0035	0.0035	0.0035	0.0034	0.0034	0.0033	0.0033	0.0032	0.0031	0.0027	0.0704
102														

Finding:

When the experiment is repeated 100 times, the experiment cannot obtain the steady-state population's genetic composition.

Repeat time n=1000; Number of A1=A2=100

Code:

```
N=101;% X(100)=P[100 copies of A1, 100 copies of A2]
input=zeros(1,201);
input(N)=1;%Set the initial allele distribution
N = 100;%Set N=100 diploid heterozygous individuals
```

```
% transition matrix
P=zeros(2*N+1,2*N+1);
for i = 1:2*N+1
    for j = 1:2*N+1
        P(i,j) = nchoosek(2*N,j-1)*((i-1)/(2*N))^(j-1)*(1-(i-1)/(2*N))^(2*N-
j+1);
    end
end
n=1000; % number of time steps to take
output=zeros(n+1,2*N+1); % clear out any old values

output(1,:)=input; % generate first output value
for i=1:n,
    output(i+1,:) = output(i,:)*P;
    %a tolerance check to  automatically stop the simulation when the density
is close to its steady-state
    LIT = ismembertol(output(i+1,:),output(i,:));
    if all(LIT == 1)
        break;
    end
end
```

Simulation Result:

Name	Value
i	1000
input	*1x201 double*
j	201
LIT	*1x201 logical*
n	1000
N	100
output	*1001x201 double*
P	*201x201 double*

output

1001x201 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
985	0.4946	4.4154e-05	4.9535e-05	5.1089e-05	5.1835e-05	5.2306e-05	5.2624e-05	5.2852e-05	5.3024e-05	5.3159e-05	5.3268e-05	5.3358e-05	5.3433e-05	5.3498e
986	0.4947	4.3934e-05	4.9287e-05	5.0833e-05	5.1575e-05	5.2045e-05	5.2361e-05	5.2588e-05	5.2759e-05	5.2893e-05	5.3002e-05	5.3091e-05	5.3166e-05	5.3230e
987	0.4947	4.3714e-05	4.9041e-05	5.0579e-05	5.1318e-05	5.1784e-05	5.2099e-05	5.2325e-05	5.2495e-05	5.2629e-05	5.2737e-05	5.2826e-05	5.2900e-05	5.2964e
988	0.4947	4.3495e-05	4.8795e-05	5.0326e-05	5.1061e-05	5.1526e-05	5.1839e-05	5.2063e-05	5.2233e-05	5.2366e-05	5.2473e-05	5.2561e-05	5.2636e-05	5.2699e
989	0.4948	4.3278e-05	4.8551e-05	5.0074e-05	5.0806e-05	5.1268e-05	5.1579e-05	5.1803e-05	5.1971e-05	5.2104e-05	5.2211e-05	5.2299e-05	5.2373e-05	5.2436e
990	0.4948	4.3062e-05	4.8309e-05	4.9824e-05	5.0552e-05	5.1012e-05	5.1322e-05	5.1544e-05	5.1712e-05	5.1843e-05	5.1950e-05	5.2037e-05	5.2111e-05	5.2173e
991	0.4948	4.2846e-05	4.8067e-05	4.9575e-05	5.0299e-05	5.0757e-05	5.1065e-05	5.1286e-05	5.1453e-05	5.1584e-05	5.1690e-05	5.1777e-05	5.1850e-05	5.1913e
992	0.4948	4.2632e-05	4.7827e-05	4.9327e-05	5.0047e-05	5.0503e-05	5.0810e-05	5.1030e-05	5.1196e-05	5.1326e-05	5.1431e-05	5.1518e-05	5.1591e-05	5.1653e
993	0.4949	4.2419e-05	4.7588e-05	4.9080e-05	4.9797e-05	5.0250e-05	5.0556e-05	5.0774e-05	5.0940e-05	5.1069e-05	5.1174e-05	5.1260e-05	5.1333e-05	5.1395e
994	0.4949	4.2207e-05	4.7350e-05	4.8835e-05	4.9548e-05	4.9999e-05	5.0303e-05	5.0520e-05	5.0685e-05	5.0814e-05	5.0918e-05	5.1004e-05	5.1076e-05	5.1138e
995	0.4949	4.1996e-05	4.7113e-05	4.8591e-05	4.9300e-05	4.9749e-05	5.0051e-05	5.0268e-05	5.0432e-05	5.0560e-05	5.0664e-05	5.0749e-05	5.0821e-05	5.0882e
996	0.4949	4.1786e-05	4.6877e-05	4.8348e-05	4.9054e-05	4.9500e-05	4.9801e-05	5.0017e-05	5.0179e-05	5.0307e-05	5.0410e-05	5.0495e-05	5.0567e-05	5.0628e
997	0.4950	4.1577e-05	4.6643e-05	4.8106e-05	4.8809e-05	4.9253e-05	4.9552e-05	4.9766e-05	4.9929e-05	5.0056e-05	5.0158e-05	5.0243e-05	5.0314e-05	5.0375e
998	0.4950	4.1369e-05	4.6410e-05	4.7866e-05	4.8565e-05	4.9006e-05	4.9304e-05	4.9518e-05	4.9679e-05	4.9805e-05	4.9908e-05	4.9992e-05	5.0062e-05	5.0123e
999	0.4950	4.1162e-05	4.6178e-05	4.7626e-05	4.8322e-05	4.8761e-05	4.9058e-05	4.9270e-05	4.9431e-05	4.9556e-05	4.9658e-05	4.9742e-05	4.9812e-05	4.9872e
1000	0.4950	4.0956e-05	4.5947e-05	4.7388e-05	4.8080e-05	4.8518e-05	4.8812e-05	4.9024e-05	4.9183e-05	4.9309e-05	4.9410e-05	4.9493e-05	4.9563e-05	4.9623e
1001	0.4951	4.0752e-05	4.5717e-05	4.7151e-05	4.7840e-05	4.8275e-05	4.8568e-05	4.8779e-05	4.8937e-05	4.9062e-05	4.9163e-05	4.9246e-05	4.9315e-05	4.9375e
1002														

output

1001x201 double

	188	189	190	191	192	193	194	195	196	197	198	199	200	201
985	498e-05	5.3433e-05	5.3358e-05	5.3268e-05	5.3159e-05	5.3024e-05	5.2852e-05	5.2624e-05	5.2306e-05	5.1835e-05	5.1089e-05	4.9535e-05	4.4154e-05	0.4946
986	230e-05	5.3166e-05	5.3091e-05	5.3002e-05	5.2893e-05	5.2759e-05	5.2588e-05	5.2361e-05	5.2045e-05	5.1575e-05	5.0833e-05	4.9287e-05	4.3934e-05	0.4947
987	964e-05	5.2900e-05	5.2826e-05	5.2737e-05	5.2629e-05	5.2495e-05	5.2325e-05	5.2099e-05	5.1784e-05	5.1318e-05	5.0579e-05	4.9041e-05	4.3714e-05	0.4947
988	699e-05	5.2636e-05	5.2561e-05	5.2473e-05	5.2366e-05	5.2233e-05	5.2063e-05	5.1839e-05	5.1526e-05	5.1061e-05	5.0326e-05	4.8795e-05	4.3495e-05	0.4947
989	436e-05	5.2373e-05	5.2299e-05	5.2211e-05	5.2104e-05	5.1971e-05	5.1803e-05	5.1579e-05	5.1268e-05	5.0806e-05	5.0074e-05	4.8551e-05	4.3278e-05	0.4948
990	173e-05	5.2111e-05	5.2037e-05	5.1950e-05	5.1843e-05	5.1712e-05	5.1544e-05	5.1322e-05	5.1012e-05	5.0552e-05	4.9824e-05	4.8309e-05	4.3062e-05	0.4948
991	913e-05	5.1850e-05	5.1777e-05	5.1690e-05	5.1584e-05	5.1453e-05	5.1286e-05	5.1065e-05	5.0757e-05	5.0299e-05	4.9575e-05	4.8067e-05	4.2846e-05	0.4948
992	653e-05	5.1591e-05	5.1518e-05	5.1431e-05	5.1326e-05	5.1196e-05	5.1030e-05	5.0810e-05	5.0503e-05	5.0047e-05	4.9327e-05	4.7827e-05	4.2632e-05	0.4948
993	395e-05	5.1333e-05	5.1260e-05	5.1174e-05	5.1069e-05	5.0940e-05	5.0774e-05	5.0556e-05	5.0250e-05	4.9797e-05	4.9080e-05	4.7588e-05	4.2419e-05	0.4949
994	138e-05	5.1076e-05	5.1004e-05	5.0918e-05	5.0814e-05	5.0685e-05	5.0520e-05	5.0303e-05	4.9999e-05	4.9548e-05	4.8835e-05	4.7350e-05	4.2207e-05	0.4949
995	882e-05	5.0821e-05	5.0749e-05	5.0664e-05	5.0560e-05	5.0432e-05	5.0268e-05	5.0051e-05	4.9749e-05	4.9300e-05	4.8591e-05	4.7113e-05	4.1996e-05	0.4949
996	628e-05	5.0567e-05	5.0495e-05	5.0410e-05	5.0307e-05	5.0179e-05	5.0017e-05	4.9801e-05	4.9500e-05	4.9054e-05	4.8348e-05	4.6877e-05	4.1786e-05	0.4949
997	375e-05	5.0314e-05	5.0243e-05	5.0158e-05	5.0056e-05	4.9929e-05	4.9766e-05	4.9552e-05	4.9253e-05	4.8809e-05	4.8106e-05	4.6643e-05	4.1577e-05	0.4950
998	123e-05	5.0062e-05	4.9992e-05	4.9908e-05	4.9805e-05	4.9679e-05	4.9518e-05	4.9304e-05	4.9006e-05	4.8565e-05	4.7866e-05	4.6410e-05	4.1369e-05	0.4950
999	872e-05	4.9812e-05	4.9742e-05	4.9658e-05	4.9556e-05	4.9431e-05	4.9270e-05	4.9058e-05	4.8761e-05	4.8322e-05	4.7626e-05	4.6178e-05	4.1162e-05	0.4950
1000	623e-05	4.9563e-05	4.9493e-05	4.9410e-05	4.9309e-05	4.9183e-05	4.9024e-05	4.8812e-05	4.8518e-05	4.8080e-05	4.7388e-05	4.5947e-05	4.0956e-05	0.4950
1001	375e-05	4.9315e-05	4.9246e-05	4.9163e-05	4.9062e-05	4.8937e-05	4.8779e-05	4.8568e-05	4.8275e-05	4.7840e-05	4.7151e-05	4.5717e-05	4.0752e-05	0.4951
1002														

Finding:

When the experiment is repeated 1000 times and all of the parents have A1A2, the simulation result is nearly

[0.5,0,0,0,…,0,0,0,0.5]

The steady-state population's genetic composition is that approximately 0.5 probability that the offspring has A1 and 0.5 probability that the offspring has A2.

Repeat time n=1000; Number of A1=50, A2=150

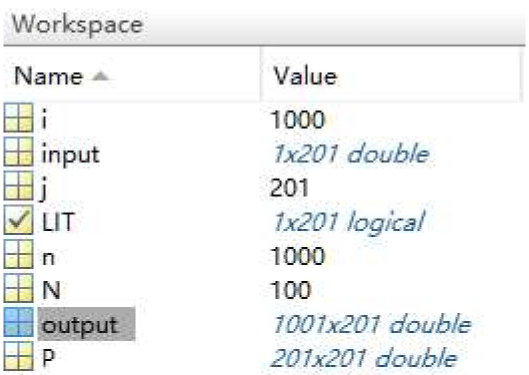
Part 2:

Code:

```
N=50;% X(50)=P[50 copies of A1, 150 copies of A2]
input=zeros(1,201);
input(N)=1;%Set the initial allele distribution
N = 100;%Set N=100 diploid heterozygous individuals
```

```
% transition matrix
P=zeros(2*N+1,2*N+1);
for i = 1:2*N+1
    for j = 1:2*N+1
        P(i,j) = nchoosek(2*N,j-1)*((i-1)/(2*N))^(j-1)*(1-(i-1)/(2*N))^(2*N-
j+1);
    end
end
n=1000; % number of time steps to take
output=zeros(n+1,2*N+1); % clear out any old values

output(1,:)=input; % generate first output value
for i=1:n,
    output(i+1,:) = output(i,:)*P;
    %a tolerance check to  automatically stop the simulation when the density
is close to its steady-state
    LIT = ismembertol(output(i+1,:),output(i,:));
    if all(LIT == 1)
        break;
    end
end
```

Simulation Result:



output

1001x201 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
985	0.7510	3.2674e-05	3.6655e-05	3.7805e-05	3.8357e-05	3.8706e-05	3.8941e-05	3.9110e-05	3.9237e-05	3.9337e-05	3.9418e-05	3.9484e-05	3.9540e-05	3.9587e
986	0.7511	3.2511e-05	3.6472e-05	3.7616e-05	3.8165e-05	3.8513e-05	3.8747e-05	3.8914e-05	3.9041e-05	3.9140e-05	3.9220e-05	3.9287e-05	3.9342e-05	3.9389e
987	0.7511	3.2348e-05	3.6290e-05	3.7428e-05	3.7974e-05	3.8320e-05	3.8553e-05	3.8720e-05	3.8846e-05	3.8944e-05	3.9024e-05	3.9090e-05	3.9145e-05	3.9192e
988	0.7511	3.2186e-05	3.6108e-05	3.7241e-05	3.7785e-05	3.8128e-05	3.8360e-05	3.8526e-05	3.8651e-05	3.8750e-05	3.8829e-05	3.8895e-05	3.8950e-05	3.8996e
989	0.7511	3.2025e-05	3.5927e-05	3.7054e-05	3.7596e-05	3.7938e-05	3.8168e-05	3.8333e-05	3.8458e-05	3.8556e-05	3.8635e-05	3.8700e-05	3.8755e-05	3.8801e
990	0.7511	3.1865e-05	3.5748e-05	3.6869e-05	3.7408e-05	3.7748e-05	3.7977e-05	3.8142e-05	3.8266e-05	3.8363e-05	3.8442e-05	3.8507e-05	3.8561e-05	3.8607e
991	0.7512	3.1706e-05	3.5569e-05	3.6685e-05	3.7220e-05	3.7559e-05	3.7787e-05	3.7951e-05	3.8074e-05	3.8171e-05	3.8249e-05	3.8314e-05	3.8368e-05	3.8414e
992	0.7512	3.1547e-05	3.5391e-05	3.6501e-05	3.7034e-05	3.7371e-05	3.7598e-05	3.7761e-05	3.7884e-05	3.7980e-05	3.8058e-05	3.8122e-05	3.8176e-05	3.8222e
993	0.7512	3.1389e-05	3.5214e-05	3.6319e-05	3.6849e-05	3.7184e-05	3.7410e-05	3.7572e-05	3.7694e-05	3.7790e-05	3.7868e-05	3.7932e-05	3.7985e-05	3.8031e
994	0.7512	3.1232e-05	3.5038e-05	3.6137e-05	3.6665e-05	3.6998e-05	3.7223e-05	3.7384e-05	3.7506e-05	3.7601e-05	3.7678e-05	3.7742e-05	3.7795e-05	3.7841e
995	0.7512	3.1076e-05	3.4863e-05	3.5956e-05	3.6481e-05	3.6813e-05	3.7037e-05	3.7197e-05	3.7318e-05	3.7413e-05	3.7490e-05	3.7553e-05	3.7606e-05	3.7652e
996	0.7513	3.0921e-05	3.4688e-05	3.5777e-05	3.6299e-05	3.6629e-05	3.6852e-05	3.7011e-05	3.7132e-05	3.7226e-05	3.7303e-05	3.7365e-05	3.7418e-05	3.7463e
997	0.7513	3.0766e-05	3.4515e-05	3.5598e-05	3.6117e-05	3.6446e-05	3.6668e-05	3.6826e-05	3.6946e-05	3.7040e-05	3.7116e-05	3.7179e-05	3.7231e-05	3.7276e
998	0.7513	3.0612e-05	3.4342e-05	3.5420e-05	3.5937e-05	3.6264e-05	3.6484e-05	3.6642e-05	3.6761e-05	3.6855e-05	3.6930e-05	3.6993e-05	3.7045e-05	3.7089e
999	0.7513	3.0459e-05	3.4171e-05	3.5242e-05	3.5757e-05	3.6082e-05	3.6302e-05	3.6459e-05	3.6577e-05	3.6671e-05	3.6746e-05	3.6808e-05	3.6860e-05	3.6904e
1000	0.7513	3.0307e-05	3.4000e-05	3.5066e-05	3.5578e-05	3.5902e-05	3.6120e-05	3.6276e-05	3.6394e-05	3.6487e-05	3.6562e-05	3.6624e-05	3.6675e-05	3.6719e
1001	0.7513	3.0155e-05	3.3830e-05	3.4891e-05	3.5400e-05	3.5722e-05	3.5939e-05	3.6095e-05	3.6212e-05	3.6305e-05	3.6379e-05	3.6440e-05	3.6492e-05	3.6536e
1002														

output

1001x201 double

	188	189	190	191	192	193	194	195	196	197	198	199	200	201
985	578e-05	3.9531e-05	3.9475e-05	3.9408e-05	3.9328e-05	3.9228e-05	3.9100e-05	3.8932e-05	3.8697e-05	3.8348e-05	3.7796e-05	3.6646e-05	3.2666e-05	0.2410
986	381e-05	3.9333e-05	3.9278e-05	3.9211e-05	3.9131e-05	3.9032e-05	3.8905e-05	3.8737e-05	3.8503e-05	3.8156e-05	3.7607e-05	3.6463e-05	3.2502e-05	0.2411
987	184e-05	3.9137e-05	3.9081e-05	3.9015e-05	3.8936e-05	3.8837e-05	3.8710e-05	3.8544e-05	3.8311e-05	3.7965e-05	3.7419e-05	3.6281e-05	3.2340e-05	0.2411
988	988e-05	3.8941e-05	3.8886e-05	3.8820e-05	3.8741e-05	3.8642e-05	3.8517e-05	3.8351e-05	3.8119e-05	3.7775e-05	3.7232e-05	3.6099e-05	3.2178e-05	0.2411
989	793e-05	3.8746e-05	3.8691e-05	3.8626e-05	3.8547e-05	3.8449e-05	3.8324e-05	3.8159e-05	3.7929e-05	3.7587e-05	3.7046e-05	3.5919e-05	3.2017e-05	0.2411
990	599e-05	3.8553e-05	3.8498e-05	3.8433e-05	3.8355e-05	3.8257e-05	3.8133e-05	3.7968e-05	3.7739e-05	3.7399e-05	3.6860e-05	3.5739e-05	3.1857e-05	0.2411
991	406e-05	3.8360e-05	3.8306e-05	3.8241e-05	3.8163e-05	3.8066e-05	3.7942e-05	3.7779e-05	3.7550e-05	3.7212e-05	3.6676e-05	3.5561e-05	3.1698e-05	0.2412
992	214e-05	3.8168e-05	3.8114e-05	3.8050e-05	3.7972e-05	3.7875e-05	3.7753e-05	3.7590e-05	3.7363e-05	3.7026e-05	3.6493e-05	3.5383e-05	3.1540e-05	0.2412
993	023e-05	3.7977e-05	3.7924e-05	3.7860e-05	3.7782e-05	3.7686e-05	3.7564e-05	3.7402e-05	3.7176e-05	3.6841e-05	3.6310e-05	3.5206e-05	3.1382e-05	0.2412
994	833e-05	3.7787e-05	3.7734e-05	3.7670e-05	3.7593e-05	3.7498e-05	3.7376e-05	3.7215e-05	3.6990e-05	3.6657e-05	3.6129e-05	3.5030e-05	3.1225e-05	0.2412
995	644e-05	3.7599e-05	3.7545e-05	3.7482e-05	3.7405e-05	3.7310e-05	3.7189e-05	3.7029e-05	3.6805e-05	3.6473e-05	3.5948e-05	3.4855e-05	3.1069e-05	0.2412
996	456e-05	3.7411e-05	3.7358e-05	3.7295e-05	3.7218e-05	3.7124e-05	3.7003e-05	3.6844e-05	3.6621e-05	3.6291e-05	3.5769e-05	3.4681e-05	3.0914e-05	0.2413
997	268e-05	3.7224e-05	3.7171e-05	3.7108e-05	3.7032e-05	3.6938e-05	3.6818e-05	3.6660e-05	3.6438e-05	3.6110e-05	3.5590e-05	3.4507e-05	3.0759e-05	0.2413
998	082e-05	3.7037e-05	3.6985e-05	3.6923e-05	3.6847e-05	3.6754e-05	3.6634e-05	3.6476e-05	3.6256e-05	3.5929e-05	3.5412e-05	3.4335e-05	3.0605e-05	0.2413
999	897e-05	3.6852e-05	3.6800e-05	3.6738e-05	3.6663e-05	3.6570e-05	3.6451e-05	3.6294e-05	3.6075e-05	3.5749e-05	3.5235e-05	3.4163e-05	3.0452e-05	0.2413
1000	712e-05	3.6668e-05	3.6616e-05	3.6555e-05	3.6480e-05	3.6387e-05	3.6269e-05	3.6113e-05	3.5894e-05	3.5571e-05	3.5059e-05	3.3992e-05	3.0300e-05	0.2413
1001	529e-05	3.6485e-05	3.6433e-05	3.6372e-05	3.6297e-05	3.6205e-05	3.6088e-05	3.5932e-05	3.5715e-05	3.5393e-05	3.4883e-05	3.3822e-05	3.0149e-05	0.2413
1002														

Finding:

1. When the the population contains 150 copies of A2 and 50 copies of A1, the simulation result is nearly

[0.75,0,0,0,…,0,0,0,0.25]

The steady-state population's genetic composition is that approximately 0.75 probability that the offspring has A2 and 0.25 probability that the offspring has A1.

2. With the large number of generations, if the initial situation has more gene A1 than gene A2, the steady state genetic composition will contain more gene A1 than gene A2.

2. This scenario defy the Markov chain ergodic theorem and Perron-Frobenius theorem because the Perron-Frobenius theorem asserts that a real square matrix with positive entries has a unique largest real eigenvalue and the corresponding eigenvector can be chosen to have strictly positive components, and the Markov chain ergodic theorem asserts that state i is said to be ergodic if it is aperiodic and positive recurrent. However, the model does not confirm every state can communicate with every other state. Thus, not all states in the Markov chain are irreducible and

aperiodic. Therefore, This scenario defy the Markov chain ergodic theorem and Perron-Frobenius theorem.

Conclusion

Overall, the simulations of three experiments focus on Markov chains theory and discrete events system. The discrete event simulations are completed in the first experiment using nonhomogeneous Poisson Process based on the different cases of single server queueing systems. The second trial simulates the number of packets in buffer or process of a HOL blocking switch under heavy-load assumption and computes the overall efficiency of the switch. The third experiment uses Markov chain theory to simulate the population's genetic drift and obtain the steady-state population's genetic composition after multiple generations. In conclusion, the project uses mathematical methods to simulate Markov chain system and discrete time stochastic processes.