University of Southern California

EE511 Simulation Methods for Stochastic Systems

Project #5 – Markov chains and discrete events

BY

Mohan Krishna Thota

USC ID: 6683486728

mthota@usc.edu

Problem 1

- Server will be following a probability distribution in time taken to assist customer.
- The difference between non-homogeneous Poisson process and a normal Poisson process is that the average rate of arrivals can be varied with time.
- With the help of non-homogeneous process random points that are generated in time are modeled correctly in many applications.
- As given in the problem, Single server Queuing System, where customers are arriving according to a non-homogeneous Poisson process with lambda(t), t>=0
- If the server is free, it handles customers else it joins queue.
- As soon as it completes assists a customer, it then assists long waiting customer else it will be set free.
- If no customer arrives, then it will be fixed time T. Server will be handling all customers in queue.
- Two important things are:
    - ✓ Customer spent average time in queue.
    - ✓ Average time Past T that last customer departs.

For non-homogenous Poisson process, an algorithm is used:

let lambda(t) be the bounded intensity function. Let the first arrival after time s be Ts.

Algorithm:

Lambda is chosen in such a way that lambda(t) is less than lambda.
Given lambda(t), and t greater than zero and lambda:
Let t=p
Generate uniform random numbers(U1) using MATLAB rand() function
Let t=1/lambda*log(U1)
Now generate another set of random numbers (U2)
If U2<=(lambda(t)/lambda)

Then Ts=t and stopped
In order to calculate lambda there are two scenarios.
First Scenario:
For the first five hours, there is a linear increase from 4 to 19 hours, which can be expressed as 3t+4.
Second Scenario:
In the next 5 hours, there is a linear decrease from 19 hours to 4 hours. It can be expressed as 3t-4.
Exponential distribution is followed by service-time at a rate of 25 jobs per hour.
Random numbers from exponential distribution is generated as an array using MATLAB's exprnd() with mean parameter MU function.
As already mentioned MU=25.
Distribution followed by the waiting time is uniform (0,0.3).
Therefore break time is incremented in such a way that when there are no jobs in the queue.
Answer obtained is 11.2251.
From the above answer we could see that the server is idle for most of time in 100 hours.

CODE:

```
Y=0;
M=1;
lambda_value=20;
time_break=0;
Number_of_hours=100; %number of hours%
K=0;
while K<Number_of_hours
   random_numbers1=rand();
   random_numbers2=rand();
   K=K-(log(random_numbers1)/lambda_value);
   if(random_numbers2<=value)
       stop_time(M)=K;
       M=M+1;
   end
   if(mod(K,10) > 5)
       lambda_t_value=-3*K-4;
       value=lambda_t_value/lambda_value;
   elseif(mod(K,10)<5)
       lambda_t_value=3*K+4;
       value=lambda_t_value/lambda_value;
   end
end
for i=1:length(stop_time)-1
   p=exprnd(25);
   while((p + stop_time(i)) < stop_time(i+1))
       const=0.3*rand();
       time_break=time_break+const;
       p=p+const;
   end
end
disp('Break time is:')
disp(time_break)
```

OUTPUT:

Break time is:
  11.8535

Problem 2:

In class we considered the 2x2 HOL-blocking switch performance under the heavy-load assumption, i.e. there was always a packet at each input. A more realistic model includes modeling the buffer for each input. Assume the probability that a packet arrives at input port i in a time slot is pi=p(a constant) for each input. Assume also that the probability that a packet arriving at input i should be switched to output j is $r$ij. Define the system state to be the number of packets in the buffer at each input in the middle of time slot k and the desired output port of the packet at the HOL of each input, i.e. decide the desired output when the packet moves to the HOL-slot. Assume that packets arrive to the input buffers at the end of a time slot. If there is a packet in the buffer at the beginning of a time slot the switch attempts. If the delivery attempt succeeds the switch removes the packet from the input queue at the end of the time slot. If the delivery attempt fails then the packet stays at the head of the line and the switch will attempt delivery in the next time slot.

SOLUTION

- Here 2 probabilities are assigned for the two cases of packet reaching port1 and packet reaching port 2 in our case.
- As per given problem let the probability that the packet reaching case 2 is 0.7.
- A random number is generated using rand() function and it is verified whether the generated random number is less than 0.7.
- If the value is lesser than 0.7, then it is case 1 i.e., packet reached port1.
- P_switch which is variable is not switched, if it is less than 0.5.
- If the variable Is less than 0.75 at port1 and 0.25 at port2 then switch wont be switched.
- Generally a total of 4 states are possible,
    1. 00
    2. 11
    3. 10
    4. 01

- For cases 1 and 2, this will result in contention, only one of the packet will be sent.
- Then we should buffer, if the packet arrived at the other port and this packet will be scheduled in the next time slot.
- For cases 3 and 4, both the packets will be sent.
- For the case of contention(collision), two flag variables are used.
- If the first flag is 1, then the port2 state will be changed to previous state as the packet is in buffered state and the same way if the second flag is 1 the port 1 and buffer will be reset.
- Efficiency of the switch is calculated by total number of packets sent by the received number of packets in a given timeslot.
- By using bootci command, 95% confidence interval is calculated as asked in the question

CODE:

```
Number=1000;
Buffer_1 = zeros(1,Number);
Buffer_2 = zeros(1,Number);
Number_p1 = 0;
Number_p2 = 0;
L = 1;
P1 = zeros(1,Number);
P2 = zeros(1,Number);
switch1 = zeros(1,Number);
switch2 = zeros(1,Number);
prob1 = 0.7;
prob2 = 0.7;
flag_1 = 0;
flag_2 = 0;
No_packets = zeros(1,Number);
packet1 = 0;
packet2 = 0;
buff1 = 0;
buff2 = 0;
for k = 1:Number
packet_switched = 0;
X = 1;
Y = 1;
nopacket_flag = 0;
P1(k) = rand();
P2(k) = rand();
if (P1(k) < prob1)
packet1 = packet1 + 1;
switch1(k) = rand();
else
nopacket_flag = 1;
end
if (P2(k) < prob2)
packet2 = packet2 + 1;
switch2(k) = rand();
else
nopacket_flag = 1;
end
if (switch1(k) < 0.75)
switch1(k) = 0;
else
switch1(k) = 1;
end
if (switch2(k) < 0.25)
switch2(k) = 0;
else
switch2(k) = 1;
end
if(k>1)
if(flag_1 == 1)
```

```
switch2(k)=switch2(k-1);
if ( buff2 == 0 ), flag_1 = 0; end
end
if(flag_2 == 1)
switch1(k)=switch1(k-1);
if ( buff1 == 0 ), flag_2 = 0; end
end
end
if ((switch1(k) == 1 && switch2(k) == 0) || (switch1(k) == 0 && ...
switch2(k) == 1) || nopacket_flag)
packet_switched = packet_switched + ...
packet1 + packet2; X = 0; Y = 0;
else
contention = rand(1);
if(contention < 0.5)
buff2 = buff2 + 1;
packet_switched = packet_switched + 1;
if (buff1 > 0)
buff1 = buff1 - 1;
end
flag_1 = 1; X = 0;
else
buff1 = buff1 + 1;
packet_switched = packet_switched + 1;
if (buff2 > 0)
buff2 = buff2 - 1;
end
flag_2 = 1; Y = 0;
end
end
No_packets(k) = packet_switched;
if(packet1*packet2 == 1)
J_switch(L) = No_packets(k)/(packet1 + ...
packet2);
L = L + 1;
end
Buffer_1(k) = buff1; Buffer_2(k) = buff2;
Number_p1 = Number_p1 + packet1 - X;
packet1 = 0;
Number_p2 = Number_p2 + packet2 - Y;
packet2 = 0;
end
confidence_interval = bootci(1000, @mean, J_switch);
disp('Overall efficiency Confidence interval ');
disp(confidence_interval);
Mean_B2 = mean(Buffer_2);
Mean_B1 = mean(Buffer_1);
switched_M = mean(No_packets);
```

```
total_switched = Number_p1 + Number_p2;
figure(1);
histogram(Buffer_1);
disp('mean of packets at port1:');
disp(Mean_B1);
ylabel('Frequency');
xlabel('Packet count');
title('Distribution of Count of packets  at i/p Line 1 in Buffer');
figure(2);
histogram(No_packets);
disp('mean of packets switched in timeslot1:');
disp(switched_M);
ylabel('Frequency');
xlabel('Packet count');
title('Distribution of Count of switched packets  in timeslot one');
figure(3);
histogram(Buffer_2);
disp('mean of packets at port2:');
disp(Mean_B2);
ylabel('Frequency');
xlabel('Packet count');
title('Distribution of Count of packets at i/p Line 2 in Buffer');
```

OUTPUT:

Case1:

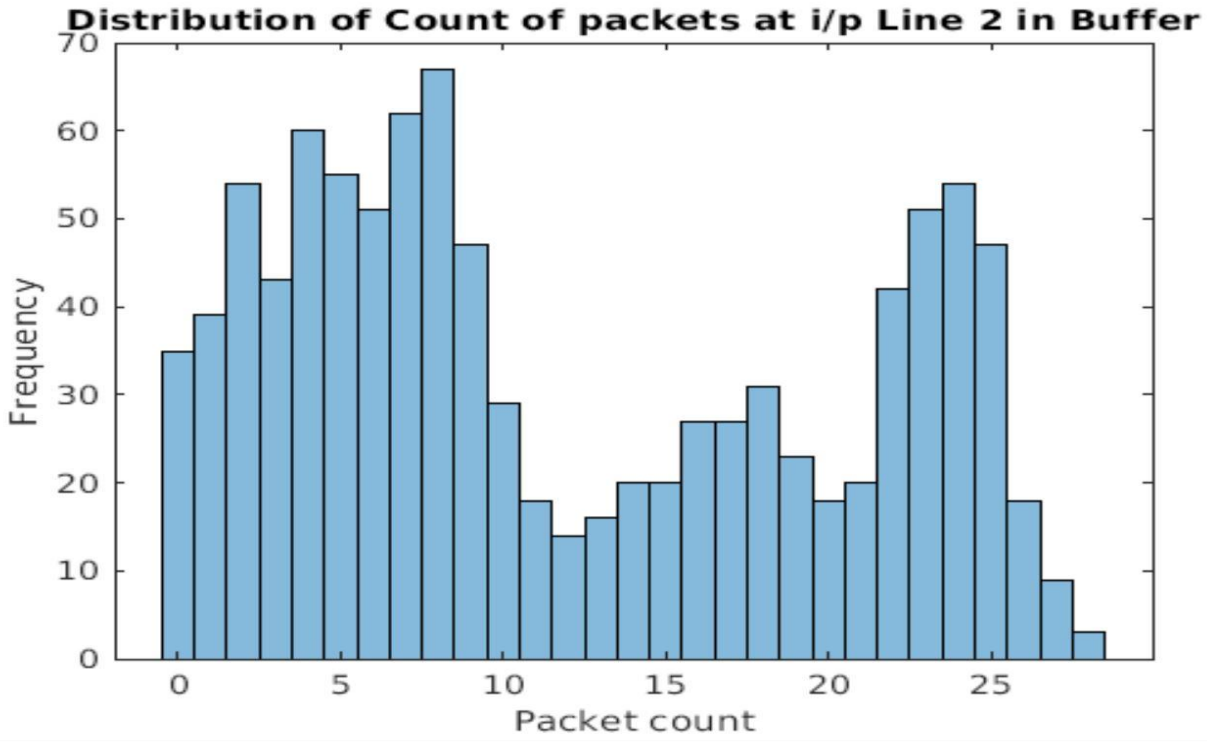
when switch=0.5

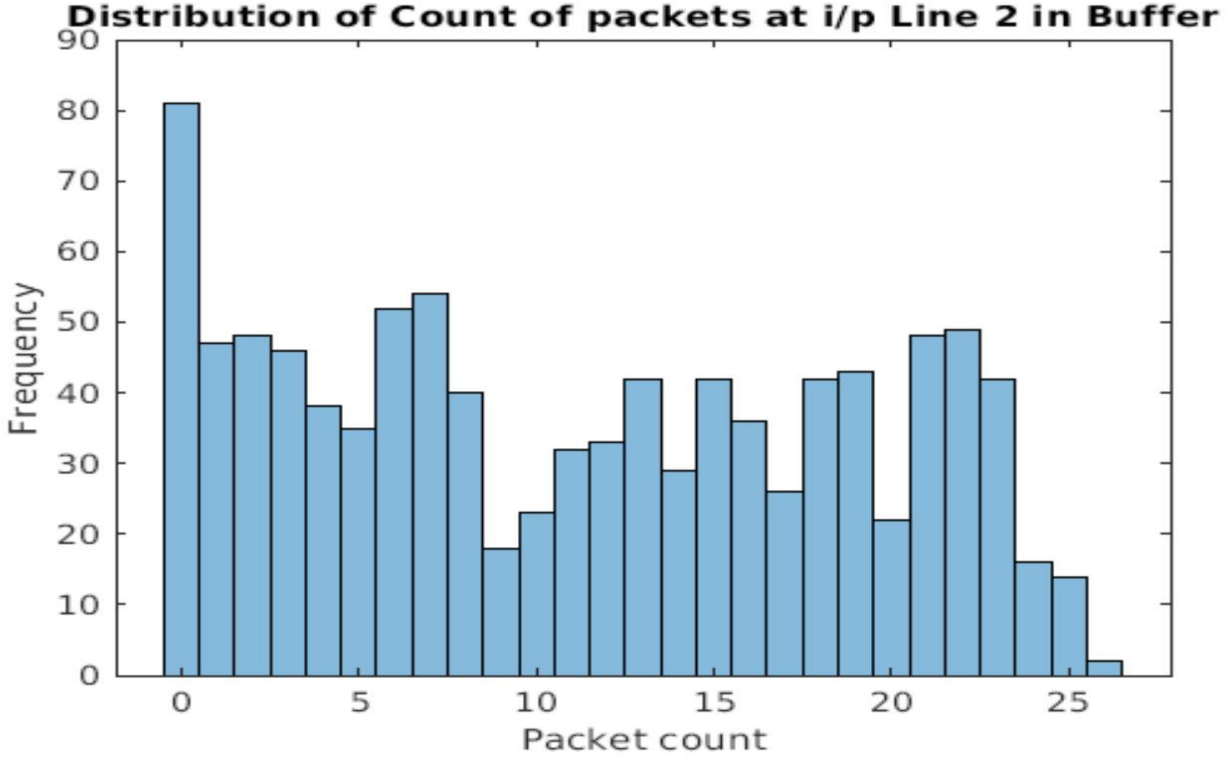
Overall efficiency Confidence interval
0.5434
0.5705

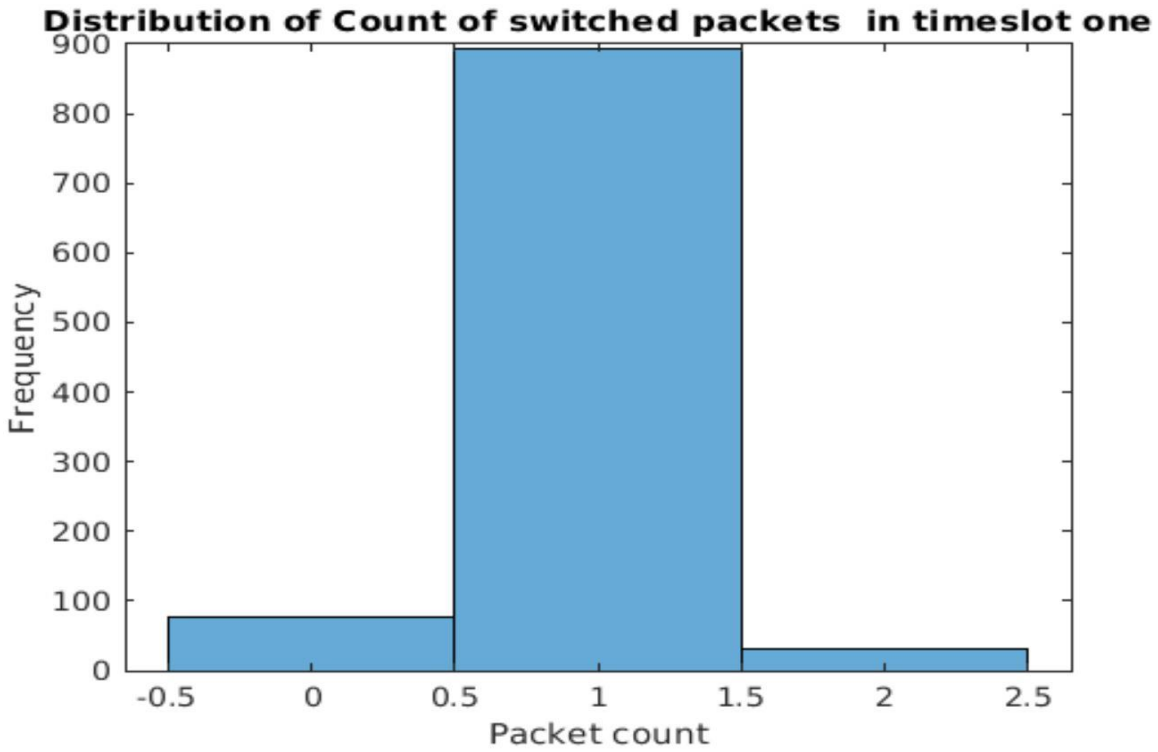
mean of packets at port1:
13.2610

mean of packets switched in timeslot1:
0.9590

mean of packets at port2:
12.3050


Distribution of Count of packets at i/p Line 2 in Buffer
Frequency
Packet count
0
10
20
30
40
50
60
70
0
5
10
15
20
25


Distribution of Count of packets at i/p Line 2 in Buffer
Frequency
Packet count
0
10
20
30
40
50
60
70
80
90
0
5
10
15
20
25



Case2:

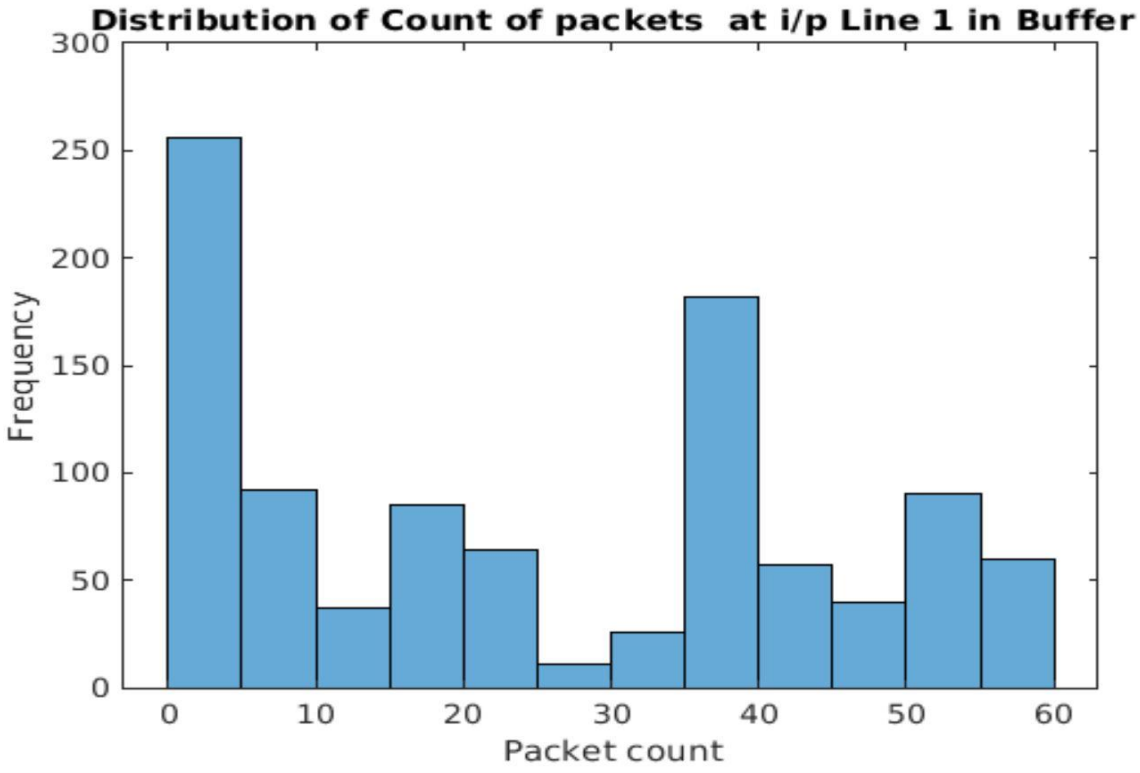
when switch1=0.75 and switch2=0.25

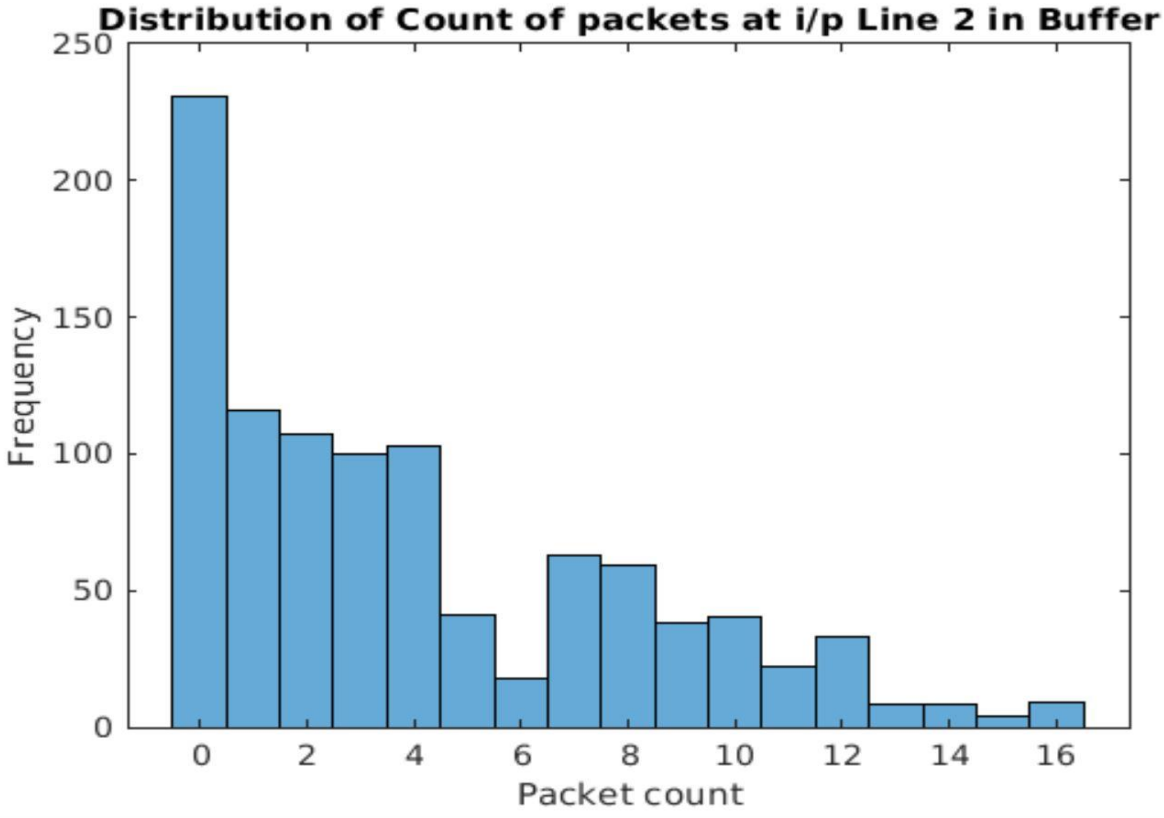
Overall efficiency Confidence interval
0.5444
0.5713

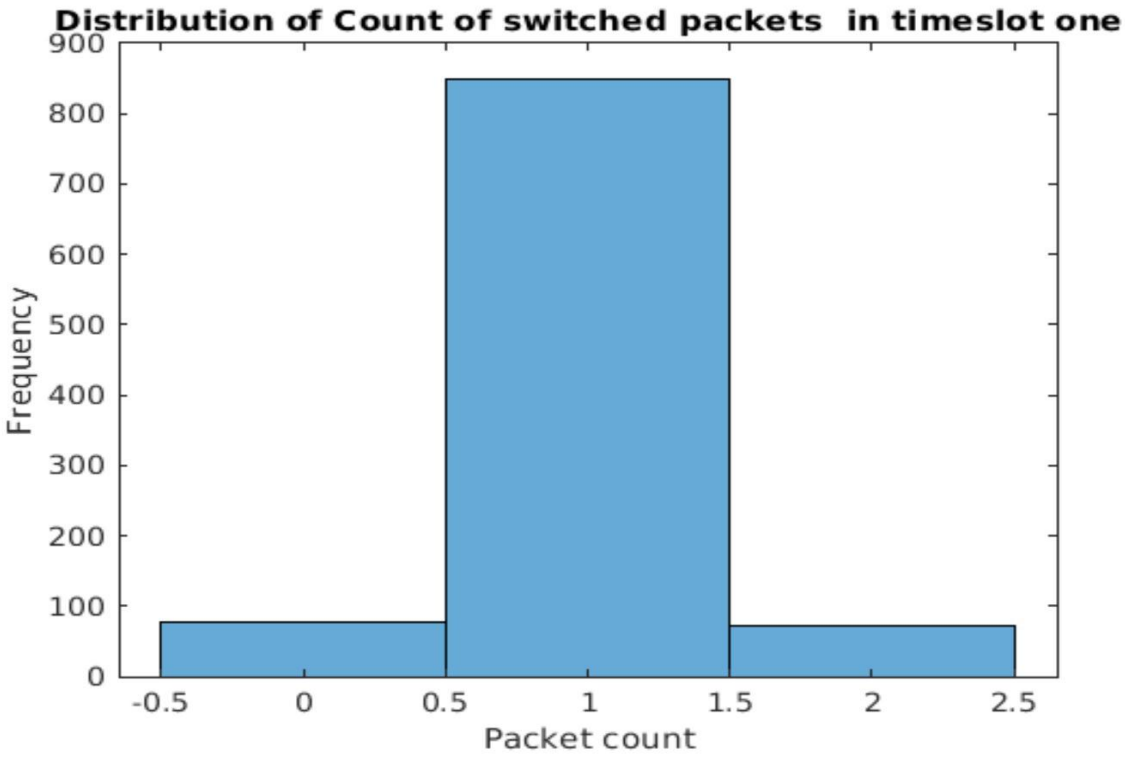
mean of packets at port1:
5.1650

mean of packets switched in timeslot1:
0.9570

mean of packets at port2:
18.0820


Distribution of Count of packets at i/p Line 1 in Buffer
Frequency
Packet count
0
50
100
150
200
250
300
0
10
20
30
40
50
60


Distribution of Count of packets at i/p Line 2 in Buffer
Frequency
Packet count
0
50
100
150
200
250
0
2
4
6
8
10
12
14
16



CONCLUSION:

From the above results, we could see that the confidence interval is more in the symmetric when compared to asymmetric.

Efficiency of the switch is higher in Symmetric case.

Problem 3:

Wright Fisher Model:

- Let us take a simple haploid model, which consists of a total of 2N genes of random reproduction where N is the number of diploid organisms.
- Each haploid consists of either A1 or A2 allele.
- During the initial phase, we may ignore mutation and selective forces.
- A gene reproduces some number of offspring, which will be exact like itself and dies immediately.
- Therefore, each gene has a lifespan of just one generation.
- In this way we could see how genes are transmitted from one generation to other.
- The only observation in this process is just the frequency of allies which are transmitted from one generation to another and the death and birth processes are hidden.
- A genetic drift governs the frequency of the next generation. It is defined as a force which reduces heterozygosity by this random loss of alleles.
- In this population of 2N haploids, the focus is on the frequency of allele A1.
- We can see this process of changing from one generation to another as in terms of markov chain, where the state X of the chain corresponds to number of haploids of type A1.
- Let a generation X takes one value among 0,1,2,…….,2N which is a state space and a generation is represented using t.
- Xt denotes the value taken by X in generation t.
- Here our model assumes that the value at next generation is obtained by sampling of the genes of the previous generation with replacement.
- Therefore we could obtain next generation by 2N independent Bernoulli trials and each Xt is a binomial random variable.
- Initial generation has I genes of type A1 and 2N-i genes of type A2.
- For a Bernoulli Trial probability of success pi is 1/2N and the probability of failure is qi=1-pi i.e., 1-(1/2N).
- Here probability of success means the genes resulting in allele A1 and the other is allele A2.

- This could result in a Markov Chain(Xn) where in a population of 2N individuals, Xn is the number of genes in the nth generation.
- Here Xt+1 is a binomial random variable with index 2N and parameter (probability of success) Xt/2N.

$$P(X_{t+1} = j | X_t = i) = p_{ij} = \binom{2N}{j} p_i^j q_i^{2N-j}$$
$$= \binom{2N}{j} (i/2N)^j \{1 - (i/2N)\}^{2N-j}$$

- A stochastic matrix is one which consists of probability vectors as columns vectors.
- We know a Markov chain is one which consists of a finite states and some known pij where pij is the probability of moving from state I to j.
- A steady vector is one in which Ap=p, where p is a probability vector which is an eigen vector of A associated to eigen value 1.
- We know from Perron-Frobenius Theorem, that a square matrix which is real with positive entries has a largest unique real eigen value and the eigen vector is chosen in such a way that it has positive components and this applies to non negative matrices too.
- A markov chain can be ergodic if the number of steps taken are bounded by a finite positive integer N.
- For a fully connected transition matrix, which has a non zero probability for all transitions, and this condition is fulfilled with N=1.
- We know that for a Markov chain is either irreducible or not aperiodic, with more than one state and just one transition per state. Therefore it is not ergodic.

CODE DESCRIPTION:

- As it can be seen from the code, we will be first taking the location of A1 Allele in the gene as a user input.
- Then an array is initialized with zeros and the user input is made 1.
- Let the number of individuals be 100. Now a transition matrix is introduced of order with both rows and columns to be 2*100+1=201.

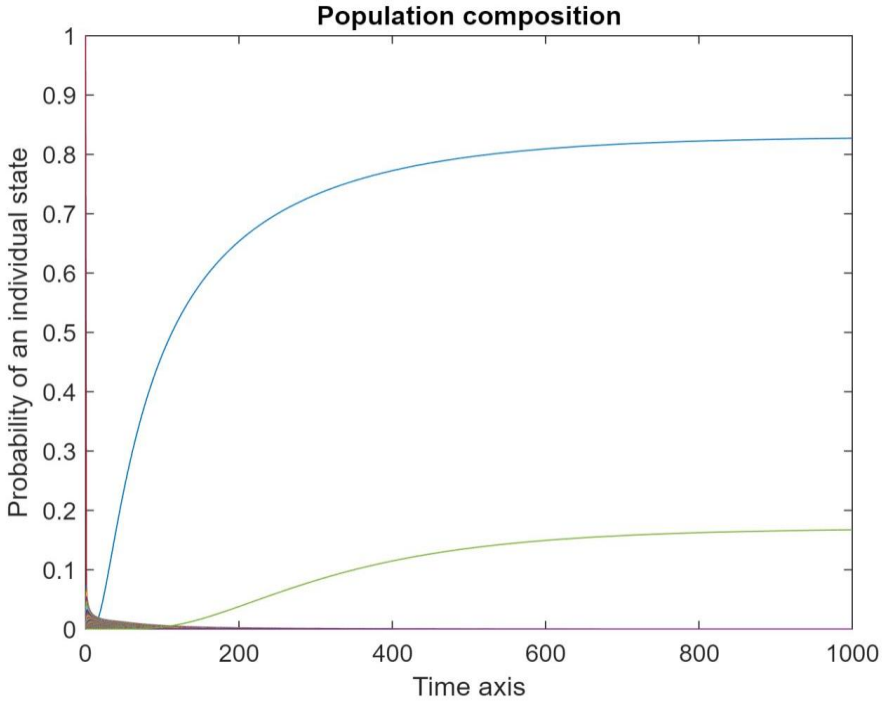
- Now each element of this transition is calculated using formula in code, MATLAB nchoosek function is used to obtain nCr value. 1000 time indices are taken.
- Now the output is calculated by the product of this transition matrix with the number of time steps.
- From the below observations, we could see that results are in steady state.

CODE:

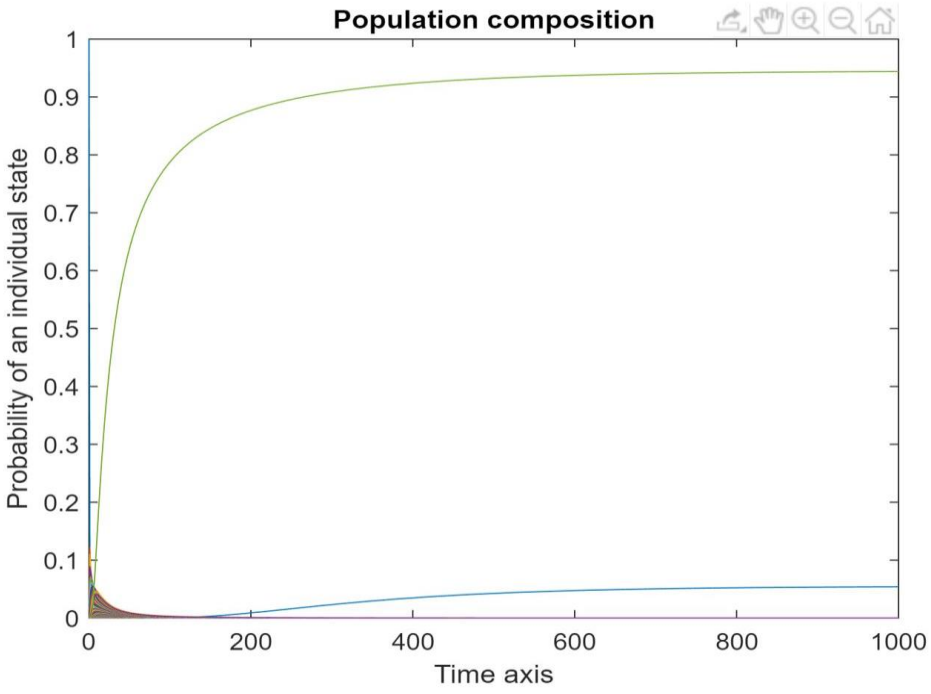
```
clear;
inp_dist=zeros(1,201);
user_input=input('Location of A1 allele');
inp_dist(user_input)=1;
No_individuals=100;
kk=2*No_individuals+1;
time_steps=1000;
output=zeros(time_steps+1,kk);
time_indices=0:time_steps;
tras_matrix=zeros(kk,kk);
for m=1:kk
    for n=1:kk
        tras_matrix(m,n) = nchoosek(2*No_individuals,n-1)*((m-
1)/(2*No_individuals))^(n-1)*(1-(m-1)/(2*No_individuals))^(2*No_individuals-n+1);
        %tras_matrix(m,n)=nchoosek(2*No_individuals,n-1)*((m-
1)/(2*No_individuals))^(n-1)*(1-(m-1)/(2*No_individuals))^(2*No_individuals-n+1);
    end
end
output(1,:)=inp_dist;


for j=1:time_steps
    output(j+1,:)=output(j,:)*tras_matrix;
    threshold=output(j+1,:)-output(j,:);
    if(threshold==1)
        break;
    end
end
plot(time_indices,output);
xlabel('Time axis');
ylabel('Probability of an individual state');
title('Population composition');
```

OUTPUT:

User input=35



User input=190



CONCLUSION:

- From the above graphs, we could see that for distinct allele distribution initially:
- With respect to scenario 1, steady state reaching time is much lesser than compared to scenario 2.
- Therefore, this contradicts with the Perron-Frobenius Theorem and Markov chains ergodic theorem as it is evident above.