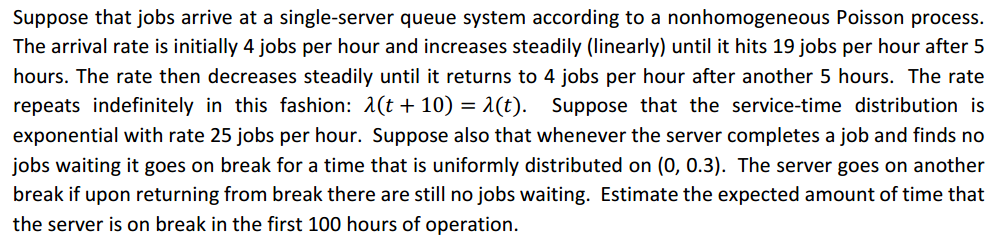
EE511 ZHANG Fan USCID: 1417-68-5115

Q1:

**Analysis:**

* Job arrival:
  + Nonhomogeneous Poisson process:
    - =19
  + Algorithm for Nonhomogeneous Poisson process
    - Suppose that λ(t) is the bounded intensity function (arrival function) for a non-homogenous Poisson process.  To generate a sample Ts that is the time of the first arrival after time(in this case, we exclude :

1. Let
2. Generate
3. Let % generate random time interval for t
4. Generate
5. If % ??
6. Go to step2
   * : the time from the first time the job arrivals(exclude )

* Job service:
  + , t<1
  + We can substitute t with
* Waiting time if no job
  + U~(0,0.3)
* Algorithm
  + Define , this is the time between two job arrivals.
  + Define remainder(i)=interval(i)-s(i)+remainder(i-1), this is the time when the server doesn’t have any job
  + Define R=U~(0,0.3)
  + Remainder(i)=remainder(i-1)-U~R, sum=sum+R, until remainder(i)
  + i=i+1 and go back to the step two

**Theoretically:**

* The average for 100 hours is 11.5. That means. So Theoretically there will be 1150 jobs during 100 hours.
* For the service, the exponential distribution, the average serve-time is . So assume there are 1150 jobs, it means that it will spend hours to serve all of the jobs. Therefore, for the waiting time, it should be 100-46=54 hours.

**Code:**

clear

close all; clc;

% initial value of all variables

x = 0; % 50 products on hand ??? x=0

y = 0;%number ----job number

Ts = 0; %time interval

T = 100; % the total time we want to analyze

i = 1;

while(Ts(i) < T)dn

% generate nonhomogeneous Poission event

% lamda(t) = is the time !! for one job

lambda = 11.5;

t=0;

while(t<T)

u1 = rand ();

t = Ts(i) - 1/lambda\*log(u1);%why log?

u2 = rand();

if(mod(t,10)<5&&mod(t,10)>0)

%pare=11111

pare=(3\*mod(t,10)+4)/lambda;

else

pare=(34-3\*mod(t,10))/lambda;

end

if (u2 <= pare)% ok, draw a digram and you will understand

Ts(i+1) = t;

i = i + 1;

break;

end

end

end

Ts(end)=100

interval=diff(Ts)

remainder=interval(1);

sum=0;

for i=2:length(Ts)-1

while(remainder>0)

r=0.3\*rand();

remainder=remainder-r;

sum=sum+r;

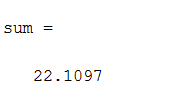
end

remainder=interval(i)-exprnd(1/25)+remainder

end

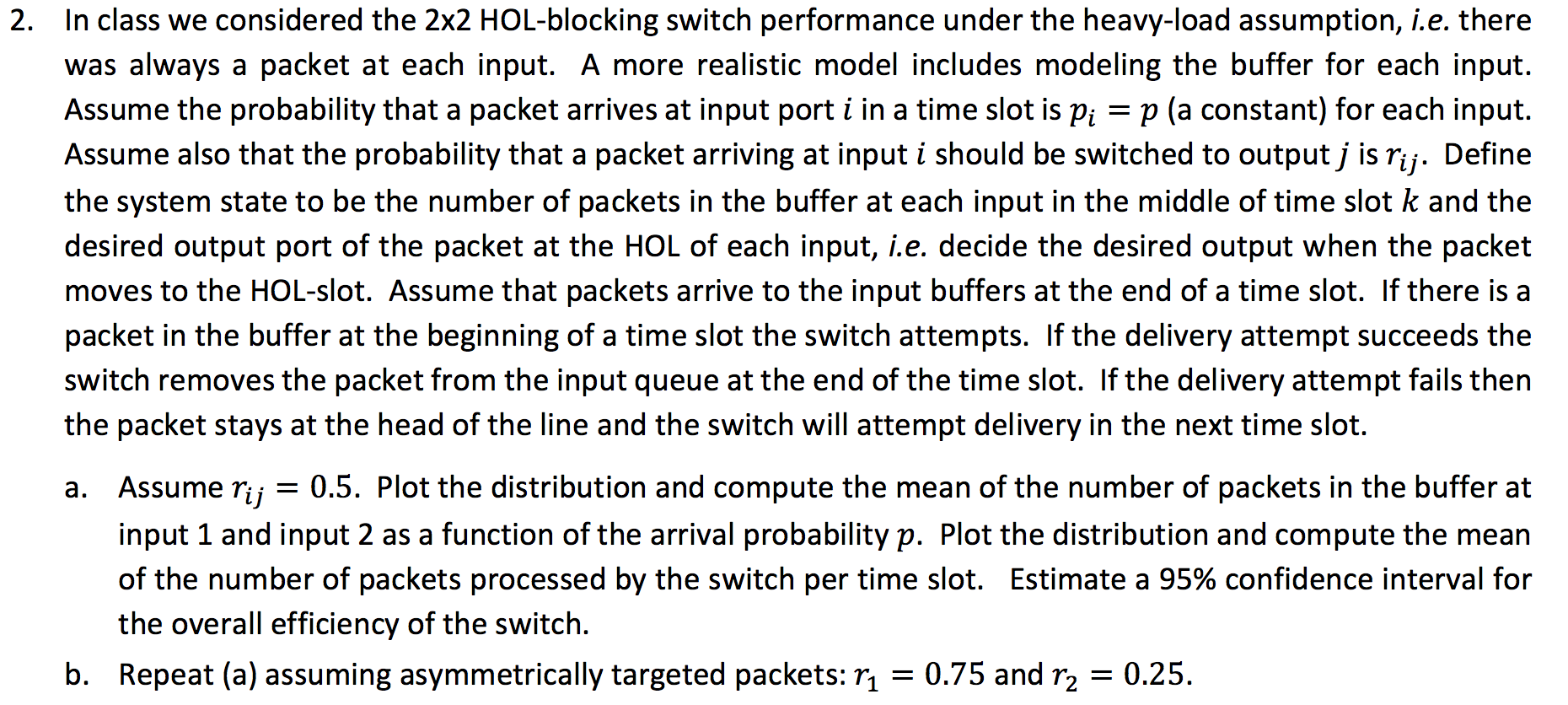
sum

**Results**: 24

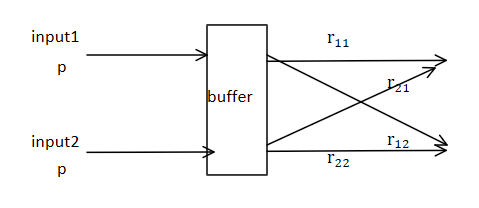


**Comment:**

* The simulation result is roughly equal to the theoretical value. So it can be considered as reasonable.



**Analysis**:



The algorithm is as follow:

* let u1=rand() and u2=rand();
* when u1<p, buffer(2)=buffer(1)+1; when u2<p, buffer(2)=buffer(2)+1
* there are three condition:
  + input1->output1, input2->output2, this condition can transfer 2 packet at one time
  + both buffer(1) and buffer(2) don’t have any packets, then the transfer 0 packet at one time
  + other conditions, including either buffer1 or buffer2 has a packet or collision occurs, then transfer 1 packet
* calculate the mean of the packets that is transferred each time slot

**code for calculate the distribution and internal with a given p:**

clc;clear

x=0;mean\_buffer\_temp=0;mean\_buffer\_temp2=0;

p=1

r=1/2

buffer=[0,0];%initial

out=[0,0];n=100000;sum\_out=0;

for i=1:n

u1=rand();u2=rand();

if(u1<p)

buffer(1)=buffer(1)+1;

end

if(u2<p)

buffer(2)=buffer(2)+1;

end

u3=rand();%the propability for input 1 to out 1, and input to out 2 is 1-u3

u4=rand();%the probaility for input 2 to out 1, and input to out 2 is 1-u4

if(buffer(1)~=0)

if(u3<r)

out(1)=out(1)+1;

else

out(2)=out(2)+1;

end

buffer(1)=buffer(1)-1;

end

if(buffer(2)~=0)

if(u4<r)

out(1)=out(1)+1;

else

out(2)=out(2)+1;

end

buffer(2)=buffer(2)-1;

end

%mean of per slot

if(out(1)==1&&out(2)==1)%1+1

sum\_out=sum\_out+2;

elseif(out(1)==0&&out(2)==0)

sum\_out;

else

sum\_out=sum\_out+1;

end

%%collision!

if(out(1)==2)%detect collision in out1

u5=rand();%randomly choose one to send

if(u5<1/2)

buffer(1)=buffer(1)+1;

else

buffer(2)=buffer(2)+1;

end

out(1)=out(1)-1;

end

if(out(2)==2)%detect collision in out1

u6=rand();%randomly choose one to send

if(u6<1/2)

buffer(1)=buffer(1)+1;

else

buffer(2)=buffer(2)+1;

end

out(2)=out(2)-1;

end

distribution\_out(i)=sum(out);

distribution\_buffer(i)=sum(buffer);

out=[0,0];

mean\_buffer\_temp=mean\_buffer\_temp+buffer(1);

end

figure(1)

hist(distribution\_out);

title('distribution for average transfer ')

figure(2)

hist(distribution\_buffer);

title('distribution for packet in buffer')

p

r

z\_afa = norminv([0.025 0.975],0,1);

efficiency= distribution\_out/2;

MeanBuffer=mean(distribution\_buffer)

MeanProccesed=mean(efficiency)

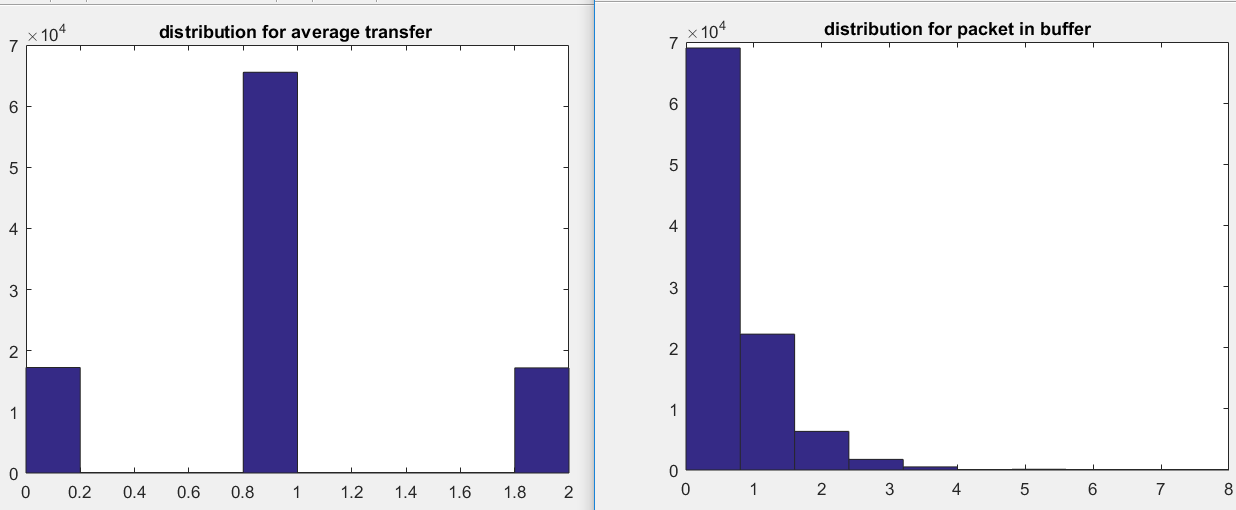
StdAll=std(efficiency);

BoundAll(1)=MeanProccesed-z\_afa(2)\*StdAll/sqrt(10000);

BoundAll(2)=MeanProccesed+z\_afa(2)\*StdAll/sqrt(10000);

BoundAll

Distribution for p=0.5, time slot number = 10000

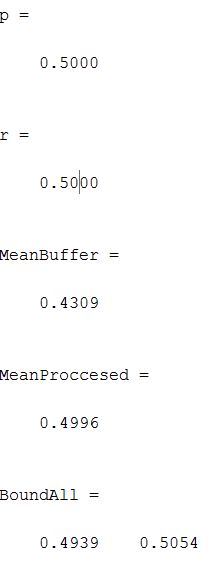


The confidence for p=1, n=10000

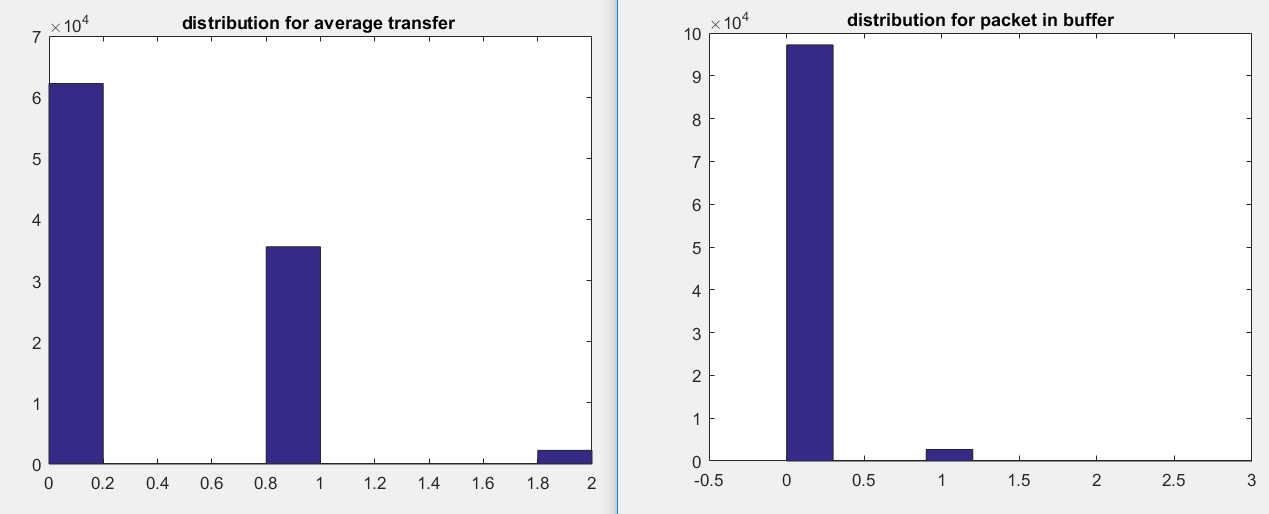
Confidence interval for p=0.2 is [0.4939,0.5054]

The mean of the number of the packet processed is 0.9992: and the effifiency is 0.4996

The mean of the packet in the buffer is 0.4309.



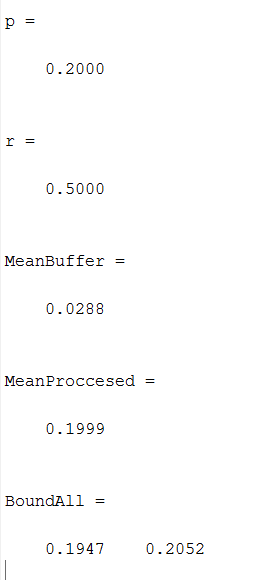
Distribution for p=0.2, time slot number =10000:



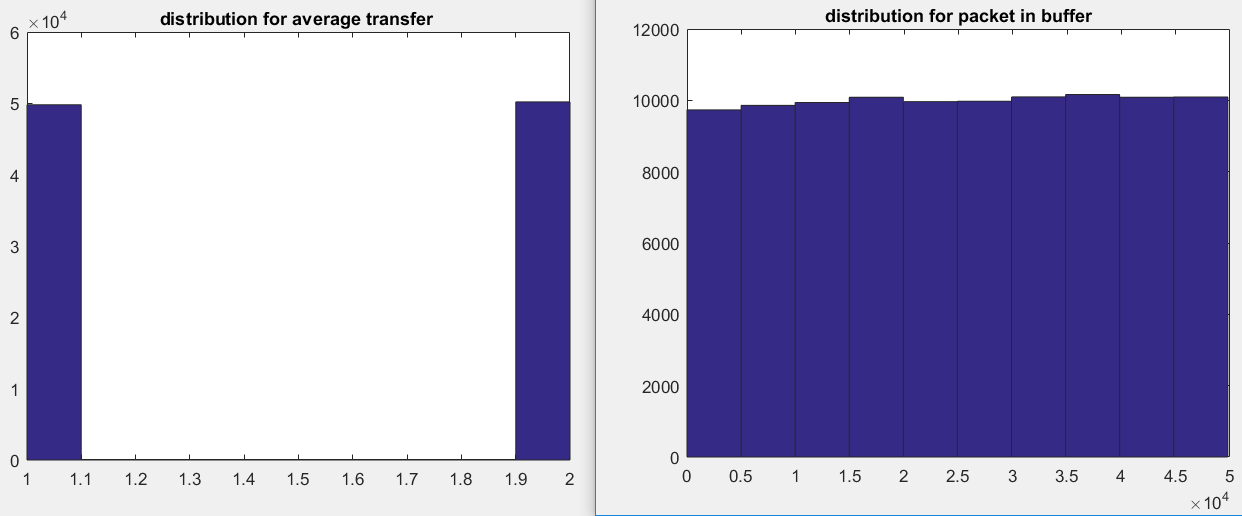
Confidence interval for p=0.2 is [0.1947,0.2052]

The mean of the number of the packet processed is 0.3998: and the efficiency is 0.1999

The mean of the packet in the buffer is 0.0288.



Distribution for p=1, time slot number =10000:

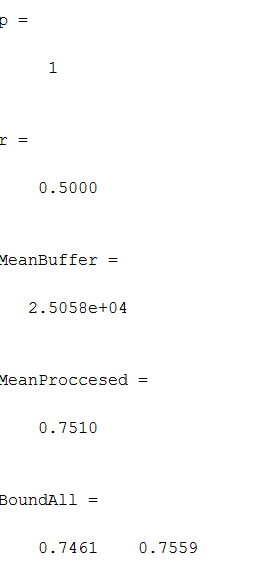


Confidence interval for p=0.2

Confidence interval for p=0.2 is [0.7461,0.7559]

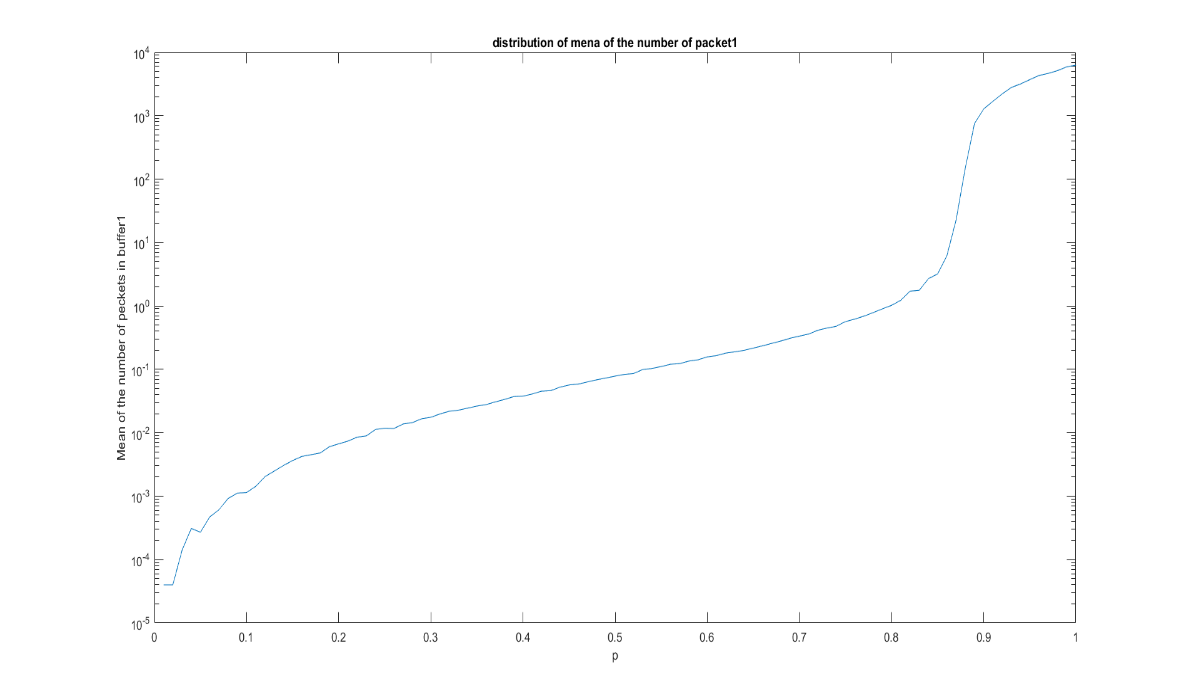
The mean of the number of the packet processed is 1.520: and the effieincy is 0.7510

The mean of the packet in the buffer is 2.5e+05, which is not convergent.

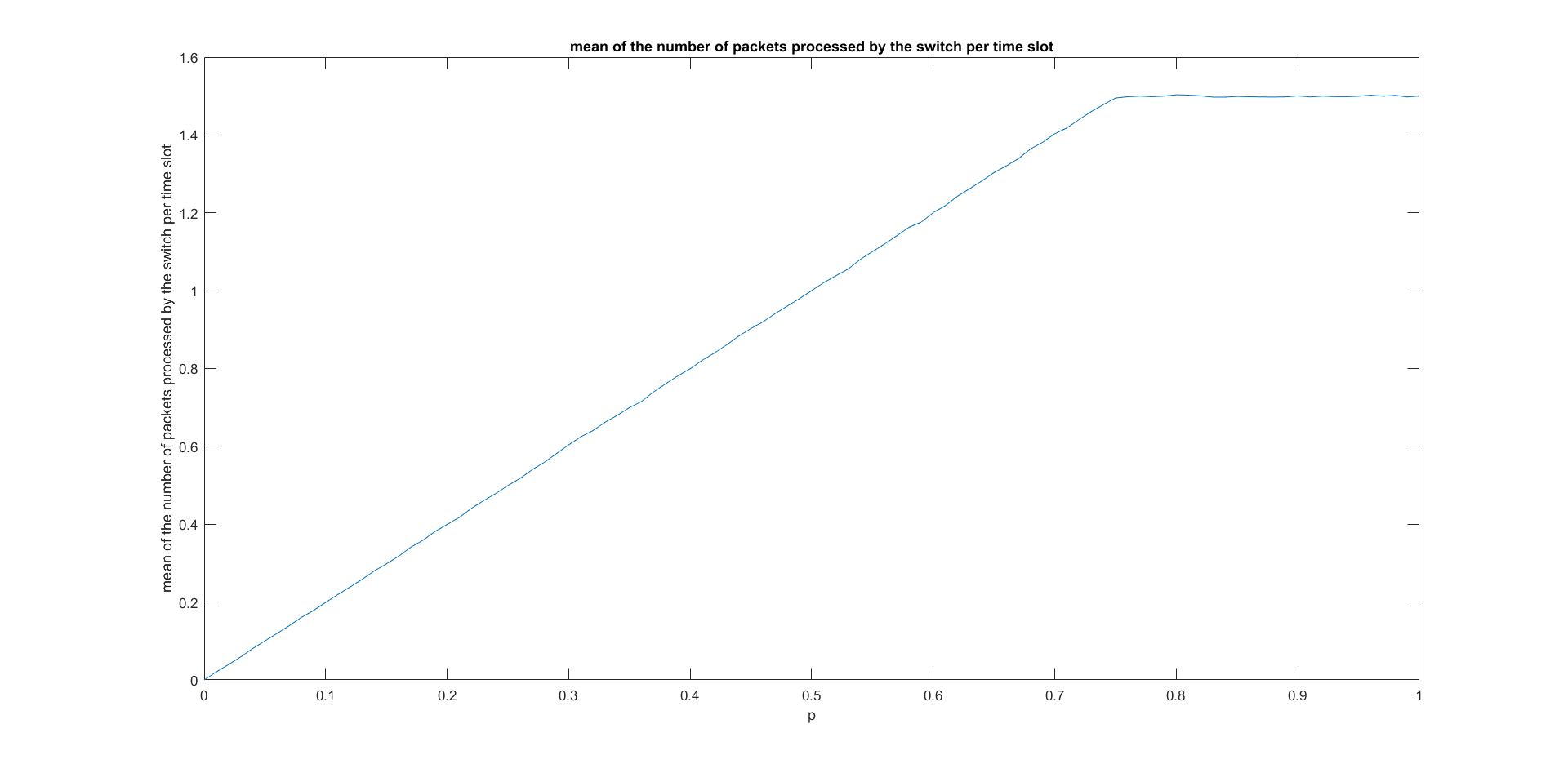


Then we simulate the mean of the number of packet in buffer 1 with p:

**The mean of the number of packet in buffer2 with the different p:**



**The mean of the number of packet processed by the switch per time slot with the different p:**



**Comment:**

In this case, time slot number t=10000 is selected.

We can conclude that, the mean of the number of packets processed by the switch per time slot increase linearly with the p, and at last it is a steady value 1.5 when p>0.88.

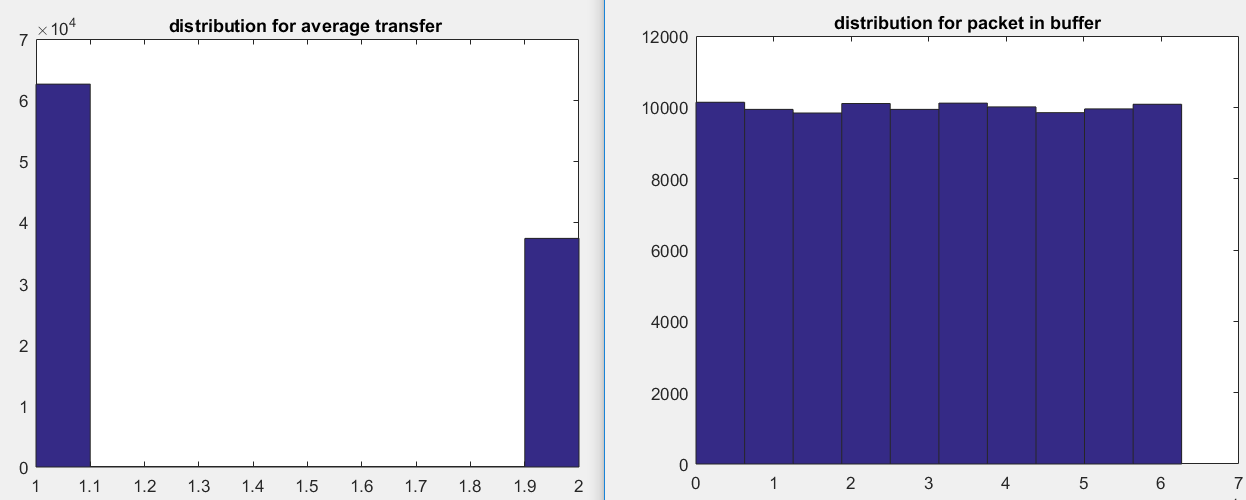
And the mean packets in the buffer will increase with the time slot when p is large enough; and while it is zero when p is small. The bound is also around 0.88 .

**Q2(b). analysis:**

The algorithm in question 2 is the same is that in question a. Revise the parameter to let , we obtain the following results:

Mean of the packet number in buffer with the 0<P<1 and the mean of the packet number transferred with the 0<p<1.

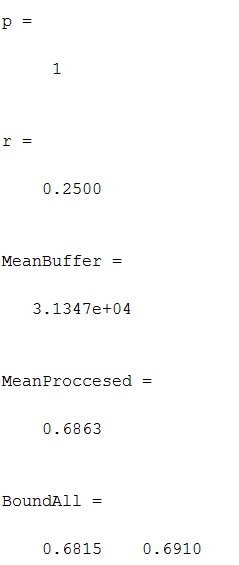
Distribution for p=1, n=1000 and r1 =0.25, r2=0.75



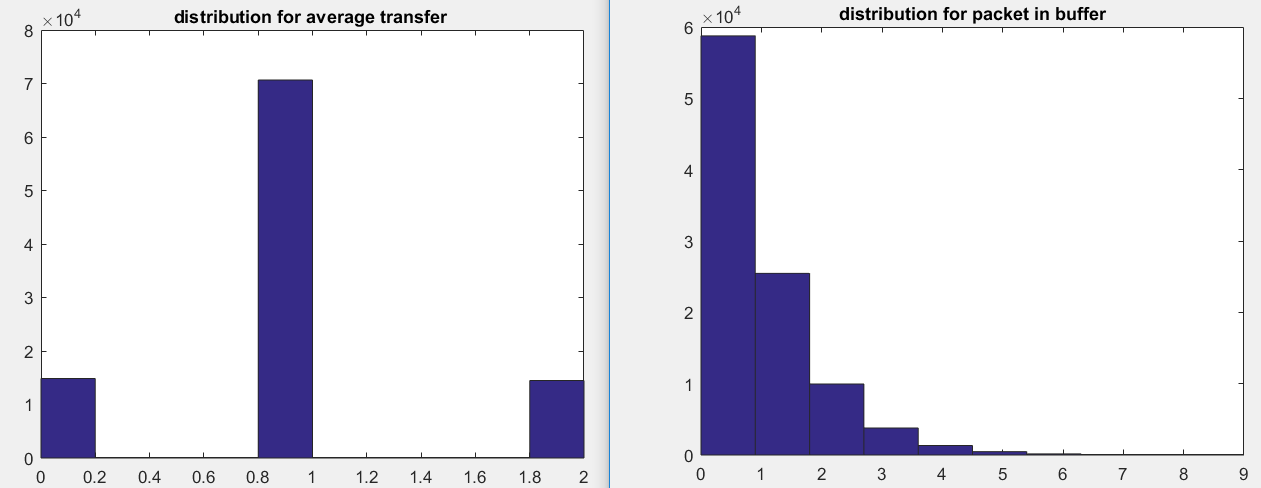
The confidence interval for p=1 is [0.6815,0.6910]

The mean of the packet number in the buffer is 3.1347e+04, which is not convergence

And the mean of the packet processed is 0.6863\*2=1.3626, and the effifiency is 0.6863



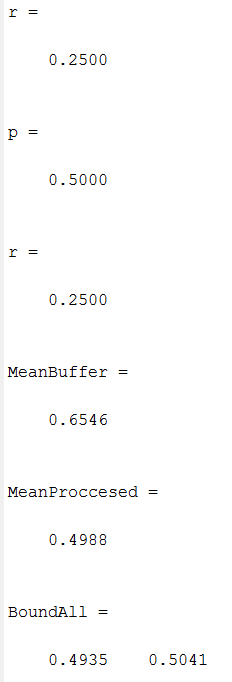
Distribution for p=0.5, n=10000 and r1 =0.25, r2=0.75



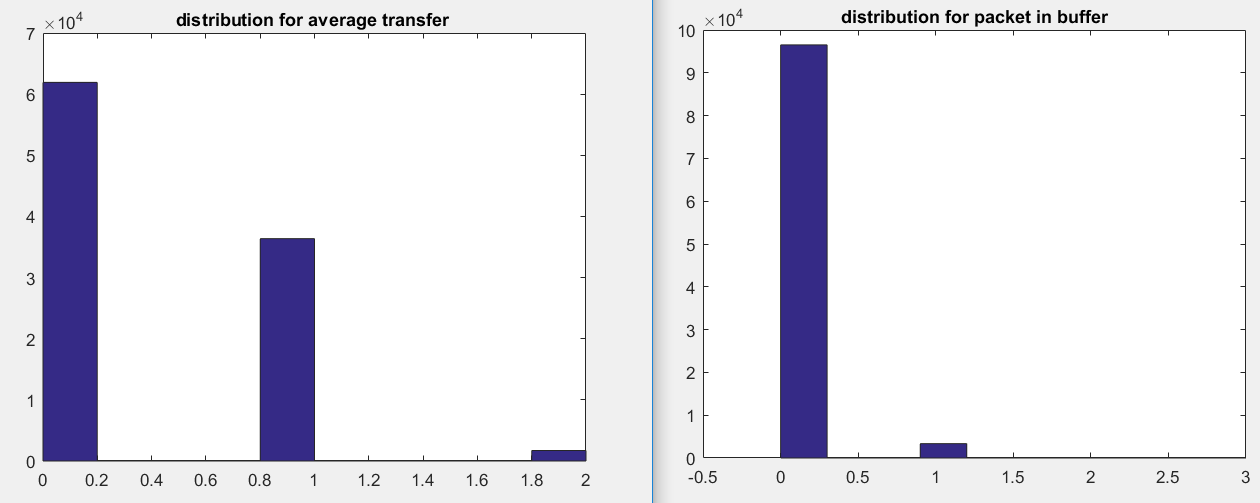
Mean of the transferred(processed) packet number=0.9976, and the efficiency is 0.4988

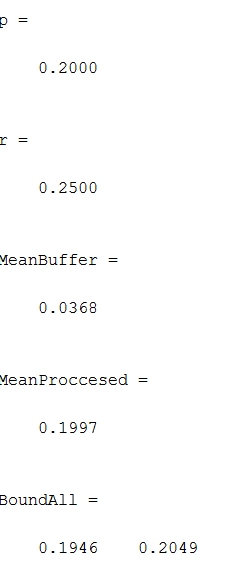
The confidence interval for p=0.5 is [0.4935,0.5041]

Mean of the packet number in the buffer is : 0.6546



Distribution for p=0.2, n=10000 and r1 =0.25, r2=0.75

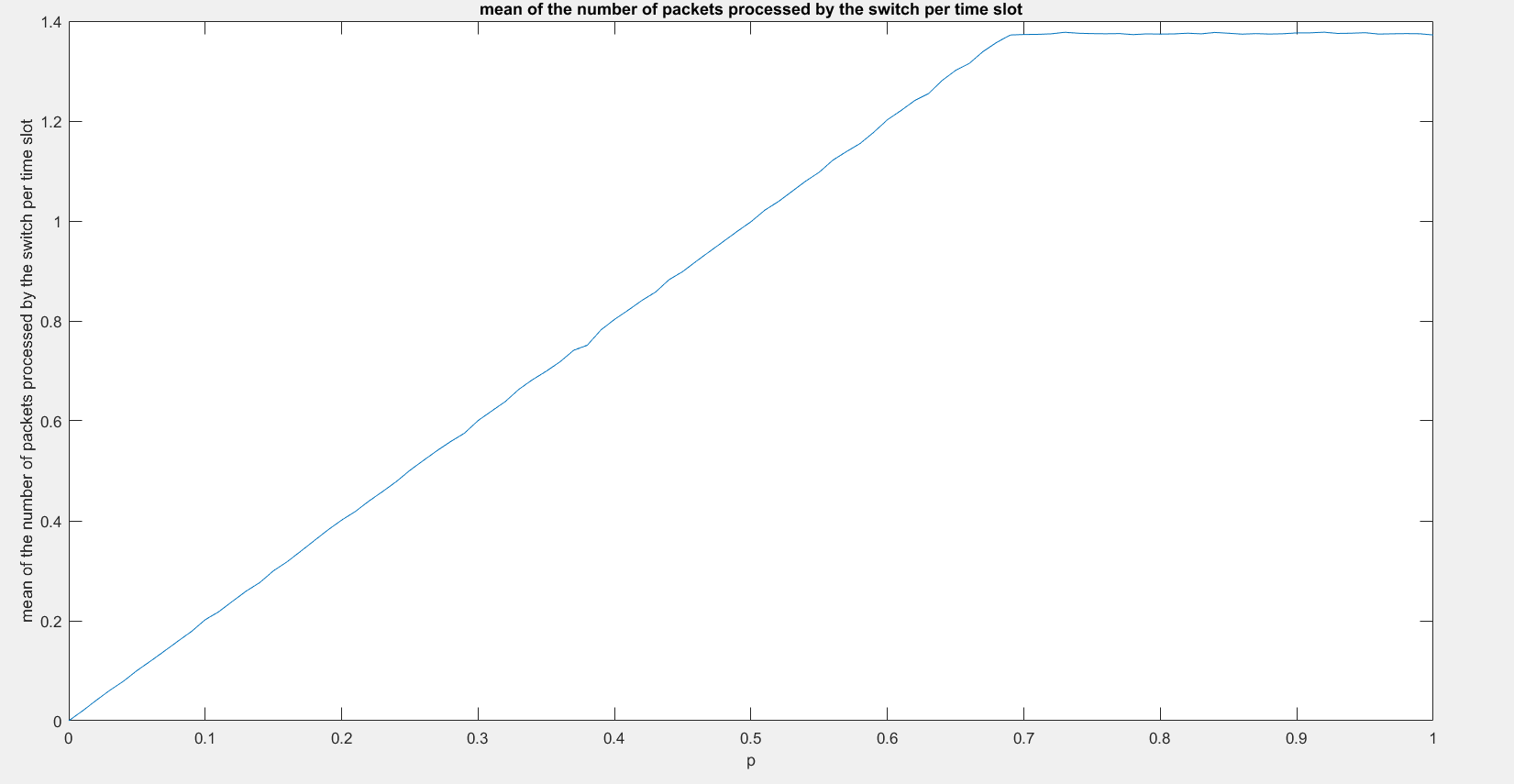




The mean of the packet in the buffer is 0.0368

Mean of the proccesed packet number=0.3994, and the efficiency is 0.1994

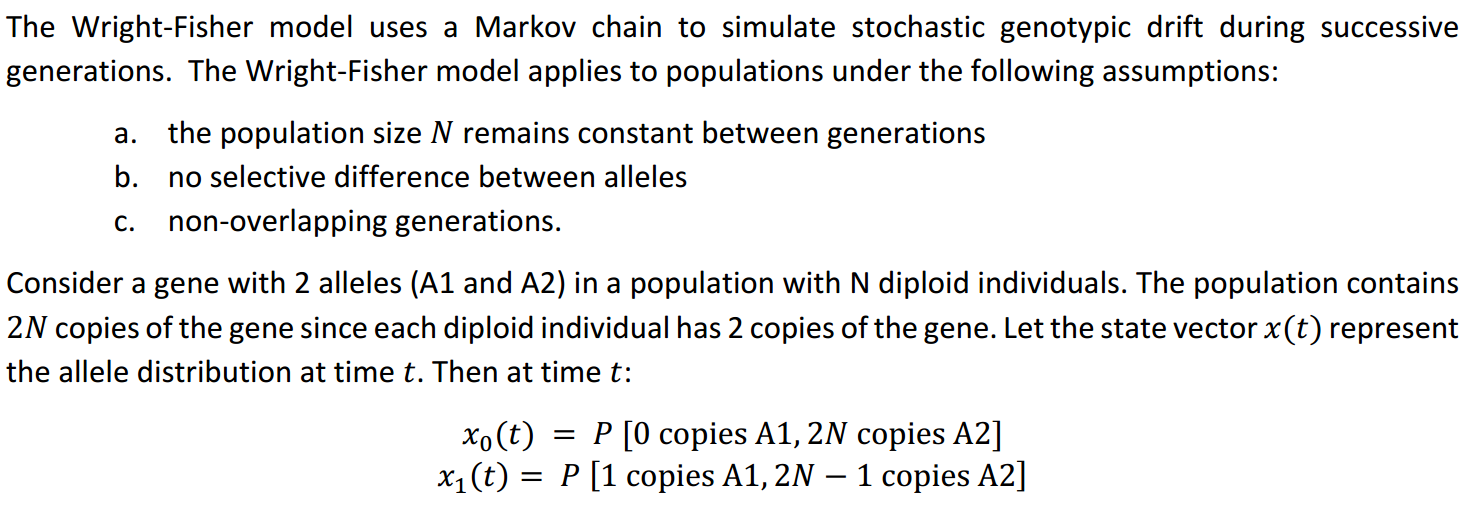
The condifence interval for p=0.2 is [0.1946,0.2049]

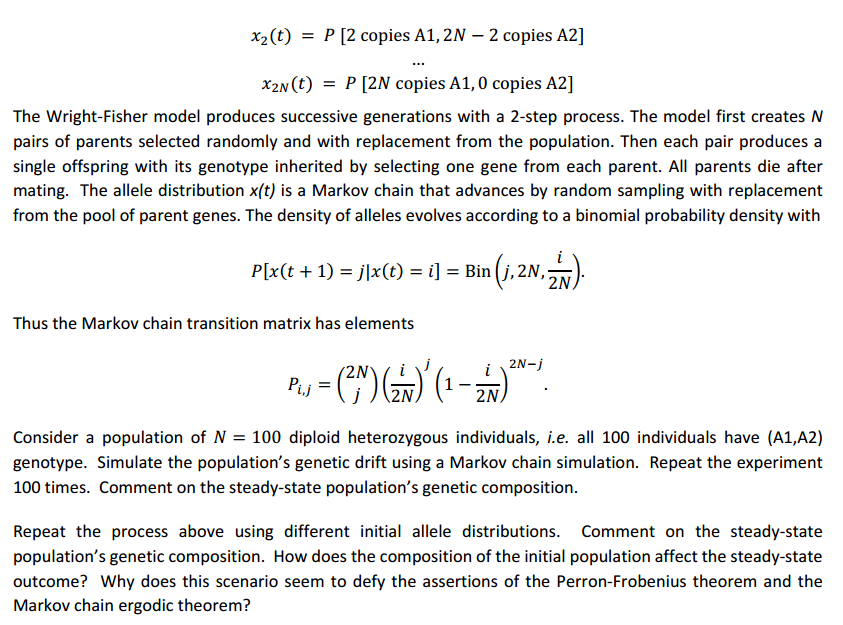


**Comment**:

The different from the question 1 is that steady mean of transferred packet changes to 1.4 while the original number is 1.5.

Compare the question **a** and **b**, we can conclude that the more symmetrical it is to transfer the packet, the higher efficiency it will be. Since the more symmetrical, the lower probability that the collision will occurs, which can results in higher frequency to transfer 2 packets without collision.

3. 



**Algorithm:**

When N=100, 201 condition is considered as the statement shows: …., select the initial state like [0…0,1], [0..0,1,1]…[1,0,0…0],[0,1,0,.0]. All of them are 201 dimensions.

Create the transition matrix as the statement of question shows.

Each time it changes, which can be denoted by the Output(i+1)=output(i), output(1)=input;

**Code: the code is revised from the one that professor shows in the piazza**

% Program to simulate a Markov chain

% The program assumes that the states are labeled 1, 2, ...

% Below is a sample, which you can change them according to the project

clear all; close all; clc;

input=eye(201)

N = 100; % number of 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

for state=1:201

n=2000; % number of time steps to take

output=zeros(n+1,2\*N+1); % clear out any old values

t=0:n; % time indices

output(1,:)=input(state,:); % generate first output value

i = 0;

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

steady(state,:)=output(end,:);

end

end

plot(steady(:,1)')

hold on

plot(steady(:,end)')

legend('(200A1,0A2)','(200A2,0A1)')

xlabel('initial state where the "1" locates');

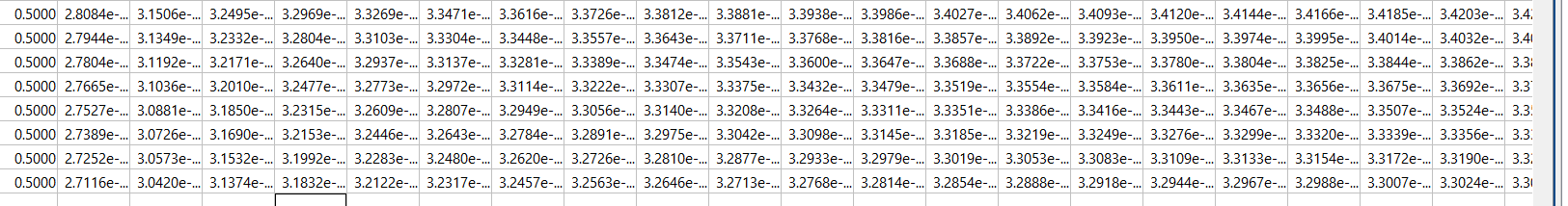
title('propability for state 1 (200A1, 0A2)] and (0A1, 200A2)');

ylabel('propability');

**results and comment:**

The steady result depends on the initial input:[, …]

All of the initial condition will result in a steady condition: [200A1, 0A2] or [0A1, 200A2]. E.g. The following result is when the copies of A1 are the same as the copies of A2, that is [100 A1, 100A2], which is state 101(). The steady result is as follows:

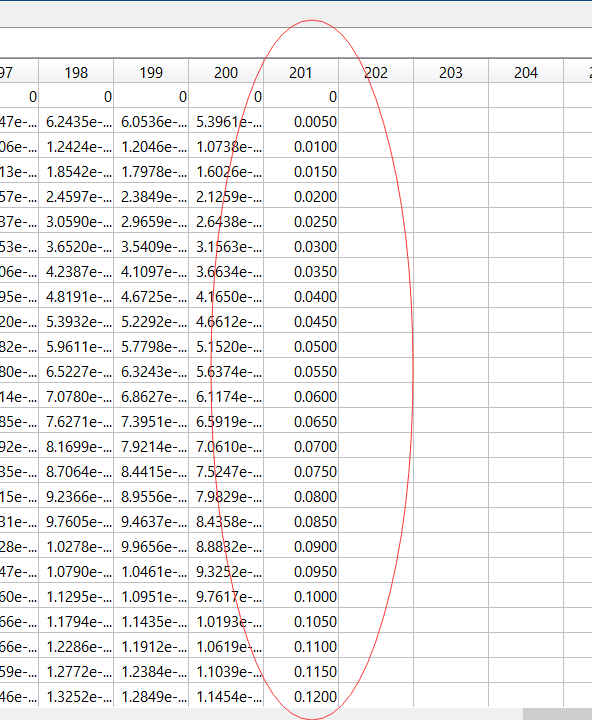
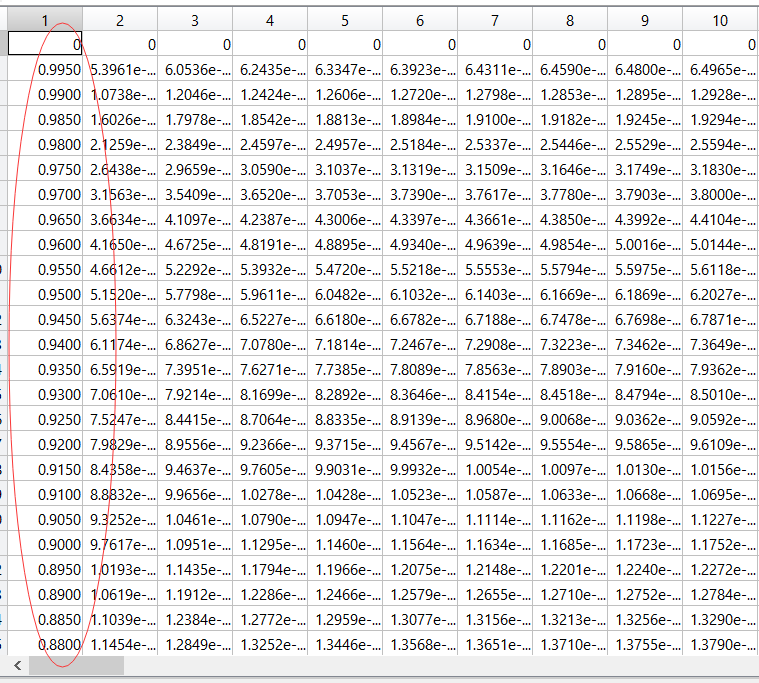


initial input = [00000..1]->steady result [00000,…1];

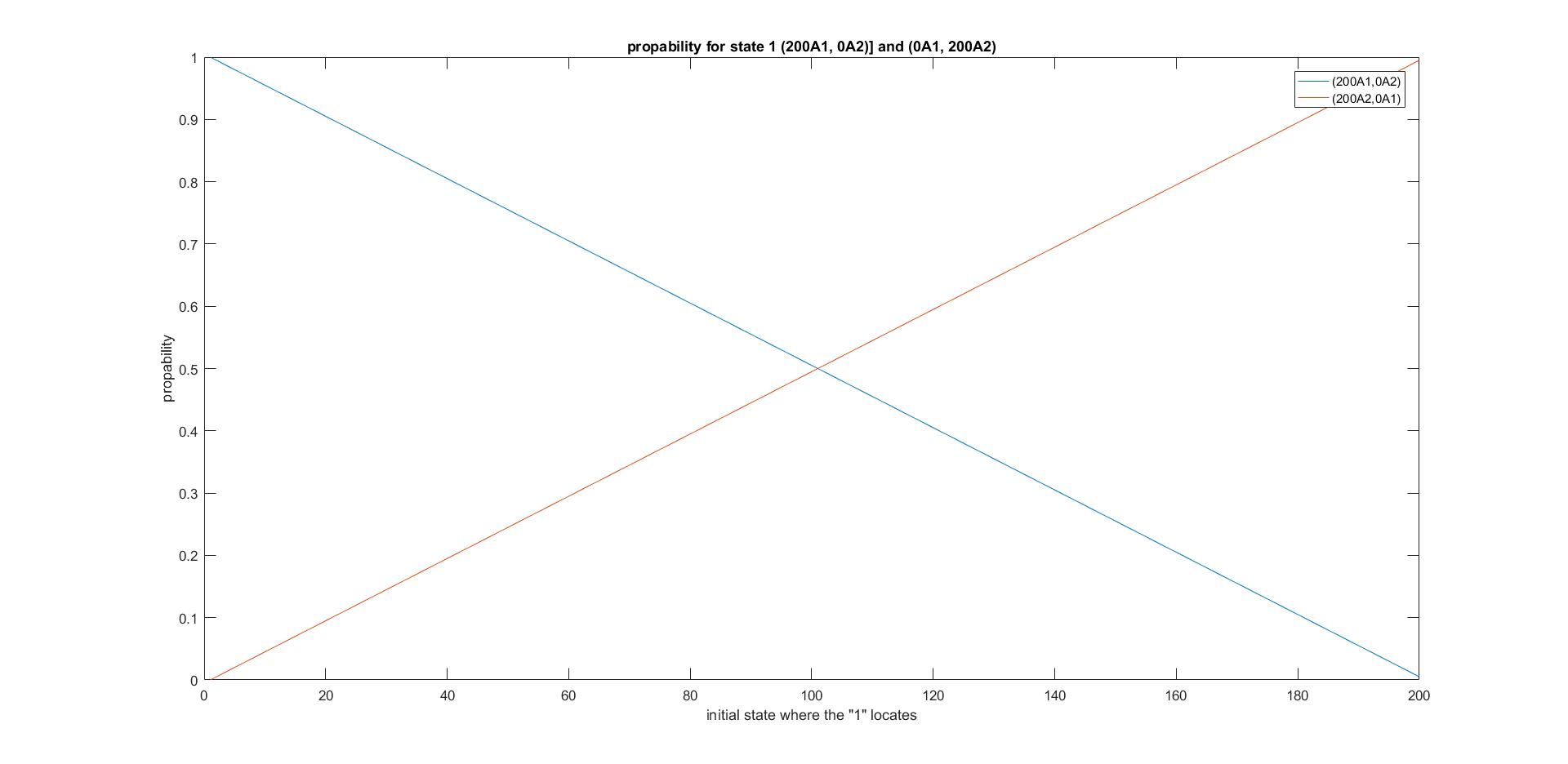
initial input = [1,0…..0]->steady result [1,0000…00];

initial input = [0,1,0,00000]->steady result [0.9950,000000,0.005]

initial input=[0000…0,1,0]->steady result [0005,000000,0.9950]



**…………………….**

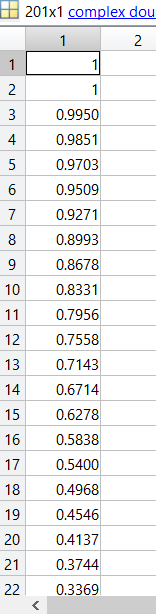
And we can find all of the middle values are zero. So we only consider the sides value that is the sate 1=[200A1, 0A2] and state 201 = [0A1, 200A2]. And a figure of the first coloumn and the last coloumn of steady result is obtained as the following shows:

And then find the steady distribution. We can find that there are only two kinds of distribution [200A1, 0A2] or [0 A1 or 200A2] when it is reach to steady distribution. And the probability of [200A1, 0A2] and [0A1, 200A2] when it is steady are as the above figure shows.

We can find the probability of steady distribution [200A1, 0 A2] and [0A1, 200A2] changes linearly with different initial distribution.

Perron-Frobenius theorem: Let P be a regular statistic matrix, suppose P is irreducible and aperiodic . Then P has a unique positive eigenoector with for

With the MATLAB command eig(P’), we can obtain a series of eignoetors(as the following figure shows). And each eignoetor has a unique convergent result. So it DOESN’T defy the Perro-Frobenius theory.



**Markov chain ergodic theorem: for every non-negative vector x∈, then if P satisfies the requirement for Peron-Fronisu**

Actually, this experiment does also doesn’t defy the Markov chain ergodic theorem. Because a discrete-time Markov chain is a sequence of random variables X1, X2, X3, ... with the Markov property, namely that the probability of moving to the next state depends only on the present state and not on the previous states

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