ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΜΠ

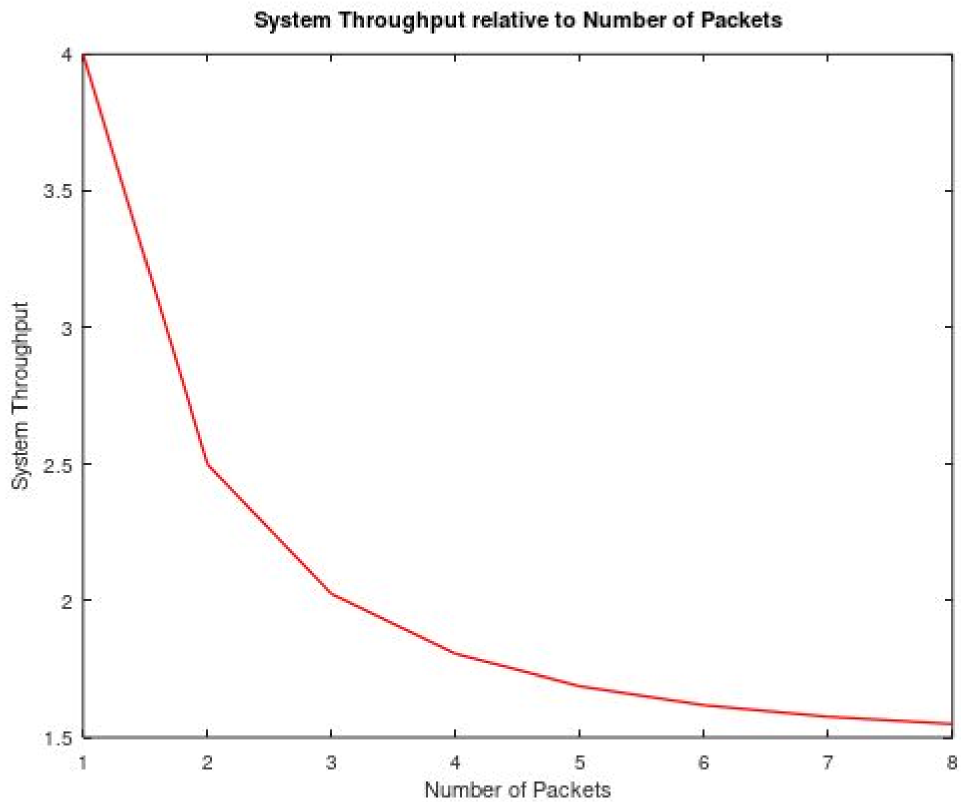
“QUEUING SYSTEMS”

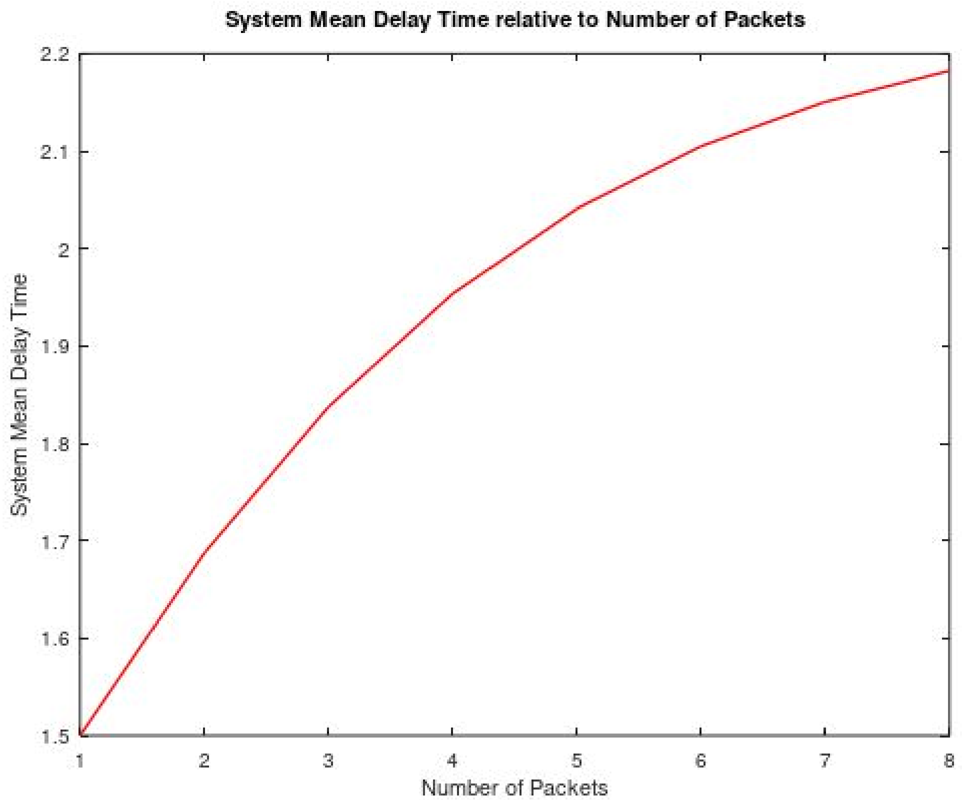
Ορφανουδάκης Φίλιππος 03113140

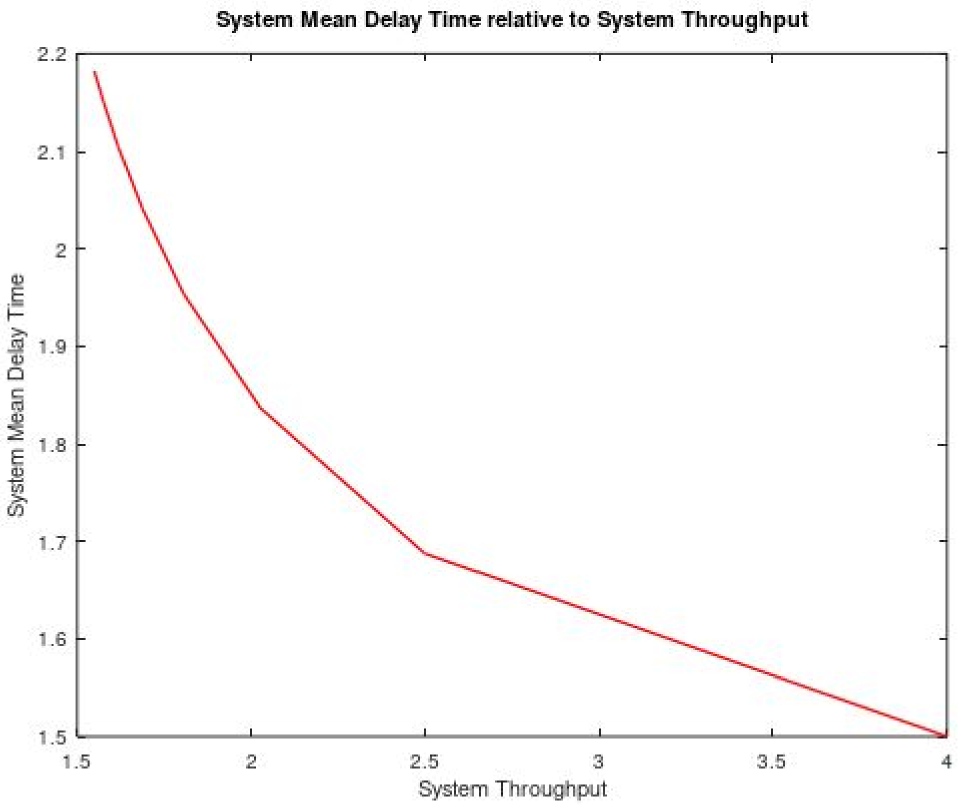
<u>6η Ομάδα Ασκήσεων</u>

<u>Μηχανισμός ελέγχου ροής παραθύρου</u>

(1)




System Mean Delay Time relative to Number of Packets
System Mean Delay Time
Number of Packets
2.2
2.1
2
1.9
1.8
1.7
1.6
1.5
1
2
3
4
5
6
7
8


System Mean Delay Time relative to System Throughput
System Mean Delay Time
System Throughput
2.2
2.1
2
1.9
1.8
1.7
1.6
1.5
1.5
2
2.5
3
3.5
4

(2)

Utilization

```
u =
{
  [1,1] =

     0.55385   0.27692   0.27692   0.27692   0.83077

  [1,2] =

     0.55385   0.27692   0.27692   0.27692   0.83077

  [1,3] =

     0.55385   0.27692   0.27692   0.27692   0.83077

  [1,4] =

     0.55385   0.27692   0.27692   0.27692   0.83077

  [1,5] =

     0.55385   0.27692   0.27692   0.27692   0.83077

}
```

Response Time

```
r =
{
  [1,1] =

     1.74074   0.65123   0.65123   0.65123   3.52778

  [1,2] =

     0.87037   0.32562   0.32562   0.32562   1.76389

  [1,3] =

     0.58025   0.21708   0.21708   0.21708   1.17593

  [1,4] =
```

```
    0.43519   0.16281   0.16281   0.16281   0.88194

  [1,5] =

    0.34815   0.13025   0.13025   0.13025   0.70556

}

Average numbers of packets

q =
{
  [1,1] =

    0.96410   0.36068   0.36068   0.36068   1.95385

  [1,2] =

    0.96410   0.36068   0.36068   0.36068   1.95385

  [1,3] =

    0.96410   0.36068   0.36068   0.36068   1.95385

  [1,4] =

    0.96410   0.36068   0.36068   0.36068   1.95385

  [1,5] =

    0.96410   0.36068   0.36068   0.36068   1.95385

}

Throughput


x =
{
  [1,1] =

    0.55385   0.55385   0.55385   0.55385   0.55385

  [1,2] =

    1.1077   1.1077   1.1077   1.1077   1.1077
```

[1,3] =

1.6615   1.6615   1.6615   1.6615   1.6615

[1,4] =

2.2154   2.2154   2.2154   2.2154   2.2154

[1,5] =

2.7692   2.7692   2.7692   2.7692   2.7692

}

Παρατηρούμε σταθερότητα στον Μεσο αριθμο΄πακέτων , και στο βαθμό χρησιμοποίησης ενώ πτώση έχουμε στο μέσο χρόνο εξυπηρέτησης και αύξηση στη ρυθμαπόδοση.
Η πτώση στό χρόνο εξυπηρέτησης είναι προφανής αφού αυξάνουμε τον ρυθμό εξυπηρέτησης, εξού και η αύξηση της ρυθμαπόδοσης.
Από διαφάνειες έχουμε τους εξής τύπους για τα μεγέθη που είναι σταθερά :

$$E[n_i] = \sum_{k=1}^{N} X_i^k \frac{G(N-k)}{G(N)}, \quad P(n_i \geq 1) = X_i G(N-1)/G(N)$$

Το Xi είναι ανάλογο του λόγου λ/μi οπότε είναι σταθερό και η διαδικασία παραγωγής του G(N) εξαρτάται καθαρά από τα Xi οπότε και αυτά είναι σταθερά.

Ο αλγόριθμος του Buzen

(1)
$\mu_1 = 2$, $\mu_2 = 1$, $p = 0.3$.

Χρησιμοποιώντας το θεώρημα **Gordon-Newell** έχουμε

$$\mu_j X_j = \sum_{i=1}^{M} \mu_i X_i \mathrm{p}_{ij}, \qquad j = 1, \ldots, N$$

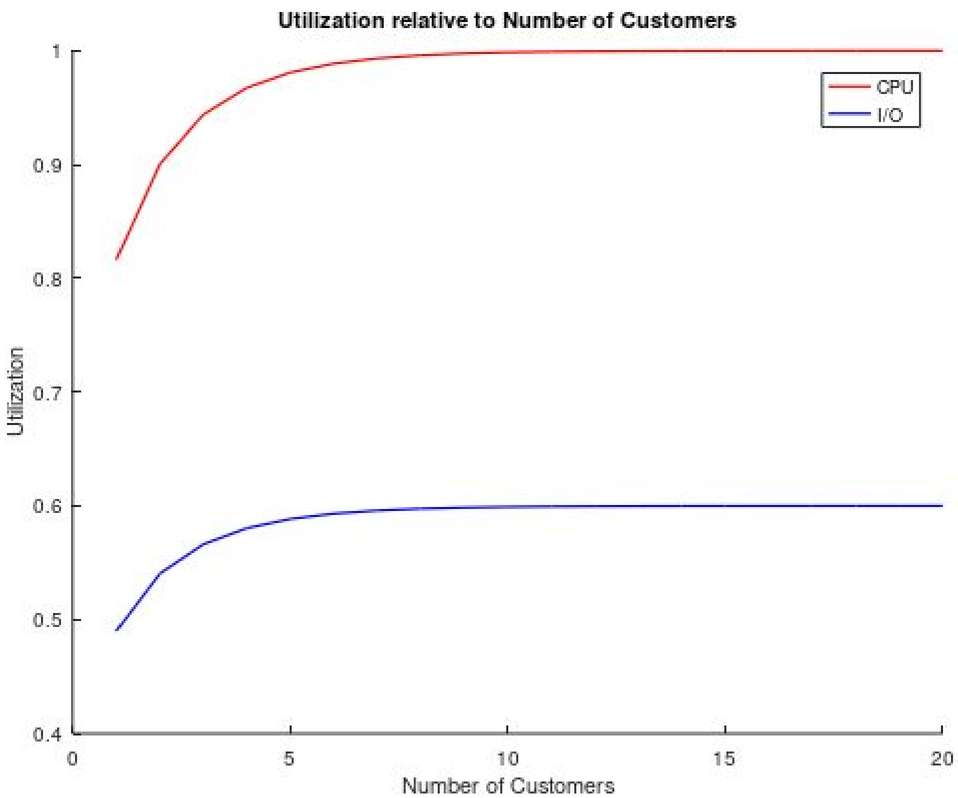
$\mu_1 X_1 = (1-p)\mu_1 X_1 + \mu_2 X_2 \Leftrightarrow 2X_1 = 1.4X_1 + X_2$
και $\mu_2 X_2 = p\mu_1 X_1 \Leftrightarrow 0.6X_1 = X_2$
από τα οποία για $X_1 = 1$ προκύπτει $X_2 = 0.6$.

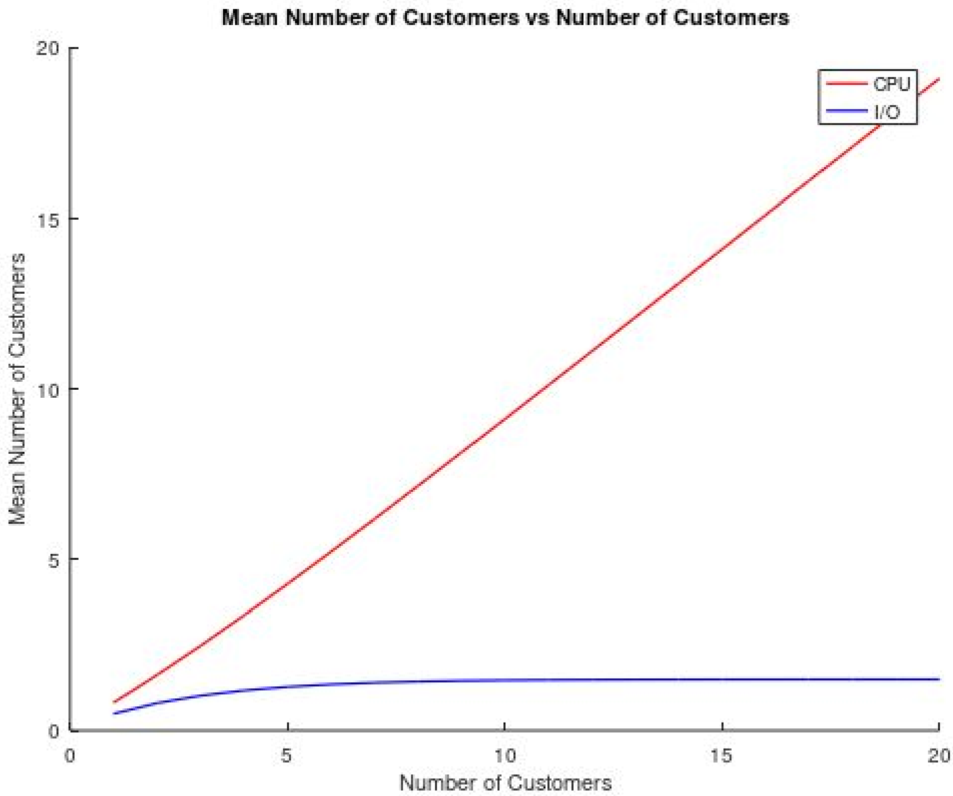
(2)

Η συνάρτηση που υλοποιήσαμε είναι η εξής :

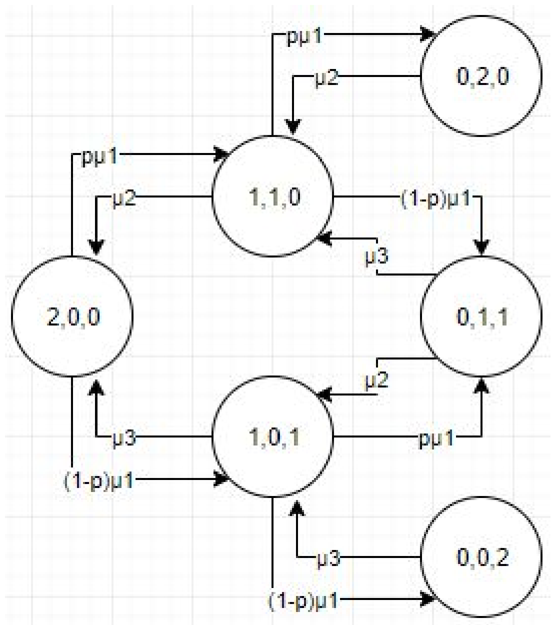
```
function [retval] = buzen (N, M, X)
  for m=1:M
    G(1,m) = 1;
  endfor
  for n=1:N+1
    G(n,1) = X(1)^n;
  endfor
  for n=2:N+1
    for m=2:M
      G(n,m) = G(n,m-1) + X(m)*G(n-1,m);
    endfor
  endfor
  retval = G(N+1,M);
endfunction
```

(3)





<u>Προσομοίωση σε κλειστό δίκτυο εκθετικών ουρών αναμονής</u>



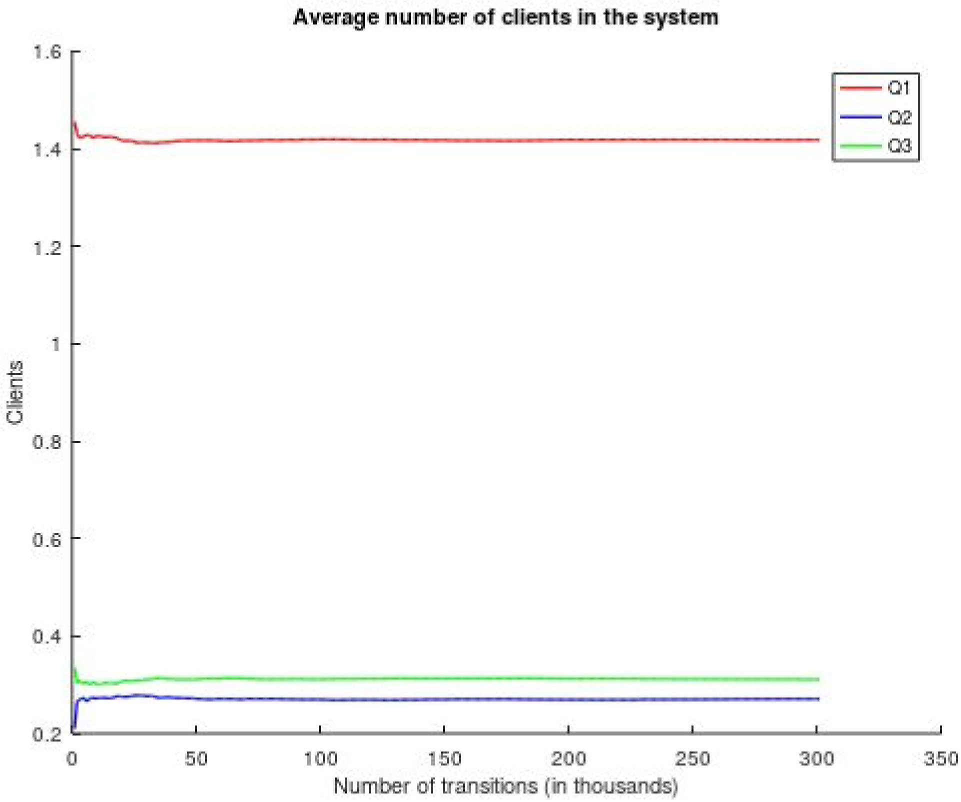
(1)

P(200)=0.551948

P(020)=0.0393522
P(002)=0.0501234
P(110)=0.147731
P(101)=0.16613
P(011)=0.0447148

(2)

clients_1=1.41776
clients_2=0.27115
clients_3=0.311092
sum of clients=2



Παρατηρούμε ότι το άθροισμα μας κάνει 2 όσους πελάτες έχουμε δηλαδή όντως μέσα στο σύστημα μας.

Ο κώδικας που χρησιμοποιήθηκε είναι ο εξής :

```
clc;
clear all;
close all;

pkg load queueing;

#Window Slider Control Mechanism
```

```
#1
N = 1:8;
mu = [1, 2, 2, 2, 2/3]; #lambda = 1

S = 1./mu;

P = [0 1 0 0 0;
     0 0 1 0 0;
     0 0 0 1 0;
     0 0 0 0 1;
     1 0 0 0 0;];

V = qncsvisits(P, 1);

for n = 1 : length(N)
  [U R Q X] = qnclosed(n, S, V);
  gamma(n) = R(1)/Q(1);
  T(n) = R(2)+R(3)+R(4);
endfor


figure(1);
plot(N, gamma, 'r',"linewidth",1.3);
xlabel("Number of Packets");
ylabel("System Throughput");
title("System Throughput relative to Number of Packets");

figure(2);
plot(N, T, 'r',"linewidth",1.3);
xlabel("Number of Packets");
ylabel("System Mean Delay Time");
title("System Mean Delay Time relative to Number of Packets");

figure(3);
plot(gamma, T, 'r',"linewidth",1.3);
xlabel("System Throughput");
ylabel("System Mean Delay Time");
title("System Mean Delay Time relative to System Throughput");

n=4
#2
for k = 1 : 5
  mu = k.*[1, 2, 2, 2, 2/3];

  S = 1./mu;
```

```
  #P is the same as above

  V = qncsvisits(P, 1);

    [U R Q X] = qnclosed(n, S, V);
    u{k} = U
    r{k} = R
    q{k} = Q
    x{k} = X

endfor

display(u);
display(r);
display(q);
display(x);


function [retval] = buzen (N, M, X)
  for m=1:M
    G(1,m) = 1;
  endfor
  for n=1:N+1
    G(n,1) = X(1)^n;
  endfor
  for n=2:N+1
    for m=2:M
      G(n,m) = G(n,m-1) + X(m)*G(n-1,m);
    endfor
  endfor
  retval = G(N+1,M);
endfunction

#Buzen Algorithm
#3
N = 20;
M = 2;
X = [1,0.6];

G = buzen(N, M, X);
for i=1:N
  U1(i) = X(1)*buzen(i,M,X)/buzen(i+1,M,X);
  U2(i) = X(2)*buzen(i,M,X)/buzen(i+1,M,X);
  En1(i) = 0;
  En2(i) = 0;
  for j=1:i
```

```
    En1(i) = En1(i) + X(1)^j*buzen(i-j+1,M,X)/buzen(i+1,M,X);
    En2(i) = En2(i) + X(2)^j*buzen(i-j+1,M,X)/buzen(i+1,M,X);
  endfor
endfor

k = 1:20;

figure(4);
hold on;
plot(k,U1,'r','linewidth',1.2);
plot(k,U2,'b','linewidth',1.2);
hold off;
title("Utilization relative to Number of Customers");
xlabel("Number of Customers");
ylabel("Utilization");
legend("CPU", "I/O");

figure(5);
hold on;
plot(k,En1,'r','linewidth',1.2);
plot(k,En2,'b','linewidth',1.2);
hold off;
title("Mean Number of Customers vs Number of Customers");
xlabel("Number of Customers");
ylabel("Mean Number of Customers");
legend("CPU", "I/O");

#Closed network simulation
mu1 = 2;
mu2 = 3;
mu3 = 4;
p = 0.4;

arrivals(002) = 0;
arrivals(020) = 0;
arrivals(200) = 0;
arrivals(110) = 0;
arrivals(101) = 0;
arrivals(011) = 0;
total_arrivals = 0;

% threshold definition
threshold = mu1/(mu1 + mu2);
% system starts at state 200
current_state = 200;
% count the time steps of the simulation
```

```
steps = 0;

previous_mean1 = 0;
previous_mean2 = 0;
previous_mean3 = 0;

% times checked for convergence
times = 0;

while true
  steps = steps + 1;
  % every 1000 steps check for convergence
  if mod(steps,1000) == 0
    times = times + 1;

    % total time in every state
    T200 = 1/mu1 * arrivals(200);
    T020 = 1/mu2 * arrivals(020);
    T002 = 1/mu3 * arrivals(002);
    T011 = 1/(mu2 + mu3) * arrivals(011);
    T101 = 1/(mu3 + mu1) *arrivals(101);
    T110 = 1/(mu2 + mu1) * arrivals(110);

    % total time in all states
    total_time = T200 + T020 + T002 + T110 + T101 + T011;
    % Probability of every state
    Pnew(200) = T200/total_time;
    Pnew(020) = T020/total_time;
    Pnew(002) = T002/total_time;
    Pnew(110) = T110/total_time;
    Pnew(101) = T101/total_time;
    Pnew(011) = T011/total_time;

    % mean number of clients in queues 1, 2 and 3
    current_mean1 = Pnew(110) + Pnew(101) + 2 * Pnew(200);
    current_mean2 = Pnew(110) + Pnew(011) + 2 * Pnew(020);
    current_mean3 = Pnew(101) + Pnew(011) + 2 * Pnew(002);

    clients_1(times) = current_mean1;
    clients_2(times) = current_mean2;
    clients_3(times) = current_mean3;

    % check all queues for convergence
    if abs(current_mean1 - previous_mean1)<0.00001 && abs(current_mean2 -
previous_mean2) < 0.00001 && abs(current_mean3 - previous_mean3) < 0.00001
      break;
```

```
  endif

  if steps > 300000
    break;
  endif

  previous_mean1 = current_mean1;
  previous_mean2 = current_mean2;
  previous_mean3 = current_mean3;

endif

arrivals(current_state) = arrivals(current_state) + 1;
total_arrivals = total_arrivals + 1;

% get a random number from uniform distribution
random_number = rand(1);
if current_state == 002
  current_state = 101;
elseif current_state == 020
  current_state = 110;
elseif current_state == 200
  threshold = p;
  if random_number < threshold
    current_state = 110;
  else
    current_state = 101;
  endif
elseif current_state == 110
  threshold1 = mu2/(mu2 + mu1);
  threshold2 = (mu2 + p*mu1)/(mu2 + mu1);
  if random_number < threshold1
    current_state = 200;
  elseif random_number < threshold2
    current_state = 020;
  else
    current_state = 011;
  endif
elseif current_state == 101
  threshold1 = mu3/(mu3 + mu1);
  threshold2 = (mu3 + p*mu1)/(mu3 + mu1);
  if random_number < threshold1
    current_state = 200;
  elseif random_number < threshold2
    current_state = 011;
  else
```

```
    current_state = 002;
   endif
  else  #if current_state == 011
   threshold = mu2/(mu2 + mu3);
   if random_number < threshold
    current_state = 101;
   else
    current_state = 110;
   endif
  endif

endwhile

fprintf("P(200)=%d\n",Pnew(200));
fprintf("P(020)=%d\n",Pnew(020));
fprintf("P(002)=%d\n",Pnew(002));
fprintf("P(110)=%d\n",Pnew(110));
fprintf("P(101)=%d\n",Pnew(101));
fprintf("P(011)=%d\n",Pnew(011));
fprintf("clients_1=%d\n",clients_1(end));
fprintf("clients_2=%d\n",clients_2(end));
fprintf("clients_3=%d\n",clients_3(end));
fprintf("sum of clients=%d\n",clients_1(end)+clients_2(end)+clients_3(end));

figure(6);
hold on;
plot(clients_1,'r',"linewidth",1.3);
plot(clients_2,'b',"linewidth",1.3);
plot(clients_3,'g',"linewidth",1.3);
xlabel("Number of transitions (in thousands)");
ylabel("Clients");
title("Average number of clients in the system");
legend("Q1","Q2","Q3");
hold off;
```