ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΕΜΠ

“QUEUING SYSTEMS”

Ορφανουδάκης Φίλιππος 03113140

<u>3η Ομάδα Ασκήσεων</u>

<u>Σύγκριση συστημάτων M/M/1 και M/D/1</u>

1. Δεν μας δίνεται κάποιος περιορισμός στην χωρητικότητα της ουράς μας οπότε θεωρούμε ότι δεν θα έχουμε απώλειες επομένως P{Blocking}=0 => γ=λ.

   Άρα σε κατάσταση ισορροπίας από νόμο Little θα έχω το εξής :

- Μέσος χρόνος καθυστέρησης E(T)= $E[n(t)]/\gamma = E[n(t)]/\lambda = 1/\mu + 1/2 * (\rho/\mu(1-\rho))$
- Μέσος χρόνος αναμονής

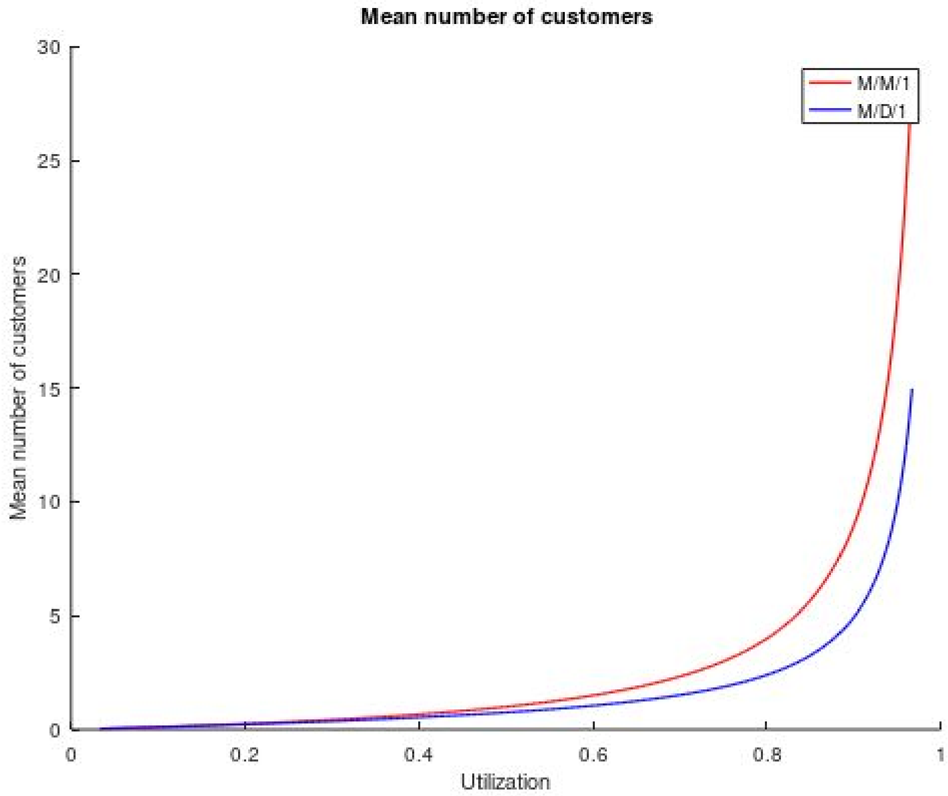
E(W)$= E[n_q(t)]/\gamma = (E[n(t)] - E[n_s(t)])/\gamma = 1/\mu + 1/2 * (\rho/\mu(1-\rho)) - \gamma/\gamma\mu = 1/2 * (\rho/\mu(1-\rho))$

Για να είναι εργοδική η απαραίτητη προυπόθεση είναι ρ<1.

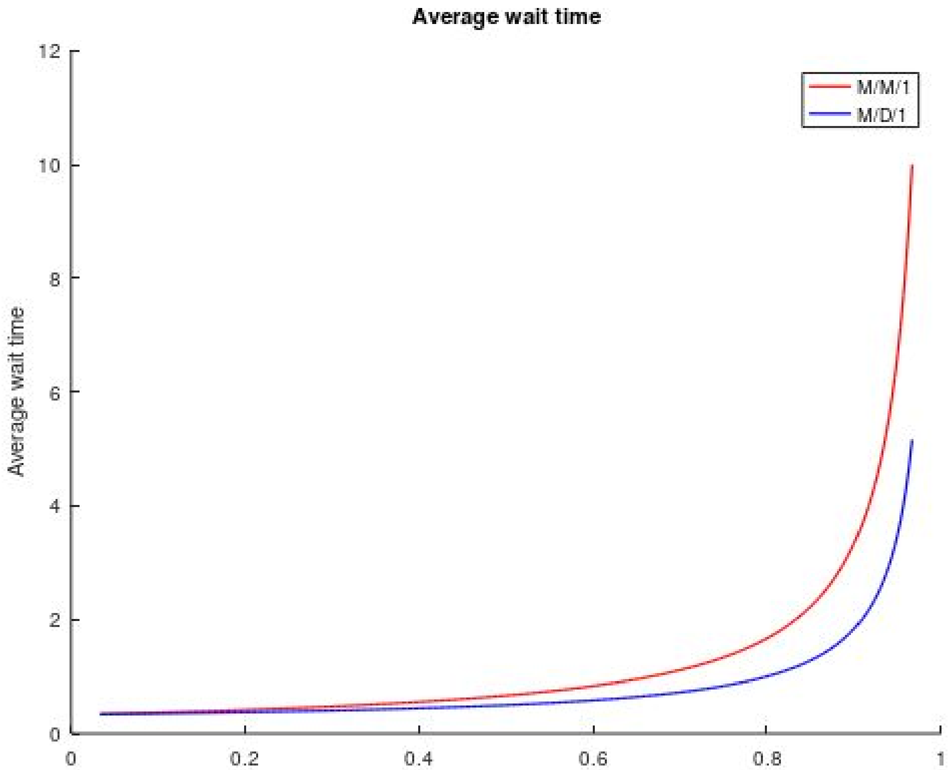
2.

```
function [U, R, Q, X] = qsmd1(lambda, mu)
  U = lambda/mu;
  if(U >= 1)
    error ("System is not ergodic");
  endif
  R = 1/mu + 1/2*U/(mu*(1-U));
  Q = U + 1/2*U^2/(1-U);
  X = lambda;
endfunction
```

3. α)


Mean number of customers
M/M/1
M/D/1
Mean number of customers
Utilization
0
5
10
15
20
25
30
0
0.2
0.4
0.6
0.8
1

β)


Average wait time
M/M/1
M/D/1
Average wait time
Utilization
0
2
4
6
8
10
12
0
0.2
0.4
0.6
0.8
1

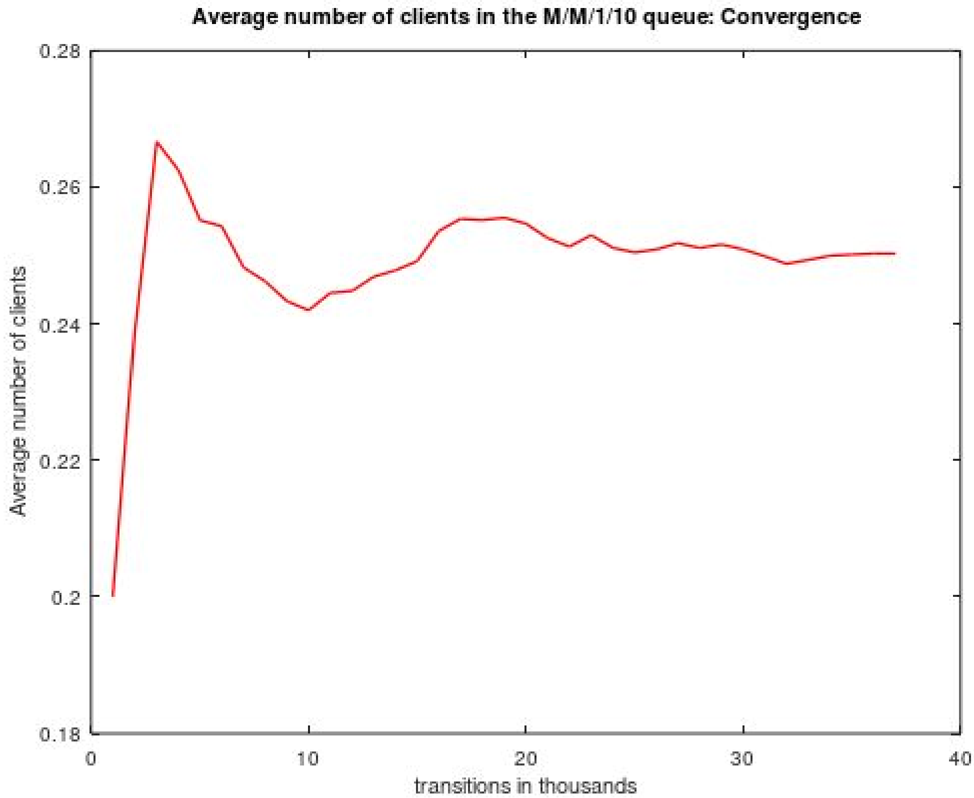
Με βάση τα παραπάνω διαγράμματα το σύστημα M/D/1 παρουσιάζει καλύτερους, δηλάδη μικρότερους χρόνους εξυπηρέτησης και καλύτερο - μικρότερο αριθμό πελατών κατα μέσο όρο στην ουρά. Επομένως θα προτιμήσουμε το M/D/1.

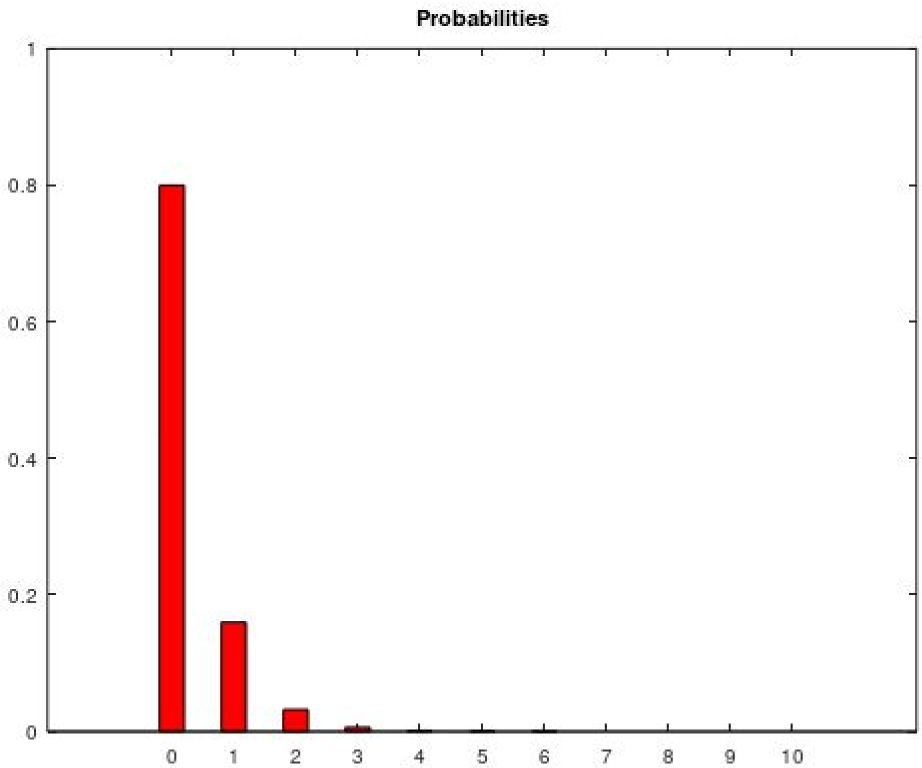
Προσομοίωση συστήματος M/M/1/10

1) Ο κώδικάς για το debugging μπορεί να παρατηρηθεί στο τέλος του αρχείου
2)

λ=1

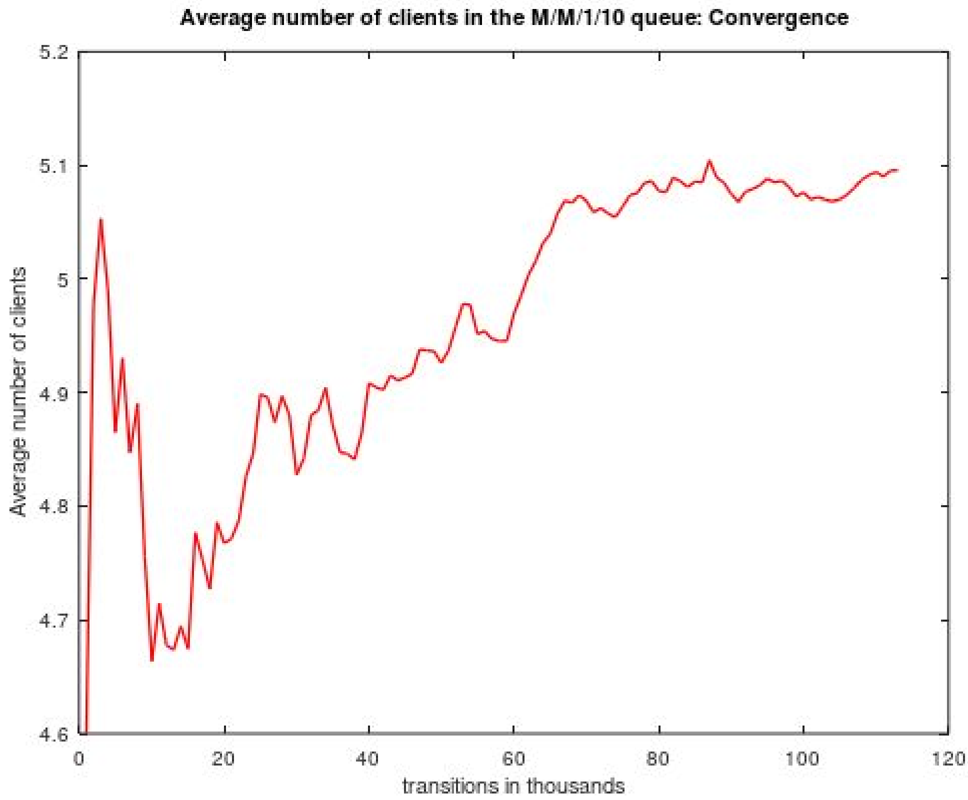
```
Ergodic probabilities for lambda = 1
P(0) = 0.800216
P(1) = 0.159568
P(2) = 0.0322162
P(3) = 0.00621622
P(4) = 0.0012973
P(5) = 0.000432432
P(6) = 5.40541e-05
P(7) = 0
P(8) = 0
P(9) = 0
P(10) = 0
Pblock for lambda = 1: Pblock = 0
Mean number of clients for lambda = 1: E[n(t)] = 0.250324
Mean wait time for lambda = 1: E[T] = 0.250324
```

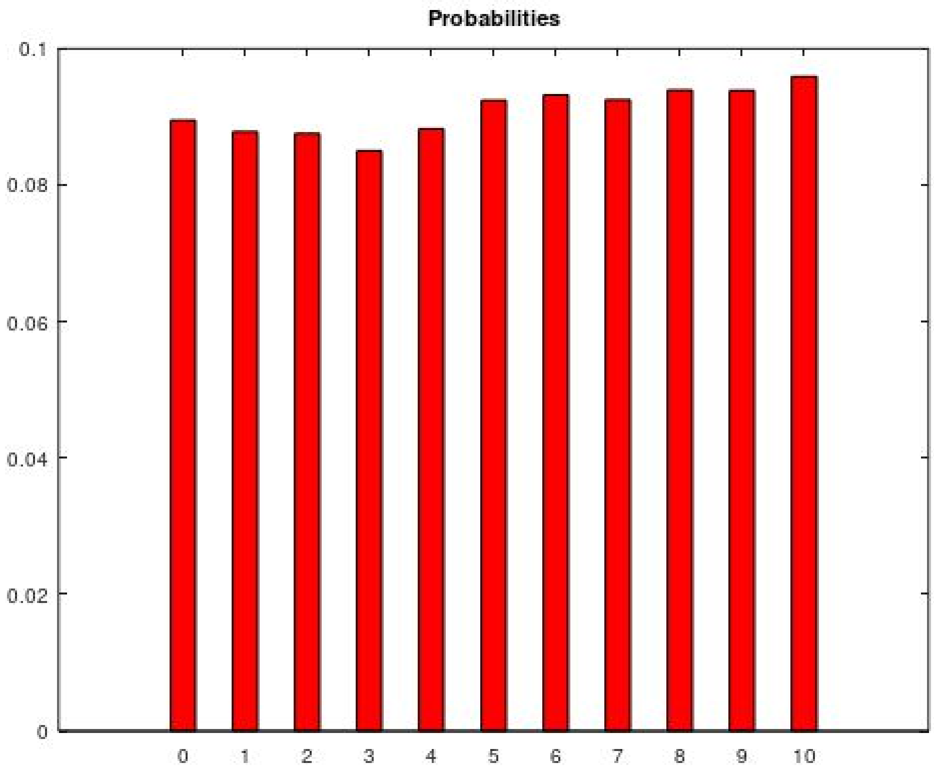

Average number of clients in the M/M/1/10 queue: Convergence
Average number of clients
transitions in thousands
0.28
0.26
0.24
0.22
0.2
0.18
0
10
20
30
40


Probabilities
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
7
8
9
10

λ=5

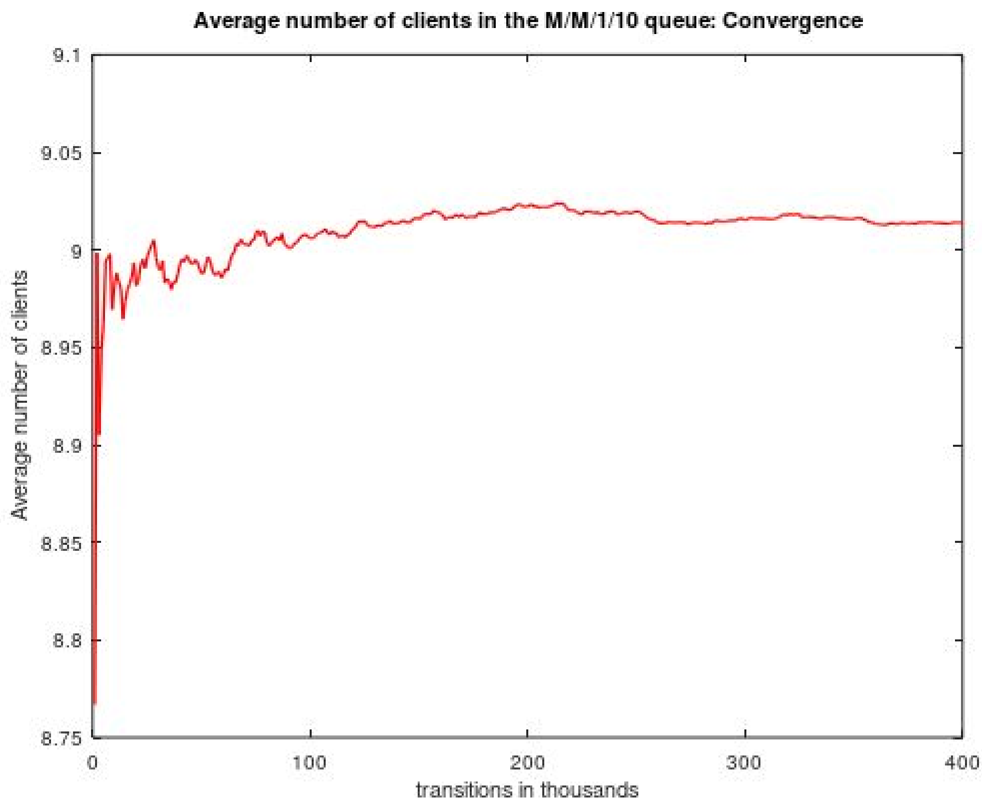
```
Ergodic probabilities for lambda = 5
P(0) = 0.0894921
P(1) = 0.0878239
P(2) = 0.0875543
P(3) = 0.0850268
P(4) = 0.0882452
P(5) = 0.0924241
P(6) = 0.0932161
P(7) = 0.0925083
P(8) = 0.0939238
P(9) = 0.0938732
P(10) = 0.0959121
Pblock for lambda = 5: Pblock = 0.0959121
Mean number of clients for lambda = 5: E[n(t)] = 5.09534
Mean wait time for lambda = 5: E[T] = 1.12718
```

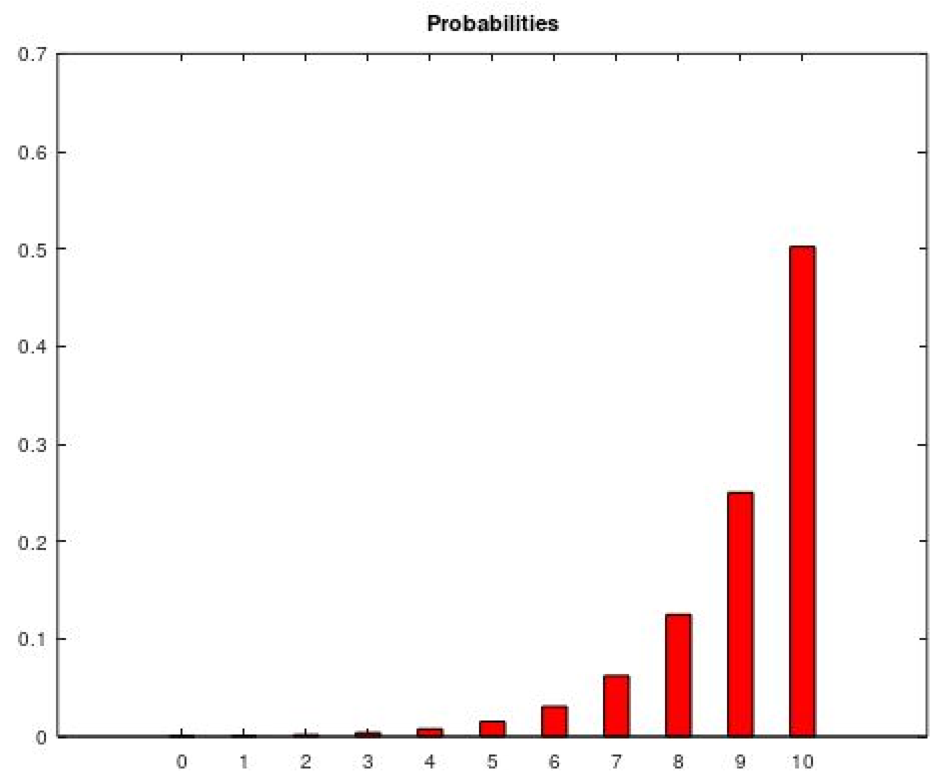




λ=10

```
Ergodic probabilities for lambda = 10
P(0) = 0.000479279
P(1) = 0.000853715
P(2) = 0.00174487
P(3) = 0.0038567
P(4) = 0.00772837
P(5) = 0.0154455
P(6) = 0.0308311
P(7) = 0.0619393
P(8) = 0.124785
P(9) = 0.250112
P(10) = 0.502224
Pblock for lambda = 10: Pblock = 0.502224
Mean number of clients for lambda = 10: E[n(t)] = 9.01415
Mean wait time for lambda = 10: E[T] = 1.81088
```


Average number of clients in the M/M/1/10 queue: Convergence
Average number of clients
transitions in thousands
9.1
9.05
9
8.95
8.9
8.85
8.8
8.75
0
100
200
300
400


Probabilities
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
1
2
3
4
5
6
7
8
9
10

3) Όσο αυξάνεται το λ και επομένως αυξάνεται το ρ=λ/μ ήταν αναμενόμενο να έχουμε πιο αργή ταχύτητα σύγκλισης στις εργοδικές μας πιθανότητες , όπως επίσης έχουμε και πολύ χειρότερη συμπεριφορα του συστήματος μας.

Όπως βλέπουμε στα διαγράμματα και στο πόρισμα που βγάλαμε τις λιγότερες μεταβάσεις τις χρειάζεται ο λ=1 , όπου για τις πρώτες περίπου 30000 μεταβάσεις δεν έχει βρεθεί σε εργοδική κατάσταση , οπότε θα μπορούσαμε να τις αγνοήσουμε για να επιτύχουμε πιο γρήγορη σύγκλιση .

Προσομοίωση συστήματος M/M/1/5 με μεταβλητό μέσο ρυθμό εξυπηρέτησης

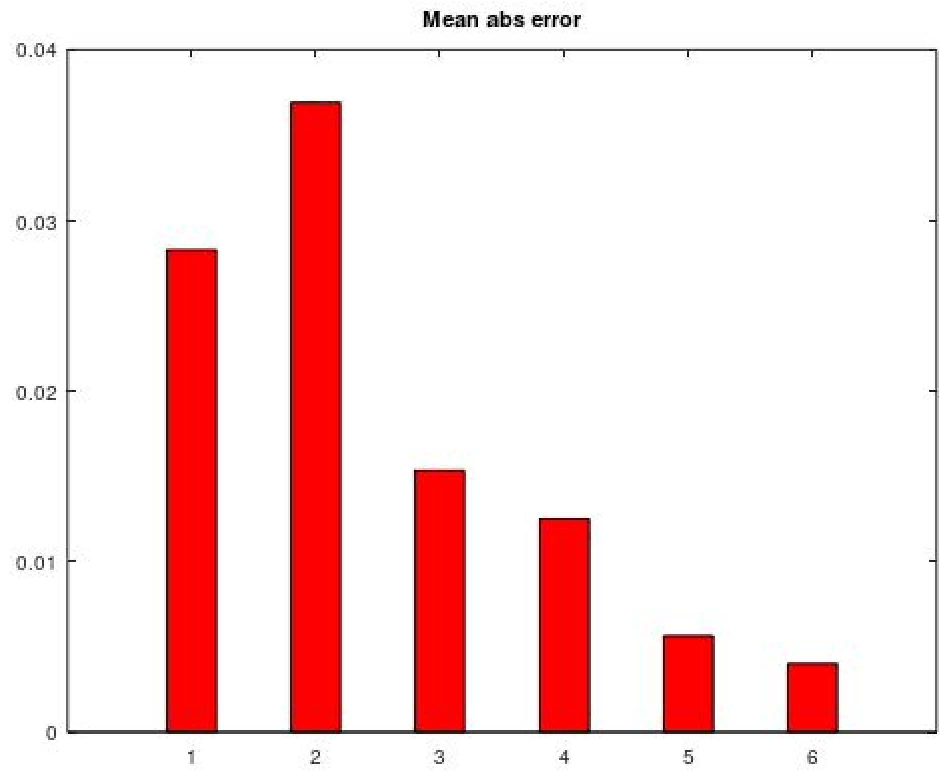
1)
Τα αποτελέσματα του πακέτου του Octave :

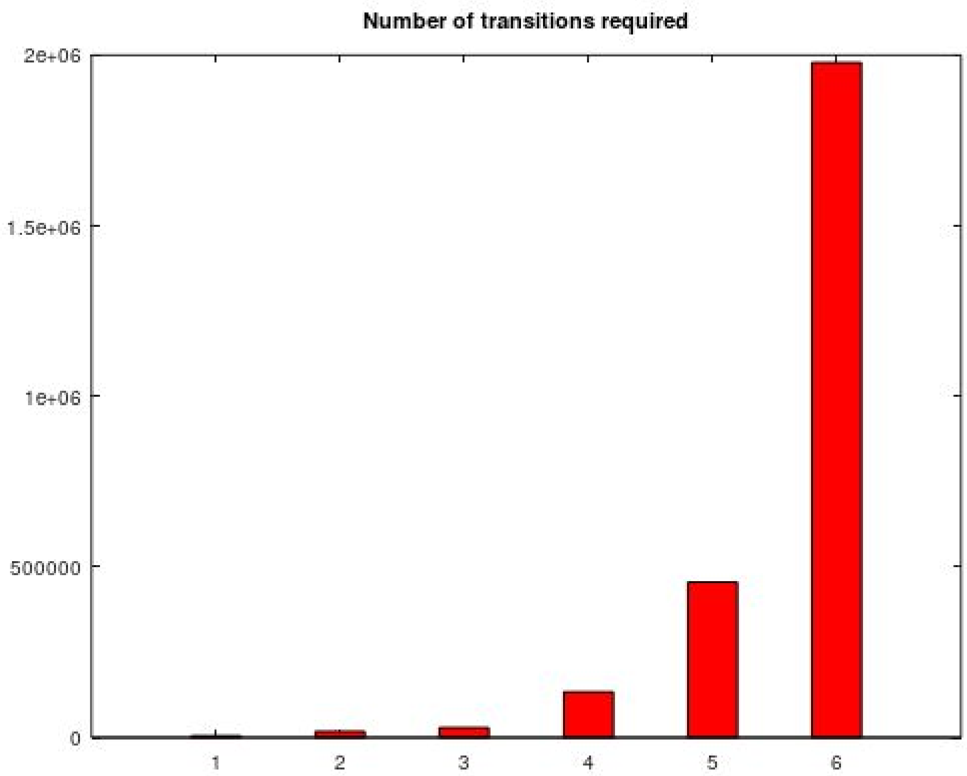
```
Ergodic probabilities of M/M/1/5 as calculated by queuing package:
   0.162933   0.244399   0.244399   0.183299   0.109980   0.054990
Mean clients of M/M/1/5 as calculated by queuing package:
 1.9980
```

2)

Αυτές είναι οι τιμές που υπολογίζουμε στη προσομοίωση μας για κριτήριο σύγκλισης 0.00001%:

```
P(0) = 0.161896
P(1) = 0.243772
P(2) = 0.245086
P(3) = 0.183918
P(4) = 0.110352
P(5) = 0.054975
Pblock = 0.054975
Mean number of clients: E[n(t)] = 2.00198
```


Mean abs error
0.04
0.03
0.02
0.01
0
1
2
3
4
5
6


Number of transitions required
2e+06
1.5e+06
1e+06
500000
0
1
2
3
4
5
6

- Αρχικά να αναφέρουμε πως όσο πιο αυστηρό το κριτήριο σύγκλισης έχουμε μικρότερο σφάλμα αλλά τόσο πιο χρονοβόρα είναι η προσομοίωση , πιο συγκεκριμένα το καλύτερο trade off που θα προτείνουμε είναι για το κριτήριο 0.00001% , ύστερα απο αυτό έχουμε πολύ μεγάλες καθυστερήσεις και στον κώδικα μας θα παρατηρήσετε ότι δεν τα υπολογίζουμε καθώς ξεπερνούσε πολύ μεγάλα χρονικά διαστήματα

- Μπορούμε να τοποθετήσουμε σαν δικλείδα ασφαλείας ότι θα τελείωσει η προσωμοίωση μας, τον αριθμό των μεταβάσεων

Ο κώδικας που χρησιμοποιήθηκε :

```
clc;
clear all;
close all;

pkg load queueing;


function [U, R, Q, X] = qsmd1(lambda, mu)
  U = lambda/mu;
  if(U >= 1)
    error ("System is not ergodic");
  endif
  R = 1/mu + 1/2*U/(mu*(1-U));
  Q = U + 1/2*U^2/(1-U);
  X = lambda;
endfunction


#Comparison between M/M/1 and M/D/1
#3
lambda = 0.1:0.01:2.9;
mu = 3;
for i = 1:columns(lambda)
  [Um(i), Rm(i), Qm(i), Xm(i), pm0(i)] = qsmm1(lambda(i), mu);
  [Ud(i), Rd(i), Qd(i), Xd(i)] = qsmd1(lambda(i), mu);
endfor

colors = "rbgm";
figure(1);
```

```
hold on;
plot(Um,Qm,colors(1),"linewidth",1.2);
plot(Ud,Qd,colors(2),"linewidth",1.2);
hold off;
title("Mean number of customers");
xlabel("Utilization");
ylabel("Mean number of customers");
legend("M/M/1","M/D/1");

figure(2);
hold on;
plot(Um,Rm,colors(1),"linewidth",1.2);
plot(Ud,Rd,colors(2),"linewidth",1.2);
hold off;
title("Average wait time");
xlabel("Utilization");
ylabel("Average wait time");
legend("M/M/1","M/D/1");

% M/M/1/10 simulation. We will find the probabilities of the first states.
% Note: Due to ergodicity, every state has a probability >0.

lambda = [1,5,10];
mu = 5;

for j=1:1:3
  %disp("Starting trace...");
  states = [0,1,2,3,4,5,6,7,8,9,10];
  rand("seed",1);
  total_arrivals = 0; % to measure the total number of arrivals
  blocked_arrivals = 0;
  current_state = 0;  % holds the current state of the system
  previous_mean_clients = 0; % will help in the convergence test
  index = 0;

  threshold = lambda(j)/(lambda(j) + mu); % the threshold used to calculate probabilities
  arrivals = zeros(1,11);

  transitions = 0; % holds the transitions of the simulation in transitions steps

  while transitions >= 0
    transitions = transitions + 1; % one more transitions step

    if mod(transitions,1000) == 0 % check for convergence every 1000 transitions steps
      index = index + 1;
      for i=1:1:length(arrivals)
```

```
      P(i) = arrivals(i)/total_arrivals; % calculate the probability of every state in the system
    endfor
    Pblock = blocked_arrivals / total_arrivals;

    mean_clients = 0; % calculate the mean number of clients in the system
    for i=1:1:length(arrivals)
      mean_clients = mean_clients + (i-1).*P(i);
    endfor
    gamma = lambda(j)*(1-Pblock);
    mean_wait_time = mean_clients / gamma;

    to_plot(index) = mean_clients;

    if abs(mean_clients - previous_mean_clients) < 0.00001 || transitions > 1000000 %
convergence test
      break;
    endif

    previous_mean_clients = mean_clients;

  endif

  random_number = rand(1); % generate a random number (Uniform distribution)
  if current_state == 0 % arrival
    total_arrivals = total_arrivals + 1;
    arrivals(current_state + 1) = arrivals(current_state + 1) + 1; % increase the number of
arrivals in the current state
    %if transitions <= 30
      %fprintf("State: 0, arrival, arrivals in this state: %d\n", arrivals(1));
    %endif
    current_state = current_state + 1;
  else
    if random_number < threshold % arrival
      if current_state == 10 % blocked
        total_arrivals = total_arrivals + 1;
        arrivals(current_state + 1) = arrivals(current_state + 1) + 1;
        blocked_arrivals = blocked_arrivals + 1;
        %if transitions <= 30
          %fprintf("State: 10, arrival, arrivals in this state: %d\n", arrivals(11));
        %endif
      else
        total_arrivals = total_arrivals + 1;
        arrivals(current_state + 1) = arrivals(current_state + 1) + 1;
        %if transitions <= 30
          %fprintf("State: %d, arrival, arrivals in this state: %d\n", current_state,
arrivals(current_state+1));
```

```
        %endif
        current_state = current_state + 1;
      endif
    else % departure
      %if transitions <= 30
        %fprintf("State: %d, departure, arrivals in this state: %d\n", current_state,
arrivals(current_state+1));
      %endif
      current_state = current_state - 1;
    endif
  endif
endwhile
fprintf("Ergodic probabilities for lambda = %d\n", lambda(j));
for i=1:1:length(arrivals)
  fprintf("P(%d) = %d\n", i-1, P(i));
endfor
fprintf("Pblock for lambda = %d: Pblock = %d\n", lambda(j), Pblock);
fprintf("Mean number of clients for lambda = %d: E[n(t)] = %d\n", lambda(j), mean_clients);
fprintf("Mean wait time for lambda = %d: E[T] = %d\n\n", lambda(j), mean_wait_time);

figure(2*j+1);
plot(to_plot,"r","linewidth",1.3);
title("Average number of clients in the M/M/1/10 queue: Convergence");
xlabel("transitions in thousands");
ylabel("Average number of clients");

figure(2*j+2);
bar(states, P,'r',0.4);
title("Probabilities");
endfor


%M/M/1/5 with variable service rate simulation
lambda = 3;
mu = [2,3,4,5,6];

Q = ctmcbd(lambda.*ones(1,5), mu);
p = ctmc(Q);
fprintf("Ergodic probabilities of M/M/1/5 as calculated by queuing package:\n");
disp(p);
mean_clients_real = 0;
for i=1:1:length(p)
  mean_clients_real = mean_clients_real + p(i)*(i-1);
endfor
fprintf("Mean clients of M/M/1/5 as calculated by queuing package:\n");
disp(mean_clients_real); disp("");
```

```
%disp("Starting trace...");
states = [0,1,2,3,4,5];
difference=[0.01,0.001,0.0001,0.00001,0.000001,0.0000001];

for j=1:1:length(difference)
  rand("seed",1);
  total_arrivals = 0; % to measure the total number of arrivals
  blocked_arrivals = 0;
  current_state = 0;  % holds the current state of the system
  previous_mean_clients = 0; % will help in the convergence test
  index = 0;

  threshold = lambda./(lambda .+ mu); % the threshold used to calculate probabilities
  arrivals = zeros(1,6);
  clear P; clear to_plot;

  transitions = 0; % holds the transitions of the simulation in transitions steps

  while transitions >= 0
    transitions = transitions + 1; % one more transitions step

    if mod(transitions,1000) == 0 % check for convergence every 1000 transitions steps
      index = index + 1;
      for i=1:1:length(arrivals)
          P(i) = arrivals(i)/total_arrivals; % calculate the probability of every state in the system
      endfor
      Pblock = blocked_arrivals / total_arrivals;

      mean_clients = 0; % calculate the mean number of clients in the system
      for i=1:1:length(arrivals)
         mean_clients = mean_clients + (i-1).*P(i);
      endfor

      to_plot(index) = mean_clients;

      if abs(mean_clients - previous_mean_clients) < difference(j) || transitions > 20000000 %
convergence test
        break;
      endif

      previous_mean_clients = mean_clients;

    endif
```

```
    random_number = rand(1); % generate a random number (Uniform distribution)
    if current_state == 0 % arrival
      total_arrivals = total_arrivals + 1;
      arrivals(current_state + 1) = arrivals(current_state + 1) + 1; % increase the number of
arrivals in the current state
      %if transitions <= 30
        %fprintf("State: 0, arrival, arrivals in this state: %d\n", arrivals(1));
      %endif
      current_state = current_state + 1;
    else
      if random_number < threshold(current_state) % arrival
        if current_state == 5 % blocked
          total_arrivals = total_arrivals + 1;
          arrivals(current_state + 1) = arrivals(current_state + 1) + 1;
          blocked_arrivals = blocked_arrivals + 1;
          %if transitions <= 30
            %fprintf("State: 10, arrival, arrivals in this state: %d\n", arrivals(11));
          %endif
        else
          total_arrivals = total_arrivals + 1;
          arrivals(current_state + 1) = arrivals(current_state + 1) + 1;
          %if transitions <= 30
            %fprintf("State: %d, arrival, arrivals in this state: %d\n", current_state,
arrivals(current_state+1));
          %endif
          current_state = current_state + 1;
        endif
      else % departure
        %if transitions <= 30
          %fprintf("State: %d, departure, arrivals in this state: %d\n", current_state,
arrivals(current_state+1));
        %endif
        current_state = current_state - 1;
      endif
    endif
  endwhile
  fprintf("Ergodic probabilities of M/M/1/5 by simulation:\n");
  for i=1:1:length(arrivals)
    fprintf("P(%d) = %d\n", i-1, P(i));
  endfor
  fprintf("Pblock = %d\n", Pblock);
  fprintf("Mean number of clients: E[n(t)] = %d\n", mean_clients);
  mean_error(j) = abs(mean_clients_real-mean_clients);
  trans(j) = transitions;
endfor
```

```
figure(11);
bar(mean_error,'r',0.4);
title("Mean abs error");

figure(12);
bar(trans,'r',0.4);
title("Number of transitions required");
```