Συστήματα Αναμονής

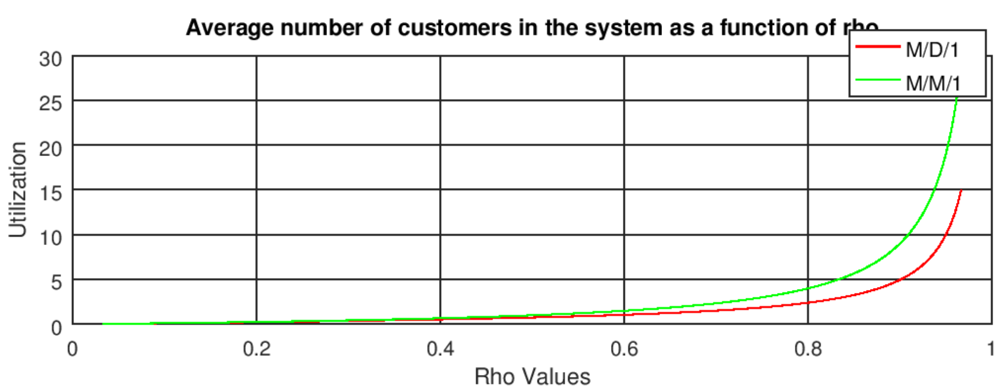
3η Εργαστηριακή Άσκηση

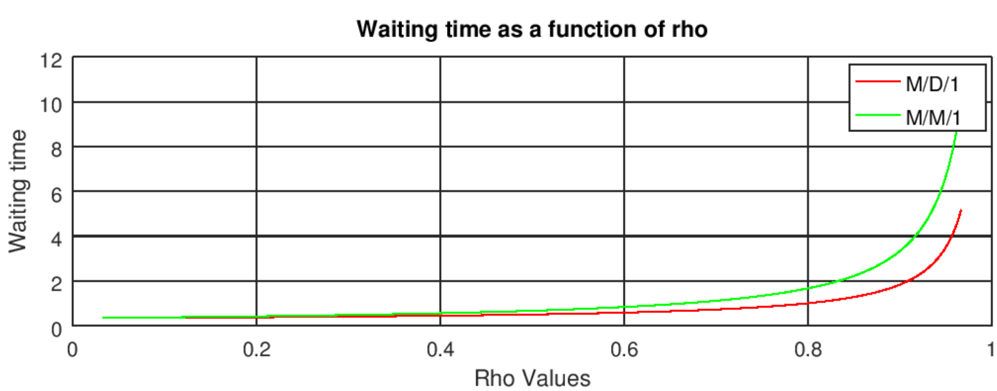
Όνομα: Σταύρος Σταύρου
ΑΜ: 03115701
Εξάμηνο: 6ο-ΣΗΜΜΥ

Σύγκριση συστημάτων M/M/1 και M/D/1

(1) Γνωρίζουμε πως $E[n(t)] = \rho + \frac{1}{2}(\frac{\rho^2}{1-\rho})$. Γνωρίζουμε ακόμη, από τον νόμο του Little πως $E[T] = \frac{E[n(t)]}{\gamma} = \frac{E[n(t)]}{\lambda} = \frac{\rho + \frac{1}{2}\left(\frac{\rho^2}{1-\rho}\right)}{\lambda}$. Ακόμη $E[T] = E[W] + E[s]$. Όμως γνωρίζουμε πως $E[s] = \frac{1}{\mu}$ σταθερό ➔ $E[W] = \frac{\rho + \frac{1}{2}\left(\frac{\rho^2}{1-\rho}\right)}{\lambda} - \frac{1}{\mu}$.
Τέλος, η συνθήκη εργοδικότητας δεν αλλάζει από τις ουρές M/M/1 και παραμένει $\rho = \frac{\lambda}{\mu} < 1$, ούτως ώστε ο εξυπηρετητής να μπορεί να ξεκουράζεται.

(2) Παίρνουμε τις εξής γραφικές παραστάσεις:





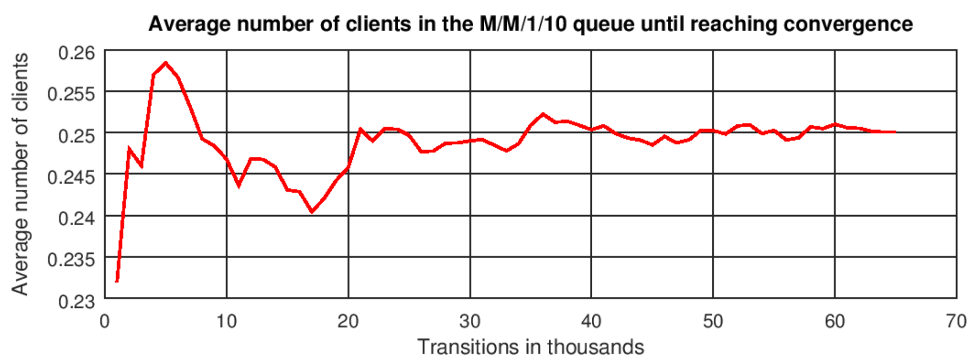
Βλέπουμε, πως με την ουρά M/D/1, έχουμε μικρότερο μέσο όρο πελάτων στο σύστημα και παράλληλα αυτοί εξυπηρετούνται πιο γρήγορα από την αντίστοιχη ουρά M/M/1. Η ουρά M/D/1, λοιπόν είναι καλύτερη μεταξύ των 2.

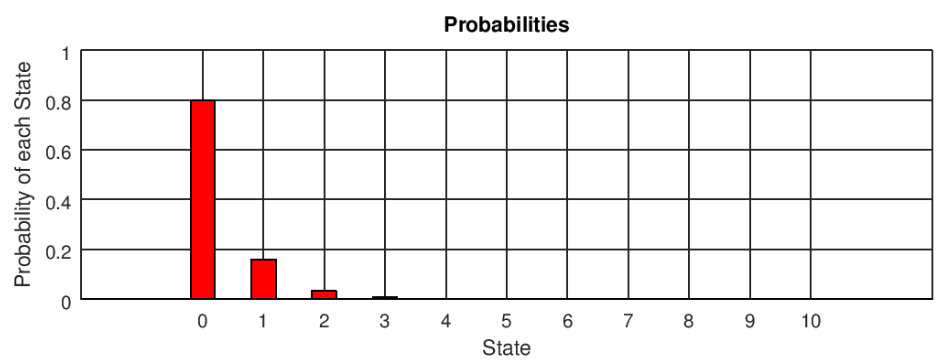
Οι κώδικες για τη σύγκριση των 2 ουρών, καθώς και ο κώδικας του ερωτήματος (2) βρίσκονται στο παράρτημα στο τέλος.

Προσομοίωση συστήματος M/M/1/10

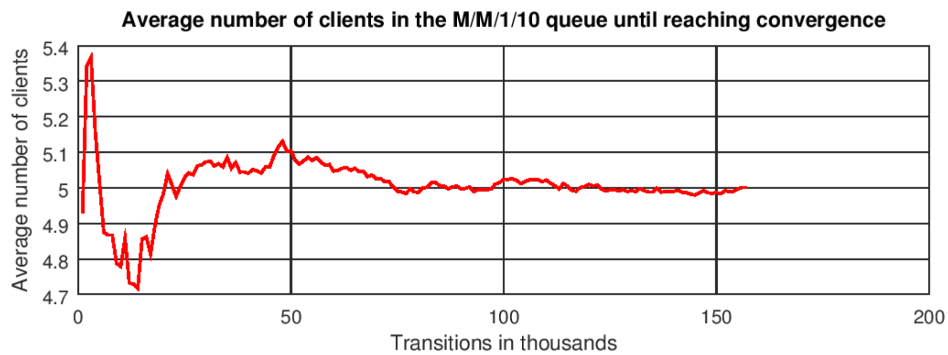
(2) Με εκτέλεση προσομοίωση στην Octave παίρνουμε τις εξής γραφικές παραστάσεις (Ο κώδικας βρίσκεται στο παράρτημα):

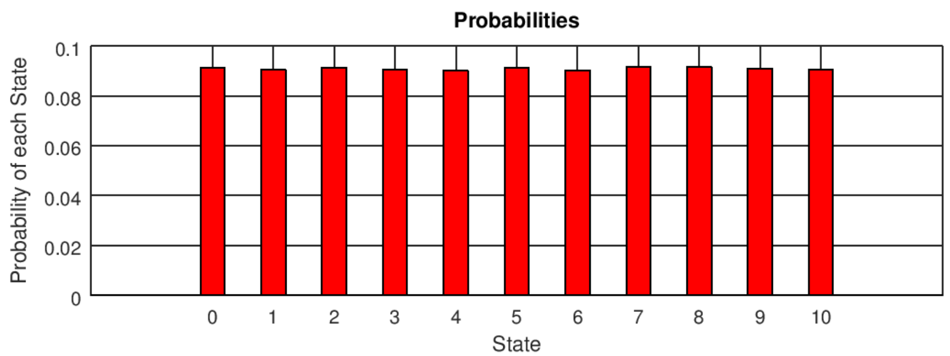
Για λ = 1, μ = 5:



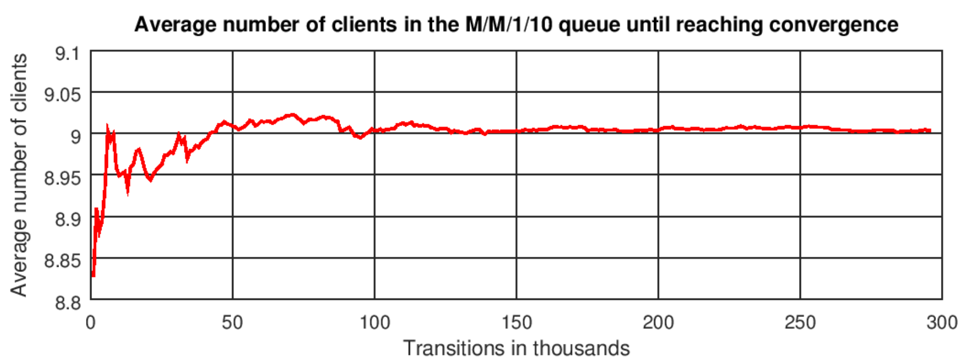


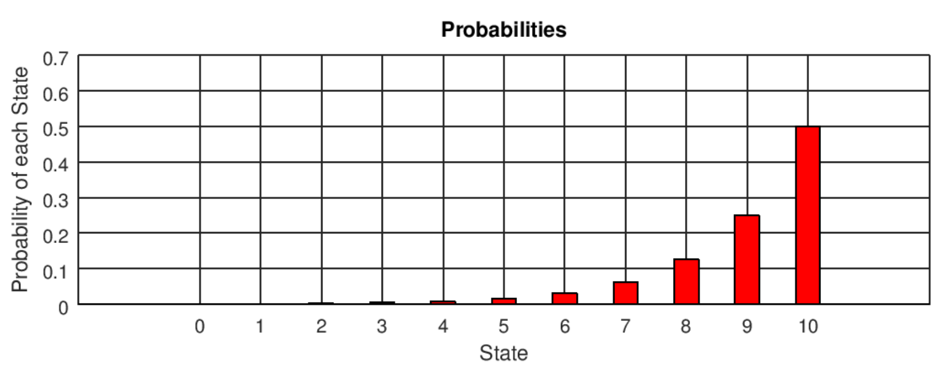
Για λ = 5, μ = 5:





Για λ = 10, μ = 5:





(3) Παρατηρούμε, πως για μεγαλύτερα λ η προσομοίωσή μας απαιτεί μεγαλύτερο αριθμό μεταβάσεων, μέχρι να έχουμε σύγκλιση.
Όπως φαίνεται από τις γραφικές παραστάσεις μας θα μπορούσαμε να αγνοήσουμε τις πρώτες χιλιάδες μεταβάσεων (ίσως μέχρι τις 3000-4000), καθώς οι αποκλίσεις σε αυτό το διάστημα είναι πολύ μεγάλες λόγω του μεταβατικού φαινομένου. Για να είμαστε πιο ασφαλείς, όμως μπορούμε απλά να αγνοήσουμε τις 1000 πρώτες μεταβάσεις.

<u>Προσομοίωση συστήματος M/M/1/5 με μεταβλητό μέσο ρυθμό εξυπηρέτησης</u>

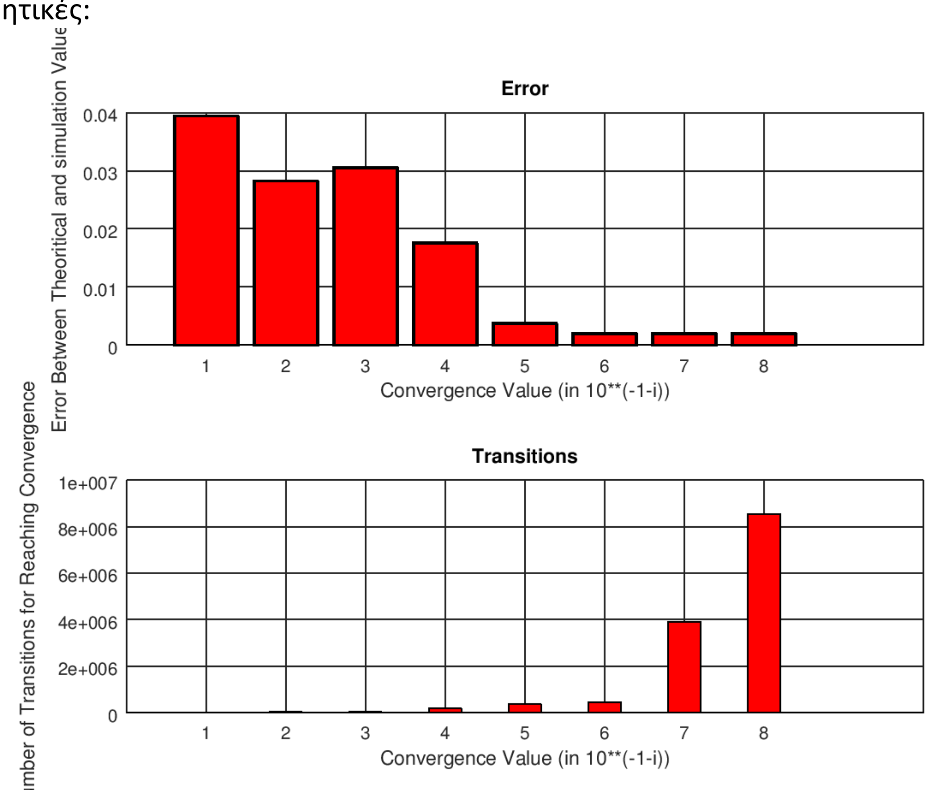
(1) Η Octave δίνει τα εξής (οι κώδικες στο παράρτημα):

```
The ergodic probabilities of the system are (for each state):
ergodic_prob =

   0.162933   0.244399   0.244399   0.183299   0.109980   0.054990

The average number of customers in the system is
 1.9980
```

(2) Παίρνουμε τα εξής διαγράμματα, αφού πρώτα ελέγξαμε τις τιμές που πήραμε με τις θεωρητικές:



Θα επέλεγα την τιμή 0.000001% για το κριτήριο σύγκλισης, καθώς από εκείνο το σημείο και έπειτα, δεν κερδίζουμε τόσο σε ακρίβεια και πληρώνουμε πολύ σε χρόνο προσομοίωσης.
Για να αντιμετωπίσουμε φαινόμενα μη-τερματισμού του προγράμματος μας όταν το κριτήριο είναι υπερβολικά αυστηρό, θα μπορούσαμε να θέσουμε ένα όριο για τις μεταβάσεις που εκτελούνται (για παράδειγμα το 10000000), ούτως ώστε αν το ξεπεράσουμε η προσομοίωση να τερματίζει επιστρέφοντας τα αποτελέσματα μέχρι εκείνη τη στιγμή.

ΠΑΡΑΡΤΗΜΑ

qsmd1.m:

```
function [U R Q X p0] = qsmd1( lambda, mu )
 if ( nargin != 2 )
   print_usage();
 endif
 ( isvector(lambda) && isvector(mu) ) || ...
     error( "lambda and mu must be vectors" );
 [ err lambda mu ] = common_size( lambda, mu );
 if ( err )
   error( "parameters are of incompatible size" );
 endif
 lambda = lambda(:)';
 mu = mu(:)';
 all( lambda >= 0 ) || ...
     error( "lambda must be >= 0" );
 all( mu > lambda ) || ...
     error( "The system is not ergodic" );
 U = rho = lambda ./ mu; # utilization
 Q = rho + 0.5 * rho.^2./(1-rho); # average number of customers
 R = Q ./ lambda;  # average waiting time
 X = lambda; # througphut
endfunction
```

qsmm1VSqsms1.m:

```
clc ;
clear all;
close all;

# orizoume ena dianisma me diafores times tou lambda apo 0.1 eos 2.9
# kai tin timi tou mu = 3
mu = 3;
lambda = 0.1:0.01:2.9;

# trexoume tin 2 sinartiseis gia tis 2 oures
[utilization_md1, waiting_time_md1, mean_state_md1] = qsmd1 (lambda, mu);
[utilization_mm1, waiting_time_mm1, mean_state_mm1] = qsmm1 (lambda, mu);

# pernoume ta diagrammata sinartisi tou rho
figure(1);
subplot (2, 1, 1);
hold on;
plot(lambda/mu, mean_state_md1, 'r');
plot(lambda/mu, mean_state_mm1, 'g');
hold off;
grid on;
title("Average number of customers in the system as a function of rho");
```

```
xlabel("Rho Values");
ylabel("Utilization");
legend("M/D/1","M/M/1");
legend("show");

subplot(2, 1, 2);
hold on;
plot(lambda/mu, waiting_time_md1, 'r');
plot(lambda/mu, waiting_time_mm1, 'g');
hold off;
grid on;
title("Waiting time as a function of rho");
xlabel("Rho Values");
ylabel("Waiting time");
legend("M/D/1","M/M/1");
legend("show");
```

qsmm110.m:

```
clc;
clear all;
close all;

total_arrivals = 0;   # initializing values
current_state = 0;
previous_mean_clients = 0;
index = 0;
sigklisi = false;
transitions = 0;
arrivals = zeros (1, 11);
lambda = 1;
mu = 5;
threshold = lambda/(lambda + mu); % the threshold used to calculate probabilities


# the next lines of code were used for debugging the code
#{
while transitions <= 30
  decision = rand (1);
  transitions = transitions + 1;
  disp(strcat("current state is ", num2str(current_state)));
  if (current_state == 0 || (decision < threshold && current_state < 10))
    disp("now comes an arrival");
    total_arrivals = total_arrivals + 1;
    arrivals(current_state + 1) = arrivals(current_state + 1) + 1;
    current_state = current_state + 1;
   else
    disp("now comes a departure");
    current_state = current_state - 1;
  endif
  disp(strcat("total arrivals in the systems are ", num2str(total_arrivals)));
end
#}

while (transitions <= 1000000 && !sigklisi)

  decision = rand (1);      # generationg a random number between 0 and 1
  transitions = transitions + 1;

  # the system gets an arrival if it is empty or if the random number is less than
  # the threshold and the system isn't full
  # else it gets a departure
  if (current_state == 0 || (decision < threshold && current_state < 11))
    total_arrivals = total_arrivals + 1;
    arrivals(current_state + 1) = arrivals(current_state + 1) + 1;
    current_state = current_state + 1;
```

```
  else
   current_state = current_state - 1;
  endif

  # every 1000 events we check for convergence
  if mod(transitions, 1000) == 0
   index = index +1;

   # calculating possibilites and average number of clients in the system
   for i = 1:1:length(arrivals)
     P(i) = arrivals(i)/total_arrivals;
   endfor

   mean_clients = 0;
   for i = 1:1:length(arrivals)
     mean_clients = mean_clients + (i-1) * P(i);
   endfor
   to_plot(index) = mean_clients;

   # convergence check here
   if abs(mean_clients - previous_mean_clients) < 0.00001
     sigklisi = true;
   endif
   previous_mean_clients = mean_clients;
  endif

endwhile

states = zeros(1, length(arrivals));
for i=1:1:length(arrivals)
  states(i) = i - 1;
endfor

figure(1);
subplot(2, 1, 1);
plot(to_plot,"r","linewidth",1.3);
grid on;
title("Average number of clients in the M/M/1/10 queue until reaching convergence");
xlabel("Transitions in thousands");
ylabel("Average number of clients");

subplot(2, 1, 2);
bar(states, P, 'r',0.4);
grid on;
title("Probabilities");
xlabel("State");
ylabel("Probability of each State");
```

qsmm15_ready_functions.m:

```
clc;
clear all;
close all;

# orizete to sistima kai i arxiki katastasi tou
lambda = 3;
mu = 1;
states = [0, 1, 2, 3, 4, 5];
initial_state = [1, 0, 0, 0, 0, 0];

genniseis = [lambda, lambda, lambda, lambda, lambda];
thanatoi = [2, 3, 4, 5, 6];

# ipologismos kai tipoma tou metavatikou pinaka
metavatikos = ctmcbd(genniseis, thanatoi);
# ipologismos kai tipoma ergodikon pithanotiton
ergodic_prob = ctmc(metavatikos);

disp("The ergodic probabilities of the system are (for each state):");
display(ergodic_prob);

# ipologismos kai tipoma mesou arithmou pelaton sto sistima
avg_customers = ergodic_prob(2) + 2 * ergodic_prob(3) + 3 * ergodic_prob(4) + 4 * ergodic_prob(5) + 5 * ergodic_prob(6);
disp("The average number of customers in the system is "), disp (avg_customers);
```

qmm15.m:

```
clc;
clear all;
close all;

lambda = [3, 3, 3, 3, 3];
mu = [2, 3, 4, 5, 6];
threshold = [1, 3/5, 3/6, 3/7, 3/8, 3/9, 0];
metavaseis = zeros (1, 8);
sfalma = zeros (1,8);

for j = 1:1:8
  clear arrivals;
  clear P;
  total_arrivals = 0;   # initializing values for every loop
  current_state = 0;
  previous_mean_clients = 0;
  sigklisi = false;
  transitions = 0;
  while !sigklisi
```

```
      decision = rand (1);      # generationg a random number between 0 and 1
      transitions = transitions + 1;

      # the system gets an arrival if it is empty or if the random number is less than
      # the threshold and the system isn't full
      # else it gets a departure
      if decision < threshold(current_state + 1)
        total_arrivals = total_arrivals + 1;
        try
          arrivals(current_state + 1) = arrivals(current_state + 1) + 1;
          current_state = current_state + 1;
        catch
          arrivals(current_state + 1) = 1;
          current_state = current_state + 1;
        end_try_catch
       else
        current_state = current_state - 1;
      endif

      # every 1000 events we check for convergence
      if mod(transitions, 1000) == 0
        # calculating possibilites and average number of clients in the system
        for i = 1:1:length(arrivals)
          P(i) = arrivals(i)/total_arrivals;
        endfor

        mean_clients = 0;
        for i = 1:1:length(arrivals)
          mean_clients = mean_clients + (i-1) * P(i);
        endfor

        # convergence check here
        if abs(mean_clients - previous_mean_clients) < (0.01/(10**(j-1)))
          sigklisi = true;
          metavaseis(j) = transitions;
          sfalma(j) = abs(mean_clients - 1.9980);
        endif
        previous_mean_clients = mean_clients;
      endif

    endwhile
  endfor


states = 1:1:8;

# plotting the results
figure(1);
```

```
subplot(2, 1, 1);
bar(states, sfalma, "r","linewidth",1.3);
grid on;
title("Error");
xlabel("Convergence Value (in 10**(-1-i))");
ylabel("Error Between Theoritical and simulation Value");

subplot(2, 1, 2);
bar(metavaseis, 'r',0.4);
grid on;
title("Transitions");
xlabel("Convergence Value (in 10**(-1-i))");
ylabel("Number of Transitions for Reaching Convergence");
```