Συστήματα Αναμονής

# 3η Εργαστηριακή Άσκηση

**Όνομα:** Σταύρος Σταύρου

**ΑΜ:** 03115701

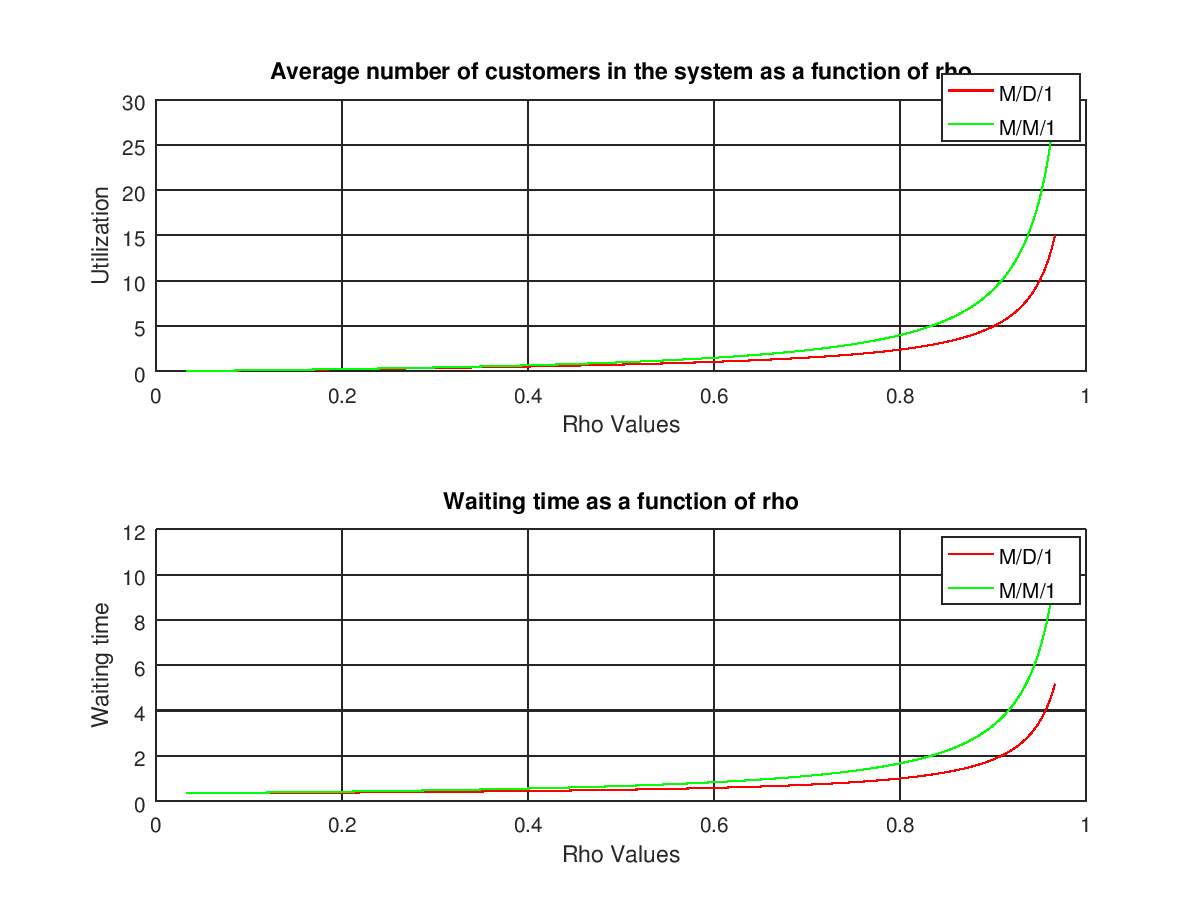
**Εξάμηνο:** 6ο-ΣΗΜΜΥ

**Σύγκριση συστημάτων Μ/Μ/1 και M/D/1**

1. Γνωρίζουμε πως . Γνωρίζουμε ακόμη, από τον νόμο του Little πως Ακόμη Όμως γνωρίζουμε πως σταθερό 🡺 .

Τέλος, η συνθήκη εργοδικότητας δεν αλλάζει από τις ουρές Μ/Μ/1 και παραμένει , ούτως ώστε ο εξυπηρετητής να μπορεί να ξεκουράζεται.

1. Παίρνουμε τις εξής γραφικές παραστάσεις:



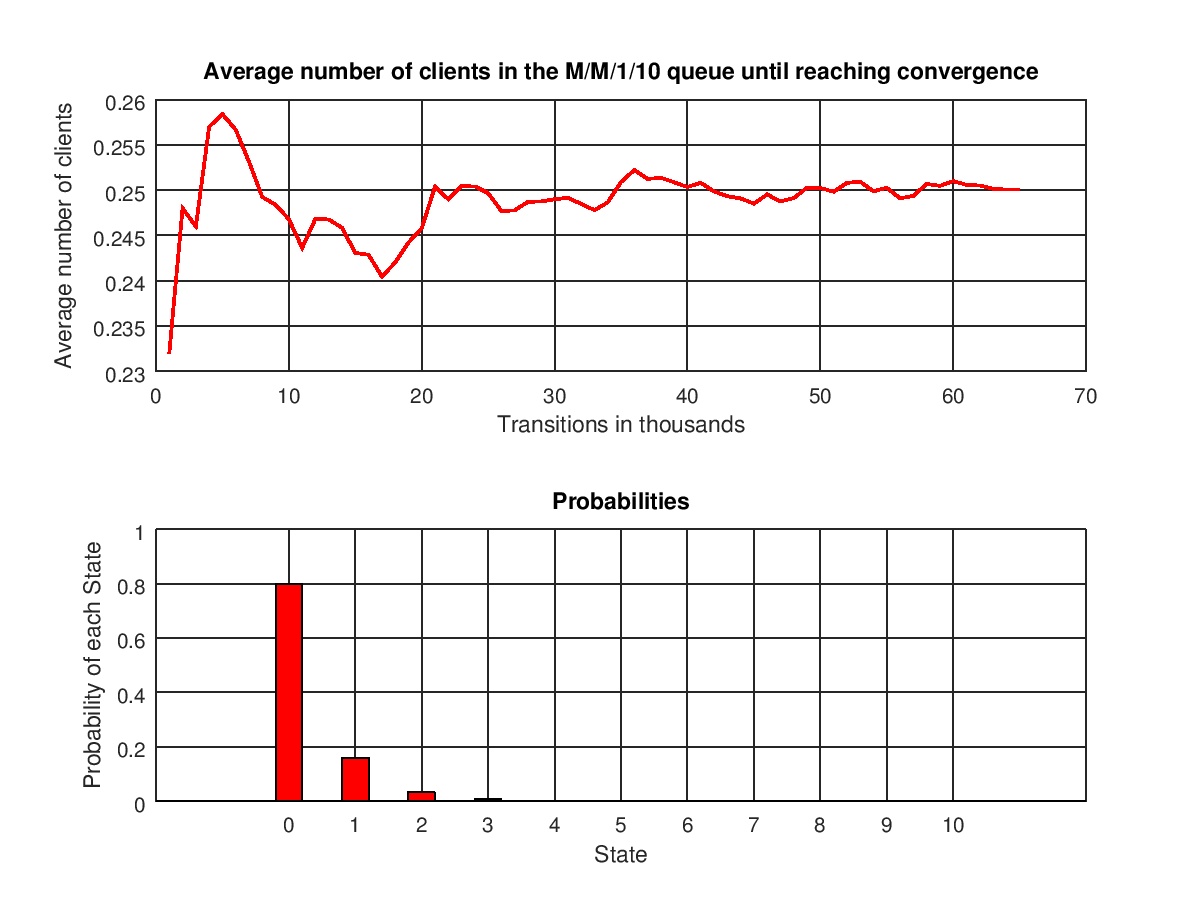
Βλέπουμε, πως με την ουρά M/D/1, έχουμε μικρότερο μέσο όρο πελάτων στο σύστημα και παράλληλα αυτοί εξυπηρετούνται πιο γρήγορα από την αντίστοιχη ουρά M/M/1. Η ουρά M/D/1, λοιπόν είναι καλύτερη μεταξύ των 2.

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

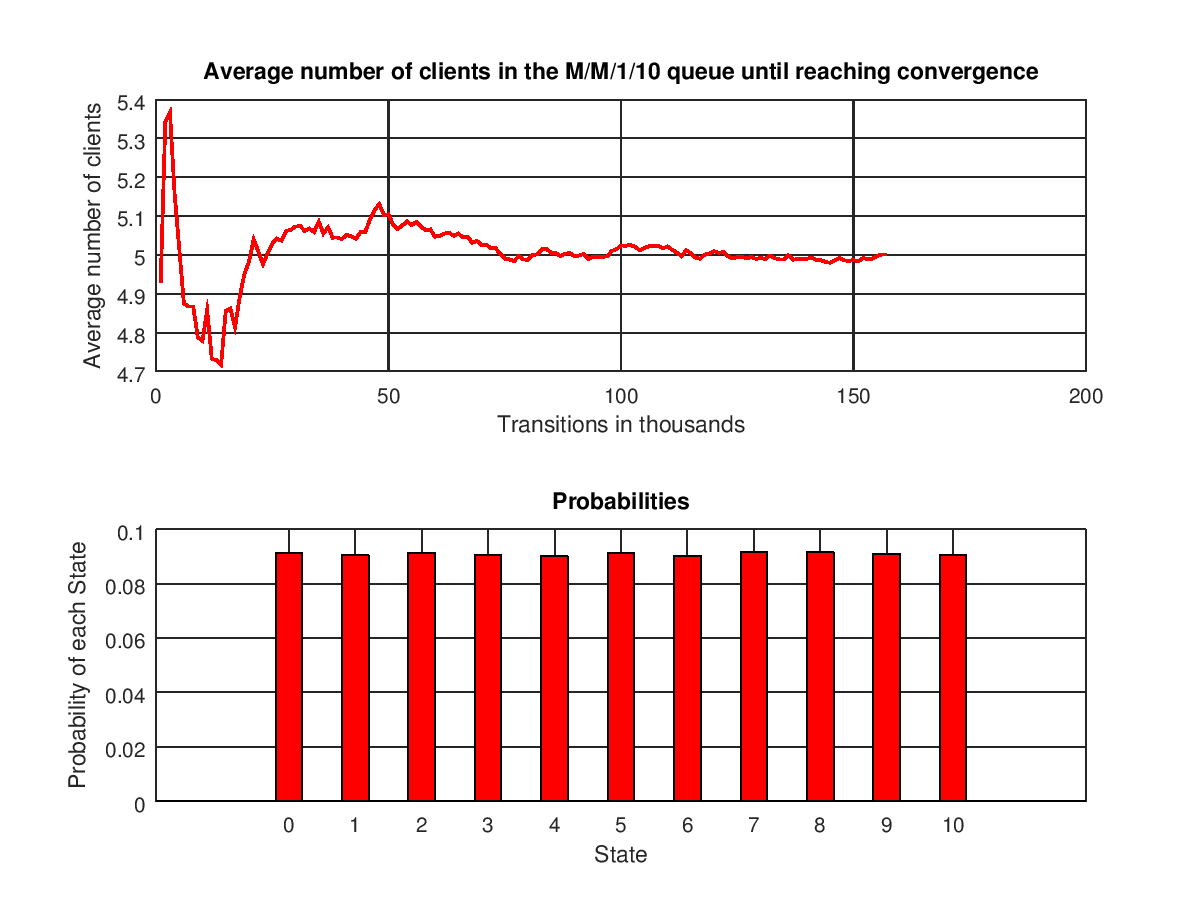
**Προσομοίωση συστήματος Μ/Μ/1/10**

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

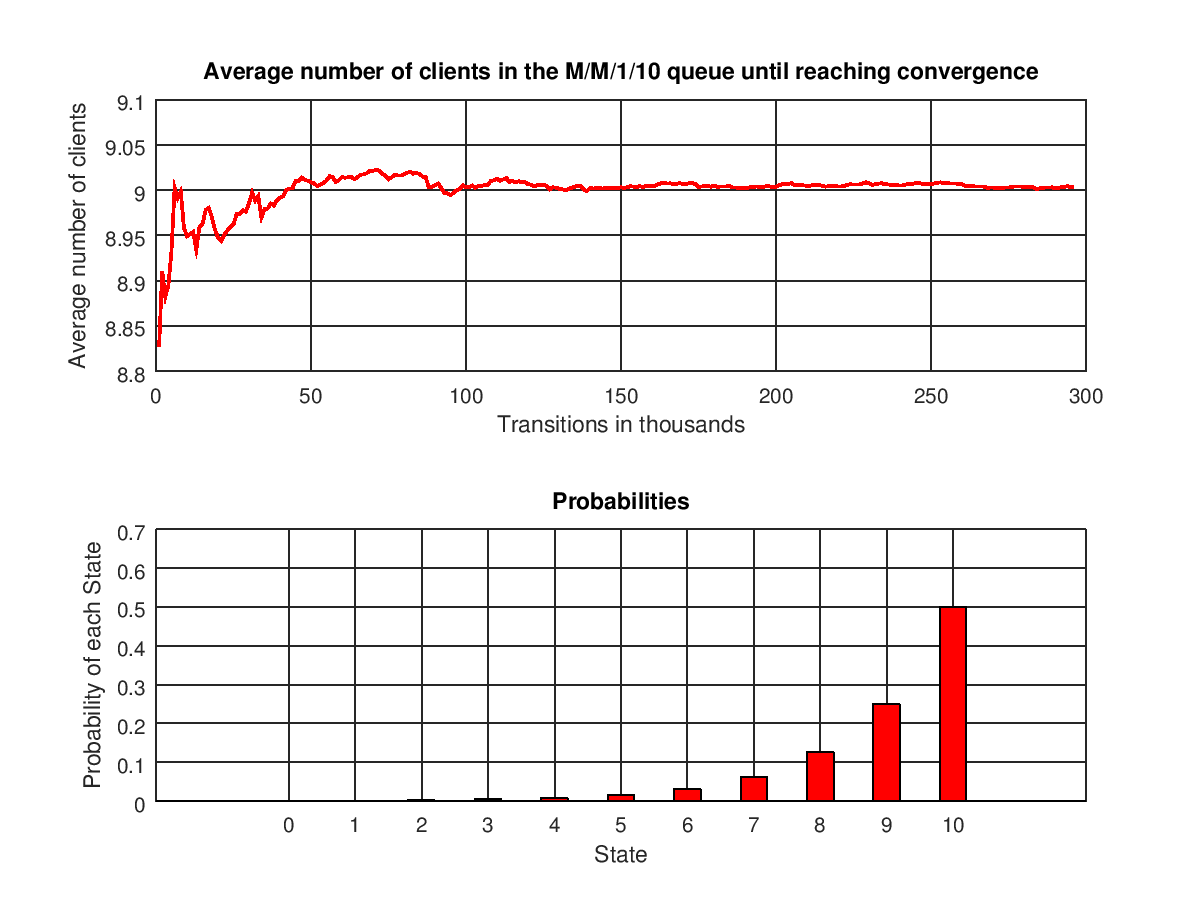
Για λ = 1, μ = 5:



Για λ = 5, μ = 5:



Για λ = 10, μ = 5:

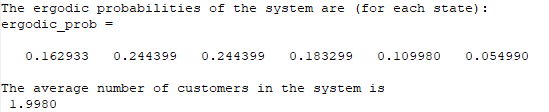


1. Παρατηρούμε, πως για μεγαλύτερα λ η προσομοίωσή μας απαιτεί μεγαλύτερο αριθμό μεταβάσεων, μέχρι να έχουμε σύγκλιση.

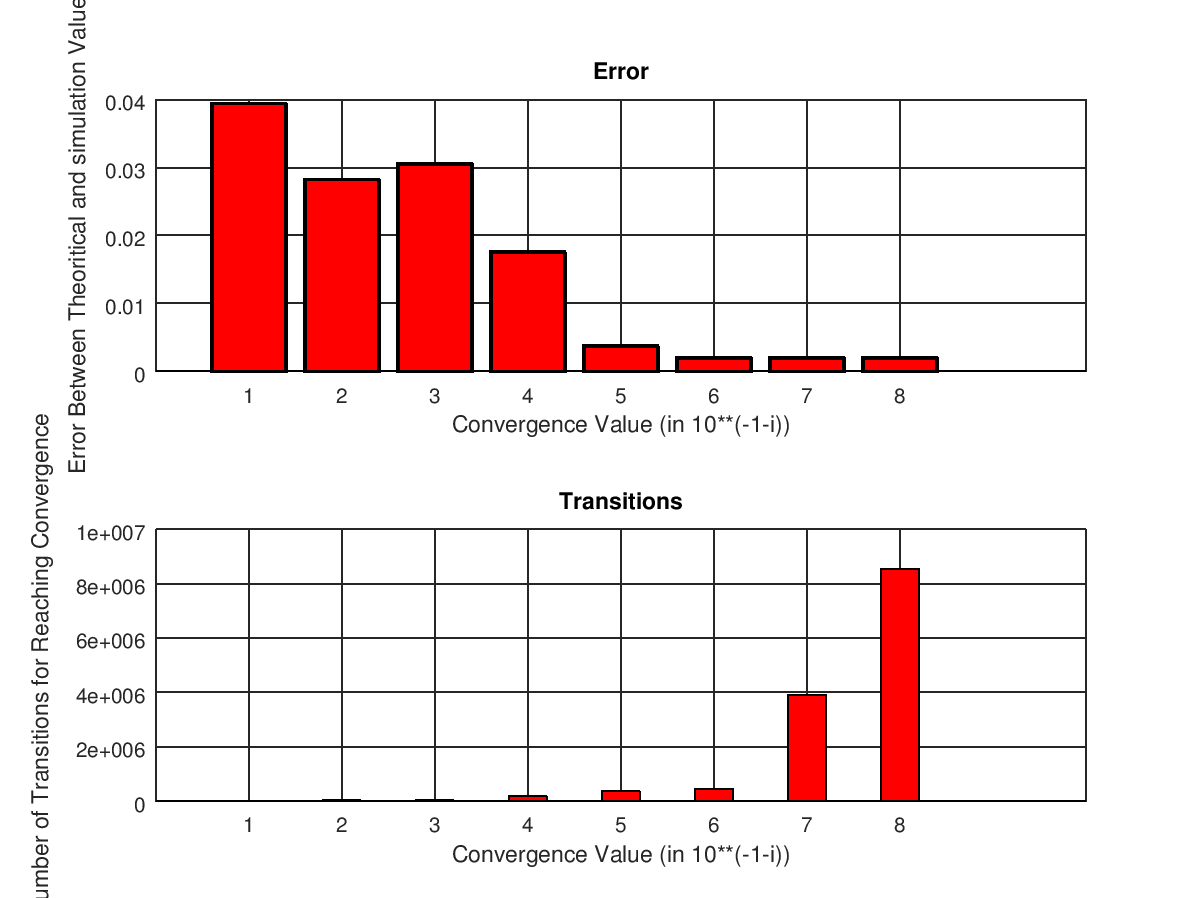
Όπως φαίνεται από τις γραφικές παραστάσεις μας θα μπορούσαμε να αγνοήσουμε τις πρώτες χιλιάδες μεταβάσεων (ίσως μέχρι τις 3000-4000), καθώς οι αποκλίσεις σε αυτό το διάστημα είναι πολύ μεγάλες λόγω του μεταβατικού φαινομένου. Για να είμαστε πιο ασφαλείς, όμως μπορούμε απλά να αγνοήσουμε τις 1000 πρώτες μεταβάσεις.

**Προσομοίωση συστήματος Μ/Μ/1/5 με μεταβλητό μέσο ρυθμό εξυπηρέτησης**

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



1. 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");