ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
Τομέας Επικοινωνιών, Ηλεκτρονικής & Συστημάτων Πληροφορικής
Εργαστήριο Διαχείρησης και Βέλτριστου Σχεδιασμού Δικτύων Τηλεματικής – NETMODE

Πεγειώτη Νάταλυ
Α.Μ: 03117707

Συστήματα Αναμονής (Queuing Systems)

<u>3η Ομάδα Ασκήσεων</u>

<u>Προσομοίωση συστήματος M/M/1/10</u>

Ο κώδικας που χρησιμοποιήθηκε παρατίθεται στο αρχείο qsmm110.m (βλ. Παράρτημα Κώδικα)

(1) Το debugging για τις πρώτες 30 μεταβάσεις πραγματοποιήθηκε επιτυχώς.
Ο κώδικας που χρησιμοποιήθηκε φαίνεται στο #PART1 του αρχείου qsmm110.m.

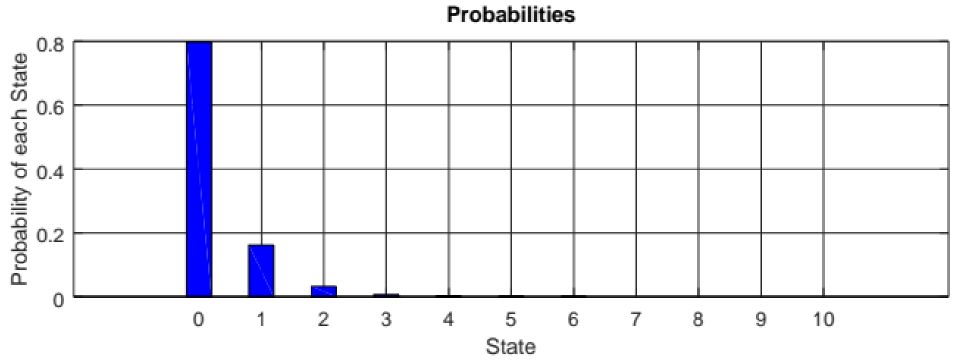
```
Current state:0
New arrival
Total arrivals in the system:1


Current state:1
New arrival
Total arrivals in the system:2


Current state:2
New arrival
Total arrivals in the system:3


Current state:3
Departure
Total arrivals in the system:3


Current state:2
Departure
Total arrivals in the system:3


Current state:1
Departure
Total arrivals in the system:3


Current state:0
New arrival
Total arrivals in the system:4
```

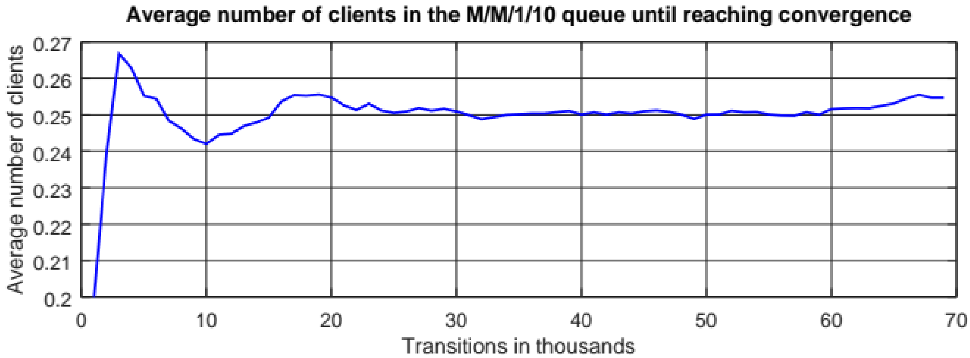
Στο διπλανό στιγμιότυπο παρουσιάζεται κομμάτι του output του debbugging.

Πρόκειται για τις 7 πρώτες μεταβάσεις. Με παρόμοιο τρόπο παρουσιάζονται και οι υπόλοιπες 23 μεταβάσεις του συστήματος.

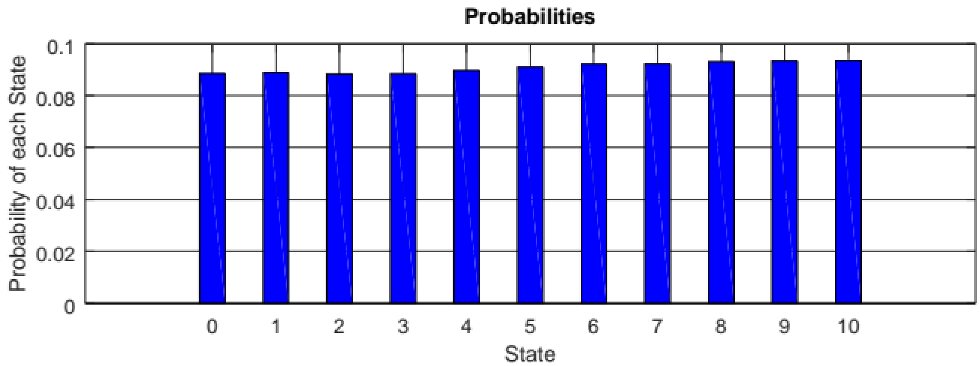
(2) Η προσωμοίωση εκτελέστηκε για τις τιμές του λ = {1, 5, 10} και οι γραφικές παραστάσεις που προέκυψαν για κάθε μία από τις τιμές αυτές παρουσιάζονται πιο κάτω.
Ο κώδικας που χρησιμοποιήθηκε φαίνεται στο #PART2 του αρχείου qsmm110.m και για κάθε περίπτωση η υλοποίηση του προγράμματος επαναλαμβανόταν με τροποποιημένο λ.

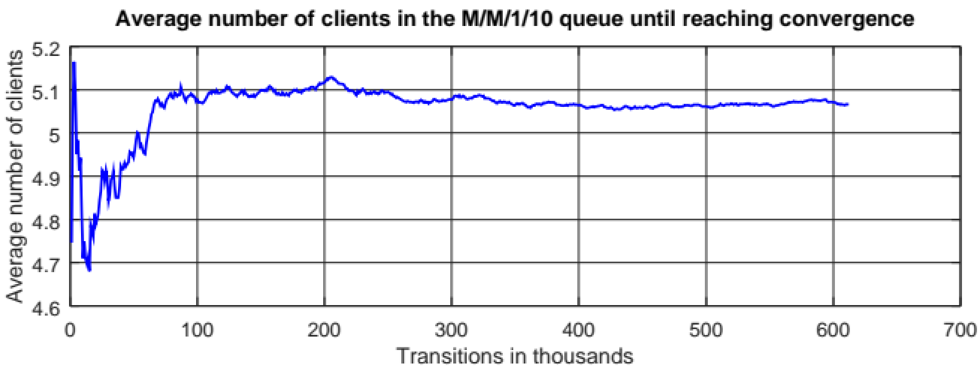
Για λ = 1:



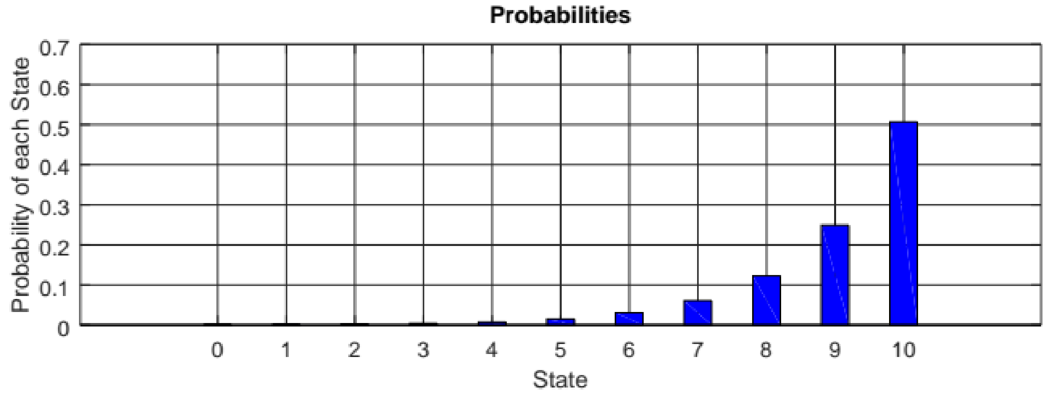


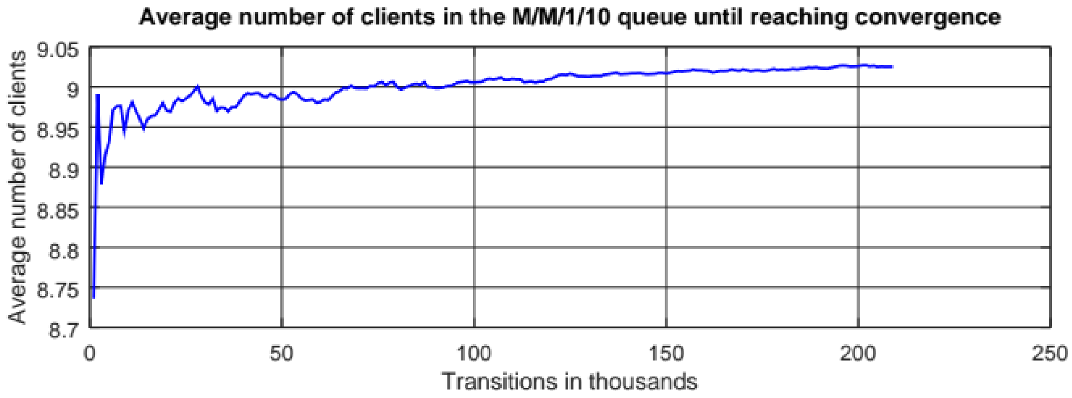
Για λ = 5:





Για λ = 10:





(3) Παρατηρούμε ότι στην 1η περίπτωση (λ = 1) το σύστημα λόγω του μικρού φορτίου μένει περισσότερο στις πρώτες καταστάσεις και φτάνει γρήγορα σε σύγκλιση.
Κάτι παρόμοιο συμβαίνει και στην 3η περίπτωση (λ = 10) όπου όμως λόγω του μεγάλου φορτίου το σύστημα μένει περισσότερο στις τελευταίες καταστάσεις.
Όσον αφορά τη 2η περίπτωση (λ = 5) παρατηρείται ότι είναι σχεδόν ισοπίθανο για οποιαδήποτε κατάσταση το σύστημα να βρίσκεται εκεί, πράγμα που σημαίνει ότι είναι πολύ πιθανό να αλλάζει καταστάσεις πιο συχνά, με αποτέλεσμα την καθυστέρηση της σύγκλισης.
Σύμφωνα με την θεωρία, θα μπορούσαμε να αγνοήσουμε με ασφάλεια τις πρώτες 1000 μεταβάσεις.

(4) Σε περίπτωση που ο ρυθμός εξυπηρέτησης μ είναι μεταβλητός, τότε αλλάζει και η τιμή κατωφλίου που καθορίζει αν η επόμενη κατάσταση θα είναι άφιξη ή αναχώρηση. Θεωρώντας μέσο ρυθμό εξυπηρέτησης $\mu_i = \mu*(i+1)$, όπου μ = 1 πελάτης/sec, και i = {1, 2, ..., 10} τότε το κατώφλι είναι συνάρτηση και της παρούσας κατάστασης του συστήματος.
Εφόσον ο ρυθμός εξυπηρέτησης καθορίζεται με μονάδα μέτρησης [πελάτης/sec], τότε πρέπει αντίστοιχα να καθοριστεί και η μονάδα μέτρησης του ρυθμού άφιξης λ, η οποία προηγουμένως είχε δοθεί σε [πελατης/min].

<u>Παράρτημα Κώδικα</u>

qsmm110.m

```
clc;
clear all;
close all;
rand("seed",1)
total_arrivals = 0; # initializing values
current_state = 0;
previous_mean_clients = 0;
index = 0;
sigklisi = false;
transitions = 0;
arrivals = zeros (1, 11);
lambda = 1; #1 or 5 or 10
m = 5;
threshold = lambda/(lambda + m);


#PART1
#the next lines of code were used for debugging the code
#{
while transitions <= 30
	decision = rand (1);
	transitions = transitions + 1;
	disp(strcat("Current state: ", num2str(current_state)));
	if (current_state == 0 || (decision < threshold && current_state < 10))
		disp("New arrival");
		total_arrivals = total_arrivals + 1;
		arrivals(current_state + 1) = arrivals(current_state + 1) + 1;
		current_state = current_state + 1;
	else
		disp("Departure");
		current_state = current_state – 1;
	endif
	disp(strcat("Total arrivals in the system: ", num2str(total_arrivals)));
	disp("\n");
end
#}
```

```
#PART2
while (transitions <= 1000000 && !sigklisi)
	decision = rand (1); #generating 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);
bar(states, P, “b”,0.4);
grid on;
xlabel("State");
ylabel("Probability of each State");
title("Probabilities");

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