Συστήματα Αναμονής

# 4η Εργαστηριακή Άσκηση

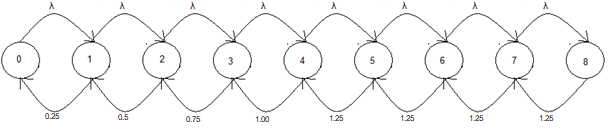
**Όνομα:** Σταύρος Σταύρου

**ΑΜ:** 03115701

**Εξάμηνο:** 6ο-ΣΗΜΜΥ

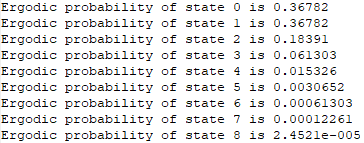
**Σύστημα Μ/Μ/Ν/Κ (call center)**

1. Έχουμε το εξής διάγραμμα μεταβάσεων:

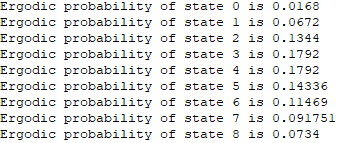


1. Με τον κώδικα qsmm58 .m (στο παράρτημα παίρνουμε τα εξής αποτελέσματα:

Για :



Για :



1. Έχουμε

Για έχουμε . Η συνάρτηση erlangc δίνει .

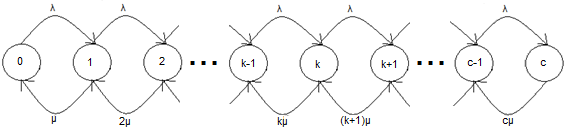
Για έχουμε . Η συνάρτηση erlangc δίνει

Παρατηρούμε, πως παρότι η συνάρτηση erlangc αφορά συστήματα άπειρης χωρητικότητας, εν τούτοις δίνει εξαιρετικά ακριβή αποτελέσματα (με πολύ μικρή απόκλιση) και για συστήματα πεπερασμένης χωρητικότητας εφόσον το φορτίο που λαμβάνει είναι αρκετά μικρό.

Από την άλλη για μεγαλύτερα φορτία ρ αυτό δεν ισχύει, καθώς η απόρριψη κάποιων πακέτων σε ένα σύστημα πεπερασμένης χωρητικότητας, μειώνει την αναμονή κάποιου μελλοντικού πελάτη.

**Ανάλυση και Σχεδιασμός τηλεφωνικού κέντρου**

1. Έχουμε το εξής διάγραμμα μεταβάσεων:



Παίρνοντας τις εξισώσεις ισορροπίας του τυχαίου κόμβου k έχουμε 🡺 .

Επίσης από συνθήκη κανονικοποίησης 🡺 .

Και από τις 2 πιο πάνω σχέσεις .

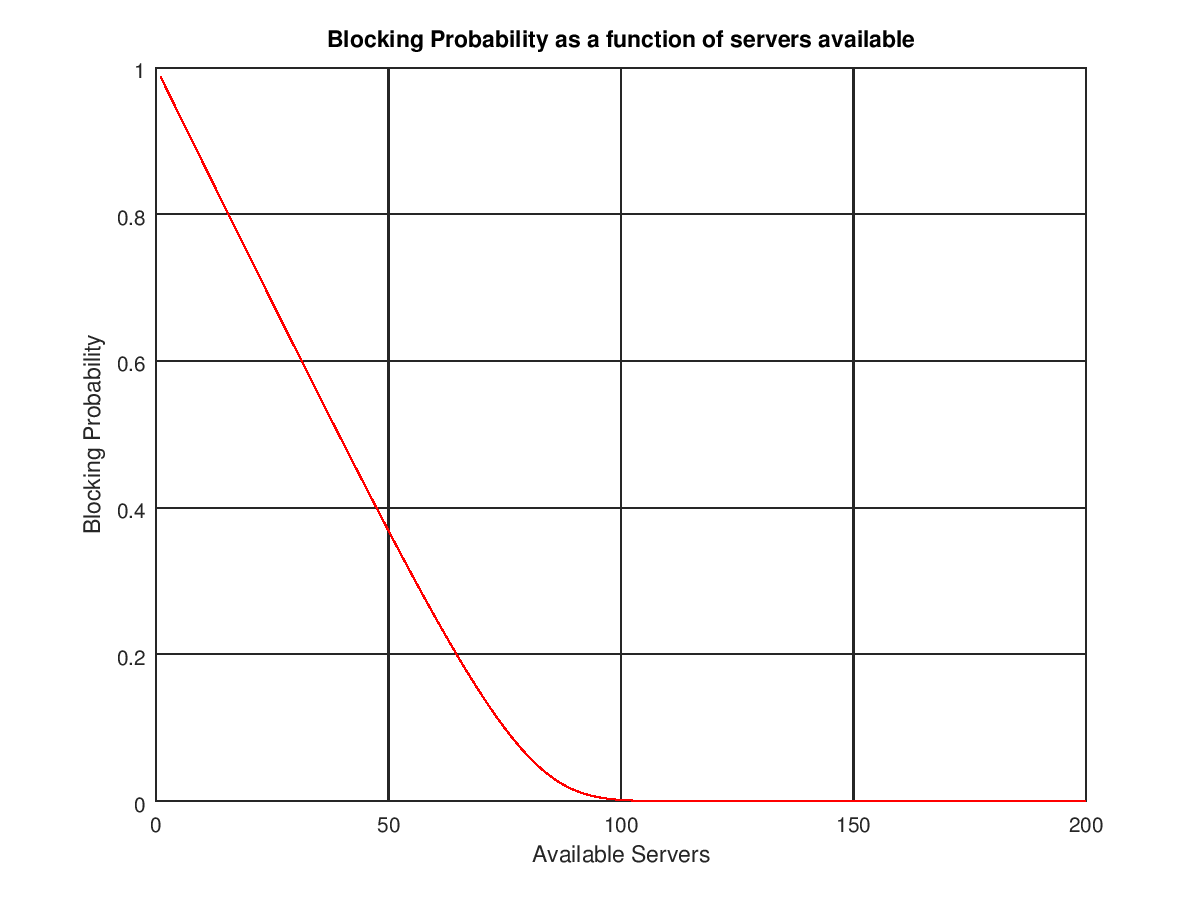
Ο μέσος ρυθμός απωλειών δίδεται από τη σχέση .

1. Έχουμε . Παίρνουμε για c+1 την εξής σχέση:

και αντικαθιστώντας το c+1 με n παίρνουμε την τελική σχέση . Από την προηγούμενη σχέση παίρνουμε και την «βάση» της αναδρομής .

1. Παρατηρούμε πως πράγματι η erlangb\_iterative μας δίνει τη σωστή τιμή 0.024524. Όμως η erlanbg\_factorial δίνει τιμή NaN. Αυτό συμβαίνει, επειδή στην erlangb\_factorial έχουμε τον υπολογισμό πολύ μεγάλων τιμών (πχ ) με αποτέλεσμα να έχουμε υπερχείλιση στην Octave και αδυναμία υπολογισμού κάποιου αποτελέσματος.
2. Μοντελοποιώ το σύστημα ως εξής: με διάρκεια 🡺 . Με διαφορετική μοντελοποίηση καταλήγω στο ίδιο αποτέλεσμα. Για παράδειγμα με διάρκεια 🡺 . Αυτό όμως είναι η «συνεισφορά» μόνο ενός χρήστη. Συνεπώς το προσφερόμενο φορτίο είναι

Με χρήση της Octave παίρνουμε το εξής διάγραμμα:



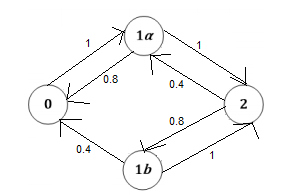
Και την εξής έξοδο για εύρεση της τιμής που δίνει πιθανότητα απόρριψης κλήσης <1%:



Όλοι οι κώδικες που χρησιμοποιήθηκαν, καθώς και οι συναρτήσεις που ζητούνται, βρίσκονται στο παράρτημα στο τέλος.

**Σύστημα εξυπηρέτησης με δύο ανόμοιους εξυπηρετητές**

1. Έχουμε το εξής διάγραμμα καταστάσεων:

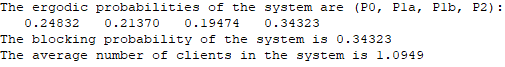


Έχουμε , , και τη συνθήκη κανονικοποίησης . Λύνοντας παίρνουμε και

Έχουμε, επίσης

Τέλος, ο μέσος αριθμός πελατών δίνεται από τον πιθανοτικό τύπο

1. Η προσομοίωση στην Octave δίνει:



Παρατηρούμε τα αποτελέσματα της προσομοίωσης να είναι αρκετά κοντά στις τιμές που υπολογίσαμε στο προηγούμενο ερώτημα.

Ο κώδικας της προσομοίωσης ακολουθεί στο παράρτημα.

**ΠΑΡΑΡΤΗΜΑ**

**qsmm58.m:**

clc;

clear all;

close all;

lambda = 1;

mu = 1/4;

states = 0:1:8;

initial\_sate = [1, 0, 0, 0, 0, 0, 0, 0 , 0]; # initial state of the system

genniseis = [lambda, lambda, lambda, lambda, lambda, lambda, lambda, lambda];

thanatoi = [mu, 2\*mu, 3\*mu, 4\*mu, 5\*mu, 5\*mu, 5\*mu, 5\*mu];

metavatikos = ctmcbd(genniseis, thanatoi);

ergodic\_prob = ctmc(metavatikos);

format short g;

for i = 1:1:9

message = cstrcat ("Ergodic probability of state ", num2str(i-1), " is ", num2str(ergodic\_prob(i)));

disp(message);

endfor

P\_waiting1 = ergodic\_prob(6) + ergodic\_prob(7) + ergodic\_prob(8) + ergodic\_prob(9);

disp(cstrcat("The possibility of every server being busy is ", num2str(P\_waiting1)));

P\_waiting = erlangc(lambda/mu, 5);

disp(cstrcat("The possibility of every server being busy (for an infinite system) is ", num2str(P\_waiting)));

**erlangb\_factorial.m:**

function pblock = erlangb\_factorial(rho, c)

if ( nargin != 2 )

print\_usage();

endif

sum = 0;

for k = 1:1:c+1

sum = sum + (rho\*\*(k-1))/(factorial(k-1));

endfor

pblock = (rho\*\*c)/((factorial(c))\*sum);

endfunction

**erlangb\_iterative.m:**

function pblock = erlangb\_iterative(rho, c)

if ( nargin != 2 )

print\_usage();

endif

results = zeros(1, c + 1);

results(1) = 1;

for i = 2:1:c+1

results(i) = rho\*results(i-1) / (rho\*results(i-1) + i-1);

endfor

pblock = results(c + 1);

endfunction

**Call\_center.m:**

clc;

clear all;

close all;

blocking\_prob = zeros(1, 200);

rho = 76.66667;

c = 1:1:200;

#calculating the blocking probabilities

for i = 1:1:200;

blocking\_prob(i) = erlangb\_iterative(rho, i);

endfor

#finding the first probability less than 1%

for i = 1:1:200

if blocking\_prob(i) < 0.01

break;

endif

endfor

disp(cstrcat("We want ", num2str(i), " servers for a probability less than 1%. The probability is ", num2str(blocking\_prob(i))));

disp(cstrcat("The probability for ", num2str(i-1), " servers is ", num2str(blocking\_prob(i-1))));

figure(1);

plot(c, blocking\_prob, 'r');

grid on;

title("Blocking Probability as a function of servers available");

xlabel("Available Servers");

ylabel("Blocking Probability");

**qsmm2:**

clc;

clear all;

close all;

state1b = 0;

state1a = 0;

total\_arrivals = 0;

current\_state = 0;

previous\_mean\_clients = 0;

sigklisi = false;

transitions = 0;

arrivals = zeros(1, 4);

P = zeros(1, 4);

while transitions <= 300000 && !sigklisi

decision = rand(1);

transitions = transitions + 1;

if current\_state == 0

total\_arrivals = total\_arrivals + 1;

current\_state = 1;

arrivals(1) = arrivals(1) + 1;

state1a = 1;

elseif state1a && (decision < 1 / 2.2)

total\_arrivals = total\_arrivals + 1;

current\_state = 2;

arrivals(2) = arrivals(2) + 1;

state1a = 0;

elseif state1a && (decision > 1 / 2.2) && (decision < 1.8 / 2.2)

current\_state = 0;

state1a = 0;

elseif state1b && (decision < 1 / 2.2)

total\_arrivals = total\_arrivals + 1;

current\_state = 2;

arrivals(3) = arrivals(3) + 1;

state1b = 0;

elseif state1b && (decision > 1.8 / 2.2)

current\_state = 0;

state1b = 0;

elseif (current\_state == 2) && (decision < 1 / 2.2)

total\_arrivals = total\_arrivals + 1;

arrivals(4) = arrivals(4) + 1;

elseif (current\_state == 2) && (decision > 1.8 / 2.2)

current\_state = 1;

state1a = 1;

elseif (current\_state == 2) && (decision > 1 / 2.2) && (decision < 1.8 / 2.2)

current\_state = 1;

state1b = 1;

endif

if mod(transitions, 1000) == 0

for i = 1:1:4

P(i) = arrivals(i)/total\_arrivals;

endfor

mean\_clients = P(2) + P(3) + 2 \* P(4);

if abs(mean\_clients - previous\_mean\_clients) < 0.00001

sigklisi = true;

endif

previous\_mean\_clients = mean\_clients;

endif

endwhile

disp("The ergodic probabilities of the system are (P0, P1a, P1b, P2):");

disp(P);

disp(cstrcat("The blocking probability of the system is ", num2str(P(4))));

disp(cstrcat("The average number of clients in the system is ", num2str(mean\_clients)));