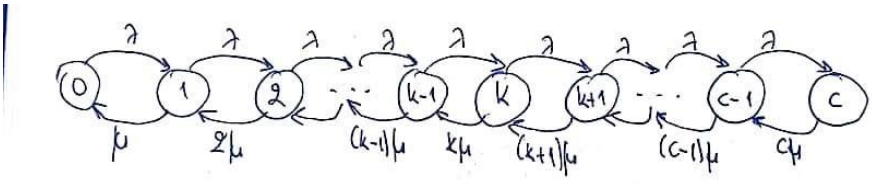
# Συστήματα Αναμονής

## 4η Ομάδα Ασκήσεων

*Άννα Κουτσώνη 03120019*

## Ανάλυση και Σχεδιασμός Τηλεφωνικού Κέντρου

**1.** Διάγραμμα ρυθμού μεταβάσεων του συστήματος:






Οι εξισώσεις ισορροπίας είναι οι εξής:

$$P_k = \left(\frac{\lambda}{k\mu}\right) * P_{k-1} \quad , k = 1,2, \dots c \qquad (\mathbf{1})$$

$$P_0 + P_1 + \cdots + P_{c-1} + P_c = 1 \qquad (\mathbf{2})$$

Από την (1) προκύπτει αναδρομικά:

$$P_k = \left(\frac{\lambda}{k\mu}\right) * P_{k-1} = \left(\frac{\lambda}{k\mu}\right) * \left(\frac{\lambda}{(k-1)\mu}\right) * P_{k-2} = \left(\frac{\lambda^2}{k(k-1)\mu^2}\right) * P_{k-2} = \cdots$$

$$= \left(\frac{\lambda^k}{k!\,\mu^k}\right) * P_0 \qquad (\mathbf{3})$$

$$(2) \overset{(3)}{\Rightarrow} P_0 + \frac{\lambda}{\mu} * P_0 + \cdots + \frac{\lambda^{c-1}}{(c-1)! * \mu^{c-1}} * P_0 + \frac{\lambda^c}{c! * \mu^c} * P_0 = 1 \quad \Rightarrow$$

$$P_0 * \left[\frac{\lambda^0}{0!\,\mu^0} + \frac{\lambda^1}{1!\,\mu^1} + \cdots + \frac{\lambda^{c-1}}{(c-1)!\,\mu^{c-1}} + \frac{\lambda^c}{c!\,\mu^c}\right] = 1 \quad \Rightarrow$$

$$P_0 * \sum_{k=0}^{c} \left(\frac{\lambda^k}{k!\,\mu^k}\right) = 1 \quad \overset{\rho=\frac{\lambda}{\mu}}{\Longrightarrow}$$

$$P_0 * \sum_{k=0}^{c} \left(\frac{\rho^k}{k!}\right) = 1 \quad \Rightarrow$$

$$P_0 = \frac{1}{\sum_{k=0}^{c} \frac{\rho^k}{k!}} \qquad (\mathbf{4})$$

Η πιθανότητα απόρριψης ενός πελάτη από το σύστημα είναι η πιθανότητα το σύστημα να βρίσκεται στην κατάσταση k:

$$P_k = P_0 * \frac{\rho^k}{k!} = \frac{\frac{\rho^k}{k!}}{\sum_{k=0}^{c} \frac{\rho^k}{k!}}$$

Όπου k βάζουμε c και προκύπτει ο τύπος Erlang-B:

$$P_{blocking} = B(\rho, c) = \frac{\frac{\rho^c}{c!}}{\sum_{k=0}^{c} \frac{\rho^k}{k!}} \quad , \text{όπου } \rho = \frac{\lambda}{\mu}$$

Ο μέσος αριθμός απωλειών πελατών από μια ουρά ισούται με την πιθανότητα απόρριψης πελάτη πολλαπλασιασμένη με το ρυθμό άφιξης των πελατών:

$$\lambda - \gamma = \lambda - \lambda(1 - P_{blocking}) = \lambda * P_{blocking} = \lambda * \frac{\frac{\rho^c}{c!}}{\sum_{k=0}^{c} \frac{\rho^k}{k!}}$$

Κώδικας octave για την υλοποίηση της συνάρτησης erlangb_factorial και την επιβεβαίωση της με τη συνάρτηση erlangb του πακέτου queueing του octave:

```
2  clc;
3  clear all;
4  close all;
5  pkg load queueing;
6
7  % r = lambda / mu
8  % c: number of servers
9
10 function p = erlangb_factorial (r,c)
11   s = 0;
12   for k = 0:1:c
13     s = s + (power(r,k)/factorial(k));
14   endfor
15   p = (power(r,c)/factorial(c))/s;
16 endfunction
17
18
19 display("erlangb_factorial(9,9) =");
20 disp(erlangb_factorial(9,9));
21
22 display("erlangb(9,9) =");
23 disp(erlangb(9,9));
```

Τα αποτελέσματα των δύο παραπάνω συναρτήσεων όπως παρατηρούμε είναι ίδια:

```
erlangb_factorial(9,9) =
0.2243
erlangb(9,9) =
0.2243
>>
```

2. Κώδικας octave για την υλοποίηση της συνάρτησης erlangb_iterative και την επιβεβαίωση της με τη συνάρτηση erlangb του πακέτου queueing του octave:

```
function p = erlangb_iterative (r,c)
  p = 1;
  for i=0:1:c
    p = ((r*p)/((r*p)+i));
  endfor
endfunction


display("erlangb(9,9) =");
disp(erlangb(9,9));

display("erlangb_iterative(9,9) =");
disp(erlangb_iterative(9,9));
```

Τα αποτελέσματα των δύο παραπάνω συναρτήσεων όπως παρατηρούμε είναι ίδια και μεταξύ τους αλλά και με προηγουμένως με την erlangb_factorial:

```
erlangb(9,9) =
0.2243
erlangb_iterative(9,9) =
0.2243
>>
```

3. Κώδικας octave:

```
display("erlangb_factorial(1024,1024) =");
disp(erlangb_factorial(1024,1024));

display("erlangb_iterative(1024,1024) =");
disp(erlangb_iterative(1024,1024));
```

Τα αποτελέσματα που προκύπτουν είναι τα εξής:

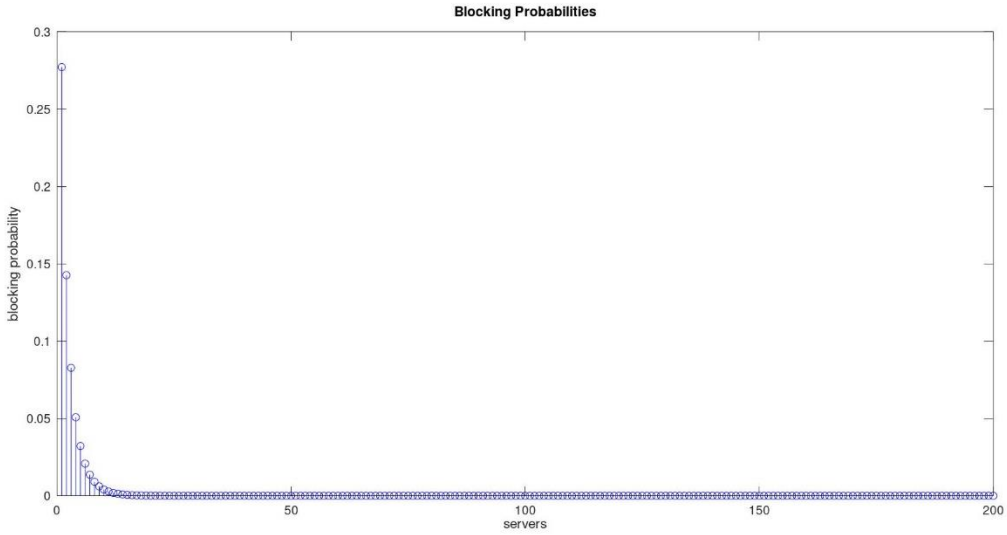
```
erlangb_factorial(1024,1024) =
NaN
erlangb_iterative(1024,1024) =
0.024524
>>
```

Παρατηρούμε ότι στην περίπτωση της συνάρτησης erlangb_factorial(1024,1024) κατά τη διάρκεια του υπολογισμού του 1024! προκύπτει ένας πολύ μεγάλος αριθμός πριν τον υπολογισμό του τελικού πηλίκου οπότε δεν υπάρχει σαφές αποτέλεσμα. Αντίθετα η συνάρτηση erlangb_iterative(1024,1024) υπολογίζει κανονικά την πιθανότητα.

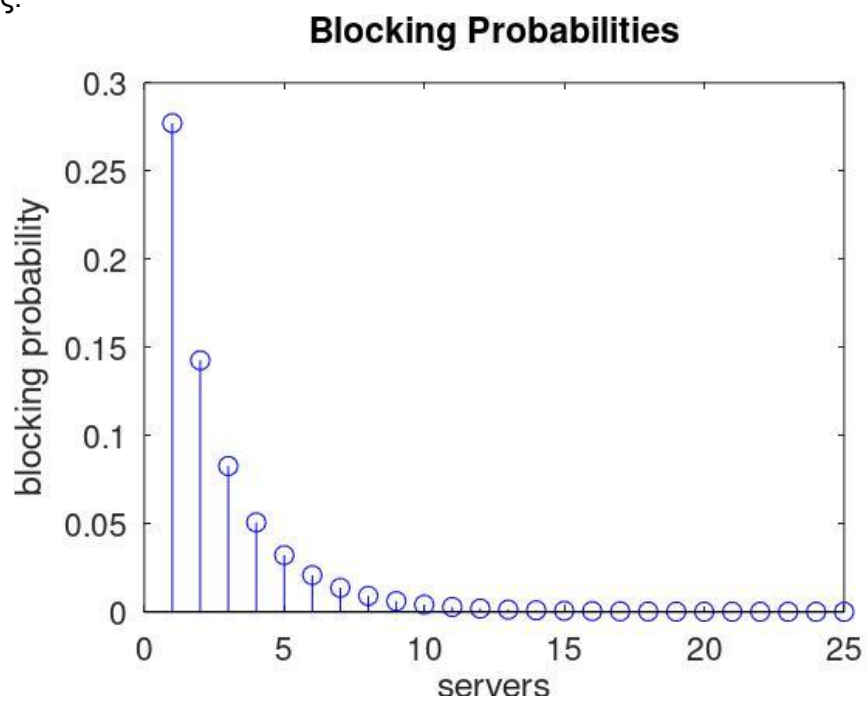
**4.** **α)** Χρησιμοποιώντας ως πρότυπο τον πιο απαιτητικό χρήστη η συνολική ένταση του φορτίου που καλείται να εξυπηρετήσει το τηλεφωνικό δίκτυο της εταιρείας είναι ίση με:

$$\rho = \frac{200 * 23}{60} = 76{,}67\ Erlangs$$

**β)** Διάγραμμα της πιθανότητας απόρριψης πελάτη από το σύστημα ως προς τον αριθμό των τηλεφωνικών γραμμών, επιλέγοντας από 1 έως 200 τηλεφωνικές γραμμές, έχοντας χρησιμοποιήσει τη συνάρτηση erlangb_iterative για τον υπολογισμό της:




**γ)** Στο παραπάνω διάγραμμα παρατηρούμε ότι για να είναι η πιθανότητα απόρριψης τηλεφωνικής κλήσης κάτω από 1% ο οικονομικότερος αριθμός τηλεφωνικών γραμμών είναι 8.
Μεγεθύνουμε το παραπάνω διάγραμμα για να φανεί καλύτερα η παρατήρηση μας:

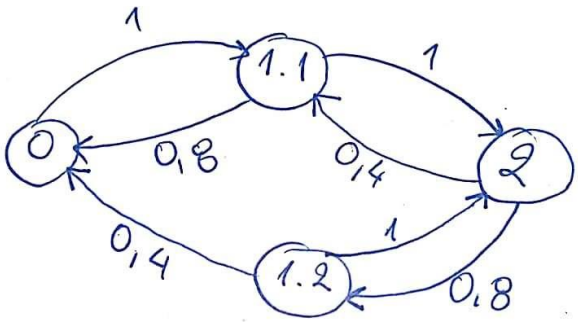


Κώδικας octave:

```
#Ανάλυση και Σχεδιασμός τηλεφωνικού κέντρου
clc;
clear all;
close all;
pkg load queueing;

% r = lambda / mu
% c: number of servers

function p = erlangb_factorial (r,c)
  s = 0;
  for k = 0:1:c
    s = s + (power(r,k)/factorial(k));
  endfor
  p = (power(r,c)/factorial(c))/s;
endfunction


function p = erlangb_iterative (r,c)
  p = 1;
  for i=0:1:c
    p = ((r*p)/((r*p)+i));
  endfor
endfunction

display("erlangb_factorial(9,9) =");
disp(erlangb_factorial(9,9));

display("erlangb(9,9) =");
disp(erlangb(9,9));

display("erlangb_iterative(9,9) =");
disp(erlangb_iterative(9,9));

display("erlangb_factorial(1024,1024) =");
disp(erlangb_factorial(1024,1024));

display("erlangb_iterative(1024,1024) =");
disp(erlangb_iterative(1024,1024));

P = zeros(0,200);

for i = 1:1:200
  P(i) = erlangb_iterative (i*(23/60),i)
endfor




figure(1);
stem(P,'b',"linewidth",0.4);
title("Blocking Probabilities")
xlabel("servers");
ylabel("blocking probability");
```

**<u>Σύστημα εξυπηρέτησης με δύο ανόμοιους εξυπηρετητές</u>**

1. Διάγραμμα ρυθμών μεταβάσεων του συστήματος στην κατάσταση ισορροπίας:






---

**α)**Οι εξισώσεις ισορροπίας είναι οι εξής:

$$\lambda * P_0 = \mu_1 * P_{11} + \mu_2 * P_{12}$$
$$(\lambda + \mu_1) * P_{11} = p * \lambda * P_0 + \mu_2 * P_2$$
$$(\lambda + \mu_2) * P_{12} = (1 - p) * \lambda * P_0 + \mu_1 * P_2$$
$$P_0 + P_{11} + P_{12} + P_2 = 1$$

Για $\mu_1 = 0.8$ , $\mu_2 = 0.4$ , $\lambda = 1$ , $p = 1$ θα είναι:

$$P_0 = 0.8 * P_{11} + 0.4 * P_{12}$$
$$1.8 * P_{11} = P_0 + 0.4 * P_2$$
$$1.4 * P_{12} = 0.8 * P_2$$
$$P_0 + P_{11} + P_{12} + P_2 = 1$$

Και τελικά προκύπτει:

$$P_0 = 0.24951$$
$$P_{11} = 0.21442$$
$$P_{12} = 0.19493$$
$$P_2 = 0.34113$$

**β)** Η πιθανότητα απόρριψης πελάτη από το σύστημα είναι ίση με:

$$P_2 = 0.34113$$

**γ)** Ο μέσος αριθμός πελατών στο σύστημα είναι ίσος με:

$$\sum_{k=0}^{2} k * P(k) = 0 * P_0 + 1 * P_{11} + 1 * P_{12} + 2 * P_2 = 1.0917$$

**2. α)** Συμπληρώνουμε τα thresholds ως εξής:

- threshold_0: Από την κατάσταση 0 μπορεί να πραγματοποιηθεί μόνο άφιξη επομένως το threshold αυτό θα ισούται με 1.

$$threshold_0 = \frac{\lambda}{1}$$

- threshold_1a: Από την κατάσταση 1_1 μπορούμε να έχουμε είτε αναχώρηση είτε άφιξη και να μεταβούμε στην κατάσταση 0 με ρυθμό μ1 είτε στην κατάσταση 2 με ρυθμό λ αντίστοιχα. Επομένως ο παρονομαστής του threshold ισούται με λ + μ1 και ο αριθμητής ισούται με λ

$$threshold_{1a} = \frac{\lambda}{\lambda + \mu_1}$$

- threshold_1b: Από την κατάσταση 1_2 μπορούμε να έχουμε είτε αναχώρηση είτε άφιξη και να μεταβούμε στην κατάσταση 0 με ρυθμό μ2 είτε στην κατάσταση 2 με ρυθμό λ αντίστοιχα. Επομένως ο παρονομαστής του threshold ισούται με λ + μ2 και ο αριθμητής ισούται με λ

$$threshold_{1b} = \frac{\lambda}{\lambda + \mu_2}$$

- threshold_2_first: Από την κατάσταση 2 μπορούμε να έχουμε άφιξη και απόρριψη του πελάτη με ρυθμό λ, επομένως ο αριθμητής του threshold_2_first θα είναι λ. Μπορούμε να έχουμε αναχώρηση και να πάμε στην κατάσταση 1_1 με ρυθμό μ2, άρα ο παρονομαστής ισούται με λ + μ1 + μ2.

$$threshold_{2first} = \frac{\lambda}{\lambda + \mu_1 + \mu_2}$$

- threshold_2_second: Από την κατάσταση 2 μπορούμε να έχουμε αναχώρηση και να μεταβούμε στην κατάσταση 1_2 με ρυθμό μ1, επομένως ο αριθμητής του threshold_2_second θα είναι λ+μ1. Μπορούμε να έχουμε αναχώρηση και να πάμε στην κατάσταση 1_1 με ρυθμό μ2, άρα ο παρονομαστής ισούται με λ + μ1 + μ2.

$$threshold_{2second} = \frac{\lambda + \mu_1}{\lambda + \mu_1 + \mu_2}$$

**β)**Τα κριτήρια σύγκλισης για την προσομοίωση είναι η διαφορά μεταξύ δύο διαδοχικών μέσων αριθμών πελατών να είναι κάτω από 0.001%.

**γ)** Υπολογίζουμε τις εργοδικές πιθανότητες του συστήματος μέσω της προσομοίωσης και επιβεβαιώνουμε ότι είναι ίσες με τις πιθανότητες που υπολογίσαμε πριν με μικρές αποκλίσεις.

```
0.2436
0.2122
0.1976
0.3465
>>
```

Κώδικας octave:

```
#Σύστημα εξυπηρέτησης με δύο ανόμοιους εξυπηρετητές
clc;
clear all;
close all;
pkg load queueing;

lambda = 1;
m1 = 0.8;
m2 = 0.4;

threshold_0 = lambda/1;
threshold_1a = lambda/(lambda+m1);
threshold_1b = lambda/(lambda+m2);
threshold_2_first = lambda/(lambda+m2+m1);
threshold_2_second = (m1+lambda)/(lambda+m2+m1);

current_state = 0; # what is my state now?
arrivals = zeros(1,4); # arrivals in each state of the system
total_arrivals = 0; # The total number of client arrivals in the system
maximum_state_capacity = 2; # maximum number of clients in the system
previous_mean_delay = 0; # estimated mean delay in the previous time interva
delay_counter = 0;
time = 0;

while 1 > 0
  time = time + 1;

  if mod(time,1000) == 0
    for i=1:1:4
      P(i) = arrivals(i)/total_arrivals;
    endfor

    delay_counter = delay_counter + 1;

    mean_delay = 0*P(1) + 1*P(2) + 1*P(3) + 2*P(4);

    delay_table(delay_counter) = mean_delay;

    if abs(mean_delay - previous_mean_delay) < 0.00001
       break;
    endif
    previous_mean_delay = mean_delay;
  endif

  random_number = rand(1);

  if current_state == 0
    if random_number < threshold_0
      current_state = 1;
      arrivals(1) = arrivals(1) + 1;
      total_arrivals = total_arrivals + 1;
    endif
  elseif current_state == 1
    if random_number < threshold_1a
      current_state = 3;
      arrivals(2) = arrivals(2) + 1;
      total_arrivals = total_arrivals + 1;
    else
      current_state = 0;
    endif
  elseif current_state == 2
    if random_number < threshold_1b
      current_state = 3;
      arrivals(3) = arrivals(3) + 1;
      total_arrivals = total_arrivals + 1;
    else
      current_state = 0;
    endif
  else
      if random_number < threshold_2_first
          arrivals(4) = arrivals(4) + 1;
          total_arrivals = total_arrivals + 1;
        elseif random_number < threshold_2_second
          current_state = 2;
        else
          current_state = 1;
        endif
    endif

endwhile

display(P(1));
display(P(2));
display(P(3));
display(P(4));
```