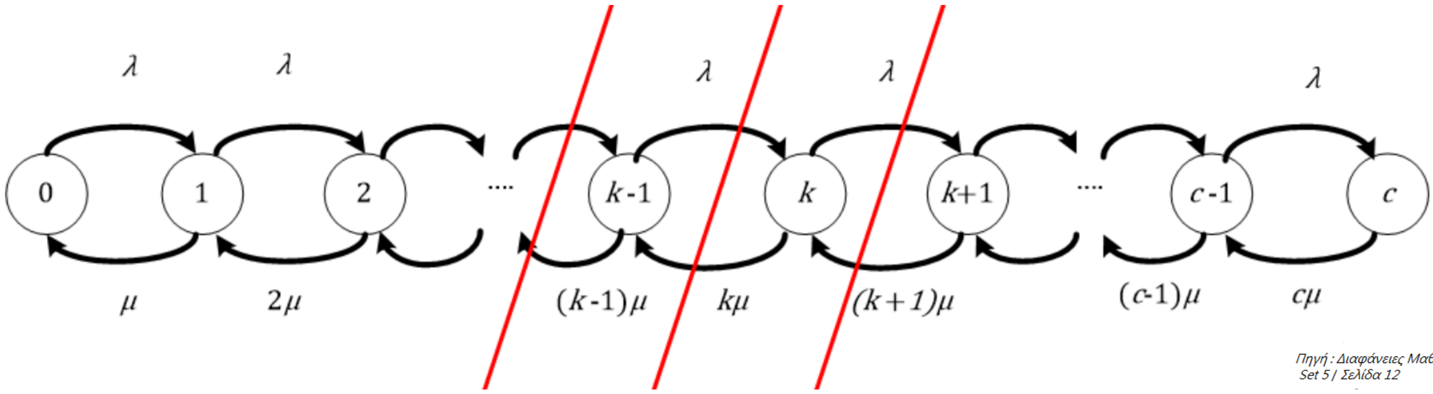
Συστήματα Αναμονής
Εργαστηριακές Ασκήσεις
4η Ομάδα Ασκήσεων

8^{o} Εξάμηνο Η.Μ.Μ.Υ. 2021 - 2022	*Αγγλογάλλος Αναστάσιος*	*031 18641*

<u>Ανάλυση και Σχεδιασμός Τηλεφωνικού Κέντρου</u>

1. Το διάγραμμα ρυθμού μεταβάσεων του συστήματος M/M/c/c :



Πηγή : Διαφάνειες Μαθήματος Set 5 / Σελίδα 12

Αρχικά ισχύει ότι :

$$k\mu P_k = \lambda P_{k-1} \Rightarrow P_k = (\lambda/k\mu)P_{k-1} \Rightarrow P_k = (\rho/k)*P_{k-1}\ ,\ k = 1,2,...,c$$

Επιπλέον, ισχύει ότι :

$$P_0+P_1+...+P_c = 1 \Rightarrow \sum_{k=0}^{c} Pk = 1 \Rightarrow P_0 = 1 / \sum_{k=0}^{c} (\rho^k / k!)$$

Άρα :

$$P_{rejecting} = P_c = (\rho^c / c!)\, Po \Rightarrow P_{Blocking} = (\rho^c / c!) / \sum_{k=0}^{c} (\rho^k / k!)$$

Σύμφωνα με τα παραπάνω , ο μέσος ρυθμός απόρριψης πελατών από την ουρά είναι ίσος με $\lambda * P_{Blocking}$

Κώδικας :

```
function Answer = erlangb_factorial (r, c)
  a = (r^c)/factorial(c);
  b = 0;
  i = 0;
  for i = 1:1:c
      b = b + (r^c)/factorial(i);
  endfor
  Answer = a/b;
endfunction
```

2. Επαναληπτική Υλοποίηση του erlangb :

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

3. Αποτελέσματα Octave :

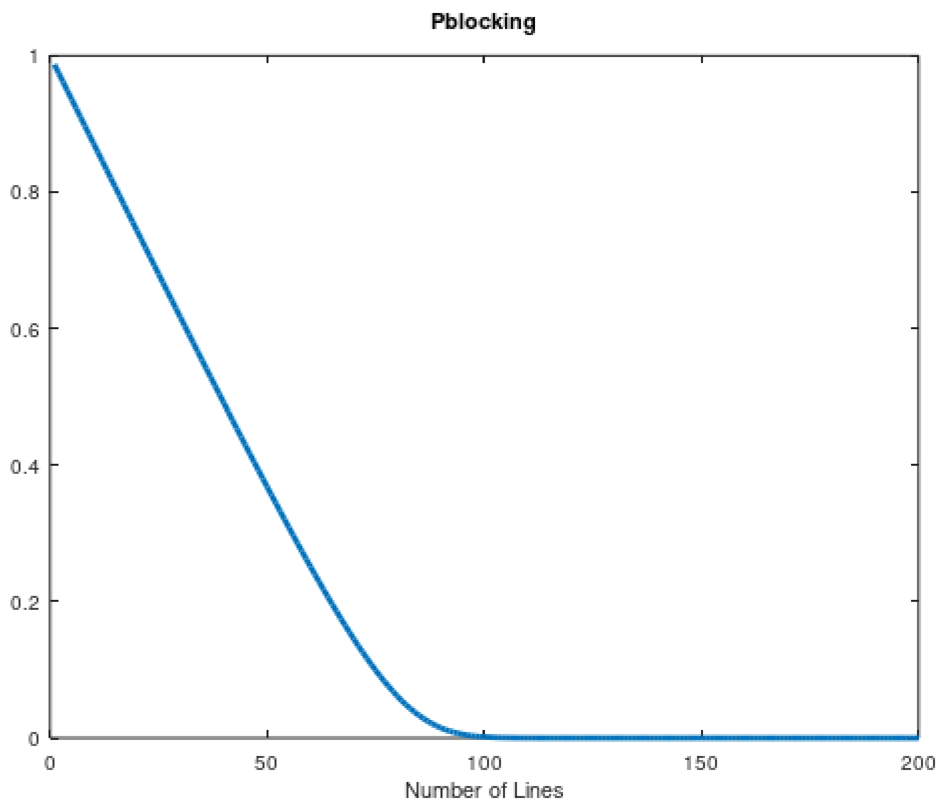
erlangb_factorial : `NaN`

erlangb_iterative : `0.024524`

Παρατήρηση : Το αποτέλεσμα της erlangb_factorial είναι NaN επειδή χρησιμοποιεί πολύ μεγάλους αριθμούς εξαιτίας του παραγοντικού.

4. Έχοντας για βάση τον πιο απαιτητικό χρήστη η συνολική ένταση φορτίου που καλείται να εξυπηρετήσει το τηλεφωνικό δίκτυο της εταιρείας είναι:

$$\varrho = 200 \cdot 23/60 = 76.666 \ \mathit{Erlang}.$$



Ο κατάλληλος αριθμός τηλεφωνικών γραμμών που απαιτείται και ικανοποιεί την προϋπόθεση ότι η πιθανότητα απόρριψης πελάτη από το σύστημα να είναι μικρότερη του 0.01 είναι :

Αποτέλεσμα Octave :

```
Minimum Number of Lines with Pblocking < 0.01 is 93
```

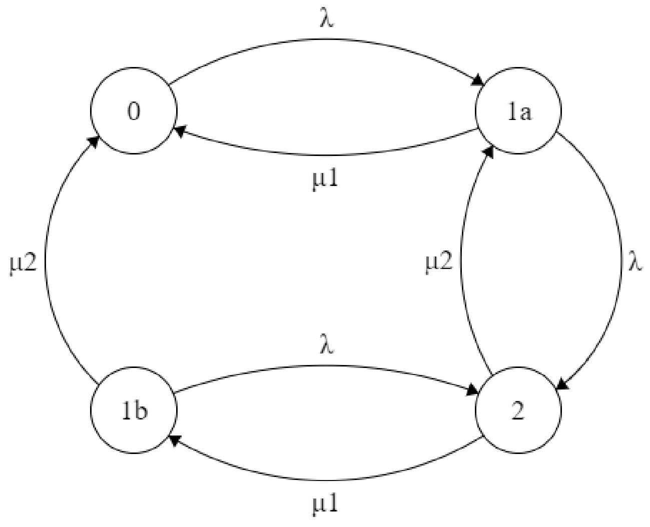
Κώδικας Άσκησης (Queuing Systems 4a) :

```
pkg load queueing;
clc;
clear all;
close all;


function Answer = erlangb_factorial (r, c)
  a = (r^c)/factorial(c);
  b = 0;
  i = 0;
  for i = 1:1:c
      b = b + (r^c)/factorial(i);
  endfor
  Answer = a/b;
endfunction

display(erlangb_factorial (1024, 1024));
```

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

display(erlangb_iterative (1024, 1024));

found_answer = false;
r = 200*23/60;
c = 1:200;
for i = 1:200
  B(i) = erlangb_iterative(r,i);
  if B(i) < 0.01 && !found_answer
    answer = i;
    found_answer = true;
  endif
endfor

figure(1);
plot(c, B, "linewidth",1.4);
title("Pblocking");
xlabel("Number of Lines");


display(strjoin({"Minimum Number of Lines with Pblocking < 0.01 is", num2str(answer)}));
```

<u>Συστήματα Εξυπηρέτησης με δύο ανόμοιους εξυπηρετητές</u>

1.Το διάγραμμα ρυθμών μεταβάσεων του συστήματος στην κατάσταση ισορροπίας:



Άρα οι εργοδικές πιθανότητες του συστήματος προκύπτουν ως εξής:
(Ισχύει $\lambda = 1$, $\mu_1 = 1/1.25$, $\mu_2 = 1/2.5$)

$\lambda P_0 = \mu_1 P_{1a} + \mu_2 P_{1b} \rightarrow P_0 = 0.8P_{1a} + 0.4P_{1b}$
$\mu_1 P_2 + \mu_2 P_2 = \lambda P_{1a} + \lambda P_{1b} \rightarrow P_2 = (5/6)\ (P_{1a} + P_{1b})$
$\mu_1 P_{1a} + \lambda P_{1a} = \lambda P_0 + \mu_2 P_2 \rightarrow P_{1a} = (5/9)^* P_0 + (2/9)^* P_2$
$\mu_2 P_{1b} + \lambda P_{1b} = \mu_1 P_2 \rightarrow P_{1b} = (4/7)^* P_2$

Συνεπώς, προκύπτει :
$P_{1\alpha} = 0.86P_0$
$P_{1b} = 0.78P_0$
$P_2 = 1.37P_0$

Επιπλέον:
$P_0 + P_{1a} + P_{1b} + P_2 = 1 \rightarrow P_0 = 0.249$
$P_{1a} = 0.214$
$P_{1b} = 0.194$
$P_2 = 0.341 = P_{\mathrm{Blocking}}$

Ο μέσος αριθμός πελατών στο σύστημα υπολογίζεται από τη σχέση:

$$E[N] = \sum_{k=0}^{2} kP_k = 1.09$$

2.Τα κενά του προγράμματος (demo4.m) συμπληρώθηκαν με τον παρακάτω κώδικα :

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

Τα παραπάνω αποτελούν τα κριτήρια σύγκλισης του προγράμματος.

Οι πιθανότητες που υπολογίζονται από το πρόγραμμα είναι :
Αποτελέσματα Octave :

```
0.2516
0.2159
0.1911
0.3414
```

Οι οποίες βρίσκονται πολύ κοντά στις πιθανότητες που υπολογίστηκαν.
Κώδικας Άσκησης (Queuing Systems 4b) :

```
clc;
clear all;
close all;


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

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

current_state = 0;
arrivals = zeros(1,4);
total_arrivals = 0;
maximum_state_capacity = 2;
previous_mean_clients = 0;
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_clients = 0*P(1) + 1*P(2) + 1*P(3) + 2*P(4);

    delay_table(delay_counter) = mean_clients;

    if abs(mean_clients - previous_mean_clients) < 0.00001
       break;
    endif
    previous_mean_clients = mean_clients;
  endif

  random_number = rand(1);

  if current_state == 0
      current_state = 1;
      arrivals(1) = arrivals(1) + 1;
      total_arrivals = total_arrivals + 1;
  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));
```