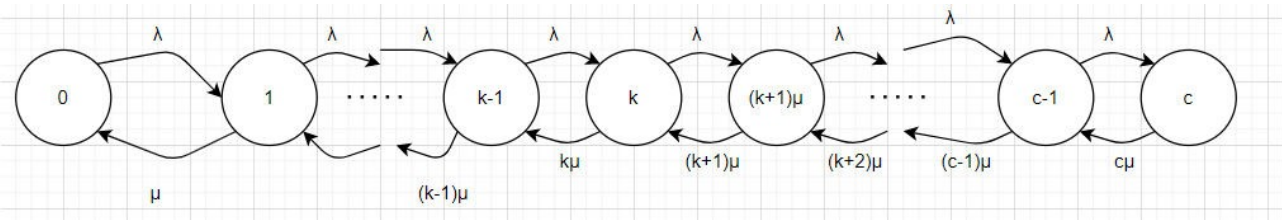
Συστήματα Αναμονής (Queuing Systems)

4ο Εργαστήριο
Θοδωρής Παπαρρηγόπουλος (el18040)

Ερώτηση 1)

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



Από τις λεπτομερείς εξισώσεις ισορροπίας έχουμε: $P_k=\frac{\lambda}{k_{\mu}}*P_{k-1}=\frac{\rho^k}{k!}P_0$.

Από συνθήκη κανονικοποίησης $P_0+P_1+...+P_c=1 \Rightarrow P_0=\frac{1}{\sum_{k=0}^{c}\frac{\rho^k}{k!}}$

.

Από τις 2 σχέσεις $P_c=P[Blocking]=B(\rho,c)=\frac{\rho^c}{c!}*\frac{1}{\sum_{k=0}^{c}\frac{\rho^k}{k!}}$.

Ο μέσος ρυθμός απωλειών δίδεται από τη σχεση λ – γ = λ * P[Blocking] = $\frac{\rho^c}{c!}*\frac{\lambda}{\sum_{k=0}^{c}\frac{\rho^k}{k!}}$.

2) $B(\rho,c)=\frac{\rho^c}{c!}*\frac{1}{\sum_{k=0}^{c}\frac{\rho^k}{k!}}=\frac{1}{\sum_{k=0}^{c}\frac{c!}{\rho^c}\frac{\rho^k}{k!}}=\frac{1}{\sum_{k=0}^{c}\frac{c!}{\rho^{c-k}k!}}$

Για c + 1 έχουμε

$$B(\rho,c+1)=\frac{1}{\sum_{k=0}^{c+1}\frac{(c+1)!}{\rho^{c+1-k}k!}}=\frac{1}{1+\sum_{k=0}^{c}\frac{(c+1)!}{\rho^{c+1-k}k!}}=\frac{1}{1+\frac{c+1}{\rho}\sum_{k=0}^{c}\frac{c!}{\rho^{c-k}k!}}=\frac{1}{1+\frac{c+1}{p}*\frac{1}{B(\rho,c)}}=\frac{\rho B(\rho,c)}{\rho B(\rho,c)+c+1}$$

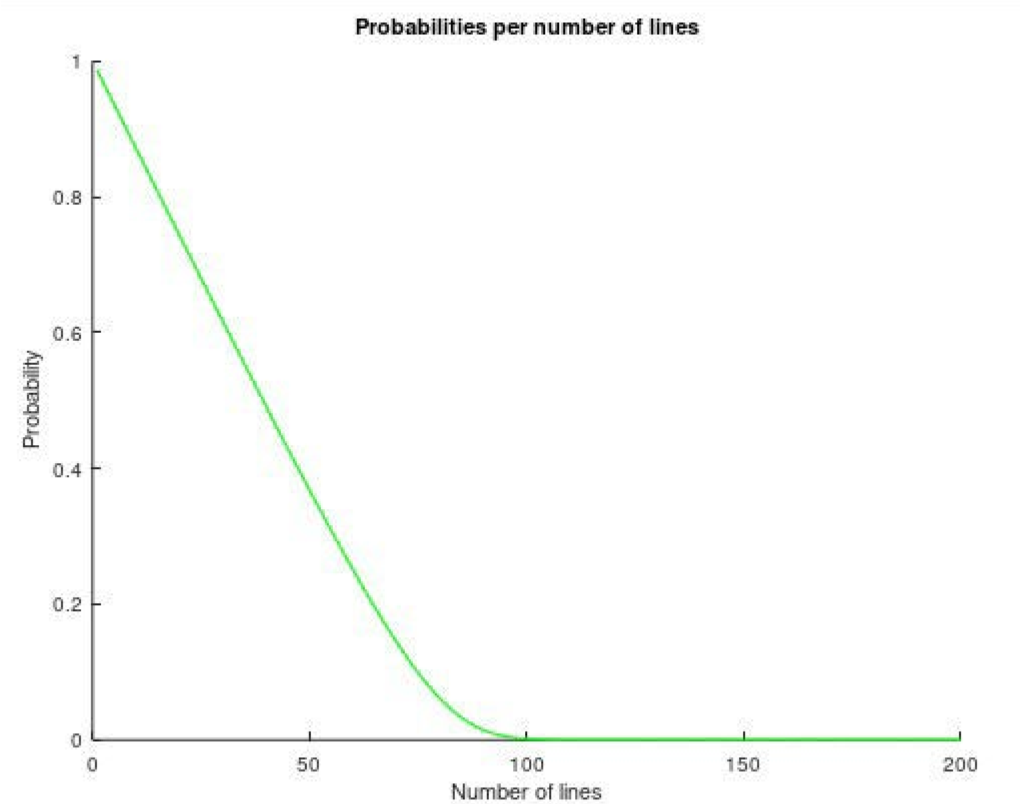
Έτσι, $B(\rho,n)=\frac{\rho B(\rho,n-1)}{\rho B(\rho,n-1)+n}$ και βρίσκουμε πως $B(\rho,0)=\frac{\rho^0}{0!}\cdot\frac{1}{\sum_{k=0}^{0}\frac{\rho^k}{k!}}=1$.

3) Παρατηρούμε πως πράγματι η erlangb_iterative μας δίνει τη σωστή τιμή 0.024524. Όμως, η erlangb_factorial δίνει τιμή NaN. Αυτό συμβαίνει επειδή έχουμε να υπολογίσουμε πολύ μεγάλες τιμές όπως το 1024! με αποτέλεσμα να έχουμε overflow και αδυναμία υπολογισμού της τιμής.

4) Μοντελοποιώ το σύστημα με λ = 1 κλήση την ώρα και διάρκεια 1/μ = 23/60 ώρες.
Έτσι ρ = λ/μ = 23/60 = 0.38333 Erlangs. Με διαφορετική μοντελοποίηση καταλήγουμε στο ίδιο αποτέλεσμα. Αυτό είναι η συνεισφορά μόνο από έναν χρήστη. Έτσι

$$\rho_{o\lambda} = 200 * \rho = 76.666667\ Erlangs.$$

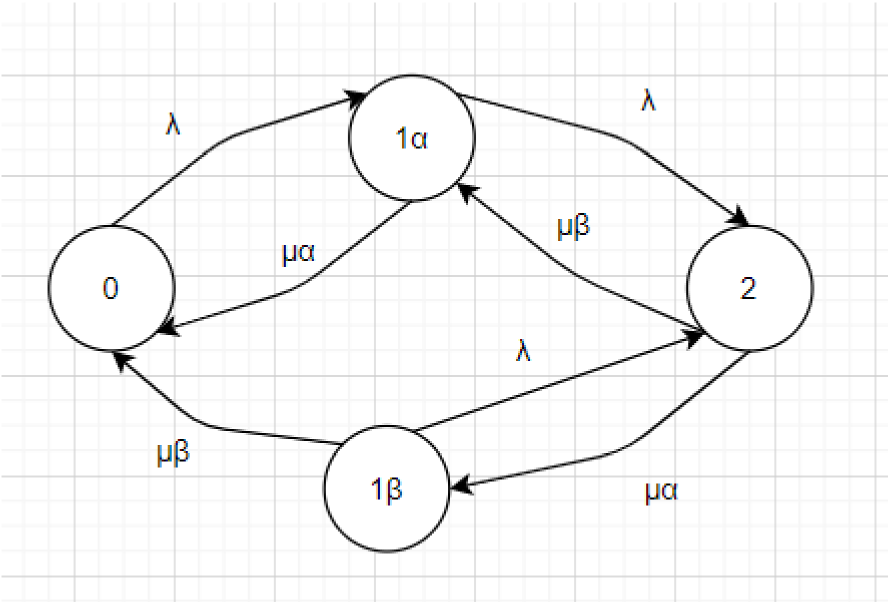
Προκύπτει,



Ο ελάχιστος αριθμός γραμμών προκειμένου να έχουμε P[Blocking] < 1% είναι:
minlines = 93.

Ερώτηση 2)

1) Το διάγραμμα ρυθμού μεταβάσεων σε κατάσταση ισορροπίας είναι:



Αξιοποιώντας της global εξισώσεις ισορροπίας έχουμε:

$$\lambda P_0 = m\,a\,P_{1a} + \mu\,b\,P_{1b} \Rightarrow P_0 = 0.8\,P_{1a} + 0.4\,P_{1b}$$

$$(\mu a + \mu b)P_2 = \lambda\,P_{1a} + \lambda P_{1b} \Rightarrow P_2 = \frac{5}{6}(P_{1a} + P_{1b})$$

$$\mu a P_{1a} + \lambda P_{1a} = \lambda P_0 + \mu b P_2 \Rightarrow P_{1a} = \frac{5}{9}P_0 + \frac{2}{9}P_2$$

$$P_0 + P_{1a} + P_{1b} + P_2 = 1$$

$$\text{Έτσι},$$

$$P_0 = 0.24951$$

$$P_{1a} = 0.21443$$

$$P_{1b} = 0.19493$$

$$P_2 = 0.34113 = Pblocking$$

$$\text{Ο μέσος αριθμός πελατών είναι}:$$

$$E[n(t)] = 1\cdot(0.19483 + 0.21442) + 2\cdot 0.34113 = 1.09161\,\pi\varepsilon\lambda\acute{\alpha}\tau\varepsilon\varsigma$$

2) Αρχικά επιβεβαιώνουμε τα παραπάνω αποτελέσματα με χρήση της octave:

```
 0.25160
 0.21453
 0.19378
 0.34009
mean_clients =  1.0885
```

Στη συνέχεια γεμίζουμε τα κενά που ζητούνται ως:

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

Τα παραπάνω thresholds δικαιολογούνται ως εξής:

Αν ήμαστε στην κατάσταση 1α ή 1β χωρίζουμε το διάστημα (0,1) σε 2 κομμάτια όπου το μέγεθος του πρώτου είναι ίσο ως $\frac{\lambda}{\lambda+\mu_{a,b}}$ και αντιστοιχεί στη περίπτωση που έχουμε άφιξη ενώ το υπόλοιπο είναι για το εάν έχουμε αναχώρηση.

Όσον αφορά τώρα τη κατάσταση 2 (εδώ χρειαζόμαστε και το λ και το μα και το μ_b καθώς από την κατάσταση 2 μπορούμε να πάμε και στην 1α και στην 1β) αφού αποφασίσουμε για το εάν έχουμε άφιξη ή αναχώρηση σπάμε το διάστημα των αφίξεων σε 2 κομμάτια, όπου το μέγεθός τους εξαρτάται από τα μα και μ_b (τα thresholds φαίνονται παραπάνω). Εάν ήμαστε ενδιάμεσα από το threshold_2_first και threshold_2_second τότε έχουμε αναχώρηση προς το 1β (δηλαδή έχουμε μα) αλλιώς έχουμε αναχώρηση προς το 1α (δηλαδή έχουμε μ_b).

Το κριτήριο σύγκλισης της προσομοίωσης μας είναι ο αριθμός των πελατών στο σύστημα δεν μεταβάλλεται παραπάνω από 1/10000 της προηγούμενης τιμής του.

Όπως προανέφερα και παραπάνω, οι θεωρητικές τιμές των εργοδικών πιθανοτήτων και του μέσου αριθμού πελατών καθώς και οι τιμές από την προσομοίωση είναι πολύ κοντά αλλά όχι ίδιες ακριβώς και αυτό οφείλεται σε προσεγγίσεις κατά τον θεωρητικό υπολογισμό αλλά και στο κριτήριο σύγκλισης τής προσομοίωσης μας λόγο του οποίου δεν φτάνει σε ακριβής τιμές (θα έπρεπε να αφήσουμε την προσομοίωση επάπειρον προκειμένου να βρει τις ακριβής τιμές).

Κώδικας

demo4.m

```
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/(m1 + m2 + lambda);
threshold_2_second = (m1+lambda)/(m1 + m2 + lambda);

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));
display(mean_clients);
```

erlangb_factorial.m

```
clc;
clear all;
close all;


pkg load queueing

function erlangb_factorial(r, c)
  nominator = (r^c)/factorial(c)
  Pblock = 0;
  denominator = 0;
  for k=0:c
    denominator = (denominator + (r^(k))/factorial(k));
  endfor

  Pblock = nominator/denominator;

  disp("On hand Pblocking")
  disp(Pblock)

  Pblocking = erlangb(r, c);

  disp("erlangb Pblocking")
  disp(Pblocking)

endfunction
```

erlangb_iterative.m

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

function res = erlangb_iterative (p,c)
  beta = 1;
  arr = zeros(length(c));
  figure_counter = 0;
  result = 0;
  counter = 0
  for j=1:length(c)
    for i = 1:c(j)
      beta = (p * beta)/((p * beta) + i);
    endfor
    if beta < 0.01 && counter == 0
     result = c(j)
     counter += 1
    endif
    arr(j) = beta;
  endfor
  %-----------------------------------
  %for first iterative
  res = beta;
  checkres = erlangb(p,c);
  display(checkres);
  %---------------------------------

  disp("minimuc c is:")
  disp(result)
  %-----------------------------------



endfunction
```

task1_4_2.m

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

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

p1 = 200*(23/60);
xstate = 1:200;
for k = 1:200
  to_plot(k) = erlangb_iterative(p1,k);
endfor
figure(1);
hold on;
title("Probabilities per number of lines");
xlabel("Number of lines");
ylabel("Probability");
plot(xstate,to_plot,'g',"linewidth",1.2);

for i = 1:200
  if (to_plot(i) < 0.01)
    minlines = i;
    display(minlines);
    break;
  endif
endfor
```