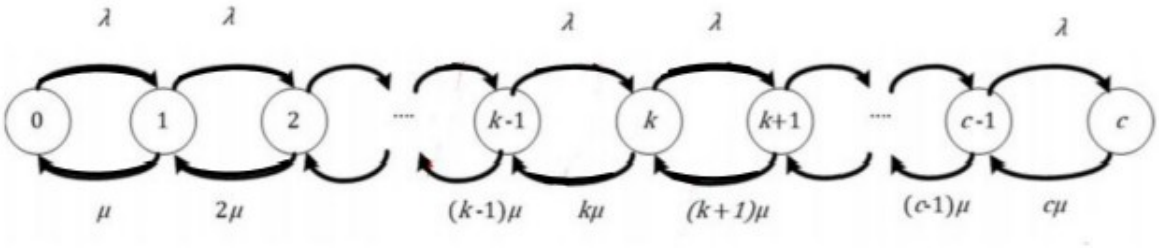
ΣΥΣΤΗΜΑΤΑ ΑΝΑΜΟΝΗΣ
4Η ΟΜΑΔΑ ΑΣΚΗΣΕΩΝ

Ονοματεπώνυμο: Δημήτριος Βασιλείου
Α.Μ: el19830
Εξάμηνο: 6ο
Σχολή: ΗΜΜΥ

Ανάλυση και σχεδιασμός τηλεφωνικού κέντρου:

(1). Σχεδιάζουμε το διάγραμμα του ρυθμού μεταβάσεων του συστήματος M/M/c/c.



Για το σύστημα M/M/c/c έχουμε τις ακόλουθες εξισώσεις ισορροπίας:

- $P_k=\frac{\lambda}{k\mu}P_{k-1}$,k = 1, 2, …, c-1 (1)
- $P_0+P_1+\ldots+P_{c-1}+P_c=1$ (2)

Λύνοντας αναδρομικά την σχέση (1) προκύπτει ότι:

$P_k=\frac{\lambda}{k\mu}P_{k-1}=\frac{\lambda}{k\mu}[(\frac{\lambda}{(k-1)\mu})P_{k-1}]=\frac{\lambda^2}{k(k-1)\mu^2}P_{k-2}=\ldots=\frac{\lambda^k}{k!\mu^k}P_0$. Αντικαθιστώντας στην σχέση (2) προκύπτει ότι:

$P_0+\frac{\lambda}{\mu}P_0+\ldots\frac{\lambda^{(c-1)}}{(c-1)!\mu^{(c-1)}}P_0+\frac{\lambda^c}{c!\mu^c}P_0=1\Rightarrow P_0[1+\frac{\lambda^1}{1!\mu^1}+\ldots\frac{\lambda^c}{c!\mu^c}]=1$ =>

$P_0\sum_{k=0}^{c}(\frac{\lambda^k}{k!\mu^k})=1$. Θέτοντας ρ = λ/μ έχουμε

$P_0\sum_{K=0}^{c}(\frac{\rho^k}{k!})=1\Rightarrow P_0=\frac{1}{\sum_{K=0}^{c}(\frac{\rho^k}{k!})}$

Ισχύει ότι P(blocking) = Pc = $P_0 \frac{\rho^c}{c!} = \frac{\frac{\rho^c}{c!}}{\sum_{k=0}^{c} (\frac{\rho^k}{k!})}$. Τελικά αποδείξαμε ότι

$P_{blocking} = B(\rho, c) = \frac{\frac{\rho^c}{c!}}{\sum_{k=0}^{c} (\frac{\rho^k}{k!})}$, όπου ρ = λ/μ.

Ο μέσος ρυθμός απωλειών πελατών στην ουρά ισούται με την πιθανότητα απόρριψης ενός πελάτη επί τον ρυθμό αφίξεων, δηλαδή $\lambda \cdot P_{blocking} = \lambda \frac{\frac{\rho^c}{c!}}{\sum_{k=0}^{c} (\frac{\rho^k}{k!})}$

Στη συνέχεια παραθέτουμε τον κώδικα για την συνάρτηση erlangb_factorial:

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

Με r συμβολίζουμε την ένταση φορτίου ρ = λ / μ και με c τον αριθμό των εξυπηρετητών. Η ορθότητα της συνάρτησης φαίνεται συγκρίνοντας το αποτέλεσμά της με την συνάρτηση erlangb του Octave:

```
>> erlangb_factorial(4,2)
ans = 0.6154
>> erlangb(4,2)
ans = 0.6154
```

(2). Παραθέτουμε τον κώδικα για την συνάρτηση erlangb_iterative:

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

Παρατηρούμε πως η έξοδός της είναι ίδια με αυτή της erlangb_factorial και της erlangb του Octave:

```
>> erlangb_iterative(4,2)
ans = 0.6154
```

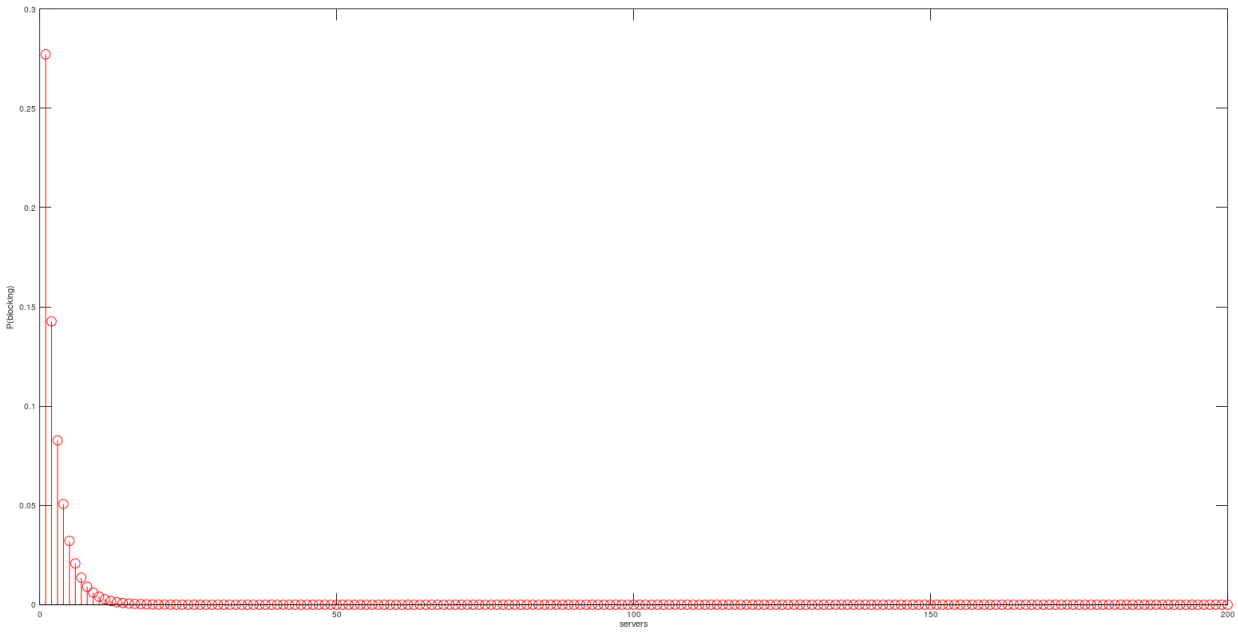
(3). Έχουμε τις ακόλουθες εξόδους για τις δύο συναρτήσεις:

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

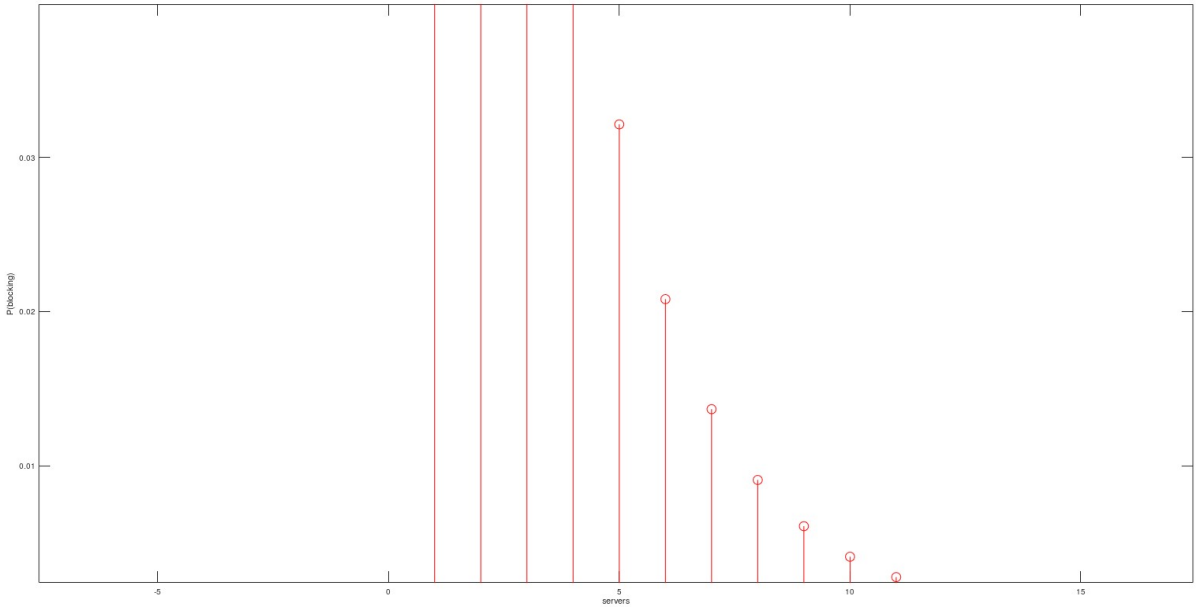
Για την erlangb_factorial εμφανίζεται το αποτέλεσμα NaN(not a number), διότι το παραγοντικό του 1024 είναι πολύ μεγάλος αριθμός και δεν μπορεί να υπολογιστεί από το Octave.
Για την erlangb_iterative εμφανίζεται κανονικά το αποτέλεσμα.

(4).
(α). Έχοντας ως πρότυπο τον πιο απαιτητικό χρήστη, προκύπτει ότι η συνολική ένταση φορτίου ισούται με ρ = (200*23) / 60 ≈ 76.67 Erlangs.

(β). Παραθέτουμε το διάγραμμα της πιθανότητας απόρριψης πελάτη από το σύστημα ως προς τον αριθμό των τηλεφωνικών γραμμών, επιλέγοντας από 1 έως 200 τηλεφωνικές γραμμές:



(γ). Κάνουμε zoom στο διάγραμμα:



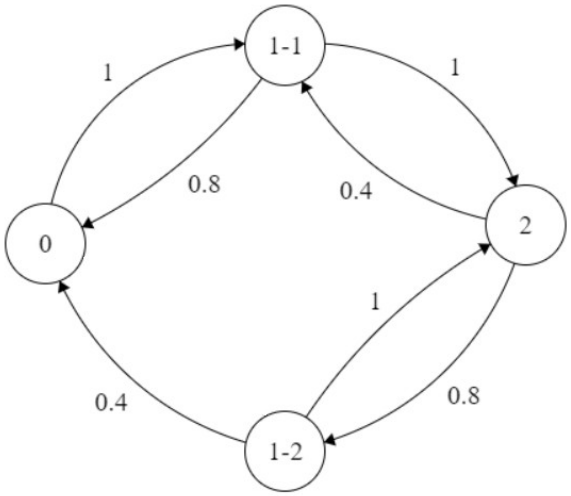
Παρατηρούμε πως για να είναι η πιθανότητα απόρριψης μικρότερη από 1%(0.01) χρειαζόμαστε 8 τηλεφωνικές γραμμές.
Ο κώδικας που χρησιμοποιήθηκε είναι ο ακόλουθος:

```
P = zeros(0, 200)
for i = 1:1:200
  a = i * (23/60)
  P(i) = erlangb_iterative(a, i)
endfor

figure(1);
stem(P, 'r', "linewidth", 0.4);
xlabel("servers");
ylabel("P(blocking)");
```

Σύστημα εξυπηρέτησης με δύο ανόμοιους εξυπηρετητές:

(1). Από την εκφώνηση βλέπουμε ότι 1/μ1 = 1.25 άρα μ1 = 0.8 πελάτες/sec και 1/μ2 = 2.5 άρα μ2 = 0.4 πελάτες/sec. Το διάγραμμα μεταβάσεων του συστήματος στην κατάσταση ισορροπίας είναι το ακόλουθο:



Περιγραφή των καταστάσεων:

- 0 : και οι δύο εξυπηρετητές είναι άδειοι
- 1-1 : εξυπηρετεί μόνο ο 1
- 1-2 : εξυπηρετεί μόνο ο 2
- 2 : εξυπηρετούν και οι 2

(α). Έχουμε το ακόλουθο σύστημα από τις εξισώσεις ισορροπίας:

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

Αντικαθιστώντας με λ = 1, μ1 = 0.8 και μ2 = 0.4 λύνουμε το σύστημα και βρίσκουμε τις ακόλουθες τιμές για τις εργοδικές πιθανότητες:

- $P_0 = 0.249$
- $P_{11} = 0.214$
- $P_{12} = 0.195$
- $P_2 = 0.341$

(β). Απόρριψη στο σύστημα θα έχουμε όταν και οι δύο εξυπηρετητές είναι γεμάτοι. Επομένως Pblocking = P_2 = 0.341.

(γ). Ο μέσος αριθμός πελατών στο σύστημα ισούται με

$1 \cdot P_{11} + 1 \cdot P_{12} + 2 \cdot P_{22} = 1.091$

(2).

(α). Συμπληρώνουμε τα thresholds:

```
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);
```

(β). Κριτήριο σύγκλισης της προσομοίωσης είναι η απόλυτη τιμή της διαφοράς μεταξύ δύο διαδοχικών μέσων αριθμών πελατών στο σύστημα να είναι μικρότερη από 0.00001.

(γ). Τρέχουμε την προσομοίωση και λαμβάνουμε ως αποτέλεσμα τις εργοδικές πιθανότητες του συστήματος. Παρατηρούμε ότι οι τιμές τους είναι αρκετά κοντά με αυτές που υπολογίσαμε στο προηγούμενο ερώτημα.

```
0.2483
0.2168
0.1955
0.3394
```

Ο κώδικας που χρησιμοποιήθηκε είναι ο ακόλουθος:

```
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));
```