ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΣΙΤΩΝ
ΡΟΗ Δ - ΣΥΣΤΗΜΑΤΑ ΑΝΑΜΟΝΗΣ (QUEUING SYSTEMS)

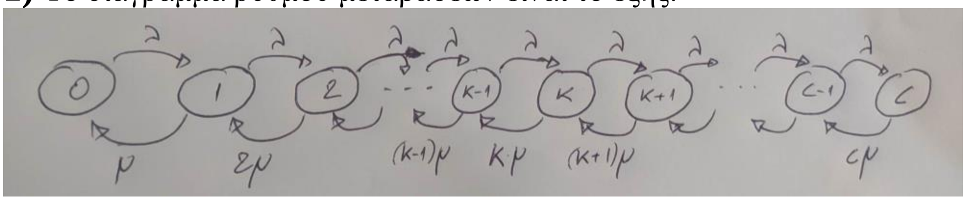
ΓΕΩΡΓΑΚΟΠΟΥΛΟΣ ΓΕΩΡΓΙΟΣ 03120827

ΑΝΑΦΟΡΑ 4ΗΣ ΕΡΓΑΣΤΗΡΙΑΚΗΣ ΑΣΚΗΣΗΣ

ΑΣΚΗΣΗ 1:
ΑΝΑΛΥΣΗ ΚΑΙ ΣΧΕΔΙΑΣΜΟΣ ΤΗΛΕΦΩΝΙΚΟΥ ΚΕΝΤΡΟΥ

Στην ασκηση αυτή θα ασχοληθούμε με την ανάλυση και το σχεδιασμό ενός τηλεφωνικού κέντρου, που μοντελοποιείται ως μία ουρά M/M/c/c. Η ουρά αυτή διαθέτει c εξυπηρετητές ίδιων δυνατοτήτων και μέγιστη χωρητικότητα c πελάτες. Οι αφίξεις στο σύστημα ακολουθούν την κατανομή Poisson με ομοιόμορφο μέσο ρυθμό λ και οι εξυπηρετήσεις ακολουθούν την εκθετική κατανομή με ομοιόμορφο μέσο ρυθμό μ.

1) Το διάγραμμα ρυθμού μεταβάσεων είναι το εξής:



Με βάση τις εξισώσεις ισορροπίας:

$$P_k = \left[\frac{\lambda}{k\mu}\right] P_{k-1} = \left(\frac{\rho^k}{k!}\right) P_0, \ k = 1, 2, \ldots, c \qquad \rho \triangleq \frac{\lambda}{\mu} \text{ Erlangs}$$

$$P_0 + P_1 + \cdots + P_{c-1} + P_c = 1$$

(Απο διαφάνειες μαθήματος)

Προκύπτει ότι:

$$\mathrm{P}_0 = \frac{1}{\sum_{k=1}^{c} \frac{\rho^k}{k!}}$$

και οτι

$$P_c = P_{\text{blocking}} = \frac{\rho^c / c!}{\sum_{k=0}^{c} \frac{\rho^k}{k!}} \triangleq \mathrm{B}(\rho, c)$$

(Απο διαφάνειες μαθήματος)

Ισχύει ότι ο μέσος αριθμό απωλειών πελατών από μια ουρά είναι η πιθανότητα απόρριψης από πελάτη επί με τον ρυθμό άφιξης πελατών. Επομένως ισχύει οτι:

$$\lambda \cdot P_{blocking} = \lambda \cdot \frac{\frac{p^c}{c!}}{\sum_{\kappa=0}^{c} \frac{p^k}{k!}}$$

Για να υπολογίσουμε την πιθανότητα απόρριψης πελάτη από το σύστημα B(ρ,c) υλοποιούμε την συνάρτηση erlangb_factorial με βάση τον τύπο που βρήκαμε πριν. Συγκρίνουμε το αποτέλεσμα αυτής της συναρτησης με την συνάρτηση erlangb του πακέτου queueing του Octave.

```
Command Window
>> erlangb_factorial(7,9)
ans = 0.1221
>> erlangb(7,9)
ans = 0.1221
>> erlangb_factorial(6,6)
ans = 0.2649
>> erlangb(6,6)
ans = 0.2649
>>
```

Παρατηρούμε ότι τα αποτελέσματα συμπίπτουν.
Ο κώδικας που χρησιμοποιήσαμε για το ερώτημα αυτό είναι ο εξής:

```
#Ερώτημα 1
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
```

2) Υλοποιούμε την συνάρτηση erlangb_iterative με βάση τον επαναληπτικό τύπο της εκφώνησης για να υπολογίσουμε την πιθανότητα απόρριψης πελάτη από το σύστημα B(ρ,c) Συγκρίνουμε την συνάρτηση μας την συνάρτηση erlangb του πακέτου queueing του Octave.

```
Command Window
>> erlangb_iterative(7,9)
ans = 0.1221
>> erlangb(7,9)
ans = 0.1221
>> erlangb_iterative(6,6)
ans = 0.2649
>> erlangb(6,6)
ans = 0.2649
>>
```

Παρατηρούμε ότι τα αποτελέσματα συμπίπτουν.
Ο κώδικας που χρησιμοποιήσαμε για το ερώτημα αυτό είναι ο εξής:

```
#Ερώτημα 2
function p = erlangb_iterative (r,c)
  p = 1;
  for i=0:1:c
    p = ((r*p)/((r*p)+i));
  endfor
endfunction
```

3) Τρεχουμε τις συναρτήσεις erlangb_iterative και erlangb _actorial με τιμες (1024,1024). Παρατηρούμε ότι ενώ η συνάρτηση erlangb_iterative βγαζει κανονικά αποτέλεσμα η συνάρτηση erlangb_factorial δεν βγαζει αποτέλεσμα (τυπώνει NaN). Αυτό συμβαίνει καθως στην συνάρτηση αυτή πρέπει να υπολογίσουμε το παραγοντικό του 1024, το οποίο είναι ενας πολύ μεγάλος αιρθμός και η Octave δεν το υποστηρίζει

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

4)

Σχεδιάσουμε από την αρχή το τηλεφωνικό δίκτυο μίας εταιρείας στην οποία απασχολούνται 200 εργαζόμενοι. Κάθε εργαζόμενος διαθέτει και μία εξωτερική γραμμή, δηλαδή η εταιρεία πληρώνει για 200 γραμμές. Στην προσπάθεια μας να σχεδιάσουμε ένα πιο οικονομικό δίκτυο, κάνουμε μετρήσεις στην ώρα αιχμής και βρίσκουμε ότι ο πιο απαιτητικός χρήστης χρήστης χρησιμοποιεί συνολικά το τηλέφωνο του για εξωτερικές κλήσεις κατά μέσο όρο 23 λεπτά σε μία ώρα

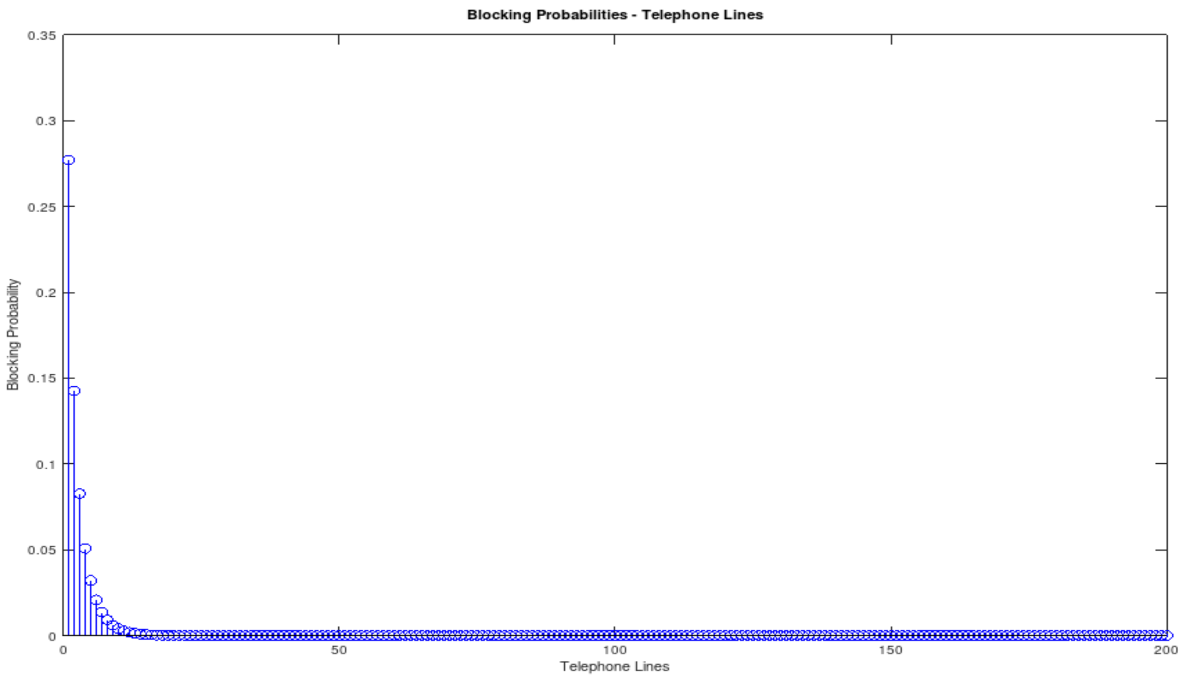
Ερώτημα Α:

Χρησιμοποιώντας ως πρότυπο τον πιο απαιτητικό χρήστη η συνολική ένταση του φορτίου που καλείται να εξυπηρετήσει το τηλεφωνικό δίκτυο της εταιρείας είναι:

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

Ερώτημα Β:

Το διάγραμμα της πιθανότητας απόρριψης πελάτη από το σύστημα ως προς τον αριθμό των τηλεφωνικών γραμμών, επιλέγοντας από 1 έως 200 τηλεφωνικές γραμμές είναι:



Για τον υπολογισμό αυτών των πιθανοτήτων χρησιμοποιήσαμε τη συνάρτηση erlangb_iterative

Ο κώδικας που χρησιμοποιήσαμε για το ερώτημα αυτό είναι ο εξής:

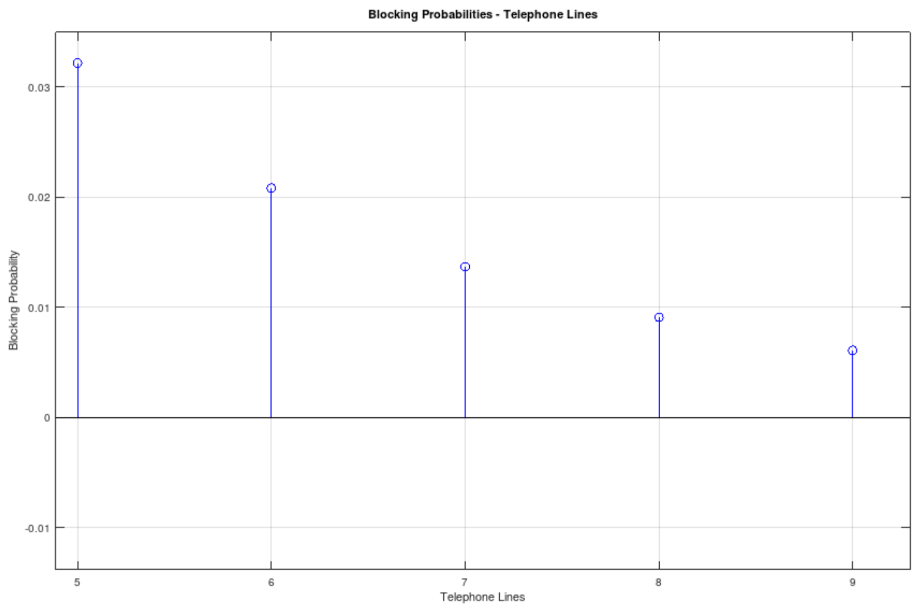
```
clc;
clear all;
close all;
pkg load queueing;

#Ερώτημα 1
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

#Ερώτημα 2
function p = erlangb_iterative (r,c)
  p = 1;
  for i=0:1:c
    p = ((r*p)/((r*p)+i));
  endfor
endfunction

#Ερώτημα 4
Pblocking = zeros(0,200);

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

figure(1);
stem(Pblocking,'b',"linewidth",0.4);
title("Blocking Probabilities - Telephone Lines")
xlabel("Telephone Lines");
ylabel("Blocking Probability");
```

Ερώτημα Γ

Για να βρούμε τον κατάλληλο αριθμό τηλεφωνικών γραμμών ώστε η πιθανότητα απόρριψης τηλεφωνικής κλήσης να είναι μικρότερη από 1%, μελετούμε προσεκτικά το παραπάνω διάγραμμα

Βλέπουμε ότι ο αριθμός αυτός είναι 8 τηλεφωνικές γραμμές

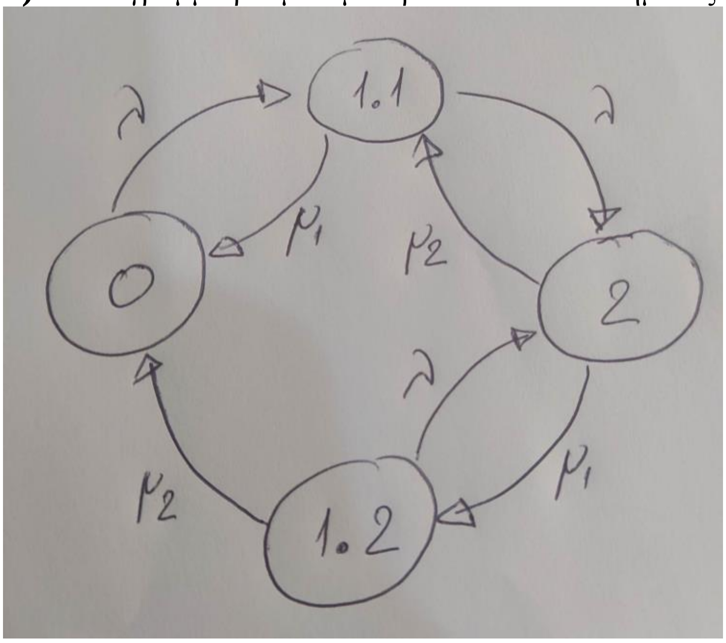
Αυτό φαίνεται και στο παρακάτω διάγραμμα (μεγέθυνση του προηγούμενου)



ΑΣΚΗΣΗ 2:
ΣΥΣΤΗΜΑ ΕΞΥΠΗΡΕΤΗΣΗΣ ΜΕ 2 ΑΝΟΜΟΙΟΥΣ ΕΞΥΠΗΡΕΤΗΤΕΣ

Θεωρούμε ένα σύστημα εξυπηρέτησης που αποτελείται από δύο εξυπηρετητές 1 και 2 χωρίς δυνατότητα αναμονής πελατών. Όταν και οι δύο εξυπηρετητές είναι διαθέσιμοι, ένας πελάτης δρομολογείται πάντα στον εξυπηρετητή 1, αρα ρ=1. Σε περίπτωση που ο εξυπηρετητής 1 δεν είναι διαθέσιμος, ένας νέος πελάτης δρομολογείται στον εξυπηρετητή 2. Σε περίπτωση που και οι δύο εξυπηρετητές δεν είναι διαθέσιμοι, ένας νέος πελάτης απορρίπτεται χωρίς επανάληψη προσπάθειας εξυπηρέτησης. Οι αφίξεις πελατών στο σύστημα ακολουθούν την κατανομή Poisson με μέσο ρυθμό λ = 1 πελάτες/sec. Οι εξυπηρετήσεις είναι εκθετικά κατανεμημένες με μέσους χρόνους εξυπηρέτησης 1/μ1 = 1.25 sec και 1/μ2 = 2.5sec αντίστοιχα, δηλαδή ο εξυπηρετητής 2 είναι χαμηλότερων δυνατοτήτων από τον εξυπηρετητή 1.

1) Το διάγραμμα ρυθμών μεταβάσεων του συστήματος στην κατάσταση ισορροπίας είναι:



Ουσιαστικά οι καταστάσεις 1.1 και 1.2 δείχνουν τις καταστάσεις οπου μονο ενας εξυπηρετητής είναι απασχολημένος (ο 1 ή ο 2 αντίστοιχα)

Ερώτημα Α

Εξισώσεις Ισορροπίας:

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

Ωστόσο ξέρουμε ότι: μ1 = 0.8 πελάτες/sec , μ2 = 0.4 πελάτες/sec, λ=1 και ρ=1
Αρα, έπειτα από προσεγγισεις

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

Ερώτημα Β

Επίσης ισχύει ότι Pblocking = P2 = 0.341

Ερώτημα Γ

Ο μέσος αριθμός πελατών στο σύστημα είναι:

$$\sum_{k=0}^{2} k + P(k) \;=\; 0 \cdot P_0 + 1 \cdot P_{11} + 1 \cdot P_{12} \;+\; 2 \cdot P_2 \;=\; 1.09$$

2) Θα βρουμε τα ζητούμενα του προηγούμενου ερωτήματος με τη μέθοδο της προσομοίωσης συστημάτων αναμονής. Χρησιμοποιούμε το αρχείο κώδικα demo4.m που μας έχει δοθεί.

Ερώτημα Α

Συμπληρώνουμε τα κενά στα thresholds του προγράμματος:

- threshold_1a = lambda/(lambda+m1)

Εχουμε είτε αναχώρηση και να μεταβούμε στην κατάσταση 0 με ρυθμό m1 είτε άφιξη και να μεταβούμε στην κατάσταση 2 με ρυθμό λ.

- threshold_1b = lambda/(lambda+m2)

Εχουμε είτε αναχώρηση και να μεταβούμε στην κατάσταση 0 με ρυθμό m2 είτε άφιξη και να μεταβούμε στην κατάσταση 2 με ρυθμό λ.

- threshold_2_first = lambda/(lambda+m1+m2)

Μπορούμε να έχουμε άφιξη ενός πελάτη και άρα απόρριψη του με ρυθμό λ είτε αναχώρηση με ρυθμό m2

- threshold_2_second = (m1+lambda)/(lambda+m1+m2)

Μπορούμε να έχουμε αναχώρηση με ρυθμό m1 είτε αναχώρηση με ρυθμό m2

Ερώτημα Β

Τα κριτήρια σύγκλισης της προσωμοίωσης είναι εάν δυο διαδοχικοί μέσοι αριθμοί πελατών διαφέρουν λιγότερο από 0.001%. Αυτος ο ελεγχος γινεται κάθε 1000 iterations (δηλαδή κάθε 1000 αφίξεις ή αναχωρήσεις)

Ερώτημα Γ

Υπολογίζουμε με βάση το πρόγραμμα τις εργοδικές πιθανότητες καθώς επίσης και τον μεσο αριθμό πελατών

```
Command Window
0.2496
0.2163
0.1915
0.3426
mean_clients = 1.0930
>> |
```

Παρατηρούμε ότι οι αριθμοί της προσομοίωσης είναι ίσοι με τους παραπάνω που υπολογίσαμε με μικρές αποκλίσεις

Ο κώδικας που χρησιμοποιήσαμε για το ερώτημα αυτό είναι ο εξής:

```
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 = (m1+lambda)/(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 #αν εχω αφιξη
      current_state = 3;
      arrivals(2) = arrivals(2) + 1;
      total_arrivals = total_arrivals + 1;
    else #αν εχω αναχώρηση
      current_state = 0;
    endif
  elseif current_state == 2
    if random_number < threshold_1b # αν εχω αφιξη
      current_state = 3;
      arrivals(3) = arrivals(3) + 1;
      total_arrivals = total_arrivals + 1;
    else # αν εχω αναχωρηση
      current_state = 0;
    endif
  else # μια απο τις 3 περιπτώσεις
      if random_number < threshold_2_first
        arrivals(4) = arrivals(4) + 1; # αν εχω αφιξη και απορριψη
        total_arrivals = total_arrivals + 1;
      elseif random_number < threshold_2_second # αν εχω αναχώρηση με ρυθ
        current_state = 2;
      else # αν εχω αναχώρηση με ρυθμό m2
        current_state = 1;
      endif
   endif

endwhile

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