



ΣΥΣΤΗΜΑΤΑ ΑΝΑΜΟΝΗΣ

6η Ομάδα Ασκήσεων

Νικόλαος Δημητριάδης

03114016

HMMY 80

1 Μηχανισμός ελέγχου ροής παραθύρου

ερώτημα 1

Για τις τιμές των παραμέτρων της εκφώνησης, τα ζητούμενα διαγράμματα είναι τα ακόλουθα:

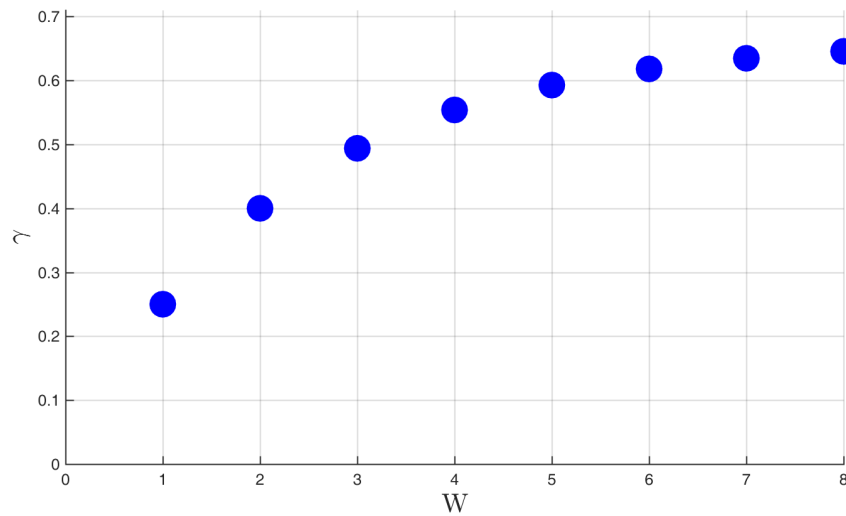


Figure 1: ρυθμαπόδοση (throughput) του συστήματος ως συνάρτηση του αριθμού των πακέτων στο σύστημα (μέγεθος παραθύρου W)

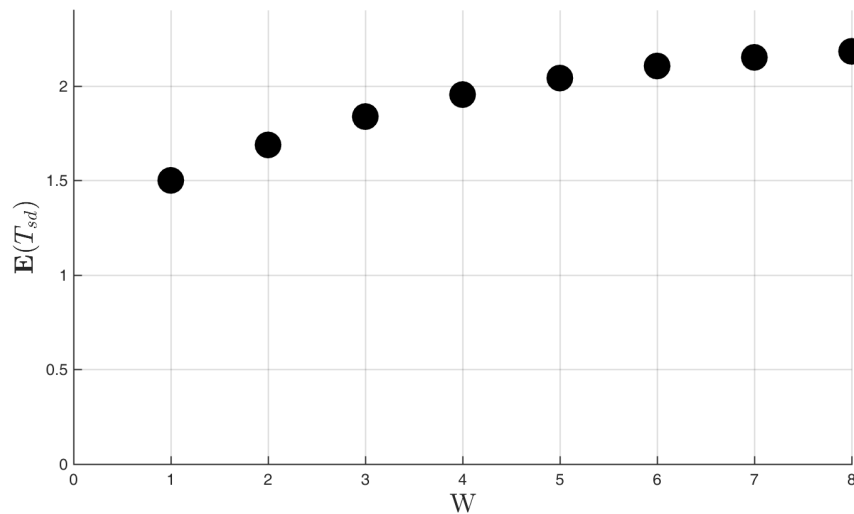


Figure 2: Μέσος χρόνος καθυστέρησης του συστήματος από το S μέχρι το D ως συνάρτηση του αριθμού των πακέτων στο σύστημα

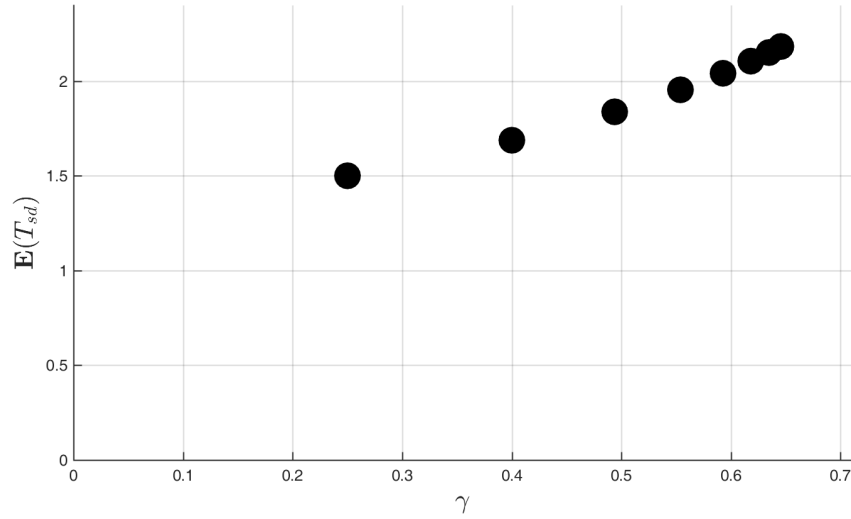


Figure 3: Μέσος χρόνος καθυστέρησης του συστήματος από το S μέχρι το D ως συνάρτηση της ρυθμιαπόδοσης του συστήματος

Παρατηρούμε ότι με την αύξηση του παραθύρου W , ο μέσος χρόνος καθυστέρησης του συστήματος από το S μέχρι το D και η ρυθμιαπόδοση γ του συστήματος τείνουν να σταθεροποιηθούν.

ερώτημα 2

Σε αυτό το σημείο θεωρούμε ότι οι παράμετροι μεταβάλλονται σύμφωνα με το k με τον ακόλουθο τρόπο:

$$\begin{bmatrix} k\lambda \\ k\mu_1 \\ k\mu_2 \\ k\mu_3 \\ k\mu_r \end{bmatrix} \quad k = 1, 2, 3, 4, 5 \quad (1)$$

Επιπλέον, οι τιμές των παραμέτρων διέπονται από την ακόλουθη σχέση:

$$\lambda X_0 = \mu_1 X_1 = \mu_2 X_2 = \mu_3 X_3 = \mu_r X_r \quad \text{με } X_0 = 1 \quad (2)$$

Από τη σχέση αυτή συμπεραίνουμε ότι οι παράμετροι X_i , με $i = 1, 2, 3, 4, r$ αυξάνονται αναλογικά. Με άλλα λόγια, για οποιοδήποτε k παραμένουν σταθεροί. Από τον ορισμό της σταθεράς κατακερματισμού $G(N)$ βρίσκουμε ότι αυτή εξαρτάται μόνο από τις παραμέτρους X_i και, συνεπώς, παραμένει αμετάβλητη. Αυτό σημαίνει ότι:

- ο βαθμός χρησιμοποίησης U παραμένει σταθερός $\forall k$, καθώς εξαρτάται από X_i και $G(N)$,
- ο μέσος αριθμός πελατών Q παραμένει σταθερός $\forall k$, καθώς εξαρτάται από X_i και $G(N)$,
- η ρυθμιαπόδοση γ μεταβάλλεται, αφού εξαρτάται από τους ρυθμούς λ, μ_i . Αυτό φαίνεται και στο αντίστοιχο διάγραμμα παρακάτω.

- ο μέσος χρόνος καθυστέρησης $\mathbb{E}[T]$ μεταβάλλεται, αφού εξαρτάται από τους ρυθμούς λ, μ_i .

Τα αποτελέσματα παρουσιάζονται σε μορφή διαγράμματος για τα ζητούμενα μεγέθη. Επιλέγεται η παρουσίαση των ίδιων διαγραμμάτων με το προηγούμενο ερώτημα:

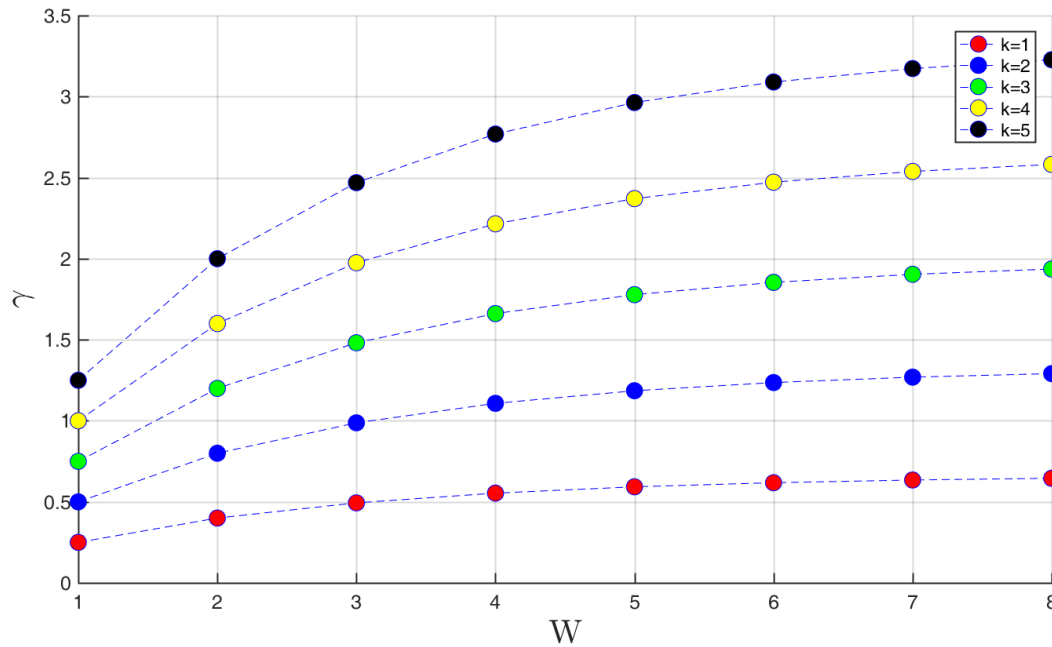


Figure 4: ρυθμαπόδοση (throughput) του συστήματος ως συνάρτηση του αριθμού των πακέτων στο σύστημα (μέγεθος παραθύρου W)

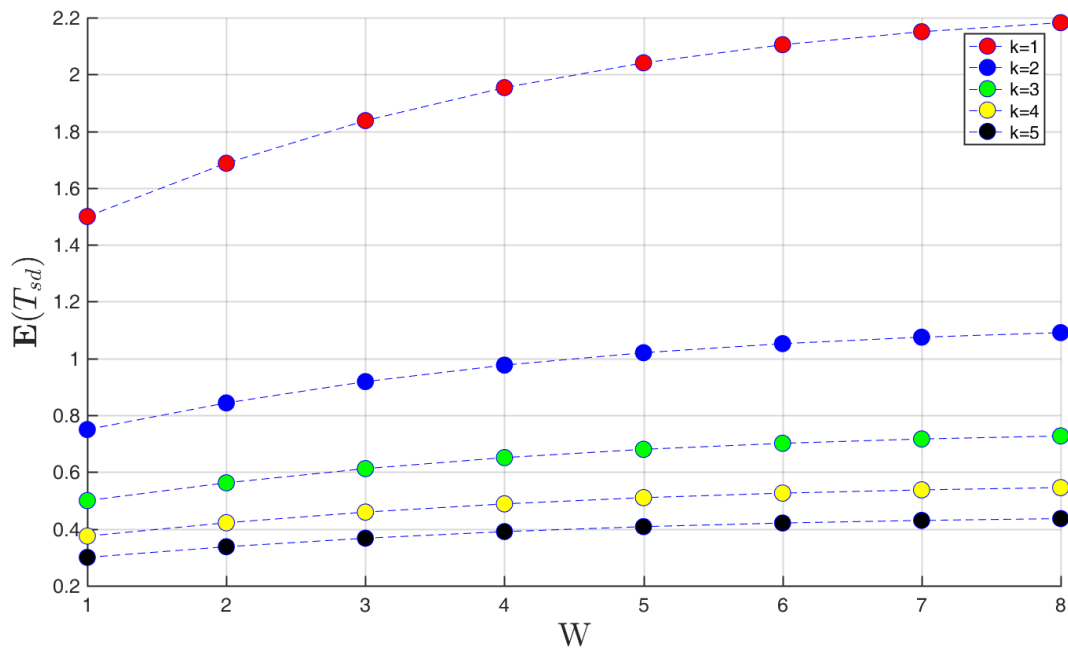


Figure 5: Μέσος χρόνος καθυστέρησης του συστήματος από το S μέχρι το D ως συνάρτηση του αριθμού των πακέτων στο σύστημα

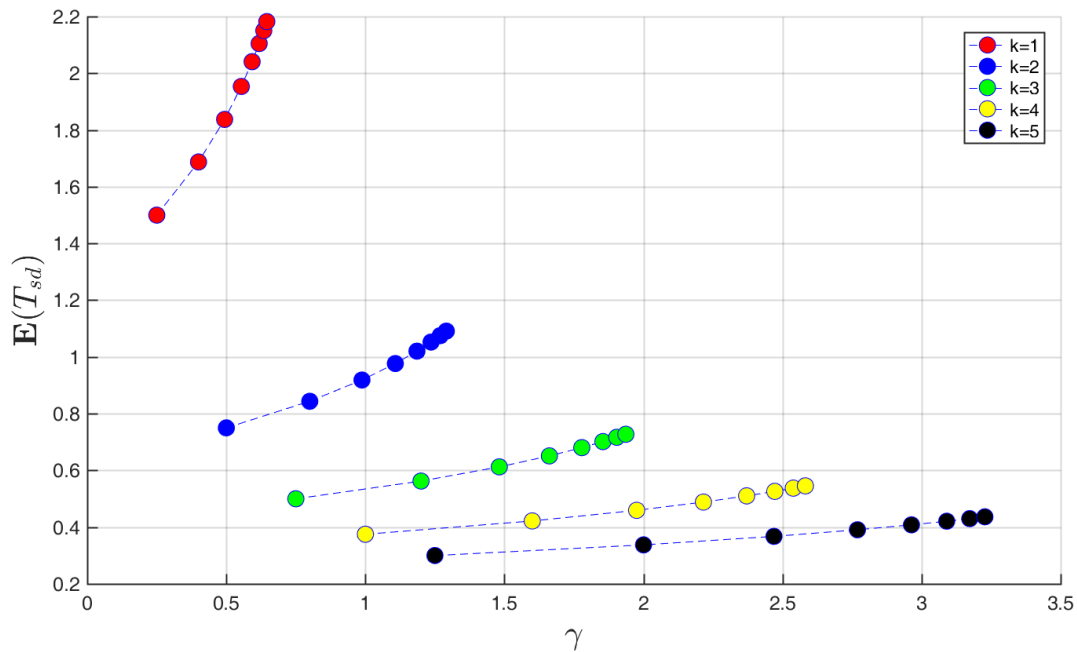


Figure 6: Μέσος χρόνος καθυστέρησης του συστήματος από το S μέχρι το D ως συνάρτηση της ρυθμιαπόδοσης του συστήματος

Πράγματι, ρυθμιαπόδοση γ μεταβάλλεται για τις διάφορες τιμές του k

2 Ο αλγόριθμος του Buzen

ερώτημα 1

Από το κλειστό δίκτυο της εκφώνησης προκύπτουν οι ακόλουθες εξισώσεις

$$\mu_1 X_1 = (1 - p)\mu_1 X_1 + \mu_2 X_2 \quad (3)$$

$$\mu_2 X_2 = p\mu_1 X_1 \quad (4)$$

$$(4) \xrightarrow{X_1=1} X_2 = \frac{p\mu_1 \cdot 1}{\mu_2} = 0.6 \quad (5)$$

Άρα

$$\boxed{X_1 = 1} \text{ και } \boxed{X_2 = 0.6} \quad (6)$$

ερώτημα 2

Η συνάρτηση buzen παρουσιάζεται παρακάτω:

Listing 1: buzen

```

1 function G_N = buzen( N, M , X)
2
3     for n = 1:N
4         g(n,1) = 1;
5     end
6
7     for m = 1:M
8         g(1,m) = 1;
9     end
10
11    for m = 2:M
12        for n = 2:N
13            g(n,m) = g(n,m-1) + X(m)*g(n-1,m);
14        end
15    end
16
17    G_N = g(N,M);
18
19 end

```

ερώτημα 3

Σύμφωνα με τα προηγούμενα προκύπτουν τα ακόλουθα διαγράμματα:

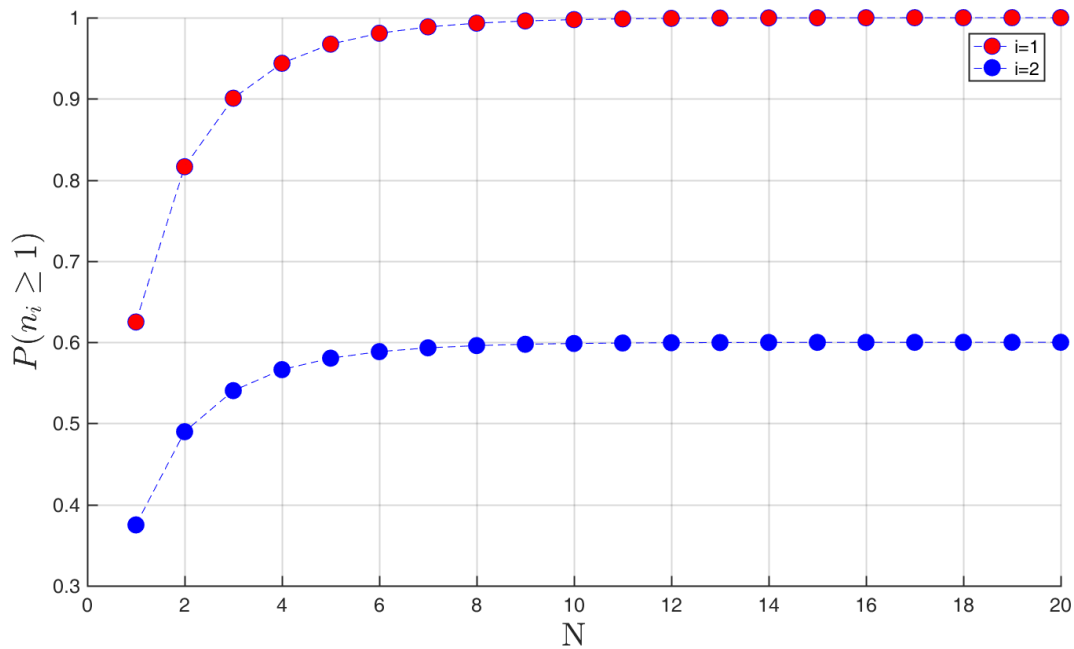


Figure 7: Βαθμός χρησιμοποίησης (utilization) των δύο ουρών ως συνάρτηση του αριθμού πελατών N σε κοινό διάγραμμα αξόνων

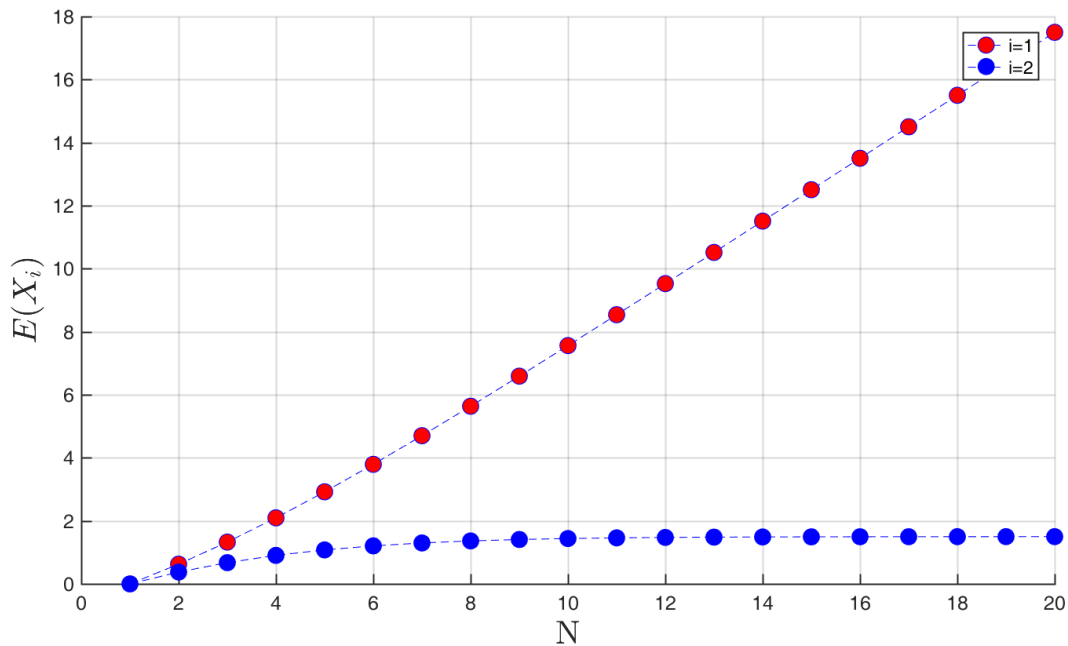


Figure 8: Το μέσο αριθμό πελατών στις δύο ουρές ως συνάρτηση του αριθμού πελατών N σε κοινό διάγραμμα αξόνων

Παρατηρούμε ότι με την αύξηση των πελατών στο κλειστό σύστημα, οι βαθμοί χρησιμοποίησης του συστήματος τείνουν να λάβουν τις τιμές των παραμέτρων X_i . Τέλος, εφόσον έχουμε

κλειστό σύστημα, το πλήθος των πελατών παραμένει σταθερό. Αυτό είναι εμφανές και στο παραπάνω σχήμα.

3 Προσομοίωση σε κλειστό δίκτυο εκθετικών ουρών αναμονής

Ως διάνυσμα κατάστασης επιλέγεται το $\mathbf{n} = (n_1, n_2, n_3)$.

Το διάγραμμα ρυθμού μεταβάσεων είναι:

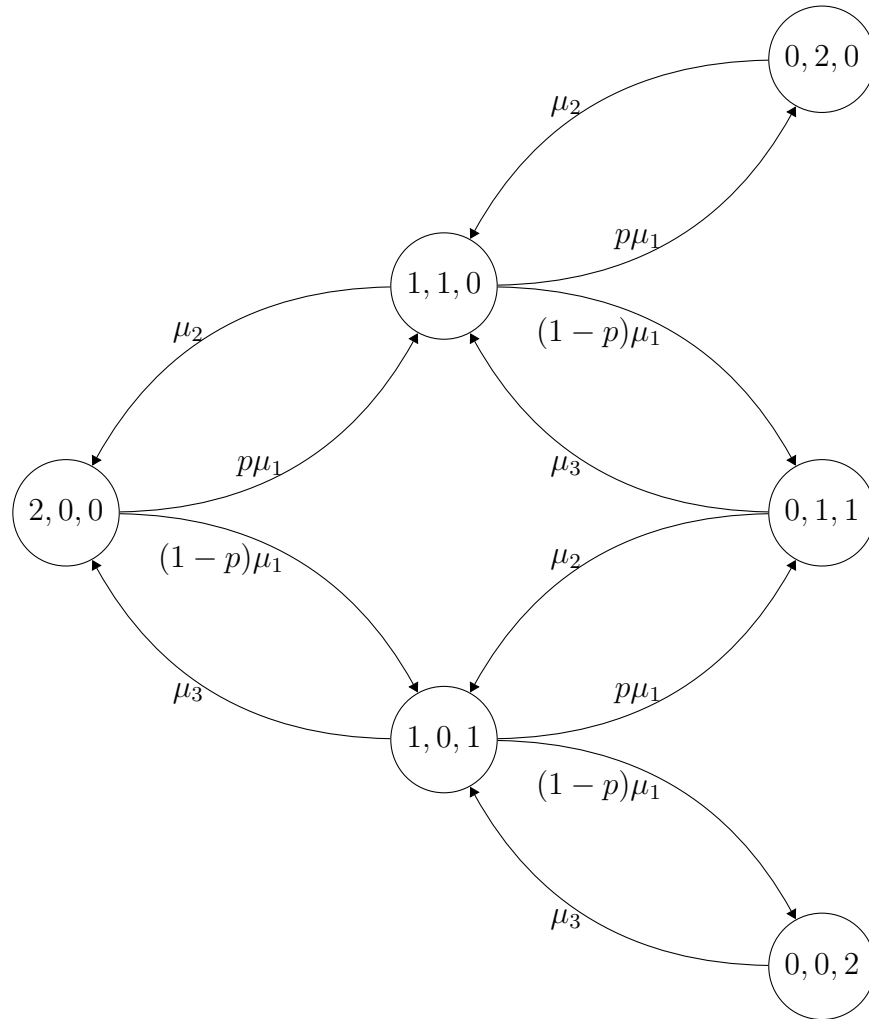


Figure 9: Διάγραμμα ρυθμού μεταβάσεων συστήματος

ερώτημα 1

Μέσω της προσομοίωσης βρίσκουμε τις ακόλουθες εργοδικές πιθανότητες:

$$\begin{bmatrix} P_{(2,0,0)} \\ P_{(1,0,1)} \\ P_{(1,1,0)} \\ P_{(0,1,1)} \\ P_{(0,2,0)} \\ P_{(0,0,2)} \end{bmatrix} = \begin{bmatrix} 0.5520 \\ 0.1664 \\ 0.1475 \\ 0.0448 \\ 0.0389 \\ 0.0504 \end{bmatrix} \quad (7)$$

ερώτημα 2

Ο μέσος αριθμός πελατών σε κάθε ουρά του συστήματος, όπως αυτός εξελίσσεται κατά της διάρκεια της προσομοίωσης παρουσιάζεται παρακάτω:

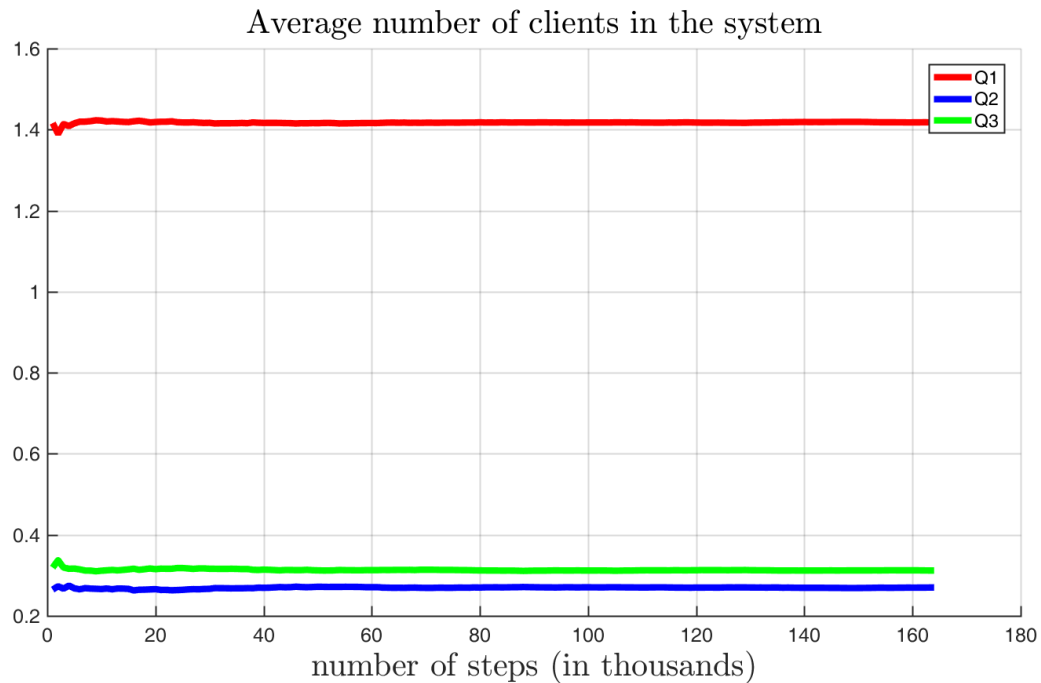


Figure 10: Μέσος αριθμός πελατών σε κάθε ουρά του συστήματος, όπως αυτός εξελίσσεται κατά της διάρκεια της προσομοίωσης

Παρατηρούμε ότι το άθροισμα των τριών μέσων όρων όταν το σύστημά έχει φτάσει στη σύγκλιση ισούται με $N = 2$.

4 Appendix

Listing 2: OCTAVE initialization

```
1 W = 8;
2 S = 1./[1, 2, 2, 2, 2/3];
3 P = [ 0 1 0 0 0 ; 0 0 1 0 0 ; 0 0 0 1 0 ; 0 0 0 0 1 ; 1 0 0 0 0 ];
4 V = qncsvisits(P);
5
6
7 for i = 1:W
8     % X = throughput
9     [U R Q X] = qnclosed( i , S , V );
10    gamma(i) = X(1);
11    E_T_sd(i) = 0;
12    for j = 2:4
13        E_T_sd(i) = E_T_sd(i) + R(j) ;
14    endfor
15    % disp(E_T_sd)
16 endfor
17
18 % PLOTS
19
20
21
22 %% erwthma 2
23
24
25 for i = 1:W
26     for k = 1:5
27         S_now = S / k;
28         % X = throughput
29         [U R Q X] = qnclosed( i , S_now , V );
30         U_analytics(i,k,:) = U;
31         Q_analytics(i,k,:) = Q;
32         gamma2(i,k) = X(1);
33         E_T_sd2(i,k) = 0;
34         for j = 2:4
35             E_T_sd2(i,k) = E_T_sd2(i,k) + R(j) ;
36         endfor
37     endfor
38 endfor
```

Listing 3: part 1

```
1 close all;
```

```
2 clear all;
3 clc;
4 %%
5 % TAKEN gamma, E_T_sd from Octave
6 gamma = [ 0.25000, 0.40000, 0.49383, 0.55385, 0.59271, 0.61803,
           0.63461, 0.64549 ];
7 E_T_sd = [ 1.5000, 1.6875, 1.8375, 1.9537, 2.0410, 2.1049, 2.1504,
            2.1823 ];
8
9 fontsize = 12;
10 lineWidth = 1;
11 plotLineWidth = 4;
12 width=1024;
13 height=568;
14
15 fig1 = figure();
16 fig1.Color = 'w';
17 set(gcf,'units','pixels','position',[0,0,width,height]) ;
18 hold on
19 plot(gamma,'bo','MarkerSize',20,'MarkerFaceColor','b');
20 temp = gca;
21 temp.Color = 'w';
22 temp.LineWidth = lineWidth;
23 temp.GridColor = 'k';
24 % temp.GridAlpha = 0.5;
25 temp.FontSize = fontsize;
26 xlim([0 8]);
27 ylim([0 1.1*max(gamma)]);
28 hold off
29
30
31 ylabel(['$\gamma$'],'FontSize',20,'Interpreter','latex');
32 xlabel(['W'],'FontSize',20,'Interpreter','latex');
33 grid on;
34 saveas(gcf,'figure1.png');
35
36 %%
37 fig2 = figure();
38 fig2.Color = 'w';
39 set(gcf,'units','pixels','position',[0,0,width,height]) ;
40 hold on
41 plot(E_T_sd,'ko','MarkerSize',20,'MarkerFaceColor','k');
42 temp = gca;
43 temp.Color = 'w';
44 temp.LineWidth = lineWidth;
```

```
45 temp.GridColor = 'k';
46 % temp.GridAlpha = 0.5;
47 temp.FontSize = fontsize;
48 xlim([0 8]);
49 ylim([0 1.1*max(E_T_sd)]);
50 hold off
51
52
53 ylabel(['$\mathbf{E}(T_{sd})$'], 'FontSize',20, 'Interpreter', 'latex');
54 xlabel(['W'], 'FontSize',20, 'Interpreter', 'latex');
55 grid on;
56 saveas(gcf, 'figure2.png');
57
58 %%
59 fig3 = figure();
60 fig3.Color = 'w';
61 set(gcf, 'units', 'pixels', 'position', [0,0,width,height]) ;
62 hold on
63 plot(gamma, E_T_sd, 'ko', 'MarkerSize',20, 'MarkerFaceColor', 'k');
64 temp = gca;
65 temp.Color = 'w';
66 temp.LineWidth = lineWidth;
67 temp.GridColor = 'k';
68 % temp.GridAlpha = 0.5;
69 temp.FontSize = fontsize;
70 xlim([0 1.1*max(gamma)]);
71 ylim([0 1.1*max(E_T_sd)]);
72 hold off
73
74
75 ylabel(['$\mathbf{E}(T_{sd})$'], 'FontSize',20, 'Interpreter', 'latex');
76 xlabel(['$\gamma$'], 'FontSize',20, 'Interpreter', 'latex');
77 grid on;
78 saveas(gcf, 'figure3.png');
```

Listing 4: part 2

```
1 close all;
2 clear all;
3 clc;
4 %%
5 M = 2;
6 X = [1,0.6];
7 for N = 1:21
8 %     arithmhths = buzen(N,M,X);
```

```
9      G_N(N) = buzen(N,M,X);      % N+1 giati h arithmhsh tw n grammwn ston
      Buzen xekinaei apo to 0
10 end
11
12 for N = 1:20
13     for i = 1:M
14         E_n(i,N) = 0;
15         for k = 1:(N-1)
16             E_n(i,N) = E_n(i,N) + X(i)^k * G_N(N-k) / G_N(N);
17         end
18         P(i,N) = X(i) * G_N(N) / G_N(N+1);
19     end
20     pop(N) = E_n(1,N)+E_n(2,N);
21 end
22
23 %%
24 fontsize = 12;
25 lineWidth = 1;
26 plotLineWidth = 4;
27 width=1024;
28 height=568;
29 colors = 'rbgk';
30 fig1 = figure();
31 fig1.Color = 'w';
32 set(gcf,'units','pixels','position',[0,0,width,height]) ;
33 hold on
34 for k = 1:M
35     plot(P(k,:), 'b—o', 'MarkerSize',10, 'MarkerFaceColor',colors(k));
36 end
37 temp = gca;
38 temp.Color = 'w';
39 temp.LineWidth = lineWidth;
40 temp.GridColor = 'k';
41 % temp.GridAlpha = 0.5;
42 temp.FontSize = fontsize;
43 hold off
44
45
46 ylabel(['$P(n_{i} \geq 1)$'], 'FontSize',20, 'Interpreter', 'latex');
47 xlabel(['N'], 'FontSize',20, 'Interpreter', 'latex');
48 legend('i=1', 'i=2');
49 grid on;
50 saveas(gcf, 'figure_2a.png');
51
52
```

```
53 %%
54 colors = 'rbgyk';
55 fig1 = figure();
56 fig1.Color = 'w';
57 set(gcf, 'units', 'pixels', 'position', [0,0,width,height]) ;
58 hold on
59 for k = 1:M
60     plot(E_n(k,:), 'b—o', 'MarkerSize', 10, 'MarkerFaceColor', colors(k));
61 end
62 temp = gca;
63 temp.Color = 'w';
64 temp.LineWidth = lineWidth;
65 temp.GridColor = 'k';
66 % temp.GridAlpha = 0.5;
67 temp.FontSize = fontsize;
68 hold off
69
70
71 ylabel(['$E(X_i)$'], 'FontSize', 20, 'Interpreter', 'latex');
72 xlabel(['N'], 'FontSize', 20, 'Interpreter', 'latex');
73 legend('i=1', 'i=2');
74 grid on;
75 saveas(gcf, 'figure_2b.png');
```

Listing 5: part 3

```
1 % Example: two queues in a row. The output of the first queue is the input of
  the second.
2 % The output of the second queue is the input of the first queue.
3
4 clc;
5 clear all;
6 close all;
7 N = 3;
8 mu1 = 2; % queue 1
9 mu2 = 3; % queue 2
10 mu3 = 4; % queue 3
11 p = 0.4;
12 rng(1)
13 arrivals(200) = 0;
14 arrivals(110) = 0;
15 arrivals(011) = 0;
16 arrivals(020) = 0;
17 arrivals(002) = 0;
18 arrivals(101) = 0;
```

```
19 total_arrivals = 0;
20 %%
21 % threshold definition
22 threshold = mu1/(mu1 + mu2);
23 % system starts at state 3
24 current_state = 200;
25 % count the time steps of the simulation
26 steps = 0;
27
28 previous_mean1 = 0;
29 previous_mean2 = 0;
30 previous_mean3 = 0;
31 % times checked for convergence
32 times = 0;
33
34 while steps < 300000
35     steps = steps + 1;
36     % every 1000 steps check for convergence
37     if mod(steps,1000) == 0
38         times = times + 1;
39
40
41     %
42         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
43
44     % total time in every state
45     T200 = 1/mu1 * arrivals(200);
46     T002 = 1/mu3 * arrivals(002);
47     T020 = 1/mu2 * arrivals(020);
48     T101 = 1/(mu1 + mu3) * arrivals(101);
49     T110 = 1/(mu1 + mu2) * arrivals(110);
50     T011 = 1/(mu2 + mu3) * arrivals(011);
51
52     % total time in all states
53     total_time = T200 + T002 + T020 + T101 + T110 + T011;
54     % Probability of every state
55     P(200) = T200/total_time;
56     P(002) = T002/total_time;
57     P(020) = T020/total_time;
58     P(110) = T110/total_time;
59     P(101) = T101/total_time;
60     P(011) = T011/total_time;
61
62     % mean number of clients in queues 1 and 2
63     current_mean1 = P(101) + P(110) + 2*P(200);
```



```
62     current_mean2 = P(011) + P(110) + 2*P(020);
63     current_mean3 = P(101) + P(011) + 2*P(002);
64
65     clients_1(times) = current_mean1;
66     clients_2(times) = current_mean2;
67     clients_3(times) = current_mean3;
68
69     % check both queues for convergence
70     if abs(current_mean1 - previous_mean1) < 0.00001 && abs(current_mean2 -
        previous_mean2) && abs(current_mean3 - previous_mean3) < 0.00001
71         break;
72     end
73
74     previous_mean1 = current_mean1;
75     previous_mean2 = current_mean2;
76     previous_mean3 = current_mean3;
77
78 end
79
80 arrivals(current_state) = arrivals(current_state) + 1;
81 total_arrivals = total_arrivals + 1;
82
83 % get a random number from uniform distribution
84 random_number = rand(1);
85 if current_state == 200
86     if (random_number < p)
87         current_state = 110;
88     else
89         current_state = 101;
90     end
91 elseif current_state == 110
92     if random_number < mu2 / (mu1 + mu2)
93         current_state = 200;
94     else
95         if random_number < (mu2 + p * mu1) / (mu1 + mu2)
96             current_state = 020;
97         else
98             current_state = 011;
99         end
100     end
101 elseif current_state == 101
102     if random_number < mu3 / (mu1 + mu3)
103         current_state = 200;
104     else
105         if random_number < (mu3 + p * mu1) / (mu1 + mu3)
```

```
106         current_state = 011;
107     else
108         current_state = 002;
109     end
110 end
111 elseif current_state == 020
112     current_state = 110;
113 elseif current_state == 002
114     current_state = 101;
115 else
116     if random_number < mu2 / (mu2 + mu3)
117         current_state = 101;
118     else
119         current_state = 110;
120     end
121 end
122
123 end
124
125
126 pop = [P(200) P(101) P(110) P(011) P(020) P(002) ];
127 P(200)
128 P(020)
129 P(002)
130 P(110)
131 P(101)
132 P(011)
133 clients_1(times)
134 clients_2(times)
135 clients_3(times)
136
137
138
139
140 %%
141 fontsize = 12;
142 lineWidth = 1;
143 plotLineWidth = 4;
144 width=1024;
145 height=568;
146 colors = 'rbgyk';
147 fig1 = figure();
148 fig1.Color = 'w';
149 set(gcf,'units','pixels','position',[0,0,width,height]) ;
150 hold on
```

```
151 plot(clients_1,'r','linewidth',plotLineWidth);
152 plot(clients_2,'b','linewidth',plotLineWidth);
153 plot(clients_3,'g','linewidth',plotLineWidth);
154 temp = gca;
155 temp.Color = 'w';
156 temp.LineWidth = lineWidth;
157 temp.GridColor = 'k';
158 % temp.GridAlpha = 0.5;
159 temp.FontSize = fontsize;
160 hold off
161
162 legend('Q1','Q2','Q3');
163 title(['Average number of clients in the system'],'FontSize',20,'Interpreter'
    , 'latex');
164 xlabel(['number of steps (in thousands)'],'FontSize',20, 'Interpreter','latex
    ');
165 grid on;
166 saveas(gcf,'figure3_1.png');
167
168
169
170 %%
171
172 for i = 1:times
173     t(i) = clients_1(i)+clients_2(i)+clients_3(i);
174 end
175
176 t(times)
```