Συστήματα Αναμονής
3η εργαστηριακή άσκηση

Νικόλαος Παγώνας, el18175

Προσομοίωση συστήματος M/M/1/10

(1)

Εκτυπώνουμε τις πρώτες 30 μεταβάσεις για λ = {1,5,10} προκειμένου να επιβεβαιώσουμε την ορθή λειτουργία της προσομοίωσης.

Για λ = 1, μ = 5:

```
01. State  0 | Arrival   | 0 Arrivals
02. State  1 | Departure | 0 Arrivals
03. State  0 | Arrival   | 1 Arrivals
04. State  1 | Arrival   | 0 Arrivals
05. State  2 | Departure | 0 Arrivals
06. State  1 | Departure | 1 Arrivals
07. State  0 | Arrival   | 2 Arrivals
08. State  1 | Arrival   | 1 Arrivals
09. State  2 | Departure | 0 Arrivals
10. State  1 | Arrival   | 2 Arrivals
11. State  2 | Departure | 0 Arrivals
12. State  1 | Departure | 3 Arrivals
13. State  0 | Arrival   | 3 Arrivals
14. State  1 | Departure | 3 Arrivals
15. State  0 | Arrival   | 4 Arrivals
16. State  1 | Departure | 3 Arrivals
17. State  0 | Arrival   | 5 Arrivals
18. State  1 | Departure | 3 Arrivals
19. State  0 | Arrival   | 6 Arrivals
20. State  1 | Departure | 3 Arrivals
21. State  0 | Arrival   | 7 Arrivals
22. State  1 | Arrival   | 3 Arrivals
23. State  2 | Departure | 0 Arrivals
24. State  1 | Departure | 4 Arrivals
25. State  0 | Arrival   | 8 Arrivals
26. State  1 | Departure | 4 Arrivals
27. State  0 | Arrival   | 9 Arrivals
28. State  1 | Departure | 4 Arrivals
29. State  0 | Arrival   | 10 Arrivals
30. State  1 | Arrival   | 4 Arrivals
```

Για $\lambda = 5$, $\mu = 5$:

```
01. State  0 | Arrival   | 0 Arrivals
02. State  1 | Departure | 0 Arrivals
03. State  0 | Arrival   | 1 Arrivals
04. State  1 | Arrival   | 0 Arrivals
05. State  2 | Departure | 0 Arrivals
06. State  1 | Departure | 1 Arrivals
07. State  0 | Arrival   | 2 Arrivals
08. State  1 | Arrival   | 1 Arrivals
09. State  2 | Departure | 0 Arrivals
10. State  1 | Arrival   | 2 Arrivals
11. State  2 | Departure | 0 Arrivals
12. State  1 | Arrival   | 3 Arrivals
13. State  2 | Arrival   | 0 Arrivals
14. State  3 | Arrival   | 0 Arrivals
15. State  4 | Arrival   | 0 Arrivals
16. State  5 | Arrival   | 0 Arrivals
17. State  6 | Arrival   | 0 Arrivals
18. State  7 | Arrival   | 0 Arrivals
19. State  8 | Arrival   | 0 Arrivals
20. State  9 | Departure | 0 Arrivals
21. State  8 | Departure | 1 Arrivals
22. State  7 | Arrival   | 1 Arrivals
23. State  8 | Departure | 1 Arrivals
24. State  7 | Arrival   | 2 Arrivals
25. State  8 | Departure | 1 Arrivals
26. State  7 | Arrival   | 3 Arrivals
27. State  8 | Arrival   | 1 Arrivals
28. State  9 | Arrival   | 0 Arrivals
29. State 10 | Arrival   | 0 Arrivals
30. State 10 | Arrival   | 1 Arrivals
```

Για $\lambda = 10$, $\mu = 5$:

```
01. State  0 | Arrival   | 0 Arrivals
02. State  1 | Departure | 0 Arrivals
03. State  0 | Arrival   | 1 Arrivals
04. State  1 | Arrival   | 0 Arrivals
05. State  2 | Departure | 0 Arrivals
06. State  1 | Departure | 1 Arrivals
07. State  0 | Arrival   | 2 Arrivals
08. State  1 | Arrival   | 1 Arrivals
09. State  2 | Departure | 0 Arrivals
10. State  1 | Arrival   | 2 Arrivals
11. State  2 | Departure | 0 Arrivals
12. State  1 | Arrival   | 3 Arrivals
13. State  2 | Arrival   | 0 Arrivals
14. State  3 | Arrival   | 0 Arrivals
15. State  4 | Arrival   | 0 Arrivals
16. State  5 | Arrival   | 0 Arrivals
17. State  6 | Arrival   | 0 Arrivals
18. State  7 | Arrival   | 0 Arrivals
19. State  8 | Arrival   | 0 Arrivals
20. State  9 | Departure | 0 Arrivals
21. State  8 | Departure | 1 Arrivals
22. State  7 | Arrival   | 1 Arrivals
23. State  8 | Departure | 1 Arrivals
24. State  7 | Arrival   | 2 Arrivals
25. State  8 | Departure | 1 Arrivals
26. State  7 | Arrival   | 3 Arrivals
27. State  8 | Arrival   | 1 Arrivals
28. State  9 | Arrival   | 0 Arrivals
29. State 10 | Arrival   | 0 Arrivals
30. State 10 | Arrival   | 1 Arrivals
```

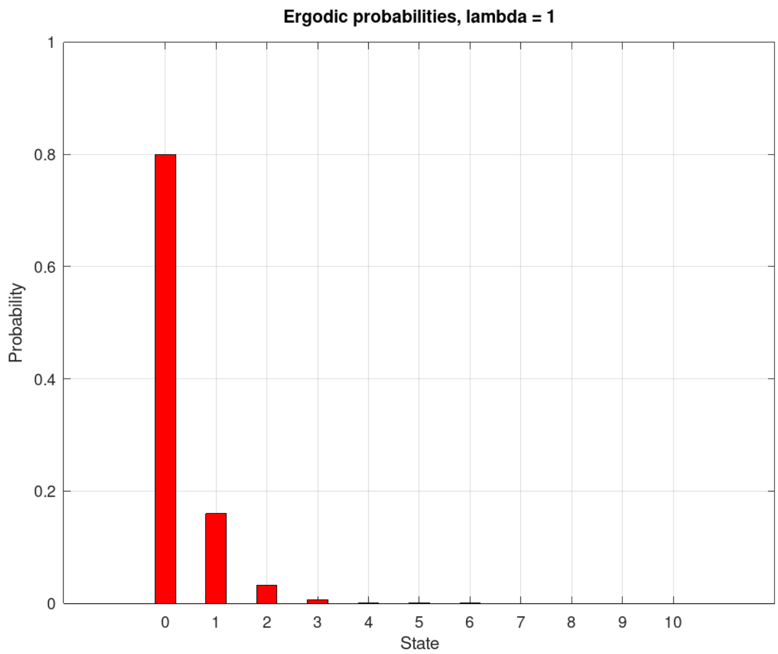
(2)

Χρησιμοποιούμε την προσομοίωση που φτιάξαμε για να παρακολουθήσουμε τη σύγκλιση των ζητούμενων μεγεθών. Η σύγκλιση είναι εγγυημένη καθώς πρόκειται για ουρά πεπερασμένης χωρητικότητας. Σημειώνεται ότι οι οριζόντιες μαύρες γραμμές αποτελούν την τιμή-στόχο της σύγκλισης και προέκυψαν μέσω της εντολής qsmmmk() του πακέτου queueing, η χρήση της οποίας έγινε καθαρά και μόνο για επαλήθευση των αποτελεσμάτων.

Για $\lambda = 1$, $\mu = 5$:

Εργοδικές πιθανότητες των καταστάσεων του συστήματος:

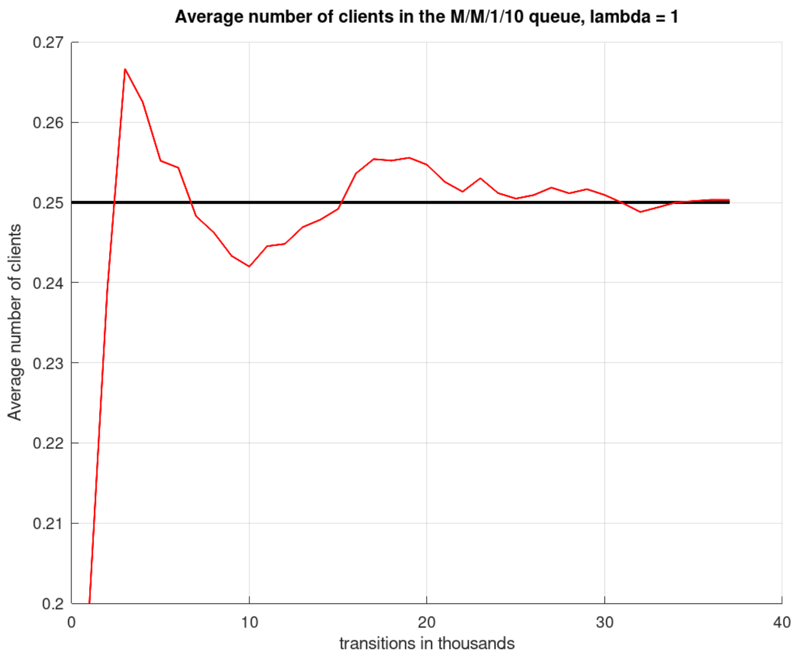
```
P(0) = 80.021622%
P(1) = 15.956757%
P(2) = 3.221622%
P(3) = 0.621622%
P(4) = 0.129730%
P(5) = 0.043243%
P(6) = 0.005405%
P(7) = 0.000000%
P(8) = 0.000000%
P(9) = 0.000000%
P(10) = 0.000000%
```



Πιθανότητα απόρριψης πελάτη από το σύστημα:

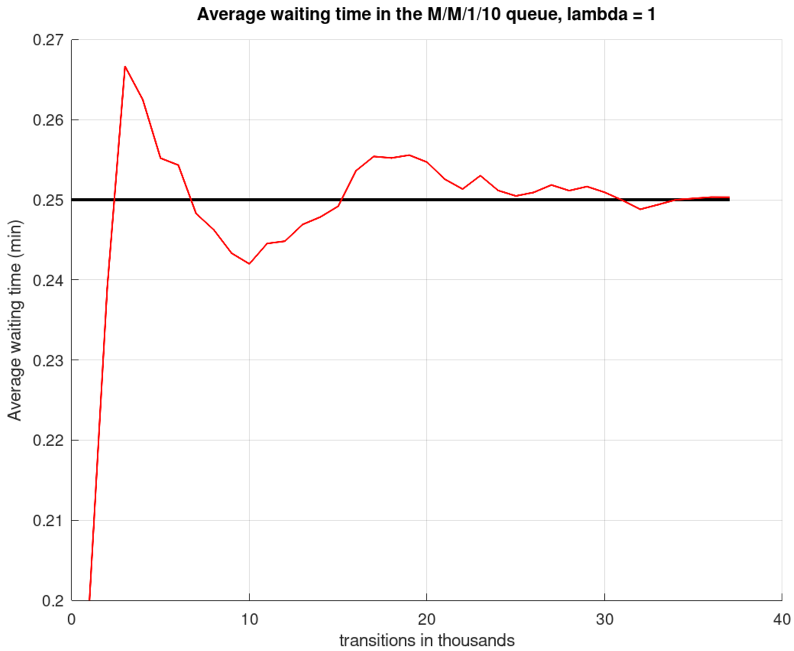
`P[blocking] = P[10] = 0.000000%`

Εξέλιξη του μέσου αριθμού πελατών:



Εξέλιξη του μέσου χρόνου καθυστέρησης:

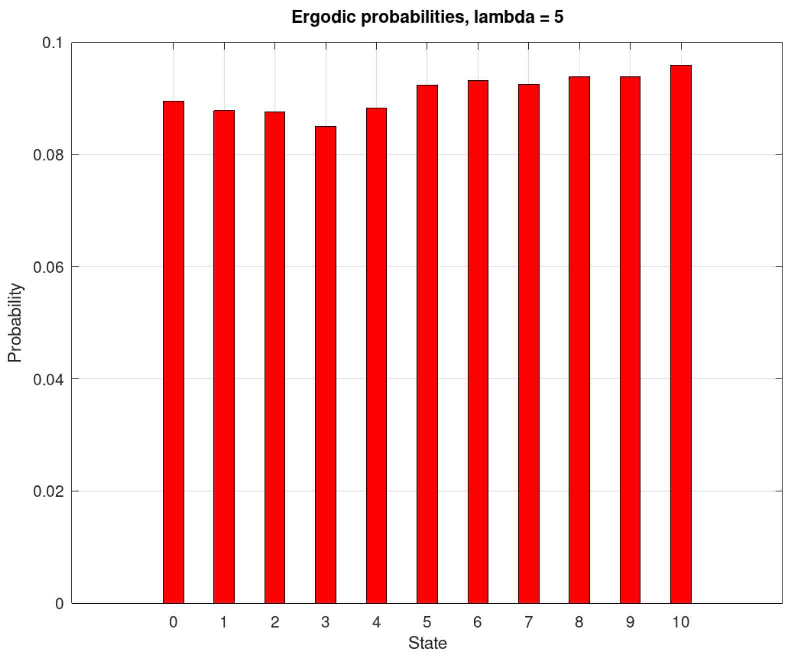
Από τον νόμο του Little: $E[T] = \dfrac{E[n(t)]}{\lambda(1 - P[\text{blocking}])}$



Για $\lambda = 5$, $\mu = 5$:

Εργοδικές πιθανότητες των καταστάσεων του συστήματος:

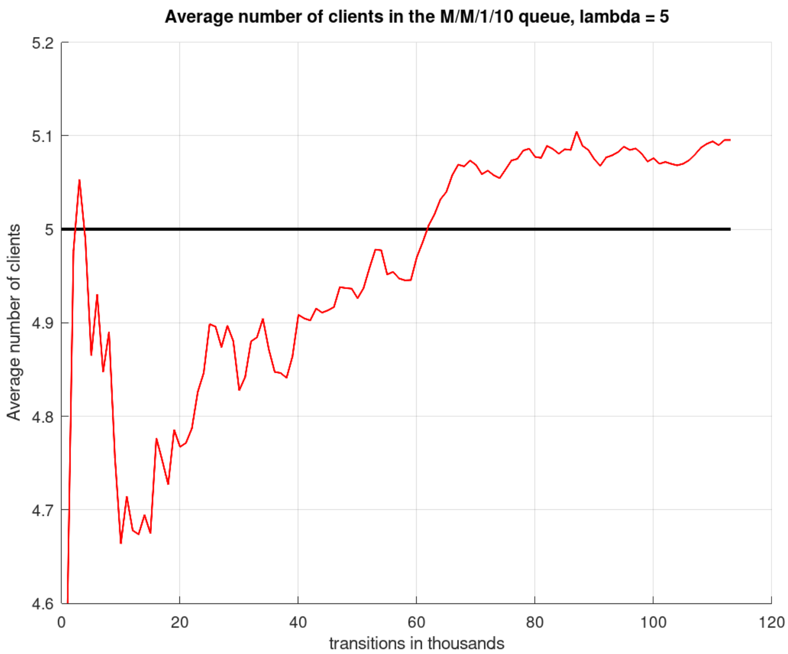
```
P(0) = 8.949213%
P(1) = 8.782395%
P(2) = 8.755434%
P(3) = 8.502679%
P(4) = 8.824521%
P(5) = 9.242409%
P(6) = 9.321605%
P(7) = 9.250834%
P(8) = 9.392377%
P(9) = 9.387322%
P(10) = 9.591211%
```



Πιθανότητα απόρριψης πελάτη από το σύστημα:

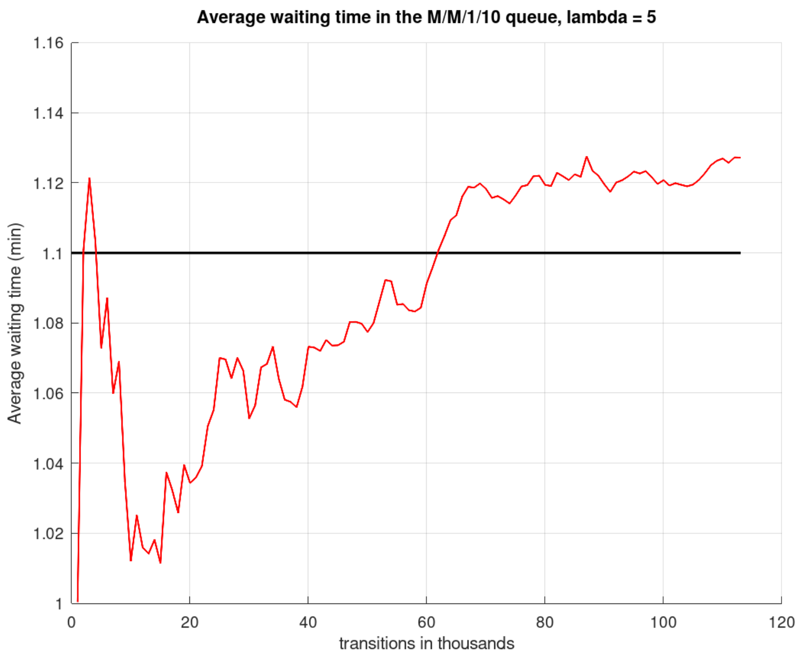
```
P[blocking] = P[10] = 9.387322%
```

Εξέλιξη του μέσου αριθμού πελατών:



Εξέλιξη του μέσου χρόνου καθυστέρησης:

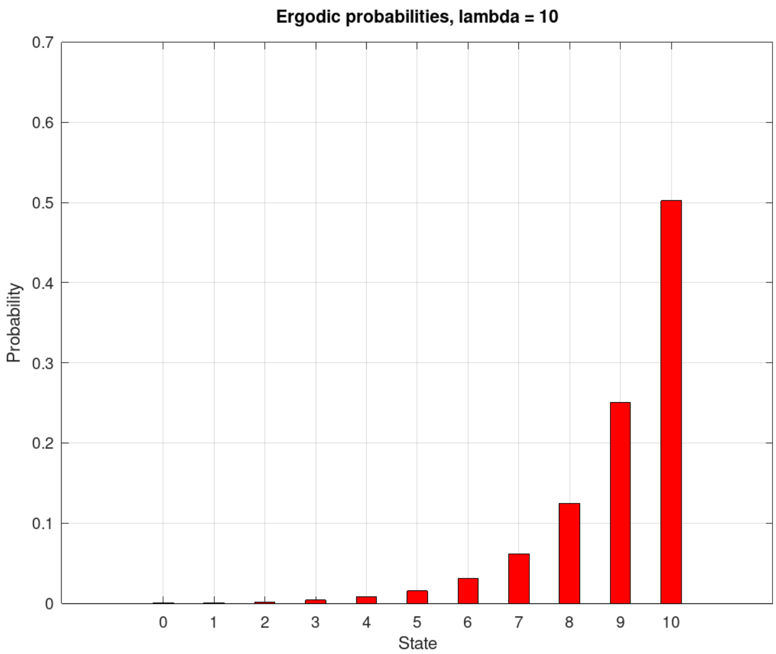
Από τον νόμο του Little: $E[T] = \dfrac{E[n(t)]}{\lambda(1 - P[\text{blocking}])}$



Για λ = 10, μ = 5:

Εργοδικές πιθανότητες των καταστάσεων του συστήματος:

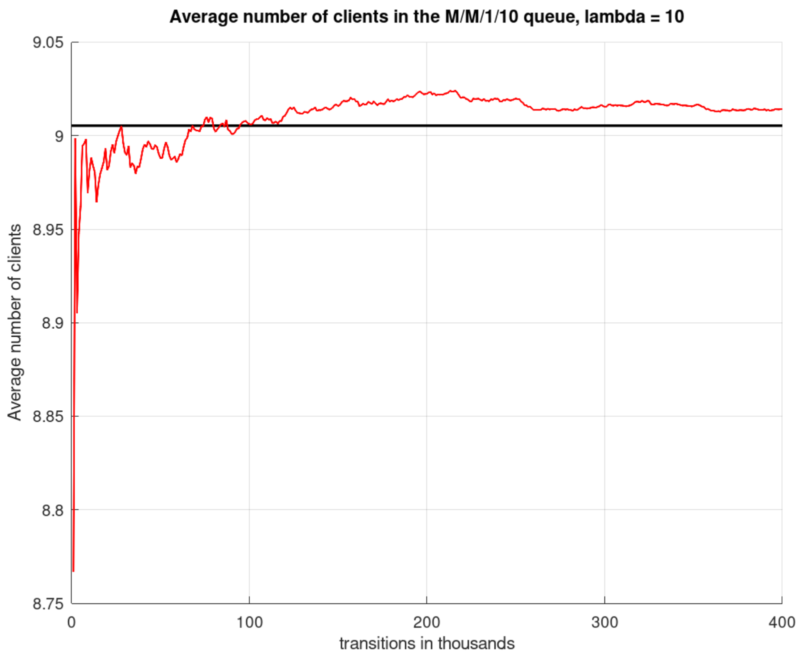
```
P(0) = 0.047928%
P(1) = 0.085372%
P(2) = 0.174487%
P(3) = 0.385670%
P(4) = 0.772837%
P(5) = 1.544550%
P(6) = 3.083110%
P(7) = 6.193928%
P(8) = 12.478470%
P(9) = 25.011233%
P(10) = 50.222415%
```



Πιθανότητα απόρριψης πελάτη από το σύστημα:

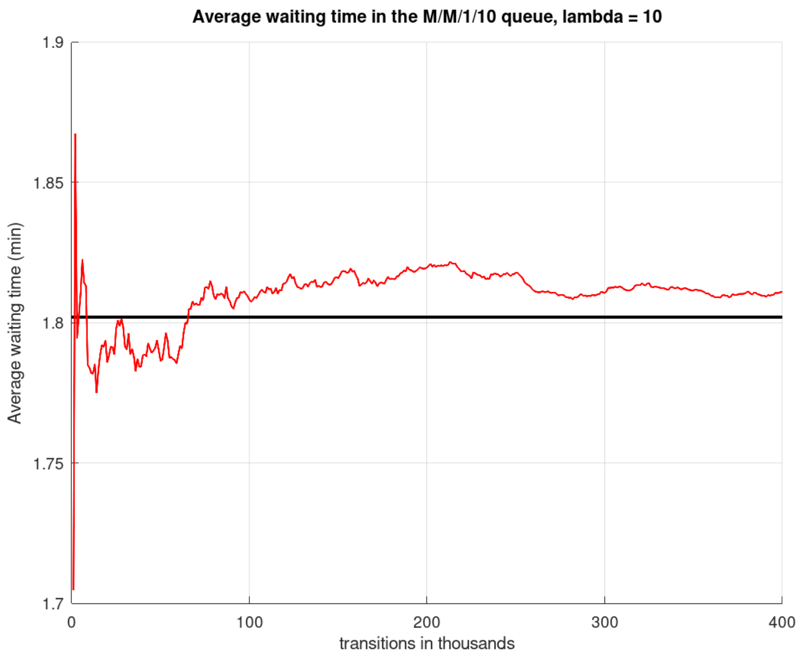
```
P[blocking] = P[10] = 25.011233%
```

Εξέλιξη του μέσου αριθμού πελατών:



Εξέλιξη του μέσου χρόνου καθυστέρησης:

Από τον νόμο του Little: $E[T] = \dfrac{E[n(t)]}{\lambda(1 - P[\text{blocking}])}$



(3)

Παρατηρούμε ότι όσο μεγαλώνει το λ, η προσομοίωση χρειάζεται περισσότερο χρόνο για να φτάσει στη σύγκλιση. Αυτό συμβαίνει διότι ο εξυπηρετητής δεν μπορεί να ικανοποιήσει με την ίδια επάρκεια τον ολοένα και αυξανόμενο φόρτο αφίξεων, με αποτέλεσμα να επισκεπτόμαστε πιο συχνά τις τελευταίες καταστάσεις. Έτσι, απαιτείται επιπλέον χρόνος προκειμένου:

- να μεταβούμε στις τελευταίες καταστάσεις (και ενδεχομένως να αρχίσουμε να απορρίπτουμε πελάτες)
- οι τελευταίες καταστάσεις να συγκλίνουν κι αυτές.

Πριν που το λ ήταν μικρότερο, η επισκεψιμότητα των τελευταίων καταστάσεων, και άρα η διαφοροποίησή τους μεταξύ διαδοχικών convergence tests ήταν κι αυτή μικρότερη. Λογικό είναι λοιπόν η διάρκεια της προσομοίωσης να αυξηθεί.

Επίσης, παρατηρώντας προσεκτικά τα διαγράμματα για λ = {1,5,10}, διακρίνουμε κάποια μεταβατικά φαινόμενα, δηλαδή ο μέσος αριθμός πελατών παρουσιάζει μεγάλες διακυμάνσεις αρχικά, μέχρι να σταθεροποιηθεί. Συγκεκριμένα, για λ = {1,5,10} οι διακυμάνσεις παρατηρούνται για τα πρώτα {15.000, 30.000, 40.000} transitions αντίστοιχα (περίπου). Επομένως, ένας τρόπος να επιταχύνουμε την προσομοίωση είναι να μην κάνουμε έλεγχο σύγκλισης μέχρι να φτάσουμε έναν ικανοποιητικό αριθμό transitions (15.000-40.000, ανάλογα την τιμή του λ) αφού βλέπουμε ότι πιθανότατα θα είναι πολύ νωρίς για να έχει συγκλίνει η προσομοίωση.

(4)

Αν είχαμε $\mu_i = \mu \cdot (i+1)$, με $\mu = 1$, θα έπρεπε να προσέξουμε ότι το threshold πλέον αλλάζει κάθε φορά που βρισκόμαστε σε άλλη κατάσταση. Έτσι, η ανάθεση του threshold πρέπει να μπει μέσα στο while-loop, ώστε αυτό να αλλάζει σε κάθε transition, και όχι μόνο την πρώτη φορά που τρέχουμε την προσομοίωση. Κατά τα άλλα, θα έχουμε:

$$threshold = \frac{\lambda_i}{\lambda_i + \mu_i} = \frac{\lambda_i}{\lambda_i + i + 1},\ \text{αφού μ} = 1$$

Παράρτημα Κώδικα

```
% M/M/1/10 simulation. We will find the probabilities of the first
    states.
% Note: Due to ergodicity, every state has a probability >0.
pkg load queueing;
clc;
clear all;
close all;
nfig = 0;
```

```
if !(exist("images","dir") == 7)
  mkdir "images";
endif

lambdas = [1,5,10];

capacity = 10;
accuracy = 0.00001;
total_transitions = 1000000;


for iteration = 1:3
  rand("seed",1);
  fd = fopen(strcat("images/trace",num2str(iteration),".txt"),"w");
  arrivals = zeros(1,capacity+1);
  P = zeros(1,capacity+1);
  to_plot = 0;
  to_plot_mean_time = 0;
  total_arrivals = 0; % to measure the total number of arrivals
  current_state = 0;  % holds the current state of the system
  previous_mean_clients = 0; % will help in the convergence test
  index = 0;

  lambda = lambdas(iteration);
  mu = 5;
  threshold = lambda/(lambda + mu); % the threshold used to calculate probabilities

  transitions = 0; % holds the transitions of the simulation in transitions steps

  while transitions >= 0
    transitions = transitions + 1; % one more transitions step

    if mod(transitions,1000) == 0 % check for convergence every 1000 transitions steps
      index = index + 1;
      for i=1:1:length(arrivals)
          P(i) = arrivals(i)/total_arrivals; % calculate the probability of every state in the system
      endfor

      mean_clients = 0; % calculate the mean number of clients in the system
      for i=1:1:length(arrivals)
         mean_clients = mean_clients + (i-1).*P(i);
      endfor

      to_plot(index) = mean_clients;
      to_plot_mean_time(index) = mean_clients/(lambda*(1-P(capacity+1)));

      if abs(mean_clients - previous_mean_clients) < accuracy || transitions > total_transitions % convergence test
        break;
      endif

      previous_mean_clients = mean_clients;

    endif

```

```
    random_number = rand(1); % generate a random number (Uniform distribution)
    if current_state == 0 || random_number < threshold % arrival
      if current_state <= capacity % not full system
        total_arrivals = total_arrivals + 1;
        if transitions <= 30
          fprintf(fd, "%02d. State %2d | Arrival   | %d Arrivals\n", transitions, current_state, arrivals(current_state+1));
        endif
        arrivals(current_state + 1) = arrivals(current_state + 1) + 1;
        if current_state < capacity
          current_state = current_state + 1;
        endif
      endif
    else % departure
      if current_state != 0 % no departure from an empty system
        if transitions <= 30
          fprintf(fd, "%02d. State %2d | Departure | %d Arrivals\n", transitions, current_state, arrivals(current_state+1));
        endif
        current_state = current_state - 1;
      endif
    endif
  endwhile

  fclose(fd);

  fd = fopen(strcat("images/prob",num2str(iteration),".txt"),"w");
  for i=1:1:length(arrivals)
    fprintf(fd,"P(%d) = %f%%\n",i-1,100*P(i));
  endfor
  fclose(fd);

  fd = fopen(strcat("images/pblocking",num2str(iteration),".txt"),"w");
  fprintf(fd,"P[blocking] = P[%d] = %f%%",capacity,100*P(capacity));
  fclose(fd);

  nfig=nfig+1; figure(nfig);
  x = 0:10;
  bar(x,P,'r',0.4);
  grid on;
  title(strcat("Ergodic probabilities, lambda = ",{" "},num2str(lambda)));
  xlabel("State");
  ylabel("Probability");
  saveas(nfig,strcat("images/prob",num2str(iteration),".png"));

  [dummy,response_time,average_requests] = qsmmmk(lambda,mu,1,10);

  nfig=nfig+1; figure(nfig);
  hold on;
  line([0,transitions/1000],[average_requests,average_requests],"color","black","linewidth",2);
  plot(to_plot,"r","linewidth",1.3);
  hold off;
  grid on;
  title(strcat("Average number of clients in the M/M/1/10 queue, lambda = ",{" "},num2str(lambda)));
##  if iteration == 2
```

```
##     ylim([3.4,7]);
##   endif
  xlabel("transitions in thousands");
  ylabel("Average number of clients");
  saveas(nfig,strcat("images/clients",num2str(iteration),".png"));

  nfig=nfig+1; figure(nfig);
  hold on;
  line([0,transitions/1000],[response_time,response_time],"color","
    black","linewidth",2);
  plot(to_plot_mean_time,"r","linewidth",1.3);
  hold off;
  grid on;
  title(strcat("Average waiting time in the M/M/1/10 queue, lambda
    = ",{" "},num2str(lambda)));
##   if iteration == 2
##     ylim([0.5,1.7]);
##   endif
  xlabel("transitions in thousands");
  ylabel("Average waiting time (min)");
  saveas(nfig,strcat("images/time",num2str(iteration),".png"));
endfor
```