3^{η} Εργαστηριακή Άσκηση στο μάθημα Συστήματα Αναμονής

Λεωνίδας Αβδελάς | ΑΜ: 03113182

Προσομοίωση συστήματος M/M/1/10

Σχεδιάσαμε και υλοποιήσαμε την προσομοίωση του συστήματος σε Octave. Σχεδιαστικά βασιστήκαμε στον κώδικα που μας δόθηκε από το εργαστήριο. Η αλλαγή που κάναμε ήταν ότι μετά τις 10 καταστάσεις, ουσιαστικά θεωρούσαμε ότι το $\lambda = 0$ και άρα δεν υπήρχε πιθανότητα για περαιτέρω μετάβαση.

(1)

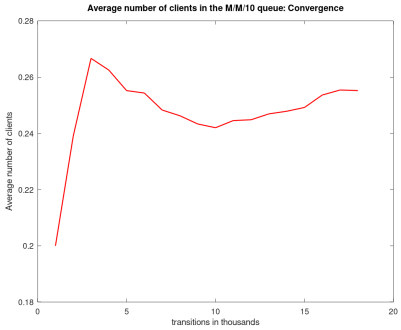
Εξετάσαμε τα αποτελέσματα για τα πρώτα 30 βήματα της προσομοίωσης και εξακριβώσαμε ότι όντως δούλευε με τον αναμενόμενο τρόπο.

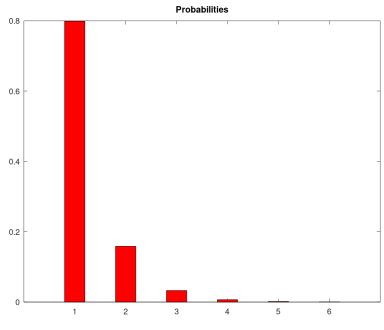
(2)

Στο Σχήμα 1, βλέπουμε τα αποτελέσματα της προσομοίωσης για $\lambda = 1$. Στο Σχήμα 2, βλέπουμε τα αποτελέσματα της προσομοίωσης για $\lambda = 5$. Στο Σχήμα 3, βλέπουμε τα αποτελέσματα της προσομοίωσης για $\lambda = 10$. Όλες οι προσομοιώσεις έχουν την ίδια σειρά ψευδό-τυχαίων αριθμών. Όπως βλέπουμε, για $\lambda = 10$, το σύστημα βρίσκεται πάντα στην κατάσταση 10, δηλαδή είναι γεμάτο, οπότε δεν μπορεί να υπάρξει άλλη άφιξη. Για $\lambda = 5$, έχουμε ισορροπία στο σύστημα, και μικρότερη πιθανότητα απόρριψης.

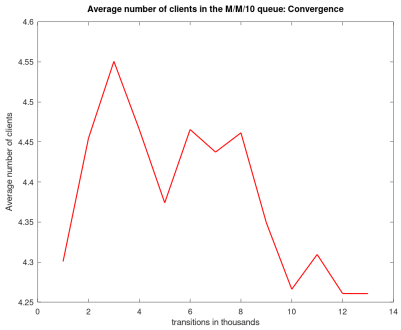
(3)

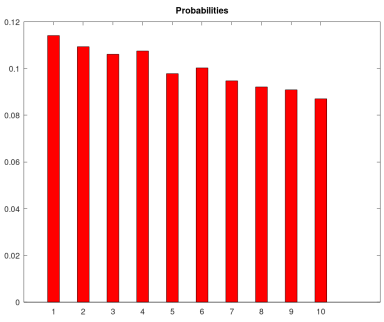
Η ταχύτητα σύγκλισης φαίνεται να μικραίνει όσο μεγαλώνει το λ. Παρόλα αυτά βλέπουμε ότι το διάστημα στο οποίο κινήται ο μέσος αριθμός πελατών, μεγαλώνει



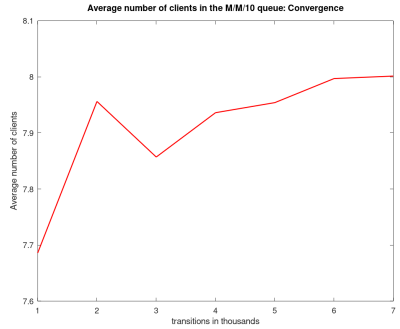


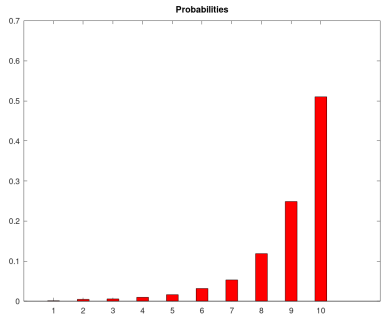
Σχήμα 1: Προσομοίωση για $\lambda = 1$





Σχήμα 2: Προσομοίωση για $\lambda = 5$





Σχήμα 3: Προσομοίωση για $\lambda = 10$

όσο μεγαλώνει το λ.

Έτσι, για $\lambda = 1$, μετά τις πρώτες 5000 μεταβάσεις, οι τιμές φαίνονται αρκετά σταθερές. Για $\lambda = 5$, το σημείο αυτό είναι μετά από 8000 μεταβάσεις. Για $\lambda = 10$, μετά από περίπου 6000 μεταβάσεις. Έτσι συνολικά, μετά από περίπου 6000 θα έχουμε φτάσει στην σταθερή κατάσταση.

(4)

Η αλλαγή θα ήταν αρκετά εύκολη, αφού απλά θα ορίζαμε το μ ώς συνάρτηση της τρέχουσας κατάστασης σε κάθε μετάβαση. Επίσης το threshold θα έπρεπε να υπολογίζεται από την αρχή κάθε φορά.

Παράρτημα: Κώδικας Octave

```
function [arrivals , P, P_block , mean_cl_time , mean_wait_time]  = ...
  finite_storage (lambda , debugging , seed , variable_mu)
% M/M/1/10 simulation. We will find the probabilities of the first states.
% Note: Due to ergodicity , every state has a probability >0.

  if seed
    rand("seed", 1);
  endif
  % to measure the total number of arrivals
  total_arrivals = 0;

  % holds the current state of the system
  current_state = 0;

  % will help in the convergence test
  previous_mean_clients = 0;
  index = 0;

  mu = 5;

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

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

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

    if variable_mu
      mu = 1 * (current_state + 1);
      threshold = lambda/(lambda + mu);
    endif

    if debugging
      disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')
      printf('Iteration number: %d.\n', transitions);
      printf('Current state is %d.\n', current_state);
      printf('Total number of arrivals is %d.\n', total_arrivals);
```

```
41      P = 0;
42      to_plot = 0;
43      P_block = 0;
44    endif
45
46    % check for convergence every 1000 transitions steps
47    if mod(transitions,1000) == 0
48      index = index + 1;
49      for i=1:1:length(arrivals)
50          % calcuate the probability of every state in the system
51          P(i) = arrivals(i)/total_arrivals;
52
53      endfor
54          P_block = P(length(arrivals)) * lambda;
55
56      % calculate the mean number of clients in the system
57      mean_clients = 0;
58      for i=1:1:length(arrivals)
59         mean_clients = mean_clients + (i-1).*P(i);
60      endfor
61
62      mean_wait_time(index) = mean_clients./(lambda*(1-P(length(arrivals))));
63      mean_cl_time(index) = mean_clients;
64
65      % convergence test
66      if abs(mean_clients - previous_mean_clients)...
67        < 0.001 * previous_mean_clients || transitions > 1000000
68        break;
69      endif
70
71      previous_mean_clients = mean_clients;
72
73    endif
74
75    % generate a random number (Uniform distribution)
76      random_number = rand(1, seed);
77
78    % arrival
79    if (current_state == 0 || random_number < threshold) && (current_state <
      10)
80      total_arrivals = total_arrivals + 1;
81
82      if debugging
83        printf('We have an arrival.\n')
84      endif
85
86      try
87        % to catch the exception if variable arrivals(i) is undefined.
88        % Required only for systems with finite capacity.
89        % increase the number of arrivals in the current state
90        if current_state == 10
91          continue
92        else
93          arrivals(current_state + 1) = arrivals(current_state + 1) + 1;
94          current_state = current_state + 1;
95        endif
96      catch
97        arrivals(current_state + 1) = 1;
98        current_state = current_state + 1;
99      end
100   else
101    % departure
102    if debugging
103       printf('We have a departure.\n');
104     endif
105
106     if current_state != 0 % no departure from an empty system
107       current_state = current_state - 1;
```

```
      endif
    endif

    if (debugging == true) && (transitions > 30)
      break
    endif
  endwhile
endfunction
```

```
clc;
clear all;
close all;

[arrivals, P, P_block, to_plot, mean_wait_time] = ...
  finite_storage(5, false, true, true);

for i=1:1:length(arrivals)
  display(P(i));
endfor

figure(1);
plot(to_plot,"r","linewidth",1.3);
title("Average number of clients in the M/M/10 queue: Convergence");
xlabel("transitions in thousands");
ylabel("Average number of clients");

figure(2);
bar(P,'r',0.4);
title("Probabilities")

figure(3);
plot(mean_wait_time,"r","linewidth",1.3);
title("Average waiting time of clients in the M/M/10 queue: Convergence");
xlabel("transitions in thousands");
ylabel("Average wait time of clients");


disp('Sum of all probabilities is:')
disp(sum(P))
```