Συστήματα Αναμονής

(Queuing Systems)

3η Εργαστηριακή Άσκηση

Λεούσης Σάββας

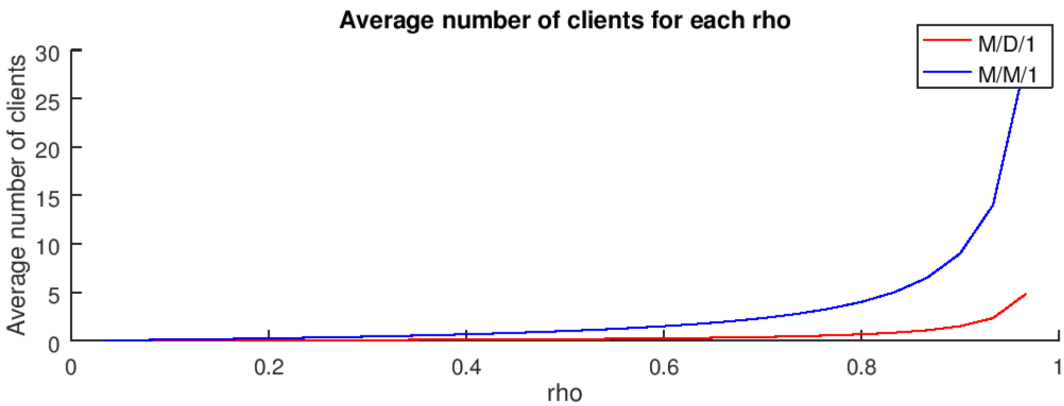
Α.Μ.: 03114945

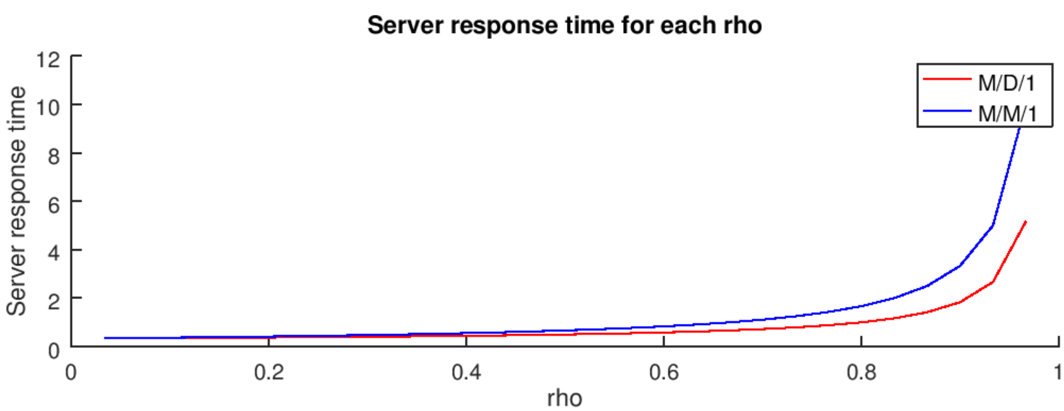
Σύγκριση συστημάτων M/M/1 και M/D/1

1. Ο μέσος χρόνος καθυστέρησης ενός πελάτη στην ουρά M/D/1, σύμφωνα με τον τύπο του Little, είναι $E(T) = \frac{E[n(t)]}{\lambda} = \frac{\rho + \frac{1}{2}\left(\frac{\rho^2}{1-\rho}\right)}{\lambda} = \frac{\frac{2\rho - 2\rho^2 + \rho^2}{2-2\rho}}{\lambda} = \frac{2\rho - \rho^2}{(2-2\rho)\lambda} = \frac{\rho(2-\rho)}{2(1-\rho)\lambda}$ και ο μέσος χρόνος αναμονής είναι $E(W) = E(T) - \frac{1}{\mu} = \frac{\rho(2-\rho)}{2(1-\rho)\lambda} - \frac{1}{\mu} = \frac{\rho}{2(1-\rho)\mu} = \frac{\rho}{2(\mu-\lambda)}$, ενώ η απαραίτητη συνθήκη έτσι ώστε η ουρά M/D/1 να είναι εργοδική είναι να ισχύει ότι $\rho < 1 \Leftrightarrow \lambda < \mu$.
2. Η ζητούμενη συνάρτηση `qsmd1.m` είναι για ουρές M/D/1 είναι η παρακάτω:

```
function [U R Q X] = qsmd1( lambda, mu )
  if ( nargin != 2 )
    print_usage();
  endif
  ( isvector(lambda) && isvector(mu) ) || ...
      error( "lambda and mu must be vectors" );
  [ err lambda mu ] = common_size( lambda, mu );
  if ( err )
    error( "parameters are of incompatible size" );
  endif
  lambda = lambda(:)';
  mu = mu(:)';
  all( lambda >= 0 ) || ...
      error( "lambda must be >= 0" );
  all( mu > lambda ) || ...
      error( "The system is not ergodic" );
  U = rho = lambda ./ mu; % Server utilization
  R = (2-rho)./((1-rho).*mu*2); % Server response time
  Q = rho./(2*(mu-lambda)); % Average number of requests
in the system
  X = lambda; % Server throughput
endfunction
```

3. Χρησιμοποιώντας τις συναρτήσεις `qsmm1` και `qsmd1`, προκύπτουν οι παρακάτω γραφικές παραστάσεις:





Παρατηρώντας τις παραπάνω γραφικές παραστάσεις, βλέπουμε ότι από τις δύο ουρές τόσο ο μέσος αριθμός πελατών στο σύστημα, όσο και ο μέσος χρόνος καθυστέρησης είναι σαφώς μικρότεροι για την ουρά M/D/1. Επομένως, η ουρά M/D/1 είναι το καλύτερο σύστημα από τα δύο.

Προσομοίωση συστήματος M/M/1/10

Η ζητούμενος κώδικας προσωμοίωσης ενός συστήματος M/M/1/10 είναι ο παρακάτω:

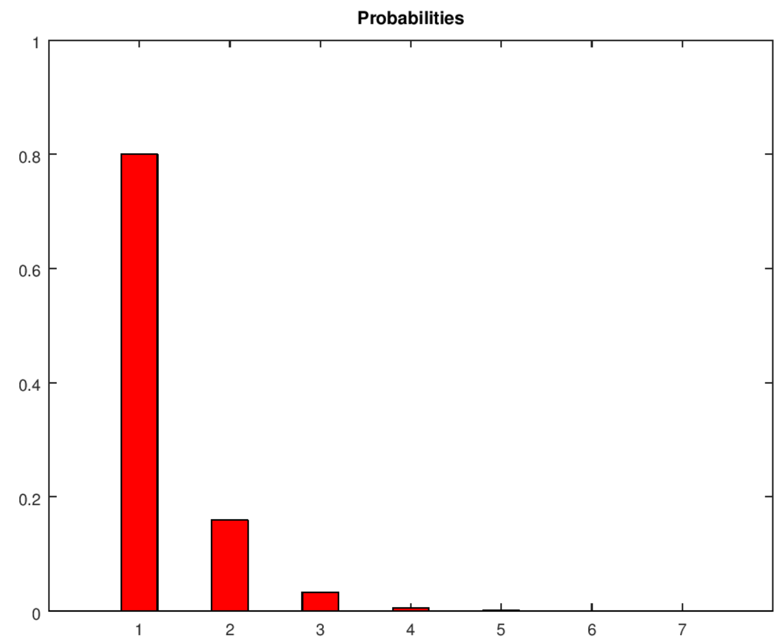
```
lambda = [1,5,10];
mu = 5;

for lambda = [1,5,10]
  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;
  clear arrivals;
  clear P;
  display(lambda);
  threshold = lambda/(lambda + mu); % the threshold used to
```

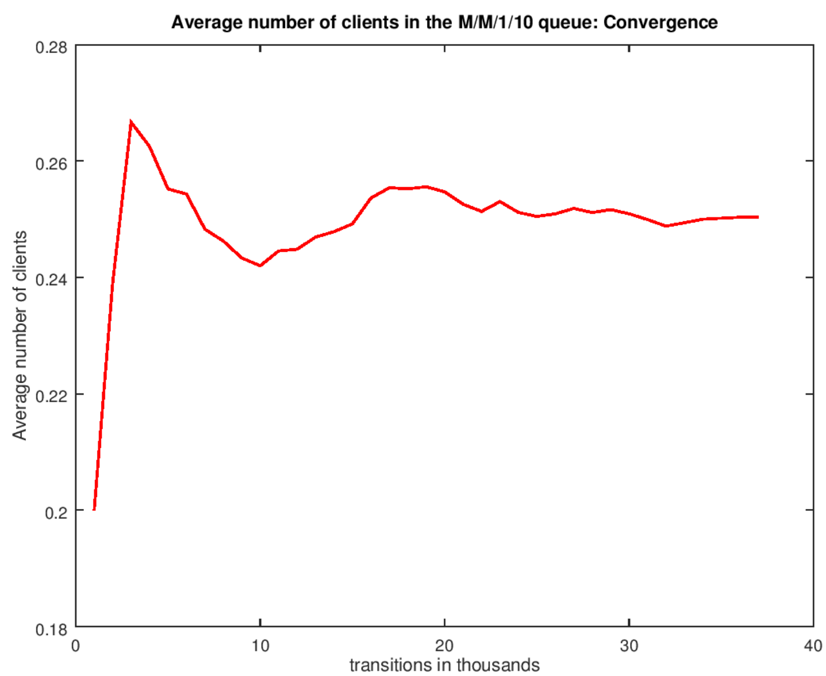
```
calculate probabilities
  rand("seed",1);
  transitions = 0; % holds the transitions of the simulation
in transitions steps

  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
      P_blocking = P(length(arrivals));
      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;

      if abs(mean_clients - previous_mean_clients) < 0.00001
|| transitions > 1000000 % convergence test
        break;
      endif

      previous_mean_clients = mean_clients;

    endif

    random_number = rand(1); % generate a random number
(Uniform distribution)
#    if (transitions<=30) % debugging
#      display("##### NEW TRANSITION #####");
#      display(transitions);
#      display(current_state);
#      if current_state == 0 || random_number < threshold
#        display("Next transition is an arrival.");
#      else
#        display("Next transition is a departure.");
#      endif
#      display(total_arrivals);
#    endif
    if current_state == 0 || random_number < threshold %
arrival
      total_arrivals = total_arrivals + 1;
      try % to catch the exception if variable arrivals(i) is
```

```
undefined. Required only for systems with finite capacity.
        arrivals(current_state + 1) = arrivals(current_state
+ 1) + 1; % increase the number of arrivals in the current
state
      catch
        arrivals(current_state + 1) = 1;
      end
      if current_state == 10
        continue;
      else
        current_state = current_state + 1;
      endif
    else % departure
      if current_state != 0 % no departure from an empty
system
        current_state = current_state - 1;
      endif
    endif
  endwhile

  for i=1:1:length(arrivals)
    display(P(i));
  endfor
  display(P_blocking);
  figure(fig_num++);
  plot(to_plot,"r","linewidth",1.3);
  title("Average number of clients in the M/M/1/10 queue:
Convergence");
  xlabel("transitions in thousands");
  ylabel("Average number of clients");

  figure(fig_num++);
  bar(P,'r',0.4);
  title("Probabilities");
endfor
```

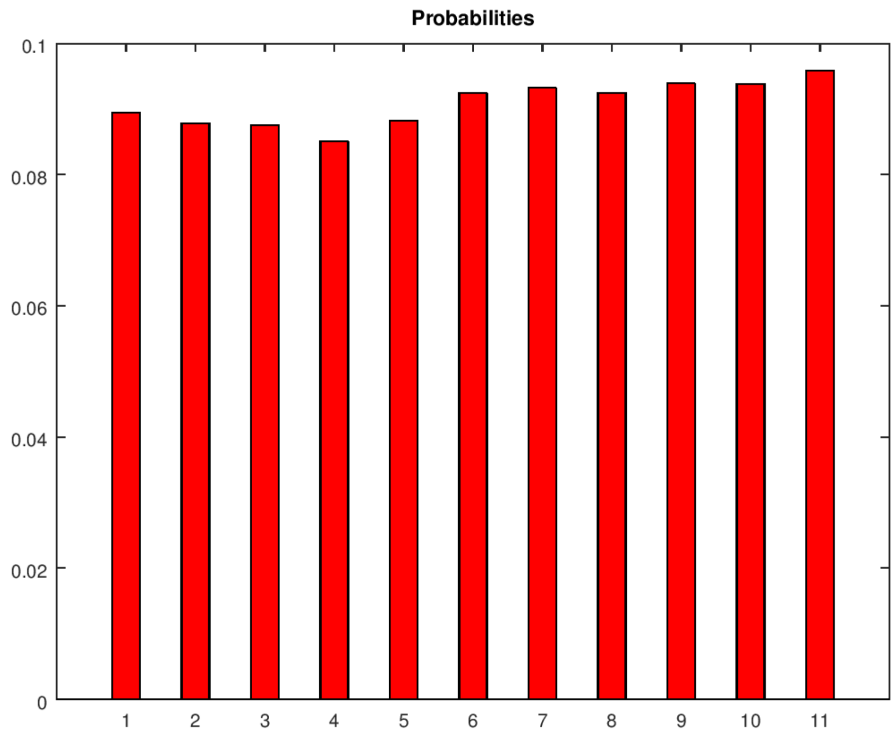
Για κάθε τιμή του λ προέκυψαν οι παρακάτω γραφικές παραστάσεις των εργοδικών πιθανοτήτων καθώς και της εξέλιξης του μέσου αριθμού πελατών στο σύστημα ανά 1000 μεταβάσεις:

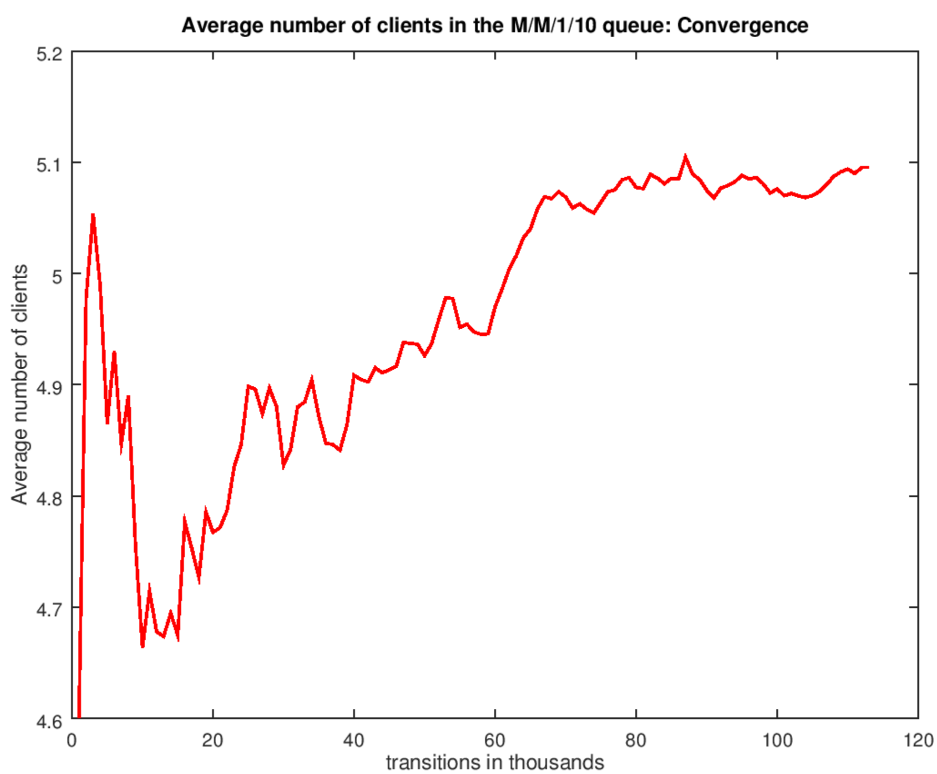
- λ=1



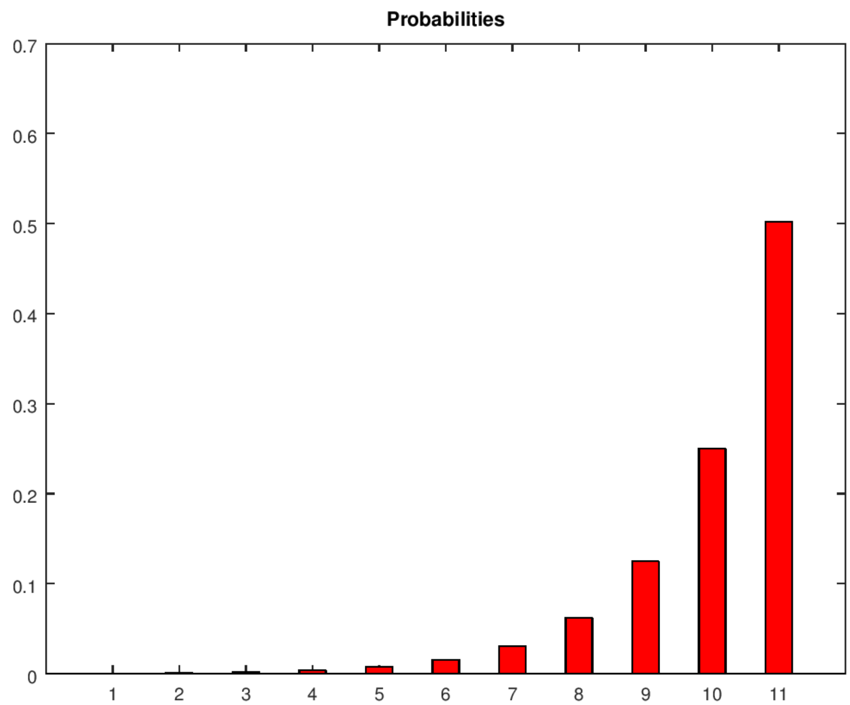


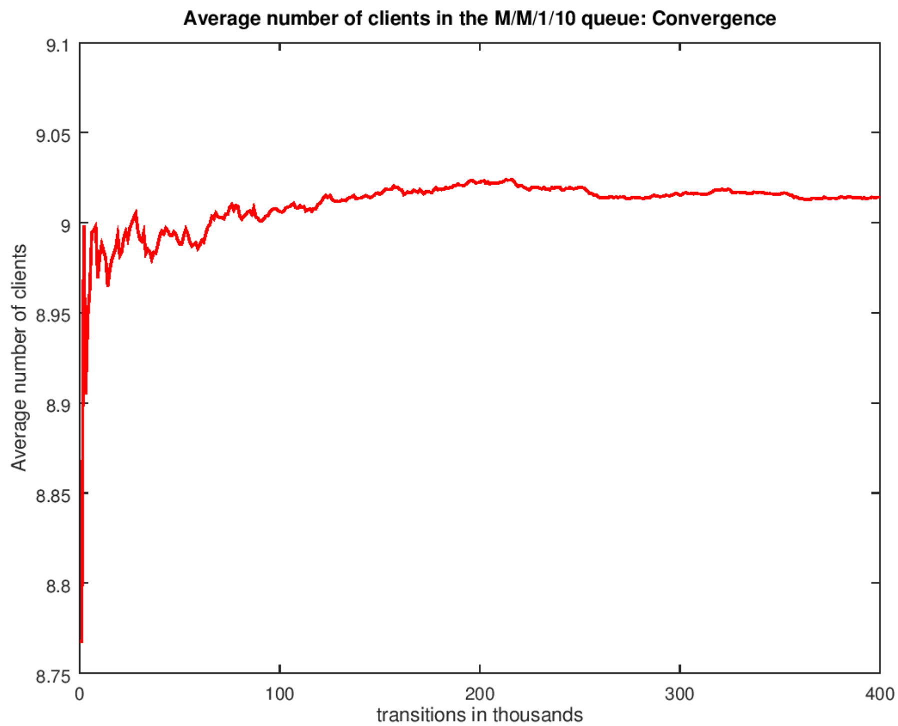
- λ=5


Probabilities
0.1
0.08
0.06
0.04
0.02
0
1
2
3
4
5
6
7
8
9
10
11


Average number of clients in the M/M/1/10 queue: Convergence
Average number of clients
5.2
5.1
5
4.9
4.8
4.7
4.6
0
20
40
60
80
100
120
transitions in thousands

- λ=10





Σύμφωνα με τις παραπάνω γραφικές παραστάσεις, παρατηρούμε ότι όσο αυξάνεται το λ, τόσο αυξάνεται ο απαιτούμενος αριθμός μεταβάσεων έτσι ώστε το σύστημα να ικανοποιήσει το κριτήριο σύγκλισης. Προκειμένου να επιταχυνθεί η προσωμοίωση, θα μπορούσαμε να αγνοήσουμε τουλάχιστον 40.000 αρχικές μεταβάσεις, καθώς για τη σύγκλιση μας ενδιαφέρει η μόνιμη κατάσταση του συστήματος.

Προσομοίωση συστήματος M/M/1/5 με μεταβλητό μέσο ρυθμό εξυπηρέτησης

1. Οι εργοδικές πιθανότητες του συστήματος με τη βοήθεια του πακέτου queueing του Octave είναι οι εξής:

```
P0 =

   0.172822   0.252368   0.244132   0.177036   0.103107   0.050535
```

2. Τα ζητούμενα αποτελέσματα της προσωμοίωσης για την κάθε τιμή του κριτηρίου τερματισμού είναι τα παρακάτω:
   a. 1%

```
criterion =  1
transitions =  2000
P =

   0.174464   0.249513   0.237817   0.197856   0.091618   0.048733

P_blocking =  0.048733
mean_clients =  1.9288
```

b. 0.1%

```
criterion =  0.10000
transitions =  1000
P =

   0.200000   0.262136   0.223301   0.161165   0.095146   0.058252

P_blocking =  0.058252
mean_clients =  1.8641
```

c. 0.01%

```
criterion =  0.010000
transitions =  1000
P =

   0.200000   0.262136   0.223301   0.161165   0.095146   0.058252

P_blocking =  0.058252
mean clients =  1.8641
```

d. 0.001%

```
criterion =  0.0010000
transitions =  1000
P =

   0.200000   0.262136   0.223301   0.161165   0.095146   0.058252

P_blocking =  0.058252
mean_clients =  1.8641
```

e. 0.0001%

```
criterion =   1.0000e-004
transitions =  1000
P =

   0.200000   0.262136   0.223301   0.161165   0.095146   0.058252

P_blocking =  0.058252
mean_clients =  1.8641
```

f. 0.00001%

```
criterion =   1.0000e-005
transitions =  1000
P =

   0.200000   0.262136   0.223301   0.161165   0.095146   0.058252

P_blocking =  0.058252
mean_clients =  1.8641
```

g. 0.000001%

```
criterion =   1.0000e-006
transitions =  1000
P =

   0.200000   0.262136   0.223301   0.161165   0.095146   0.058252

P_blocking =  0.058252
mean clients =  1.8641
```

h. 0.0000001%

```
criterion =   1.0000e-007
transitions =  1000
P =

   0.200000   0.262136   0.223301   0.161165   0.095146   0.058252

P_blocking =  0.058252
mean_clients =  1.8641
```

Παράρτημα (κώδικας Lab3.m)

```
clc;
clear all;
close all;
fig_num = 1;
############# M/M/1 AND M/D/1 SYSTEMS COMPARISON
#############

# 3

lambda=0.1:0.1:2.9;
mu=[3];
[UD RD QD XD] = qsmd1(lambda,mu);
[UM RM QM XM p0] = qsmm1(lambda,mu);
figure(fig_num++);
subplot(2,1,1);
hold on;
plot(lambda./mu,QD,"r");
plot(lambda./mu,QM,"b");
hold off;
title("Average number of clients for each rho");
xlabel("rho");
ylabel("Average number of clients");
legend("M/D/1","M/M/1");

subplot(2,1,2);
hold on;
plot(lambda./mu,RD,"r");
plot(lambda./mu,RM,"b");
hold off;
title("Server response time for each rho");
xlabel("rho");
ylabel("Server response time");
legend("M/D/1","M/M/1");

############# M/M/1/10 SYSTEM SIMULATION #############

lambda = [1,5,10];
mu = 5;

for lambda = [1,5,10]
  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;
  clear arrivals;
  clear P;
  display(lambda);
  threshold = lambda/(lambda + mu); % the threshold used
to calculate probabilities
  rand("seed",1);
  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
      P_blocking = P(length(arrivals));
      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;

      if abs(mean_clients - previous_mean_clients) <
0.00001 || transitions > 1000000 % convergence test
        break;
      endif

      previous_mean_clients = mean_clients;

    endif

    random_number = rand(1); % generate a random number
(Uniform distribution)
#    if (transitions<=30) % debugging
#      display("##### NEW TRANSITION #####");
#      display(transitions);
#      display(current_state);
#      if current_state == 0 || random_number <
threshold
#        display("Next transition is an arrival.");
```

```
#      else
#        display("Next transition is a departure.");
#      endif
#      display(total_arrivals);
#    endif
    if current_state == 0 || random_number < threshold %
arrival
      total_arrivals = total_arrivals + 1;
      try % to catch the exception if variable
arrivals(i) is undefined. Required only for systems with
finite capacity.
        arrivals(current_state + 1) =
arrivals(current_state + 1) + 1; % increase the number
of arrivals in the current state
      catch
        arrivals(current_state + 1) = 1;
      end
      if current_state == 10
        continue;
      else
        current_state = current_state + 1;
      endif
    else % departure
      if current_state != 0 % no departure from an empty
system
        current_state = current_state - 1;
      endif
    endif
  endwhile

  for i=1:1:length(arrivals)
    display(P(i));
  endfor
  display(P_blocking);
  figure(fig_num++);
  plot(to_plot,"r","linewidth",1.3);
  title("Average number of clients in the M/M/1/10
queue: Convergence");
  xlabel("transitions in thousands");
  ylabel("Average number of clients");

  figure(fig_num++);
  bar(P,'r',0.4);
  title("Probabilities");
endfor

############# M/M/1/5 SYSTEM SIMULATION WITH VARIABLE MU
#############

# 1

states = [0,1,2,3,4,5];
```

```
initial_state = [1,0,0,0,0,0];
lambda = 3;
mu = 1;
births_B = [lambda,lambda,lambda,lambda,lambda];
deaths_D = [mu*2,mu*3,mu*4,mu*5,mu*6];
transition_matrix = ctmcbd(births_B,deaths_D);
P = ctmc(transition_matrix);
for i=[1,2,3,4,5,6]
  index = 0;
  for T=0:0.01:50
    index = index + 1;
    P0 = ctmc(transition_matrix,T,initial_state);
    Prob0(index) = P0(i);
    if P0-P < 0.01
      break;
    endif
  endfor
endfor
display(P0);

# 2

lambda = 3;
mu = [1,2,3,4,5,6];

for criterion =
[1,0.1,0.01,0.001,0.0001,0.00001,0.000001,0.0000001]
  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;
  clear arrivals;
  clear P;
  rand("seed",1);
  transitions = 0; % holds the transitions of the
simulation in transitions steps

  while transitions >= 0
    threshold = lambda/(lambda + current_state + 1);
    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
      P_blocking = P(length(arrivals));
```

```
      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;
      if abs(mean_clients - previous_mean_clients) <
1/criterion || transitions > 1000000 % convergence test
        display(criterion);
        display(transitions);
        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
      total_arrivals = total_arrivals + 1;
      try % to catch the exception if variable
arrivals(i) is undefined. Required only for systems with
finite capacity.
        arrivals(current_state + 1) =
arrivals(current_state + 1) + 1; % increase the number
of arrivals in the current state
      catch
        arrivals(current_state + 1) = 1;
      end
      if current_state == 5
        continue;
      else
        current_state = current_state + 1;
      endif
    else % departure
      if current_state != 0 % no departure from an empty
system
        current_state = current_state - 1;
      endif
    endif
  endwhile
  display(P);
  display(P_blocking);
  display(mean_clients);
endfor
```