Queuing Theory Exercise Series 3

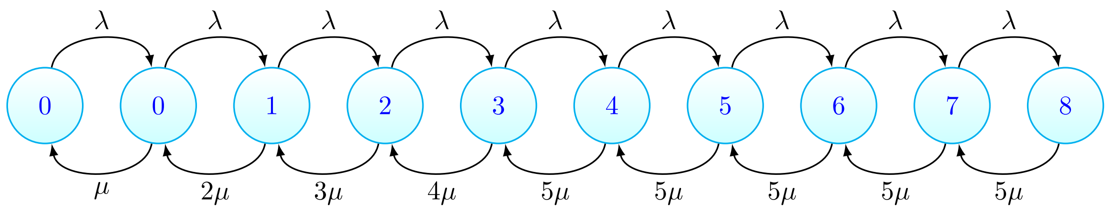
Alexandros Kyriakakis (03112163)

May 2020

1 System Simulation M/M/N/K

1.1

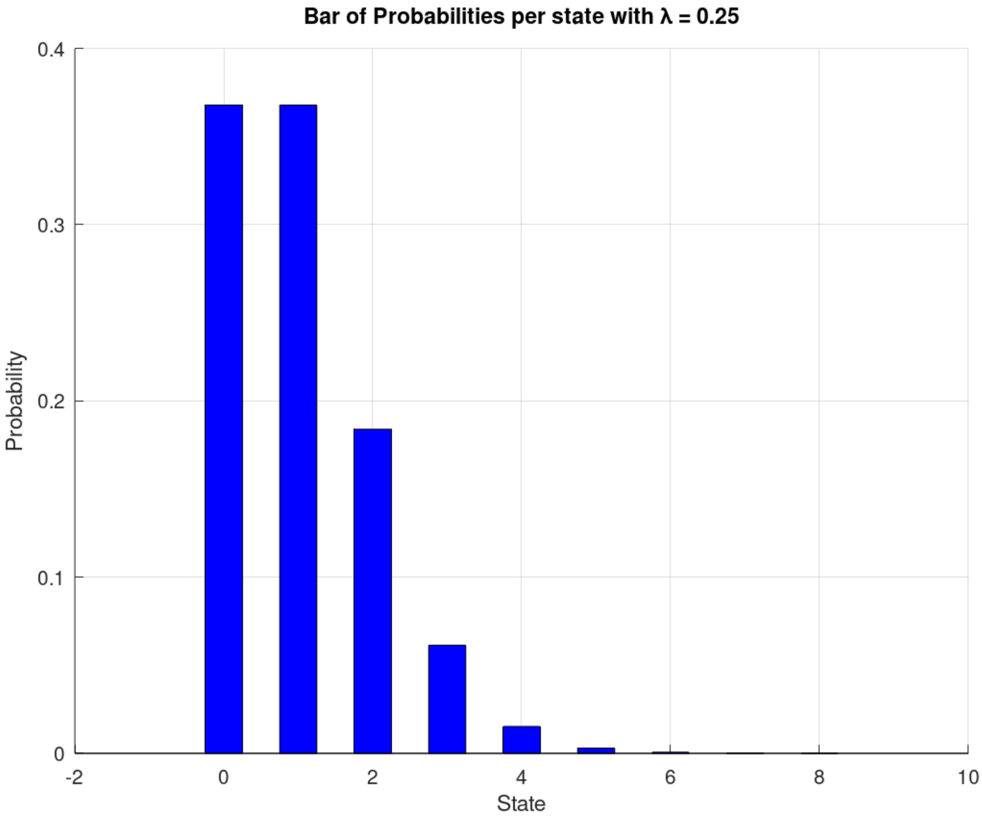
Following we draw the diagram of transition rates of the system between the ergodic states.



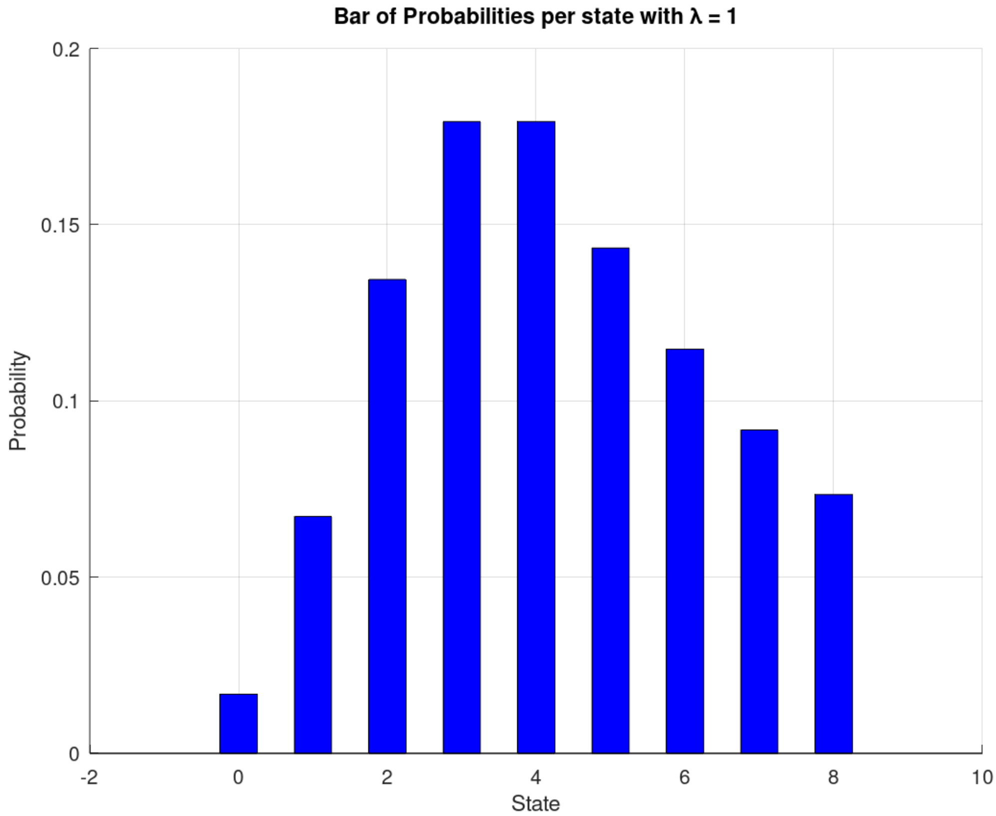
1.2

Using the recomended Queuing packages for Octave we ploted the ergodic probabilities of the system.

1.2.1 $\lambda = \frac{1}{4}\frac{customers}{min}$



1.2.2 $\lambda = 1 \frac{customers}{min}$



1.3

The probability of having customers waiting in the queue, is the probability of being in states 5 or 6 or 7, hence

$$P_{waiting} = P_5 + P_6 + P_7$$

1.3.1 $\lambda = \frac{1}{4} \frac{customers}{min}$

$$Erlang\ C: \ 0.0038314$$

$$Estimated\ Prob: \ 0.0038008$$

We notice that the prices above are very close.

1.3.2 $\lambda = 1 \frac{customers}{min}$

$$Erlang\ C: \ 0.55411$$

$$Estimated\ Prob: \ 0.3498$$

We notice that the prices above are not very close. This happens because there is large amount of services per server.

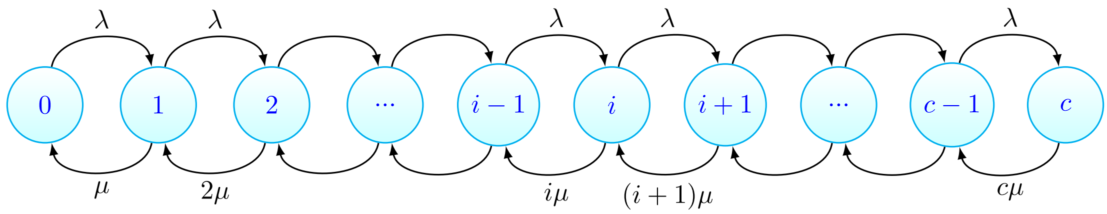
1.4 Code

```
pkg load statistics
pkg load queueing

clc;
clear all;
close all;


lambda = [1/4, 1];
for i=1:columns(lambda)
    mu = 1/4;
    states = [0, 1, 2, 3, 4, 5, 6, 7, 8]; % system with capacity 4 states
    % the initial state of the system. The system is initially empty.
    initial_state = [1, 0, 0, 0, 0, 0, 0, 0, 0];

    % define the birth and death rates between the states of the system.
    births_B = [ lambda(i), lambda(i), lambda(i), lambda(i), lambda(i), lambda(i), lambda(i), lambda(i)];
    deaths_D = [ mu, mu*2, mu*3, mu*4, mu*5, mu*5, mu*5, mu*5];

    % get the transition matrix of the birth-death process
    transition_matrix = ctmcbd(births_B, deaths_D);
    display (transition_matrix)

    # (ii)
    % get the ergodic probabilities of the system
    P = ctmc(transition_matrix);
    display (P);

    figure(i);
    hold on;
    title(strjoin({"Bar of Probabilities per state with \\lambda = ",num2str(lambda(i))}, ""))
    xlabel("State")
    ylabel("Probability")
    bar(states, P, "b", 0.5);
    grid on;
    saveas (i, strjoin({"figures/figure_lambda",num2str(i),".png"},""))
    hold off;
    display(strjoin({"Erlang C for lambda = ",num2str(lambda(i)),": ", num2str(erlangc(lambda(i)/mu,5))}, ""))
    display(strjoin({"Estimated Prob for lambda = ",num2str(lambda(i)),": ", num2str(P(6)+P(7)+P(8))}, ""))
endfor

clc;
clear all;
close all;
exit;
```

2 Calling Center Analysis

2.1

The transmision rate diagram of the system M/M/c/c.



Now we are going to evaluate Erlang-B formula. We know that

$$k\mu P_k = \lambda P_{k-1} \Rightarrow P_k = \frac{\lambda}{k\mu} P_{k-1} \Rightarrow P_k = \frac{\rho}{k} P_{k-1},\ k = 1, 2, .., c$$

This leads as to a recursive formula which can be solved as

$$k = 1 \rightarrow P_1 = \rho P_0$$

$$k = 2 \rightarrow P_2 = \rho P_1 = \frac{\rho^2}{2!} P_0$$

$$k = 3 \rightarrow P_3 = \rho P_2 = \frac{\rho^3}{3!} P_0$$

So we notice that

$$P_k = \frac{\rho^k}{k!} P_0,\ k = 1, 2, ..., c$$

Now if we use the cumulative probability formula we take the probability of rejecting a customer.

$$P_0 + P_1 + ... + P_c = 1 \Rightarrow \sum_{k=0}^{c} P_k = 1 \Rightarrow P_0 = \frac{1}{\sum_{k=0}^{c} \frac{\rho^k}{k!}} \Rightarrow P_{rejecting} = P_c = \frac{\rho^c}{c!} P_0 \Rightarrow P_{blocking} = \frac{\frac{\rho^c}{c!}}{\sum_{k=0}^{c} \frac{\rho^k}{k!}}$$

Hence, the average rate of blocking in the M/M/c/c is $\lambda P_{blocking}$.

2.1.1 Erlang Factorial

```
pkg load queueing
addpath(pwd);
function Result = erlang_factorial(ro,c)
    arithmitis = (ro^c)/factorial(c)
    paranomasths = 0
    i = 0;
    while i <= c
        paranomasths += (ro^i)/factorial(i);
        i++;
    endwhile
    Result = arithmitis/paranomasths;
endfunction
%display(erlang_factorial(1024,1024));
%display(erlangb(1024,1024));
exit;
```

2.2 Erlang Iterative

```
addpath(pwd);
function Result = erlang_iterative(ro,n)
    i = 0;
    Result = 1;
    while i <= n
        Result = ro * Result/(ro*Result + i);
        i = 1+i;
    endwhile
endfunction
```

2.3

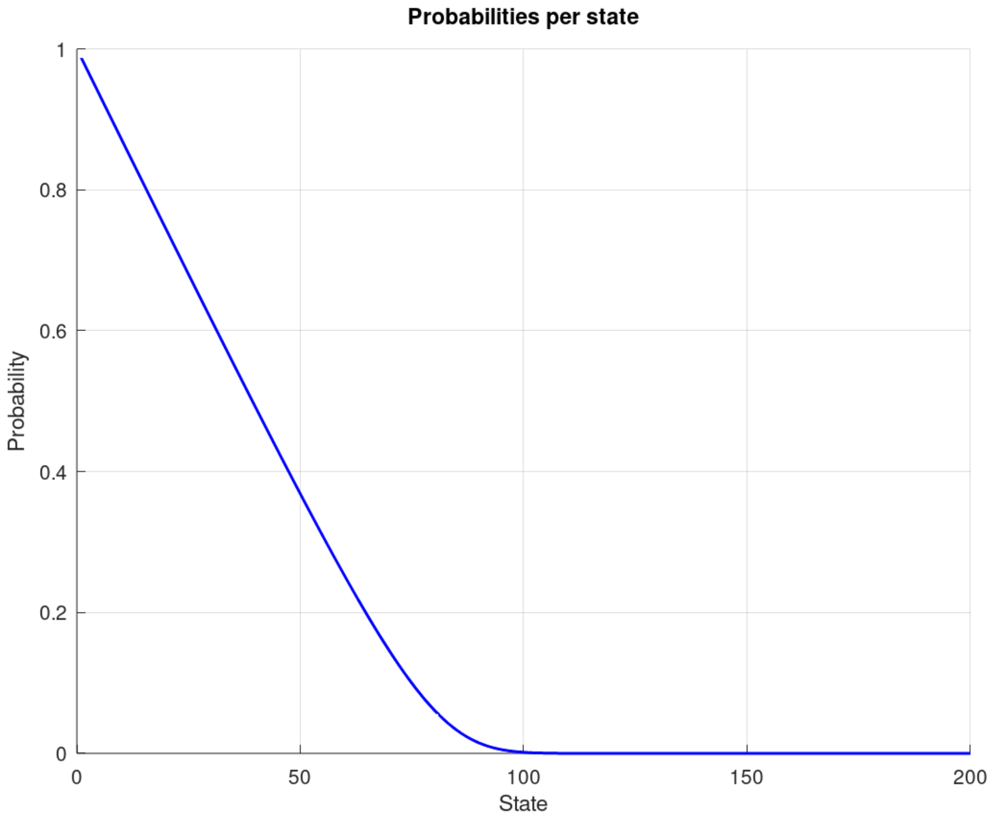
We notice that the factorial is using very large numbers like 1024^{1024} and 1024! which cannot handle. So we take aw result NaN.

2.4

2.4.1 α

Using as prototype the most demanding customer then $\rho = 200\frac{23}{60} \Rightarrow \rho = 76.67 Erlangs$

2.4.2 β



2.5 γ

We calculated using octave that the minimum lines we need is 94.

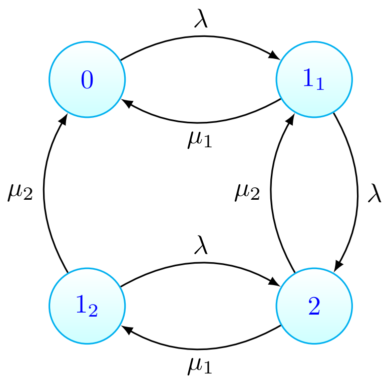
2.6 Code

```
pkg load queueing
addpath(pwd);
function Result = erlang_iterative(ro,n)
    i = 0;
    Result = 1;
    while i <= n
        Result = ro * Result/(ro*Result + i);
        i = 1+i;
    endwhile
endfunction

display(erlang_iterative(1024,1024));
display(erlangb(1024,1024));

ro = 200*23/60;
```

```
c = 1:200;
for k=1:200
    erl(k) = erlang_iterative(ro,k)
endfor
figure(1);
hold on;
title("Probabilities per state")
xlabel("State")
ylabel("Probability")
plot(c, erl, "b", "linewidth", 1.5);
grid on;
saveas (1, "figures/figureIterative.png")
hold off;
P=1;
lines = 0;
while P>0.01
    P = erlang_iterative(ro,lines);
    lines++;
endwhile
display(lines);
clc;
clear all;
close all;
exit;
```

3 Customer service with 2 diferent servers

3.1



So,

$$\lambda P_0 = \mu_1 P_{1_1} + \mu_2 P_{1_2} \Rightarrow P_0 = 0.8 P_{1_1} + 0.4 P_{1_2}$$

$$\mu_2 P_2 + \mu_1 P_2 = \lambda P_{1_1} + \lambda P_{1_2} \Rightarrow P_2 = \frac{5}{6}(P_{1_1} + P_{1_2})$$

$$\mu_1 P_{1_1} + \lambda P_{1_1} = \lambda P_0 + \mu_2 P_2 \Rightarrow P_{1_1} = \frac{5}{9} P_0 + \frac{2}{9} P_2$$

$$\mu_2 P_{1_2} + \lambda P_{1_2} = \mu_1 P_2 \Rightarrow P_{1_2} = \frac{4}{7} P_2$$

Hence,

$$P_{1_1} = 0.85938 P_0$$
$$P_{1_2} = 0.78125 P_0$$
$$P_2 = 1.3672 P_0$$

and,

$$P_0 + P_{1_1} + P_{1_2} + P_2 = 1 \Rightarrow P_0 = 0.24951$$
$$P_{1_1} = 0.21443$$
$$P_{1_2} = 0.19493$$
$$P_2 = 0.34113 = P_{blocking}$$

As for the average number of customers in the system we have,

$$E[n(t)] = \sum_{k=0}^{2} k P_k = 0 \cdot P_0 + 1(P_{1_1} + P_{1_2}) + 2 \cdot P_2 = 1.0916$$

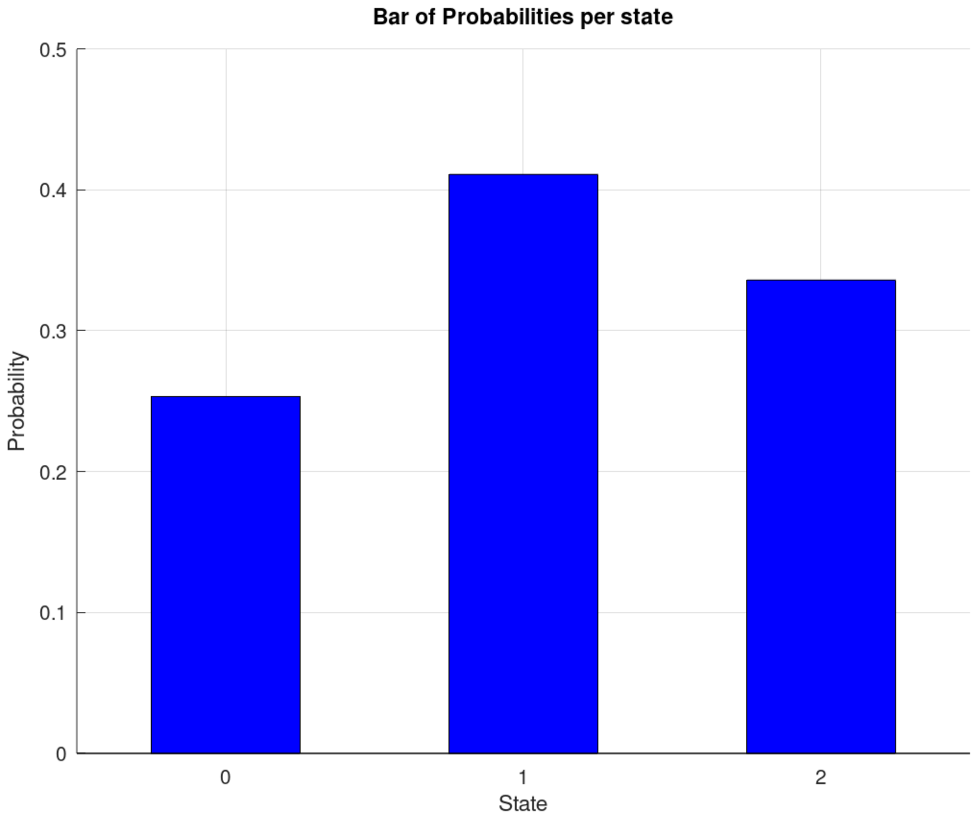
3.2

3.2.1 Results

3.2.2 Threshold

```
threshold_1a = lambda/(lambda +m1);
threshold_1b = lambda / (lambda +m2);
threshold_2_first = lambda/ (lambda + m1 + m2);
threshold_2_second = (lambda + m1) / (lambda + m1 + m2);
```

3.2.3 Figure



3.3 Code

```
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 = (lambda + m1) / (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
      current_state = 3;
      arrivals(2) = arrivals(2) + 1;
      total_arrivals = total_arrivals + 1;
    else
      current_state = 0;
    endif
  elseif current_state == 2
    if random_number < threshold_1b
      current_state = 3;
      arrivals(3) = arrivals(3) + 1;
      total_arrivals = total_arrivals + 1;
    else
      current_state = 0;
    endif
  else
      if random_number < threshold_2_first
        arrivals(4) = arrivals(4) + 1;
        total_arrivals = total_arrivals + 1;
      elseif random_number < threshold_2_second
        current_state = 2;
      else
        current_state = 1;
      endif
   endif

endwhile

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

```
display(P(4));

figure(1);
hold on;
title("Bar of Probabilities per state")
xlabel("State")
ylabel("Probability")
bar([0,1,2], [P(1),P(2)+P(3),P(4)], "b", 0.5);
xticks ([0,1,2]);
grid on;
saveas (1, "figures/figureCallingCenter.png")
hold off;

clc;
clear all;
close all;
exit;
```