Modelowanie systemów liczących. Ćwiczenie 3.

1. System M/M/1

W systemie M/M/1 intensywność strumienia wejściowego i intensywność obsługi są niezależne od stanu systemu, czyli

$$\lambda(n) = \lambda \qquad n = 0, 1, 2, \ldots \tag{1}$$

$$\mu(n) = \mu = r/w \quad (r(n) = r) \quad n = 1, 2, 3, \ldots \tag{2}$$

gdzie r jest prędkością układu obsługującego, w jest średnią rozkładu wykładniczego czasu obsługi. Dla takich parametrów rozkład prawdopodobieństw stacjonarnych jest dany wzorami

$$p(n) - p(0)(\frac{\lambda}{\mu})^n \qquad n = 0, 1, 2, \ldots \tag{3}$$

$$p(0) = (\sum_{n=0}^{\infty}(\frac{\lambda}{\mu})^n)^{-1} \tag{4}$$

Szereg we wzorze (4) jest zbieżny dla $\lambda/\mu < 1$. Wtedy

$$p(0) = 1 - \frac{\lambda}{\mu} \tag{5}$$

Wykorzystanie stacji obsługi jest równe

$$U = 1 - p(0) = 1 - (1 - \frac{\lambda}{\mu}) = \frac{\lambda}{\mu} \tag{6}$$

Przekształcając wzór (3) i (5) otrzymuje się

$$p(n) = (1 - U)U^n \qquad n = 0, 1, 2, \ldots \tag{7}$$

Srednia długość kolejki, L, można obliczyć bezpośrednio z definicji

$$L = \sum_{n=0}^{\infty} n \cdot p(n) = \sum_{n=1}^{\infty} n(1 - U)U^n = (1 - U)U \sum_{n=1}^{\infty} n \cdot U^{n-1} \tag{8}$$

Ponieważ $\sum_{n=1}^{\infty} n \cdot U^{n-1} = 1/(1-U)^2$ więc otrzymuje się

$$L = U/(1-U) \tag{9}$$

Ze wzoru Little'a średni czas obrotu wynosi

$$R = \frac{L}{\lambda} = \frac{1/\mu}{1-U} \tag{10}$$

2. Symulacja systemu M/M/1

```
global Q_LIMIT mean_interarrival mean_service mean_delays_required
global next_event_type num_custs_delayed num_delays_required
            num_events num_in_q server_status
global area_num_in_q  area_server_status mean_interarrival
            mean_service time time_arrival
            time_last_event time_next_event total_of_delays

% Input Parameters

Q_LIMIT = 200;
mean_interarrival = 1.0;
mean_service = 0.5;
mean_delays_required = 100;
area_server_status=0;
num_events = 2

INIT;     % calls initialization routine

% run the simulation while more delays are still required

while(num_custs_delayed < mean_delays_required)
 TIMING;
 UPDATE;

 if(next_event_type==1)
  ARRIVE;   % branch to arrival routine (function)
 elseif(next_event_type==2)
  DEPART;    % branch to departure routine (function)
```

```
    end
   end

   REPORT   % call report generator function

  function INIT
  global Q_LIMIT mean_interarrival mean_service
   mean_delays_required
  global next_event_type num_custs_delayed
   num_delays_required num_events num_in_q server_status
  global area_num_in_q area_server_status mean_interarrival mean_service
   time time_arrival time_last_event time_next_event total_of_delays

  time=0.0;
  server_status=0;
  num_in_q = 0;

  time_last_event = 0.0;
  num_custs_delayed = 0;
  total_of_delays = 0.0;

  area_num_in_q = 0.0;
  area_erve_status = 0.0;

  time_next_event(1) = time+expon(mean_interarrival)
  time_next_event(2) = 1.0 * exp(30)
  %disp(['init'])

  function ARRIVE

  global Q_LIMIT mean_interarrival mean_service mean_delays_required
  global next_event_type num_custs_delayed num_delays_required num_events num_in_q
server_status
  global area_num_in_q area_server_status mean_interarrival mean_service time
time_arrival
   time_last_event time_next_event total_of_delays
  time_next_event(1) = time + expon(mean_interarrival);

  % time
```

```
 % check to see whether server is busy

 if(server_status == 1) % server is busy

  num_in_q = num_in_q + 1; % increase the no. of customers in queue

  if(num_in_q) > Q_LIMIT)
   disp(['num_in_q = ', num2str(num_in_q)]);
   disp(['Overflow of the array of time_arrival at ', num2str(time)]);
   pause
  end
  time_arrival(num_in_q) = time;
 else % server is idle
 delay = 0.0;
  total_of_delays = total_of_delays + delay;
  num_custs_delayed = num_custs_delayed + 1;
  server_status = 1;
  time_next_event(2) = time + expon(mean_service);
 end

 function DEPART
 global Q_LIMIT mean_interarrival mean_service mean_delays_required
 global next_event_type num_custs_delayed num_delays_required num_events num_in_q
server_status
 global area_num_in_q area_server_status mean_interaarival mean_service time
time_arrival
  time_last_event time_next_event total_of_delays

  if(num_in_q == 0)
  % queue is empty. Make the server idle and eliminate the departure event from
consideration

  server_status = 0;   % idle

  time_next_event(2) = 1.0 * exp(30);

 else
 % queue is not empty, so decrease the no. of customers in queue
  num_in_q = num_in_q - 1;
  delay = time - time_arrival(1);
```

```
  total_of_delays = total_of_delays + delay;
  num_custs_delays = num_custs_delays + 1;
  time_next_event(2) = time + expon(mean_service);

  % move each customers's arrival time in queue up one place
  for i = 1:num_in_q
   time_arrival(i)=time_arrival(i+1);
  end
 end

 function e=EXPON(mean)
 u = rand(1);
 e = - mean * log(u);

 % uses the inverse transformation method to generate negativeexponential ran-
dom numbers

 unction TIMING
 global Q_LIMIT mean_interarrival mean_service mean_delays_required
 global next_event_type num_custs_delayed num_delays_required num_events num_in_q
server_status
 global area_num_in_q area_server_status mean_service time time_arrival time_last_event
  time_next_event total_of_delays
  min_time_next_event = 1.0*exp(29);
  next_event_type = 3;
  % determine the event type of the next event to occur
  for i=1:num_events
  if(time_next_event(i) < min_time_next_event)
   min_time_next_event = time_next_event(i);
   next_event_type = i;
  end;
 end

 if(next_event_type == 3)
  disp(['Event List Empty at Time ', num2str(time)]);
 end

 time = min_time_next_event;

 function UPDATE
```

```
  % Update area accumulated for time-average statistics
  global Q_LIMIT mean_interarrival mean_service mean_delays_required
  global next_event_type num_custs_delayed num_delays_required num_events num_in_q
server_status
  global area_num_in_q area_server_status_ mean_interarrival
    mean_service time time_arrival time_last_event time_next_event total_of_delays

    time_since_last_event = time - time_last_event;
    time_last_event = time;
    area_num_in_q = area_num_in_q + num_in_q * time_since_last_event;
    area_server_status = area_server_status + server_status * time_since_last_event;

  function REPORT
  global Q_LIMIT mean_interarrival mean_service mean_delays_required
  global next_event_type num_custs_delayed num_delays_required num_events num_in_q
    server_status
  global area_num_in_q area_server_status mean_interarrival mean_service time
    time_arrival time_last_event time_next_event total_of_delays

  disp(['Average Delay in Queue : ', num2str(total_of_delays/num_custs_delayed),'
minutes']);
  disp(['Average number in Queue : ', num2str(area_num_in_q/time)])
  disp(['Service Utilization : ', num2str(area_server_status/time)]);
  disp(['Time Simulation Ended : ', num2str(time)]);
```

3. Przebieg ćwiczenia

Dla przyjętych parametrów systemu: *wykorzystanie* (ang. *utilization*) $\rho = 0.545$, *średniego czasu oczekiwania* (ang. *waiting time*) równym 15.04 godziny znajdź korzystając z systemu M/M/1 średnią długość kolejki. Wyniki powinny być uzyskane zarówno metodami analitycznymi, jak i symulacyjnymi.