Introduction to MATLAB

General Applications

Drs. Trani and Rakha

Civil and Environmental Engineering

Virginia Polytechnic Institute and State University

Spring 2000

Sample Applications

The following examples constitute typical applications of Matlab scripts and functions used in Transportation Engineering.

The following examples are covered:

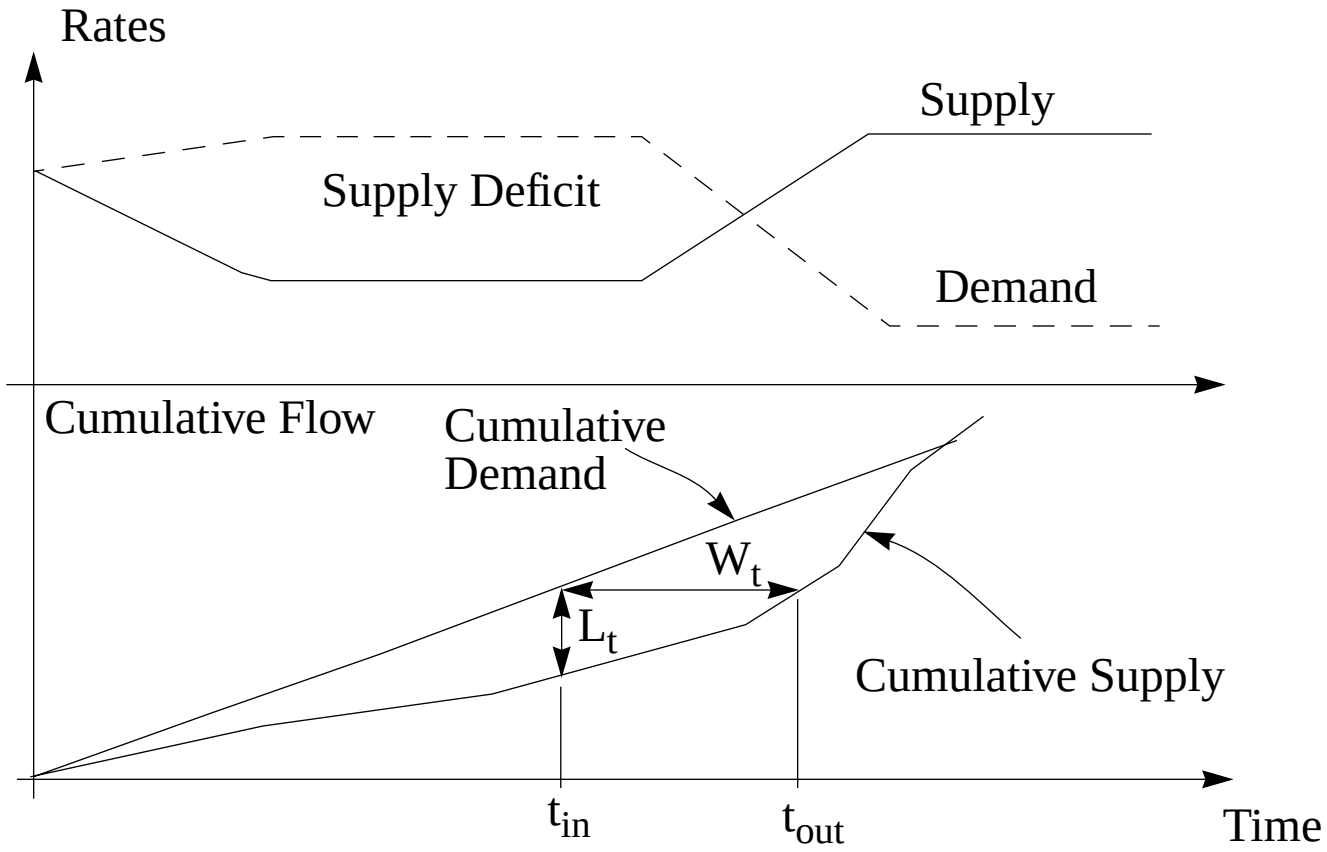
- Deterministic queueing model
- Stochastic queueing model
- Simulation model of an M/M/1 system

Deterministic Queues

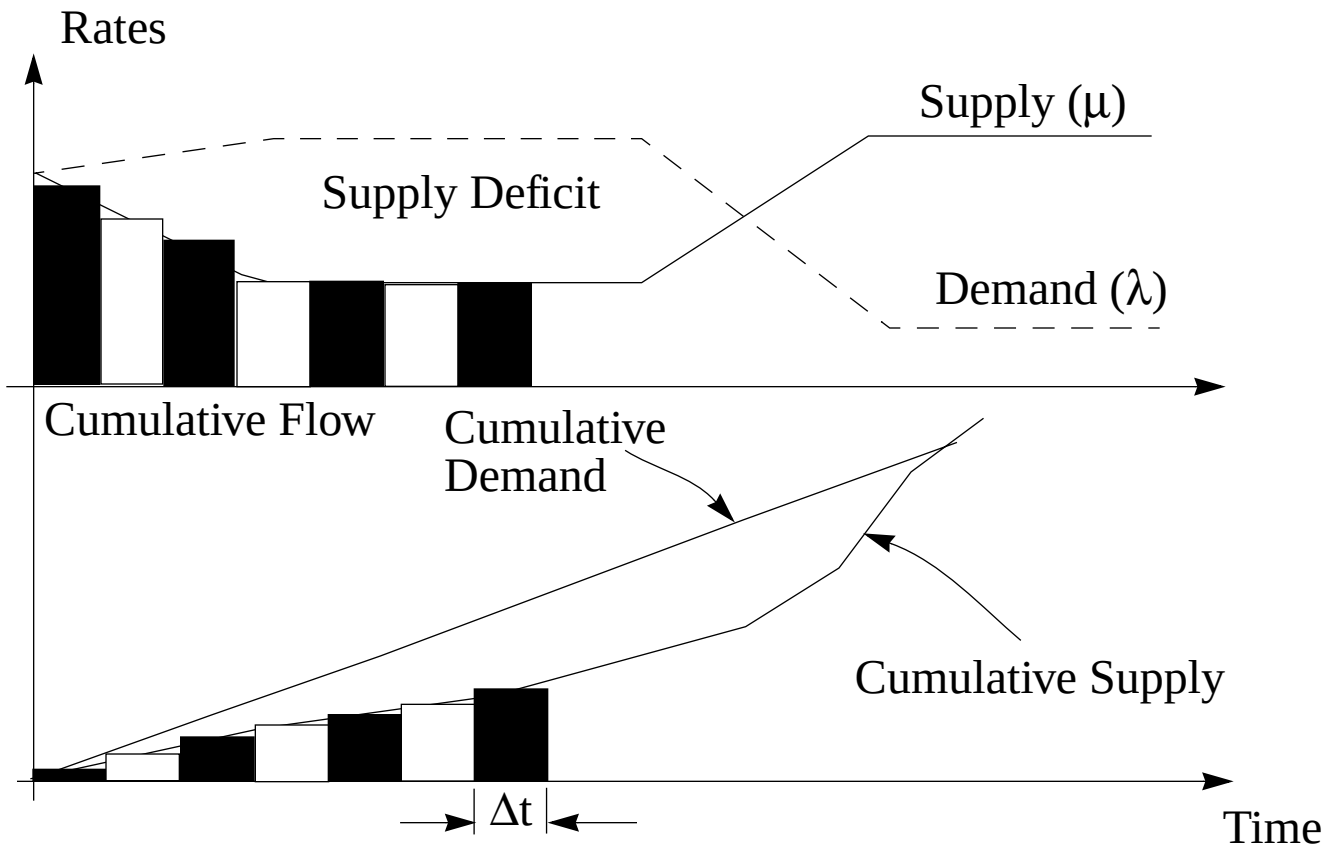
Deterministic Queues are analogous to a continuous flow of entities passing over a point over time. As Morlok [Morlok, 1976] points out this type of analysis is usually carried out when the number of entities to be simulated is large as this will ensure a better match between the resulting cumulative stepped line representing the state of the system and the continuous approximation line

The figure below depicts graphically a deterministic queue characterized by a region where demand exceeds supply for a period of time

Deterministic Queues (Continuous)



Deterministic Queues (Discrete Case)



Deterministic Queues (Parameters)

a) The queue length, L_t, (i.e., state of the system) corresponds to the maximum ordinate distance between the cumulative demand and supply curves

b) The waiting time, W_t, denoted by the horizontal distance between the two cumulative curves in the diagram is the individual waiting time of an entity arriving to the queue at time t_{in}

c) The total delay is the area under bounded by these two curves

d) The average delay time is the quotient of the total delay and the number of entities processed

Deterministic Queues

e) The average queue length is the quotient of the total delay and the time span of the delay (i.e., the time difference between the end and start of the delay)

Assumptions

Demand and supply curves are derived derived from known flow rate functions (λ and μ) which of course are functions of time.

The diagrams shown represent a simplified scenario arising in many practical situations such as those encountered in traffic engineering (i.e., bottleneck analysis).

Remarks About Deterministic Queues

- Introducing some time variations in the system we can easily grasp the benefit of the simulation
- Most of the queueing processes at transportation terminals are non-steady thus analytic models seldom apply
- Data typically exist on passenger behaviors over time that can be used to feed these deterministic, non-steady models
- The capacity function is perhaps the most difficult to quantify because human performance is affected by the state of the system (i.e., queue length among others)

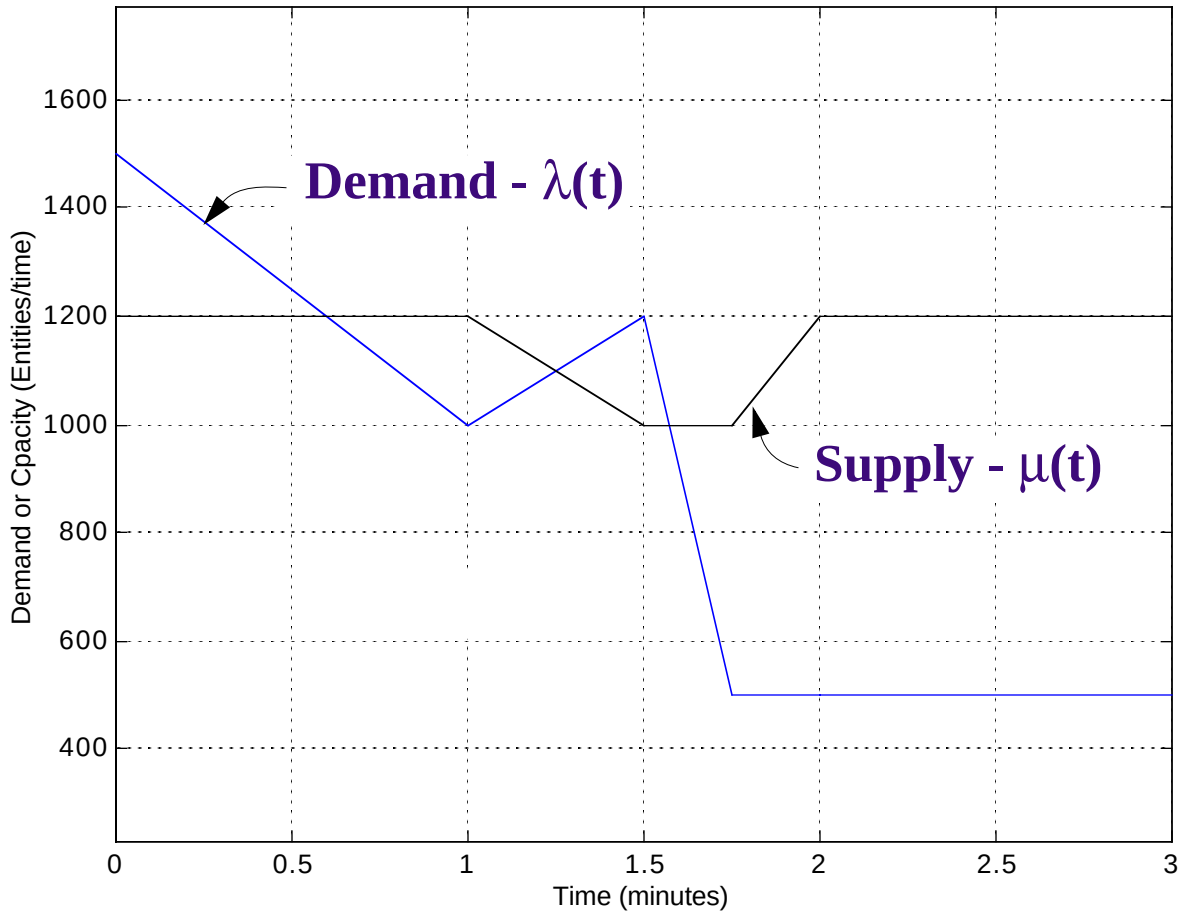
Example: Deterministic Queueing Model of the Immigration Area at an Airport

Let us define a demand function that varies with time representing the typical cycles of operation observed at airport terminals. This demand function, $\lambda(t)$ is:

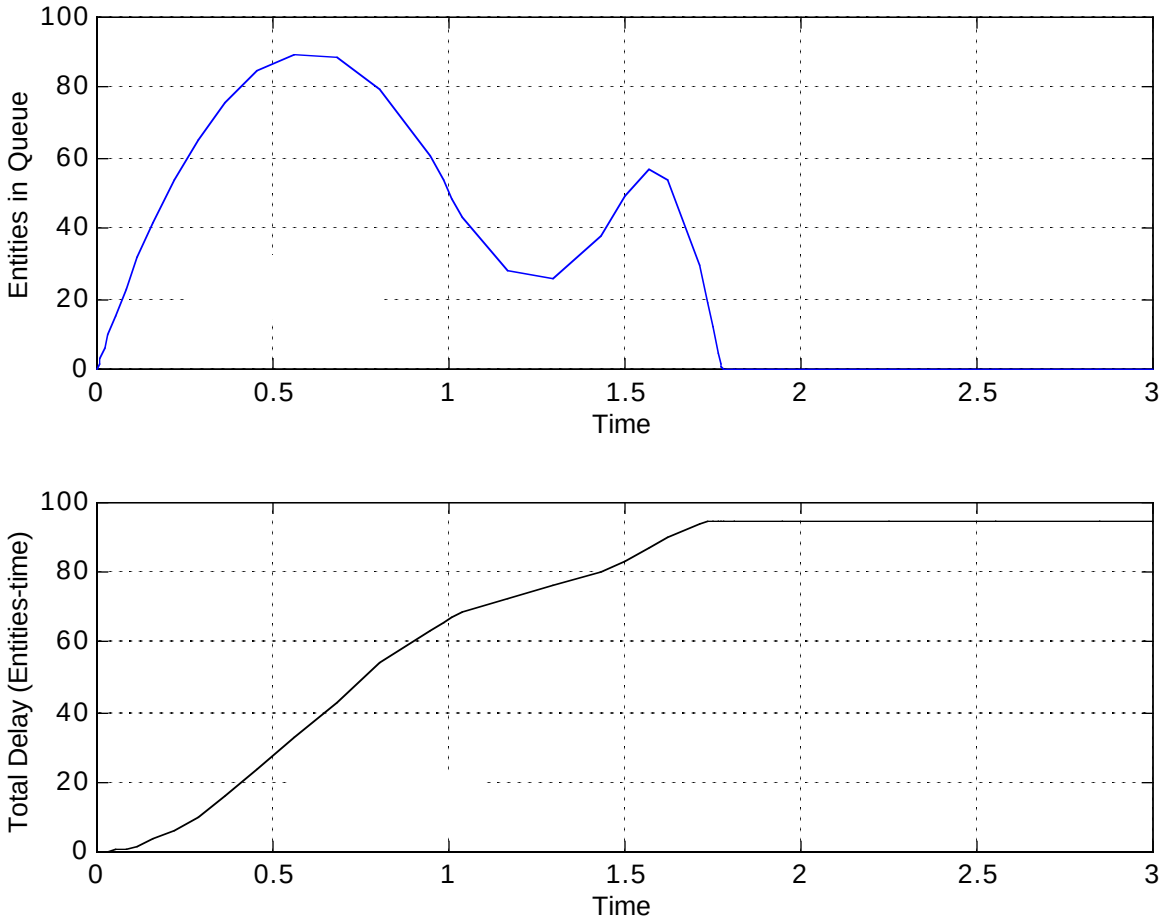
- Deterministic
- Observed or predicted
- A function of time (a table function)

Suppose the capacity of the system, $\mu(t)$, is also known and deterministic as shown in the following Matlab code

Continuous Simulation Example (Rates)



Plots of Integrals of $\lambda(t)$ and $\mu(t)$



Matlab Source Code for Deterministic Queueing Model (main file)

```
% Deterministic queueing simulation
% T. Trani (Rev. Mar 99)
global demand capacity time

% Enter demand function as an array of values over time

% general demand - capacity relationships
%

demand = [1500 1000 1200 500 500 500];
capacity = [1200 1200 1000 1000 1200 1200];
time = [0.00  1.00  1.500  1.75  2.00  3.00];

% Compute min and maximum values for proper scaling in plots
mintime                                                    = min(time);
maxtime                                                    = max(time);
```

```
maxd                                    = max(demand);
maxc                                    = max(capacity);
mind                                    = min(demand);
minc                                    = min(capacity);
scale                                   = round(.2 *(maxc+maxd)/
2)
minplot = min(minc,mind) - scale;
maxplot                                 = max(maxc,maxd) +
scale;

po = [0 0];                             % intial number of
passengers
to = mintime;
tf = maxtime;
tspan = [to tf];

% where:
% to is the initial time to solve this equation
% tf is the final time
% tspan is the time span to solve the simulation
```

```
[t,p] = ode23('fqueue_2',tspan,po);

% Compute statistics

Ltmax                                   = max(p(:,1));
tdelay                                  = max(p(:,2));
a_demand                                = mean(demand);
a_capacity                              = mean(capacity);

  clc
  disp([blanks(5),'Deterministic Queueing Model '])
  disp(' ')
  disp(' ')
  disp([blanks(5),'  Average arrival rate (entities/time) = ',
num2str(a_demand)])
  disp([blanks(5),'  Average capacity (entities/time) = ', num2str(a_capacity)])
  disp([blanks(5),'  Simulation Period (time units) = ', num2str(maxtime)])
```

```
  disp(' ')

  disp(' ')
  disp([blanks(5),'  Total delay (entities-time) = ', num2str(tdelay)])
  disp([blanks(5),'  Max queue length (entities) = ', num2str(Ltmax)])
  disp(' ')

pause

% Plot the demand and supply functions

plot(time,demand,'b',time,capacity,'k')
xlabel('Time (minutes)')
ylabel('Demand or Cpacity (Entities/time)')
axis([mintime maxtime minplot maxplot])
grid

pause
% Plot the results of the numerical integration procedure
```

```
subplot(2,1,1)
plot(t,p(:,1),'b')
xlabel('Time')
ylabel('Entities in Queue')
grid

subplot(2,1,2)
plot(t,p(:,2),'k')
xlabel('Time')
ylabel('Total Delay (Entities-time)')
grid
```

Matlab Source Code for Deterministic Queueing Model (function file)

```
% Function file to integrate numerically a differential equation
% describing a deterministic queueing system

function pprime = fqueue_2(t,p)
global demand capacity time

% Define the rate equations
demand_table = interp1(time,demand,t);
capacity_table = interp1(time,capacity,t);

if (demand_table < capacity_table) & (p > 0)
    pprime(1) = demand_table - capacity_table;    % rate of change in state
variable
elseif demand_table > capacity_table
    pprime(1) = demand_table - capacity_table;    % rate of change in state
variable
```

```
else
        pprime(1) = 0.0; % avoids accumulation of entities
end

pprime(2) = p(1);                                   % integrates the delay
curve over time
pprime = pprime';
```

Output of Deterministic Queueing Model

Deterministic Queueing Model

Average arrival rate (entities/time) = 866.6667
Average capacity (entities/time) = 1133.3333
Simulation Period (time units) = 3

Total delay (entities-time) = 94.8925
Max queue length (entities) = 89.6247

Stochastic Queueing Systems Nomenclature

The idea behind the stochastic queueing process is to analyze steady-state conditions. Lets define some notation applicable for steady-state conditions,

N = No. of customers in queueing system

P_n = Prob. of exactly n customers are in queueing system

L = Expected no. of customers in queueing system

L_q = Queue length (expected)

W = Waiting time in system (includes service time)

W_q = Waiting time in queue

Stochastic Queueing Systems

There are some basic relationships that have beed derived in standard textbooks in operations research [Hillier and Lieberman, 1991]. Some of these more basic relationships are:

$L = \lambda W$

$Lq = \lambda Wq$

Multiple Server

Infinite source (constant λ and μ)

Assumptions:

a) Probability between arrivals is negative exponential with parameter λ_n

b) Probability between service completions is negative exponential with parameter μ_n

c) Only one arrival or service occurs at a given time

Multiple Server - Infinite Source (constant λ , μ)

$\rho = \lambda / s\mu$ utilization factor of the facility

$$P_0 = 1 \Big/ \left(\sum_{n=0}^{s-1} \frac{(\lambda/\mu)^n}{n!} + \frac{(\lambda/\mu)^s}{s!} \left(\frac{1}{1-(\lambda/s\mu)} \right) \right)$$

idle probability

$$P_n = \begin{cases} \dfrac{(\lambda/\mu)^n}{n!} P_0 & 0 \le n \le s \\ \dfrac{(\lambda/\mu)^n}{s!\, s^{n-s}} P_0 & n \ge s \end{cases}$$

probability of n entities in the system

$$L = \frac{\rho P_0 \left(\frac{\lambda}{\mu}\right)^s}{s!(1-\rho)^2} + \frac{\lambda}{\mu}$$

expected number of entities in system

$$L_q = \frac{\rho P_0 \left(\frac{\lambda}{\mu}\right)^s}{s!(1-\rho)^2}$$ expected number of entities in queue

$$W_q = \frac{L_q}{\lambda}$$ average waiting time in queue

$$W = \frac{L}{\lambda} = W_q + \frac{1}{\lambda}$$ average waiting time in system

Finally the probability distribution of waiting times is,

$$P(W > t) = e^{-\mu t}\left[1 + \frac{P_0\left(\frac{\lambda}{\mu}\right)^s}{s!(1-\rho)}\left(\frac{1 - e^{-\mu t(s-1-\lambda/\mu)}}{s-1-\lambda/\mu}\right)\right]$$

if $s - 1 - \lambda/\mu = 0$ then use

$$\frac{1 - e^{-\mu t(s-1-\lambda/\mu)}}{s-1-\lambda/\mu} = \mu t$$

Example A2: Level of Service at Security Checkpoints

The airport shown in the next figures has two security checkpoints for all passengers boarding aircraft. Each security check point has two x-ray machines. A survey reveals that on the average a passenger takes 45 seconds to go through the system (negative exponential distribution service time).

The **arrival rate** is known to be random (this equates to a Poisson distribution) with a mean arrival rate of one passenger every 25 seconds.

In the design year (2010) the demand for services is expected to grow by 60% compared to that today.

Relevant Operational Questions

a) What is the current utilization of the queueing system (i.e., two x-ray machines)?

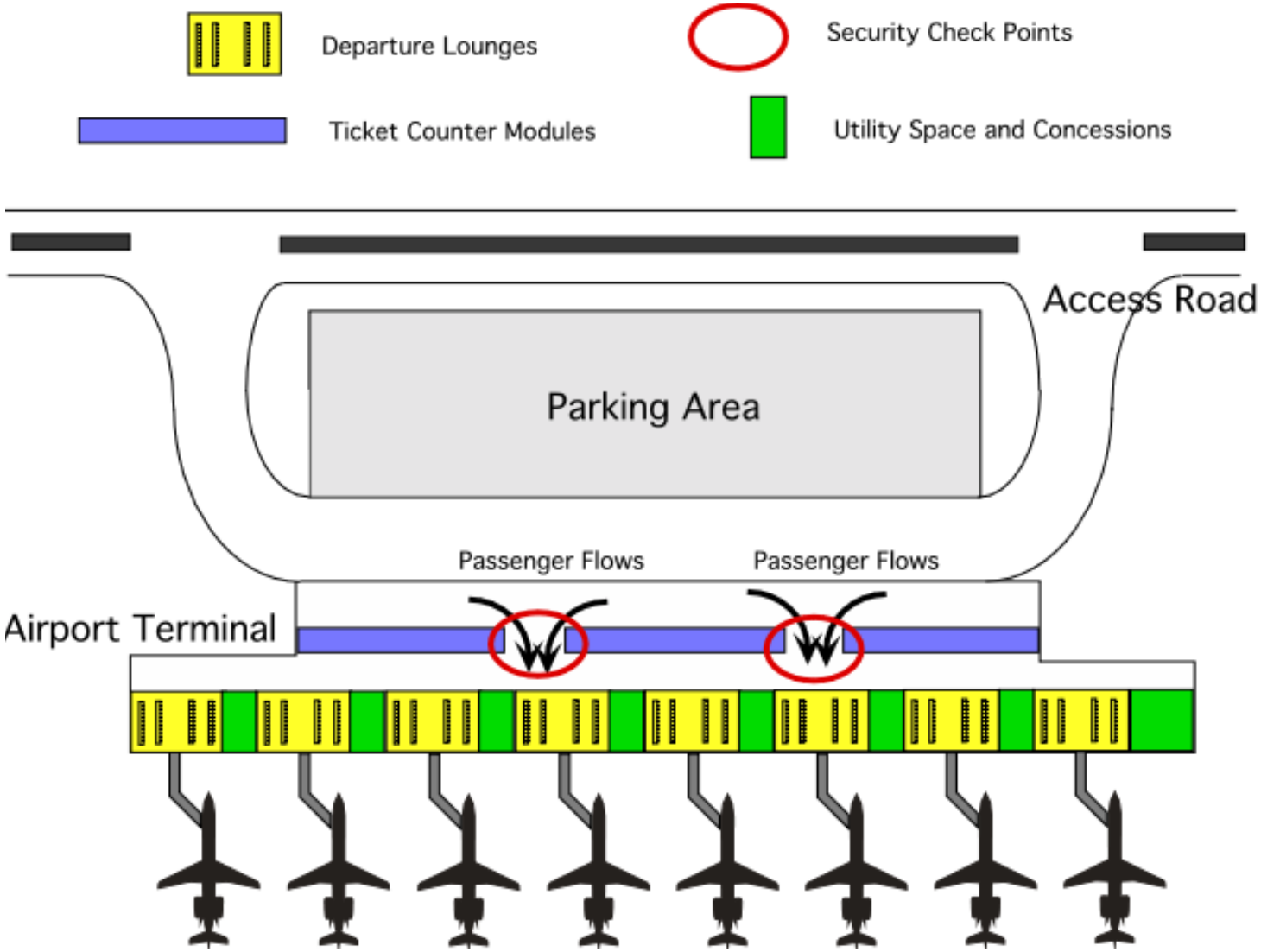
b) What should be the number of x-ray machines for the design year of this terminal (year 2010) if the maximum tolerable **waiting time in the queue** is 2 minutes?

c) What is the expected number of passengers at the checkpoint area on a typical day in the design year (year 2010)?

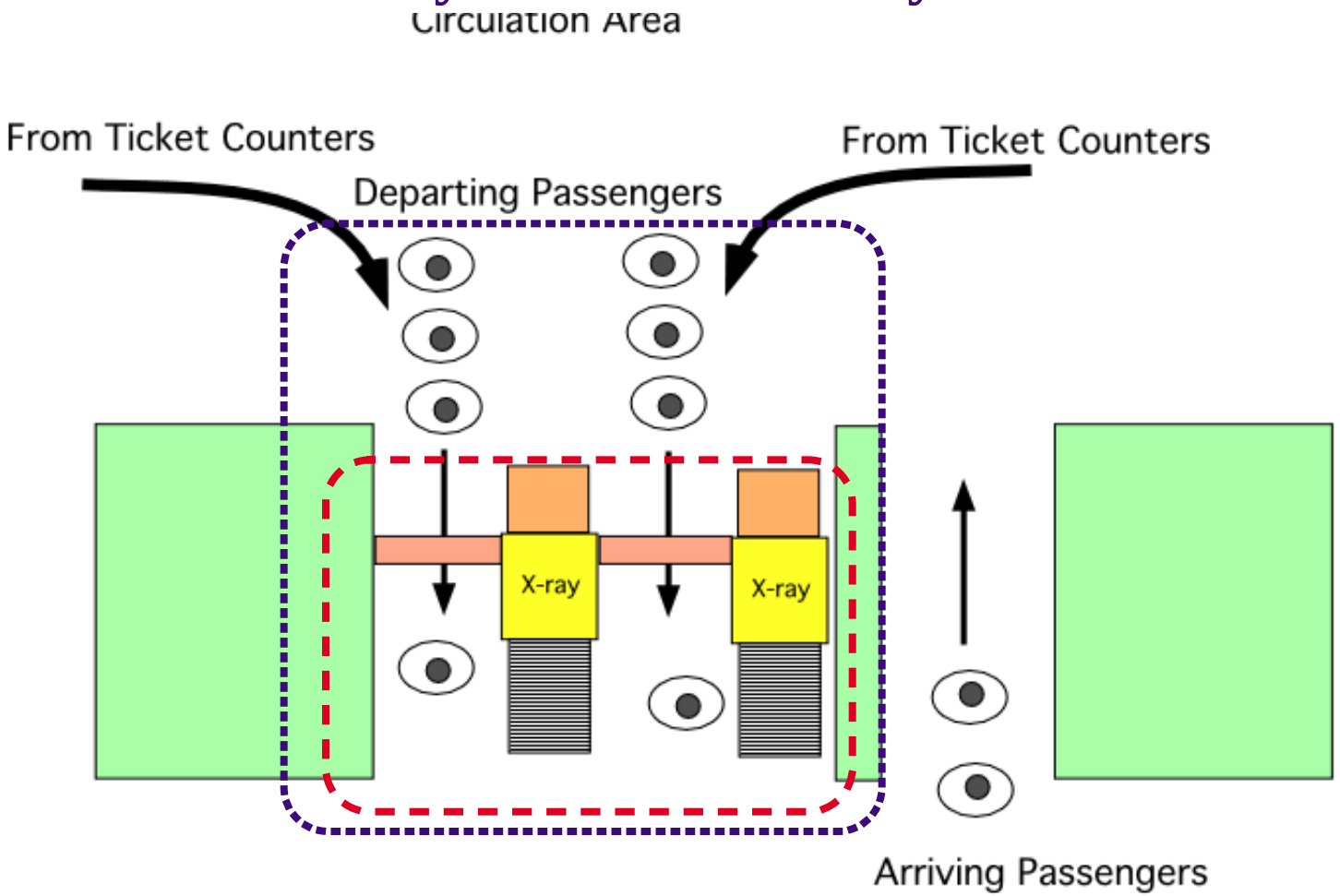
d) What is the new utilization of the future facility?

e) What is the probability that more than 4 passengers wait for service in the design year?

Airport Terminal Layout



Security Check Point Layout



Security Check Point Solutions

a) Utilization of the facility, ρ. Note that this is a multiple server case with infinite source.

$\rho = \lambda / (s\mu) = 140/(2*80) = 0.90$

Other queueing parameters are found using the steady-state equations for a multi-server queueing system with infinite population are:

Idle probability = 0.052632
Expected No. of customers in queue (Lq) = 7.6737
Expected No. of customers in system (L) = 9.4737
Average Waiting Time in Queue = 192 s
Average Waiting Time in System = 237 s

b) The solution to this part is done by trail and error (unless you have access to design charts used in queueing models. As a first trial lets assume that the number of x-ray machines is 3 (s=3).

Finding Po, $$P_0 = \sum_{n=0}^{s-1} \frac{(\lambda/\mu)^2}{n!} + \frac{(\lambda/\mu)^s}{s!}\left(\frac{1}{1-(\lambda/s\mu)}\right)$$

Po = .0097 or less than 1% of the time the facility is idle

Find the waiting time in the queue,

***Wq* = 332** s

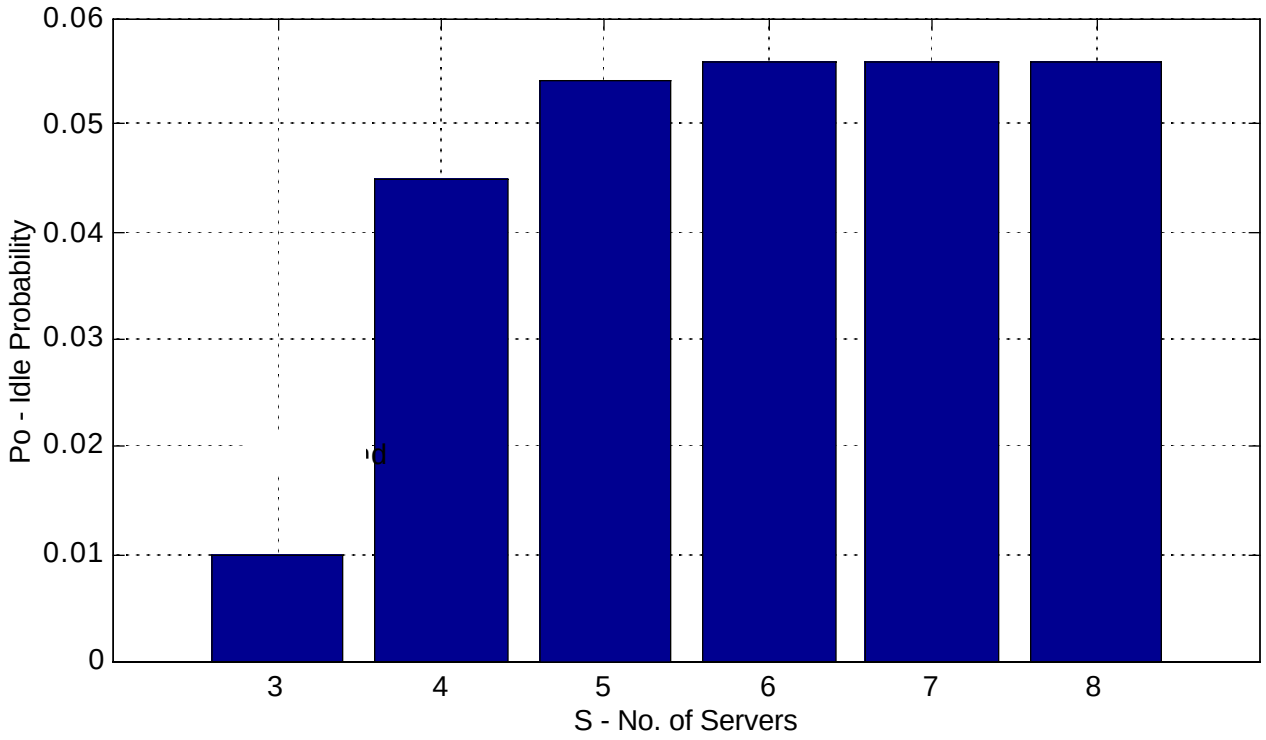
Since this waiting time violates the desired two minute maximum it is suggested that we try a higher number of x-ray machines to expedite service (at the expense of

cost). The following figure illustrates the sensitivity of P_o and L_q as the number of servers is increased.

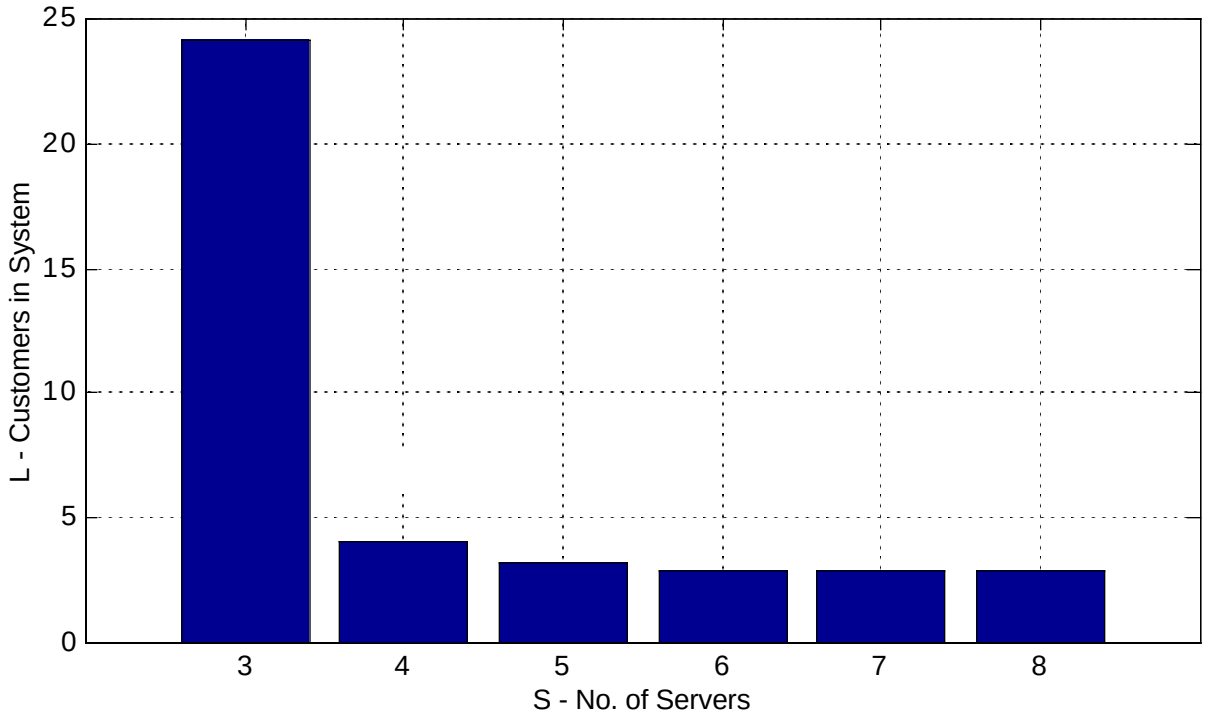
Note that four x-ray machines are needed to provide the desired average waiting time, *Wq*.

Sensitivity of *Po* with *S*

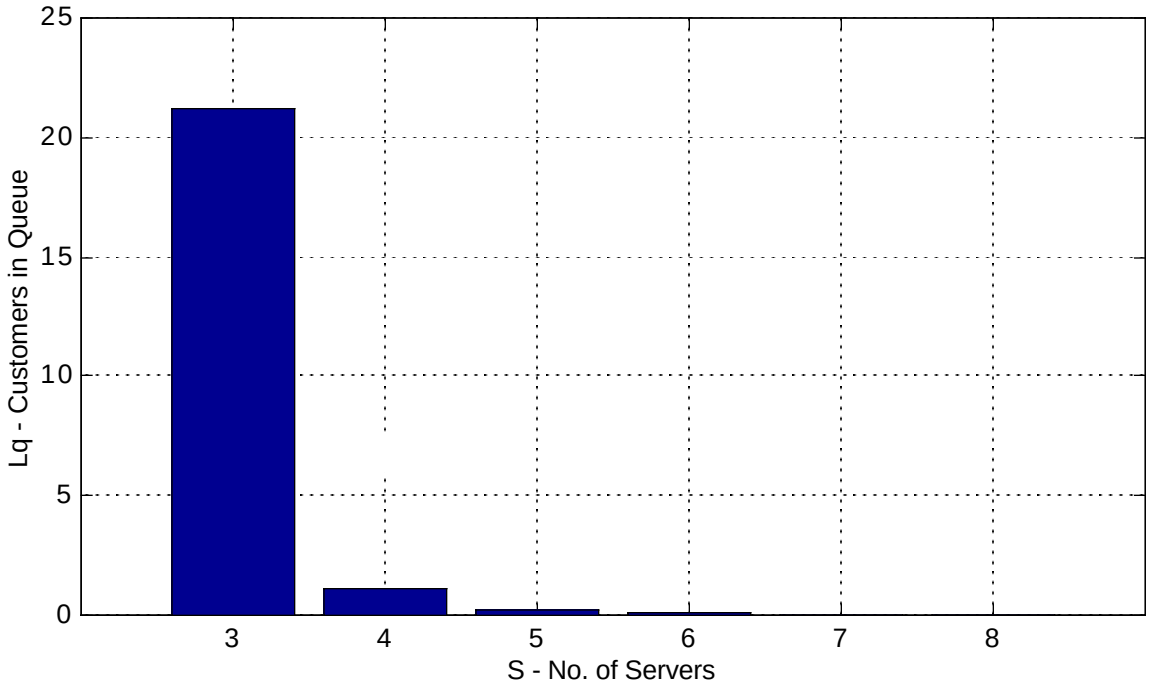
Note the quick variations in Po as S increases.



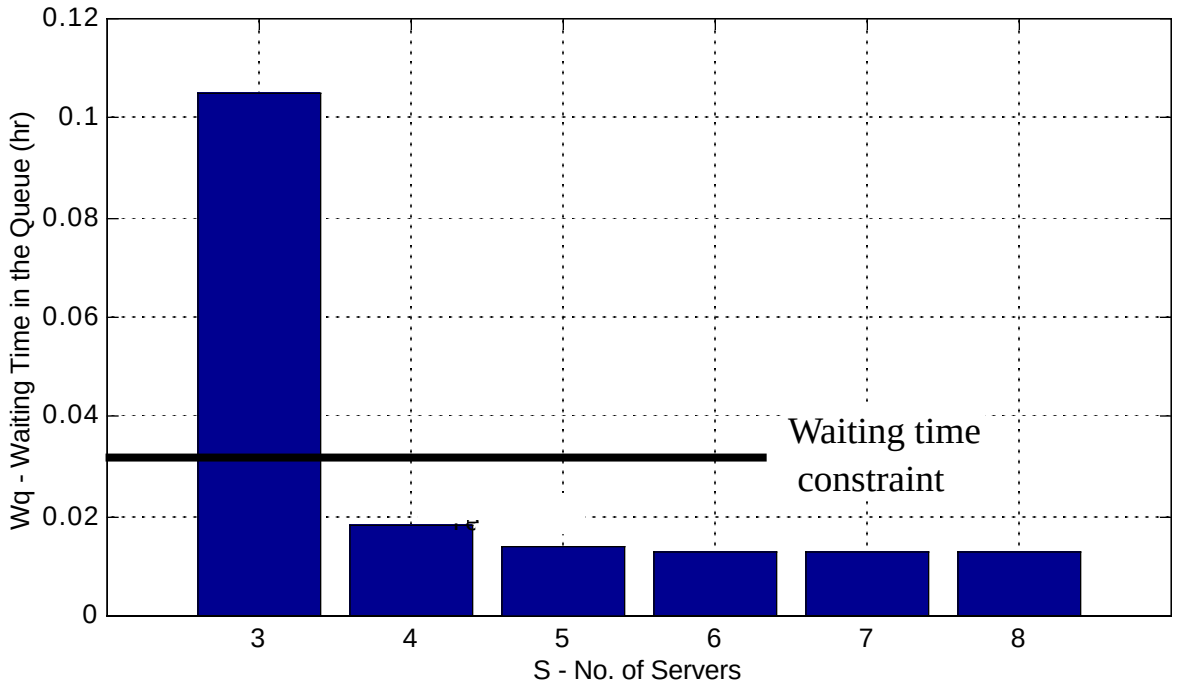
Sensitivity of *L* with *S*



Sensitivity of *Lq* with *S*



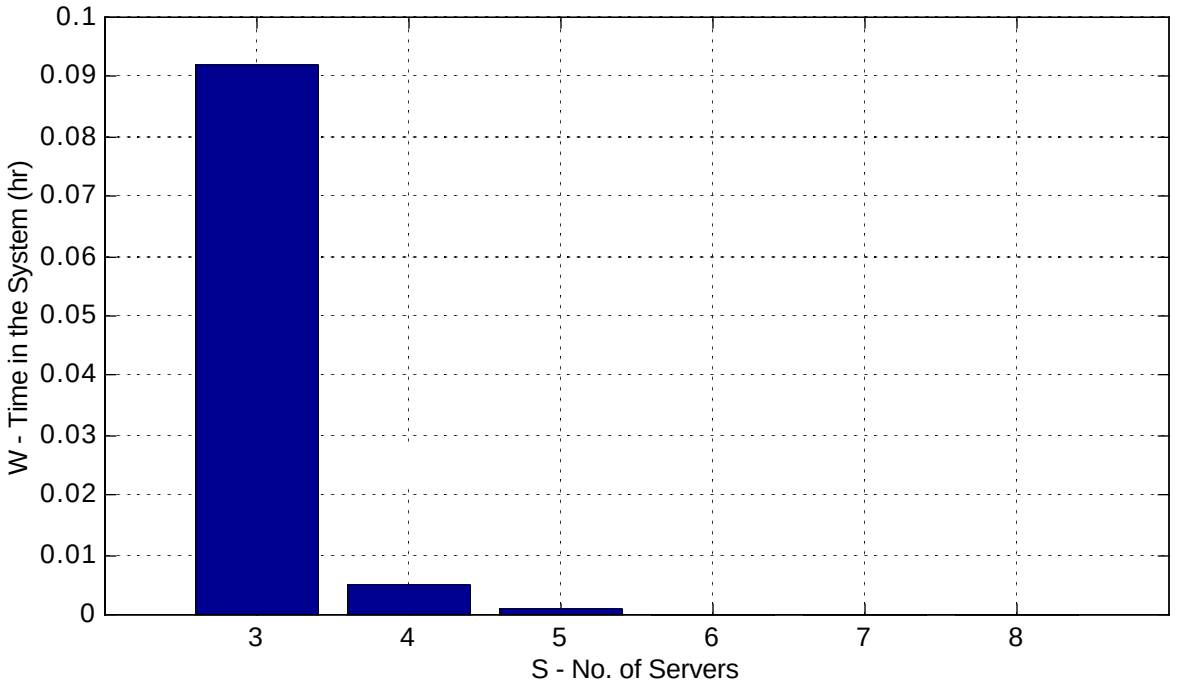
Sensitivity of *Wq* with *S*



This analysis demonstrates that 4 x-ray machines are needed to satisfy the 2-minute operational design constraint.

Sensitivity of *W* with *S*

Note how fast the waiting time function decreases with *S*



Security Check Point Results

c) The expected number of passengers in the system is (with $S = 4$),

$$L = \frac{\rho P_0 \left(\frac{\lambda}{\mu}\right)^s}{s!(1-\rho)^2} + \frac{\lambda}{\mu}$$

***L* =** 4.04 passengers in the system on the average design year day.

d) The utilization of the improved facility (i.e., four x-ray machines) is

$\rho = \lambda / (s\mu) = 230/ (4*80) =$ **0.72**

e) The probability that more than four passengers wait for service is just the probability that more than eight passengers are in the queueing system, since four are being served and more than four wait.

$$P(n > 8) = 1 - \sum_{n=0}^{8} P_n$$

where,

$$P_n = \frac{(\lambda/\mu)^n}{n!} P_0 \quad \text{if} \quad n \leq s$$

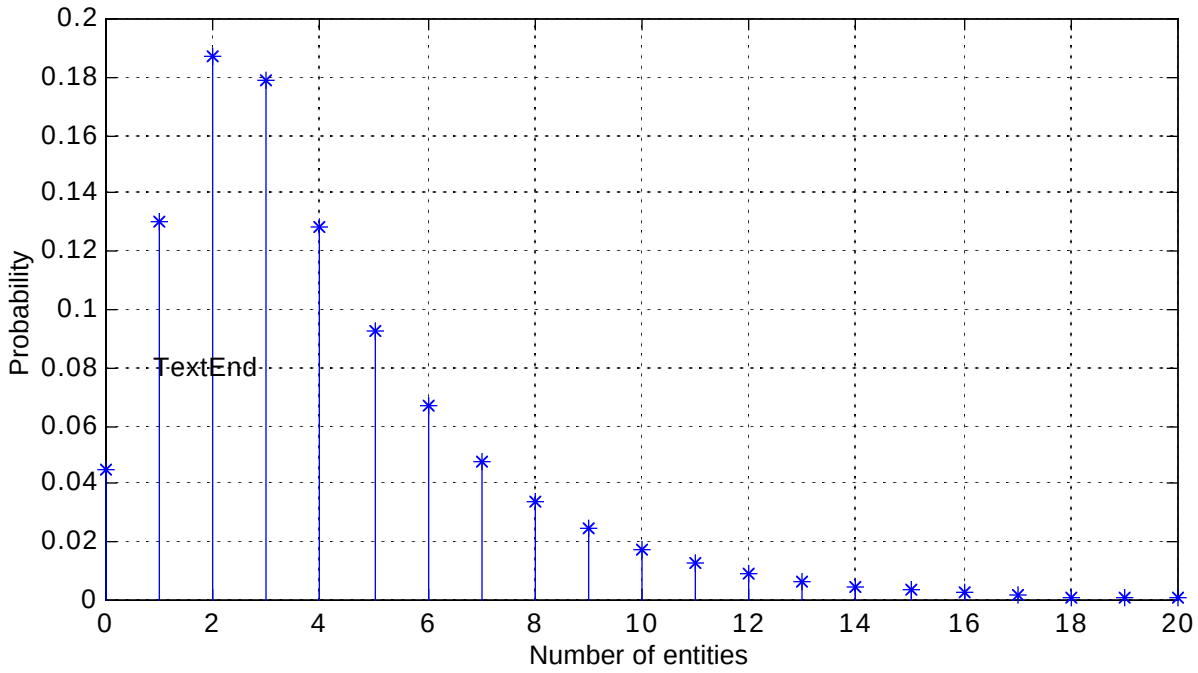
$$P_n = \frac{(\lambda/\mu)^n}{s! s^{n-s}} P_0 \quad \text{if} \quad n > s$$

from where, $P_n > 8$ is 0.0879.

Note that this probability is low and therefore the facility seems properly designed to handle the majority of the expected traffic within the two-minute waiting time constraint.

PDF of Customers in System (L)

The PDF below illustrates the stochastic process resulting from poisson arrivals and neg. exponential service times



Matlab Computer Code

```
% Multi-server queue equations with infinite population

% Sc = Number of servers
% Lambda = arrival rate
% Mu = Service rate per server
% Rho = utilization factor
% Po = Idle probability
% L = Expected no of entities in the system
% Lq = Expected no of entities in the queue
% nlast - last probability to be computed

% Initial conditions

 S=5;
 Lambda=3;
 Mu = 4/3;
```

```
  nlast = 10;                                   % last probability value
computed

  Rho=Lambda/(S*Mu);

% Find Po
    Po_inverse=0;
    sum_den=0;

  for i=0:S-1 %                                 for the first term in the
denominator (den_1)
    den_1=(Lambda/Mu)^i/fct(i);
    sum_den=sum_den+den_1;
  end

    den_2=(Lambda/Mu)^S/(fct(S)*(1-Rho)); % for the second part of den
(den_2)
    Po_inverse=sum_den+den_2;
    Po=1/Po_inverse
```

```
% Find probabilities (Pn)

Pn(1) = Po;     % Initializes the first element of Pn column vector to be Po
n(1) = 0;       % Vector to keep track of number of entities in system

% loop to compute probabilities of n entities in the system

for j=1:1:nlast
      n(j+1) = j;
      if (j) <= S
           Pn(j+1) = (Lambda/Mu)^j/fct(j) * Po;
      else
            Pn(j+1) = (Lambda/Mu)^j/(fct(S) * Sc^(j-S)) * Po;
      end
end

% Queue measures of effectiveness

 Lq=(Lambda/Mu)^S*Rho*Po/(fct(S)*(1-Rho)^2)
```

```
L=Lq+Lambda/Mu
Wq=Lq/Lambda
W = L/Lambda

plot(n,Pn)
xlabel('Number of entities')
ylabel('probability')
```

A Discrete Event Simulation Model

This example illustrates how Matlab functions can be used to model an M/M/1 queuing process using an explicit discrete event simulation model

Two modeling approaches of discrete event simulation are generally implemented in large-scale airport/airspace simulations:

a) Event-scheduling simulation

Estimation of state variables at times when each event occurs

b) Process or activity-based simulation

Description of processes (i.e., time-ordered sequences of events) to model activities “experienced” by each entity throughout the simulation

Elements of a Simulation Model (Kelton et al.)

Entities: dynamic objects that are created and destroyed as the simulation moves forward in time

Attributes: characteristics of entities used to differentiate behavior in the simulation process (i.e.,transfer vs. terminating passengers)

Global Variables: provide general parameters of a simulation model (i.e., simulation stopping criteria)

Resources: elements that are occupied in the simulation by entities for a finite period of time (i.e., ticket counter occupied by a passenger)

Elements of a Simulation Model (continuation)

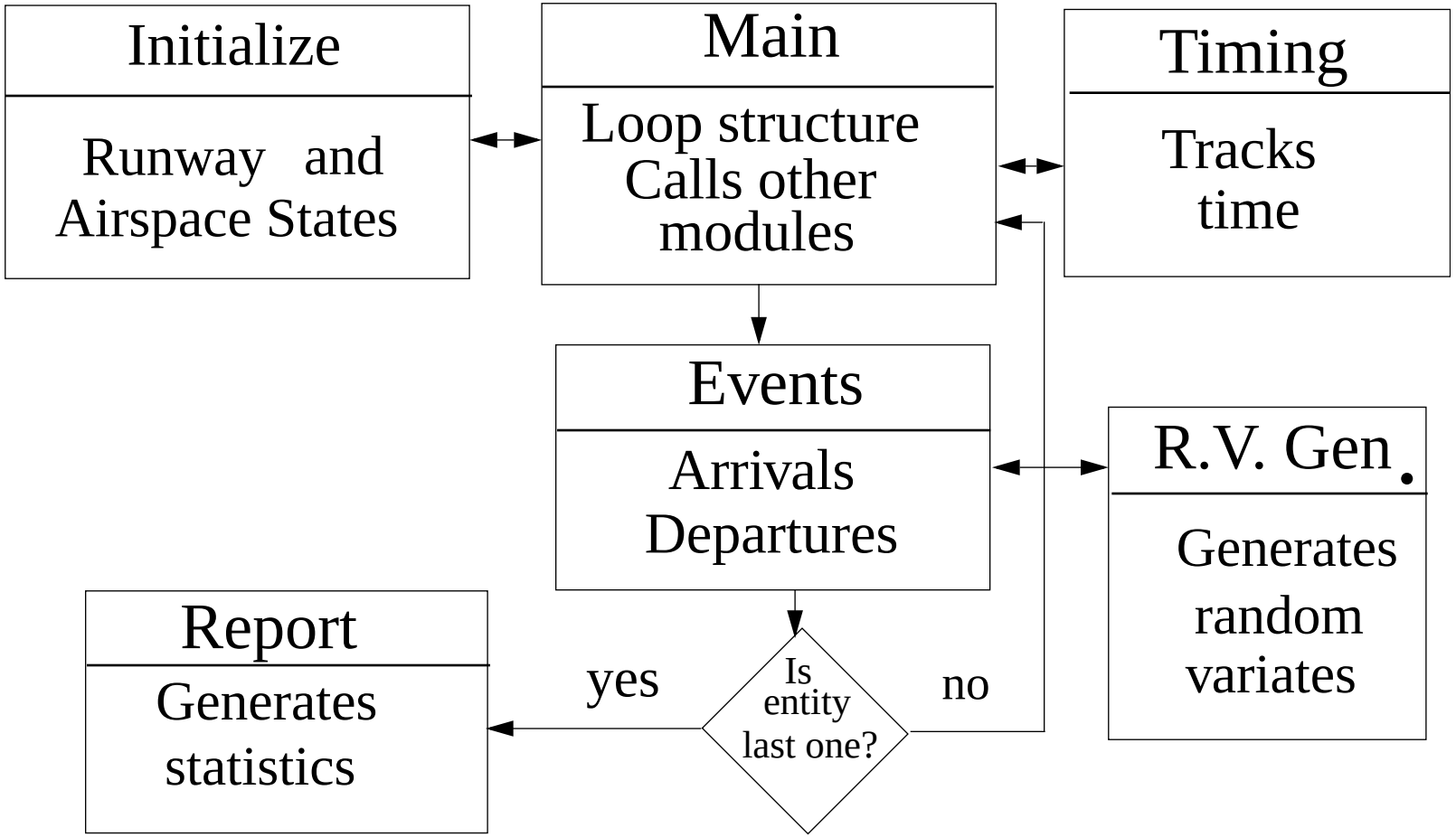
Queues: representation of physical waiting spaces in the simulation model (i.e., uniqueue before a security check point)

Accumulators: variables that keep track of simulation performance measures (i.e., queue length, level of service, etc.)

Events: important milestones in the simulation process (i.e., arrivals, departures, simulation initiation and completion). Events can be:

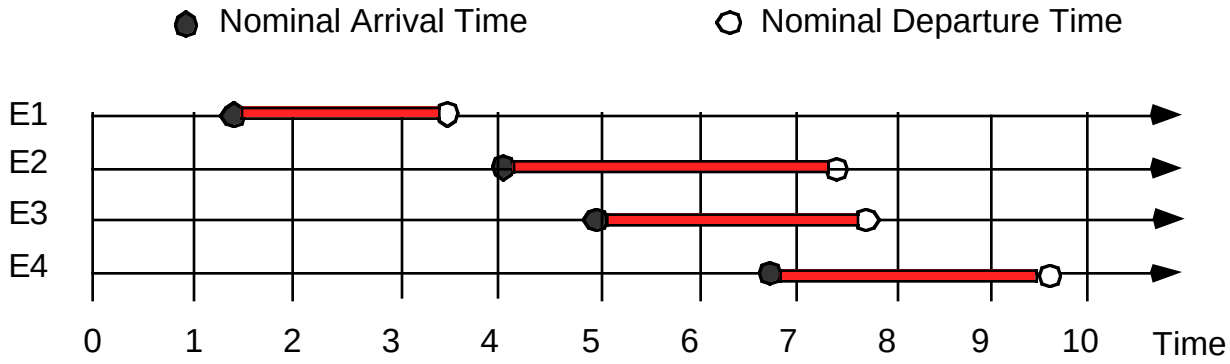
- External (user inputs)
- Internal (the result of entity process conflicts)

Structure of a D.E. Simulation (Law and Kelton)



Sample Discrete Event Simulation

Suppose that we have four processes as shown below



Entity	Arrival Time	Interarrival Time	Service Time
1	1.4	2.6	2.1
2	4.0	0.9	3.5
3	4.9	1.8	2.8
4	6.7	0.0	2.9

Verbal Description of Events

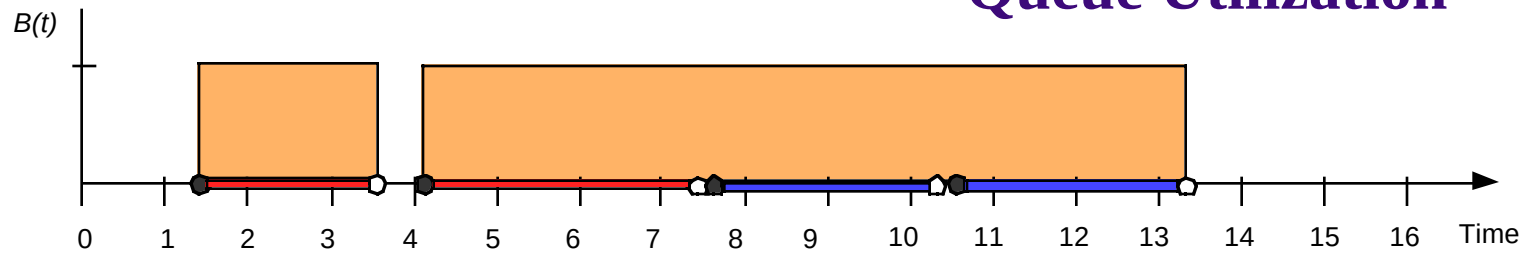
Assume First-In-First-Out events apply:

- Entity 1 enters the system and departs without delay
- Entity 2 enters the system and departs without delay
- Entity 3 arrives 0.9 time units after Entity 2 and is delayed 2.6 time units before service (at 7.5 time units)
- Entity 4 arrives at 6.7 time units while Entity 2 is being serviced and Entity 3 waits in the queue
- Entity 4 waits until 10.3 time units to be serviced
- Entity 3 departs at 10.3 time units
- Entity 4 departs the system at 13.2 time units

Queue Utilization and Queue Length Interpretation

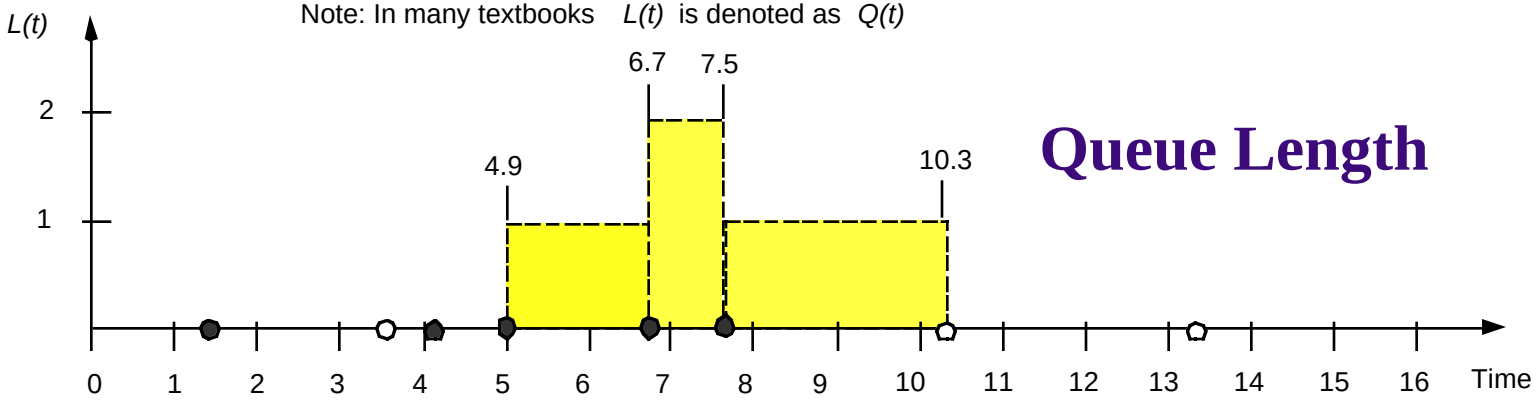
The following diagrams represent times when the system is in use and the queue length observed

Queue Utilization



Note: In many textbooks $L(t)$ is denoted as $Q(t)$

Queue Length



Simulation Parameters

Assume a single resource (server) provides service to these entities. Let,

D_i be the delay for the ith entity entering a system

N is the number of entities processed during the simulation time T

L is the time-average number of entities in the queue

$Q(t) = L(t)$ is the instantaneous queue length

$B(t)$ is a busy function such that,

$$B(t) = \begin{cases} 1 \text{ if system is busy} \\ 0 \text{ if system is idle} \end{cases}$$

A statistic to compute the average delay of entities processed by the system is,

$$\bar{D} = \frac{\sum_{i=1}^{N} D_i}{N} \qquad (1)$$

Similarly, the time-average number of entities in the queue is computed by integrating $L(t)$ over time (0,T)

$$\bar{L} = \int_0^T L(t)dt/T \qquad (2)$$

Finally, the utilization of the system can be deduced from direct observation of the busy function, $B(t)$ and its integral over time (0,T),

$$\rho = \int_0^T B(t)dt/T \qquad (3)$$

These metrics constitute the foundations on how a simulation model can be effectively used to predict levels

of service inside airport terminals and practically everywhere where a waiting line forms.

Computations for Hand Calculation Example

Looking back at the previous example we compute the queueing parameters using equations 10-12. Use 16 time units as the simulation life-span.

$$\bar{D} = (0 + 0 + 2.6 + 3.6)/4 = 1.55 \quad \text{time units}$$

$$\bar{L} = \int_0^{16} L(t)dt/T \approx \sum_{i=1}^{16} L(t)\Delta t/16$$

$$\bar{L} = \frac{(6.7 - 4.9) \times 1 + (7.5 - 6.7) \times 2 + (10.3 - 7.5) \times 1}{16} = 0.525$$

entities

And the utilization factor is,

$$\rho = \int_0^T B(t)dt/T = \sum_{i=1}^{16} B(t)\Delta t/16 = 0.706$$

From these statistics we conclude the following,

- The use of the single server system is quite good (about 70% of the time the server is busy)
- Simulation is an intensive book keeping activity that is obviously suited to computers
- Simple formulae can be used to obtain vital statistics of the system modeled

A More Formal Simulation Process

- A more formal simulation process (other than hand calculations) is introduced here
- The example and nomenclature used here correspond to that used by Law and Kelton (1991)
- A simple single server queueing process is first explained and then an example is presented
- Results of the simulation process are compared with analytic results derived from a stochastic queueing model

Simulation Blocks

Initialization: initializes counters (to keep statistics) and global variables

Main: controls other routines and acts as director. Performs calls to others

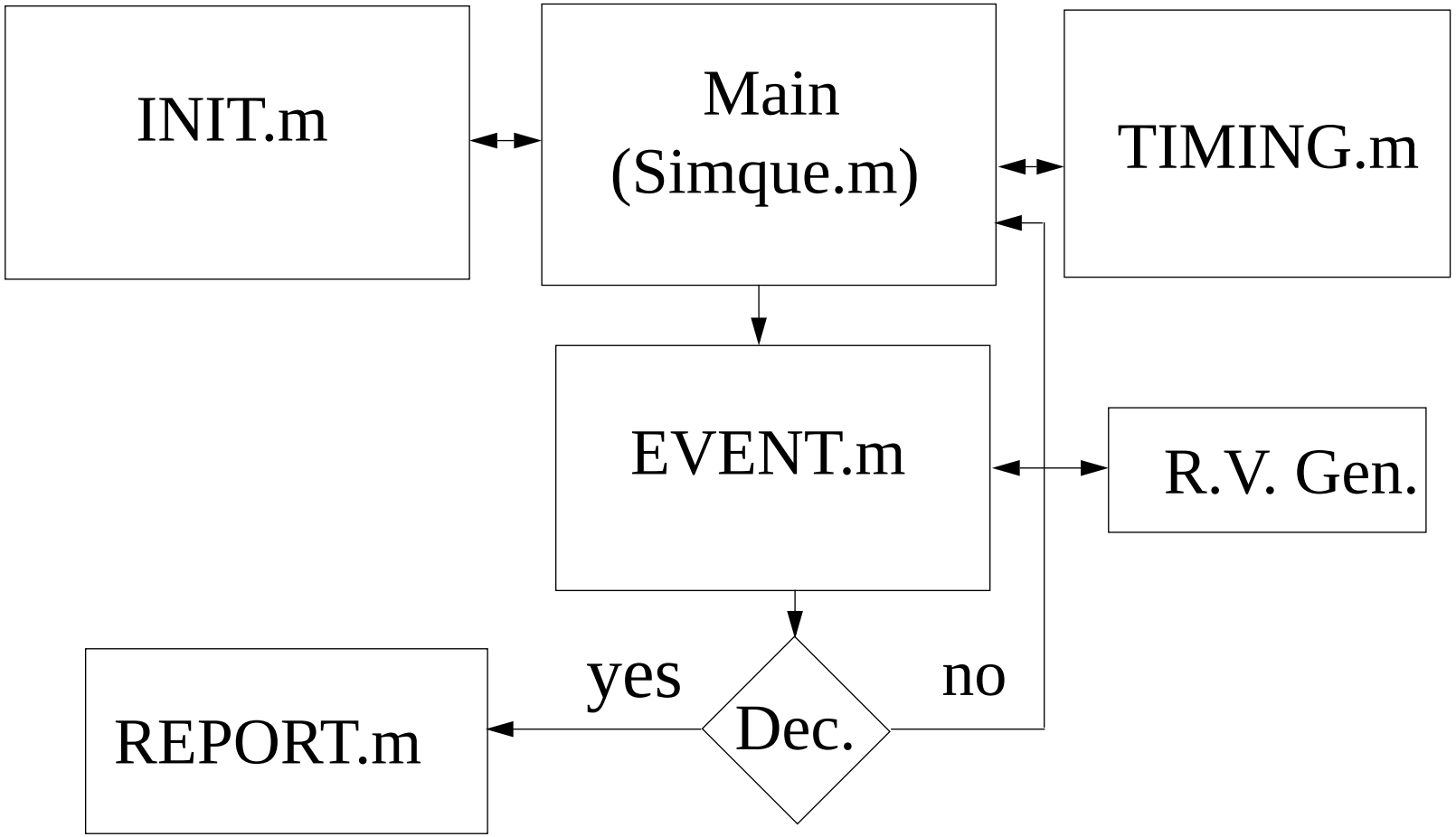
Timing: keeps the simulation clock up-to-date

Report: writes and plots summary statistics of the simulation model

Event: tracks and schedules events in the simulation

Generator: generates random variates needed in the simulation

Sample Single Server Queue Simulation



M/M/1 Queue Simulation

```
% For simulation model of a single server queue
% Adapted from Law and Kelton (1991)
% Programmers: H. Baik and A. Trani (1997)

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
```

```
% mean_interarrival = mean time between arrivals
% mean_service = mean service time
% mean_delays_required = total no. of customers to be
                         processed

% area_server_status =
% num_events = type of events
% Q_LIMIT = maximum number of storage locations for
                         queue
% num_custs_delayed = customers served at time t

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;               % determine the next event
  UPDATE;               % update time-average statistical
                          accumulators

  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
```

INIT - Initialization Routine

```
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;                  % clock
 server_status=0;           %idle
 num_in_q=0;      % initial queue length
```

```
  time_last_event=0.0; % time of the most recent event
  num_custs_delayed=0;% no. of customers delayed
  total_of_delays=0.0;% total delay to customers
Function INIT - Initialization Routine

  area_num_in_q=0.0;
  area_serve_status=0.0;

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

ARRIVE - Arrival Routine

```
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);
                        % next arrival time
  %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;
   totoal_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
```

DEPART - Departure Routine

```
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

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

     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 customer's arrival time in queue up one
                        place
```

```
    for i = 1:num_in_q
      time_arrival(i)=time_arrival(i+1);
    end
end
```

EXPON - Estimates Neg. Exponential Random Variates

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

% uses the inverse transformation method to generate
        negative exponential random numbers
```

TIMING - Timing Routine

```
function 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_interarrival...
      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;
```

UPDATE - Updates Queue Statistics

```
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;
```

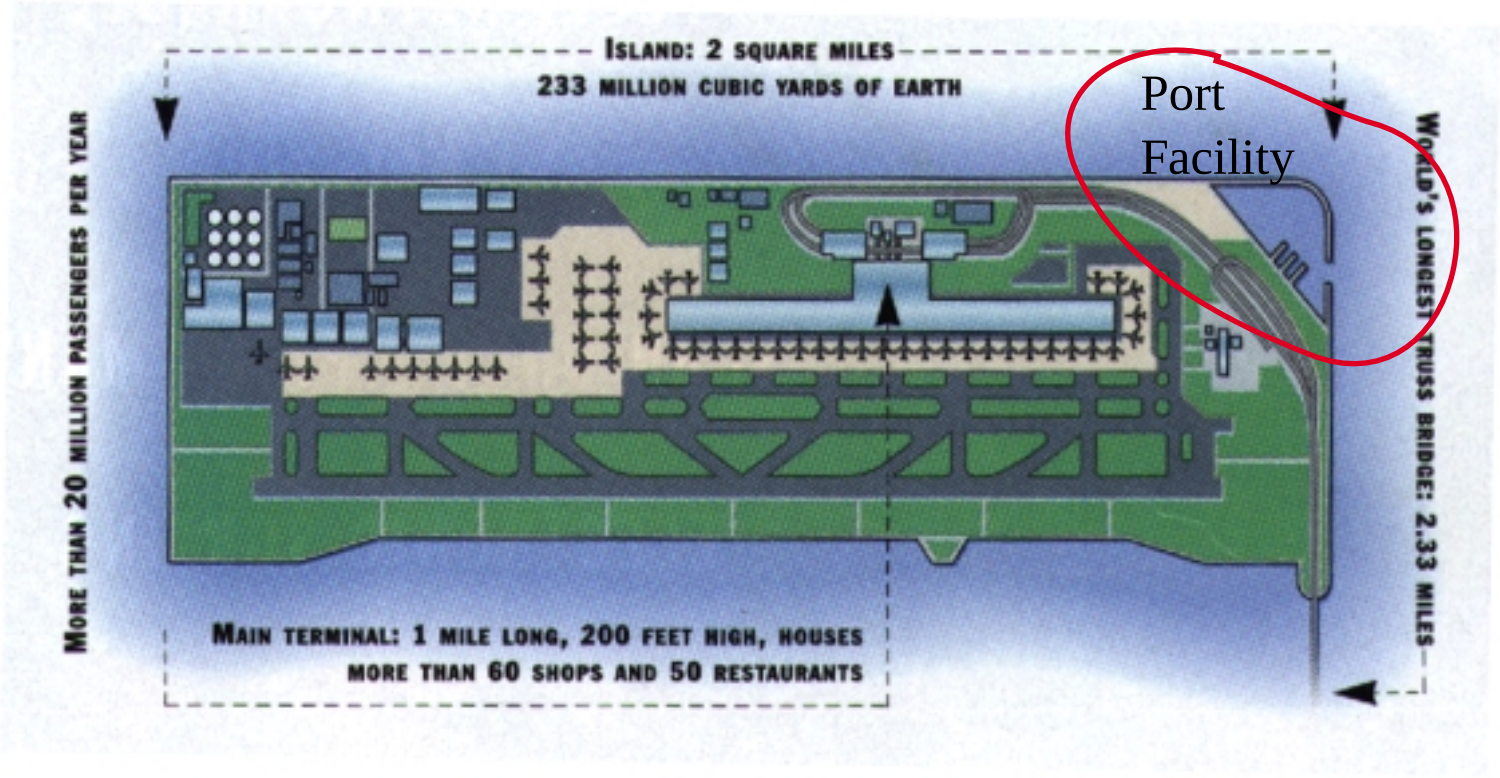
REPORT - Report Generation Routine

```
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)]);
```

M/M/1 Simulation Example

Kansai Airport in Osaka Bay has a small port facility to serve large intermodal cargo shipments.



M/M/1 Simulation Example (cont.)

Suppose that only one berth facility is currently available to serve large ships.

Initial conditions of the problem:

a) Port facility is open 24 hours per day

b) A ship arrives every 22 hours (on the average - Poisson distribution)

c) A ship is loaded/unloaded in 12 hours (average - negative exponential distribution) by a single 12-ton crane

The idea is to find the operational parameters of the facility using the M/M/1 simulation model developed.

M/M/1 Queue Simulation Results

The following results have been obtained using the single server queue simulator

Let:

L_q be the expected number of ships in the queue

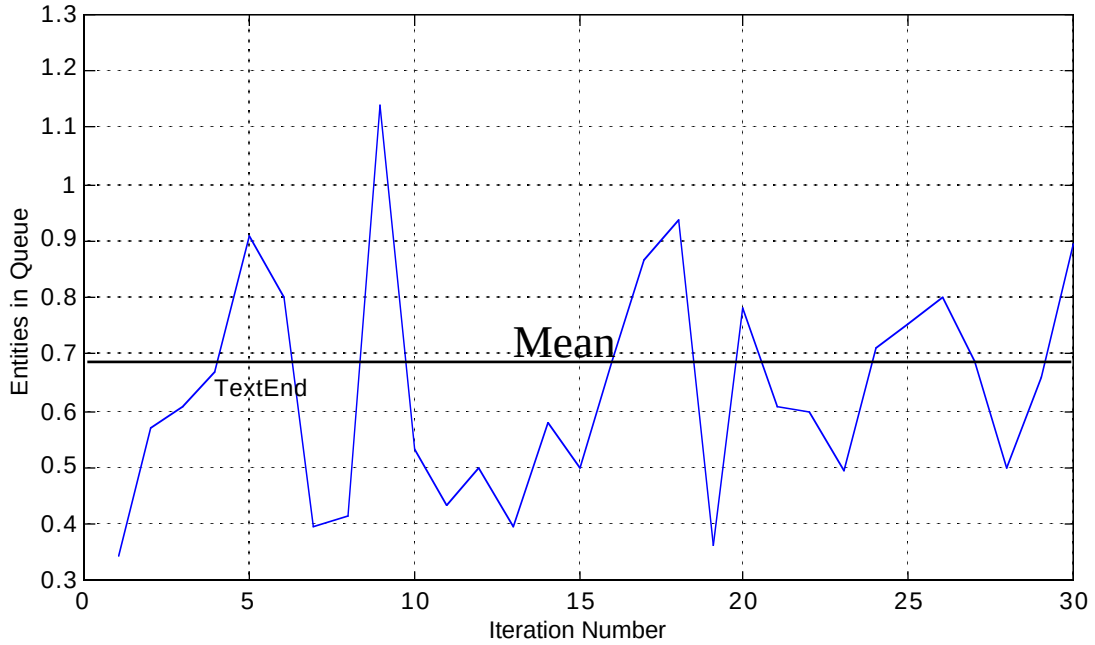
W_q be the expected waiting time in the queue

T_{final} be the final simulation time (days) to process 500 ships

ρ be the utilization of the service facility (i.e., crane)

M/M/1 Queue Simulation Results

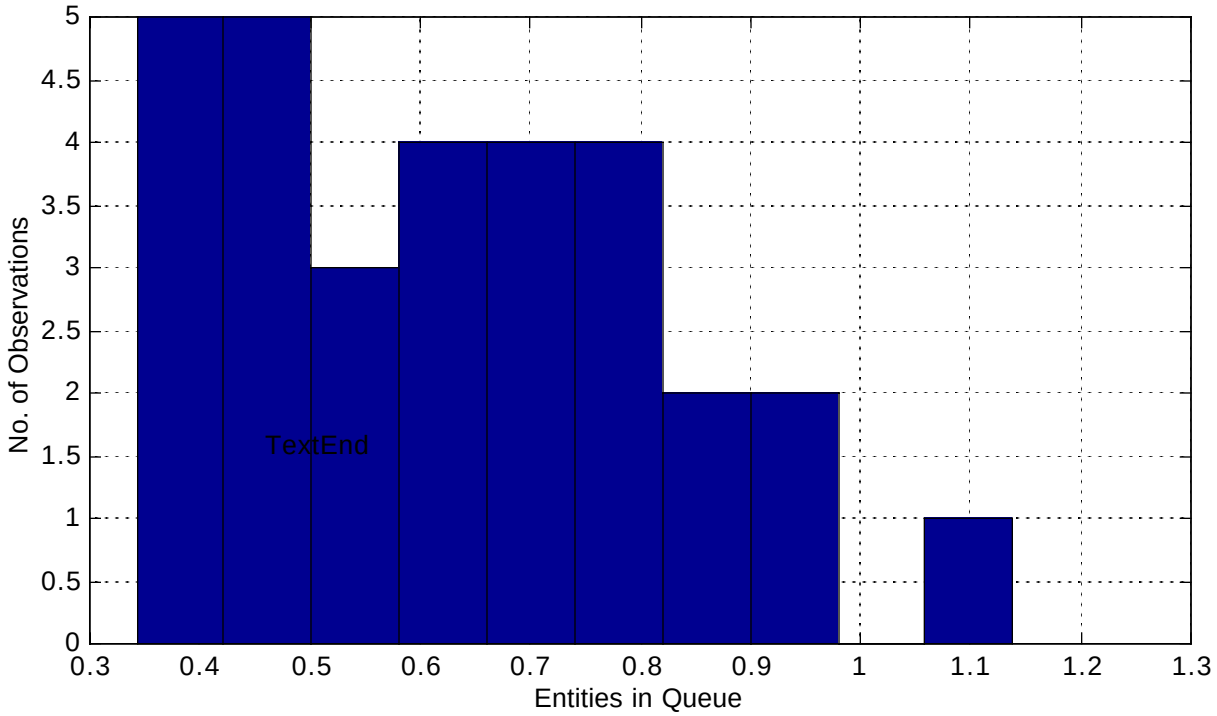
Performing 30 iterations of the simulation model yields the following results:



Mean L_q = 0.6885 ships in the queue

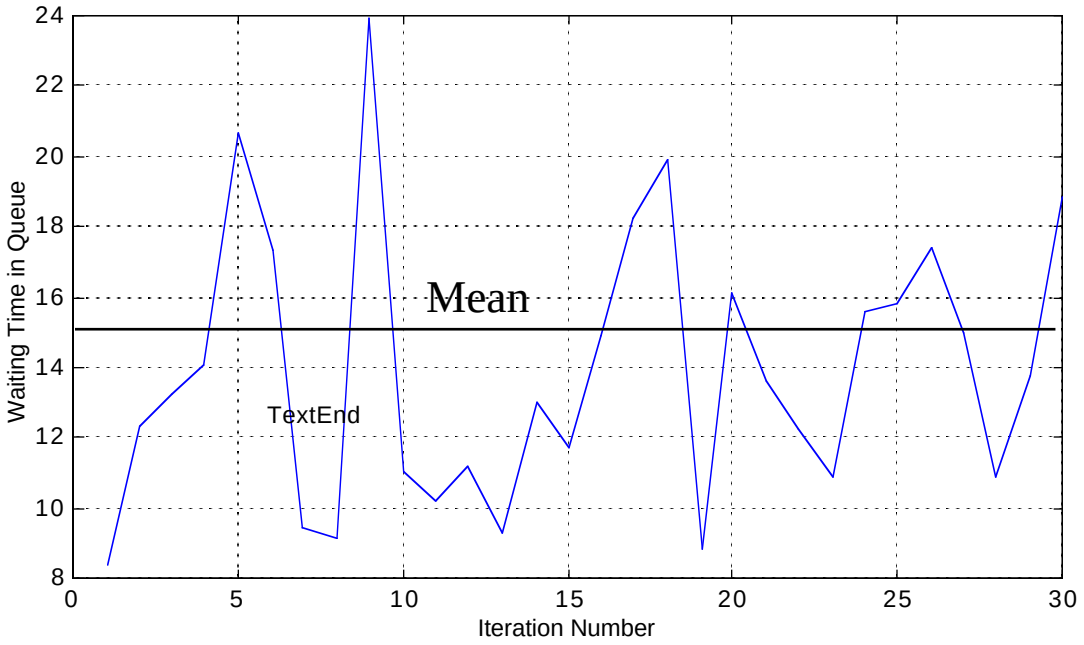
M/M/1 Queue Simulation Results

Histogram for L_q .



M/M/1 Queue Simulation Results

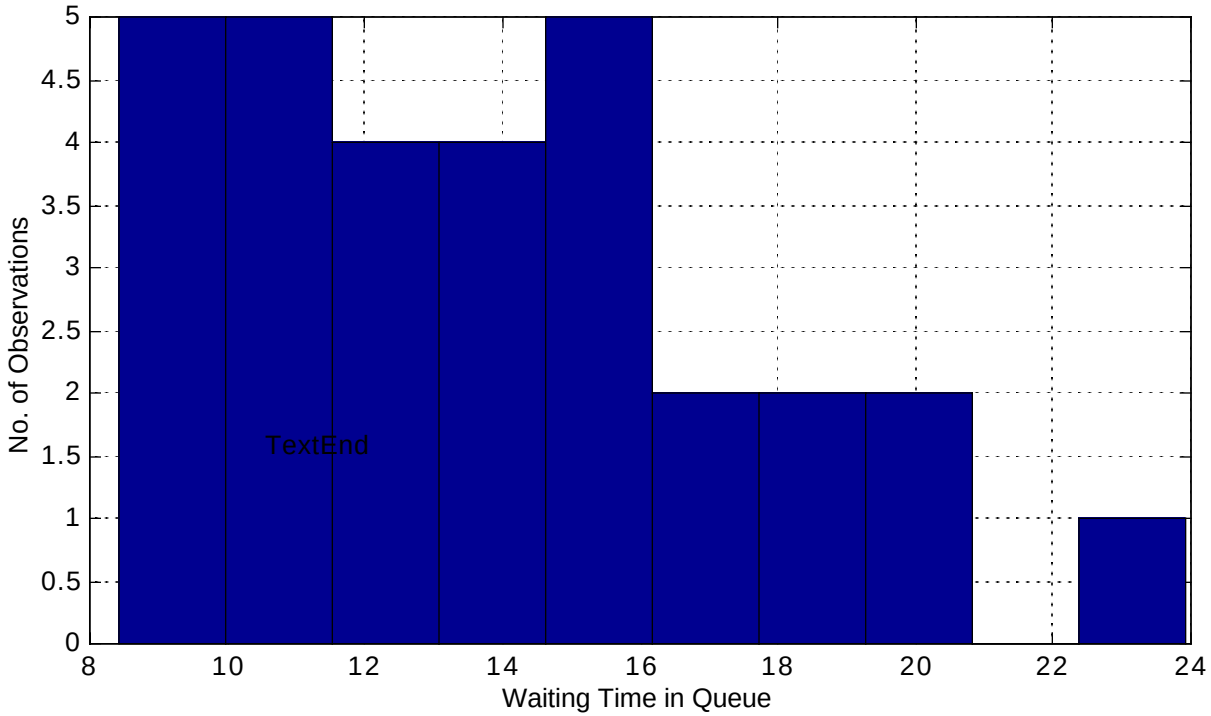
Results for waiting time in the queue, w_q:



Mean w_q = 15.04 hours in the queue

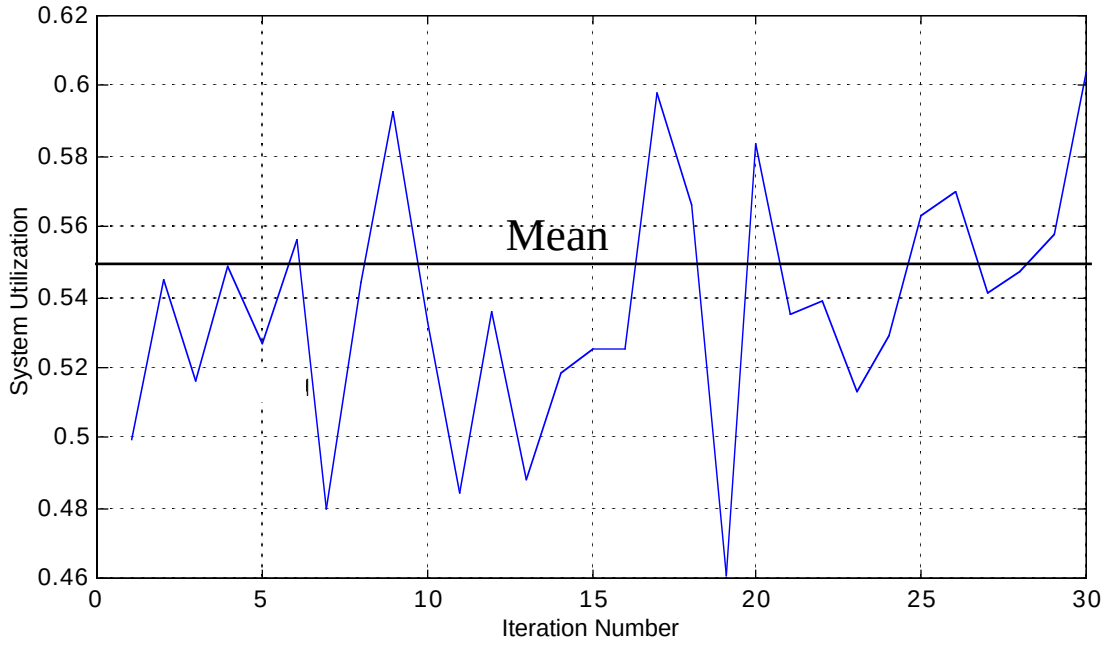
M/M/1 Queue Simulation Results

Results for waiting time in the queue, W_q:



M/M/1 Queue Simulation Results

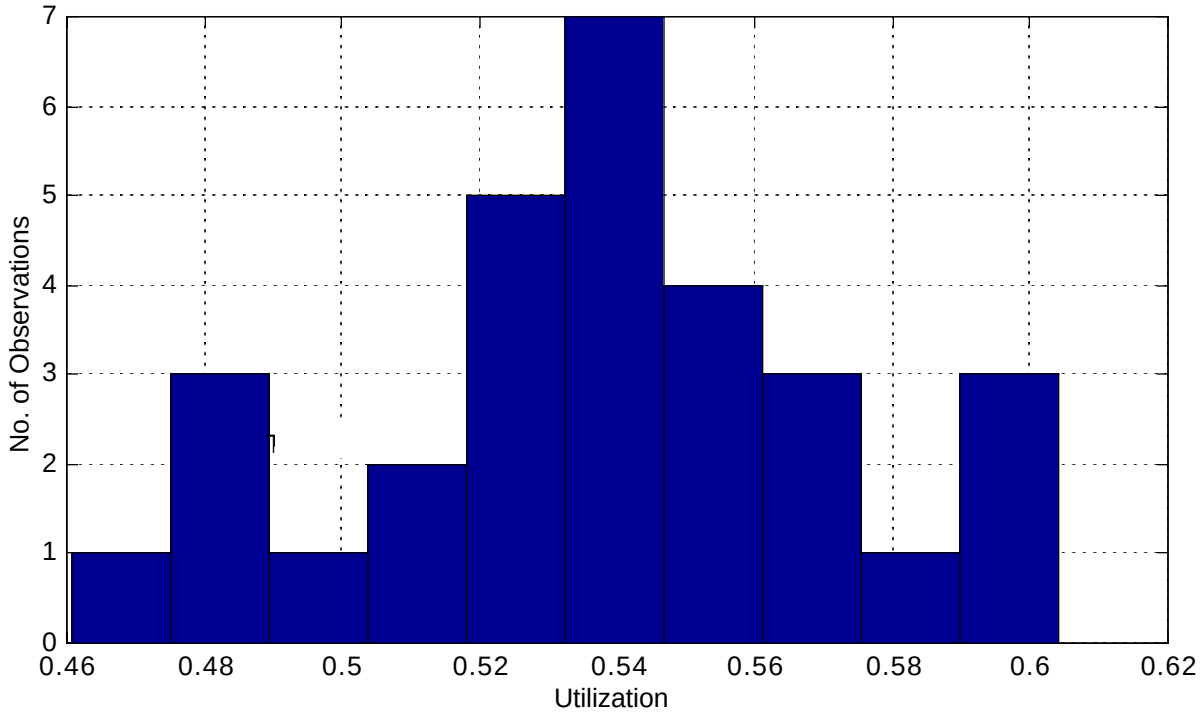
Results for crane utilization, ρ :



Mean ρ = 0.5492

Results for M/M/1 Queue

Histogram of crane utilization (note the central tendency)

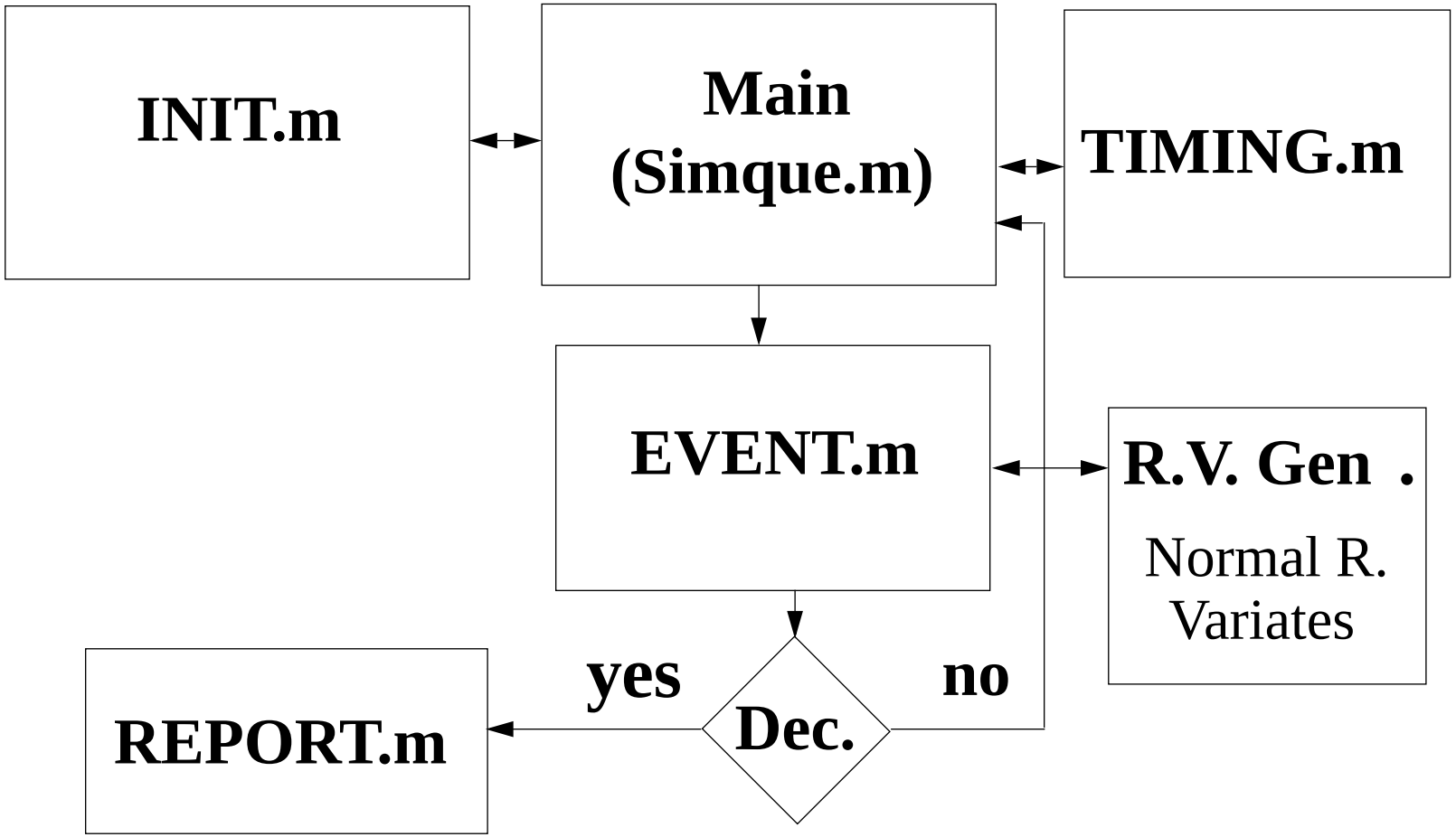


M/M/1 Queue Comparison of Results

Parameter	Simulation Results	Analytic Solution
ρ = Utilization	0.5492	0.5455
L_q = No. of Ships	0.6885 ships	0.6545 ships
W_q = Waiting Time	15.04 hours	15.04 hours

Note: **500 replications** and **30 trials** (average values shown)

Model M/Normal/1 Queue



Matlab Model (M/Normal/1 Queue Simulation)

```
% For simulation model of a single server queue
% Adapted from Law and Kelton (1991)

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
```

```
% mean_interarrival = mean time between arrivals
% mean_service = mean service time
% mean_delays_required = total no. of customers to be
                         processed
```

Matlab Model (M/Normal/1 Queue Simulation)

```
% area_server_status =
% num_events = type of events
% Q_LIMIT = maximum number of storage locations for
                    queue
% num_custs_delayed = customers served at time t

Q_LIMIT=200;
mean_interarrival=22;
mean_service=12;% mean service times
std_mean_service = 2;% standard deviation of service
                    times
mean_delays_required=500;
area_server_status=0;
```

num_events=2

INIT; % calls initialization routine

SIMQUE_n.m File (Normal RV File)

```
% run the simulation while more delays are still required

while(num_custs_delayed < mean_delays_required)

  TIMING;          % determine the next event
  UPDATE;          % update time-average statistical
                   accumulators

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

REPORT_n; % call the report generator function

Normally Distributed Random Variates

Only affects the DEPART function (called DEPART_n) here on:

```
function DEPART_n

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 ...
    std_interarrival dep std_mean_service
```

```
if(num_in_q == 0)
% queue is empty. so make the server idle and eliminate
                    the departure event from
    the considerarion

    server_status = 0;                %idle
    time_next_event(2) = 1.0*exp(30);

else
% queue is not empty, so decrement 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;
```

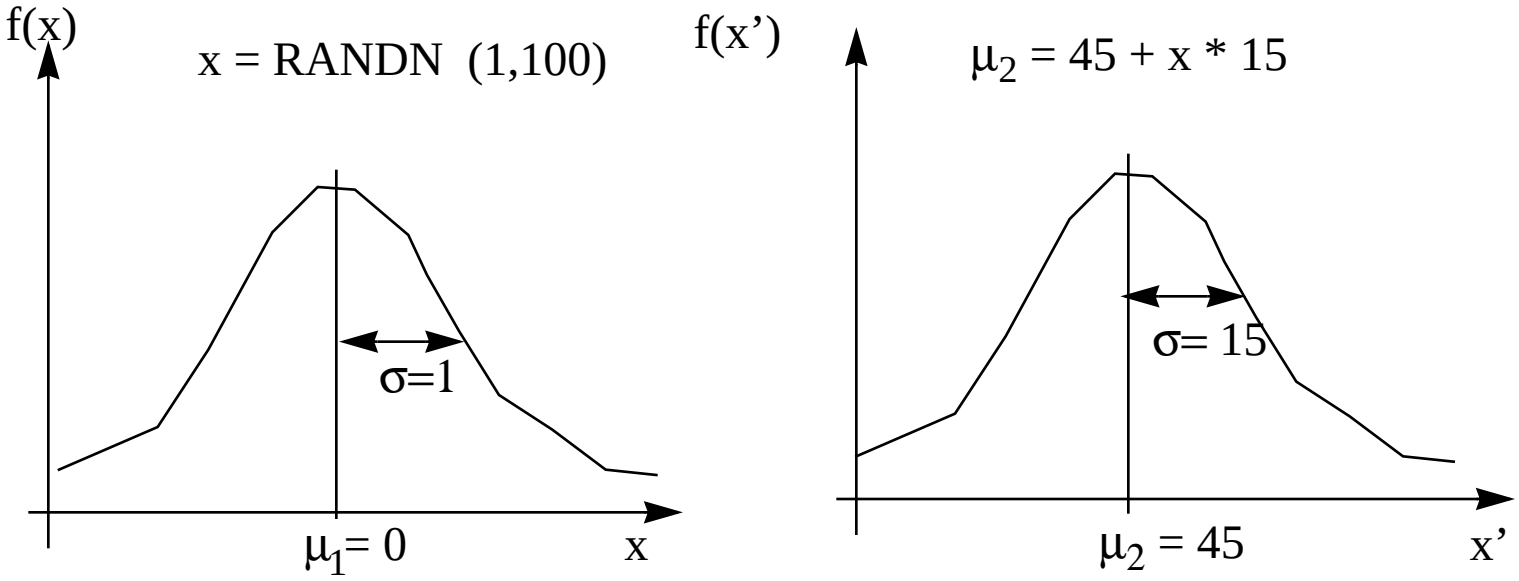
```
num_custs_delayed = num_custs_delayed + 1;

time_next_event(2) = time +
                     normal(mean_service,std_mean_service) ;
dep (num_custs_delayed + 1) =
                     normal(mean_service,std_mean_servi
                     ce);


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

Normally Distributed Random Variates

New function to estimate normal random variates
function e=NORMAL(mean,std)

```
k = randn(1);        % generates one normally distributes
                       R.V.
e = mean + k*std;  % scales k to the desired mean and
                       standard deviation values
```

Physical Interpretation



M/Normal/1 Queue Simulation Results

The following results have been obtained using the single server queue simulator

Let:

L_q = the expected number of ships in the queue

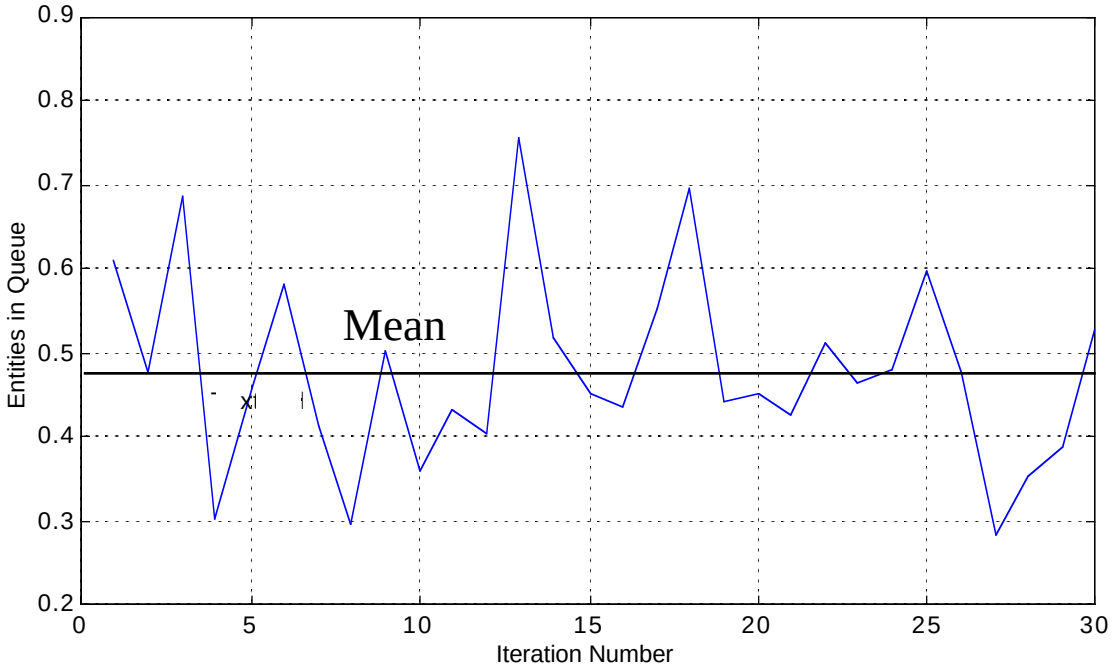
W_q = the expected waiting time in the queue

T_{final} = final simulation time (days) to process n entities

ρ = the utilization of the service facility

Queue Simulation Results

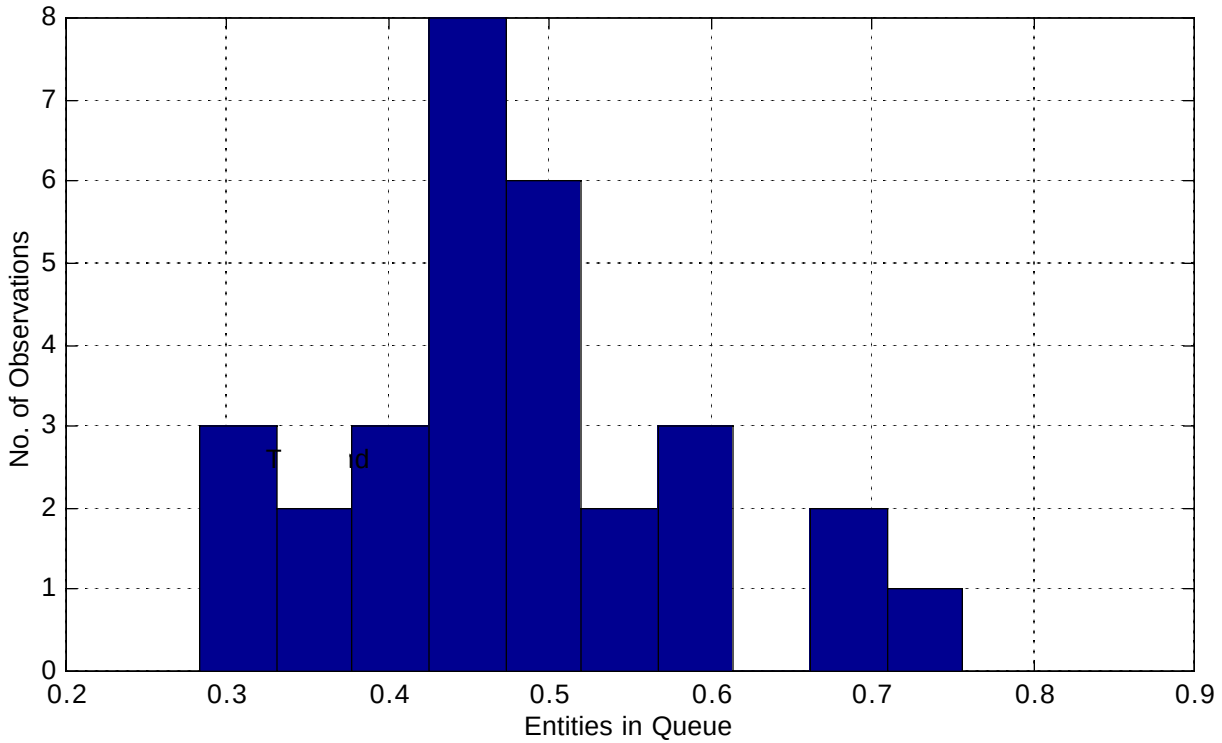
Performing 30 iterations of the new simulation file yields the following results (normally distributed service times):



Mean L_q = 0.4782 ships in the queue

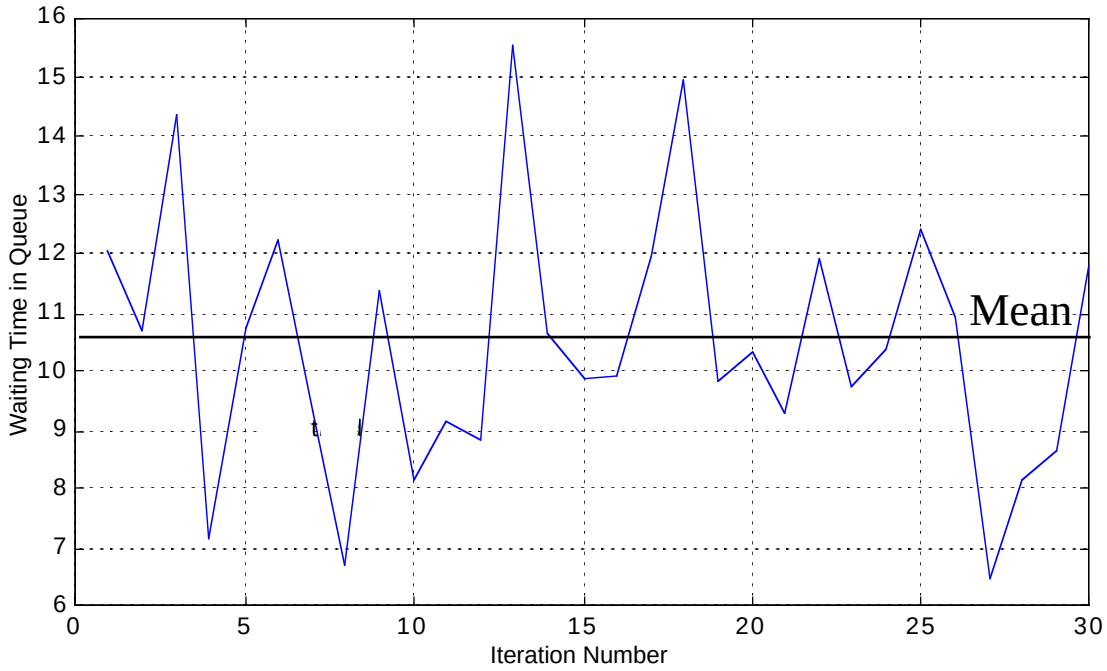
M/Normal/1 Queue Simulation

Histogram for L_q with normally distributed service time.



M/Normal/1 Queue Simulation

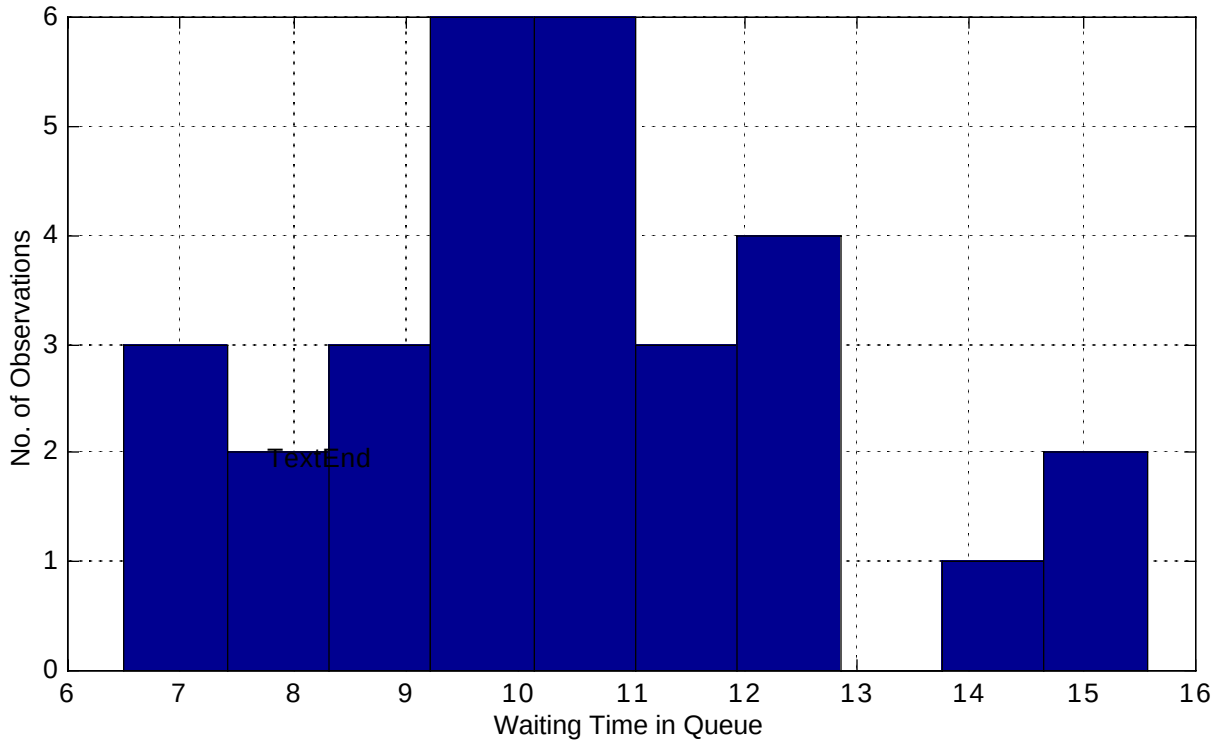
Results for waiting time in the queue, w_q:



Mean w_q = 10.45 hours in the queue

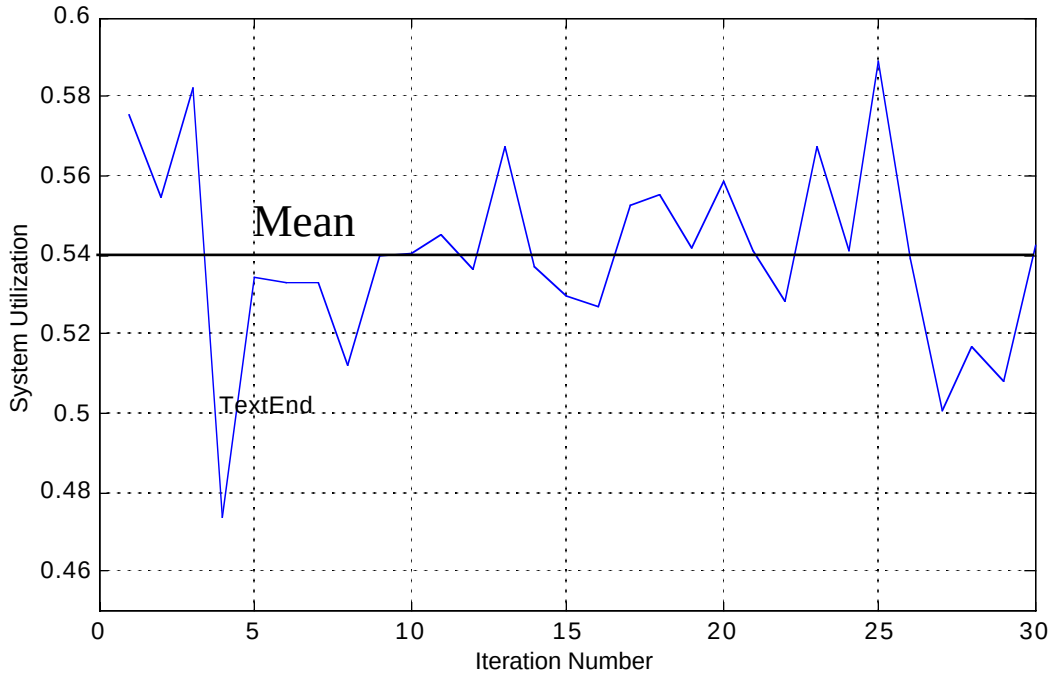
M/Normal/1 Queue Simulation

Waiting time in the queue (W_q), with normally distributed service times (12,2 hours).



M/Normal/1 Queue Simulation

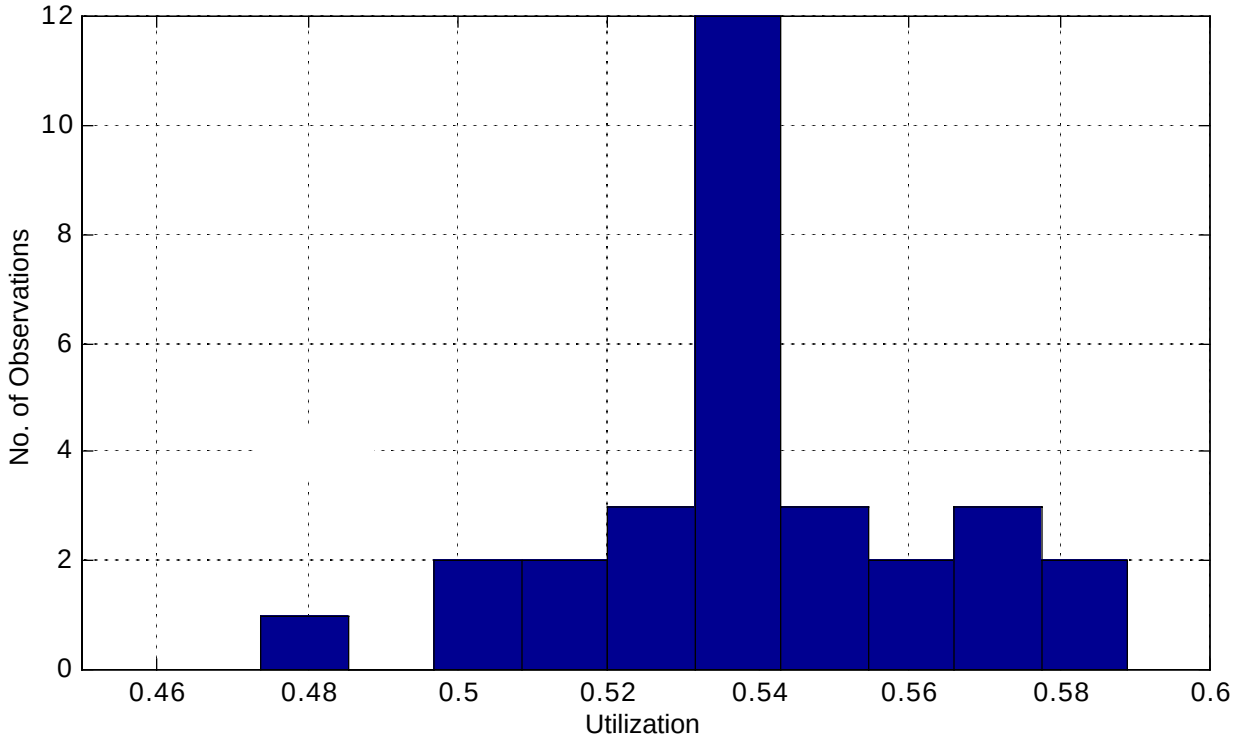
Results for system utilization, ρ :



Mean ρ = 0.5402

M/Normal/1 Results

The following results illustrates the central tendency of the utilization



Queue Simulation Comparison

Parameter	Neg. Exponential Distribution (β = 12 hours)	Normally Distributed Service Times (μ=12, σ=2 hours)
ρ = Utilization	0.5492	0.5402
L_q = No. of Ships	0.6585 ships	0.4786 ships
W_q = Waiting Time	15.04 hours	10.45 hours

Note: **500 replications** and **30 trials** (average values shown)

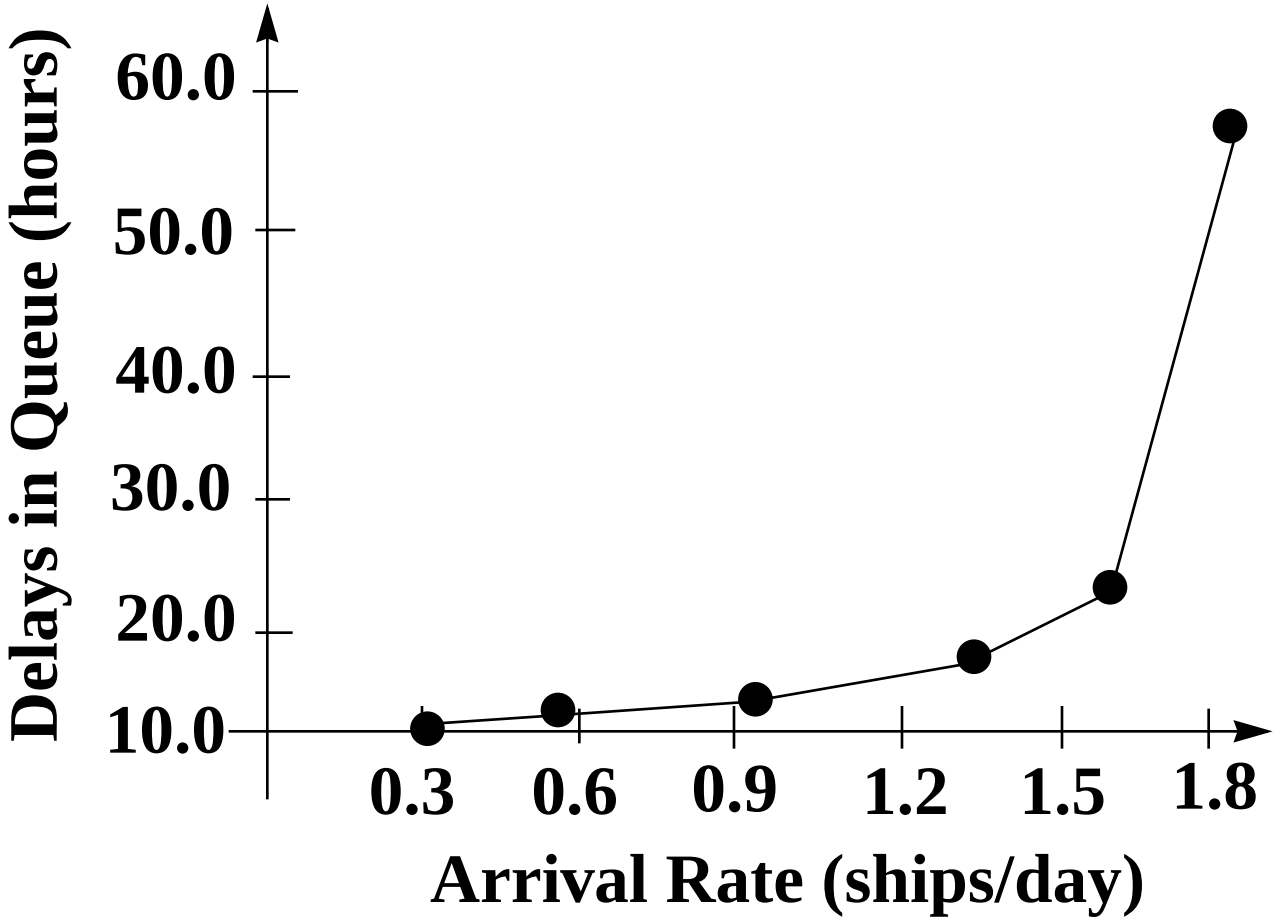
Sensitivity of System to Demand Changes

The demand function can be changed with simple modifications to the arrival rate function (**mean_interarrival**). Look at Input File (simque_n.m)

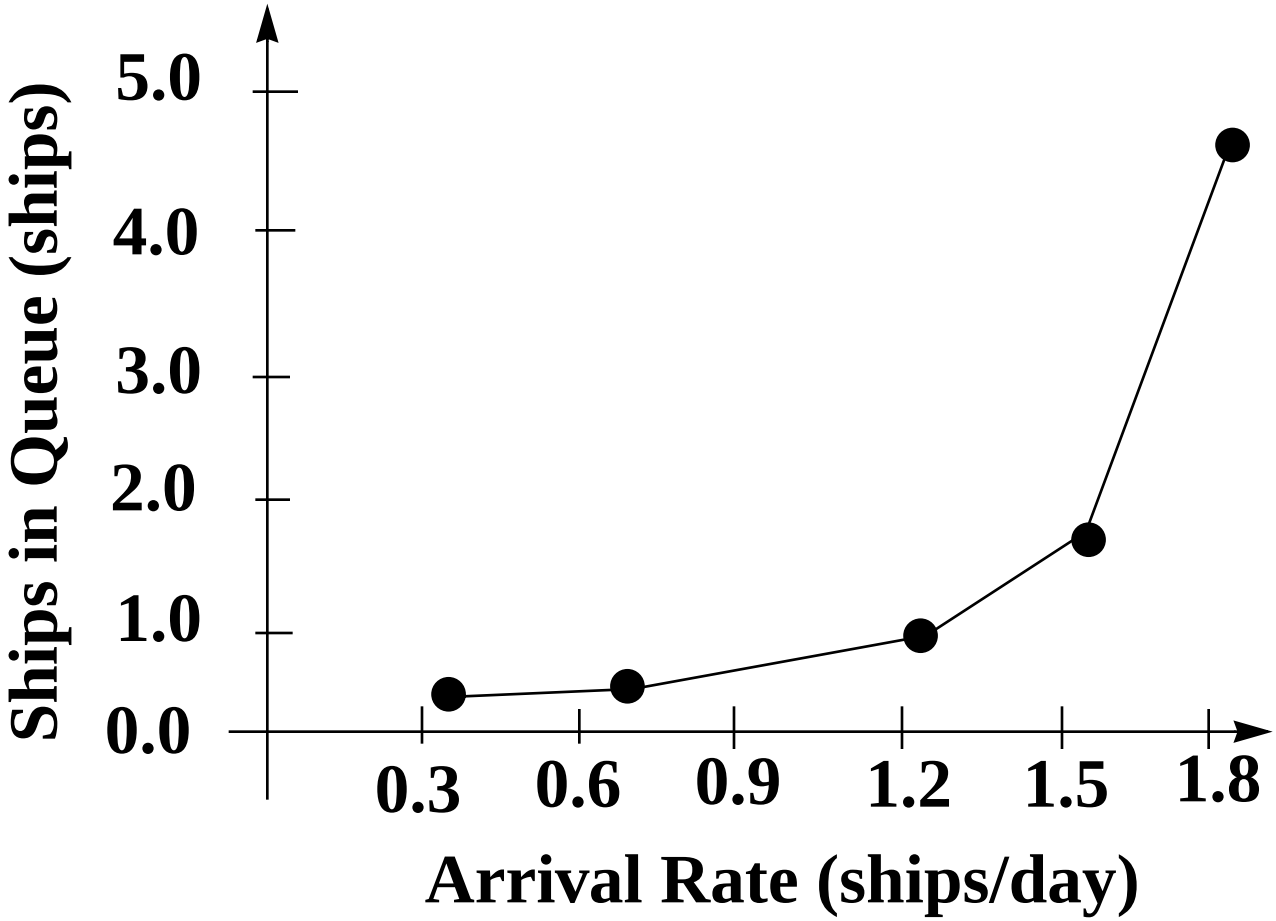
```
Q_LIMIT=200;
mean_interarrival = 22;
mean_service=12;                % mean service times
std_mean_service = 2;           % standard deviation of service times
mean_delays_required=500;
```

Suppose mean_interarrival is increased from **13 to 36 hours.** The following results illustrate the variations to delay times.

Sensitivity to Demand Changes



Sensitivity to Demand Changes



Sample Source Code (Matlab)

```
% For simulation model of a single server queue

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

% mean_interarrival = mean time between arrivals
% mean_service = mean service time
% mean_delays_required = total no. of customers to be processed
```

Matlab Model (M/M/1 Queue Simulation)

```
% area_server_status =
% num_events = type of events
% Q_LIMIT = maximum number of storage locations for queue
% num_custs_delayed = customers served at time t

Q_LIMIT=200;                    % queue limit in the simulation
mean_interarrival=22/24;
mean_service=12/24;
mean_delays_required=500;       % total number of vehicles simulated
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;                    % determine the next event
  UPDATE;                    % update time-average statistical accumulators

  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
```