CHAPTER 6

SIMULATION OF QUEUING SYSTEMS

6.1 Introduction

Waiting line queues are one of the most important areas, where the technique of simulation has been extensively employed. The waiting lines or queues are a common site in real life. People at railway ticket window, vehicles at a petrol pump or at a traffic signal, workers at a tool crib, products at a machining center, television sets at a repair shop and consumers at a ration depot are a few examples of waiting lines. The waiting line situations arise, either because,

- There is too much demand on the service facility so that the customers or entities have to wait for getting service, or
- There is too less demand, in which case the service facility have to wait for the entities.

Thus, when facilities are inadequate, the entities wait and incur cost due to waiting time, and when the demand is inadequate, the facilities wait and incur cost due to idle time. The objective in the analysis of queuing situations is to balance the waiting time and idle time, so as to keep the total cost at minimum.

The queuing theory owe its development to an engineer named A.K. Erlang, who in 1920, studied waiting line queues of telephone calls in Copenhagen, Denmark. The problem was that during the busy period, telephone operators were unable to handle the calls, there was too much waiting time, which resulted in consumer dissatisfaction.

Many researchers carried on the research work in the telephone traffic further and it was only after World War II, that the queuing theory encompassed the waiting line situations from other fields as business and industry.

6.2 The Components of a Waiting Line System

(*i*) **Calling Source** or the population from which customers are drawn. The calling source may be finite or infinite. When the arrival of a customer does not affect the next arrival, the source of customers is said to be infinite. Population of vehicles arriving at a toll booth or patients arriving at a hospital have infinite calling source. In case of a repair crew looking after a group of 5 machines, if one breaks down and put under repair, then the next breakdown has to come from only 4 machines, the probability of which will be more than the previous. The population in this case is finite.

(*ii*) **Waiting Line** or queue, *i.e.*, the number of customers waiting to be served. Depending upon the space available, it may again be of finite or infinite length. In a barbershop having four chairs, the queue can have a maximum length of four, because people seldom queue up outside a barber shop. An important case of finite queue is the buffer used to decouple two sequential workstations, where the output of first station is not perfectly matched to the second workstation.

(*iii*) **Service Facility** or the number of service channels. The simplest case is of one service channel, where all the customers form one queue and are attended by one server, a single server model. In many cases, waiting line systems have more than one service facility.

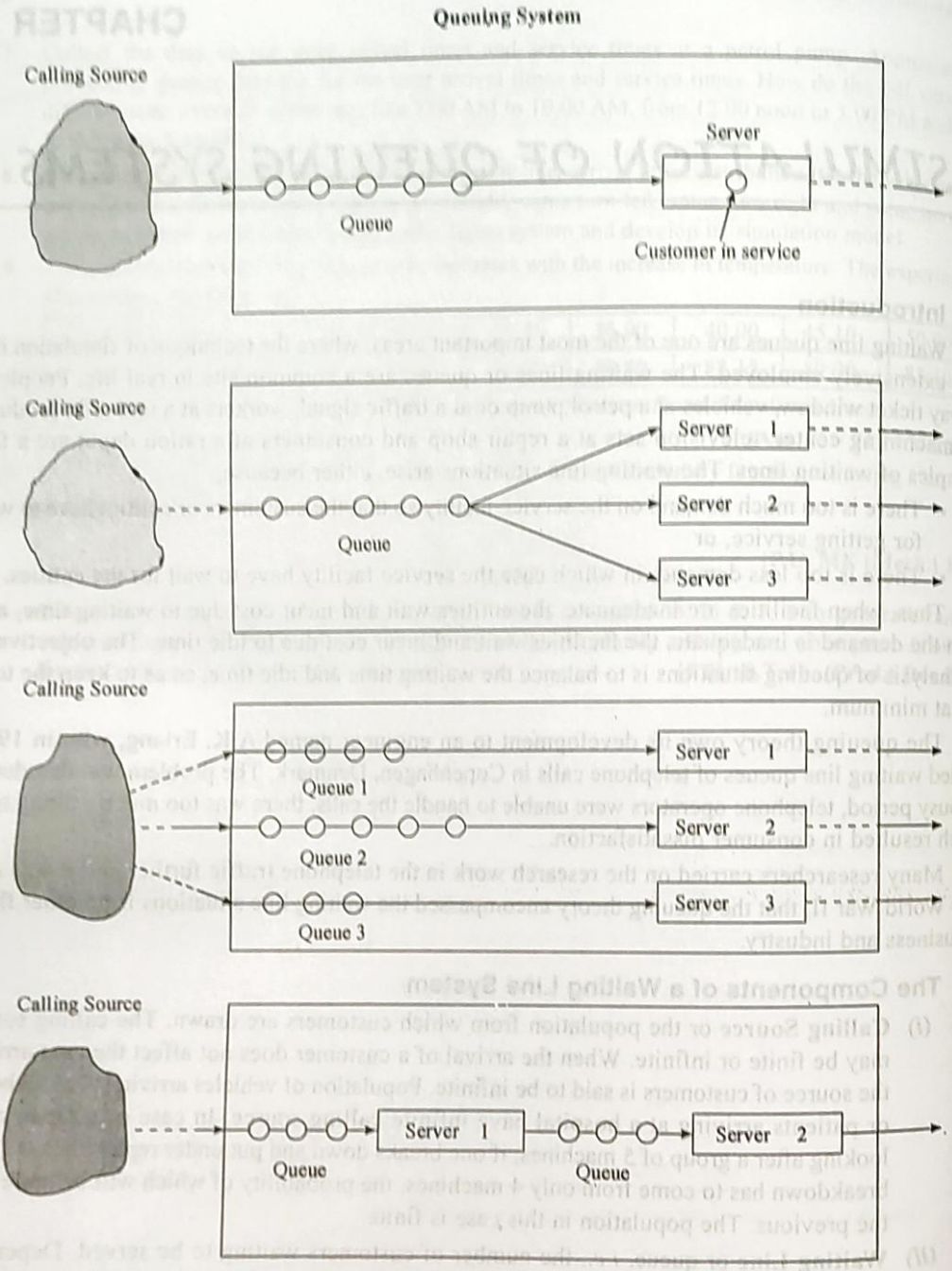

Fig. 6.1

These facilities may be working in parallel or in series. When in parallel, there may be a single queue or multiple queues. Some typical queuing systems are shown in Fig. 6.1.

The important **attributes** that determine the properties of a waiting line queue are,

- Input or arrival rate.
- Output or service rate.
- Service or queue discipline.

The arrival rate is the average number of entities, which join the waiting line per unit time. Depending upon the system, the time unit may be a second, a minute, an hour or a day etc. The average time between consecutive arrivals is called **inter-arrival time.** In most of the situations, the arrivals are a random phenomenon. Different probability density functions are applicable to different arrival patterns, but a commonly made assumption is that arrival rates follow the Poisson distribution.

The service rate or the departure rate is the average number of customers served per unit time. Average service time is the reciprocal of the service rate. Depending upon the situation, the service rate may be constant or variable and may follow any distribution as Uniform, Normal, Exponential, and Erlang etc. But in most of the situations service time is assumed to follow exponential distribution.

The third factor for describing the waiting line is the **queue discipline** that determines how the customers are selected from the queue for service. The most common queue disciplines are given below.

(*i*) **First-in, First-out (FIFO):** Also called the first come first served, is the most common service discipline, according to which the customers are served in the order of their arrival.

(*ii*) **Last-in, First-out (LIFO):** In some situations, the last arrival is served first, as in big go-downs the items coming last are taken out first, in crowded trains or elevators, passengers getting in last come out first.

(*iii*) **Priority:** An arrival may be given priority over the customers waiting in line. A particular machine in a production shop may be more important than the others, and when it breaks down, its repair may be taken up on priority as compared to the other broken down machines. When an arrival, not only goes to the head of the queue, but displaces any unit already in service, it is said to have **pre-emptive priority**. The new arrival is said to pre-empt or interrupt the service.

(*iv*) **Random:** The service discipline is said to be random, when all waiting customers have equal chance of getting selected for service. The selection follows purely random choice.

Some other terms commonly associated with waiting lines are given below.

- **Reneging:** When a queue grows excessively long, a customer waiting in the queue may become impatient and may leave the queue before it is due to enter the service facility. This process is called reneging.
- **Balking:** When a queue grows very long and an arrival refuses to join the queue, it is called balking.
- **Jockeying:** In multiple queues before multiple service channels, where all the channels are providing the same service, a customer may leave one queue and join the other looking faster. This process is called jockeying.
- **Polling:** When there are more than one queues forming for the same service, the action of sharing service between the queues is called polling. A bus picking up passengers from different stoppages along its route is an example of polling service. Separate queues for ladies and gents at a ticket window, is another example of polling service.

6.3 Stationary and Time Dependent Queues

The statistical properties of the arrival and service are generally assumed to be constant and independent of time. Such systems are called stationary systems. However in some situations, both the arrival and service rates may vary with time. For example, in the early morning the number of vehicles reporting at a toll office may be less, and the arrival rate may increase as the day advances towards noon. Similarly, the service rate may be higher in the morning and may come down near lunchtime. Such systems are said to be non-stationary or time dependent or time variant systems.

6.4 Transient and Steady States of the System

A system is said to be in transient state when its operating characteristics vary with time. If we start with an empty queuing system, the customer arriving first does not have to wait at all. The waiting time of customers goes on increasing as the queue builds up with the passage of time. After some time the system reaches steady state, where its behavior becomes almost steady. It is the steady state condition of the system, which is more important. For a system to reach steady state, the arrival rate should be less than the service rate. If the arrival rate is greater than the service rate, the queue length goes on increasing with time and the system never reaches the steady state. Such a state of the system is called **explosive.** In such a system, a maximum limit on the length of the queue should be imposed to ensure steady state.

6.5 Measures of System Performance

The performance of a queuing system can be evaluated in terms of a number of response parameters, however the following four are generally employed.

(*i*) Average number of customers in the queue or in the system.
(*ii*) Average waiting time of the customers in the queue or in the system.
(*iii*) System utilization.
(*iv*) The cost of waiting time and idle time.

Each of these measures has its own importance. The knowledge of average number of customers in the queue or in the system helps to determine the space requirements of the waiting entities. Also too long a waiting line may discourage the prospectus customers, while no queue may suggest that service offered is not of good quality to attract customers.

The knowledge of average waiting time in the queue is necessary for determining the cost of waiting in the queue.

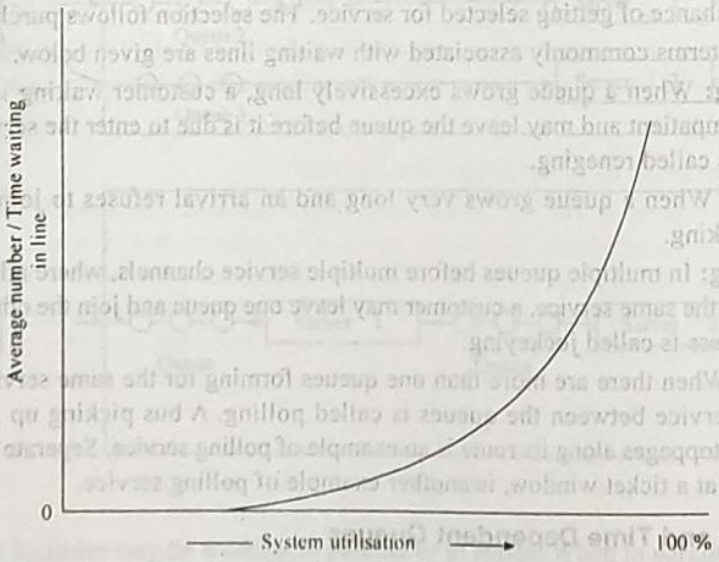

Fig. 6.2

System utilization, that is, the percentage capacity utilized reflects the extent to which the facility is busy rather than idle. System utilization factor (S) is the ratio of average arrival rate (λ) to the average service rate (μ).

$S = \lambda / \mu$ in case of a single server model.
$= \lambda / \mu n$ in case of a 'n' server model.

The system utilization can be increased by increasing the arrival rate which amounts to increasing the average queue length as well as the average waiting time, as shown in Fig. 6.2. Under normal circumstances 100% system utilization is not a realistic goal.

6.6 Costs of Customer Waiting Time and Idle Capacity

The customers in the waiting line who may be human being or physical objects, incur cost while waiting. It may be workers waiting at a tool crib or the trucks waiting for loading or unloading, they incur cost. The capacity cost relates to maintaining the ability to provide service. When a service facility is idle, the capacity is wasted. Both the costs are functions of system capacity, as illustrated in Fig. 6.3. The objective of analysis of any queuing system is to determine the capacity, which minimizes the total cost.

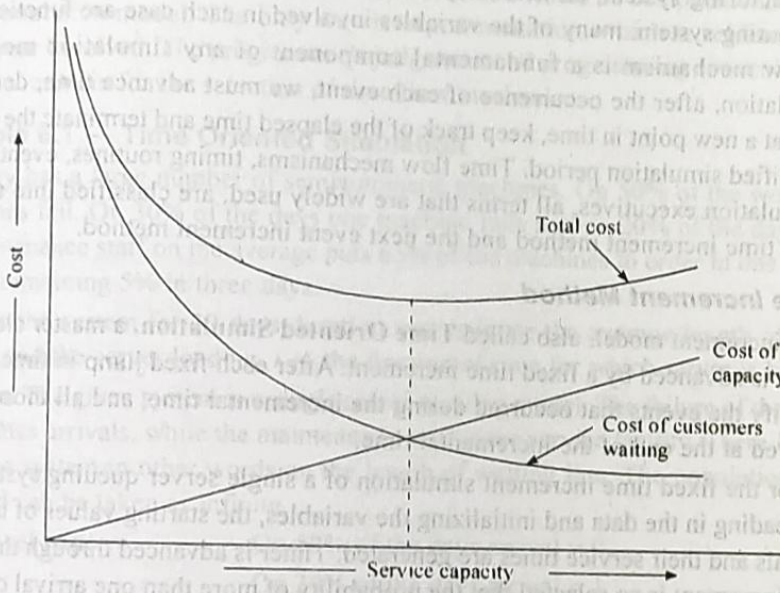

Fig. 6.3

6.7 Kendall's Notation

Kendall's Notation for specifying the characteristics of a queue is V/W/X/Y/Z, where,

V indicates the arrival pattern,
W indicates the service pattern,
X gives the number of servers,
Y represents the system capacity, and
Z indicates the queue discipline.

The symbols used for the inter-arrival times, service times, and the queue discipline are,

Queue characteristic	*Symbol*	*Meaning*
Inter-arrival time or Service time	D	Deterministic
	M	Exponential
	Ek	Erlang with value of parameter k
		Any other distribution
Queue discipline	FIFO	First-in First-out
	LIFO	Last-in First-out
	SIRO	Service in random order
	PRI	Priority ordering
	GD	Any other specified ordering

If the capacity Y is not specified, it is taken to be ∞ (infinite) and if the queue discipline is not specified, it is taken as FIFO.

An M/D/2/5/FIFO stands for a queuing system having exponential arrival times, deterministic service times, two servers, with a capacity of 5 customers and having the first in first out queue discipline.

And an M/D/2 will mean exponential inter-arrival times, deterministic service times, two servers, infinite system capacity and FIFO queue discipline.

6.8 Time Flow Mechanism

Time is an essential element of most of the real life dynamic systems. It may be a queuing situation, a manufacturing system, an inventory control system, a maintenance and replacement problem or a computing system, many of the variables involved in each case are functions of time. Thus the time flow mechanism is a fundamental component of any simulation model. While conducting a simulation, after the occurrence of each event, we must advance time, determine the state of the system at a new point in time, keep track of the elapsed time and terminate the experiment at the end of a specified simulation period. Time flow mechanisms, timing routines, event scheduling procedures, or simulation executives, all terms that are widely used, are classified into two general methods, the fixed time increment method and the next event increment method.

6.8.1 *Fixed Time Increment Method*

In fixed time increment model, also called **Time Oriented Simulation,** a master clock or timer is simulated, which is advanced by a fixed time increment. After each fixed jump in time, the system is scanned to identify the events that occurred during the incremental time, and all those events are treated as if occurred at the end of the incremental time.

Flow chart for the fixed time increment simulation of a single server queuing system is given in Fig. 6.4. After reading in the data and initializing the variables, the starting values of the variables as number of arrivals and their service times are generated. Timer is advanced through the fixed time interval. The time increment is so selected that the probability of more than one arrival or more than one departure during this time is negligible. System is checked for arrival, if there is one it is added to the queue. Then the server is checked if busy or idle. If idle, time is updated, a customer is taken from the queue if available, otherwise, server remains idle. If server is busy, waiting time of the customer is updated. Then timer is advanced to next step. Process is continued until the simulation runs through the specified length of time.

6.8.2 *Next Event Increment Simulation*

In the next event increment model, also called the **Event Oriented Simulation**, the timer is advanced from event to event. At each point in time, the next earliest event is identified and the clock is advanced to that event. The state of the system is updated at each event.

Flow chart for the next event increment simulation of a single server queuing system is given in Fig. 6.5. After reading the data and initializing the variables, one arrival time and one service time are generated, assuming that the first arrival takes place at zero time. The next event, the event to occur at the earliest is then identified. In single server case, only two events are possible, the next arrival time and the next departure (service completion) time. If arrival time (AT) is earlier, clock is advanced to AT, the arrival is added to the queue, and next arrival is generated. If departure time (DT) is earlier, timer is advanced to DT and queue is checked, if a customer is available, it is put on service, clock is advanced to AT and next arrival and departure are generated. The process continues till the simulation ends.

6.8.3 *Comparison of the Two Methods*

The fixed time increment model is always used in the simulation of continuous systems, while both the fixed time and the next event time flow mechanisms are employed in discrete simulation. When the types of events encountered in the system are not very large, next event incrementing may be preferred. Since, in this case the system is examined only at those points in time, where some events happen, next event increment method proves economical. On the other hand in a complex system where the number of events occurring in a small interval of time is very large, it becomes difficult to develop a computer program by using the next event increment method. In such situations, the fixed time incrementing may prove more beneficial in programming as well as in computation time.

However, there is no yardstick to decide which method would be computationally more efficient for a given simulation model. The only way is experimentation, which is seldom justified by the amount of labour involved. The programmer's judgement and programming convenience are thus the most important criteria for the selection of time flow mechanism.

6.9 Example 6.1 — Time Oriented Simulation

A factory has a large number of semiautomatic machines. On 50% of the working days none of the machines fail. On 30% of the days one machine fails and on 20% of the days two machines fail. The maintenance staff on the average puts 65% of the machines in order in one day, 30% in two days and the remaining 5% in three days.

Simulate the system for 30 days duration and estimate the average length of queue, average waiting time, and the server loading, *i.e.*, the fraction of time for which server is busy.

Solution: The given system is a single server queuing model. The failure of the machines in the factory generates arrivals, while the maintenance staff is the service facility. There is no limit on the capacity of the system in other words on the length of waiting line. The population of machines is very large and can be taken as infinite.

Arrival pattern:
- On 50% of the days arrival = 0
- On 30% of the days arrival = 1
- On 20% of the days arrival = 2

Expected arrival rate = $0 \times .5 + 1 \times .3 + 2 \times .2 = 0.7$ per day.

Service pattern:
- 65% machines in 1 day
- 30% machines in 2 days
- 05% machines in 3 days

Average service time = $1 \times .65 + 2 \times .3 + 3 \times .05 = 1.4$ days

Expected service rate = $1/1.4 = 0.714$ machines per day.

The expected arrival rate is slightly less than the expected service rate and hence the system can reach a steady state. For the purpose of generating the arrivals per day and the services completed per day the given discrete distributions will be used.

Random numbers between 0 and 1 will be used to generate the arrivals as under.

- $0.0 < r <= 0.5$ Arrivals = 0
- $0.5 < r <= 0.8$ Arrivals = 1
- $0.8 < r <= 1.0$ Arrivals = 2

Similarly, random numbers between 0 and 1 will be used for generating the service times (ST).

- $0.0 < r <= 0.65$ ST = 1 day
- $0.65 < r <= 0.95$ ST = 2 days
- $0.95 < r <= 1.0$ ST = 3 days

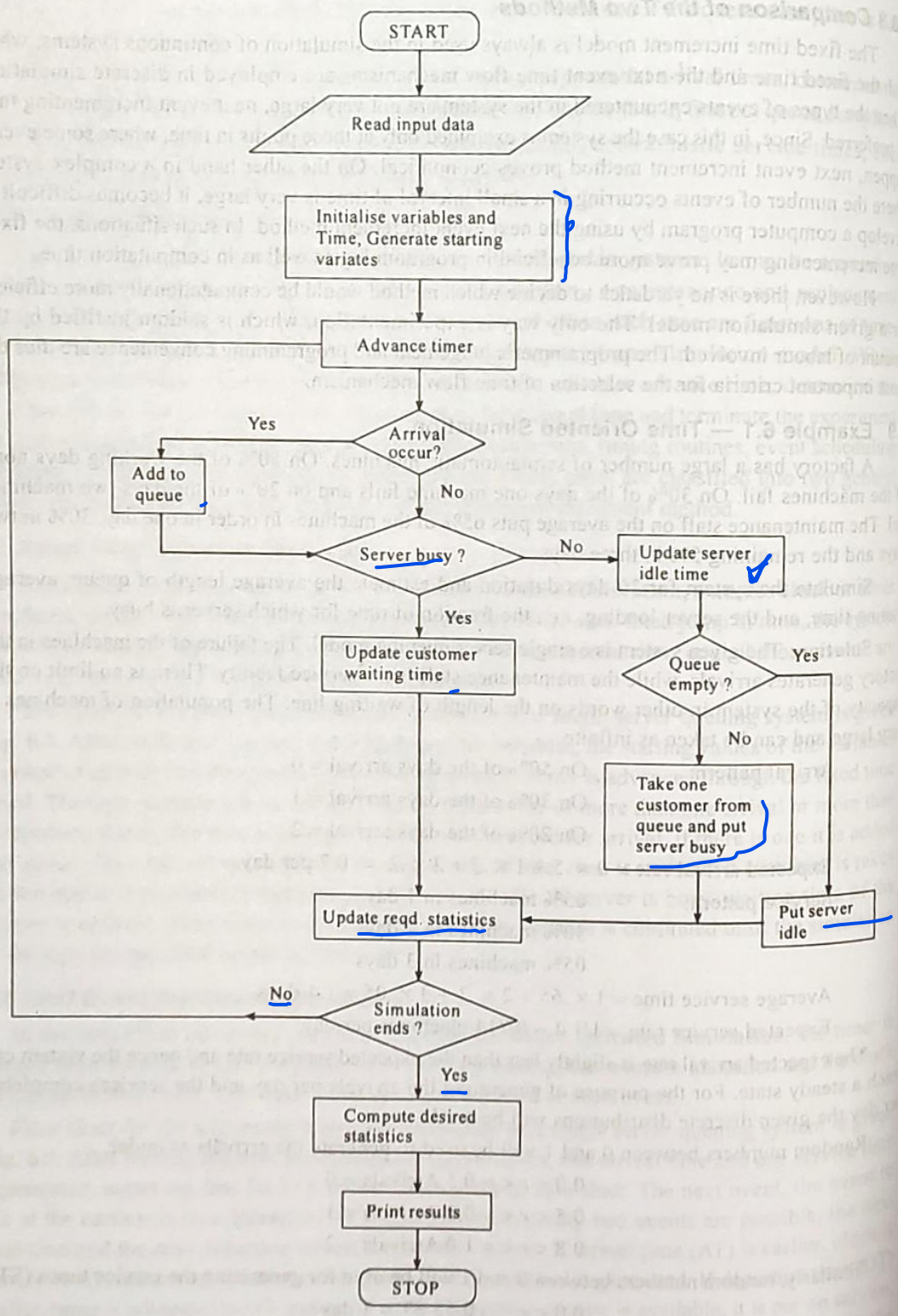
START
Read input data
Initialise variables and Time, Generate starting variates
Advance timer
Arrival occur?
Yes
Add to queue
No
Server busy ?
No
Update server idle time
Yes
Update customer waiting time
Queue empty ?
Yes
No
Take one customer from queue and put server busy
Update reqd. statistics
Put server idle
Simulation ends ?
No
Yes
Compute desired statistics
Print results
STOP

Fig. 6.4

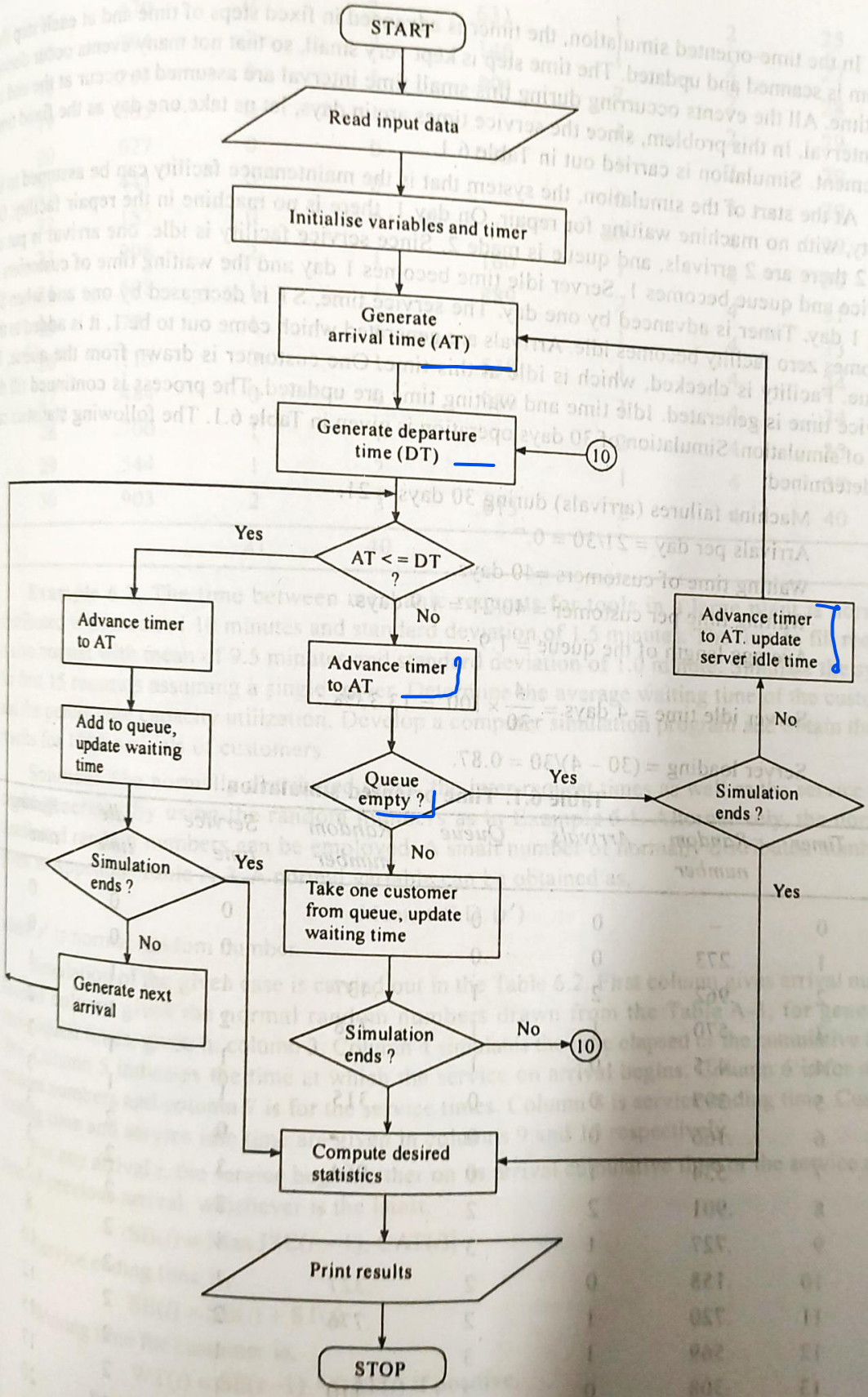
START
Read input data
Initialise variables and timer
Generate arrival time (AT)
Generate departure time (DT)
10
AT < = DT ?
Yes
No
Advance timer to AT
Add to queue, update waiting time
Advance timer to AT
Queue empty ?
Yes
No
Simulation ends ?
Yes
No
Generate next arrival
Take one customer from queue, update waiting time
Simulation ends ?
No
10
Advance timer to AT. update server idle time
No
Simulation ends ?
Yes
Compute desired statistics
Print results
STOP

Fig. 6.5

In the time-oriented simulation, the timer is advanced in fixed steps of time and at each step the system is scanned and updated. The time step is kept very small, so that not many events occur during this time. All the events occurring during this small time interval are assumed to occur at the end of the interval. In this problem, since the service times are in days, let us take one day as the fixed time increment. Simulation is carried out in Table 6.1.

At the start of the simulation, the system that is the maintenance facility can be assumed to be empty, with no machine waiting for repair. On day 1, there is no machine in the repair facility. On day 2 there are 2 arrivals, and queue is made 2. Since service facility is idle, one arrival is put on service and queue becomes 1. Server idle time becomes 1 day and the waiting time of customers is also 1 day. Timer is advanced by one day. The service time, ST is decreased by one and when ST becomes zero facility becomes idle. Arrivals are generated which come out to be 1, it is added to the queue. Facility is checked, which is idle at this time. One customer is drawn from the queue, its service time is generated. Idle time and waiting time are updated. The process is continued till the end of simulation. Simulation of 30 days operation is given in Table 6.1. The following statistics can be determined.

Machine failures (arrivals) during 30 days = 21.

Arrivals per day = 21/30 = 0.7

Waiting time of customers = 40 days.

Waiting time per customer = 40/21 = 1.9 days.

Average length of the queue = 1.9

Server idle time = 4 days = $\frac{4}{30} \times 100 = 13.33\%$

Server loading = (30 – 4)/30 = 0.87.

Table 6.1. Time oriented simulation

Timer	*Random number*	*Arrivals*	*Queue*	*Random number*	*Service time*	*Idle time*	*Waiting time*
0	–	0	0		0	0	0
1	.273	0	0		0	0	0
2	.962	2	1	.437	1	1	1
3	.570	1	1	.718	2	1	2
4	.435	0	1		1	1	3
5	.397	0	0	.315	1	1	3
6	.166	0	0		0	2	3
7	.534	1	0	.964	3	2	3
8	.901	2	2		2	2	5
9	.727	1	3		1	2	8
10	.158	0	2	.327	1	2	10
11	.720	1	2	.776	2	2	12
12	.569	1	3		1	2	15
13	.308	0	2	.110	1	2	17
14	.871	2	3	.469	1	2	20
15	.678	1	3	.462	1	3	23

16	.470	0	2	.631	1	2	25
17	.794	1	2	.146	1	2	27
18	.263	0	1	.801	2	2	28
19	.065	0	1		1	2	29
20	.027	0	0	086	1	2	29
21	.441	0	0		0	3	29
22	.152	0	0		0	4	29
23	.998	2	1	.160	1	4	30
24	.508	1	1	.889	2	4	31
25	.771	1	2		1	4	33
26	.115	0	1	.538	1	4	34
27	.484	0	0	.989	3	4	34
28	.700	1	1		2	4	35
29	.544	1	2		1	4	37
30	.903	2	3	.813	2	4	40
		21	40				

Example 6.2. The time between mechanic requests for tools in a large plant is normally distributed with mean of 10 minutes and standard deviation of 1.5 minutes. The time to fill requests is also normal with mean of 9.5 minutes and standard deviation of 1.0 minute. Simulate the system for first 15 requests assuming a single server. Determine the average waiting time of the customers and the percentage capacity utilization. Develop a computer simulation program and obtain the said results for 1000 arrivals of customers.

Solution: The normally distributed times, the inter-request times as well as the service times can be generated by using the random numbers as in Example 6.1. Alternatively, the normally distributed random numbers can be employed. A small number of normally distributed numbers is given in Appendix Table A-3. A normal variable can be obtained as,

$$x = \text{Mean} + \text{S.D.}(r')$$

where r' is normal random number.

Simulation of the given case is carried out in the Table 6.2. First column gives arrival number. Second column gives the normal random numbers drawn from the Table A-3, for generating inter-request times, given in column 3. Column 4 simulates the time elapsed or the cumulative arrival time. Column 5 indicates the time at which the service on arrival begins. Column 6 is for normal random numbers and column 7 is for the service times. Column 8 is service-ending time. Customer waiting time and service idle time are given in columns 9 and 10 respectively.

For any arrival i, the service begins either on its arrival cumulative time or the service ending time of previous arrival, whichever is the latest.

$$SB(i) = \text{Max.}[SE(i-1), CAT(i)]$$

Service ending time is,

$$SE(i) = SB(i) + ST(i)$$

Waiting time for customer is,

$$WT(i) = SE(i-1) - CAT(i) \text{ if positive.}$$

Service idle time is,

$$IT(i) = CAT(i) - SE(i-1), \text{ if positive.}$$

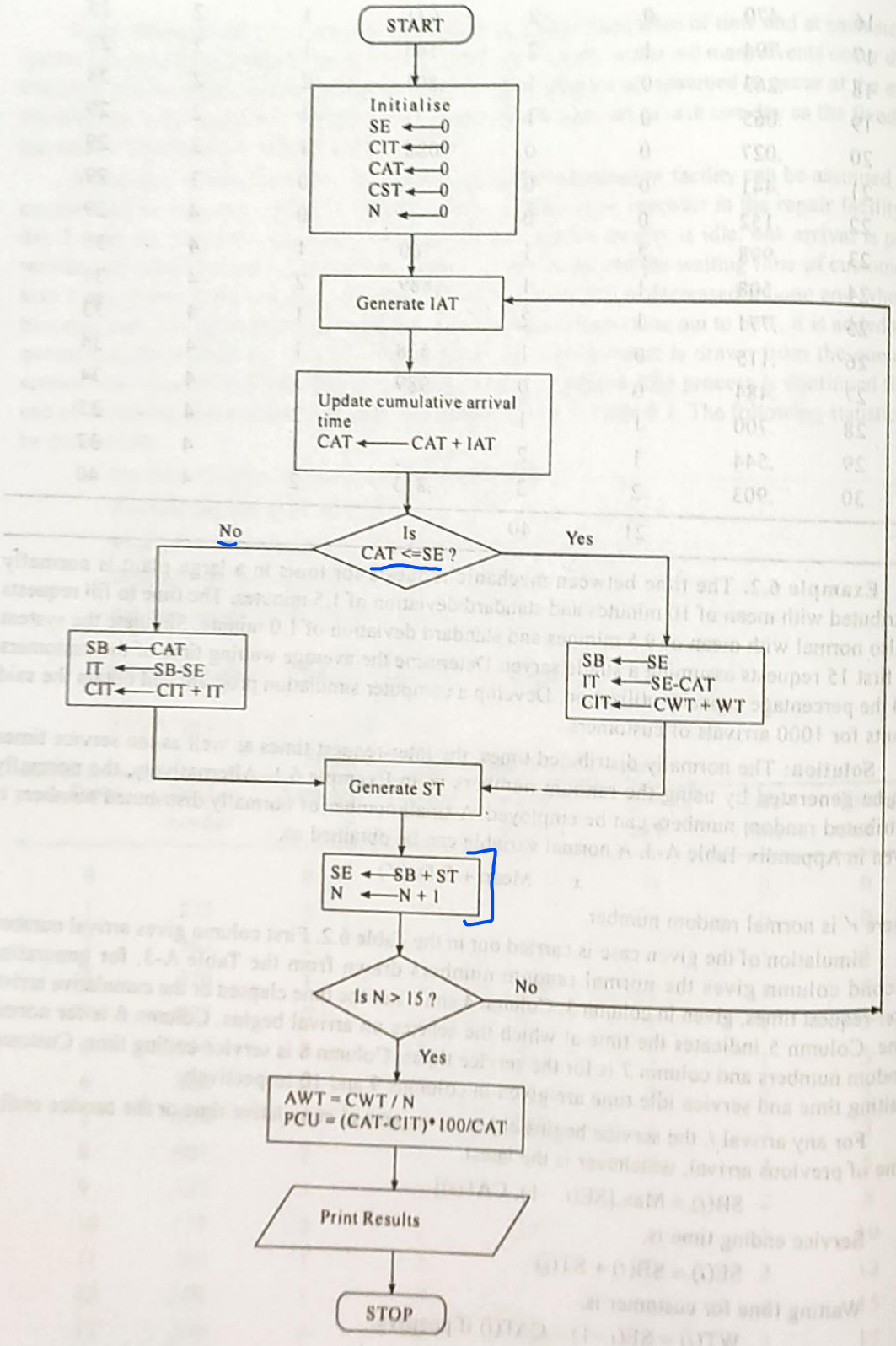
START
Initialise
SE ← 0
CIT ← 0
CAT ← 0
CST ← 0
N ← 0
Generate IAT
Update cumulative arrival time
CAT ← CAT + IAT
Is CAT <=SE ?
No
Yes
SB ← CAT
IT ← SB-SE
CIT ← CIT + IT
SB ← SE
IT ← SE-CAT
CIT ← CWT + WT
Generate ST
SE ← SB + ST
N ← N + 1
Is N > 15 ?
No
Yes
AWT = CWT / N
PCU = (CAT-CIT)*100/CAT
Print Results
STOP

Fig. 6.6

Flow chart for this simulation is given in Fig. 6.6.

At the end of 15 simulations,

Total time elapsed = 159.615 mins.

Total customer waiting time = 2.89 mins.

Total server idle time = 26.915 mins.

Since there are 15 arrivals,

Average waiting time = 2.89/15 = 0.193 per arrival.

Percentage capacity utilization = (167.755 – 26.915)/(167.755)*100 = 83.96%.

Table 6.2. Simulation of a single server queuing system

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Arrival number	*Normal random number*	*Time between arrivals*	*Cumm. arrival time*	*Service begins*	*Normal random number*	*Service time*	*Service ends time*	*Cust. waiting time*	*Server idle*
i	*r'*	*IAT*	*CAT*	*SB*	*r'*	*ST*	*SE*	*WT*	*IT*
1	–0.46	9.31	9.31	9.31	0.59	10.09	19.40	0.00	9.31
2	–1.15	8.275	17.585	19.40	–0.67	8.83	28.23	1.815	0.00
3	0.15	10.225	27.81	28.23	0.41	9.91	38.14	0.42	0.00
4	0.81	11.215	39.025	39.025	0.51	10.01	49.035	0.00	0.885
5	0.74	11.11	50.135	50.135	1.53	11.03	61.165	0.00	1.10
6	–0.39	9.415	59.55	61.165	–0.37	9.13	70.295	1.615	0.00
7	0.45	10.675	70.225	60.295	–0.27	9.23	79.525	0.07	0.00
8	2.44	13.66	83.885	83.885	–0.15	9.35	93.235	0.00	4.36
9	0.59	10.885	94.77	94.77	–0.02	9.48	104.25	0.00	1.535
10	–0.06	9.91	104.68	104.68	–1.60	7.90	112.58	0.00	0.43
11	0.09	10.135	114.815	114.815	0.19	9.31	124.125	0.00	2.235
12	0.56	10.84	125.655	125.655	0.16	9.66	135.315	0.00	1.53
13	0.65	10.97	136.625	136.625	–0.07	9.43	146.055	0.00	1.31
14	3.10	14.65	150.275	150.275	0.24	9.74	160.015	0.00	4.22
15	–0.44	9.34	159.615	160.015	–1.76	7.74	167.755	0.04	0.00
							Total	2.89	26.915

These 15 observations cannot be expected to give a reliable estimate of the results. Much longer simulation run is required which is possible only with a computer simulation of the system. A computer program for the single server queuing model is given below in which both the inter-arrival and service times have been taken as normally distributed.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
main()
   {
```

```
/*Single server queue:
Arrivals and service times are normally distributed.
Mean and standard deviation of arrivals are 10. and 1.5 minutes
mean and S.D. of service times are 9.5 and 1.0 minutes.*/
int kk,i,j,run=100;
float x,iat,st,awt,pcu,wt=0.,it=0. ;
    float mean=10. , sd=1.5 , mue=9.5 , sigma=1.0 ;
    float sb=0., se=0.,cit=0.,cat=0.,cwt=0.;
printf("\n  IAT    CAT     SB      ST     SE     CWT     CIT");
    for(j=1;j<=run;++j) {
    /* generate inter arrival time */
    float sum=0.;
    for(i=1;i<=12;++i) {
    x=rand()/32768.0;
    sum=sum+x; }
    iat=mean+sd*(sum-6.);
    cat=cat+iat;
    /*printf("\n iat= %6.2f cat=%6.2f",iat,cat);*/
    if(cat<=se) {sb=se;wt=se-cat;cwt=cwt+wt;}
    else{sb=cat; it=sb-se; cit=cit+it;}

    /*generate service time*/
    sum=0.;
    for(i=1;i<=12;++i) {
    x=rand()/32768.0;
    sum=sum+x;              }
    st=mue+sigma*(sum-6.);
    se=sb+st;
    printf("\n %5.2f %6.2f %6.2f %6.2f %6.2f%6.2f
%6.2f",iat,cat,sb,st,se,cwt,cit);
    }
    awt=cwt/run;
    pcu=(cat-cit)*100./cat;
    printf("\n Average waiting time=%6.2f",awt);
    printf("\n Percentage Capacity Utilisation=%6.2f",pcu);
  printf("\n any digit");
    scanf("%d",kk);
    }
```

The output of this program for 1000 arrivals is as below.

Inter-arrival time: Mean = 10.00 S.D.= 1.5 minutes

Service time: Mean = 9.5 S.D. = 1.0 minutes

Length of run (Max. Arrivals) = 1000

Average waiting time = 1.65 minutes

Percentage capacity utilization = 94.68%.

6.10 Example 6.3. Event Oriented Simulation

Simulate an M/M/1/∞ queuing system with mean arrival rate as 10 per hour and the mean service rate as 15 per hour, for a simulation run of about 3 hours. Determine the average customer waiting time, percentage idle time of the server, maximum length of the queue and average length of queue.

Solution: The given queuing system has exponential arrival and exponential service distributions. There is single server and the capacity of the system is infinite.

Mean arrival rate = 10/hr or 1/6 per minute.

Mean service rate = 15/hr or 1/4 per minute.

In actual simulation, the inter-arrival times and the service times are generated, as and when the particular events occur. However, to simplify the simulation table, the inter-arrival times for the first 25 arrivals and their service times (ST) are generated in Table 6.3. The random numbers (R) have been taken from a random number table.

Table 6.3

S. No.	*Inter-arrival times* $IAT = -6\ ln\ (1 - r)$		*Service times* $ST = -4\ ln\ (1 - r)$	
	R	*IAT*	*R*	*ST*
1	.100	0.63	.209	0.94
2	.375	2.82	.048	0.20
3	.084	0.53	.689	4.67
4	.990	27.63	.025	0.10
5	.128	0.82	.999	27.63
6	.660	6.47	.747	5.50
7	.310	2.23	.108	0.46
8	.852	11.46	.776	5.98
9	.659	6.45	.321	1.55
10	.737	8.01	.457	2.44
11	.985	25.20	.177	0.78
12	.118	0.75	.054	0.22
13	.840	10.99	.996	22.09
14	.886	13.03	.402	2.06
15	.995	31.79	.673	4.47
16	.654	6.37	.176	0.77
17	.801	9.69	.947	11.76
18	.743	8.15	.914	9.83
19	.159	1.04	.356	1.76
20	.283	2.00	.268	1.25
21	.699	7.20	.205	0.92
22	.098	0.62	.268	24.86
23	.802	9.71	.951	12.09
24	.813	10.05	.885	8.65
25	.543	4.70	.437	2.30

Simulation of the system is illustrated in Table 6.4. Event to event time flow mechanism has been employed. System is assumed to be empty at zero time. First arrival takes place at 0.63 minutes; clock is advanced to that time. Service on the customer is generated and the same ends at 1.57 mins. The next arrival is to occur at 3.45 mins. Clock is advanced to the earliest event, *i.e.*, to 1.57 mins.

As soon as an arrival occurs, the server is checked, if free, service on customer begins, otherwise, it is added to queue. As soon as a service ends, the queue is checked, if there is a customer waiting, service on the customer begins, otherwise the server becomes idle. For example, at time 31.61 service on a customer begins, while the previous service ended at 8.65 minutes giving 31.61 – 8.65 = 22.96 mins waiting time. The waiting time of customers is the number of customers waiting, multiplied by the incremental time. For example, between 41.3 and 52.59 minutes, 2 customers are in queue, giving waiting time 2 (52.29 – 41.13) = 22.92 minutes.

Flow chart for the given simulation is given in Fig. 6.7. It simulates the cumulated server idle time and the cumulated customer waiting time from which various results can be computed.

Following statistics are obtained from the simulation Table 6.4.

Time elapsed or Simulation run =183.26 mins.

Number of arrivals = 22

Cumulated waiting time for customers = 106.55 mins.

Cumulated idle time of server = 85.01 mins.

Taking the different steps in time as the observations, the number of observations made is 39.

Length of the queue has been observed to be

0 on 23 times
1 on 6 times
2 on 5 times
3 on 4 times
4 on 1 time

From this information, following results are obtained.

(*a*) Average waiting time = 106.55/22 = 4.84 mins.

(*b*) Server idle time = (85.01/183.26) 100 = 46.39%

(*c*) Maximum length of the queue = 4

(*d*) Average length of the queue = $(23 \times 0 + 6 \times 1 + 5 \times 2 + 4 \times 3 + 4)/39 = 32/39 = 0.82$

Table 6.4. Event oriented simulation

Clock time	*Inter-arrival time IAT*	*Next arrival time NAT*	*Queue Q*	*Service begins SB*	*Service time ST*	*Service ends SE*	*Server idle time IT*	*Customer waiting time WT*
0.00	0.63	0.63	0	–		0.00	0.00	0.00
0.63	2.82	3.45	0	0.63	0.94	1.57	0.63	–
1.57		3.45	0					
3.45	0.53	3.98	0	3.45	0.20	3.65	1.88	–
3.65		3.98	0					
3.98	27.63	31.61	0	3.98	4.67	8.65	0.33	–
8.65		31.61	0					
31.61	0.82	32.43	0	31.61	0.10	31.71	22.96	–

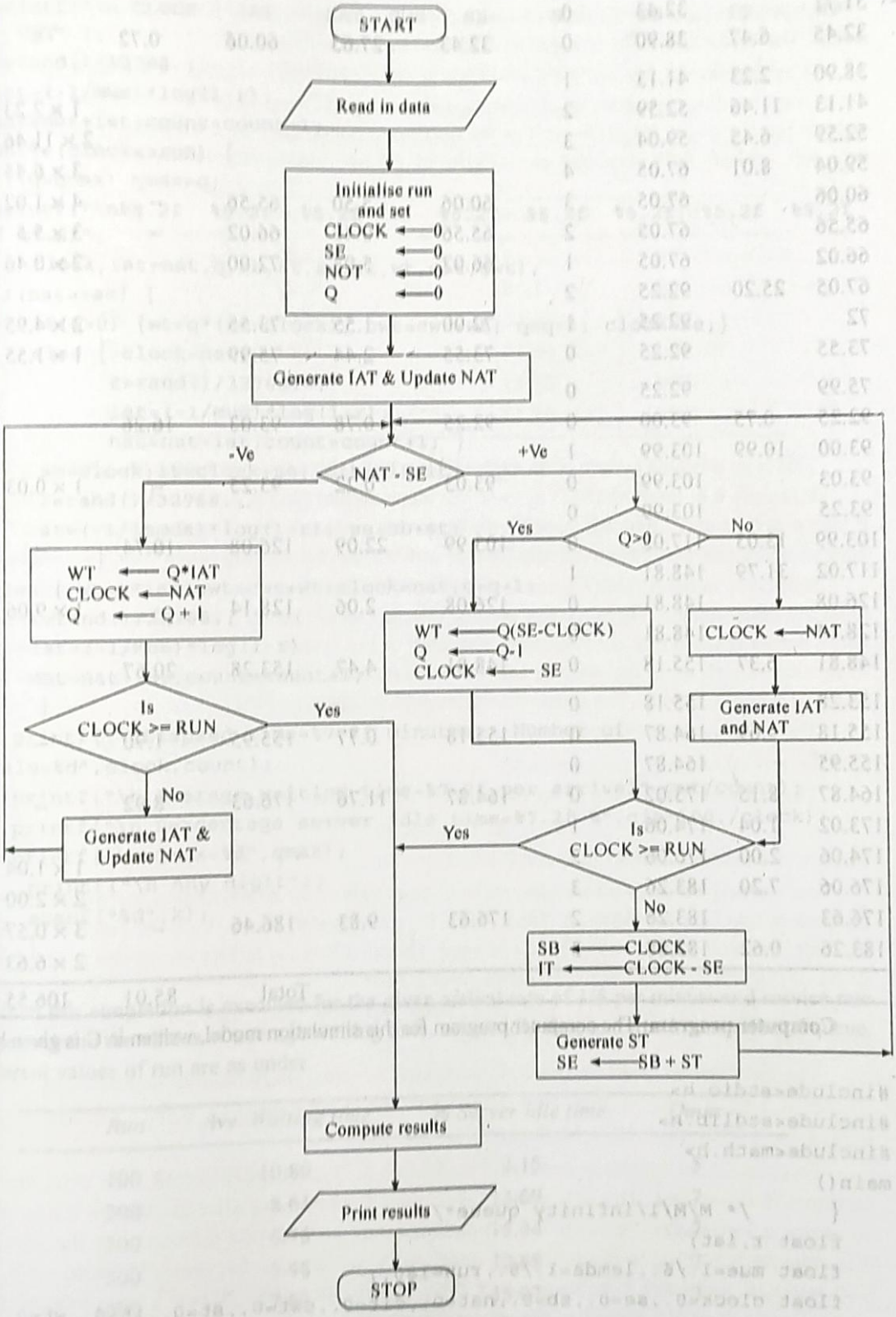
START
Read in data
Initialise run and set
CLOCK ← 0
SE ← 0
NOT ← 0
Q ← 0
Generate IAT & Update NAT
-Ve
NAT - SE
+Ve
WT ← Q*IAT
CLOCK ← NAT
Q ← Q + 1
Is CLOCK >= RUN
Yes
No
Generate IAT & Update NAT
Yes
Q>0
No
WT ← Q(SE-CLOCK)
Q ← Q-1
CLOCK ← SE
CLOCK ← NAT
Generate IAT and NAT
Yes
Is CLOCK >= RUN
No
SB ← CLOCK
IT ← CLOCK - SE
Generate ST
SE ← SB + ST
Compute results
Print results
STOP

Fig. 6.7

31.71		32.43	0					
32.45	6.47	38.90	0	32.43	27.63	60.06	0.72	–
38.90	2.23	41.13	1					
41.13	11.46	52.59	2					
52.59	6.45	59.04	3					1 × 2.23
59.04	8.01	67.05	4					2 × 11.46
60.06		67.05	3	60.06	5.50	65.56	–	3 × 6.45
65.56		67.05	2	65.56	0.46	66.02	–	4 × 1.02
66.02		67.05	1	66.02	5.98	72.00	–	3 × 5.5
67.05	25.20	92.25	0					2 × 0.46
72		92.25	1	72.00	1.55	73.55	–	2 × 4.95
73.55		92.25	0	73.55	2.44	75.99	–	1 × 1.55
75.99		92.25	0					
92.25	0.75	93.00	0	92.25	0.78	93.03	16.26	–
93.00	10.99	103.99	1					
93.03		103.99	0	93.03	0.22	93.25	–	1 × 0.03
93.25		103.99	0					
103.99	13.03	117.02	0	103.99	22.09	126.08	10.74	–
117.02	31.79	148.81	1					
126.08		148.81	0	126.08	2.06	128.14	–	1 × 9.06
128.14		148.81	0					
148.81	6.37	155.18	0	148.81	4.47	153.28	20.67	–
153.28		155.18	0					
155.18	9.69	164.87	0	155.18	0.77	155.95	1.90	–
155.95		164.87	0					
164.87	8.15	173.02	0	164.87	11.76	176.63	8.92	–
173.02	1.04	174.06	1					
174.06	2.00	176.06	2					1 × 1.04
176.06	7.20	183.26	3					2 × 2.00
176.63		183.26	2	176.63	9.83	186.46	–	3 × 0.57
183.26	0.62	183.88	3					2 × 6.63
						Total	85.01	106.55

Computer program: The computer program for this simulation model, written in C is given below.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
main()
    {       /* M/M/1/infinity queue*/
    float r,iat;
    float mue=1./6.,lemda=1./5.,run=180.;
    float clock=0.,se=0.,sb=0.,nat=0.,cit=0.,cwt=0.,st=0.,it=0.,wt=0.;
    int q=0,cq=0,k,count=0,qmax=0;
```

```
    printf("\n CLOCK    IAT    NAT   QUE    SB     ST     SE     IT     WT
CIT    CWT" );
    r=rand()/32768.;
    iat=(-1/mue)*log(1-r);
    nat=nat+iat;count=count+1;
    while(clock<=run) {
    if(q>qmax) qmax=q;
    printf("\n%5.2f   %5.2f  %5.2f %3d  %5.2f  %5.2f  %5.2f  %5.2f  %5.2f
%5.2f %5.2f",
       clock,iat,nat,q,sb,st,se,it,wt,cit,cwt);
    if(nat>=se) {
       if(q>0) {wt=q*(se-clock); cwt=cwt+wt; q=q-1; clock=se;}
       else { clock=nat;
              r=rand()/32768.0;
              iat=(-1/mue)*log(1-r);
              nat=nat+iat;count=count+1; }
       sb=clock;it=clock-se; cit=cit+it;
       r=rand()/32768.;
       st=(-1/lemda)*log(1-r); se=sb+st;
       }
    else { wt=q*iat;cwt=cwt+wt;clock=nat;q=q+1;
       r=rand()/32768.;
       iat=(-1/mue)*log(1-r);
       nat=nat+iat;count=count+1; }
       }
     printf("\nElapsed time=%7.2f minutes    Number of
arrivals=%d",clock,count);
     printf("\n Average waiting time=%7.2f per arrival",cwt/count);
     printf("\n Percentage server idle time=%7.2f %",cit*100./clock);
     printf("\n Qmax=%d",qmax);
      printf("\n Any digit");
      scanf("%d",k);
     }
```

When this simulation is executed for the given arrival rate of 1/6 per minute and service rate of 1/4 per minute the values of average waiting time, server idle time and maximum value of queue, for different values of run are as under.

Run	*Ave. Waiting time*	*% Server idle time*	*Qmax*
100	10.80	2.15	5
200	8.61	13.69	7
300	6.73	15.34	7
500	5.48	32.68	7
1000	7.63	35.07	7
2000	7.74	32.59	7

Thus the system stabilizes after some time with average waiting time of about 7.65 minute per arrival, the server idle time of about 35.00% and maximum queue at 7.

When the arrival rate and the service rate are made identical, because of the variability in times the average waiting time goes on increasing with passage of time while the server idle time goes on decreasing, the queue length as well as the maximum queue goes on increasing. The output of the program, for both the arrival and service rates of 1/6 per minute, is given below.

Run	*Ave. waiting time*	*% Server idle time*	*Qmax*
100	3.15	33.47	2
200	11.28	20.52	11
300	29.56	14.65	16
500	66.14	8.81	17
1000	61.32	7.29	17
2000	68.13	7.02	20

This is an unstable system or an explosive system, where the queue goes on building with time.

Example 6.4. In an M/D/2/3 system, the mean arrival time is 3 minutes and the Servers I and II take exactly 5 and 7 minutes respectively to serve a customer. Simulate the system for the first one hour of operation. Determine the idle time of servers and the waiting time of the customers.

Solution: In the M/D/2/3 system, the arrivals are distributed exponentially, while the services times are deterministic. There are two servers, Server I takes exactly 5 minutes to serve the customer, while Server II takes exactly 7 minutes. Capacity of the system is 3, *i.e.*, the number of customers in service as well as in the waiting line cannot exceed 3. The next customer in that case will be returned without service. The queue discipline will be taken as FIFO.

The mean inter-arrival time = 3 mins.

Or arrival rate $\lambda = 1/3$ per minute

The random observation x, for exponential distribution is generated as,

$$x = (-1/\lambda) \times \ln (1 - r)$$

where r is a random number between 0 and 1.

For operation time of 60 minutes, we will require about 20 arrivals; let us generate 25, for which 25 random numbers are required. Since r and $1 - r$ are equally likely, both the random number and its complementary random number can be used. This also helps to reduce the variance. The following string of random numbers has been taken from a table.

.218, .782, .119, .881, .711, .289, .344, .420, .580, .656, .354, .696, .175, .825, .913, .350, .087, .076, .944, .650, .130, .670, .440, .560, .569.

The inter-arrival times (in minutes) generated by using these random numbers are.

0.74, 4.57, 0.38, 6.39, 3.72, 1.02, 1.26, 1.63, 2.60, 3.20, 1.09, 3.12, 0.58, 5.23, 7.32, 1.29, 0.28, 0.24, 8.65, 3.15, 1.20, 6.12, 1.74, 2.46, 2.52.

Working of the system is simulated in Table 6.5. It is assumed that at zero time, there is no customer in the system. The first arrival takes place 0.74 minutes after the system starts. The first customer is assigned to Server I. The second customer arrives 4.57 minutes after the first, *i.e.*, at 5.31 minutes according to the simulated clock, and is assigned to Server II. Third arrival occurs at 5.69 minutes, and since both servers are busy it waits in queue. Server I completes service on customer one at 5.74 minutes, and customer 3 moves to Server I. This process continues. When customer 8 arrives, Server I is busy with customer 6 and Server II is busy with customer 5, while

customer 7 is in queue. The system being full to capacity, customer number 8 returns without getting service. Similar situation occurs at clock times 31.02 and 45.38 minutes. When both servers are idle as at times 53.57 and 64.18 mins, the next arrival will go to the faster server, *i.e.*, Server I.

Table 6.5. Event oriented simulation

Clock	*Arrival No.*	*IAT*	*NAT*	*Servr I cstmr. in service*	*SE1*	*IT1*	*Servr II cstmr. in service*	*SE2*	*IT2*	*Cstmr. in queue*	*Cstmr. wtng time*
0.00	1	0.74	0.74	0	–	–	0	–	–	–	–
0.74	2	4.57	5.31	1	5.74	0.74	0	–	–	–	–
5.31	3	0.38	5.69	1			2	12.31	5.31	–	–
5.69	4	6.39	12.08	1			2			3	–
5.74				3	10.74	0.00	2			–	0.05
10.74							2			–	–
12.08	5	3.72	15.80	4	17.08	1.34	2			–	–
12.31				4						–	–
15.80	6	1.02	16.82	4			5	22.80	3.49	–	–
16.82	7	1.26	18.08	4			5			6	–
17.08				6	22.08	0.00	5			–	0.26
18.08	8	1.63	19.71	6			5			7	–
19.71	9	2.60	22.31	6			5			7,8	–
22.08				7	27.08	0.00	5			–	4.00
22.31	10	3.20	25.51	7			5			9	–
22.80				7			9	29.80	0.00	–	0.49
25.51	11	1.09	26.60	7			9			10	–
26.60	12	3.12	29.72	7			9			10,11	–
27.08				10	32.08	0.00	9			–	1.57
29.72	13	0.58	30.30	10			9			12	–
29.80				10			12	36.80	0.00	–	0.08
30.30	14	5.23	35.53	10			12			13	–
32.08				13	37.08	0.00	12			–	1.78
35.53	15	7.32	42.87	13			12			–	–
35.53	15	7.32	42.87	13			12			14	–
36.80				13			14	43.80	0.00	–	1.27
37.08				–			14			–	–
42.87	16	1.29	44.16	15	47.87	5.79	14			–	–
43.80				15			–			–	–
44.16	17	0.28	44.44	15			16	51.16	7.36	–	–
44.44	18	0.24	44.68	15			16			17	–
44.68	19	8.65	53.33	15			16			17,18	–
47.87				17	52.87	0.00	16			–	3.43

51.16				17			–			–	–
52.87				–			–			–	–
53.33	20	3.15	56.48	19	58.33	5.46	–			–	–
56.48	21	1.20	57.68	19			20	63.48	5.32	–	–
57.68	22	6.12	63.80	19			20			21	–
58.33				21	63.33	0.00	20			–	0.65
63.33				–			20			–	–
63.48				–			–			–	–
63.80	23	1.74	65.54	22	68.80	0.47	–			–	–
Total						13.80			21.48		13.58

From the simulation in Table 6.5, we observe that in the first 63.80 minutes of operation 21 – 3 = 18 customers have been served, while 3 customers have been returned without service. The idle time of Server I is 13.80 minutes, while that of Server II is 21.48 minutes. The customer had to wait in line a total of 13.58 minutes. Thus, we find that while about 31.71% of the time, there had been a customer in the queue, Server I remained idle for about 21.63% of the time and Server II for about 33.67% of the time.

State of the system at 63.80 mins is,

Number of customers arrived = 22

Customers in service= one (number 22)

Number of customers returned without service = 3 (Nos. 8, 11 and 18)

Number of customers served = 21 – 3 = 18

Idle time of Server I = 13.80 minutes

Idle time of Server II = 21.48 minutes

Out of 41 observations, there was queue on 13 times

Waiting time of customers =13.58 mins.

% Idle time of Server I = (13.80/63.80) 100 = 21.63%

% Idle time of Server II = (21.48/63.80) 100 = 33.67%

The computer simulation program of this M/D/2/3 model is given below.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
main()
{     /*M/D/2/3 queuing system*/
    float r,iat,clock,nat,it1,it2,run=150.,cit1=0.,cit2=0.;
    float mean=3.,lemda1=5.,lemda2=4.,se1=0.,se2=0.;
    int k,q=0, qmax=3,kont=0,counter;
    printf("\n  CLOCK    IAT     NAT     SE1     SE2   QUE   KONT cit1   cit2");
    /*generate first arrival*/
    r=rand()/32768.;
    iat=( -mean)*log(1-r);
    nat=nat+iat; se1=lemda1;   counter=1;

    printf("\n %6.2f %6.2f %6.2f   %6.2f %6.2f   %d     %d %6.2f %6.2f ",
          clock,iat,nat,se1,se2,q,kont,cit1,cit2);
    while(clock<=run) {
```

```
    if(nat<=se1 && nat<=se2) {
       clock=nat;q=q+1;
       r=rand()/32768.;
       iat=(-mean)*log(1-r);
       nat=nat+iat; counter=counter+1;}
    else if(se1<= nat && se1<=se2) clock=se1;
         else clock=se2;

    if(q>qmax){kont=kont+1;
       q=q-1;}
    if(q>=1 && se1<=clock) {
       it1=clock-se1; cit1=cit1+it1;
       se1=clock+lemda1;
       q=q-1;}

    if(q>=1 && se2<=clock) {
       it2=clock-se2; cit2=cit2+it2;
       se2=clock+lemda2;q=q-1;}
    if(q==0 && se1<=clock) {
       clock=nat;it1=clock-se1;cit1=cit1+it1;
       se1=nat+lemda1;
       r=rand()/32768.;
       iat=(-mean)*log(1-r);
       nat=nat+iat; counter=counter+1;}
    if(q==0 && se2<=clock) {
       clock=nat;it2=clock-se2;cit2=cit2+it2;
       se2=nat+lemda2;
       r=rand()/32768.;
       iat=(-mean)*log(1-r);
       nat=nat+iat; counter=counter+1;
       }

    printf("\n %6.2f %6.2f %6.2f  %6.2f %6.2f  %d    %d %6.2f %6.2f ",
            clock,iat,nat,se1,se2,q,kont,cit1,cit2);
    }
    printf("\n clock=%8.2f cit1=%6.2f cit2=%6.2f counter=%d",
            clock,cit1,cit2,counter);
    printf("\n\n                    Queuing System:M/D/2/3");
    printf("\n\n Mean arrival time=%5.2f minutes exponentially
distributed",mean);
    printf("\n Service time Server I=%5.2fminutes   Server II=%5.2f
minutes"
            lemda1,lemda2);
    Printf("\n Simulation Run (Elapsed time)=%7.2f minutes",clock);
    Printf("\n Number of customers arrived=%d",counter);
    Printf("\n Number of customers returned without service=%d",kont);
    Printf("\n Idle Time of Server I=%6.2f minutes",cit1);
```

```
    printf("\n Idle time of Server II=%6.2f minutes",cit2);
    printf("\n Percentage Idle time of Server I=%6.2f %",cit1*100./
clock);
    printf("\n Percentage Idle time of Server II=%6.2f %",cit2*100./
clock);


    printf("\n any digit");
    scanf("%d",&k);
    }
```

The output of this program when executed for 1000 minutes is given below, which shows that the idle time of the two servers tends to be identical in the long run.

Queue M/D/2/3

Mean arrival time = 3.00 minutes exponentially distributed.
Service time Server I = 5.00 minutes Server II = 7.00 minutes.
Simulation run (Elapsed time) = 1001.13 minutes.
Number of customers arrived = 334.
Number of customers returned without service = 45.
Idle time of Server I = 177.72 minutes.
Idle time of Server II = 151.33 minutes.
Percentage Idle time of Server I = 17.75%
Percentage Idle time of Server II = 15.12%

6.11 Two Servers in Parallel Queuing System

The servers placed in parallel may be doing identical service as the railway ticket-windows or different types of service as clinics in OPD of a hospital, but all the servers draw customers from a single queue at the entry to the system. In systems, where different type of service is provided at parallel servers, the customers drawn from the single queue are sorted into different types and put further into different queue one before each server.

Let us consider the manufacturing system shown below in Fig 6.8, where a mixture of components A and B pass through workstation I. These are then sorted into two parts, the components A that are 90% and the components B that are 10%. Then components A are processed at workstation A and components B are processed at workstation B. Station I processes the components at a rate of one in 5 minutes with exponential distribution. Station A and B has normally distributed processing times with mean values of 4.0 and 10.0 minutes and standard deviations of 2.0 and 5.0 minutes respectively. There is no dearth of components before workstation I, and unlimited space is available before workstations A and B. Sorting and transportation consume negligible time. What is the percentage idle time of workstations A and B and what is the maximum length of queue before each of A and B?

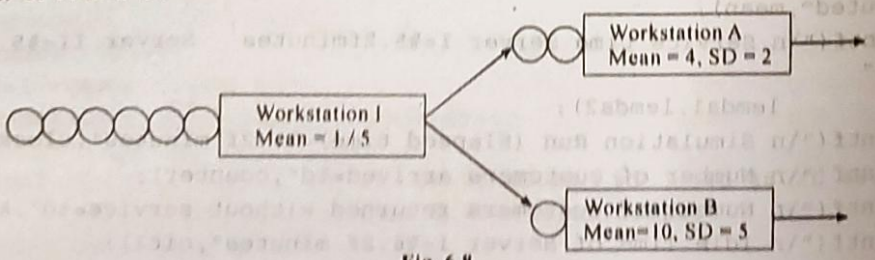

Fig. 6.8

Since there is no shortage of components at the head of workstation I, the workstation is always busy and have zero idle time. The processing rate of this workstation can be taken as the arrival rate of customers to the parallel Servers A and B from an infinite population. Each component processed at I is checked whether it is of type A or of type B and is added to the respective queue. The two work stations work independently, behaving like single server queues. The computer simulation model of this system, developed in C language, is given below.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
main()
   {
   /* Simulation of a system having two servers in parallel.
      mue is arrival rate,
      meana and siga are mean and standard deviation
      of service time of server A,
      meanb and sigb are for server B.
      count is counter of arrivals while konta and
      kontb are counters for components served by A and B.
      qa(qb) is the queue length of components A(B)
      sea(seb) is the service ending on a component A(B)
      iat and nat are inter arrival time and next arrival time.
      wta(wtb) is the waiting time of components in queue A(B)
      ita(itb) is the idle time of server A(B).
      sta(stb) is the service time of server A(B).*/
      int i,count=0,konta=0,kontb=0,qa=0,qb=0;
      float r,sea=0.,seb=0.,iat,nat=0.,wta=0.,wtb=0.,
      clock=0.,ita=0.,itb=0.;
      float delt=0.1,run;
      float mue,meana,meanb,siga,sigb,time,sum,sta,stb;
      int qamax=0,qbmax=0;
      mue=1./5.;meana=4.0;siga=2.0;meanb=10.;sigb=5.;
      printf("\n\n Mean of expo.processing time of stationI=%6.2f",mue);
      printf("\n Processing time of station A: mean=%6.2f SD=%6.2f",
         meana,siga);
      printf("\n Processing time of station B: mean=%6.2f SD=%6.2f",
         meanb,sigb);

      printf("\n Enter the value of run");
      scanf("%f",&run);

      while(clock<=run) {
      /* Check the state of arrival and update queues*/
      if(clock>=nat)
       {
      /* Generate next arrival*/
      count=count+1;
```

```
        r=rand()/32768.;
        iat=(-1/mue)*log(1-r);
        nat=clock+iat;
        /* Sort into A andB and add to respective queue*/
        r=rand()/32768.;
        if(r<=0.9) qa=qa+1;
        else qb=qb+1;
        if(qa>qamax) qamax=qa;
        if(qb>qbmax) qbmax=qb;
        }
        /*   printf("\nclock=%6.2f iat=%6.2f nat=%6.2fqa=%d qb=%d
wta=%6.2f sta=%6.2f sea=%6.2f",
          clock,iat,nat,qa,qb,wta,sta,sea); */
        /* check the state of the servers */
        if(clock>=sea)
        {
        wta=wta+qa*delt;
       if(qa>=1)
       {
       qa=qa-1;
        /*  Generate service time*/
       sum=0.;
       for(i=1;i<=12;++i){
       r=rand()/32768.;
       sum=sum+r;          }
       sta= meana+siga*(sum-6.);
       sea=clock+sta;
       konta=konta+1;
       }
         else ita=ita+delt;
        }
         if(clock<sea) wta=wta+qa*delt;
         /* Check state of srever B*/
         if(clock>=seb)
        {
        wtb=wtb+qb*delt;
       if(qb>=1) {
       qb=qb-1;
         /*  Generate service time*/
       sum=0.;
       for(i=1;i<=12;++i){
       r=rand()/32768.;
       sum=sum+r;          }
       stb= meanb+sigb*(sum-6.);
       seb=clock+stb;
       kontb=kontb+1;
          }
```

```
        else itb=itb+delt;
         }
        if(clock<seb) wtb=wtb+qb*delt;
        clock=clock+delt;

        }
      printf("\n clock=%6.2f count=%d  Kont A=%d  Kont B=%d",
          clock,count,konta,kontb);
      printf("\nIdle time of server A= %6.2f%",100.*ita/clock);
      printf("\nIdle time of server B= %6.2f%",100.*itb/clock);
      printf("\nAverage waiting time of components A= %6.2fper
component",
          wta/konta);
      printf("\nAverage waiting time of components B= %6.2f per
component",
          wtb/kontb);
      printf("\n Maximum number in buffer A=%d",qamax);
      printf("\n Maximum number in buffer B=%d",qbmax);
      }
```

An output of this program for a run of 1000 minutes is given below.

Clock = 1000 count = 214 Kont A = 198 Kont B = 16

Idle time of workstation A = 19.58%

Idle time of workstation B = 82.51%

Average waiting time of components A = 7.88 per component.

Average waiting time of components B = 0.69 per component.

Maximum number in buffer A = 6.

Maximum number in buffer B = 1.

The simulation run of 1000 may not be sufficient. As the simulation run is increased the values of statistics recorded vary and stabilize at about 50,000 minutes length of run. The variation of the results with length of simulation run is given below.

Length of run	*1000*	*2000*	*5000*	*10000*	*20000*	*50000*
Idle time of A (%)	19.58	23.81	28.06	27.77	26.44	26.69
Idle time of B (%)	82.51	79.12	81.02	81.41	80.11	79.59
AWTCA	7.88	6.03	5.68	5.58	6.40	6.46
AWTCB	0.69	1.44	1.66	1.39	1.30	1.42
Max. buffer A	6	6	6	7	9	11
Max. buffer B	1	1	2	2	2	3

Example 6.5. Dr. Strong is a dentist who schedules all her patients for 30 minutes appointments. Some of the patients take more or less than 30 minutes depending upon the type of dental work to be done. The following table shows the various categories of work, their probabilities and the time actually needed to complete the work.

Category	*Time required in minutes*	*Probability*
Filling	45	25
Crowning	60	15
Cleaning	15	25
Extraction	45	10
Checkup	15	25

Simulate the dentist's clinic for about four hours and determine the average waiting time for the patients and the percentage idle time for the dentist. Assume that all the patients show up at the clinic at exactly their scheduled arrival time starting at 8:00 AM. Use the following random numbers for simulating the process: 40, 82, 11, 34, 52, 66, 17 and 70. **[P.T.U. B.Tech. (Prod.), Dec. 2006]**

Solution: The dentist's clinic represents a single server queuing system, where the arrival times of the patients are known to be at 30 minutes interval. The service times are random and can be determined using the given probability distribution and the sequence of random numbers. The category of work, service time required, probability of occurrence of the category and the cumulative probability are given in table below. The last column of the table gives the range of random numbers corresponding to each category of work.

Category of work	*Time required*	*Probability*	*Comm. Prob.*	*Random numbers*
Filling	45	25	25	00 – 24
Crowning	60	15	40	25 – 39
Cleaning	15	25	65	40 – 64
Extraction	45	10	75	65 – 74
Checkup	15	25	100	75 – 99

Now using the given sequence of random numbers, the service time of patients can be generated. The first random number is 40, which lies in the range 40 to 64, which corresponds to cleaning, for which time required is 15 minutes. Thus the service time for 1st patient is 15 minutes. The service times for the 8 patients are given below:

Patient No.	1	2	3	4	5	6	7	8
Random No.	40	82	11	34	52	66	17	70
Service time	15	15	45	60	15	45	45	45

The simulation of Dr. Strong's clinic is carried out in Table 6.6. Since the patients are scheduled at 30 minutes interval, total 8 patients will arrive at the clinic in 4 hrs. The simulation table comprises of columns for patients number, arrival time, service begin time, service end time, dentist's idle time, and patient waiting time.

Patient number 1, arrives exactly at 8:00 AM, service begins immediately, service time for first patient being 15 minutes, service ends at 8:15. Dentist's idle time is zero, and patient waiting time is zero. Patient two arrives after 30 minutes at 8.30 AM. The service on this patient begins at 8.30, causing dentist to ideal for 15 minutes. Service time again is 15 minutes. Service ends at 8:45 AM. There is no waiting time from the second patient. Similarly, third patient arrive at 9:00, service begins at 9:00, dentist is idle for 15 minutes. Service on 3rd patent ends at 9:45. The fourth patient arrives at his scheduled time of 9:30 but service can being only at 9:45, when the doctor is free from the previous patient. Thus 4th patient has to wait for 15 minutes.

The service on 7th patient begins at 11:45 and ends at 12:30. Since the simulation is to be done for about 4 hrs, it can be stopped here. Thus in a simulation of 270 minutes, seven patients have been serviced. Idle time of the dentist is 30 minutes, or $\frac{30}{270} \times 100 \approx 11\%$. The waiting time of all the patients is 135 minutes, which amounts to an average of $\frac{135}{7} \approx 19$ minutes per patient.

Table 6.6

Patient number	*Arrival time*	*Service begin time*	*Service time*	*Service end time*	*Dentist's idle time*	*Patient waiting time*
1.	8:00	8:00	15	8:15	00	00
2.	8:30	8:30	15	8:45	15	00
3.	9:00	9:00	45	9:45	15	00
4.	9:30	9:45	60	10:45	00	15
5.	10:00	10:45	15	11:00	00	45
6.	10:30	11:00	45	11:45	00	30
7.	11:00	11:45	45	12:35	00	45
8.	11:30	12:30	45	–	–	–
				Total	30	135

Example 6.6. The distribution of inter-arrival times in a single server model is

t	:	1	2	3
$f(t)$	:	.25	.50	.25

and distribution of service times is

s	:	1	2	3
$f(s)$	:	.50	.25	.25

Complete the following table using the two-digit random numbers 11, 20, 47, 68, 90, 62 and 35 to generate arrivals and 15, 86, 20,42, 11, 36 and 48 to generate the corresponding service times.

Arrival No.	*Arrival time*	*Service begin time*	*Service end time*	*Waiting time in queue*	*Server idle time*

[P.U. M.E. (Mech.), 1987]

Solution: The arrival times of the customers can be determined by generating the inter-arrival times, from the given discrete distribution. The inter-arrival times, probabilities of their occurrence, cumulative probabilities and the corresponding two-digit random numbers are given in Table 6.7.

Table 6.7

Inter-arrival time	*Probability*	*Comm. probability*	*Range of random number*
1	.25	.25	00 – 24
2	.50	.75	25 – 74
3	.25	1.00	75 – 99

Now using the given string of random numbers, the inter-arrival times can be generated. Corresponding to first random number, *i.e.*, 11, the inter-arrival times is 1, *i.e.*, first customer will arrive 1 time unit after opening the system for service. The inter-arrival times for seven arrivals are computed below.

Arrival number	:	1	2	3	4	5	6	7
Random number	:	11	20	47	68	90	62	35
Inter-arrival time	:	1	1	2	2	3	2	2

Similarly, the service times are generated, using the sequence of random numbers provided for this purpose.

Sr. No.	:	1	2	3	4	5	6	7
Random number	:	15	86	20	52	11	66	48
Service time	:	1	3	1	2	1	2	1

The given table can now be computed. It is assumed that service system opens at zero time.

Table 6.8

Arrival No.	*Arrival time*	*Service begin time*	*Service end time*	*Waiting time*	*Server idle time*
1.	0 + 1 = 1	1	1 + 1 = 2	0	1
2.	1 + 1 = 2	2	2 + 3 = 5	0	0
3.	2 + 2 = 4	5	5 + 1 = 6	1	0
4.	4 + 2 = 6	6	6 + 2 = 8	0	0
5.	6 + 3 = 9	9	9 + 1 = 10	0	1
6.	9 + 2 = 11	11	11 + 2 = 13	0	1
7.	11 + 2 = 13	13	13 + 1 = 14	0	0

Example 6.7. Simulate an M/D/2 system over the first 35 minutes of operation taking mean inter-arrival time as 3 minutes and the service times of Servers I and II as 5 and 6 minutes each. The inter-arrival times for the first 12 arrivals in min : sec have been generated as:

4:13, 2:00, 6:09, 1:37, 3:54, 6:09, 0:05, 2:49, 1:26, 0:52, 3:39, 8:54

Determine the percentage idle time of each server. **[P U. M.E. (Prod.), 1991]**

Solution: The notation M/D/2 stands for the queuing system having exponential arrival distribution, deterministic service times, two servers, infinite system capacity and the queue discipline is first-in first-out. In present example, mean inter-arrival time is 3 min. Using exponential distribution, the inter-arrival times have already been generated. The two servers; Server I and Server II take exactly 5 and 6 minutes respectively to serve the customer. There is no limit on the number of customers in the system.

Working of the system is simulated in Table 6.9. It is assumed that at zero time, there is no customer in the system. The first arrival takes place at 4:13. When both the servers are idle, the choice of the customer is faster server, *i.e.*, Server I. Service begins at 4:13 and ends at 9:13 as Server I takes exactly 5 minutes. Second customer arrives after 2:00 minutes i.e., at 6:13 and goes to Server II, where service beings at 6:13 and ends at 12:13. Third arrival takes place at 12:22 minutes, both the servers are idle, customer goes to Server I. Here the idle time of Server I is 12:22 – 9:13 = 3:09 mins. Fourth arrival takes place at 13:59, goes to idle Server II. The idle time of Server II is 13:59 – 12:13 = 1:46 mins. The process continues, when customer No. 8 arrives at 26:56 minute both the servers are busy, customer waits till Server I completes pervious service at 29:02 minutes. Here waiting time of customer is 2:06 mins. During 35 minutes, service on 8 customer is completed, 9th and 10th customer are in service.