

## Simulation of a single-server Queueing system relevant to hospital services

**Pratima Singh**

Department of Mathematics

P.G. College Ghazipur, 233001, India

E-mail:- pratima234340@gmail.com

**Abstract:** The system is simulated until a fixed number of patients have completed their delays in queue with the help of FORTRAN PROGRAM. The simulation stops when the  $n$ th patient enters service. The time the simulation ends is a random variable, depending on the observed values for the inter-arrival and service time random variables.

**Keywords :** Queueing models, Single-doctor, Simulation

### 1. Introduction :

Due to wide scope and applications of queueing models in health services, consider a single-doctor queueing system for which the interarrival times  $A_1, A_2, \dots$  are independent, identically distributed (IID) random variables. ("Identically distributed" means that the interarrival times have the same probability distribution). A patient, who arrives and finds the doctor idle enters service immediately, and the service times  $S_1, S_2, \dots$  of the successive patients are IID random variables that are independent of the interarrival times. A patient who arrives and finds the doctor busy joins the end of a single queue. Upon completing service for a patient, the doctor chooses a patient from the queue (if any) in a first-in, first-out (FIFO) manner.

The simulation will begin in the "empty-and-idle" state; i.e., no patients are present and the doctor is idle. At time 0, we will begin waiting for the arrival of the first patient, which will occur after the first interarrival time,  $A_1$ , rather than at time 0 (which would be a possibly valid, but different, modeling assumption). We wish to simulate this system until a fixed number ( $n$ ) of patients have completed their delays in queue; i.e., the simulation will stop when the  $n$ th patient enters service. Note that the time the simulation ends is thus a random variable, depending on the observed values for the interarrival and service-time random variables.

To measure the performance of this system, we will look at estimates of three quantities. First, we will estimate the expected average delay in queue of the  $n$  Patients completing their delays during the simulation; we denote this quantity

by  $d(n)$ . The word “expected” in the definition of  $d(n)$  means this: On a given run of the simulation (or, for that matter, on a given run of the actual system the simulation model represents), the actual average delay observed of observations that happen to have been obtained. On another run of the simulation (or on a different time for the real system) there would probably be arrivals at different times, and the service times required would also be different; this would give rise to a different value for the average of the  $n$  delays. Thus, the average delay on a given run of the simulation is properly regarded as a random variable itself. What we want to estimate,  $d(n)$ , is the expected value of this random variable. One interpretation of this is that  $d(n_i)$  is the average of a large (actually, infinite) number of  $n$ -Patient average delays. From a single run of the simulation resulting in Patient delays  $D_1, D_2, \dots, D_n$ , an obvious estimator of  $d(n)$  is

$$\hat{d}(n) = \frac{\sum_{i=1}^n D_i}{n}$$

which is just the average of the  $n, D_i, 's$  that were observed in the simulation [so that  $\hat{d}(n)$  could also be denoted by  $\bar{D}(n)$ ]. It is important to note that by “delay” we do not exclude the possibility that a patient could have a delay of zero in the case of an arrival finding the system empty and idle (with this model, we know for sure that  $D_1 = 0$ ); delays with a value of zero are counted in the average, since if many delays were zero this would represent a system providing very good service, and our output measure should reflect this,

while an estimate of  $d(n)$  gives information about system performance from the patients’ point of view, the management of such a system may want different information.

One such measure for our simple model here is the expected average number of patients in the queue (but not being served), denoted by  $q(n)$ , where the  $n$  is necessary in the notation to indicate that this average is taken over the time period needed to observe the  $n$  delays defining our stopping rule. This is a different kind of “average” than the average delay in queue, because it is taken over (continuous) time, rather than over patients (being discrete). Thus, we need to define what is meant by this time-average number of patients in queue. To do this, let  $Q(t)$  denote the number of patients in queue at time  $t$ , for any real number  $t \geq 0$ , and let  $T(n)$  be the time required to observe our  $n$  delays in queue. Then for any time  $t$  between 0 and  $T(n)$ ,  $Q(t)$  is a nonnegative integer. Further, if we let  $p_i$  be the expected proportion (which will be between 0 and 1) of the time that  $Q(t)$  is equal

to  $i$ , then a reasonable definition of  $q(n)$  would be

$$\hat{q}(n) = \sum_{i=0}^{\infty} i p_i$$

Thus,  $q(n)$  is a weighted average of the possible values  $i$  for the queue length  $Q(t)$ , with the weights being the expected proportion of time the queue spends at each of its possible lengths. To estimate  $q(n)$  from a simulation, we simply replace the  $p_i$ 's with estimates of them, and get

$$\hat{q}(n) = \sum_{i=0}^{\infty} i \hat{p}_i \quad (1)$$

where  $\hat{p}_i$  is the observed (rather than expected) proportion of the time during the simulation that there were  $i$  Patients in the queue. Computationally however, it is easier to rewrite  $\hat{q}(n)$  using some geometric considerations. If we let  $T_i$  be the total time during the simulation that the queue is of length  $i$ , then  $T(n) = T_0 + T_1 + T_2 + \dots$  and  $p_i = T_i/T(n)$ , so that we can rewrite Eq.(1) above as

$$\hat{q}(n) = \frac{\sum_{i=0}^{\infty} i T_i}{T(n)} \quad (2)$$

In other words, the summation in the numerator of Eq. (2) is just the area under the  $Q(t)$  curve between the beginning and the end of the simulation. Remembering that “area under a curve” is an integral, we can thus write

$$\sum_{i=0}^{\infty} i T_i = \int_0^{T(n)} Q(t) dt$$

and the estimator of  $q(n)$  can then be expressed as

$$\hat{q}(n) = \frac{\int_0^{T(n)} Q(t) dt}{T(n)} \quad (3)$$

While Eqs. (3) and (2) are equivalent expressions for  $q(n)$ , Eq. (3) is preferable since the integral in this equation can be accumulated as simple areas of rectangles as the simulation progresses through time. It is less convenient to carry out the computations to get the summation in Eq. (2) explicitly. Moreover, the appearance

of Eq. (3) suggests a continuous average of  $Q(t)$ , since in a rough sense, an integral can be regarded as a continuous summation.

The third and final output measure of performance for this system is a measure of how busy the doctor is. The expected utilization of the doctor is the expected proportion of time during the simulation [from time 0 to time  $T(n)$ ] that the doctor is busy (i.e., not idle) and is thus a number between 0 and 1; denote it by  $u(n)$  = the observed proportion of time during the simulation that the doctor is busy. Now  $\hat{u}(n)$  could be computed directly from the simulation by noting the times at which the doctor changes status (idle to busy or vice versa) and then doing the appropriate subtractions and division. However, it is easier to look at this quantity as a continuous-time average, similar to the average queue length, by defining the “busy function”

$$B(t) = \begin{cases} 1 & \text{if the doctor is busy at time } t \\ 0 & \text{if the doctor is idle at time } t \end{cases}$$

The single-doctor queue with exponential interarrival and service times is commonly called the M/M/1 queue.

To simulate this model we need a way to generate random variates from an exponential distribution. The subprograms used by the FORTRAN, Pascal, and C codes all operate in the same way, which we will now develop. First, a random-number generator is invoked to generate a variate  $U$  that is distributed (continuously) uniformly between 0 and 1; this distribution will henceforth be referred to as  $U(0,1)$  and has probability density function

$$f(x) = \begin{cases} 1, & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

It is easy to show that the probability that a  $U(0,1)$  random variable falls in any subinterval  $[x, x + \Delta x]$  contained in the interval  $[0,1]$  is (uniformly)  $\Delta x$ . The  $U(0,1)$  distribution is fundamental to simulation modeling because, it is known that a random variate from any distribution can be generated by first generating one or more  $U(0,1)$  random variates and then performing some kind of transformation. After obtaining  $U$ , we shall take the natural logarithm of it, multiply the result by  $\beta$ , and finally change the sign to return what we will show to be an exponential random variate with mean  $\beta$ , that is,  $-\beta \ln U$ .

To see why this algorithm works, W.C.Giffin [3] recall that the (cumulative) distribution function of a random variable  $X$  is defined, for any real  $x$ , to be  $F(x) = P(X \leq x)$ . If  $X$  is exponential with mean  $\beta$ , then

$$F(x) = \int_0^x \frac{1}{\beta} e^{-t/\beta} dt = (1 - e^{-x/\beta})$$

for any real  $x \geq 0$ , since the probability density function of the exponential distribution at the argument  $t \geq 0$  is  $(1/\beta)e^{-t/\beta}$ . To show that our method is correct, we can try to verify that the value it returns will be less than or equal to  $x$  (any nonnegative real number), with probability  $F(x)$  given above :

$$\begin{aligned} P(-\beta \ln U \leq x) &= P\left(\ln U \geq \frac{x}{\beta}\right) \\ &= P(U \geq e^{-x/\beta}) \\ &= P(e^{-x/\beta} \leq U \leq 1) \\ &= 1 - e^{-x/\beta} \end{aligned}$$

The first line in the above is obtained by dividing through by  $-\beta$  (recall that  $\beta > 0$ , so  $-\beta < 0$  and the inequality reverses), the second line is obtained by exponentiating both sides (the exponential function is monotone increasing, so the inequality is preserved), the third line is just rewriting, together with knowing that  $U$  is in  $[0,1]$  anyway, and the last line follows since  $U$  is  $U(0,1)$ , and the interval  $[e^{-x/\beta}, 1]$  is contained within the interval  $[0, 1]$ . Since the last line is  $F(x)$  for the exponential distribution, we have verified that our algorithm is correct.

In our programs, we prefer to use a particular method for random number generation to obtain the variate  $U$  described above, as expressed in the FORTRAN, Pascal, and C codes while most compilers do have some kind of built-in random-number generator, many of these are of extremely poor quality and should not be used.

It is convenient (if not the most computationally efficient) to modularize the programs into several subprograms to clarify the logic and interactions, in addition to a main program, the simulation program includes routines for initialization, timing, report generation, and generating exponential random variates. It also simplifies matters if we write a separate routine to update the continuous time statistic, being the accumulated areas under the  $Q(t)$  and  $B(t)$  curves. The most important action, however, takes place in the routines for the events, which we number as follows :

---

| Event description                                               | Event type |
|-----------------------------------------------------------------|------------|
| Arrival of a patient to the system                              | 1          |
| Departure of a patient from the system after completing service | 2          |

---

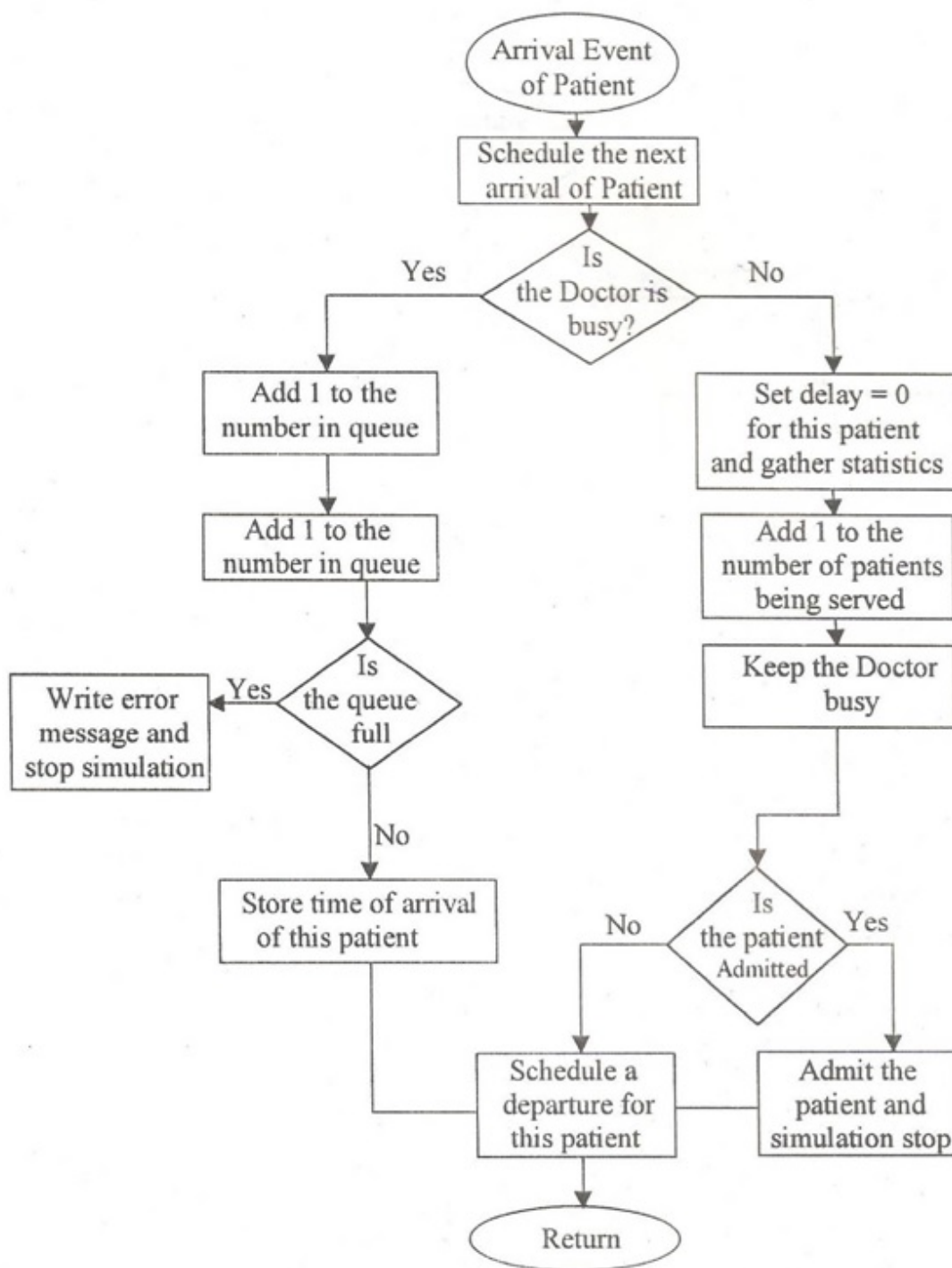


Figure 1: Flowchart for arrival routine, queueing model

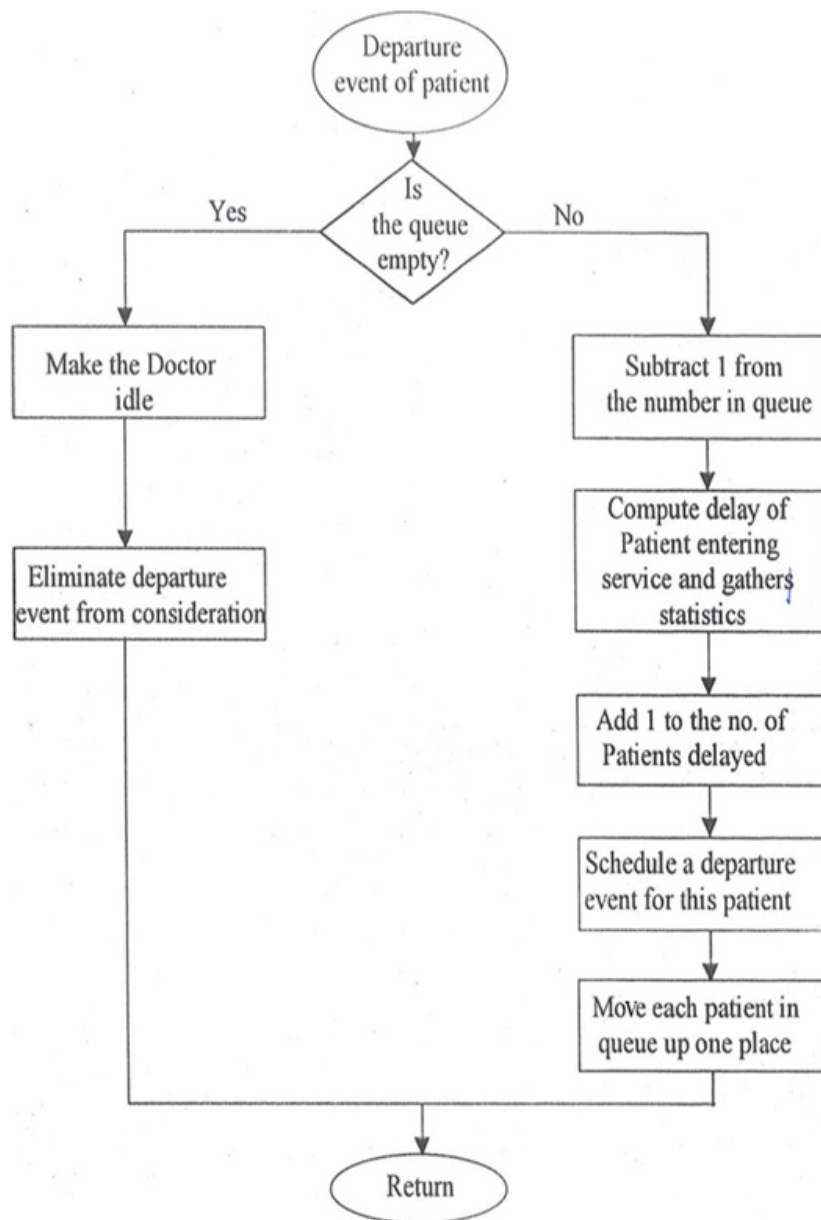


Figure 2: Flowchart for departure routine, queueing model

As the logic of these event routines is independent of the particular language to be used, we shall discuss it here. Figure 1 contains a flowchart for the arrival event. First, the time of the next arrival in the future is generated and placed in the event list. Then a check is made to determine whether the doctor is busy. If so, the number of patients in the queue is incremented by one, and we ask whether the storage space allocated to hold the queue is already full. If the queue is already full, an error message is produced and the simulation is stopped; if there is still room in the queue, the arriving patient's time of arrival is put at the (new) end of the queue. On the other hand if the arriving patient finds the doctor idle, then this patient has a delay of zero, which is counted as a delay, and the number of patient delays completed is incremented by one. The doctor must be made busy, and the time of departure from service of the arriving patient is scheduled into the event list.

The departure event's logic is depicted in the flowchart of Fig. 2. Recall that this routine is invoked when a service completion (and subsequent departure) occurs. If the departing patient leaves no other patients behind in queue, the doctor is idled and the departure event is eliminated from consideration, since the next event must be an arrival. On the other hand, if one or more patients are left behind by the departing patient, the first patient in queue will leave the queue and enter service, so the queue length is reduced by one, and the delay in queue of this patient is computed and registered in the appropriate statistical counter. The number delayed is increased by one, and a departure event for the patient now entering service is scheduled. Finally, the rest of the queue (if any) is advanced one place. Our implementation of the list for the queue will be very simple.

C MAIN PROGRAM FOR SINGLE DOCTOR

INTEGER QLIMIT

PARAMETER(QLIMIT=100)

INTEGER BUSY, IDLE, NEVNTS, NEXT .NIQ, NUMPAT, DOCTOR,  
TOTPAT

REAL AN 1Q, AUTIL, MARRVT, MDOVT, TARRVL (QLIMIT), T1ME,  
TLEVNT, & TNE(2),TOTDEL

REAL EXPON

COMMON/ MODEL/ ANIQ, AUTIL, BUSY, IDLE, MARRVT, MDOCVT,  
NEVNTS, & NEXT, NIQ, NUMPAT, DOCTOR, TARRVL, TIMETLEVNT  
& TNE, TOTPAT, TOTDEL

C OPEN INPUT AND OUTPUT FILES

OPEN (5, FILE = 'MM 1. IN', STATU='OLD')

OPEN (6, FILE - 'MM 1 OUT, STATUS= NEW)

```
C    SPECIFY THE NUMBER OF EVENT TYPES FOR THE TIMING
ROUTINE
NEVNTS = 2
BUSY = 1
IDLE = 0
READ(5,*) MARRVT, MIDOCVT, TOTPAT
WRITE (6, 2010)MARRVT, MDOCVT, TOTPAT
2010 FORMAT ('SINGLE-DOCTOR QUEUEING SYSTEM//
& 'MEAN INTER ARRIVAL TIME', F11.3,' MINUTES'//
& 'MEAN DOCTOR TIME, F16.3', MINUTES'//
& 'NUMBER OF PATIENTS', 114//)
CALL INTT
IF(NUMPAT LT. TOTPAT) THEN
CALL TIMING
CALL UPTAVG
GO TO (10, 20), NEXT
10  CALL ARRIVE
GO TO 30
20  CALL DEPART
END IF
30  CALL REPOR
C    END IF
END FILE 5
END FILE 6
STOP
END
SUBROUTINE IN1T
C    INCLUDE 'MM 1. FOR'
REAL TNE(2)
TIME = 0.0
DOCTOR = IDLE
NIQ = 0
TLEVNT = 0.0
NIJMPAT = 0
TOTDEL = 0.0
ANIQ 0.0
AUTIL = 0.0
TNE(1) - TIME + (EXPON(MARRVT))
```

```

TNE(2) = 1.0E+30
RETURN
END
SUBROUTINE TIMING
C   INCLUDE 'MMI FOR'
INTEGER I
REAL TNE(2)
REAL MINTNE
MINTNE = 1.0E+29
NEXT=0
DO 101= 1, NEVNTS
IF(TNE(I) .LT. MINTNE) THEN
MINTNE = TNE(I)
NEXT = I
END I F
10 CONTINUE
IF(NEXT .EQ. 0) TIEN
WRTE(6, 2010) TIME
2010 FORMATCEVEOT LIST EMPTY AT TIME, F10.3)
STOP
END II7
TIME - MINTNE
RETURN
END
SUBROUTINE ARRIVE
C   INCLUDE 'MM 1.FOR'
REAL TNE(2), TARRVL(101)
REAL DELAY
TNE(1) = TIME + EXPON(MARRVT))
IF(DOCTOR. EQ. BUSY) THEN
NIQ = NIQ + 1
IF(N1Q GT. QLIMIT) THEN
WRTE (6, 2010) TIME
2010 FORMAT ('OVERFLOW OF THE ARRAY TARRVLAT TIME', F10.3)
STOP
END IF
TARRVL (NIQ) = TIME
ELSE

```

```
DELAY = 0.0
TOTDEL = TOTDEL + DELAY
Nump AT = NumpAT + 1
DOCTOR = BUSY
TNE(2) = TIME + EXPON (MDOCVT)
END IF
RETURN
END
SUBROUTINE DEPART
C   INCLUDE 'MM 1. FOR'
REAL TNE(2), TARRVL(100)
INTEGER I
REAL DELAY
IF (N1Q , EQ, 0) THEN
DOCTOR = IDLE
TNE(2) = 1.0E + 30
ELSE
NIQ = NIQ - 1
DELAY = TIME - TARRVL(1)
TOTDEL = TOTDEL + DELAY
NumpAT = Nump AT + 1
TNE(2) = TIME + EXPON(MDOCVT)
DO 101=1, NIQ
TARRVIXI)=TARRVL(I+1)
10  CONTINUE
END IF
RETURN
END
SUBROUTINE UPTAVG
C   INCLUDE 'MM1.FOR'
REALTSLE
TSLE = TIME - TLEVNT
TLEVNT = TIME
AN1Q = AN1Q + N1Q * TSLE
AUT1L = AUTIL + DOCTOR * TSLE
RETURN
END
SUBROUTINE REPOR
```

```

C   INCLUDE 'MM 1. FOR'
REAL AVGDEL, AVGN1Q, UTEL
OPEN (6, FI LE = 'MML', OUT, STATUS = NEW)
AVGDEL = TOTDEL / NUMPAT
AVGNIQ = ANIQ / TIME
UT1L = AUTIL / TIME
WRTRE(6, 2010) AVGDEL, AVGNLQ, UTIL, TIME
2010 FORMAT ('AVERAGE DELAY IN QUEUE', F 11.3', MINUTES'//
& 'AVERAGE NUMBER IN QUEUE', F10.3//
& DOCTOR UTILIZATION', F15.3//
& TIME SIMULATION ENDED', F12.3', MINUTES')
END FILE 6
RETURN
END
REAL FUNCTION EXPON (REAL RMEAN)
REAL U
U = RAND (1)
EXPON = -RMEAN * LOG(U)
RETURN
END

```

### **Simulation Output and Discussion:**

Single-doctor queueing system

|                         |                  |
|-------------------------|------------------|
| Mean interarrival time  | 1.000 minutes    |
| Mean service time       | 0.500 minutes    |
| Number of Patients      | 1000             |
| Average delay in queue  | 0.430 minutes    |
| Average number in queue | 0.418            |
| Doctor utilization      | 0.460            |
| Time simulation ended   | 1027.915 minutes |

### **Implementation of the Program :**

General references on the FORTRAN language are Davis and Hoffmann [2] and Koffman and Friedman [1].

To determine the steady state measures our estimates based on the stopping rule  $n = 1000$ . Since  $n = 1000$  was chosen arbitrarily, the choice of a stopping rule that will give good estimates of steady state measure is quite difficult. To illustrate this point, suppose for the M/M/1 queue that the arrival rate of patient were increased one per minute to 1.98 per minute (the mean interarrival time is now 0:505 minute), that the mean service time is unchanged and if we estimate

the steady-state measures from a run of length  $n = 1000$  delays, as before, we get values for the average delay, average number in queue, and doctor utilization of 17,404 minutes, 34.831 and 0.997 respectively.

We did not estimate the expected average waiting time in the system of a customer, rather than they expected average delay in queue, where the waiting time of a patient is defined as the time interval from the instant the patient arrives to the instant the patient complete service and departs. There were two reasons.

First, for many queueing systems we believe that the patient's delay in queue while waiting for other patients to be served is the most troublesome part of the patient's wait in the system. Our second reason for focusing on the delay in queue is one of statistical efficiency.

### References

- [1] E.B. Koffman and F.L. Friedman (1987); Problem solving and structured programming in FORTRAN 77, 3rd edn., Addison Wesley, Reading, Mass.
- [2] G.B., Davis and T.R. Hoffmann (1988), FORTRAH 77, A structured Disciplined style, 3rd edn., Mc Graw Hill, New York.
- [3] W.C. Giffin; (1978), Queueing, Basic Theory and Application, Columbus Oh-Grive Publishing.