SCHEDULE INDEPENDENT SIMULATION: A NEW TECHNIQUE OF DISCRETE EVENT SIMULATION SUITABLE FOR SMALL COMPUTERS

Susanne R. Norton

College of Engineering
University of Toledo

ABSTRACT

A state space approach to discrete event simulation is presented where an explicit mathematical model of the System is obtained using recursive mathematical functions similar to those utilized in continuous system simulation. Only the processes of stated interest in a simulation study are modeled.

Since events in the simulated system are not necessarily executed in their real-time order of occurrence, many of the repeated list scans associated with conventional simulation techniques can be eliminated. This simulation algorithm constitutes an efficient simulation alternative that can be conveniently implemented in FORTRAN on today's smaller scale computers and time sharing systems.

I. BACKGROUND

Minicomputers and microcomputers constitute one of the fastest growing segments of the computer market (11). Therefore, attention focuses more keenly on development of simulation algorithms, such as the one described below, that are suitable for implementation on smaller scale computers.

Traditional simulation approaches such as GPSS, etc. contain built-in control algorithms to automatically monitor the sequencing of simulated events in their real-time order of occurrence. The resultant high storage and execution time requirements (particularly if the system is congested) generally preclude use of minicomputers.

A new technique of modeling discrete event systems, denoted SIS for Schedule Independent Simulation, was initially formulated at Northwestern University and constituted this author's recent Ph.D. thesis (8). By pinpointing simulation objectives, schedule independent techniques can be utilized. Minimum state space representations have been obtained for a variety of systems demonstrating serial and parallel connection. Computer implementations were in Fortran in batch mode on the CDC 6400 computer.

The development of SIS has been, and will continue to be, an ongoing project. Current work at the University of Toledo* continues: (i) to extend the SIS algorithm to cover a broader class of queueing systems demonstrating feedback as well as serial and parallel connection; and (ii) to support development of related Fortran software. Recent computer implementations have been performed principally in an interactive mode on a 28K, PDP-11 time sharing system.

The following sections summarize the development of the SIS algorithm. It is in no way intended that SIS be viewed as constituting a "replacement" for conventional simulation analysis but simply as a convenient, efficient simulation "alternative" suitable to implementation on a small computer in a variety of "real world" queueing situations.

II. MODELING CONCEPTS

SIS is a simulation algorithm that:

- (i) models discrete event systems using explicit recursive mathematical functions (similar to those employed in continuous system simulation)
- (ii) processes events out of their real time order of occurrence and thereby eliminates expensive list processing associated with conventional time-sequential approaches
- (iii) models only the processes of stated interest in a particular simulation study, thereby achieving a minimum state space representation of the system
- (iv) can be conveniently implemented on a smaller scale computer or in the time sharing environment in a higher level language such as FORTRAN which is familiar to users and has attractive subprogram capabilities.

In many simulation studies the analyst can identify a very specific simulation objective, such as the estimation of a particular waiting time parameter or customer transit time, etc. SIS is particularly appropriate for modeling such systems, especially if a larger computer is not available or if the simulated system is congested. In some instances, execution time of the SIS recursive equations can be independent of the degree of congestion in the simulated system.

*Research currently supported by National Science Foundation Grant #ENG76-09892.

MATHEMATICAL TECHNIQUES

A complex discrete event simulation model can be viewed as a queueing network with output from one queue comprising the input to another. Each queue is characterized by an arrival pattern, service mechanism, and a queue discipline. The queues in the model may be connected by serial, parallel, or feedback connection.

In a general state-space representation of a system S defined over T we have:

$$S \begin{cases} s(t) = f[s(to),\ \mu(to,t)] & (1) \\ y(t) = h[s(t),\ \mu(t)] & (2) \end{cases}$$

where: to $\triangleq$ starting point
s(t) $\triangleq$ state of the system at $t \in$ parameter set T.
$\mu(t)$ $\triangleq$ value of input function at $t \in T$.
y(t) $\triangleq$ value of output function at $t \in T$.
input segment (to,t) $\triangleq$ input up until $t \in T$.

In a conventional GPSS or Simscript computer implementation, for example, the parameter set T represents time and processing throughout the simulation is time-sequential, accomplished via "scheduling" techniques. Lists of potential events are maintained, and as events are scheduled, they are placed on "chains" which are subsequentially scanned by the control algorithm and executed in their real-time order of occurrence.

The contribution of this work is to demonstrate how state representations can be achieved where the parameter set T does not represent time. Instead, times such as arrival times and departure times are viewed as "states" of the system and computed via recursive mathematical equations. Processing of some systems can be totally "schedule-independent", and execution time of the recursive mathematical equations can be independent of the degree of congestion in the simulated system.

Example: As a simple example to demonstrate the modeling concepts involved, consider a system consisting of just one, single server, first-come-first-served (FCFS) queue. Let parameter set T $\triangleq$ {1,2,3,...} to represent arrival number into the system and not time. (Such processing shall be denoted as "customer-sequential processing"). Define input functions as:

$u_1(i)$ $\triangleq$ interarrival time between i^{th} and $(i+1)^{st}$ arrivals

$u_2(i)$ $\triangleq$ length of service time of i^{th} arrival

Let states of the system be defined as:

TA(i) $\triangleq$ time of i^{th} arrival into system
TD(i) $\triangleq$ time of i^{th} departure
W(i) $\triangleq$ waiting time in line of i^{th} arrival
N(i) $\triangleq$ queue size upon and including the i^{th} arrival

Then, to study waiting times or queue size, we obtain the following recursive state transition functions:

$$TA(i+1) = TA(i) + u_1(i) \quad (3)$$

$$W(i+1) = MAX\ \{0;\ W(i) + u_2(i) - u_1(i)\} \quad (4)$$

$$TD(i+1) = TA(i+1) + W(i+1) + u_2(i+1) \quad (5)$$

$$N(i+1) = N(i) + 1 - \sum_{j=i+1-N(i)}^{i} B(j) \quad (6)$$

$$\text{where } B(j) = \begin{cases} 1 \text{ if } TA(i+1) \geq TD(j) \\ 0 \text{ otherwise} \end{cases}$$

TA(1) is given; N(1) = 1

Processing in a SIS Simulation can be "customer-sequential", where a customer is processed through a major portion, if not the entire system, before processing of the next sequential customer to enter the system is even initiated (as demonstrated by the example). Also, as described in (8), processing can be "queue-sequential", which is attained by defining the parameter set T $\triangleq$ {1,2,3,...} to represent queue number in the system. Computation of the recursive state equations is performed by simulating all customers through any queue (i) before any customer is processed at queue (i+1).

STATE SPACE MODELS FOR REPRESENTATIVE SYSTEMS

Systems depicting serial, parallel, and feedback connection have been analyzed and recursive state space representations achieved. Cases have included:

(i) single server, FCFS queues in tandem with both limited and unlimited line length, using both customer sequential processing and queue sequential processing.

(ii) parallel server queues with unlimited line lengths

(iii) priority, non-preemptive queue with one class priority customer

(iv) mixture of above

(v) feedback queues with unlimited line lengths and unlimited number of feedbacks.

Earlier derivations and FORTRAN programs in (8) are confined to systems depicting only serial or parallel connection. Computer implementations were on a CDC 6400 computer. Recent work also emphasizes the modeling of feedback queues and the development of interactive FORTRAN programs executed on a 28K, PDP-11 time sharing system.

A few sample models are presented below to demonstrate the modeling theory.

1. Serial Connection: FCFS Queues in Tandem

Many real world stochastic systems can be described by a series of queues in tandem depicting serial connections (or perhaps serial connection combined with some parallel connection). Moreover, it is in the simulation of the serially connected, congested system that the SIS algorithm can make the most significant contribution.

It is here that an application of the SIS algorithm could result in totally schedule-free processing with all time consuming list scans eliminated. (See following example.) Execution time of the recursive state equations could be independent of the degree of congestion in the simulated system. A simulation model that might have significant storage requirements and execution time in a time-sequential algorithm such as GPSS could be reduced to a nominal FORTRAN problem.

<u>State Equations</u>: Consider a series of FCFS, single-server queues in tandem with unlimited line lengths. Let parameter set T $\triangleq$ {1,2,3,...} to represent arrival number into system. Define state variables as:

$TA(i)_{k+1}$ $\triangleq$ time of arrival of customer (k+1) at queue (i)

$W(i)_{k+1}$ $\triangleq$ waiting time of customer (k+1) at queue (i)

$TD(i)_{k+1}$ $\triangleq$ time of departure of customer (k+1) from queue (i)

$N(i)_{k+1}$ $\triangleq$ Number of customers at queue (i) upon and including arrival of customer (k+1) to the queue.

Input is:

$S(i)_{k+1}$ $\triangleq$ service time of customer (k+1) at queue (i)

I_k $\triangleq$ interarrival time at queue (1) between customer (k) and customer (k+1)

Then the following state transition functions are obtained:

$$TA(i)_{k+1} = \begin{cases} TA(i)_k + I_k & \text{for } i=1 \\ TD(i-1)_{k+1} & \text{for } i>1 \end{cases} \qquad (7)$$

$$W(i)^*_{k+1} = \begin{cases} W(1)_k + S(1)_k - I_k & \text{for } i=1 \\ W(i)_k + S(i)_k - S(i-1)_{k+1} & \\ + [\lambda_{i-1,k+1}]\, W(i-1)^*_{k+1} & \text{for } i>1 \end{cases} \qquad (8)$$

$$\text{where } \lambda_{i-1,k+1} = \begin{cases} 1 & \text{if } W(i-1)^*_{k+1} < 0 \\ 0 & \text{otherwise} \end{cases}$$

$$W(i)_{k+1} = \text{MAX } \{0;\ W(i)^*_{k+1}\} \qquad (9)$$

$$TD(i)_{k+1} = TA(i)_{k+1} + W(i)_{k+1} + S(i)_{k+1} \qquad (10)$$

$$N(i)_{k+1} = N(i)_k + 1 - \sum_{j=k+1-N(i)_k}^{k} B(i)_{k+1,j} \qquad (11)$$

$$\text{where: } B(i)_{k+1,j} = \begin{cases} 1 & \text{if } TA(i)_{k+1} \geq TD(i)_j \\ 0 & \text{otherwise} \end{cases}$$

<u>Computer Implementation</u>: A conservative example consisting of five single server queues in tandem with unlimited line lengths was simulated in FORTRAN (for 3 levels of system congestion) using the SIS algorithm for customer-sequential processing. An associated GPSS simulation was also performed.

Results indicated that execution time of the SIS simulation remained <u>constant</u> (ie., independent of the degree of congestion in the simulated system). SIS simulations were about 5 times faster than the associated GPSS simulations even in this conservative example.

In our current environment, such a typical SIS simulation could be performed on a 28K, PDP-11 time sharing system while the GPSS simulation (submitted as a low priority job) requires use of the IBM 360 and a 108K partition.

2. Parallel Connection

The SIS simulation algorithm has been applied to a multiple server queue. In this instance, some time-sequential processing is combined with the recursive techniques.

Consider a multiple server queue with "NS" servers. A customer is assigned to a specific server via an appropriate assignment algorithm. Let parameter set T represent arrivals. Define state variables as:

TA_{k+1} $\triangleq$ Time of arrival of customer (k+1) at the queue

TD_{k+1} $\triangleq$ Time of departure of customer (k+1) from the queue

$W(i)_{k+1}$ $\triangleq$ Waiting time of customer (k+1) in line (i)

$N(i)_{k+1}$ $\triangleq$ Number of customers in line (i) and at server (i) upon and including the arrival of customer (k+1)

Input is:

$I(k)$ $\triangleq$ Interarrival time between customer (k+1) and customer (k) at the queue

$S(i)_{k+1}$ $\triangleq$ Service time of customer (k+1) at server (i)

Then the following state transition functions can be obtained:

$$TA_{k+1} = TA_k + I(k) \qquad (12)$$

$$W(I)^*_{k+1} = \begin{cases} W(i)_{last} + S(i)_{last} - \sum_{j=last}^{k} I(j) & \\ \text{or} \quad -\sum_{j=1}^{k} I(j) & \text{if customer (k+1) is the first customer to use server (i)} \end{cases} \qquad (13)$$

where "last" refers to the index number of the last previous customer to be assigned to server (i)

$$W(i)_{k+1} = \text{MAX} \ \{0; W(i)^*_{k+1}\} \quad (14)$$

$$TD_{k+1} = TA_{k+1} + W(i)_{k+1} + S(i)_{k+1} \quad (15)$$

$$N(i)_{k+1} = \begin{cases} N(i)_k + \lambda - NGONE(i,k+1) \quad (16) \\ \quad \text{where } \lambda = \begin{cases} 1 & \text{if customer } (k+1) \text{ assigned to line } (i) \\ 0 & \text{otherwise} \end{cases} \\ \text{or} \\ 1 \quad \text{if customer } (k+1) \text{ is the first customer to enter line } (i) \end{cases}$$

where: $NGONE(i,k+1) \triangleq$ Number of customers that departed server (i) after the arrival of customer (k) and before arrival of customer (k+1)

3. Feedback Connection

Feedback connection has been more difficult to model using the SIS algorithm. A recursive model which also incorporates some time sequential techniques has been obtained for a single server FCFS queue in which customers have the option of leaving the queue upon completion of initial service or feeding back an unlimited number of times. Computer implementation is on a PDP-11 time sharing system where it is determined in an interactive mode whether a customer feeds back.

The capabilities of this model are being expanded and it will be presented in more detail at the conference.

III. COMPUTER IMPLEMENTATIONS

Since the major contribution of this work involves presenting a new discrete event modeling technique, emphasis in the computer examples has been on applying appropriate modeling concepts rather than on performing many simulations of real world systems and interpreting statistical results.

The iterative nature of the state equations makes computer implementations particularly well suited to the "horizontal" type of processing we have denoted as "customer sequential" processing. Therefore, customer sequential processing has been emphasized throughout the work.

To demonstrate the modeling theory, one typical simulation run for a series of FCFS queues in tandem is presented in the Appendix along with the associated Fortran source code. Other examples of simulations of queueing systems depicting both serial and parallel connection are presented in (8). Recent simulations of feedback queues (not described in (8)) have been performed on a PDP-11 time sharing system and are also available.

IV. FUTURE WORK PLANNED

Future work scheduled will extend the SIS algorithm to cover a broader class of queueing situations than those described in section II. Associated Fortran subprograms will be developed to offer the user a timely, efficient simulation alternative suitable to implementation on a smaller scale computer or in the time sharing environment.

REFERENCES

1. Cadzow, J. A. and Martens, H. R. Discrete-Time and Computer Control Systems. Prentice-Hall, Englewood Cliffs, N.J., 1970.
2. Cooper, Robert B. Introduction to Queueing Theory. MacMillan Co., New York, N.Y., 1972
3. Feller, William. An Introduction to Probability Theory and Its Applications, Vol. II (Chapter 6). John Wiley & Sons, New York, N.Y., 1971
4. Fishman, George and Kiviat, P. "The Analysis of Simulation - Generated Time Series, Management Science, Vol. 13, March, 1967.
5. General Purpose Simulation System/360: User's Manual, Pub. GH20-0326-4. IBM Corp., White Plains, N.Y., 1970.
6. Gordon, Geoffrey. System Simulation. Prentice-Hall, Englewood Cliffs, N.J., 1969
7. Kiviat, P. J.; Villaneuva, R. and Markowitz, H. M. The Simscript II Programming Language. Prentice Hall, Englewood Cliffs, N.J., 1968.
8. Norton, Susanne R. "Schedule Independent Simulation (SIS): A State Space Approach to Discrete Event Simulation", unpublished doctoral dissertation, Northwestern University, 1976. (thesis advisor: Brian Schaefer)
9. Pritsker, A.; Alan, B. and Kiviat, Philip J. Simulation with GASP II. Prentice Hall, Englewood Cliffs, N.J., 1969.
10. Schaefer, Brian M. Program for Interactive Multiple Process Simulation: User's Guide. Northwestern University, Evanston, Ill., 1974.
11. U.S. Industrial Outlook 1976, U.S. Dept. of Commerce, Domestic and International Business Administration, Washington, D.C., 1976 (pp 287-291)
12. Wyman, F. Paul. "Improved Event-Scanning Mechanisms for Discrete Event Simulation," Communications of the ACM, Vol. 18, June, 1975.
13. Zadeh, L. A. and Desoer, A. Linear System Theory. McGraw-Hill, New York, N.Y., 1963.

```
      C  PROGRAM QHOR(INPUT,OUTPUT)
      C  PROGRAM QHOR SIMULATES A SERIES OF FCFS SINGLE SERVER QUEUES
      C  IN TANDEM. LINE LENGTHS MAY BE INFINITE OR BOUNDED.
      C  RECURSIVE STATE EQUATIONS ARE GIVEN FOR -- WAITING TIMES IN
      C  QUEUES, NUMBERS IN QUEUES, ARRIVAL TIMES AT QUEUES, DEPARTURE
      C  TIMES FROM QUEUES,STATE EQUATIONS ARE COMPUTED SEQUENTIALLY BY
      C  CUSTOMER. THAT IS,COMPUTATIONS ARE PERFORMED HORIZONTALLY
      C  ACROSS GRID, CALCULATING ALL QUEUE DATA FOR CUSTOMER(K) BEFORE
      C  PROCESSING CUSTOMER(K+1) AT ANY QUEUE.
            COMMON/BL1/NUMBER,KSTOP,NQ,ISIZE,TARRIV,ICAP(20)
            COMMON/BL2/NUMB(20),W(20),S(20),TDEPRT,RINT,TASAVE
            COMMON/BL3/IO1,IO2
            COMMON/BL4/VEC(20,20),L(20)
            COMMON/BL5/SOLD(20)
            COMMON/BL9/A,B,X,STDEV,XMU,Q,IX,IY
            COMMON/BL10/IUNIT,JUNIT
      C  IUNIT IS THE OUTPUT DEVICE USED.
      C  JUNIT IS THE INPUT DEVICE USED.
            IUNIT=7
            JUNIT=5
            WRITE(IUNIT,600)
        600 FORMAT(1H ,'INPUT DATA FOR THIS SIMULATION RUN AS
           1 DEFINED. FOLLOW THE ALPHA')
            WRITE(IUNIT,602)
        602 FORMAT(1H ,'FORMAT SPECIFIED AND SEPARATE ALL VARIABLES
           1 BY COMMAS. ')
            WRITE(IUNIT,604)
        604 FORMAT(1H ,'*********************************************
           1****************************')
            WRITE(IUNIT,606)
        606 FORMAT(1H0,'INPUT NRUNS--NUMBER OF RUNS, FORMAT I10')
            WRITE(IUNIT,608)
        608 FORMAT(1H0,'? ')
            READ(JUNIT,100) NRUNS
        100 FORMAT(I10)
            DO 89 ICOUNT=1,NRUNS
            IOMODE=1
      C  READ INPUT DATA FOR THIS SIMULATION RUN.
      C  NUMBER --- RUN ID NUMBER
      C  KSTOP -- NUMBER OF CUSTOMERS TO BE SIMULATED
      C  NQ -- NUMBER OF QUEUES IN TANDEM
      C  ISIZE -- EQUALS 1 IF LINES BOUNDED.  0 OTHERWISE
      C  TARRIV -- TIME OF ARRIVAL OF FIRST CUSTOMER
      C  IOUT -- OPTIONAL OUTPUT IS WRITTEN EVERY IOUT CUSTOMERS
      C  ICAP(I) -- CAPACITY OF LINE(I) IF LINES ARE BOUNDED
            WRITE(IUNIT,612)
        612 FORMAT(1H0,'NUMBER-- RUN ID NUMBER, FORMAT I10')
            WRITE(IUNIT,614)
        614 FORMAT(1H ,'KSTOP-- NUMBER OF CUSTOMERS TO BE SIMULATED,
           1 FORMAT I10')
            WRITE(IUNIT,616)
        616 FORMAT(1H ,'NQ-- NUMBER OF QUEUES IN TANDEM, FORMAT I10')
            WRITE(IUNIT,618)
        618 FORMAT(1H ,'ISIZE-- EQUALS 1 IF BOUNDED, 0 OTHERWISE')
            WRITE(IUNIT,620)
        620 FORMAT(1H ,'TARRIV-- TIME OF ARRIVAL OF FIRST CUSTOMER,
           1 FORMAT F10.3')
            WRITE(IUNIT,622)
        622 FORMAT(1H ,'IOUT-- OPT. OUTPUT WRITTEN EVERY IOUT
           1 CUSTOMERS, FORMAT I10')
            WRITE(IUNIT,624)
        624 FORMAT(1H0,'? ')
            READ(JUNIT,101) NUMBER,KSTOP,NQ,ISIZE,TARRIV,IOUT
        101 FORMAT(4I10,F10.3,I10)
            IF(ISIZE.EQ.0) GO TO 799
            WRITE(IUNIT,626)
        626 FORMAT(1H0,'INPUT CAPACITIES OF LINES SEPARATED BY COMMAS,
           1 FORMAT I10')
            WRITE(IUNIT,628)
        628 FORMAT(1H0,'? ')
            READ(JUNIT,102)(ICAP(I),I=1,NQ)
        102 FORMAT((7I10))
      C  PARAMETERS FOR RANDOM NUMBER GENERATORS.
        799 WRITE(IUNIT,640)
        640 FORMAT(1H0,'INPUT PARAMETERS FOR RANDOM NUMBER
           1 GENERATORS AS DEFINED. FOLLOW')
            WRITE(IUNIT,642)
        642 FORMAT(1H ,'THE ALPHA FORMAT SPECIFIED AND SEPARATE
           1 ALL VARIABLES BY COMMAS. ')
            WRITE(IUNIT,644)
        644 FORMAT(1H ,'*****************************************
           1*****************************')
            WRITE(IUNIT,646)
        646 FORMAT(1H0,'SERVICE TIME UNIFORMLY DISTRIBUTED OVER
           1 THE INTERVAL (A,B),')
            WRITE(IUNIT,648)
        648 FORMAT(1H ,'FORMAT F10.3')
            WRITE(IUNIT,650)
        650 FORMAT(1H ,'STDEV--STANDARD DEVIATION FOR NORMALLY
           1 DISTRIBUTED RANDOM')
            WRITE(IUNIT,651)
        651 FORMAT(1H ,'       VARIABLE WHEN SERVICE TIME IS A
           1 FUNC. OF WAITING TIME, ')
            WRITE(IUNIT,652)
        652 FORMAT(1H ,'       FORMAT F10.3')
            WRITE(IUNIT,653)
```

```
  653 FORMAT(1H ,'X--MEAN OF THE NORMALLY DISTRIBUTED
     1 RANDOM VARIABLE, FORMAT F10.3')
      WRITE(IUNIT,654)
  654 FORMAT(1H ,'XMU--MEAN OF THE NEGATIVE EXPONENTIAL
     1 DISTRIBUTION FOR')
      WRITE(IUNIT,656)
  656 FORMAT(1H ,'     INTERARRIVAL TIMES, FORMAT F10.3')
      WRITE(IUNIT,658)
  658 FORMAT(1H ,'Q--MINIMUM WAITING TIME BEFORE SERVICE
     1 TIME BECOMES A FUNC.')
      WRITE(IUNIT,660)
  660 FORMAT(1H ,'   OF THE WAITING TIME AT THE PREVIOUS
     1 QUEUE, FORMAT F10.3')
      WRITE(IUNIT,661)
  661 FORMAT(1H0,'*')
      READ(JUNIT,662) A,B,STDEV,X,XMU,Q
  662 FORMAT(6F10.3)
      IX=0
      IY=0
      DO 10 I=1,100
   10 CALL RANDU(IX,IY,YFL)
C  INITIALIZATIONS
C  S(I) IS SERVICE TIME AT QUEUE(I).  TDEPRT IS DEPARTURE TIME
C  FROM QUEUE(I). W(I) IS WAITING TIME AT QUEUE(I).  NUMB(I) IS
C  NUMBER OF CUSTOMERS IN QUEUE(I) UPON AND INCLUDING ARRIVAL OF
C  CURRENT CUSTOMER.
      CALL OUTPT(IOMODE,K,I,IOUT)
      IOMODE=2
      IO1=NQ*(IOUT-1) +1
      IO2=NQ*(IOUT)
      TASAVE=TARRIV
      DO 2 I=1,NQ
      L(I)=0
      NUMB(I)=1
    2 W(I)=0.
      K=1
      CALL OUTPT(IOMODE)
      CALL ANALYS(IOMODE,K,IOUT)
      IOMODE=3
C  COMPUTE STATE EQUATIONS FOR CUSTOMER(1) AT ALL QUEUES
      DO 3 I=1,NQ
      IF(I.EQ.1)GO TO 9
      S(I)=SGET(I,W(I-1))
      GO TO 13
    9 S(I)=SGET(I)
   13 TDEPRT=TARRIV+W(I)+S(I)
      IPOS=L(I)+1
      CALL PUSHUP(2,TDEPRT,I,IPOS,NOFF)
      CALL OUTPT(IOMODE,K,I)
    3 TARRIV=TDEPRT
      CALL ANALYS(IOMODE,K,IOUT)
C  MAJOR LOOP -- COMPUTE STATE EQUATS. FOR CUST.(K), FOR K .GT. 1
      DO 44 K=2,KSTOP
      DO 5 I=1,NQ
    5 SOLD(I)=S(I)
C  GET INTERRARIVAL TIME RINT
      RINT=RGET(K)
C  INTERIOR LOOP-- COMPUTE STATE EQUATIONS FOR CUST.(K)
C  AT EACH QUEUE.
      DO 4 I=1,NQ
      TARRIV=TIMEA(I)
      IF (ISIZE.EQ.1) GO TO 6
C  CHECK FOR PREVIOUS DEPARTURES SO # OF CUSTOMERS CAN BE COMPUTD
    7 CALL PUSHUP(1,TARRIV,I,IPOS,NOFF)
      NUMB(I)=NUMB(I)+1-NOFF
    8 IF(I.EQ.1) TASAVE=TARRIV
C  GET WAITING TIME OF CURRENT CUSTOMER IN QUEUE(I)
      W(I)=WATE(I)
      IF(I.EQ.1)GO TO 12
C  GET SERVICE TIME OF CURRENT CUSTOMER AT QUEUE(I)
      S(I)=SGET(I,W(I-1))
      GO TO 14
   12 S(I)=SGET(I)
C  COMPUTE DEPARTURE TIME OF CURRENT CUSTOMER FROM QUEUE(I)
   14 TDEPRT=TARRIV+W(I)+S(I)
      IPOS=L(I)+1
C  ADJUST TD-VECTOR
      CALL PUSHUP(2,TDEPRT,I,IPOS,NOFF)
      CALL OUTPT(IOMODE,K,I)
      GO TO 4
C  LINE LENGTHS NOT INFINITE.  CHECK TO SEE IF BOUNDING OCCURS.
    6 CALL PUSHUP(3,TARRIV,I,IPOS,NOFF)
      SURPLS=NUMB(I)-NOFF-ICAP(I)
      IF(SURPLS.LE.0.) GO TO 7
C  BOUNDING OCCURS
C  ADJUST SERVICE TIME OF CURRENT CUSTOMER AT PREVIOUS SERVER OR,
C  IF I=1, DELAY ARRIVAL TIME OF CURRENT CUSTOMER.
      DUMMY=BLOCK(I)
      WRITE(IUNIT,700)K,I
  700 FORMAT(1H ,'CUSTOMER',I4,'BLOCKED AT QUEUE',I4)
      IF (I.NE.1) WRITE(IUNIT,702)DUMMY
  702 FORMAT(1H ,'SERV. TIME & DEPRT. TIME AT PREVIOUS QUEUE
     1 ARE INCRMTED. BY',F6.2)
      GO TO 7
    4 CONTINUE
   44 CALL ANALYS(IOMODE,K,IOUT)
      IOMODE=4
      CALL OUTPT(IOMODE)
   89 CALL ANALYS(IOMODE)
      STOP
      END
```

```
FORTRAN IV      V01C-03

      SUBROUTINE PUSHUP(MODE,T,I,IPOS,NOFF)
C  THIS IS THE LIST PROCESSOR WRITTEN TO HANDLE UPDATING OF TD-
C  VECTORS. TD-VECTORS ARE VECTORS OF SCHEDULED DEPARTURE TIMES
C  FOR EACH QUEUE. L(I) IS LENGTH OF ITH ROW OF MATRIX VEC.
C  IN ALL MODES NOFF = NUMBER OF ELEMENTS TO BE PUSHED OFF TOP OF
C  VECTOR VEC(I).
C  MODE=1 --- COMPARE T WITH ROW I OF VEC.  REMOVE ELEMENTS .LE.T
C  MODE=2 --- ADD T TO VEC(I) IN POSITION (IPOS) IN VECTOR.
C  MODE=3 --- OBTAIN PUSHUP VALUE ONLY.
      COMMON/BL4/VEC(20,20),L(20)
      LL=L(I)
      GO TO (1,2,1),MODE
    1 IF(LL.GT.0) GO TO 25
      NOFF=0
      RETURN
   25 DO 55 J=1,LL
      IF (T.LT.VEC(I,J)) GO TO 66
   55 CONTINUE
      J=J+1
   66 NOFF=J-1
      IF(MODE.EQ.3) RETURN
      IF(NOFF.EQ.0) RETURN
      L(I)=L(I)-NOFF
      IF(L(I).GT.0) GO TO 17
      RETURN
    2 L(I)=L(I)+1
      IF(IPOS.GT.(L(I)-1)) GO TO 90
      LL=L(I)
      JJ=IPOS+1
      DO 91 J=LL,JJ
   91 VEC(I,J)=VEC(I,J-1)
   90 VEC(I,IPOS)=T
      RETURN
   17 LL=L(I)
      DO 7 J=1,LL
    7 VEC(I,J)=VEC(I,J+NOFF)
      RETURN
      END
```

```
FORTRAN IV      V01C-03

      SUBROUTINE OUTPT(MODE,K,I,IOUT)
C  OUTPUT WRITER (PROGRAM WRITES OUTPUT EVERY IOUT CUSTOMERS)
      COMMON/BL1/NUMBER,KSTOP,NQ,ISIZE,TARRIV,ICAP(20)
      COMMON/BL2/NUMB(20),W(20),S(20),TDEPRT,RINT,TASAVE
      COMMON/BL3/IO1,IO2
      COMMON/BL10/IUNIT,JUNIT
      GO TO (1,2,3,4),MODE
C  WRITE INITIAL CONDITIONS
    1 WRITE(IUNIT,108)
  108 FORMAT(/////)
      WRITE(IUNIT,100)NUMBER
  100 FORMAT(1H ,'RUN NUMBER=',I3)
      WRITE(IUNIT,500) NQ
  500 FORMAT(1H ,I3,' SINGLE SERVER FCFS QUEUES IN TANDEM')
      WRITE(IUNIT,501) TARRIV
  501 FORMAT(1H ,'TIME OF ARRIVAL OF CUSTOMER(1)=',F5.2)
      WRITE(IUNIT,502) KSTOP
  502 FORMAT(1H ,I4,' CUSTOMERS SIMULATED')
      WRITE(IUNIT,503) IOUT
  503 FORMAT(1H ,'OUTPUT WRITTEN EVERY',I4,' CUSTOMERS')
      WRITE(IUNIT,504)
  504 FORMAT(1H ,'HORIZONTAL SIMULATION SEQUENTIAL BY CUSTOMER')
      IF(ISIZE.EQ.1) GO TO 22
      WRITE(IUNIT,101)
  101 FORMAT(22H INFINITE LINE LENGTHS)
      RETURN
   22 WRITE(IUNIT,102)(ICAP(II),II=1,NQ)
  102 FORMAT(1H ,'BOUNDED LINES'/' BOUNDS ARE=',(7I10))
      RETURN
C   WRITE HEADINGS
    2 WRITE(IUNIT,103)
  103 FORMAT(/5X,5H CUST,4X,6H QUEUE,3X,7H TARRIV,3X,7H NUMBER,6X
     12H W,8X,2H S,5X,7H TDEPRT)
      IO=IO1-1
      RETURN
C  WRITE OPTIONAL OUTPUT DATA FOR CUSTOMER(K) AT QUEUE(I) EVERY
C  IOUT CUSTOMERS
    3 IO=IO+1
      IF(IO.LT.IO1) RETURN
      WRITE(IUNIT,104)K,I,TARRIV,NUMB(I),W(I),S(I),TDEPRT
  104 FORMAT(2I10,F10.3,I10,3F10.3)
      IF(IO.NE.IO2) RETURN
      IO=0
      RETURN
    4 WRITE(IUNIT,105)
  105 FORMAT( 11H END OF RUN)
      RETURN
      END
```

```
      FUNCTION BLOCK(I)
C CURRENT CUSTOMER IS BLOCKED AT QUEUE(I). ADJUST S(I-1) OR RINT
      COMMON/BL1/NUMBER,KSTOP,NQ,ISIZE,TARRIV,ICAP(20)
      COMMON/BL2/NUMB(20),W(20),S(20),TDEPRT,RINT,TASAVE
      COMMON/BL4/VEC(20,20),L(20)
C BLOCK = LENGTH OF TIME CUSTOMER IS BLOCKED AT QUEUE(I)
      BLOCK=VEC(I,1)-TARRIV
      IF(I.EQ.1) GO TO 1
C CUSTOMER MAINTAINS USE OF PREVIOUS SERVER WHILE BLOCKED AT
C QUEUE(I).
      S(I-1)=S(I-1) + BLOCK
      TDEPRT=TDEPRT+BLOCK
      LL=L(I-1)
      VEC(I-1,LL)=TDEPRT
      TARRIV=TARRIV+BLOCK
      RETURN
C CUSTOMER BLOCKED AT QUEUE(1) -- DELAY ARRIVAL INTO SYSTEM
    1 RINT=RINT+BLOCK
      TARRIV=TIMEA(I)
      RETURN
      END
```

```
      FUNCTION TIMEA(I)
C PROGRAM COMPUTES TIME OF ARRIVAL OF CURRENT CUSTOMER AT
C QUEUE(I).
      COMMON/BL2/NUMB(20),W(20),S(20),TDEPRT,RINT,TASAVE
      IF(I.GT.1) GO TO 2
      TIMEA=TASAVE+RINT
      RETURN
    2 TIMEA=TDEPRT
      RETURN
      END
```

```
      FUNCTION SGET(I,W)
C PROGRAM COMPUTES SERVICE TIME AT CURRENT QUEUE.
C THIS PROGRAM SHOULD BE TAILORED TO MEET REQUIREMENTS OF EACH
C SIMULATION.
      COMMON/BL9/A,B,X,STDEV,XMU,Q,IX,IY
      CALL RANDU(IX,IY,YFL)
      IF(I.EQ.1)GO TO 1
      GO TO 2
C SERVICE TIME AT QUEUE(1) IS A UNIFORMLY DISTRIBUTED RANDOM
C VARIABLE ON INTERVAL (A,B)
    1 SGET=A+YFL*(B-A)
      RETURN
C SERVICE TIME IS A FUNCTION OF WAITING TIME AT PREVIOUS QUEUE #
C THE SERIES.
    2 SGET=2
      IF(W.LT.Q) GO TO 3
      RETURN
C SERVICE TIME AT QUEUE(I) IS A NORMALLY DISTRIBUTED RANDOM
C VARIABLE WITH STANDARD DEVIATION 'STDEV' AND MEAN 'X'.
    3 CALL GAUSS(ANORML)
      SGET=ANORML
      RETURN
      END
```

```
      SUBROUTINE GAUSS(ANORML)
C SUBROUTINE GAUSS IS A DIRECT METHOD OF DERIVING NORMALLY
C DISTRIBUTED RANDOM NUMBERS,BASED UPON THE PROPERTIES OF
C RANDOM NUMBERS RATHER THAN AN EVALUATION OF THE INVERSE
C FUNCTION.
      COMMON/BL9/A,B,X,STDEV,XMU,Q,IX,IY
      P=0.0
      DO 10 IR=1,12
      CALL RANDU(IX,IY,YFL)
      P=P+YFL
   10 CONTINUE
      ANORML=(P-6.0)*STDEV+X
      RETURN
      END
```

```
      FUNCTION RGET(K)
C  PROGRAM COMPUTES INTERARRIVAL TIME BETWEEN CURRENT AND PREVIOS
C     CUSTOMER.
C     ARRIVALS ARE RANDOM
C INTERARRIVAL TIMES GENERATED FROM NEGATIVE EXPON. DISTRIBUTION.
      COMMON/BL9/A,B,X,STDEV,XMU,Q,IX,IY
      CALL RANDU(IX,IY,YFL)
      RGET=(-XMU*ALOG(YFL))
      RETURN
      END
```

```
      FUNCTION WATE(I)
C  PROGRAM COMPUTES WAITING TIME W(I) OF CURRENT CUSTOMER AT
C  QUEUE(I). W(I) DOES NOT INCLUDE SERVICE TIME.
      COMMON/BL2/NUMB(20),W(20),S(20),TDEPRT,RINT,TASAVE
      COMMON/BL5/SOLD(20)
      REAL LAMDA
      LAMDA=0.
      IF(I.GT.1) GO TO 6
      WSTAR=W(1)+SOLD(1)-RINT
      GO TO 7
    6 IF(WSTAR.LT.0.) LAMDA=1.
      WSTAR=W(I)+SOLD(I)-S(I-1)+LAMDA*WSTAR
    7 W(I)=0.
      IF(WSTAR.GT.0.) W(I)=WSTAR
      WATE=W(I)
      RETURN
      END
```

```
      SUBROUTINE ANALYS(MODE,K,IOUT)
C  PROGRAM COMPUTES CUSTOMER STATISTICS -- TRANSIT TIME.
C  AVERAGE CUSTOMER TRANSIT TIME IS COMPUTED.
      COMMON/BL1/NUMBER,KSTOP,NQ,ISIZE,TARRIV,ICAP(20)
      COMMON/BL2/NUMB(20),W(20),S(20),TDEPRT,RINT,TASAVE
      COMMON/BL10/IUNIT,JUNIT
C  TRANS=TRANSIT TIME FOR CURRENT CUSTOMER.  TSUM=CUMULATIVE
C  TRANSIT TIME.
      GO TO (2,2,3,4),MODE
    3 TRANS=0.
      DO 1 I=1,NQ
    1 TRANS=TRANS+S(I)+W(I)
      TSUM=TSUM+TRANS
      IO=IO+1
      IF(IO.NE.IOUT) RETURN
      IO=0
      WRITE(IUNIT,100)K,TRANS
  100 FORMAT(1H ,'TRANSIT TIME FOR CUST',I3,' =',F10.3/)
      RETURN
    2 IO=IOUT-1
      TSUM=0.
      RETURN
    4 AVT=TSUM/KSTOP
      WRITE(IUNIT,101)AVT
  101 FORMAT(1H ,'AVGE TRANSIT TIME=',F10.3)
      RETURN
      END
```

```
RUN$EXEC
*QHORFL/CORE:26

INPUT DATA FOR THIS SIMULATION RUN AS DEFINED. FOLLOW THE ALPHA
FORMAT SPECIFIED AND SEPARATE ALL VARIABLES BY COMMAS.
*****************************************************************

INPUT NRUNS--NUMBER OF RUNS, FORMAT I10

?
1


NUMBER-- RUN ID NUMBER, FORMAT I10
KSTOP-- NUMBER OF CUSTOMERS TO BE SIMULATED,  FORMAT I10
NQ-- NUMBER OF QUEUES IN TANDEM, FORMAT I10
ISIZE-- EQUALS 1 IF BOUNDED, 0 OTHERWISE
TARRIV-- TIME OF ARRIVAL OF FIRST CUSTOMER,  FORMAT F10.3
IOUT-- OPT. OUTPUT WRITTEN EVERY IOUT  CUSTOMERS, FORMAT I10

?
201,101,5,1,3.,20


INPUT CAPACITIES OF LINES SEPARATED BY COMMAS, FORMAT I10

?
4,2,2,1,2


INPUT PARAMETERS FOR RANDOM NUMBER GENERATORS AS DEFINED. FOLLOW
THE ALPHA FORMAT SPECIFIED AND SEPARATE ALL VARIABLES BY COMMAS.
*****************************************************************
```

```
SERVICE TIME UNIFORMLY DISTRIBUTED OVER THE INTERVAL (A,B),
FORMAT F10.3
STDEV--STANDARD DEVIATION FOR NORMALLY DISTRIBUTED RANDOM
       VARIABLE WHEN SERVICE TIME IS A FUNC. OF WAITING TIME,
       FORMAT F10.3
X--MEAN OF THE NORMALLY DISTRIBUTED RANDOM VARIABLE, FORMAT F10.3
XMU--MEAN OF THE NEGATIVE EXPONENTIAL DISTRIBUTION FOR
     INTERARRIVAL TIMES, FORMAT F10.3
Q--MINIMUM WAITING TIME BEFORE SERVICE TIME BECOMES A FUNC.
   OF THE WAITING TIME AT THE PREVIOUS QUEUE, FORMAT F10.3

?
2.,4.,1.,3.,4.,1.5



RUN NUMBER=201
  5 SINGLE SERVER FCFS QUEUES IN TANDEM
TIME OF ARRIVAL OF CUSTOMER( 1)= 3.00
 101 CUSTOMERS SIMULATED
OUTPUT WRITTEN EVERY  20 CUSTOMERS
HORIZONTAL SIMULATION SEQUENTIAL BY CUSTOMER
BOUNDED LINES
BOUNDS ARE=         4          2          2          1          2

     CUST     QUEUE    TARRIV    NUMBER      W         S       TDEPRT
        1         1     3.000         1    0.000     2.986      5.986
        1         2     5.986         1    0.000     2.699      8.686
        1         3     8.686         1    0.000     4.984     13.670
        1         4    13.670         1    0.000     2.845     16.514
        1         5    16.514         1    0.000     2.544     19.058
TRANSIT TIME FOR CUST  1 =    16.058

CUSTOMER  13BLOCKED AT QUEUE   1
CUSTOMER  18BLOCKED AT QUEUE   4
SERV. TIME & DEPRT. TIME AT PREVIOUS QUEUE  ARE INCRMTED. BY  1.05
       21         1    73.385         3    4.875     3.731     81.990
       21         2    81.990         1    0.000     2.000     83.990
       21         3    83.990         2    1.543     3.166     88.699
       21         4    88.699         0    0.000     2.000     90.699
       21         5    90.699         2    1.016     1.338     93.053
TRANSIT TIME FOR CUST 21 =    19.668

       41         1   165.976         0    0.000     3.027    169.003
       41         2   169.003         1    0.000     3.417    172.420
       41         3   172.420         0    0.000     0.922    173.341
       41         4   173.341         0    0.000     3.194    176.535
       41         5   176.535         0    0.000     2.005    178.540
TRANSIT TIME FOR CUST 41 =    12.564

       61         1   230.590         3    5.750     2.262    238.602
       61         2   238.602         1    0.000     2.000    240.602
       61         3   240.602         2    0.779     2.838    244.220
       61         4   244.220         2    1.500     4.375    250.094
       61         5   250.094         1    0.623     3.310    254.027
TRANSIT TIME FOR CUST 61 =    23.437

CUSTOMER  64BLOCKED AT QUEUE   1
       81         1   305.206         0    0.000     3.351    308.557
       81         2   308.557         1    0.000     3.648    312.205
       81         3   312.205         0    0.000     2.424    314.630
       81         4   314.630         0    0.000     3.737    318.366
       81         5   318.366         0    0.000     2.249    320.615
TRANSIT TIME FOR CUST 81 =    15.409

      101         1   381.410         3    6.806     3.257    391.473
      101         2   391.473         0    0.000     2.000    393.473
      101         3   393.473         1    0.136     2.407    396.015
      101         4   396.015         2    2.835     2.549    401.399
      101         5   401.399         2    0.936     2.000    404.335
TRANSIT TIME FOR CUST101 =    22.925

END OF RUN
AVGE TRANSIT TIME=    19.330
STOP --

READY

BYEF
```