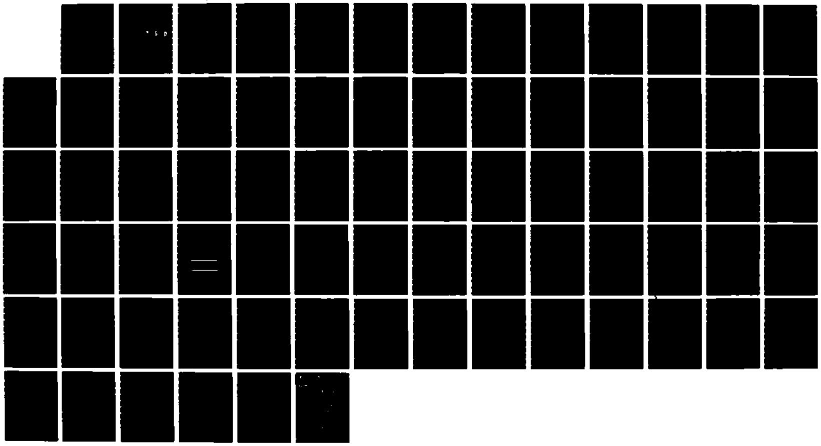
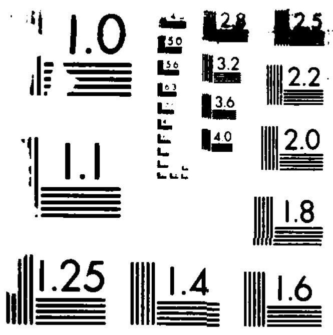
AD-A175 259 GRAPHIC SIMULATION OF A JACKSON NETWORK(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA G F GREENE SEP 86 1/1

UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART

[illegible]

(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

GRAPHIC SIMULATION OF A JACKSON NETWORK

by

Gary F. Greene

September 1986

Thesis Advisor: James D. Esary

ADA 175 259

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) Code 55	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
GRAPHIC SIMULATION OF A JACKSON NETWORK

12 PERSONAL AUTHOR(S)
Greene, Gary F.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ______ TO ______	1986 September	73

16 SUPPLEMENTARY NOTATION

17 COSATI CODES

FIELD	GROUP	SUB-GROUP

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Simulation, Poisson Process, Networks of Queues, Jackson Network, Personal Computer, Computer Graphics.

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This paper presents the development of a graphic computer simulation of a Jackson network. In particular, the simulated network is a queueing system consisting of two servers in tandem, one customer entrance point, two potential customer exit points, and one potential customer feedback path.

This system is modeled as a discrete-event simulation implemented on an IBM Personal Computer (IBM PC). The IBM PC requires a color monitor, a color/graphics adapter card, at least one disk drive, and a minimum of 128 kilo-bytes of random access memory (RAM).

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL James D. Esary	22b TELEPHONE (Include Area Code) (408) 646-2780 22c OFFICE SYMBOL Code 55Ey

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

Graphic Simulation of a Jackson Network

by

Gary F. Greene
Lieutenant, United States Coast Guard
B.S., United States Coast Guard Academy, 1977
M.B.A., University of Puget Sound, 1983

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1986

Author: Gary F Greene
Gary F. Greene

Approved by: J. D. Esary
J.D. Esary, Thesis Advisor

[illegible]
G.F. Lindsay, Second Reader

[illegible]
Alan R. Washburn, Chairman,
Department of Operations Research

Kneale T. Marshall
Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

This ~~paper~~ presents the development of a graphic computer simulation of a Jackson network. In particular, the simulated network is a queueing system consisting of two servers in tandem, one customer entrance point, two potential customer exit points, and one potential customer feedback path.

This system is modeled as a discrete-event simulation implemented on an IBM Personal Computer (IBM PC). The IBM PC requires a color monitor, a color/graphics adapter card, at least one disk drive, and a minimum of 128 kilo-bytes of random access memory (RAM). [illegible]

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

This paper presents the development of a computer simulation of a specific example of a Jackson network.

In a typical computer simulation, a series of mathematical and logical relationships are used to create a model of some real world system. The model is then exercised over some period of time, and data are collected to estimate the desired characteristics of the real word system. When the simulation run is completed, the desired statistics are printed out.

What makes this simulation different, is that it also provides the user with a visual picture of what is occuring in the queue network as the simulation progresses. It accomplishes this by actually displaying on the computer screen the arriving and departing customers and showing how they *move* through the network. This portrayal is possible because the simulation gives the observer the impression of real-time passage.

Because Jackson networks, and their associated statistical quantities of interest can be solved in closed mathematical form, there is little gained in this respect from simulation. However, by allowing a student to actually observe the inner workings of the queue network as it progresses in time, insights into the process can be obtained that are not possible from conventional instructional tools alone.

This paper is the third known effort at the Naval Postgraduate School to develop visual simulations of various stochastic processes utilizing the personal computer. Prior undertakings are a simulation of various models that are based on an underlying Poisson process [Ref. 1], and a simulation of a Machine-Repairman Model [Ref. 2]. These two papers provided many ideas for the writing of the Jackson network simulation and served as an outline for this paper.

The remainder of this paper describes the Jackson network model and its implementation on the IBM Personal Computer (IBM PC). Chapter IV is a user's guide which contains instructions for actually running the simulation. A program listing is contained in Appendix A.

II. THE JACKSON NETWORK MODEL

A. GENERAL DISCUSSION

In a network of queues, a customer[1] may receive service from one or more service centers, possibly revisiting some or all of the service centers many times before finally exiting the network. With no additional assumptions made, this type of queue network can be extremely difficult, if not impossible to analyze in closed mathematical form. In these cases, computer simulation may be the only alternative [Ref. 3: p. 225].

One type of network that does permit analysis without the need for simulation is the Jackson network. A Jackson network is a collection of queues that possess the following characteristics [Ref. 4: p. 456]:

- The network consists of N service centers numbered 1,2,...,N. Service center i contains c_i identical servers.
- Customers from outside the network arrive at service center i according to a Poisson process with rate λ_i, i = 1,2,...,N.
- After receiving service at center i, a customer leaves the network with probability $P_{i0} > 0$ or goes instantaneously to service center j with probability P_{ij}, $\sum_{j=1}^{N} P_{ij} = 1$. These probabilities are independent of the history of the network.
- Customers arriving at service center i (from inside or outside the network) are served according to a First Come First Serve (FIFO) queue discipline. The service times are independent, identically distributed exponential random variables with mean $1/\mu_i$, i = 1,2,...,N.
- There is unlimited waiting space at every service center.
- The effective arrival rate at each service center taking into account arrivals from both outside and inside the network must be less than the corresponding service rates.

Two further characteristics of a Jackson network result from the requirement that customers from outside the network arrive at the service centers according to a Poisson process. This implies that the arrival process must have *independent* and *stationary* increments. An arrival process is said to possess independent increments if, for example, the number of arrivals that occur between times 5 and 10 are independent of the number of arrivals between times 35 and 40. An arrival process has stationary increments if the distribution of the number of events which occur in any interval of time depends only on the length of the time interval. This is the same as saying that

[1]Although this paper will refer to customers, implying people, the flow through a queue network could just as easily be an object or a job.

there is no time of day effect on the arrival rate of customers. One further assumption of a Jackson network is that the servers at the different service centers work independently and do not assist one another, even if one is idle.

B. THE MODELED JACKSON NETWORK

The simulation described in this paper is a specific example of a Jackson network where $N = 2$, $c_1 = 1$, $c_2 = 1$, $\lambda_1 > 0$, $\lambda_2 = 0$, $\mu_1 > 0$, $\mu_2 > 0$, $P_{11} = 0$, $P_{22} = 0$, $P_{10} \geq 0$, $P_{20} \geq 0$, $P_{12} = 1 - P_{10}$, and $P_{21} = 1 - P_{20}$. A pictorial representation of this network is provided in Figure 2.1. In words, a customer arriving from outside the network must first go to service center 1 where he enters a FIFO queue. Upon completion of service at service center 1, the customer can either exit the system with probability P_{10} or proceed to service center 2 with probability $1 - P_{10}$, where he again enters a FIFO queue. Upon completion of service at service center 2, the customer can either exit the network with probability P_{20} or loop back to service center 1 with probability $1 - P_{20}$ again entering the FIFO queue. In this configuration a customer can loop from service center 1 to service center 2 and back to service center 1 numerous times before exiting the network. Each service center has only one server.

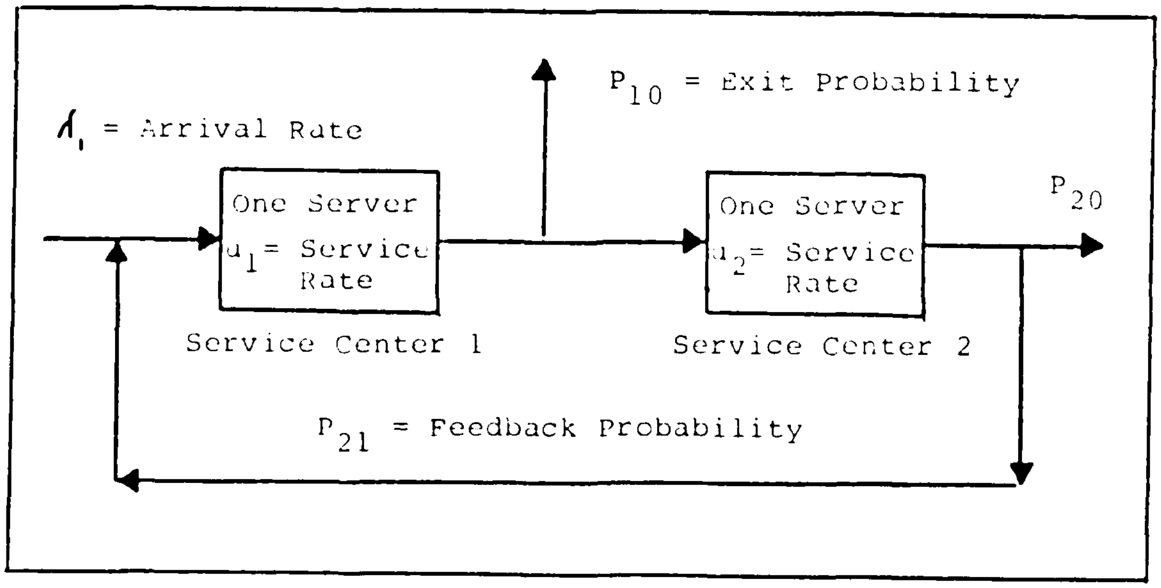


Figure 2.1 Jackson Network With Two Service Centers.

It is arguable whether any real world system would ever meet all of the requirements of a Jackson network. But then that is always the predicament faced by the modeler; making the model represent the real world without making it unreasonably complex. There are numerous real world systems that might be reasonably modeled by this simple Jackson network. For example, the network might describe the flow of a piece of equipment through a repair facility. Upon entering service center 1, the repairman evaluates the piece of equipment and attempts to repair it. If it can not be repaired, it leaves the network. If it is repairable, it is repaired and proceeds to the Quality Assurance Shop (service center 2). At Quality Assurance, the repaired item undergoes testing and is either classified as fit or unfit. If it is determined fit, it departs the network and returns to the field. If it is determined unfit, it is returned to service center 1 for additional repair.

Another possible example would be a computer system. Jobs arrive at the central processing unit (service center 1) of a computer. When a job finishes CPU service it either departs the computer system or requires Input/Output service (service center 2). When a job completes IO service it either leaves the computer system or returns to the CPU for additional service.

For this Jackson network model, the state of the system, $N(t) = n$, is defined as the total number of customers in the system both waiting and being served at time t. This state variable is further broken down into the total number of customers waiting and being served by service center 1, $N_1(t) = n_1$, and the total number of customers waiting and being served by service center 2, $N_2(t) = n_2$. It follows that $N(t) = N_1(t) + N_2(t)$. Because arrivals and departures are discrete events, the value of $N(t)$ will always be zero or a positive integer. However, because there is unlimited waiting space in front of the service centers, there is no upper limit on the value that $N(t)$ can assume. That is, n can be any integer value between 0 and positive infinity.

C. ANALYSIS OF JACKSON NETWORKS

It is not the intent of this section to develop in detail the theoretical analysis of Jackson networks. However, a general discussion will be included for completeness. The interested reader should refer to the reference list and the bibliography for books on queueing theory, paying particular attention to the chapters on networks of queues. Of particular interest might be the paper written by J.R. Jackson for whom this type of network is named.

By imposing the restrictions described in Section I-A on a network, Jackson showed that in the steady state each service center (say center i) in the network *behaved* as if it were an independent M/M/c queue with service rate u_i and Poisson input rate R_i. The input rate, R_i, can be considered the effective arrival rate at service center i. The effective arrival rate to a service center is the sum of the Poisson arrivals from outside the system (exogenous customers), plus the arrivals from any other service center in the network. The effective arrival rates can be found by solving the following set of simultaneous equations:

$$R_i = \lambda_i + \sum_{j=1}^{N} R_j P_{ij} \quad i = 1,2,\ldots,N. \qquad \text{(eqn 2.1)}$$

In order to prevent an unstable or *explosive* queue from occurring at any of the service centers, it is required that $R_i < c_i u_i$. In words, the effective arrival rate of customers at service center i must be less than the cumulative service rate of the servers at service center i.

The implications of Jackson's findings are that a complex network can be decomposed into a number of simpler M/M/c sub-systems. These sub-systems can then be analyzed using Little's formula and the standard results for the M/M/c queue.

An interesting aspect of this result is that it can be proved that allowing feedback in the network destroys the Poisson flow of arrivals [Ref. 4: p. 460]. However Jackson's results show that even for networks that allow feedback, the service centers *behave* as if they were fed totally by Poisson arrivals, when in fact they are not. Statisticians have been unable to provide an explanation as to why this is true [Ref. 5: p. 330].

D. MODEL ESTIMATIONS

The purpose of any simulation is to gain some insight into the real world system being modeled. This insight is usually gained by estimating certain quantities pertaining to the systems performance. As a part of the graphical output, the simulation provides a table of estimates for a number of these quantities. Because Jackson networks can be analyzed in closed form, the graphical output will also include the theoretical values for these same quantities. This will provide the user with a measure of how well the computer simulation is modeling the real world system.

The following equilibrium performance quantities will be both estimated and determined theoretically by the simulation:

R_i = the effective arrival rate of customers at service center i ($i = 1,2$).

S_i = the expected number of times a customer will pass through service center i before exiting the network ($i = 1,2$).

L_i = the expected number of customers both waiting and being served at service center i ($i = 1,2$).

L = the expected number of customers in the network, both waiting and being served.

W_i = the expected amount of time a customer spends waiting and being served at service center i ($i = 1,2$).

W = the total expected amount of time a customer spends in the network, both waiting and being served.

Z = the proportion of time service center one is busy and service center two is idle.

The theoretical values will be represented by the appropriate capital letter. The estimated values will be represented by the appropriate capital letter with a *hat* over it.

The following quantities are parameters of the model which can be selected by the user prior to running the simulation and are used in determining the theoretical quantities just listed. The parameter names are consistent with those used in the simulation model.

λ = the Poisson arrival rate of exogenous customers arriving at service center 1 (previously referred to as λ_1).

μ_i = the exponential service rate for service center ($i = 1,2$).

P_i = the probability of exiting the network after completing service at center i, $i = 1,2$ (previously referred to as P_{i1}).

$1 - P_1$ = the probability of proceeding to service center 2 after completing service at service center 1 (previously referred to as P_{12}).

$1 - P_2$ = the probability of proceeding to service center 1 after completing service at service center 2 (previously referred to as P_{21}).

For use in estimating some of the performance quantities, λ and μ_i will also be estimated by the simulation model.

The values of R_1 and R_2, the theoretical effective arrival rates at service center 1 and service center 2, are found by solving equation 2.1, namely:

$$R_1 = \lambda + R_2(1-P_2) \qquad \text{(eqn 2.2)}$$

$$R_2 = 0 + R_1(1-P_1), \qquad \text{(eqn 2.3)}$$

which results in:

$$R_1 = \lambda/\{1-(1-P_1)(1-P_2)\} \qquad \text{(eqn 2.4)}$$

$$R_2 = (1-P_1)R_1. \qquad \text{(eqn 2.5)}$$

The estimated value of R_1 is computed by recording the total number of customer arrivals at service center 1, both exogenous customers and feedback customers from service center 2, and dividing by the total elapsed time that the simulation has run. The estimated value of R_2 is calculated in a similar manner except that there is only one source of arrivals to service center 2, namely those customers departing service center 1 who do not exit the network.

The estimated value of λ, the Poisson arrival rate of exogenous customers arriving at service center 1 is computed by recording the number of exogenous customers arriving at service center 1 and dividing by the total elapsed time that the simulation has run.

The estimated value of μ_1, the exponential service rate for service center 1 is computed by recording the total number of customers completing service at service center 1, and dividing by the total amount of time service center 1 was busy (*NOT* the total elapsed time of the simulation). The estimated value of μ_2 is computed in a similar fashion.

The value of L_1 and L_2, the theoretical expected number of customers either waiting or being served at service center 1 and 2, are found by utilizing Little's law and the results for a M/M/1 queue:

$$L_1 = R_1/(\mu_1-R_1) \qquad \text{(eqn 2.6)}$$

$L_2 = R_2/(\mu_2 - R_2)$. (eqn 2.7)

It follows that L, the expected number of total customers in the network, both waiting and being served is simply the sum of L_1 and L_2:

$L = L_1 + L_2$. (eqn 2.8)

The estimates $\hat{L}_1$, $\hat{L}_2$ and $\hat{L}$ are found by substituting $\hat{R}_1$, $\hat{R}_2$, $\hat{\mu}_1$, and $\hat{\mu}_2$ into the above equations to obtain:

$\hat{L}_1 = \hat{R}_1/(\hat{\mu}_1 - \hat{R}_1)$ (eqn 2.9)

$\hat{L}_2 = \hat{R}_2/(\hat{\mu}_2 - \hat{R}_2)$ (eqn 2.10)

$\hat{L} = \hat{L}_1 + \hat{L}_2$. (eqn 2.11)

The values of W_1 and W_2, the theoretical expected time a customer spends waiting and being served at service center 1 and 2, are also found by utilizing Little's law and the results for a M/M/1 queue:

$W_1 = L_1 / R_1$ (eqn 2.12)

$W_2 = L_2 / R_2$. (eqn 2.13)

The value of W, the total expected time a customer spends in the network, both waiting and being served is also computed using Little's formula:

$W = L/\lambda$. (eqn 2.14)

The estimates $\hat{W}_1$, $\hat{W}_2$, and $\hat{W}$ are found by substituting $\hat{L}$, $\hat{L}_1$, $\hat{L}_2$, $\hat{R}_1$, $\hat{R}_2$, and $\hat{\lambda}$ into the above equations to obtain:

$\hat{W}_1 = \hat{L}_1 / \hat{R}_1$ (eqn 2.15)

$\hat{W}_2 = \hat{L}_2 / \hat{R}_2$ (eqn 2.16)

$\hat{W} = \hat{L}/\hat{\lambda}.$ (eqn 2.17)

The value of Z, the theoretical long run proportion of time that service center 1 is busy and service center 2 is idle, is found by using a result commonly referred to as Jackson's Theorem. This theorem, when simplified for the situation where each service center has exactly one server, states that for a Jackson network with $R_i < u_i$ and with k service centers the limiting probability is given by:

$\text{Prob}(n_i \text{ at service center } i,\ i=1,2,\ldots k) = \pi(n_1,n_2,\ldots n_k) =$ (eqn 2.18)

$\pi(n_1)\pi(n_2)\ldots\pi(n_n) = \prod_{i=1}^{k}(R_i/\mu_i)^{n_i}(1-R_i/\mu_i).$

Where n_i is the number of customers waiting and being served at service center i, and $\pi(n_i)$ is the long run probability of finding n_i customers at service center i. The theorem says that the limiting probability is equal to the product of the marginal probabilities for the individual M/M/1 queues comprising the network.

To determine the value of Z, it is necessary to determine the probability that there are no customers at service center 2 and one *or more* at service center 1. This probability is determined as follows:

$Z = \{1 - \pi(n_1=0)\}\pi(n_2=0)$ (eqn 2.19)

$= \{1-(R_1/\mu_1)^0(1-R_1/\mu_1)\}\{(R_2/\mu_2)^0(1-R_2/\mu_2)\}$ (eqn 2.20)

$= (R_1/\mu_1)(1 - R_2/\mu_2).$ (eqn 2.21)

The estimate $\hat{Z}$ is found by substituting $\hat{R}_1$, $\hat{R}_2$, $\hat{\mu}_1$, and $\hat{\mu}_2$ into equation 2.21 to obtain:

$\hat{Z} = (\hat{R}_1/\hat{\mu}_1)(1 - \hat{R}_2/\hat{\mu}_2).$ (eqn 2.22)

The value of S_1, the theoretical expected number of times a customer will pass through service center 1 before exiting the network is found by recognizing that the underlying distribution is geometric. For service center 1, it can be seen that the following relationship holds, where N_1 equals the number of times a customer visits service center 1 before exiting:

$$P(N_1 = 1) = \{P_1 + (1-P_1)P_2\} \qquad \text{(eqn 2.23)}$$

$$P(N_1 = 2) = \{P_1 + (1-P_1)P_2\}\{(1-P_1)(1-P_2)\}^1 \qquad \text{(eqn 2.24)}$$

$$P(N_1 = 3) = \{P_1 + (1-P_1)P_2\}\{(1-P_1)(1-P_2)\}^2 \qquad \text{(eqn 2.25)}$$

.

.

.

$$P(N_1 = n) = \{P_1 + (1-P_1)P_2\}\{(1-P_1)(1-P_2)\}^{n-1} \text{ , } n = 1,2,3.... \qquad \text{(eqn 2.26)}$$

Expanding the right hand side of equation 2.26,

$$P(N = n_1) = (P_1 + P_2 - P_1P_2)^1(1-P_1-P_2+P_1P_2)^{n-1} \qquad \text{(eqn 2.27)}$$

Letting $\theta = 1 - P_1 - P_2 + P_1P_2$, it follows that the quantity raised to the first power in equation 2.27 is equal to $1 - \theta$. Equation 2.26 can be rewritten in the form:

$$P(N = n) = (1-\theta)(\theta)^{n-1},\ n = 1,2,3.... \qquad \text{(eqn 2.28)}$$

Which is recognized as a geometric distribution with an expected value equal to:

$$S_1 = E(N_1) = 1/(P_1 + P_2 - P_1P_2). \qquad \text{(eqn 2.29)}$$

A similar type of analysis can be done to find the theoretical value of S_2. Letting N_2 equal the number of times a customer visits service center 2 before exiting and $\theta = (1-P_1)(1-P_2)$ as before, it can be seen that the following relationships hold:

$$P(N_2=0) = P_1 \qquad \text{(eqn 2.30)}$$

$$P(N_2=1) = \theta P_1 + (1-P_1)P_2 \qquad \text{(eqn 2.31)}$$

$$P(N_2=2) = \theta^2 P_1 + \theta^1(1-P_1)P_2 \qquad \text{(eqn 2.32)}$$

.

.

.

$$P(N_2=n) = \theta^n P_1 + \theta^{n-1}(1-P_1)P_2. \qquad \text{(eqn 2.33)}$$

Taking the expected value:

$$E(N_2) = 0P_1 + 1\theta^1P_1 + 2\theta^2P_1 + \dots + n\theta^nP_1 \qquad \text{(eqn 2.34)}$$
$$+ 1(1-P_1)P_2 + 2\theta^1(1-P_1)P_2 + 3\theta^2(1-P_1)P_2 + \dots + n\theta^{n-1}(1-P_1)P_2$$

$$= P_1\theta\{1 + 2\theta^1 + 3\theta^2 + \dots + n\theta^{n-1}\} \qquad \text{(eqn 2.35)}$$
$$+ (1-P_1)(P_2)\{1 + 2\theta^1 + 3\theta^2 + \dots + n\theta^{n-1}\}.$$

Noting that the geometric sequences converge in the limit to $(1/1-\theta)^2$, equation 2.35 can be rewritten as:

$$E(N_2) = P_1\theta(1/1-\theta)^2 + (1-P_1)(P_2)(1/1-\theta)^2. \qquad \text{(eqn 2.36)}$$

The final result after manipulation of equation 2.36 is:

$$S_2 = E(N_2) = (1-P_2)/(P_1+P_2-P_1P_2) = (1-P_1)S_1. \qquad \text{(eqn 2.37)}$$

The estimates, $\hat{S}_1$ and $\hat{S}_2$, are found by using the definition of expected value for a discrete random variable. Let $P_i(n)$ be a shorter notation for the probability that a customer is served exactly n times at service center i before he departs the network. The estimate $\hat{P}_i(n)$, is determined by recording how many customers were served exactly n times by service center i before departing the network and then dividing this number by the total number of customers who have departed the network. The estimates of S_1 and S_2 are then determined by:

$$\hat{S}_1 = \sum_{n=0}^{\infty} n\hat{P}_1(n) \qquad \text{(eqn 2.38)}$$

$$\hat{S}_2 = \sum_{n=0}^{\infty} n\hat{P}_2(n). \qquad \text{(eqn 2.39)}$$

It should be noted that the simulation program, JACKQUE, is incapable of explicitly tracking an individual customer for more than nine visits to either service center. All customers who exit the network having visited either service center, ten or more times, are counted as if they had visited that center exactly ten times. This results in S_1 and S_2 being under-estimates of the theoretical values. Section III will discuss the implications of this model limitation.

Whenever a quantity is estimated through simulation, the estimate will be a point estimate of the *true* value. This point estimate, being a random variable, will have some variation associated with it. As the simulation proceeds in time, the point estimate will fluctuate around the true value. At any particular point in time the point estimate may or may not be close to the true value. To overcome this, a simulation model is typically run a large number of times using some stopping rule. The ending estimates for each run are then averaged to get more accurate estimates. Some sort of variance reduction technique like Jack-Knifing or Boot-Strapping is also typically employed.

In JACKQUE however, the simulation model is exercised only once, then terminated. For this reason, at any particular point in time, the estimates might not coincide closely with the theoretical values. This is particularly true of the estimates for L, W, and Z which are algebraic combinations of other point estimates. For example, the quantity W_1 is estimated using the quotient $\hat{L}_1/\hat{R}_1$. If at the point in time the estimate is made, $\hat{L}_1$ is larger and $\hat{R}_1$ is smaller than their true values, the resulting estimate of Z will be very much larger than it's true value.

Since the primary purpose of JACKQUE is not estimation, this is not a serious shortcoming of the model.

III. THE SIMULATION

A. GENERAL

The Jackson network is modeled as a discrete-event simulation model. The simulation is both dynamic and stochastic in nature. A discrete system (vice continuous) is one in which the state variables can only change at a countable or finite number of points in time. These points in time are the ones at which an event occurs. For the Jackson network, the state of the system N(t), can only change as a result of an exogenous arrival, or a service completion by either service center.

A dynamic simulation (vice static) is one that changes or evolves in time. The Jackson network, which is driven by inter-arrival times and service times, is obviously of this nature.

A stochastic simulation model (vice deterministic) is one that contains one or more random variables. In other words, some characteristic of the system is random in nature. For the Jackson network, both the exogenous customer arrival rate, and the service rates are random variables generated from an underlying exponential probability distribution.

A dynamic simulation requires some mechanism for keeping track of the passage of elapsed time, allowing the program to know when, and in what order events will occur. In the case of the Jackson network, we need to keep track of when arrivals and service completions occur. The variable in the simulation model that performs this task is called the simulation clock. The unit of time for the clock is usually not explicitly stated, but needs to be consistent with the units of the input parameters.

There are two basic approaches for advancing the simulation clock. It can be accomplished using a *next-event time advance*, or a *fixed increment time advance.* In both cases the clock is initialized to zero and the times of all future events are determined.

With a *fixed-increment* clock, the clock is advanced in increments of Δt. After each Δt increment, a check is made to determine if any events should have occured. If one or more events should have occured, these events are considered to have occurred at the end of the interval, and the system state, N(t), is updated accordingly. Two obvious problems with a fixed-increment clock are that event times appear to be larger than they should be due to the rounding up effect, and if more than one event occurs in Δt it might not be possible to decide in which order the events occurred.

With a *next-event* clock, the successive jumps of the clock are unequal in size, and correspond to the amount of time till the next event occurs. After the simulation clock is advanced to the first of these future events, the state of the system $N(t)$ is updated to account for the occurrence of the event, and times of future events are generated as necessary. The *next-event* clock is used by most simulation languages because it does not introduce bias into the estimates, and by skipping over periods of inactivity the program runs faster.

For purposes of the graphical display, JACKQUE uses a *fixed-increment* clock. The clock is increased in increments of $\Delta t = 1$. This gives the viewer of the graphics the illusion of real time passage. This type of clock is also necessary so that the system state-versus-time graphs can be updated with fixed length horizontal line increments. Although the clock is incremented in fixed increments, and all events are plotted as if they occurred at the end of Δt, the program does preserve the correct chronological order of the events when plotting.

For purposes of the estimated performance measures, the program keeps track of *continuous* time, and does not round event occurrences up to Δt. In this sense, the program behaves as if it is using a *next-event* clock.

B. THE COMPUTER

The computer used for this simulation is the IBM Personal Computer (IBM PC), equipped with a color monitor. In order to run the simulation the IBM PC must be configured with at least 128 kilo-bytes of random access memory (RAM), one 5 1/4" floppy disk drive, and a Color/Graphics adapter. The color monitor and Color/Graphics adapter are essential since the program uses color graphics.

The simulation program, JACKQUE was written using version 2.1 of the PC-DOS disk operating system. Any version of PC-DOS earlier than 2.1 will also work. Because JACKQUE directly interacts with the disk operating system, there can be no guarantee that the simulation will run on IBM *Compatible* machines.

C. THE PROGRAMMING LANGUAGE

JACKQUE is written in IBM Advanced Basic (BASICA). This language is provided with PC-DOS and possesses all of the graphics capabilities necessary for the simulation. This version of Basic is an interpreter language and runs slower than a compiled language.[2] However since the illusion of elapsed time is desired in this particular application, slower program run time is not a disadvantage.

JACKQUE also uses one sub-routine, called RNGEN, which is written in 8088 assembly language.

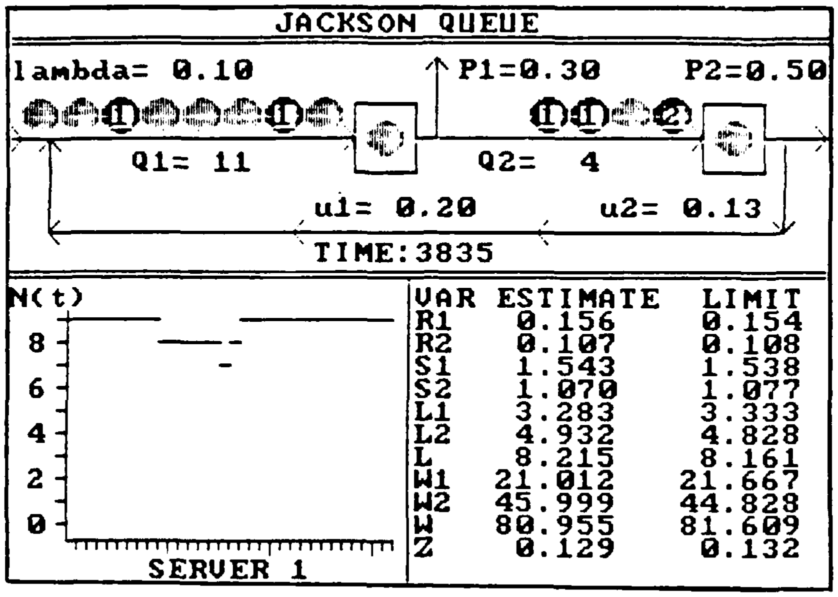
D. PROGRAM STRUCTURE

The program listing for JACKQUE is contained in Appendix A. The program, as it appears in the Appendix, has been extensively edited with comments to assist the interested reader in understanding the program. The program contained on the distribution diskette does not include the comments, since the inclusion of the comments exceeds the memory available within the Basic stack. Additionally, extensive use of subroutines were used in an effort to facilitate following the program flow. The program consists of five major modules. These modules are labeled Main Program Module (MPM), Clock Module (CM), Arrival Module (AM), Service Center 1 Module (SC1M), and Service Center 2 Module (SC2M). Any subroutine called exclusively by one of the five modules is physically located within that module. After the fifth module, all subroutines that are called from more than one module are listed, and labeled accordingly. All subroutines have been labeled with appropriate titles. After each subroutine title there is a slash (/), followed by the abbreviation of the module or modules from which it is called.

E. DESCRIPTION OF GRAPHICS DISPLAY

There are two configurations of graphics available to the user. When the simulation is first started, the graphics display screen will appear as seen in Figure 3.1. The upper half of the screen contains a pictorial representation of the status of the Jackson network for a particular point in time. A circle in a box represents a person being served at that service center. A circle in front of a box represents a person waiting at that service center. There may be more customers waiting than can be physically displayed on the screen. The number contained in the circle indicates the number of times that the particular customer has looped back to service center 1 from service center 2. The absence of a number indicates that the customer is just entering the network and has not yet experienced any feedback.

[2] There are compiled versions of Basic in existence



lambda= 0.10
P1=0.30
P2=0.50
Q1= 11
Q2= 4
u1= 0.20
u2= 0.13
TIME:3835

VAR	ESTIMATE	LIMIT
R1	0.156	0.154
R2	0.107	0.108
S1	1.543	1.538
S2	1.070	1.077
L1	3.283	3.333
L2	4.932	4.828
L	8.215	8.161
W1	21.012	21.667
W2	45.999	44.828
W	80.955	81.609
Z	0.129	0.132

Figure 3.1 Screen Display with Statistics Table.

In Figure 3.1, there are twelve customers waiting at service center 1 as indicated by Q_1 = 12, in addition to the one customer currently receiving service. At service center 2 there are four customers waiting for service in addition to the one customer receiving service. Of the customers visible, four have experienced feedback twice while one has experienced feedback once.

The bottom left corner of the screen contains the state-versus-time graph for service center 1. The system state values for the service center are displayed on the ordinate of the graph, and 32 increments representing time on the abscissa. The display shows that at time = 3818, the state of service center 1 jumped from state 8 to 7.

The table in the lower right corner contains the estimated and theoretical quantities discussed in Section II-D. The variable names appearing in this table are consistent with those defined in Chapter II.

Once the simulation is running, the user can change the configuration of the graphics screen to that shown in Figure 3.2. In this case, the state-versus-time graph for service center 2 is displayed in the lower right corner of the screen. If the state-versus-time graphs for both service centers were superimposed on each other, the resulting graph would be the *overall* system state-versus-time graph.

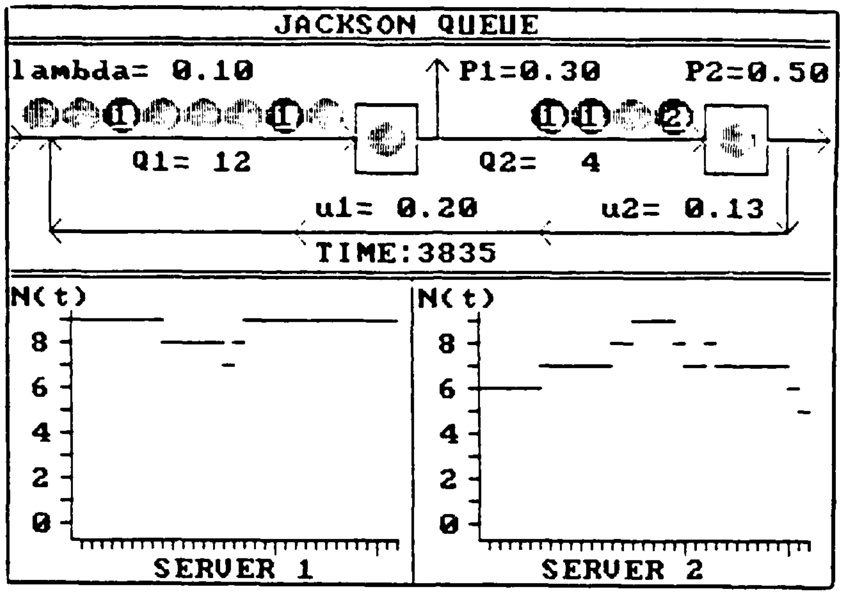


Figure 3.2 Screen Display with out Statistics Table.

A more complete description of the graphic displays, and user commands affecting them, are contained in Chapter IV, the user's manual.

F. PROGRAM DESCRIPTION

The main program module contains numerous subroutines that accomplish most of the routine housekeeping chores. These include:

- Checking to see if BASICA has been loaded from DOS, and if the machine is configured with a color graphics adapter.
- Printing the title screen.
- Loading user defined functions.
- Loading the random number generator sub-routine.
- Initializing and dimensioning variables.
- Printing the main program menu.
- Creating symbols that will be used latter in the graphic display.
- Generating the graphics display.
- Computing the theoretical values of the tabulated quantities.

The Program Menu presents the user with a number of options: (1) view the program instructions, (2) select a new seed for the random number generator, (3) start the simulation using the default parameters, (4) change the model parameters, and (5) end the program. The on-screen instructions provide a condensed version of the user instructions found in Chapter IV, the user's manual. The main program module also contains subroutines to perform the tasks selected from the program menu. The last task performed by the main program module is to transfer control of the program to the clock module.

The programming for some of the housekeeping and administrative tasks found in the main program module are taken in whole or in part from the program written by R.E. Nelsen [Ref. 2]. In particular the subroutine to check for BASICA and the graphics card, the subroutine to load the assembly language random number generator program, and the subroutine to change the seed of the random number generator were taken essentially without change.

Once a simulation run is initialized and running, the clock module controls the flow of the program for the remainder of the simulation run. The first function of the clock module is to generate an initial exogenous customer arrival time and determine if that customer will exit after completing service at service centers 1 and 2. This is a one time initialization.

The clock module determines the next-event time as being the smallest of the next arrival time, next service completion time at service center 2, or next service completion time at service center 2. The next-event time thus determined is always rounded up to an integer value so to be compatible with the fixed increment simulation clock.

Every time the simulation clock advances one time increment, the clock module updates the state-versus-time graphs for those service centers selected to be displayed by the user. Recall that the state of a particular service center is the total number waiting and being served by that service center at a particular point in time. When the elapsed time reaches 32 time units, the plotted lines on these graphs will be at the right most edge of the graphs. When this condition occurs, the clock module shifts the displayed state graphs one time unit to the left, and then plots the new time increment. This shifting of the graphs by one time unit to the left continues for all successive time units. In this manner the service center state-versus-time graphs always contain the most recent 32 time units of information, while more dated information is permanently lost. When the simulation clock eventually equals the next-event time, the clock module shifts control to either the arrival module, service center one module, or service center 2 module as appropriate.

Every time the clock module increments the simulation clock, it also checks the keyboard for certain commands issued by the user. By striking the appropriate keyboard keys, the user can suppress the audible tones associated with movement in the network, pause the simulation, continue the simulation, change the display configuration, speed up the simulation, and end the simulation. These user commands are described in detail in Chapter IV, the user's manual.

There are only five possible events that can happen in the simulation:

1. An exogenous customer can enter the network.
2. A customer can complete service at service center 1 and exit the network.
3. A customer can complete service at service center 1 and proceed to service center 2.
4. A customer can complete service at service center 2 and exit the network.
5. A customer can complete service at service center 2 and loop back to service center 1.

Event 1 is handled by the arrival module. Events 2 and 3 are handled by the service center 1 module. Events 4 and 5 are handled by the service center 2 module.

The arrival module is called when an exogenous customer enters the network. It flashes the arrow leading into service center 1. If no one is presently in service, a *blue* ball representing the customer is placed in the square box corresponding to service center 1. If there is a customer already in service but the waiting queue has less than eight people in it, a blue ball is placed at the end of the queue and the queue counter, Q_1, is incremented by one. Otherwise only Q_1 is incremented.

If the exogenous customer immediately receives service, rather than entering the queue, the arrival module calls the random number generator and computes the length of service. In all cases the arrival module calls the random number generator and computes the inter-arrival time of the next exogenous customer. The inter-arrival time, and service time if applicable, are added to the current time (continuous not rounded time), to determine when the service will be completed and when the next exogenous arrival will take place. When the program returns to the clock module, these newly created event times will be compared to the service completion time at service center 2 to determine what event should occur next in the simulation.

The service center 1 module is called when the next scheduled event is a service completion at service center 1. The departing customer can either depart the network or continue on to service center 2. In either case, the module erases the departing ball, shifts any customers waiting for service center 1 to the right, and decrements Q_1 by one as appropriate. If there is now a customer receiving service at service center 1, the random number generator is called and a new service completion time is generated.

If the customer departs the system, the appropriate arrow is flashed. If the customer proceeds to service center 2 the arrow leading from service center 1 to service center 2 is flashed. If no one is presently in service at service center 2, the transferred ball will be placed in the square box representing service center 2. If there is a customer already in service, but the waiting queue in front of service center 2 has less than five people in it, the ball is placed at the end of the queue, and the queue counter Q_2, is incremented by one. Otherwise, only Q_2 will be incremented by one.

If the transferred customer immediately receives service at service center 2, rather than exiting or entering the queue in front of service center 2, the service center 2 module calls the random number generator and determines a service completion time for that customer. The service completion time is added to the current time to determine when the service will be completed. When the program returns to the clock module, this newly created event-time will be compared to the next exogenous arrival time and service completion time at service center 2, to determine what event should occur next in the simulation. Before returning to the clock module, the service center 1 module calls the random number generator and determines whether or not the next customer leaving service center 1 exits the network or proceeds on to service center 2.

The service center 2 module is called when the next scheduled event, as determined by the clock module, is a service completion at service center 2. The departing customer can either exit the network or loop back to service center 1. In either case the service center 2 module erases the departing ball, shifts any customers waiting for service center 2 to the right, and decrements Q_2 as appropriate. If there is now a customer receiving service at service center 2, the random number generator is called and a new service completion time is generated.

If the customer departs the system, the service center 2 module flashes the appropriate arrow. If the customer loops back to service center 1, the bottom arrow from service center two to service center one is flashed. If no customer is presently in service at service center 1, the transferred ball is placed in the square box representing service center 1. If there is a customer already in service, but the waiting queue has less than 8 people in it, the transferred ball is placed at the end of the queue, and the queue counter, Q_1, is incremented by one. Otherwise, only Q_1 is incremented.

If the transfered customer immediately receives service at service center 1, rather than exiting or entering the queue in front of service center 1, the service center 2 module calls the random number generator and determines a service completion time for that customer. The service completion time is added to the current time to determine when the service will be completed. When the program returns to the clock module, this newly created event time will be compared to the next exogenous arrival time and service completion time at service center 1, to determine what event should occur next. Before returning to the clock module, the service center 2 module calls the random number generator and determines whether or not the next customer leaving service center 2 exits the network or loops back to service center 1.

The service center 2 module performs one other important function. It keeps track of how many times a customer loops back to service center 1, allowing this number to be physically placed inside the balls representing customers. When a customer first arrives and has had no opportunity to loop from service center 2 back to service center 1, he is depicted as a solid *blue* ball with no number printed on it. If a customer has looped from service center 2 to service center 1 exactly once, he will be depicted as a *purple* ball with a number "1" printed in the center of it. If a customer has looped exactly twice, he will be depicted as a *purple* ball with a number "2" printed in the center of it, etc.

All of the event modules create audible sounds that correspond to customer motion in the network. There are only two distinct tones. A short high pitched tone to indicate a departure from the network, and a longer low pitched tone to indicate arrivals from both outside the network and arrivals from another service center.

When a ball arrives at a service center and finds the *visible* queue space at capacity (8 for service center 1 and 5 for service center 2), the modules store these customers in a queue that is not visible to the user. These *invisible* queues, one for each service center, preserve the order and identity of the *overflow* customers. They are *overflow* only in the sense that there is not enough room on the computer screen to graphically display them all. If the queue counters (Q_1, Q_2) do not agree with what is physically shown on the screen, it is because there are customers in the *invisible* queue or queues. The queue counters are correct.

After returning to the clock module, the simulation updates the estimates of the tabulated performance values, using information generated in the event modules, and prints them to the screen. As previously noted, the clock module also updates the service center state-versus-time graphs. Each of the event modules records the information necessary for the clock module to know where vertically on the graphs it should draw the horizontal line segments representing one time unit. These graphs, because of space constraints, can not plot state variables greater than nine. If the state variable is equal to 9, the clock module plots a *purple* tick mark at a vertical position corresponding to 9. For state variables greater than 9, the clock module plots a *blue* tick mark also at a vertical position corresponding to 9. Therefore, observing a *blue* tick mark only indicates that the service center state is 10 or greater.

G. PROGRAMMING CONSIDERATIONS

At all times during the simulation, JACKQUE maintains a next-event time for each of the three possible events (exogenous arrivals, and service completion at service centers 1 or 2). The clock module than takes the minimum of these three times to determine the next event. Because of the structure of the network, there are times when an event should never be allowed to occur.

For example, if no customer is currently being served at a service center, the program must be prevented from selecting the service completion time of that service center as the minimum of the next-event times. If the program did select the service completion time as the next-event, the simulation would attempt to move a customer from the service center when in fact there is no customer there.

To prevent the selection of an impossible event, the program sets the next-event time of all impossible events equal to 10^{31}, which for all practical purposes is equal to infinity. The clock module takes the minimum of the next-event times. It will never select 10^{31}, because the next exogenous arrival time will always be less than this. In this particular model, one or both of the next service times may periodically be set to infinity.

When the clock module selects the minimum of the three next-event times, there is a remote possibility that there will be a two-way, or even three-way tie. In the unlikely event that a tie does occur, the program decides randomly which of the events should occur first.

Because of the discrete nature of the graphics display, events can only be depicted as having occurred at the end of a discrete time tick. For this reason, within one time tick, Δt, many events may actually occur. Some of these events may occur at the very begining of Δt, some in in the middle, and perhaps some near the end of Δt.

Although their chronological order of occurrence is preserved, they are all plotted sequentially as if they occurred at the end of Δt. For this reason what is observed on the screen is not an absolutely accurate representation of a Jackson network with exponentially distributed inter-arrival times and service times. What is graphically displayed is a Jackson network with geometrically distributed inter-arrival and service times, the geometric distribution being the discrete analog of the continuous exponential distribution. Despite this fact, the simulation still produces a realistic visual impression of a true Jackson network.

For a discrete event stochastic process, the state of the process can only change by discrete jumps of plus or minus one. However, because JACKQUE considers all events occurring in Δt to have occurred at the end of Δt, the state-versus-time graphs might exhibit jumps greater than plus or minus one. For example if service center 1 experiences 3 arrivals and 1 service completion, all during Δt, the state-versus time graph for service center 1 will jump 2 units when the horizontal line segment corresponding to Δt is plotted (see Figure 3.3). Similarly, the state-versus-time graph might plot the next tick mark at the same vertical height as the previously plotted tick mark, indicating no change when in fact there had been a change during Δt. For example, if service center 1 experiences one arrival and one service completion during Δt, there is no net change and the time tick will be plotted at the same vertical height as before (see Figure 3.4).

This undesireable effect can be minimized by reducing the probability of having multiple events occur during Δt. This can be accomplished by either reducing the size of Δt or through the judicious selection of the network parameters, as discussed in Section IV-I.

H. GENERATION AND USE OF RANDOM NUMBERS.

As demonstrated by Nelsen [Ref. 2], the random number generator residing in BASICA produces pseudo-uniform (0,1) numbers with significant lag-1 correlation. This makes it a poor random number generator and unsuitable for use in simulation.

JACKQUE uses a random number generator modified by Nelsen, and originally programmed by B.O. Shubert of the Naval Postgraduate School for use on the IBM-PC. This random number generator, named RNGEN, is written in 8088 assembly language.

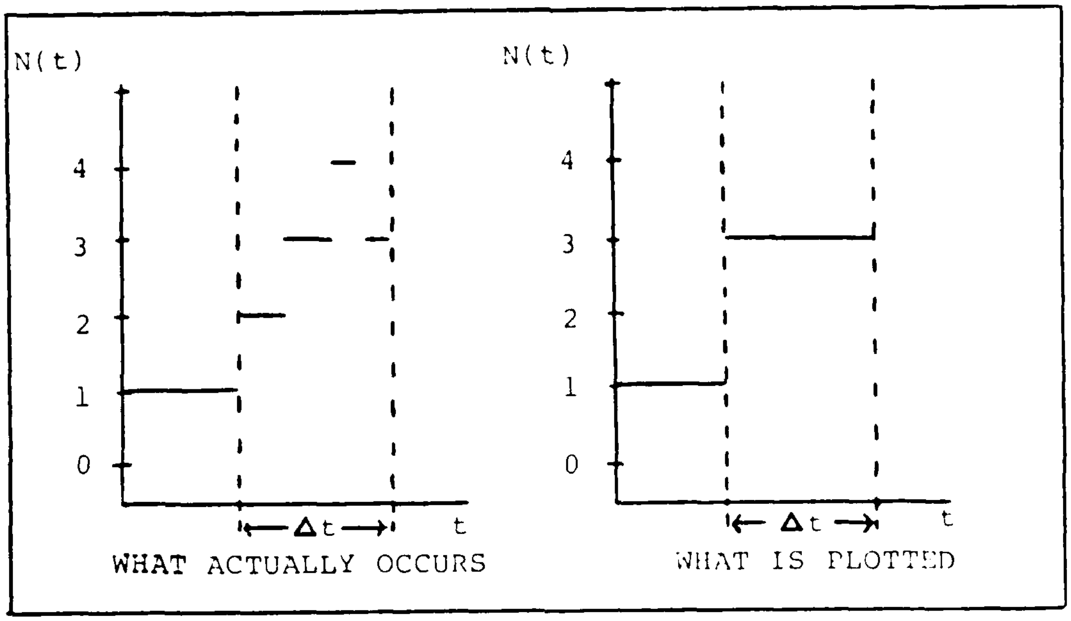


Figure 3.3 An Example of Jumping 2 States in Δt.

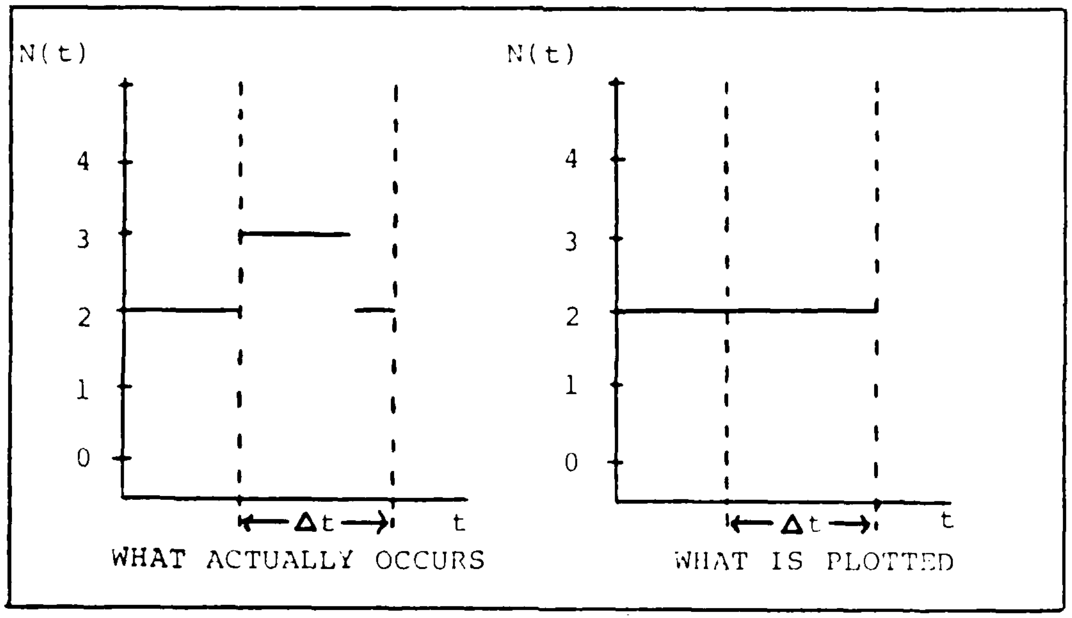


Figure 3.4 An Example of Jumping 0 States in Δt.

RNGEN is a Prime Multiplicative Linear Congruential Generator (PMMLCG) based on the following recursive formula:

$$X_{i+1} = 7^5 \times \text{Mod}(2^{31}-1), \qquad \text{(eqn 3.1)}$$

where X_i is the old seed and X_{i+1} is the new seed. A uniform (0,1) random number is obtained by dividing the new seed by the modulus. This particular PMMLCG is considered to exhibit good statistical behavior and is currently used in the LLRANDOM package, IBM's GGL, and IMSL's GGUBS [Ref. 6: p.226].

JACKQUE uses Uniform (0,1) random variables for three different purposes:

1. To generate exponentially distributed random variables.
2. To determine whether a customer exits or remains in the network after completing service at a service center.
3. To break ties in the event two or more event times are minimum.

The inter-arrival times and service times of customers in a Jackson network are assumed to be exponentially distributed. To obtain these exponential random variables, Uniform (0,1) random variables produced by RNGEN are transformed through the inverse probability method. Let U be a uniformly distributed (0,1) random variable, and let X be an exponentially distributed random number with parameter λ. The distribution function for X is:

$$F(x) = 1 - e^{-\lambda x}, x \geq 0. \qquad \text{(eqn 3.2)}$$

An exponential random number is generated by setting u = F(x) and solving for x:

$$u = 1 - e^{-\lambda x} \qquad \text{(eqn 3.3)}$$

$$1-u = e^{-\lambda x} \qquad \text{(eqn 3.4)}$$

$$\ln(1-u) = -\lambda x \qquad \text{(eqn 3.5)}$$

$$x = -1/\lambda \ln(1-u) \qquad \text{(eqn 3.6)}$$

$$x = -1/\lambda \ln(u). \qquad \text{(eqn 3.7)}$$

It is possible to replace (1-u) with u because both (1-u) and u have the same Uniform (0,1) distribution.

When a customer completes service at a service center, he can either exit the system or go to the next service center. To determine which occurs JACKQUE compares a Uniform (0,1) random number to the fixed probability that the customer exits after completing service at service center i (P_i). If $U(0,1) \leq P_i$, the customer is considered to have exited. Otherwise the customer is considered to have proceeded to the next service center.

A similar procedure is used to break a tie in the event two or three event-times are simultaneously minimum. In the case of a two-way tie, event A is chosen if $U(0,1) \leq 1/2$, and event B is chosen otherwise. In the case of a three-way tie, event A is chosen if $0 \leq U(0,1) < 1/3$, event B is chosen if $1/3 \leq U(0,1) < 2/3$, and event C is chosen otherwise. Because all events do not have equal probabilities of occurring, using equal probabilities is not totally accurate, but considered sufficient for the purposes of this simulation.

IV. THE JACKSON NETWORK USER'S GUIDE

Author
Gary F. Greene

James D. Esary
Thesis Advisor

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

A. INTRODUCTION

JACKQUE is a discrete event stochastic simulation of a Jackson network. A Jackson network is a network of queues where a customer may receive service from numerous service centers, possibly visiting the same service center more than once, before exiting the network. The program produces a variety of visual displays on the computer screen intended to provide the user with a better understanding of the dynamic nature of the model.

This manual is intended to assist the user in the actual use and operation of the program. In describing the user interaction with the program, the following notation is used:

- When a command is to be entered (typed) exactly as shown, it will be enclosed in single quotes (' ').
- When a specific key on the keyboard is to be pressed, the letter or letters on the key will be encased by brackets (< >).

For example, 'Load "JACKQUE.BAS",r' <Enter>, means type the command exactly as it appears between the *single* quotes and then press the ENTER key. PC-DOS and JACKQUE make no distinction between upper and lower case letters, so either may be used.

B. MODEL DESCRIPTION

The Jackson network modeled by this simulation consists of two service centers, each containing one server. An arriving customer from outside the network must first go to service center 1 where he enters into a First in First Out (FIFO) queue. Upon completion of service at service center 1, the customer either exits the network with probability P_1, or proceeds on to service center 2 with probability $1-P_1$, where he enters another FIFO queue. Upon completion of service at service center 2, the customer either exits the network with probability P_2, or loops back to service center 2 with probability $1-P_2$, again entering a FIFO queue. In this model, a customer can loop from service center 2 to service center 1 a large but finite number of times.

There are several assumptions required for this model. Customers from outside the network are assumed to arrive at service center 1 according to a Poisson process with rate λ. The service times at service center 1 and service center 2 are assumed exponentially distributed with rates μ_1 and μ_2 respectively. When a customer departs the network with probability P, or goes to the next service center with probability 1-P, he does so instantaneously. These probabilities are assumed independent of the history of the network. It is assumed that there is unlimited waiting space at both service centers, and that if one server is idle he does not help a busy server at the other service center.

For this model, the state of the system, $N(t)$, is the sum of $N_1(t)$ and $N_2(t)$. $N_i(t)$ $(i=1,2)$ is defined as the number of customers waiting and being served at service center i at time t. Therefore $N(t) = N_1(t) + N_2(t) = n$, where $n = 0,1,2,\ldots$.

C. HARDWARE AND SOFTWARE REQUIREMENTS

1. *Hardware*

An IBM Personal Computer (IBM-PC) with at least 128 Kilo-bytes of random access memory (RAM), a color/graphics adapter card, a color monitor, and at least one 5 1/4 inch floppy disk drive are required to run the simulation. The color monitor and color/graphics adapter are essential since the program, JACKQUE , uses color graphics. The 128 K of memory is required because JACKQUE, which is written in BASICA, uses an assembly language sub-routine which is loaded outside of Basic's 64 Kilo-byte workspace.

2. *Software*

The primary program JACKQUE.BAS, and the assembly language sub-routine RNGEN.SRT are provided to the user on the *distribution diskette.* In addition to these programs, the user must provide the Disk Operating System PC-DOS (2.1) and Advanced Basic (BASICA) in order to run the simulation. Because JACKQUE directly interacts with the Disk Operating System, there can be no guarantee that the simulation will run on *IBM compatible* machines.

3. *Program Files*

The following program files are included on the distribution diskette:

- JACKQUE.BAS - the main program written in BASICA.
- RNGEN.SRT - a random number generator written in 8088 assembly language, used as a sub-routine to the main program.
- JACKQUE.BAT - a batch file that loads and starts the simulation.
- AUTOEXEC.BAT - an autoexec batch file that loads and starts the simulation immediately after the system is started or restarted.

The first two files are absolutely necessary to run the simulation. The two batch files are two different methods of automatically loading and starting the simulation. The batch files are not essential, but enhance user-friendliness. A source code listing of RNGEN.SRT is contained in Reference 2.

D. GETTING STARTED

1. *Making an Application Diskette*

The *application diskette* is created using the *distribution diskette*. After being used for this purpose, the *distribution diskette* should be stored in a safe place, and only used to create additional *application diskettes* as needed.

Although there are many different ways that a user can create an *application diskette*, this manual will describe just one. A self-starting (bootable) diskette containing certain operating files, the BASICA command file and the simulation program files is considered the most efficient format. This eliminates the requirement for a separate diskette containing PC-DOS, and the diskette can be run using only one disk drive.

To make a bootable diskette, follow the instructions in the PC-DOS manual and format a blank diskette using the FORMAT command with the /s option. The /s option copies only the DOS files that must be on the newly formatted disk. These are the COMMAND.COM file, plus several hidden files that are necessary for DOS to operate. The Advanced Basic file, called BASICA.COM, also located on the DOS diskette can be transferred onto the *application diskette* using the copy command. The copy command is also used to transfer the simulation's four files from the *distribution diskette* onto the bootable *application diskette*. The user now has a totally self contained, self starting diskette capable of running the simulation.

2. *Starting the Simulation*

If the PC is turned off, place the bootable diskette into drive A and turn on the computer.[3] After the computer does it's internal checks taking nearly 60 seconds, the computer will immediately execute the DOS commands contained in the AUTOEXEC batch file. These DOS commands automatically load BASICA and JACKQUE.BAS, and run the simulation.

If the PC is already turned on and in the command level of DOS, place the bootable diskette in drive A and type 'JACKQUE'<ENTER>. The computer will immediately execute the DOS commands contained in the JACKQUE.BAT file and start the simulation. Alternately, with the bootable diskette in drive A, the computer can be restarted by simultaneously striking <Ctrl> <alt> <del>. In this case the simulation is started using the AUTOEXEC.BAT file.

[3]This section assumes that drive A is the default drive.

If for some reason the computer is in the command level of DOS and it is not desired to use the batch files, the simulation can be started by using the following sequence of commands:

- 'BASICA' <RETURN>
- 'Load "d:JACKQUE.BAS",r' <ENTER> where d is the label of the drive where JACKQUE.BAS can be found.

If the simulation has been terminated from the program menu, and the computer is still in the BASICA environment, another simulation run can be initiated by simply typing 'RUN' <ENTER>. If you have returned to the command level of DOS, use the methods previously described.

During the execution of the program, JACKQUE attempts to load the random number generator sub-routine from the diskette in the A disk drive. If the program fails to find the necessary file in drive A, it asks the user for the correct drive label where it can be found. Simply respond by entering the proper drive label and <ENTER>. If it can not find the sub-routine on the specified drive, or if the specified drive is not valid, the program will again ask the user for the correct drive label. This situation will never arise if the bootable application diskette is used in drive A.

3. *Title Screen*

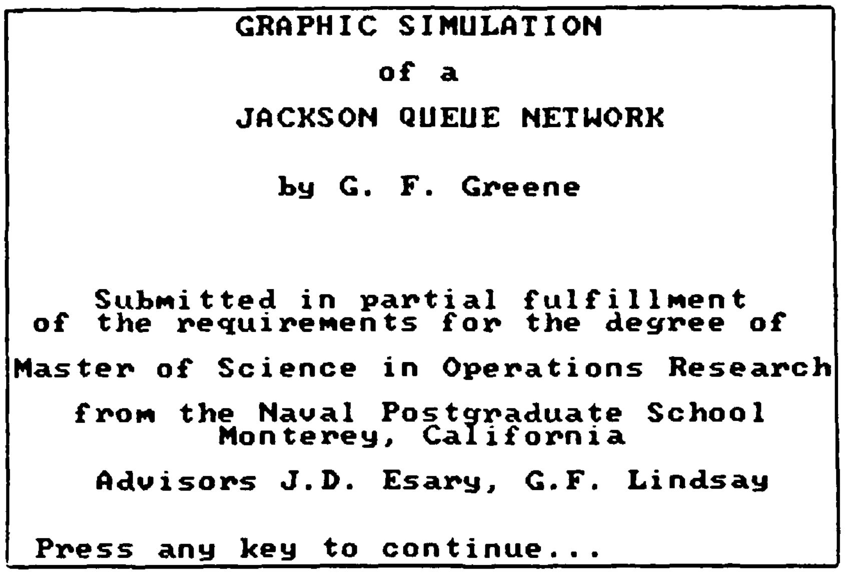
Upon starting the simulation, the first thing the user will observe on the screen will be the title screen as shown in Figure 4.1. From this point on, the on-screen prompts will guide the user through the running of the simulation.

E. PROGRAM MENU

Upon leaving the title screen, there will be a few second delay while JACKQUE initializes variables and loads the random number generator sub-routine. The next screen to be displayed will be the program menu as shown in Figure 4.2. The options on the menu allow the user to read the instructions, select a new starting seed for the random number generator, run the simulation using the default parameters, run the simulation using parameters selected by the user, and end the program.

When a simulation is ended, JACKQUE always returns to the program menu. This allows the user to conduct numerous simulations runs, perhaps with different seeds and/or parameters values, during one running of JACKQUE.

Upon departing the menu, and before running the first simulation, JACKQUE quickly plots and then erases the screen shown in Figure 4.3. Before being erased,


GRAPHIC SIMULATION

of a

JACKSON QUEUE NETWORK

by G. F. Greene

Submitted in partial fulfillment
of the requirements for the degree of

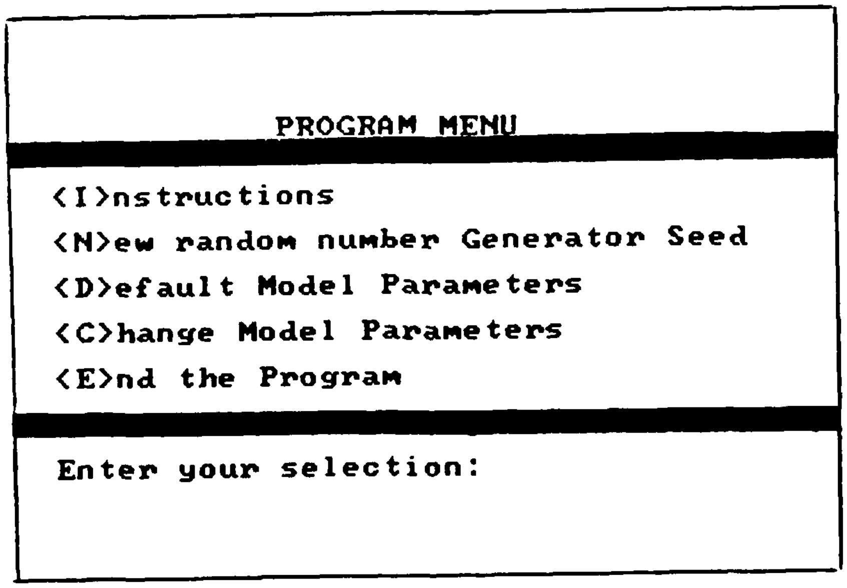
Master of Science in Operations Research

from the Naval Postgraduate School
Monterey, California

Advisors J.D. Esary, G.F. Lindsay

Press any key to continue...

Figure 4.1 The Title Screen.


PROGRAM MENU

<I>nstructions

<N>ew random number Generator Seed

<D>efault Model Parameters

<C>hange Model Parameters

<E>nd the Program

Enter your selection:

Figure 4.2 The Program Menu.

these symbols are stored in memory and are used by JACKQUE in generating graphics for all future simulation runs.

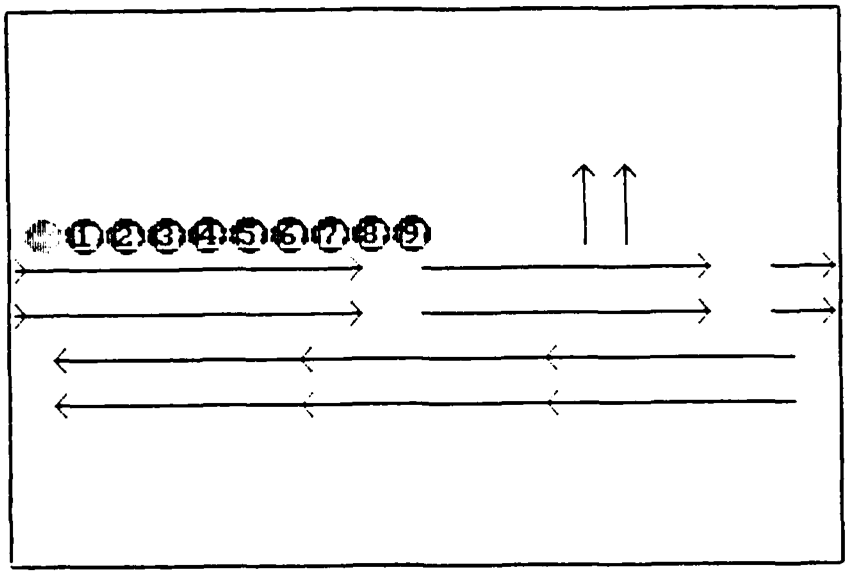


Figure 4.3 Graphic Symbols.

1. *Displaying the User Instructions*

To display the user instructions on the screen, press < I >. The instructions displayed are a condensed version of what is contained in this guide.

2. *Setting the Random Number Generator Seed*

A new seed can be selected by pressing < N >. The program will prompt you to enter the value of the seed. The seed is entered by selecting a number in the range from 1 to 2,147,483,646 and pressing < ENTER >. If a non-integer number is entered, it will be truncated to an integer.

If no seed is selected, a default starting seed is used. If the simulation is stopped, and then a new simulation started without leaving JACKQUE and without setting a new seed, the generator will continue as if the old simulation had never stopped. That is, the first random number generated in the new simulation is the same random number that *would* have been generated *next* in the old simulation.

By using the same seed with the same model parameters, the simulation can be run numerous times, obtaining the exact same results each time. This may be useful for replicating a particular model behavior.

3. *Default Model Parameters*

If this option is selected by pressing < D > , the simulation immediately starts with the model parameters set to the following default values:

- $\lambda = .10$
- $\mu_1 = .20$
- $\mu_2 = .13$
- $P_1 = .30$
- $P_2 = .5$

If a new seed is desired it must be entered *before* striking < D > .

4. *Changing the Model Parameters*

When < C > is pressed, the user is prompted for values of the arrival rate, service rate at service center 1, service rate at service center 2, probability a customer exits after departing service center 1, and probability a customer exits after departing service center 2. The rates can be any positive, non-zero real number. The probabilities must be between zero and one. The quantities are entered by typing in the appropriate value and then pressing < ENTER > in response to each of the five questions. If < ENTER > is struck without typing in a number, the parameter value will be considered to be zero.

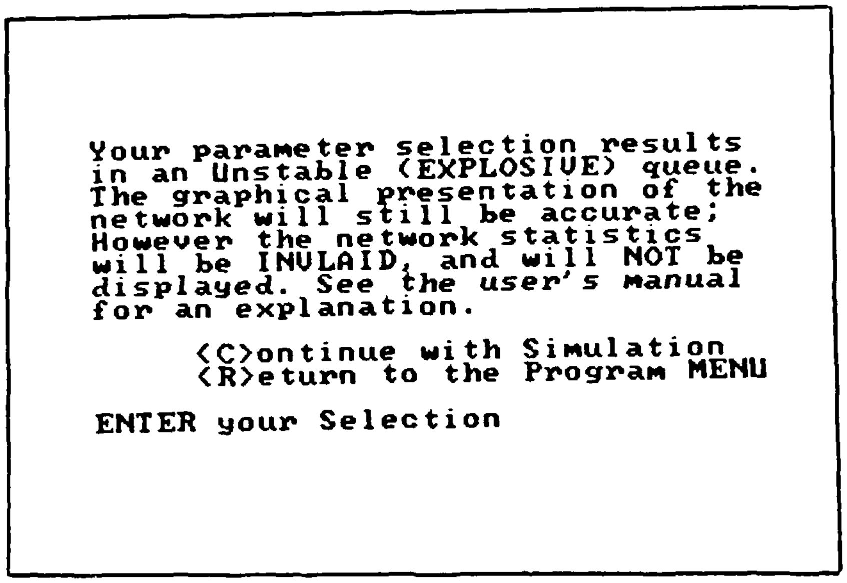
For a Jackson network to be stable, the effective arrival rate at each service center, taking into account arrivals from both outside the network and from other service centers, must be less than the corresponding service rates. If this is not the case the network will become *unstable* in that more customers are joining the queue than departing it.

In the case of an *unstable* queue, there are no steady state probabilities and the network performance measures have no meaning. The stability of the network is controlled by the interactions of the five model parameters. Setting P_1 and P_2 simultaneously equal to 0 will always result in an unstable network. Section III-I discusses the selection of model parameters. If the selection of parameters results in an unstable network, the screen shown in Figure 4.4 will be displayed.

The simulation starts immediately after entering the last requested parameter value. If a new seed is desired, it must be entered *before* striking < C > .

5. *Ending the Program*

To end the program, press < E > . This will clear the screen and leave the PC in the BASIC command mode. It also returns the screen to the standard monochrome



```
Your parameter selection results
in an Unstable (EXPLOSIVE) queue.
The graphical presentation of the
network will still be accurate;
However the network statistics
will be INVALID, and will NOT be
displayed. See the user's manual
for an explanation.

      <C>ontinue with Simulation
      <R>eturn to the Program MENU

ENTER your Selection
```

Figure 4.4 Example of Unstable Parameter Warning.

mode with column length equal to 80. If it is desired to return to the DOS command level type 'system' <ENTER>.

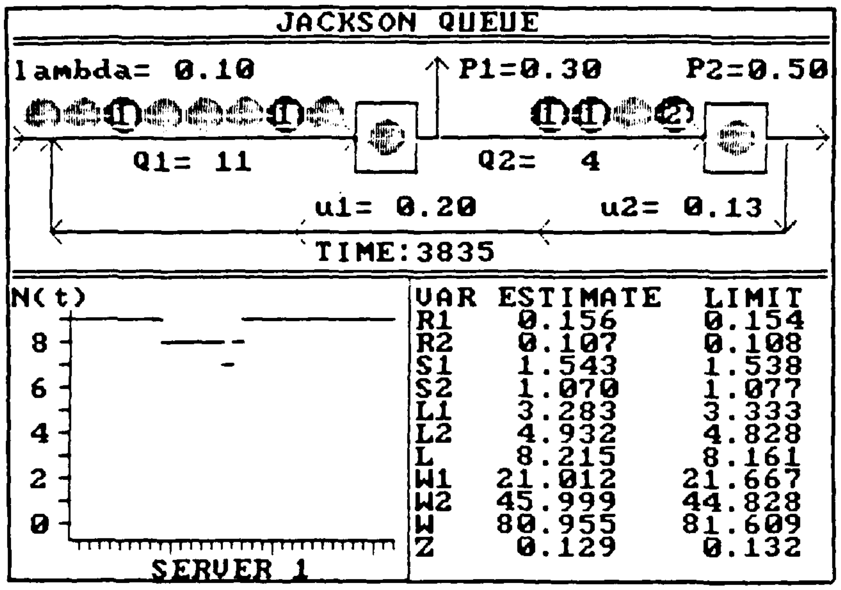
F. THE GRAPHICS DISPLAY

JACKQUE uses various graphical displays to allow the user to observe the flow of customers through the network. When the simulation is first started, the graphics display will appear as shown in Figure 4.5. The graphics display is divided into three regions: a pictorial display of the network, the state-versus-time graph for service center 1, and a table of network performance measures.

At any time after the start of the simulation, the user can chose to change the configuration of the graphics display to that shown in Figure 4.6. The table of performance measures has been replaced by the state-versus-time graph for service center 2. The user can freely change the configuration back and forth between the two modes as described in Section IV-G. The following descriptions are directed at the display mode shown in Figure 4.5.

1. *The Pictorial Network Display*

The display in the upper half of the screen is a pictorial representation of the Jackson network at a particular instant in time. The left square box represents service center 1. The right square box represents service center 2. The vertical and horizontal



VAR	ESTIMATE	LIMIT
R1	0.156	0.154
R2	0.107	0.108
S1	1.543	1.538
S2	1.070	1.077
L1	3.283	3.333
L2	4.932	4.828
L	8.215	8.161
W1	21.012	21.667
W2	45.999	44.828
W	80.955	81.609
Z	0.129	0.132

Figure 4.5 Screen Display with One State Diagram.

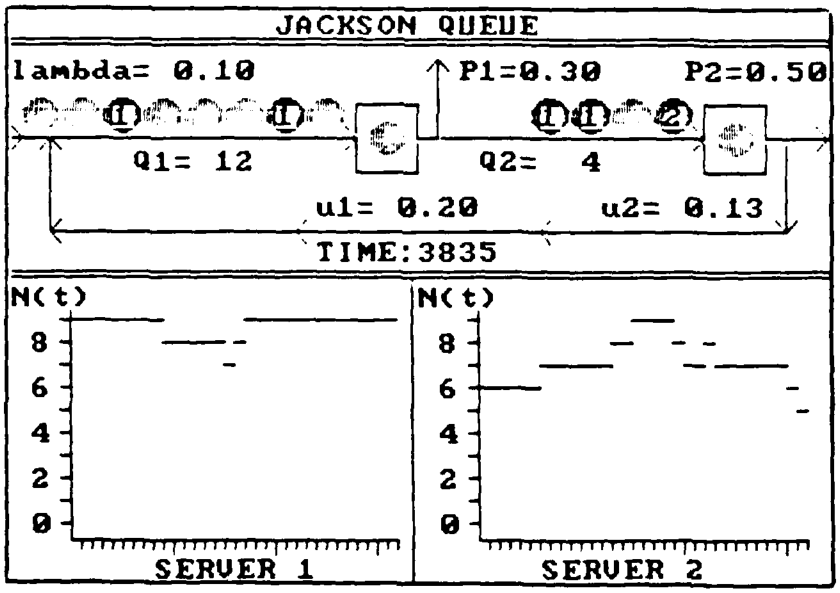


Figure 4.6 Screen Display with Two State Diagrams.

lines connecting these two boxes represent the possible paths that a customer can follow through the network. An arrow directed into a service center is the path of an arriving customer. An arrow directed out of a service center whose head is unconnected, is the path of a customer departing the network. Arrows connecting the two service centers are the paths of customers moving from one service center to the other. The lowest horizontal line directed from right to left is the *feedback* loop from service center 2 to service center 1.

Customers are represented as colored solid balls. A customer arriving from outside the network for the first time is represented as a solid *blue* ball. If a customer loops from service center 2 to service center 1, the color of the ball is changed to *purple*, and a number "1" is placed inside the ball. Each time this customer loops back from service center 2 to service center 1 the number inside the *purple* ball is incremented by one, up to a maximum of nine. Thus a *purple* ball with a number "5" in it, represents a customer who has been fed back from service center 2 to service center 1 five times. When a customer exits the network, the corresponding ball disappears from the screen.

A customer actually receiving service from a service center is placed inside the appropriate square box. If a customer is *waiting* for service at a service center, he is placed in the waiting line immediately to the left of the appropriate service center (square box). The counters under the two waiting lines, Q_1 and Q_2, display the number of customers waiting for the respective service centers. These counters do not include the person actually receiving service. Because of the limited size of the computer screen, it is possible for the values of Q_1 and Q_2 to be greater than the number of customers actually shown waiting as balls. These *invisible* waiting customers are still accounted for, and will enter the *visible* waiting line when space permits.

The five model parameters, λ, μ_1, μ_2, P_1, and P_2 are displayed in appropriate places on the pictorial network display. The current time of the system is printed at the bottom of this display.

Each time a customer (ball) moves in the network, the motion is represented by moving the affected balls, *flashing* the appropriate path arrow, and sounding a distinctive tone. There are only two distinctive tones. A short high pitched tone to indicate a departure from the network, and a long low pitched tone to indicate arrivals. Arrivals consists of both arrivals from outside the network and arrivals from other service centers.

In Figure 4.5, both service centers are busy with four customers waiting for service center 2 ($Q_2 = 4$), and twelve customers waiting for service center 1 ($Q_1 = 12$)

although only eight are physically shown. Of the visible customers, one has experienced feedback twice, four have experienced feedback once, and nine are passing through the network for the first time.

2. *The State-Versus-Time Graph*

The lower left corner of the graphics display contains the state-versus-time graph for service center 1. The values of the state variable N(t), the number of customers waiting and being served at service center 1 at time t, are plotted on the ordinate. Time is plotted on the abscissa, and is represented as 32 discrete divisions. Each of these time segments corresponds to one time tick of the simulation clock.

After each increment of the simulation clock, the program determines the current state of service center 1. The program then uses this information to plot a horizontal line one time tick in length at the appropriate height on the state-versus-time graph. In Figure 4.5, at time 3818, the state of service center 1 went from 8 to 7.

When the elapsed time reaches 32 time units, the plotted line on this graph will be at the right most edge of the graph. When this condition occurs, the program shifts the entire graph one time unit to the left, and plots the new time increment at the appropriate height. In this manner the graph always contains the most recent 32 time units of information on the state of service center 1. An identical description pertains to the state-versus-time graph for service center 2, which can be displayed at the users option.

The state-versus-time graphs, because of space constraints, can not plot state variables greater than 9. If the state variable is equal to 9, a *purple* tick mark at a vertical position corresponding to 9 is plotted. For state variables with values greater than nine, a *blue* tick mark is plotted, also at a vertical position corresponding to 9. Therefore observing a *blue* tick mark only indicates that the service center state is 10 or greater.

3. *Network Performance Measures*

The lower right hand corner of the graphics display contains a table of estimated and theoretical equilibrium quantities. These quantities are just a few of the many that can be used to evaluate the performance characteristics of the network. The variable names used in this table are defined as:

R_i = the effective arrival rate of customers at service center i ($i = 1,2$).

S_i = the expected number of times a customer will pass through service center i before exiting the network (i = 1,2).

L_i = the expected number of customers both waiting and being served at service center i (i = 1,2).

L = the expected number of customers in the network, both waiting and being served.

W_i = the expected amount of time a customer spends waiting and being served at service center i (i = 1,2).

W = the total expected amount of time a customer spends in the network, both waiting and being served.

Z = the proportion of time service center 1 is busy and service center 2 is idle.

The quantities that appear under the column header *Limit*, are the limiting values of these quantities. They are computed using results from Little's Law and from the analysis of simple M/M/1 queues. They are the theoretical values that would be obtained if the network reached steady state.

The quantities that appear under the column header *Estimate* are estimates of these theoretical limiting values. They are computed with data collected from the simulation. These estimates are updated after every event occurrence, where events are defined to be the arrival of a customer from outside the network and service completions at either service center.

At certain points during the simulation, some of the estimates might be negative and/or extremely small or large in magnitude. If an estimate exceeds the format capabilities of JACKQUE by being too large or small, the estimate will be displayed on the screen as "9999.99". Some of the estimates will temporarily become negative if the estimated effective arrival rate at a service center is greater than the estimated service rate of the service center. To avoid confusion, the absolute value of a negative estimate will be displayed on the screen.

Because of the manner in which L_1, L_2, W_1, W_2, and Z are estimated, the estimates might show considerable variance around the theoretical value. This variability will decrease as the simulation progresses in time.

If in the selection of model parameters, an effective arrival rate is greater than the service rate at the applicable service center, the user is given the option to terminate the current simulation or to continue as described in Section IV-E-4. If the user opts to continue with the present model parameters, the resulting theoretical and estimated values will no longer be valid. In this case the graphics display will appear as seen in Figure 4.7.

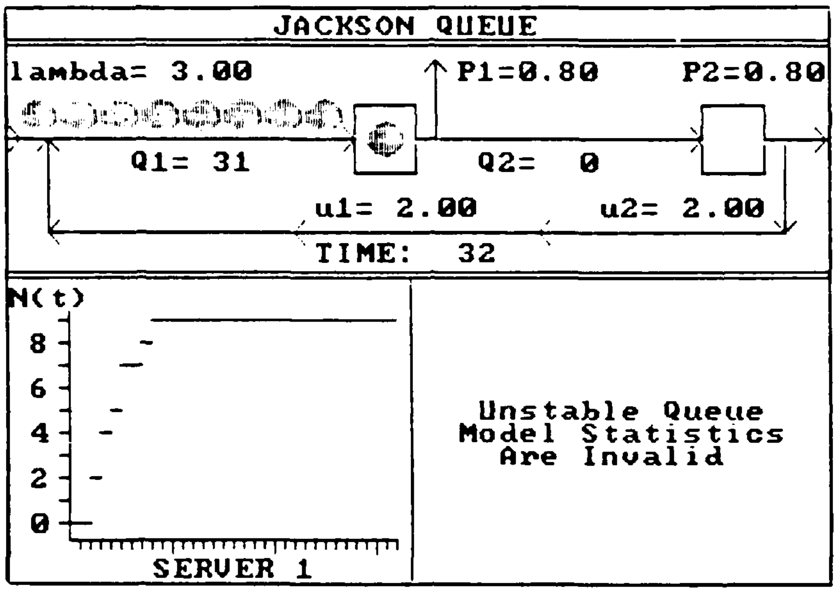


Figure 4.7 Example of Graphics Screen for Unstable Network.

G. KEYBOARD COMMANDS

While the simulation is running, the user has the ability to interact with the model through the use of the following keyboard commands:

- < P > - pressing P for "Pause" will temporarily freeze the simulation, allowing the user time to study the graphical display.
- < C > - pressing C for "Continue" counteracts pressing < P > and resumes the simulation where it left off.
- < B > - pressing B for "Beep" will turn off the audible tones that accompany customer motion through the network. Pressing < B > a second time will turn the beeps back on.
- < D > - pressing D for "Display" replaces the table of estimated quantities with the state-versus-time diagram for service center 2. Pressing < D > a second time returns the graphics screen to the original configuration.
- < F > - pressing F for "Fast", accelerates the simulation, causing the estimated values to converge to the theoretical values more quickly. In the fast mode, the pictorial network display and state-versus-time graph for service center 1 are erased and not updated. Additionally the estimates are only updated after every 10 event occurrences. Pressing < F > a second time slows the simulation and resumes updating the graphics. Upon return to the slow mode, it may take some time for the pictorial network display to catch up with the actual status of the network.

- <S> - pressing S for "Stop", returns the simulation to the program menu.

Because JACKQUE only looks for keyboard commands while incrementing the simulation clock, there may be a slight delay between the user input and the program response.

H. PROGRAM LIMITATIONS

JACKQUE always starts the simulation with the network completely empty of customers. There is no way of starting the simulation in any other state. It is also not possible to change the model parameters once the simulation is started. If it is desired to do so, the present simulation must be terminated, the parameters changed, and a new simulation started.

JACKQUE uses string arrays to keep track of the customers and their feedback count as they move through the network. The maximum length of a string in BASIC is 256. Because of this restriction, JACKQUE can not have more than 256 customers waiting for either service center 1 or service center 2. If this condition is met, the simulation will automatically be terminated and the following message displayed on the screen:

```
Program execution HALTED. Queue
length exceeds program capabilities.
Press any key to return to Main
Program Menu.
```

JACKQUE is incapable of explicitly tracking any individual customer for more than 9 feedbacks from service center 2 to service center 1. Any customer who has experienced feedback nine or more times will be represented as a *purple* ball with a number "9" in the middle. Upon seeing this symbol, it is not known if that customer has experienced feedback 9 or more times. This limitation causes the estimates of S_1 and S_2 to be smaller than the theoretical values. This bias can be minimized through the judicious selection of the model parameters as discussed in Section IV-I.

Although model events are occurring continuously in time, JACKQUE updates the graphics display only after the passage of a discrete amount of time, Δt. It is therefore possible for one or more events to occur during Δt. When JACKQUE plots the events on the pictorial display, all events occurring during Δt are plotted as if they occurred at the end of Δt. They are plotted one immediately after the other, in the proper chronological order in which they actually occurred.

The time it takes the computer to physically plot the movement of the balls is not considered by JACKQUE as part of the simulation time. Because the number of events plotted after each Δt is variable, and because certain events take longer to plot than others, the illusion of real time passage to the user is some what distorted. By distorted, it is meant that the increments of the clock as shown on the graphics display are not always of equal time duration.

The illusion of real time passage can be maintained, and the frequency of multiple state jumps can be minimized through the judicious selection of the model parameters as discussed in the next section.

I. GUIDANCE ON PARAMETER SELECTION

The behavior of the network is controlled by the five parameters λ, μ_1, μ_2, P_1, and P_2. Many of the shortcomings of the graphical display can be overcome by the careful selection of these parameters.

The illusion of real time passage can be preserved by limiting the frequency with which multiple events occur during Δt. The inter-arrival rate and service rates should be selected small enough so that the probability of numerous arrivals or service completions occurring in Δt is small. Picking small rates will also improve the users ability to follow the progress of a customer in the network. If the customers are moving too quickly, the graphics display can become confusing to the user. Finally, the selection of small rates will minimize the number of times that the state-versus-time graphs will exhibit multiple jumps.

Although some feedback makes the model more interesting, it is desirable to keep the average amount of looping for a customer below nine. This minimizes the amount of bias introduced into the estimates for S_1 and S_2. It also reduces the number of times a customer is represented as a ball with a number "9" in the center. This is desirable because the user does not know the true meaning of a number nine. The amount of looping is completely controlled by the values selected for P_1 and P_2.

It is also desirable to keep the average queue size in front of the service centers close to the number that can physically be shown on the computer screen. Although this is not essential, it makes the graphic simulation easier to understand. This is accomplished by keeping the service rates at the service centers some what larger than the effective arrival rates.

Selecting the parameters to accomplish all of these things can be difficult because they are so highly interrelated. Table 1 contains some parameter values that have yielded interesting results. One possible method for selecting parameters is presented below:

1. Select λ, the arrival rate of customers arriving at service center 1 from outside the network.
2. Select P_1 and P_2, the probabilities of exiting the network at service completion.
3. Compute R_1 and R_2, the effective arrival rates:

$$R_1 = \lambda/\{1-(1-P_1)(1-P_2)\}$$
$$R_2 = (1-P_1)(R_1)$$

4. Select μ_1 and μ_2 so that:

$$\mu_1 > R_1$$
$$\mu_2 > R_2$$

5. Run the simulation with these parameters and see how it performs.

It is required that μ_1 and μ_2 be greater than zero so that when calculating the theoretical performance quantities, division by zero is not attempted.

TABLE I

SUGGESTED PARAMETERS

λ	P_1	P_2	μ_1	μ_2	S_1	S_2	L_1	L_2
0.01	0.10	0.10	0.06	0.05	5.3	4.7	7.14	18.0
0.05	0.20	0.20	0.14	0.12	2.8	2.2	24.9	12.5
0.08	0.10	0.10	0.60	0.50	5.3	4.7	2.35	3.13
0.10	0.20	0.30	0.25	0.20	2.3	1.8	10.0	10.0
0.10	0.10	0.10	0.60	0.55	5.3	4.7	7.14	6.20
0.10	0.50	0.50	0.15	0.07	1.3	.67	8.00	20.0
0.15	0.90	0.90	0.16	0.05	1.0	.10	17.9	0.43
0.20	0.30	0.20	0.50	0.35	2.3	1.6	10.0	10.0
0.30	0.20	0.60	0.60	0.36	1.5	1.2	2.78	50.0

The Jackson network being modeled can be transformed into a simple M/M/1 queue by selecting $P_1 = 1$. Likewise, it can be transformed into a simple M/M/2 queue by selecting $P_1 = 0$ and $P_2 = 1$.

J. PROGRAMING MODIFICATIONS

This section is intended to describe some minor changes that can be made to the program JACKQUE.BAS. A BASIC programing manual should be consulted for the specifics of how to change and then save the altered program.

Program line 3306 specifies the default drive on which the program looks for the random number generator sub-routine RNGEN.SRT. To change the default drive, simply replace DRIVE$ = "A:" with the appropriate drive label. For example, if it is replaced with DRIVE$="B:", the program will automatically look on drive B for the sub-routine.

Program line 3455 contains the default parameters. These are the parameters used in the simulation if the user does not specify something else. To change one or more of the default parameters, simply replace the existing value with the desired value.

The lengths of the audible tones used in the program are set in program line 3456. The variable DUR1 sets the duration of the low pitched tone corresponding to a customer arrival. The variable DUR2 sets the duration of the high pitched tone corresponding to a departure from the network. Either tone duration can be changed by simply assigning a different positive integer to the appropriate variable. The variable assignment must also be changed in program lines 8605 and 8610.

The actual tones of the sounds are determined by the number appearing in the statements "SOUND 3000, DUR2", and "SOUND 2500,DUR1". To change the tones, simply change the number appearing after SOUND. These statements appear in program lines 12035, 18015, 18135, 23015, and 23135.

The relationship $W \geq N$ in program line 5088 determines how often the estimated quantities in the displayed table are updated. They are updated every time W exceeds N. W is the count of how many events have occurred since the last update. The value of N is set in lines 3470, 9505, and 9515. The variable W is initialized in program line statements 3475 and 5088 and incremented in line 5156. As presently written, in the slow mode the estimates are updated after every event and in the fast mode they are updated after every 10 events.

V. CONCLUSIONS

This paper presented the development of a graphic simulation of a Jackson network. The method of graphic simulation allows the user insight into the dynamic nature of the Jackson network, not possible through other instructional methods.

There are numerous other systems that could be better understood through the use of graphic simulation. A few that come to mind are:

- Queues with finite storage.
- Queues with bulk arrivals
- Queues with bulk service.
- Multi-channel queues with one waiting line.
- Multi-channel queues with multiple waiting lines, with or without jockeying.
- Priority queuing.
- Queues with balking customers.

It is hoped that this effort has been able to assist the reader in better understanding Jackson networks.

APPENDIX A

JACKQUE.BAS PROGRAM LISTING

```
10 '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
20 '++++++++++++++  MAIN PROGRAM MODULE  / MPM  ++++++++++++++++++++++++++
30 '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
35 'JACKQUE.BAS, Ver 1.0, G. F. Greene,   Aug 1986
40  CLEAR,,7300: KEY OFF: CLS
50  GOSUB 1000  'check for basica and color graphics card
60  GOSUB 2000  'Title screen
70  GOSUB 2100  'One time initialization and Housekeeping
80  GOSUB 3200  'load user defined functions
90  GOSUB 3300  'load random number generator
100 GOSUB 3400  'initialization for individual simulation run
110 GOSUB 3500  'print main program menu
120 GOSUB 3600  'compute theoretical statistics
130 GOSUB 3900  'print main display
140 GOTO  5000  'transfer control to Clock Module
150 SCREEN 0: WIDTH 80: PRINT "Program Ended at Your Request . . ."
160 END
161 '
1000 '======= BASICA & COLOR/GRAPHICS ADAPTER CHECK SBRT / MPM ==============
1005 DEF SEG = 0: IF (PEEK(&H410) AND &H30) <> &H30 THEN DEF SEG: GOTO 1030
1010 LOCATE 3,1: PRINT "Sorry...."
1015 PRINT "You do not have the color/graphics monitor adapter!"
1020 PRINT "This simulation uses graphics and requires that adapter."
1025 DEF SEG: END
1030 ON ERROR GOTO 1035: PLAY "p16": GOTO 1050
1035 WIDTH 80: CLS: LOCATE 3,1
1040 PRINT "This simulation uses advanced BASIC."
1045 PRINT "Return to DOS by typing  SYSTEM' and then"
1046 PRINT "reload this program after using the command  BASICA'.":END
1050 ON ERROR GOTO 0: RETURN   'return to line 60
1051 '
2000 '=================== TITLE SCREEN / MPM ===============================
2005 SCREEN 0,1: COLOR 14,1,1: WIDTH 40: CLS
2010 LOCATE 2,12: PRINT "GRAPHIC SIMULATION"
2015 LOCATE 4,19: PRINT "of a"
2020 LOCATE 6,12: PRINT "JACKSON QUEUE NETWORK"
2025 LOCATE 9,14: PRINT "by G. F. Greene"
2030 LOCATE 14,5: PRINT "Submitted in partial fulfillment"
2035 LOCATE 15,2: PRINT "of the requirements for the degree of"
2040 LOCATE 17,1: PRINT "Master of Science in Operations Research"
2045 LOCATE 19,4: PRINT "from the Naval Postgraduate School"
2050 LOCATE 20,11: PRINT "Monterey, California"
2055 LOCATE 22,4: PRINT "Advisors J.D. Esary, G.F. Lindsay"
2060 LOCATE 25,2: PRINT "Press any Key to continue...";: GOSUB 29000:RETURN
2061 '
2100 '=======  ONE TIME INITIALIZATION AND HOUSEKEEPING SBRT / MPM ===========
2111 SCREEN 0,1: COLOR 14,1,1: WIDTH 40: CLS
2112 LOCATE 10,5: PRINT"Program Initialization . . . "
2113 LOCATE 12,13: PRINT "Please Wait"
2114 LOCATE 13,13,0  'hide the curser
2120 DEFINT A-Z  'define all variables as integers
2130 DIM STAT!(2), Q(2), NSCT!(2), EXIT(2), STRG1$(255), STRG2$(255),SERV2(2200),
     CIRC0(35), CIRC1(35), CIRC2(35),   CIRC3(35), CIRC4(35),circ5 (35), CIRC6(35),
     CIRC7(35), CIRC8(35), CIRC9(35),TEMPCIRC(35)
2140 DIM ARROW1(300), ARROW1C(300), ARROW2(100), ARROW2C(100), ARROW3(300),
     ARROW3C(300), ARROW4(200), ARROW4C(200), ARROW5(600), ARROW5C(600),SHIFT(2200),
     STATISTICS(2200), SERVBY1(10), SERVBY2(10)
2150 FMT1$ = "####": FMT2$ = "####.###" : FMT3$ = "#.##": FMT4$ = "##.##":
     FMT5$ = "###"
2160 TRUE = -1 : FALSE = 0: FIRSTSCREEN = TRUE
```

```
2165 RETURN  'return to line 80
2166 '
3200 '================== USER DEFINED FUNCTIONS SBRT / MPM ===================
3204 DEF FNROUNDUP%(A!) = INT(A!) + 1
3205 DEF FNEXPONT!(A1!, A2!) = -LOG(A1!)/A2!
3210 DEF FNMIN!(A!,B!,C!) = (-(A!<B! AND A!<C!)*A!) + (-(B!<A! AND B!<C!)*B!) +
                                              (-(C!<A! AND C!<B!)*C!)
3220 RETURN  'return to line 90
3221 '
3300 '================ LOAD RANDOM NUMBER GENERATOR / MPM ====================
3305 RNGEN=0: U!=0: 'initialize random number generator
3306 DRIVE$ = "A:" 'program looks on A drive for RNGEN.SRT first
3310 ON ERROR GOTO 3329 'RNGEN.SRT cannot be found on default disk drive.
3315 DEF SEG = &H1A00: BLOAD DRIVE$ + "rngen.srt",0  'load rngen.srt
3320 ON ERROR GOTO 0  'turn error trapping off
3325 RETURN  'return to line 100
3327 '
3329 '=============FILE LOADING ERROR TRAPPING SBRT / MPM =================
3330 IF ERR <> 53 AND ERR <> 76 THEN ON ERROR GOTO 0  'turn error trapping off
3335 CLS: LOCATE 3,1: PRINT "On which drive can the random number"
3336 PRINT "generator program be found?"
3345 GOSUB 29000 'capture keyboard input
3350 DRIVE$ = RESPONSE$ + ":"
3355 RESUME 3315
3356 '
3400 '==========  INITIALIZATION OF SINGLE SIMULATION RUN SBRT / MPM ==========
3420 FOR I = 1 TO 2
3425     Q(I) = 0              'queue counters
3430     STAT(I) = 0           'status of servers: 1=busy 0=idle
3440     NSCT!(I) = 1E+34      'set next service time at 1 equal to infinity
3445 NEXT
3446 FOR I = 1 TO 10
3447   SERVBY1(I) = 0          'number of customers served I times by server 1/2
3448   SERVBY2(I) = 0
3449 NEXT
3450 MODE = 2            'graphics mode: 2=with stats table; 1=w/o table
3451 TIME = 0            'integer clock time
3452 CTIME! = 0          'continuous real time
3453 TA1 = 0             'total arrivals at server 1
3454 TA2 = 0             'total arrivals at server 2
3455 LAMBDA! = .1 : U1! = .2 : U2! = .13: P1! =.3 : P2! =.5 'default parameters
3456 DUR1 = 3 : DUR2 = 2       'duration of audible beeps
3457 UNSTABLE = 0              'unstable parameters 0=no 1=yes
3458 E1 = 0              'total exits after server 1
3459 E2 = 0              'total exits after server 2
3460 ET = 0              'ET = E1 + E2
3461 NA1 = 0             'exogenous arrivals at server 1
3462 FB1 = 0             'feed-back arrivals at server 1
3463 FAST = 0            'display speed: 0=slow 1=fast
3464 CUMSERV1! = 0: CUMSERV2! = 0   'amount of time server 1 & 2 busy
3465 STRG1$ = "" : STRG2$ = ""      'queue strings for server 1 and 2
3469 NEXTEVENT = 0: XX = 23
3470 N = 1                          'how frequently stats table updated
3475 W = 0                          'event counter
3485 RETURN    'return to line 110
3486 '
3500 '=================  MAIN PROGRAM MENU SBRT / MPM ====================
3505 BORDER$ = STRING$(40,223)
3510 SCREEN 0,1: COLOR 14,3,0: WIDTH 40: CLS
3515 LOCATE 3,14: PRINT "PROGRAM MENU": PRINT BORDER$
3520 LOCATE 6,3: PRINT "<I>nstructions"
3525 LOCATE 8,3: PRINT "<N>ew random number Generator Seed"
3535 LOCATE 10,3: PRINT "<D>efault Model Parameters"
3540 LOCATE 12,3: PRINT "<C>hange Model Parameters"
3545 LOCATE 14,3: PRINT "<E>nd the Program"
3547 LOCATE 15,3: PRINT BORDER$
3550 LOCATE 18,3: PRINT "Enter your selection:";
3555 GOSUB 29000 'capture keyboard response
3560 A% = INSTR("INDCEindce", RESPONSE$)
```

```
3565 IF A% = 0 THEN 3510 ELSE ON A% GOTO 4000,4200,120,4500,150,4000,4200
     ,120,4500,150
3566 '
3600 '============== THEORETICAL STATISTICS SBRT / MPM ====================
3605 IF (P1!=0 AND P1!=P2!) THEN UNSTABLE =1: GOTO 4600   'explosive queue
3610 R1! = LAMBDA!/(1-(1-P1!)*(1-P2!))
3620 R2! = (1 - P1!) * R1!
3625 IF (R1!>=U1! OR R2!>=U2!) THEN UNSTABLE= 1: GOTO 4600 'explosive queue
3630 S1! = 1/(P1! + ((1-P1!)*(P2!)))
3640 S2! = (1-P1!)* S1!
3650 L1! = R1!/(U1! - R1!)
3660 L2! = R2!/(U2! - R2!)
3670 L! = L1! + L2!
3680 W1! = L1!/R1!
3690 W2! = L2!/R2!
3700 W! = L!/LAMBDA!
3710 Z! = (R1!/U1!)*(1-R2!/U2!)
3720 RETURN  'return to line 130
3721 '
3900 '=============== PRINT MAIN DISPLAY SBRT / MPM ======================
3905 SCREEN 1,0: COLOR 0,1
3910 IF FIRSTSCREEN THEN GOSUB 4700: FIRSTSCREEN = FALSE   'one time printing
3915 GOSUB 4800 'main screen drawing sbrt                   of graphics symbols
3920 GET (158,93) - (319,199), STATISTICS   'put stats table on display
3935 LOCATE 11,21: PRINT USING FMT1$; TIME
3940 LOCATE 3,8 : PRINT USING FMT4$; LAMBDA!
3945 LOCATE 9,19: PRINT USING FMT4$; U1!
3950 LOCATE 9,33: PRINT USING FMT4$; U2!
3955 LOCATE 3,26: PRINT USING FMT3$; P1!
3960 LOCATE 3,37: PRINT USING FMT3$; P2!
3965 LOCATE 7,27 : PRINT USING FMT5$; Q(2)
3970 LOCATE 7,10 : PRINT USING FMT5$; Q(1)
3980 RETURN   'return to line 140
3981 '
4000 '=================== USER INSTRUCTION SBRT / MPM ======================
4001 SCREEN 0,1: COLOR 14,1,1: WIDTH 40: CLS
4002 LOCATE 1,6: PRINT "Instructions for the Program Menu"
4003 LOCATE 3,2: PRINT "Press <s> to set the random number seed";
4004 LOCATE 4,1: PRINT "If <d> or <c> is pressed before setting"
4006 LOCATE 5,1: PRINT "a seed, the default seed will be used"
4010 LOCATE 7,2: PRINT "Press <c> to change model parameters."
4012 LOCATE 8,1: PRINT "The program will ask for the arrival"
4014 LOCATE 9,1: PRINT "rate (lambda), the service rate for"
4016 LOCATE 10,1:PRINT "server 1 (u1), the service rate for"
4018 LOCATE 11,1:PRINT "server 2 (u2), the probability of exit"
4020 LOCATE 12,1:PRINT "after server 1 (P1), and the probability"
4022 LOCATE 13,1:PRINT "of exit after server 2.  The rates can"
4024 LOCATE 14,1:PRINT "be any positive non-zero number while "
4026 LOCATE 15,1:PRINT "P1 & P2 must be between 0 and 1. The"
4028 LOCATE 16,1:PRINT "simulation starts after entering P2."
4030 LOCATE 18,2:PRINT "Press <d> to use the default parameters";
4032 LOCATE 19,1:PRINT "The simulation will immediately start";
4034 LOCATE 20,1:PRINT "with the values lambda=.10, u1=.20,"
4036 LOCATE 21,1:PRINT "u2=.13, P1=.30, and P2=.50."
4038 LOCATE 23,2:PRINT "Press <e> to end the program.";
4040 LOCATE 25,2:PRINT "Press any key to continue...";: GOSUB 29000
4050 CLS: LOCATE 1,13: PRINT "Keyboard Commands"
4051 LOCATE 3,2: PRINT "During the simulation you may use the"
4052 LOCATE 4,2: PRINT "following keys:"
4054 LOCATE 6,4: PRINT "<p> - Pause the simulation."
4056 LOCATE 8,4: PRINT "<c> - Continue the simulation."
4058 LOCATE 10,4:PRINT "<b> - turn on and off the audio Beeps"
4060 LOCATE 12,4:PRINT "<d> - change the Display"
4062 LOCATE 13,10:PRINT "configuration back and forth "
4064 LOCATE 14,10:PRINT "between the state diagrams and"
4065 LOCATE 15,10:PRINT "the statistics table."
4066 LOCATE 17,4: PRINT "<f> - turn on and off the"
4067 LOCATE 18,10: PRINT "fast display mode."
4068 LOCATE 20,4:PRINT "<s> - Stop the simulation and "
```

```
4070 LOCATE 21,10:PRINT "return to the program menu."
4072 LOCATE 25,2:PRINT "Press any key to continue...";: GOSUB 29000
4080 CLS: LOCATE 1,13: PRINT "Graphics Display"
4081 LOCATE 3,2: PRINT "The variables appearing on the display
4082 LOCATE 4,1: PRINT "are defined as:"
4084 LOCATE 6,1: PRINT "lambda: Arrival rate from outside the"
4086 LOCATE 7,7: PRINT "queue network."
4088 LOCATE 9,1: PRINT "u1: Service rate for server 1."
4090 LOCATE 11,1:PRINT "u2: Service rate for server 2."
4092 LOCATE 13,1:PRINT "P1: Probability of exit after server 1."
4094 LOCATE 15,1:PRINT "P2: Probability of exit after server 2."
4096 LOCATE 17,1:PRINT "Q1: Number of people waiting for"
4098 LOCATE 18,5:PRINT "server 1."
4100 LOCATE 20,1:PRINT "Q2: Number of people waiting for"
4102 LOCATE 21,5:PRINT "server 2."
4104 LOCATE 23,1:PRINT "Time: Current time on simulation clock."
4106 LOCATE 25,2:PRINT "Press any key to continue...";: GOSUB 29000
4120 CLS: LOCATE 1,1: PRINT "N(t): No. of people waiting and/or being
4122 LOCATE 2,7: PRINT "served by given server at time t."
4124 LOCATE 4,1: PRINT "R1: Effective arrival rate at server 1,"
4126 LOCATE 5,5: PRINT "including feedback."
4128 LOCATE 7,1: PRINT "R2: Effective arrival rate at server 2,"
4130 LOCATE 8,5: PRINT "including feedback."
4132 LOCATE 10,1:PRINT "S1: Expected no. of times a person will"
4134 LOCATE 11,5:PRINT "be serviced by server 1."
4136 LOCATE 13,1:PRINT "S2: Expected no. of times a person will"
4138 LOCATE 14,5:PRINT "be serviced by server 2."
4140 LOCATE 16,1:PRINT "L1: Long run expected no. of people"
4142 LOCATE 17,5:PRINT "either waiting or being served by"
4144 LOCATE 18,5:PRINT "server 1."
4146 LOCATE 20,1:PRINT "L2: Long run expected no. of people
4148 LOCATE 21,5:PRINT "either waiting or being served by"
4150 LOCATE 22,5:PRINT "server 2."
4152 LOCATE 25,2:PRINT "Press any key to continue...";: GOSUB 29000
4160 CLS: LOCATE 1,1: PRINT "L:  L = L1 + L2"
4162 LOCATE 3,1: PRINT "W1: Expected time a person spends"
4164 LOCATE 4,5: PRINT "waiting or being served by server 1."
4166 LOCATE 6,1: PRINT "W2: Expected time a person spends"
4168 LOCATE 7,5: PRINT "waiting or being served by server 2."
4170 LOCATE 9,1: PRINT "W:  Total expected time in network."
4172 LOCATE 11,1:PRINT "Z:  Long run proportion of time server 1"
4174 LOCATE 12,5:PRINT "is busy and server 2 is idle."
4176 LOCATE 14,1:PRINT "Limit: Values listed under the column"
4178 LOCATE 15,8:PRINT "heading  limit' are the limiting"
4180 LOCATE 16,8:PRINT "values as time goes to infinity"
4182 LOCATE 17,8:PRINT "of the above defined quantities."
4184 LOCATE 19,1:PRINT "Estimate: Values appearing under the"
4186 LOCATE 20,11:PRINT"heading  estimate' are the"
4188 LOCATE 21,11:PRINT"estimates of the limiting"
4190 LOCATE 22,11:PRINT"values. They are calculated"
4192 LOCATE 23,11:PRINT"with data collected from"
4194 LOCATE 24,11:PRINT"the simulation.";
4196 LOCATE 25,2:PRINT "Press any key to return to the menu...";:GOSUB 29000
4198 GOTO 3500    'return to the menu
4199 '
4200 '========= RANDOM NUMBER GENERATOR SEED SBRT / MPM =====================
4205 COLOR 14,0,0: CLS
4210 LOCATE 1,3: PRINT "Set Random Number Generator Seed": PRINT BORDER$
4215 LOCATE 4,2: PRINT "Permissible seed values are integers"
4220 PRINT " in the range: 1 to 2147483646."
4225 LOCATE 7,2: INPUT "Enter the seed value :  " ,A#
4230 IF A#<1 OR A#>2147483646# THEN LOCATE 7,2: PRINT STRING$(39,32): GOTO 4225
4235 DEF SEG = &H1A00
4240 A1! = INT(A#/16777210#): A2! = INT((A#-A1!*16777210#)/65536!)
4245 'poke the seed's upper 2 bytes into RNGEN's seed storage.
4250 POKE &H164,A1!: POKE &H163,A2!
4255 A#=A#-A1!*16777210#-A2!*65536!: A1! = INT(A#/256): A2! = A#-A1!*256
4260 'poke the seed's lower 2 bytes into RNGEN's seed storage.
4265 POKE &H162,A1!: POKE &H161,A2!
```

```
4270 GOTO 3500   'return to the menu
4271 '
4500 '============= CHANGE MODEL PARAMETERS SBRT / MPM =====================
4505 SCREEN 0,1: COLOR 15,5,0: CLS
4510 MSG1$ = "Enter a positive, nonzero number only."
4515 MSG2$ = "Enter a positive value between 0 and 1."
4520 LOCATE 1,6: PRINT "CHANGE MODEL PARAMETERS": PRINT BORDER$
4525 LOCATE 4,1: INPUT "Enter Arrival Rate :  ", LAMBDA!
4530 IF LAMBDA!<=0 THEN LOCATE 23,1:PRINT MSG1$;:LOCATE 4,1:PRINT STRING$(40,32)
     :GOTO 4525
4532 LOCATE 23,1: PRINT STRING$(40,32);
4535 LOCATE 6,1: INPUT "Enter Service Rate for Server 1 :  ", U1!
4540 IF U1! <=0 THEN LOCATE 23,1: PRINT MSG1$;: LOCATE 6,1: PRINT STRING$(40,32):
     GOTO 4535
4542 LOCATE 23,1: PRINT STRING$(40,32);
4545 LOCATE 8,1: INPUT "Enter Service Rate for Server 2 :  ", U2!
4550 IF U2! <=0 THEN LOCATE 23,1: PRINT MSG1$;: LOCATE 8,1: PRINT STRING$(40,32):
     GOTO 4545
4555 LOCATE 23,1: PRINT STRING$(40,32);
4560 LOCATE 10,1: INPUT "Enter Prob of Exit after Server 1: ",P1!
4565 IF ((P1! < 0) OR (P1! > 1)) THEN LOCATE 23,1: PRINT MSG2$;:
      LOCATE 10,1: PRINT STRING$(40,32): GOTO 4560
4567 LOCATE 23,1: PRINT STRING$(40,32);
4570 LOCATE 12,1: INPUT "Enter Prob of Exit after Server 2: ",P2!
4575 IF ((P2! < 0) OR (P2! > 1)) THEN LOCATE 23,1: PRINT MSG2$;:
     LOCATE 12,1: PRINT STRING$(40,32): GOTO 4570
4577 LOCATE 23,1: PRINT STRING$(40,32);
4580 GOTO 120    'start simulation
4581 '
4600 '============== EXPLOSIVE QUEUE WARNING SBRT  / MPM ===================
4605 SCREEN 0,1: COLOR 14,3,0: WIDTH 40: CLS
4610 LOCATE 6,5: PRINT "Your parameter selection results"
4615 LOCATE 7,5: PRINT "in an Unstable (EXPLOSIVE) queue."
4620 LOCATE 8,5: PRINT "The graphical presentation of the"
4625 LOCATE 9,5: PRINT "network will still be accurate;"
4630 LOCATE 10,5:PRINT "However the network statistics"
4635 LOCATE 11,5:PRINT "will be INVALID, and will NOT be"
4640 LOCATE 12,5:PRINT "displayed. See the user's manual"
4641 LOCATE 13,5:PRINT "for an explanation."
4645 LOCATE 15,10:PRINT "<C>ontinue with Simulation"
4650 LOCATE 16,10:PRINT "<R>eturn to the Program MENU"
4655 LOCATE 18,5: PRINT "ENTER your Selection"
4660 GOSUB 29000 'Capture Keyboard input
4665 A% = INSTR("CRcr", RESPONSE$)
4670 IF A% = 0 THEN 4605 ELSE ON A% GOTO 3720, 100, 3720, 100
4671 '
4700 '============== DRAW & SAVE CIRCLES/ARROWS SBRT / MPM =================
4701 'draw circles
4702 CIRCLE (11,35),7,1:PAINT(11,35),1,1
4704 Y=4 : X=27
4706 FOR I=1 TO 9
4708     CIRCLE (X,35),7,2: PAINT(X,35),2,2
4710     LOCATE 5,Y : PRINT RIGHT$(STR$(I),1)
4712     X=X+16
4714     Y=Y+2
4716 NEXT
4717 'draw arrows colored and uncolored
4718 DRAW "c3; bm 0,47; m4,47;h4;f4;g4;e4; m135,47; h4; f4; g4"
4720 DRAW "c2; bm 0,63; m4,63;h4;f4;g4;e4; m135,63; h4; f4; g4"
4722 DRAW "c3; bm 159,47; m 271,47; h4; f4; g4"
4724 DRAW "c2; bm 159,63; m 271,63; h4; f4; g4"
4726 DRAW "c3; bm 303,79; m 15,79; f4; h4; e4"
4728 DRAW "c3; bm 207,79; f4 h4; e4; bm 111,79; f4; h4; e4"
4730 DRAW "c2; bm 303,95; m 15,95; f4; h4; e4"
4732 DRAW "c2; bm207,95; f4; h4; e4; bm 111,95; f4; h4; e4"
4734 DRAW "c3; bm 295,47; m319,47; h4; f4; g4"
4736 DRAW "c2; bm 295,63; m 319,63; h4; f4; g4"
4738 DRAW "c3; bm 223,39; m 223,11; f4; h4; g4"
4740 DRAW "c2; bm 239,39; m 239,11; f4; h4; g4"
```

```
4741 'save circles for future plotting
4742 GET (4,28) - (18,42), CIRC0
4744 GET (20,28) - (34,42), CIRC1
4746 GET (36,28) - (50,42), CIRC2
4748 GET (52,28) - (66,42), CIRC3
4750 GET (68,28) - (82,42), CIRC4
4752 GET (84,28) - (98,42), CIRC5
4754 GET (100,28) - (114,42), CIRC6
4756 GET (116,28) - (130,42), CIRC7
4758 GET (132,28) - (146,42), CIRC8
4760 GET (148,28) - (162,42), CIRC9
4761 'save arrows for future plotting
4762 GET(0,47) - (135,46), ARROW1
4764 GET(0,63) - (135,62), ARROW1C
4766 GET(223,11) - (224,39), ARROW2
4768 GET(239,11) - (240,39), ARROW2C
4770 GET(159,47) - (271,48), ARROW3
4772 GET(159,63) - (271,64), ARROW3C
4774 GET(295,47) - (319,48), ARROW4
4776 GET(295,63) - (319,64), ARROW4C
4778 GET(15,79) - (303,80), ARROW5
4780 GET(15,95) - (303,96), ARROW5C
4782 CLS
4784 RETURN  'return to line 3915
4785 '
4800 '================ MAIN SCREEN DRAWING SBRT / MPM =====================
4801 'print network picture
4802 LINE (0,8)-(319,8),2 : LINE (0, 10)-(319,10),2
4804 LOCATE 1,14: PRINT "JACKSON QUEUE",
4806 DRAW "c3; bm 0,43; m 135,43; h4; f4; g4"
4808 LINE (135,31)-(159,55),3,B
4810 DRAW "c3; bm 159,43; m 271,43; h4; f4; g4"
4812 DRAW "c3; bm 167,43; m 167,15; f4; h4; g4"
4814 LINE (271,31) - (295,55),3,B
4816 DRAW "bm 295,43; m 319,43; h4; f4; g4"
4818 DRAW "bm 303,43; m 303,76; h4; f4; e4; g4"
4820 DRAW "m 15,76; f4; h4; e4; g4"
4822 DRAW "m 15,43; f4; h4; g4"
4824 DRAW "bm4,43; h4; f4; g4; bm 207,76; f4; h4; e4; bm 111,76; f4; h4; e4"
4826 LOCATE 3,1: PRINT "lambda="
4828 LOCATE 3,23: PRINT "P1="
4830 LOCATE 3,34: PRINT "P2="
4832 LOCATE 7,7: PRINT "Q1="
4834 LOCATE 9,16: PRINT "u1="
4836 LOCATE 7,24: PRINT "Q2="
4838 LOCATE 9,30: PRINT "u2="
4840 LOCATE 11,16: PRINT "TIME:"
4841 'print state-verses time graphs
4842 LINE (0,90) - (319,90),2 : LINE (0,92) - (319,92),2
4844 LINE (157,92)-(157,199),2
4846 LINE (23,185)-(23,104)
4848 LINE (183,185)-(183,104)
4850 FOR Y = 107 TO 179 STEP 8
4852     LINE (19, Y) - (23,Y)
4854     LINE (179,Y)-(183,Y)
4856 NEXT
4858 LINE (23,185)-(151,185)
4860 LINE (183,185)-(311,185)
4862 FOR X = 27 TO 151 STEP 4
4864     IF (X=63) OR (X=103) OR (X=143) THEN GOTO 4870
4866     LINE (X,185) - (X,188)
4868     LINE (X+160, 185)-(X+160,188): GOTO 4874
4870           LINE (X,185)-(X,190)
4872           LINE (X+160,185)-(X+160,190)
4874 NEXT
4876 FOR COL = 1 TO 21 STEP 20: K = 10
4878     FOR ROW = 15 TO 23 STEP 2
4880         LOCATE ROW, COL
4882         K=K-2
```

```
4884         PRINT STR$(K)
4886      NEXT
4888 NEXT
4890 LOCATE 13,1: PRINT "N(t)"
4892 LOCATE 13,21: PRINT "N(t)"
4894 LOCATE 25,8: PRINT "SERVER 1";
4896 LOCATE 25,27: PRINT "SERVER 2";
4898 GET (158,93)-(319,199), SERV2   'save server 2 state graph
4900 LINE (158,93)-(319,199),0,BF    'erase server 2 state graph
4902 IF UNSTABLE = 0 THEN GOTO 4910  'stable do not print warning
4904 LOCATE 18, 24: PRINT "Unstable Queue"
4906 LOCATE 19,23: PRINT "Model Statistics"
4908 LOCATE 20,25: PRINT "Are Invalid": GOTO 4936
4910 LOCATE 13,21: PRINT "VAR ESTIMATE  LIMIT"
4912 LOCATE 14,21: PRINT "R1   0.000"
4914 LOCATE 15,21: PRINT "R2   0.000"
4916 LOCATE 16,21: PRINT "S1   0.000"
4918 LOCATE 17,21: PRINT "S2   0.000"
4920 LOCATE 18,21: PRINT "L1   0.000"
4922 LOCATE 19,21: PRINT "L2   0.000"
4924 LOCATE 20,21: PRINT "L    0.000"
4926 LOCATE 21,21: PRINT "W1   0.000"
4928 LOCATE 22,21: PRINT "W2   0.000"
4930 LOCATE 23,21: PRINT "W    0.000"
4932 LOCATE 24,21: PRINT "Z    0.000";
4934 Y = 32: GOSUB 30000             'print stat limits to screen
4936 RETURN   'return to line 3920
4937 '
5000 '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
5010 '+++++++++++++++++++   CLOCK MODULE  / CM  +++++++++++++++++++++++++++
5030 '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
5031 DEF SEG = &H1A00: CALL RNGEN(U!)   'generate a uniform(0,1) RV
5032 NAT! = FNEXPONT!(U!, LAMBDA!)       'initialize first exogenous arrival
5033 GOSUB 25000    'initialize exit(1)  - exit after server 1?: 1=yes 0=no
5034 GOSUB 26000    'initialize exit(2)  - exit after server 2?: 1=yes 0=no
5050 LOCATE 7,27: PRINT USING FMT5$; Q(2)
5080 LOCATE 7,10: PRINT USING FMT5$; Q(1)
5081 IF ((Q(1) >= 250) OR (Q(2) >= 250)) THEN GOSUB 9600  'program crash
5085 IF UNSTABLE = 1 THEN GOTO 5090  'skip updating statistics
5088 IF (W>=N AND MODE = 2) THEN W=0: GOSUB 9000  'update and print statistics
5090 CTIME! = FNMIN!(NAT!, NSCT!(1), NSCT!(2)) 'select next event time
5095 IF CTIME! = 0 THEN GOSUB 7500   'tie-breaking sbrt
5100 IF CTIME! = NAT! THEN NEXTEVNT = 1 ELSE IF CTIME! = NSCT!(1) THEN
        NEXTEVNT = 2 ELSE NEXTEVNT = 3   'determine what next event is
5110 NEXTEVNTTIME = FNROUNDUP%(CTIME!)   'round up to next integer
5115 DEF SEG: POKE &H6A,0:          'clear keyboard buffer
5120 WHILE TIME < NEXTEVNTTIME
5125 TIME = TIME + 1
5126 IF FAST = 1 THEN GOTO 5145     'skip updating state graphs
5127 'if time <= 32 skip screen shift sbrt, goto state diagram update sbrt
5130 IF TIME <= 32 THEN GOSUB 6000: XX=XX+4: FOR I = 1 TO 600: NEXT: GOTO 5145
5135 GOSUB 7000  'screen shift subroutine if time >= 32
5140 XX = 147: GOSUB 6000  'state diagram update sbrt
5144 LOCATE 11,21: PRINT USING FMT1$; TIME
5145 'check keyboard buffer for user input and act accordingly
5146 A$ = INKEY$: IF A$ = "" THEN GOTO 5152     'if no keyboard input skip
5147 IF (A$ = "s" OR A$ = "S") THEN GOTO 100
5148 IF (A$ = "p" OR A$ = "P") THEN GOSUB 8000 : GOTO 5152
5149 IF (A$ = "d" OR A$ = "D") THEN GOSUB 8500 : GOTO 5152
5150 IF (A$ = "B" OR A$="b") THEN GOSUB 8600 : GOTO 5152
5151 IF (A$ = "F" OR A$ = "f") THEN GOSUB 9500
5152 DEF SEG: POKE &H6A,0:   'clear keyboard buffer
5155 WEND
5156 W = W+1 'event counter - determines when stats updated in line 5088
5160 ON NEXTEVNT GOTO 10000, 15000, 20000   'arrival module, server 1 module or
5161                                           server 2 module
5162 '
6000 '========= SERVER 1 STATE DIAGRAM UPDATE  SBRT / CM  =================
6010 C1 = 2  'determines color of tic marks: 1=blue 2=purple
```

```
6020 NT1 = STAT(1) + Q(1)   'total customers waiting and being served @ 1
6030 IF NT1 > 9 THEN NT1 = 9: C1 = 1  'can not plot more than 9 circles
6040 LINE (XX, 179 - (8*NT1)) - STEP(4,0), C1  'plot the tic mark
6045 IF MODE=1 THEN GOSUB 6500 'if server 2 state graph is displayed update
6050 RETURN   'return to line 5130 or 5145
6051 '
6500 '========= SERVER 2 STATE DIAGRAM UPDATE  SBRT / CM  ==================
6510 C2 = 2
6520 NT2 = STAT(2) + Q(2)   'total customers waiting and being served  @ 2
6530 IF NT2 > 9 THEN NT2 = 9: C2 = 1  'can not plot more than 9 circles
6540 LINE (XX + 160, 179 - (8*NT2)) - STEP(4,0), C2  'plot the tic mark
6550 RETURN   'return to line 5130 or 5145
6551 '
7000 '=================== SCREEN SHIFT SBRT / CM ==========================
7010 GET (28, 103) - (155,183), SHIFT  'capture server 1 state graph
7020 PUT (24,103), SHIFT, PSET         'shift and re-plot the graph
7025 IF MODE = 2 THEN RETURN           'if not displaying server 2 return
7030 GET (188,103) - (315,183), SHIFT  'capture server 2 state graph
7040 PUT (184,103), SHIFT, PSET        'shift and replot the graph
7050 RETURN   'return to line 5140
7051 '
7500 '=================  TIE BREAKING SBRT / CM ============================
7540 DEF SEG = &H1A00: CALL RNGEN(U!)
7550 IF NSCT!(1) = NSCT!(2) AND NSCT!(1) = NAT! GOTO 7590   '3-way tie
7560 IF NAT! = NSCT!(2) GOTO 7600      '2-way tie
7570 IF NAT! = NSCT!(1) GOTO 7610      '2-way tie
7580 IF U! <= .5 THEN CTIME! = NSCT!(1): NEXTEVNT = 2: GOTO 5110 'return clock
7585 CTIME! = NSCT!(2): NEXTEVNT = 3: GOTO 5110
7590 IF U! <= .333 THEN CTIME! = NAT!: NEXTEVNT = 1: GOTO 5110 'return to clock
7595 IF U! >= .333 AND U! <= .666 THEN CTIME! = NSCT!(1):NEXTEVNT=2: GOTO 5110
7597 CTIME! = NSCT!(2): NEXTEVNT = 3: GOTO 5110  'return to clock module
7600 IF U! <= .5 THEN CTIME! = NAT!: NEXTEVNT = 1: GOTO 5110  'return to clock
7605 CTIME! = NSCT!(2): NEXTEVNT = 3: GOTO 5110  'return to clock module
7610 IF U! <= .5 THEN CTIME! = NAT!: NEXTEVNT = 1: GOTO 5110  'return to clock
7615 CTIME! = NSCT!(1): NEXTEVNT = 2: GOTO 5110  'return to clock module
7620 '
8000 '==================== PAUSE SBRT / CM ================================
8005 'used in flashing of arrows
8010 DEF SEG: POKE &H6A,0:
8020 A$ = INKEY$: IF A$="c" OR A$="C" THEN RETURN ELSE GOTO 8020
8021 '
8500 '========= CHANGE DISPLAY CONFIGURATION SBRT / CM =====================
8505 'replaces server 2 state diagram with stats table or vice/versa
8510 IF MODE = 1 THEN PUT(158,93), STATISTICS, PSET: MODE = 2: RETURN
8520 PUT (158,93), SERV2, PSET: MODE = 1
8530 RETURN  'return to line 5149
8531 '
8600 '================  BEEP SUPPRESSION SBRT / CM ========================
8601 'turns the audible beeps on and off
8605 IF DUR1 = 3 THEN DUR1 = 0 ELSE DUR1 = 3
8610 IF DUR2 = 2 THEN DUR2 = 0 ELSE DUR2 = 2
8615 RETURN  'return to line 5150
8616 '
9000 '========= UPDATE STATISTICAL ESTIMATES SBRT / CM =====================
9005 TA1 = NA1 + FB1
9010 R1! = TA1 / CTIME!
9015 R2! = TA2 / CTIME!
9020 IF CUMSERV1! = 0 THEN SR1! = 0: SR2! = 0: GOTO 9024
9021 SR1! = (E1 + TA2)/CUMSERV1!   'estimate of service rate for server 1
9022 IF CUMSERV2! = 0 THEN SR2 = 0: GOTO 9024
9023 SR2! = (E2 + FB1)/CUMSERV2!   'estimate of service rate for server 2
9024 R! = NA1/CTIME!               'estimate of exogenous arrival rate
9025 S1!=0: S2!=0: ET= E1 + E2: IF ET = 0 THEN GOTO 9056
9030 FOR I = 1 TO 10
9035     S1! = S1! + I*SERVBY1(I)
9040     S2! = S2! + I*SERVBY2(I)
9045 NEXT
9050 S1! = S1!/ET
9055 S2! = S2!/ET
```

```
9056 IF (SR1! <= R1!) THEN L1! = 0: GOTO 9062
9060 L1! = R1!/(SR1! - R1!)
9062 IF (SR2! <= R2!) THEN L2! = 0: GOTO 9070
9065 L2! = R2!/(SR2! - R2!)
9070 L! = L1! + L2!
9075 W1! = L1!/R1!
9077 IF (R2! = 0) THEN W2!=0: GOTO 9085
9080 W2! = L2!/R2!
9085 W! = L!/R!
9086 IF SR1! = 0 THEN Z! = 0: GOTO 9200
9087 IF (P1! = 1) THEN Z! = (R1!/SR1!): GOTO 9200
9088 IF SR2! = 0 THEN Z! = 0: GOTO 9200
9090 Z! = ABS((R1!/SR1!)*(1 - (R2!/SR2!)))
9200 Y=23: GOSUB 30000    'plot estimates to screen
9210 RETURN   'return to line 5090
9211 '
9500 '============== GRAPHICS SUPPRESSION SBRT / CM ========================
9501 'if fast mode is selected no graphics are displayed
9505 IF FAST = 0 THEN FAST = 1 ELSE FAST = 0: N = 1
9510 IF FAST = 1 THEN GOTO 9515 ELSE RETURN
9515 N = 10 'when fast mode selected, stats updated after every N events
9520 LINE (140,36) - (154,50),0,BF   'erase customers and state tic marks
9525 LINE (4,28) - (130,42),0,BF
9530 LINE (276,36) - (290,50),0,BF
9535 LINE (172,28) - (266,42),0,BF
9540 LINE (24,105) - (151,183),0,BF
9545 RETURN   'return to line 5152
9600 '================ PROGRAM CRASH SBRT / CM ===========================
9605 SCREEN 0,1: COLOR 14,3,0: WIDTH 40: CLS
9610 LOCATE 7,3: PRINT "Program execution HALTED. Queue"
9615 LOCATE 8,3: PRINT "length exceeds program capabilities."
9620 LOCATE 9,3: PRINT "Press any key to return to Main"
9621 LOCATE 10,3: PRINT "Program menu.";
9625 GOSUB 29000: GOTO 100  'display program menu
9626 '
10000 '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
10010 '+++++++++++++++++   ARRIVAL MODULE / AM   ++++++++++++++++++++++++
10020 '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
10030 STRG1$ = "0" + STRG1$   '0 corresponds to new arrival, add to queue strg
10035 NA1 = NA1 + 1
10037 IF FAST = 1 THEN GOTO 10042 'skip plotting routine
10040 IF Q(1)<8 THEN GOSUB 11000  ELSE GOSUB 12000: PUT(0,42),ARROW1,PSET
10042 DEF SEG = &H1A00: CALL RNGEN(U!)
10045 AINC! = FNEXPONT!(U!, LAMBDA!) 'compute next inter-arrival time
10050 NAT! = AINC! + CTIME!         'compute new exogenous arrival time
10060 IF STAT(1) = 1 THEN Q(1) = Q(1) + 1: GOTO 5080   'return to clock module
10063 CALL RNGEN(U!)
10065 S1INC! = FNEXPONT!(U!,U1!)    'compute new server 1 service duration
10070  NSCT!(1) = S1INC!  + CTIME!  'compute next service completion time for 1
10080 STAT(1) = 1  'change status of server 1 to busy
10090 GOTO 5081    'return to the clock module
10091 '
11000 '=============== ARRIVAL PLOTTING SBRT / AM ========================
11005 GOSUB 12000  'flash arrow 1
11010 IF STAT(1) = 0 THEN PUT(140,36), CIRCO:GOSUB 28000: PUT(0,42), ARROW1,PSET:
        RETURN  'return to line 10040
11020 PUT(116-Q(1)*16,28), CIRCO:GOSUB 28000: PUT (0,42), ARROW1, PSET
11030 RETURN    'return to line 10040
11031 '
12000 '================  FLASH ARROW 1 SBRT / AM =========================
12010 PUT(0,42), ARROW1C, PSET    'flash purple arrow
12020 FOR I = 1 TO 500
12030 NEXT
12035 SOUND 500,DUR1
12040 RETURN          'return to line 10040 or 11010
12041 '
15000 '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
15010 '+++++++++++++++ SERVICE CENTER 1 MODULE / SC1M  ++++++++++++++++++++
15020 '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
15025 CUMSERV1! = CUMSERV1! + S1INC! 'update amount time server 2 busy
15030 TRANSFER1$ = RIGHT$(STRG1$,1)  'move customer from Q1 to Q2
15035 IF EXIT(1) = 0 THEN STRG2$ = TRANSFER1$ + STRG2$: TA2 = TA2+1
15040 STRG1$ = LEFT$( STRG1$, LEN(STRG1$) - 1)
15050 IF FAST = 0 THEN GOSUB 17000  'server 1 plotting sbrt
15060 IF EXIT(1) = 1 THEN E1 = E1 + 1: GOSUB 18500: GOTO 15100  'estimate S1
15070 IF STAT(2) = 1 THEN Q(2) = Q(2) + 1 : GOTO 15100
15072    DEF SEG = &H1A00: CALL RNGEN(U!)
15075    S2INC! = FNEXPONT!(U!, U2!)  'compute new server 2 service duration
15080    NSCT!(2) = S2INC! + CTIME!   'compute new service completion time
15090    STAT(2) = 1   'change status for server 2 to busy
15099 'make  server 1 completion impossible event
15100 IF Q(1) = 0 THEN NSCT!(1) = 1E+31: STAT(1) = 0: GOTO 15130
15110    Q(1) = Q(1) - 1
15113 DEF SEG = &H1A00: CALL RNGEN(U!)
15115    S1INC! = FNEXPONT!(U!, U1!)   'compute new server 1 service duration
15120    NSCT!(1) = S1INC! + CTIME!    'compute new service completion time
15130 GOSUB 25000         'generate new exit(1) indicator variable
15140 GOTO 5050           'return to clock module
15141 '
17000 '================  SERVER 1 PLOTTING SBRT / SC1M =====================
17010 GET(140,36) - (154,50), TEMPCIRC     'capture circle @ server 1
17020 LINE(140,36) - (154,50),0,BF         'erase circle @ server 1
17030 IF EXIT(1) = 1 THEN GOSUB 18000: GOTO 17060   'flash arrow 2
17035 GOSUB 18100  'flash arrow 3
17040 IF STAT(2) = 0 THEN PUT(276,36), TEMPCIRC : GOTO 17055 'put at server 2
17050 IF Q(2)< 6 THEN PUT(252-(Q(2)*16),28), TEMPCIRC   'put in server 2 queue
17055 GOSUB 28000 :PUT(159,43), ARROW3, PSET   'plot white arrow 3
17060 IF Q(1) = 0 THEN RETURN   'return to line 15060
17070    GET(116,28)-(130,42), TEMPCIRC : LINE (116,28)-(130,42),0,BF:
             PUT(140,36), TEMPCIRC  'put next customer in server 1 box
17080 IF Q(1) = 1 THEN RETURN    'return to line 15060
17090 X = 100
17095 IF Q(1)<= 8 THEN K = (Q(1) -1) ELSE K = 7
17099 'shift all remaining customers to the right one
17100 FOR J = 1 TO K
17110     GET(X,28) - (X+14,42), TEMPCIRC
17115     LINE (X,28) -(X+14,42),0,BF
17120     PUT(X+16,28), TEMPCIRC
17130     X = X-16
17140 NEXT
17150 IF Q(1)<9 THEN RETURN  'return to line 15060
17154 'go to overflow queue and bring next customer into visible queue
17155 LOOPCOUNT1$ = MID$(STRG1$,(LEN(STRG1$)-8),1)
17160 X=4 : Y=28
17170 INDEX = VAL(LOOPCOUNT1$) +1
17175 GOSUB 27000  'bring circle in from queue overflow
17180 RETURN  'return to lne 15060
17181 '
18000 '=================  FLASH ARROW 2 SBRT / SC1M =======================
18010 PUT (167,15), ARROW2C, PSET   'flash purple arrow 2
18015 SOUND 3000,DUR2
18020 FOR I=1 TO 1000
18030 NEXT
18040 PUT(167,15), ARROW2, PSET     'plot white arrow 2
18050 RETURN    'return to line 17030
18051 '
18100 '=================  FLASH ARROW 3 SBRT / SC1M =======================
18110 PUT (159,43), ARROW3C, PSET   'flash purple arrow 3
18120 FOR I=1 TO 500
18130 NEXT
18135 SOUND 500,DUR1
18140 RETURN      'return to line 17030
18141 '
18500 '========= FREQUENCY OF SERVICE BY SERVER 1 SBRT / SC1M ==============
18501 'this sbrt used in calculation of S1
18510 FOR I = 0 TO 9
18520 IF VAL(TRANSFER1$) = I THEN SERVBY1(I+1) = SERVBY1(I+1) + 1: GOTO 18540
18530 NEXT
```

```
18535 IF VAL(TRANSFER1$) = 0 THEN RETURN
18540 FOR I = 1 TO 9
18550 IF VAL(TRANSFER1$) = I THEN SERVBY2(I) = SERVBY2(I) + 1: RETURN
18560 NEXT
18565 RETURN      'return to line 15060
18566 '
20000 '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
20010 '++++++++++++++ SERVICE CENTER 2 MODULE / SC2M ++++++++++++++++++++++
20020 '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
20025 CUMSERV2! = CUMSERV2! + S2INC! 'update amount of time server 2 busy
20027 TRANSFER2$ = RIGHT$(STRG2$,1)
20030 IF EXIT(2) = 1 THEN E2 = E2+1: GOSUB 23500: GOTO 20070  'estimate S2
20045 FB1 = FB1 + 1  'increment feedback at server 1 counter
20050 IF (VAL(TRANSFER2$)<9) THEN TRANSFER2$ =RIGHT$( STR$( VAL(TRANSFER2$)+1),1)
      ELSE  TRANSFER2$ = "9"
20060 STRG1$ = TRANSFER2$ + STRG1$
20070 STRG2$ = LEFT$(STRG2$, LEN(STRG2$) - 1)
20080 IF FAST = 0 THEN GOSUB 22000 'server 2 plotting subrt
20090 IF EXIT(2) = 1 THEN GOTO 20130
20100 IF STAT(1) = 1 THEN Q(1) = Q(1) +1 : GOTO 20130
20102 DEF SEG = &H1A00: CALL RNGEN(U!)
20105 S1INC! = FNEXPONT!(U!, U1!)    'compute new server 1 service duration
20110 NSCT!(1) = S1INC! + CTIME!     'compute new service completion time
20120 STAT(1) = 1  'change status of server 1 to busy
20129 'make service completion at server 2 impossible event
20130 IF Q(2) = 0 THEN NSCT!(2) = 1E+31: STAT(2) = 0: GOTO 20160
20140 Q(2) = Q(2) - 1
20143 DEF SEG = &H1A00: CALL RNGEN(U!)
20145 S2INC! = FNEXPONT!(U!, U2!)    'compute new server 2 service duration
20150 NSCT!(2) = S2INC! + CTIME!     'compute new service completion time
20160 GOSUB 26000    'generate new exit(2) indicator variable
20170 GOTO 5050     'return to clock module
20171 '
22000 '================ SERVER 2 PLOTTING SBRT / SC2M ======================
22010 LINE (276,36) - (290,50),0,BF  'erase circle @ server 2
22020 IF EXIT(2) = 1 THEN GOSUB 23000: GOTO 22060  'flash arrow 4
22030 INDEX = VAL(TRANSFER2$) +1  'increment feedback counter
22035 GOSUB 23100  'flash arrow 5
22040 IF STAT(1) = 0 THEN X=140 : Y=36 : GOSUB 27000 : GOTO 22055
22050 IF Q(1) < =7 THEN X = 116 - Q(1) * 16 : Y=28 : GOSUB 27000
22055 GOSUB 28000: PUT(15,76), ARROW5, PSET 'plot white arrow 5
22060 IF Q(2) = 0 THEN RETURN 'return to line 20090
22069 'erase ball at server 1 and place at server 2
22070 GET(252,28) - (266,42), TEMPCIRC : LINE(252,28)-(266,42),0,BF:
      PUT(276,36), TEMPCIRC
22080 IF Q(2) = 1 THEN RETURN  'return to line 20090
22090 X=236
22100 IF Q(2) <= 6 THEN K = (Q(2)-1) ELSE K = 5
22109 'shift all remaining customers one to the right
22110 FOR J = 1 TO K
22120     GET(X,28) - (X+14,42), TEMPCIRC
22130     LINE(X,28) - (X+14,42), 0, BF
22140     PUT(X+16,28), TEMPCIRC
22150     X=X-16
22160 NEXT
22170 IF Q(2) <= 6 THEN RETURN
22179 'go to overflow queue and bring next customer into visible queue
22180 LOOPCOUNT2$ = MID$(STRG2$, (LEN(STRG2$)-6),1)
22190 X = 172 : Y = 28
22195 INDEX = VAL(LOOPCOUNT2$) + 1
22200 GOSUB 27000    'bring circle in from queue overflow
22205 RETURN    'return to line 20090
22206 '
23000 '================  FLASH ARROW 4 SBRT / SC2M =========================
23010 PUT(295,43), ARROW4C, PSET  'flash purple arrow 4
23015 SOUND 3000, DUR2
23020 FOR I = 1 TO 1000
23030 NEXT
23040 PUT (295,43), ARROW4, PSET 'plot white arrow 4
```

```
23050 RETURN   'return to line 22020
23051 '
23100 '================  FLASH ARROW 5 SBRT / SC2M ========================
23110 PUT(15,76), ARROW5C, PSET   'flash purple arrow 5
23120 FOR I=1 TO 500
23130 NEXT
23135 SOUND 500,DUR1
23140 RETURN   'return to line 22040
23141 '
23500 '======== FREQUENCY OF SERVICE BY SERVER 2 SBRT / SC2M ===============
23501 'this sbrt used in calculation of S2
23510 FOR I=0 TO 9
23520 IF VAL(TRANSFER2$) = I THEN SERVBY2(I+1) = SERVBY2(I+1) + 1:
       SERVBY1(I+1) = SERVBY1(I+1) + 1: RETURN
23530 NEXT
24990 '
24991 '*************************************************************
24992 '*********** SUBROUTINES USED BY MORE THAN ONE MODULE ***************
24993 '*************************************************************
24994 '
25000 '======== PROB EXIT AFTER SERVER 1 SBRT / CM, SC1M,  =================
25004 'exit after service at server 1: 1=yes 0=no
25005 DEF SEG = &H1A00: CALL RNGEN(U!)
25010 IF U! <= P1! THEN EXIT(1) = 1 ELSE EXIT(1) = 0
25015 RETURN  'return to line 5034 or 15140
25016 '
26000 '======== PROB EXIT AFTER SERVER 2 SBRT / CM, SC2M ===================
26004 'exit after service at server 2: 1=yes 0=no
26005 DEF SEG = &H1A00: CALL RNGEN(U!)
26010 IF U!<= P2! THEN EXIT(2) = 1 ELSE EXIT(2) = 0
26015 RETURN  'return to line 5050 or 20170
26016 '
27000 '=============== QUEUE OVERFLOW SBRT / SC1M, SC2M ===================
27010 ON INDEX GOTO 27020, 27030, 27040, 27050, 27060, 27070, 27080,
                              27090, 27100, 27110
27020 PUT(X,Y), CIRC0 : RETURN   'return to line 17180 or 22040 or 22205
27030 PUT(X,Y), CIRC1 : RETURN
27040 PUT(X,Y), CIRC2 : RETURN
27050 PUT(X,Y), CIRC3 : RETURN
27060 PUT(X,Y), CIRC4 : RETURN
27070 PUT(X,Y), CIRC5 : RETURN
27080 PUT(X,Y), CIRC6 : RETURN
27090 PUT(X,Y), CIRC7 : RETURN
27100 PUT(X,Y), CIRC8 : RETURN
27110 PUT(X,Y), CIRC9 : RETURN
27111 '
28000 '================  DELAY SBRT / SC1M, SC2M =========================
28010 FOR I = 1 TO 500
28020 NEXT
28030 RETURN
28031 '
29000 '============ CAPTURE KEYBOARD INPUT SBRT / MPM, CM =================
29005 DEF SEG: POKE &H6A,0:
29010 RESPONSE$ = INKEY$: IF RESPONSE$ = "" THEN 29010
29020 RETURN
29021 '
30000 '============== PRINT STATISTICS SBRT / MPM, CM =====================
30008 'sbrt prevents displaying number that exceed formats
30009 IF ((R1! < .001 AND R1!>0) OR (R1! > 9999!)) THEN R1! = 9999.999
30010 LOCATE 14,Y: PRINT USING FMT2$; R1!
30015 IF ((R2! < .001 AND R2!>0) OR (R2! > 9999!)) THEN R2! = 9999.999
30020 LOCATE 15,Y: PRINT USING FMT2$; R2!
30025 IF ((S1! < .001 AND S1!>0) OR (S1! > 9999!)) THEN S1! = 9999.999
30030 LOCATE 16,Y: PRINT USING FMT2$; S1!
30035 IF ((S2! < .001 AND S2!>0) OR (S2! > 9999!)) THEN S2! = 9999.999
30040 LOCATE 17,Y: PRINT USING FMT2$; S2!
30045 IF ((L1! < .001 AND L1!>0) OR (L1! > 9999!)) THEN L1! = 9999.999
30050 LOCATE 18,Y: PRINT USING FMT2$; L1!
30055 IF ((L2! < .001 AND L2!>0) OR (L2! > 9999!)) THEN L2! = 9999.999
```

```
30060 LOCATE 19,Y: PRINT USING FMT2$; L2!
30065 IF ((L! < .001 AND L!>0) OR (L! > 9999!)) THEN L! = 9999.999
30070 LOCATE 20,Y: PRINT USING FMT2$; L!
30075 IF ((W1! < .001 AND W1!>0) OR (W1! > 9999!)) THEN W1! = 9999.999
30080 LOCATE 21,Y: PRINT USING FMT2$; W1!
30085 IF ((W2! < .001 AND W2!>0) OR (W2! > 9999!)) THEN W2! = 9999.999
30090 LOCATE 22,Y: PRINT USING FMT2$; W2!
30095 IF ((W! < .001 AND W!>0) OR (W! > 9999!)) THEN W! = 9999.999
30100 LOCATE 23,Y: PRINT USING FMT2$; W!;
30105 IF ((Z! < .001 AND Z!>0) OR (Z! > 9999!)) THEN Z! = 9999.999
30110 LOCATE 24,Y: PRINT USING FMT2$; Z!;
30130 RETURN  'return to line 4936 or 9210
```

LIST OF REFERENCES

1. Davison, R.J., *Graphic Simulation of the Poisson Process*, Masters Thesis, Naval Postgraduate School, Monterey, California, October 1982.

2. Nelsen, R.E., *Graphic Simulation of a Machine-Repairman Model*, Masters Thesis, Naval Postgraduate School, Monterey, California, September 1984

3. Giffin, W.C., *Queueing Basic Theory and Application*, Grid Inc., 1978.

4. Heyman, D.P. and Sobel, M.J., *Stochastic Models in Operations Research*, v. 1, McGraw-Hill Book Co., 1982.

5. Ross, S.M., *Introduction to Probability Models*, 3rd ed., Academic Press, 1985.

6. Law, A.M. and Kelton, W.D., *Simulation Modeling and Analysis*, McGraw-Hill Book Co., 1982.

BIBLIOGRAPHY

Enders, B. and Petersen, B., *Basic Primer for the IBM PC and XT*, Plume/Waite Book Co., 1984.

Fishman, G.S., *Concepts and Methods in Discrete Event Digital Simulation*, John Wiley and Sons, 1973.

Jackson, J.R., "Networks of Waiting Lines," *Operations Research*, v. 5, pp. 518-521, 1957.

Kleinrock, L., *Queueing Systems*, v. 1, John Wiley & Sons, 1975.

Maisel, H. and Gnugnoli, G., *Simulation of Discrete Stochastic Systems*, Science Research Associates Inc., 1972.

Marateck, S.L., *BASIC*, 2nd ed., Academic Press, 1982.

Metcalf, C.D., and Sugiyama, M.B., *Beginner's Guide to Machine Language on the IBM PC & PCjr*, Compute! Publications, Inc., 1985.

Waite, M. and Morgan, C.L., *Graphics Primer for the IBM PC*, McGraw-Hill Book Co., 1983.

White, J.A., Schmidt, J.W., and Bennett, G.K., *Analysis of Queueing Systems*, Academic Press, 1975.

Wolverton, V., *Running MS-DOS*, 2nd ed., Microsoft Press, 1985.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Professor J.D. Esary, Code 55Ey Department of Operations Research Naval Postgraduate School Monterey, California 93943-5000	2
4.	Commandant (G-PTE) U.S. Coast Guard 2100 2nd Street SW Washington, DC 20593	2
5.	Lt. Gary F. Greene, USCG Commandant (G-OSR-1) U.S. Coast Guard 2100 2nd Street SW Washington, DC 20593	2

END

2-81

DTIC