THE UNIVERSITY OF MICHIGAN
COLLEGE OF ENGINEERING
Department of Electrical Engineering
SYSTEMS ENGINEERING LABORATORY

Technical Report No. 2

RQA-1—THE RECURSIVE QUEUE ANALYZER

V. L. Wallace
R. S. Rosenberg

ORA Project 07742

under contract with:

ROME AIR DEVELOPMENT CENTER
RESEARCH AND TECHNOLOGY DIVISION
AIR FORCE SYSTEMS COMMAND
CONTRACT NO. AF 30(602)-3953
GRIFFISS AIR FORCE BASE, NEW YORK

administered through:

OFFICE OF RESEARCH ADMINISTRATION ANN ARBOR

February 1966

TABLE OF CONTENTS

TABLE OF CONTENTS (Concluded)

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The Recursive Queue Analyzer, RQA-1, is a computer program designed to evaluate the equilibrium joint probability distributions of queue lengths and system conditions in very large, finite Markovian queueing systems. It will rapidly treat discrete-time and continuous-time systems having as many as 5000 states through numerical solution of equations of balance. The algorithm is generally very much faster and more accurate than equivalent Monte-Carlo simulation. This report is both a description of the organization of the program and a manual for program use.

1. INTRODUCTION

The Recursive Queue Analyzer, RQA-1, is a computer program designed to evaluate the equilibrium joint probability distributions of queue lengths and system conditions in very large, finite Markovian queueing systems. This report is both a description of the organization of the program and a manual for program use.

In the study of networks of queues and servers, the Markovian models represent an exceedingly important class of models. This program has been designed to facilitate the analysis of both discrete-time and continuous-time Markov chains having as many as 5,000 states. The primary design goal has been to provide a computation fast enough to encourage experimentation with the models in the study of system design. This has been achieved through efficient use of available (32K) high-speed storage in the computer and through careful program design. The program was written in the MAD language[1] with selective use of the UMAP assembly language. Although this work has been motivated by the need for it in the design of large computer systems, the potential area of application is quite broad.

For a continuous-time Markov chain with a finite state space, one can always write the Kolmogoroff differential equation

$$\dot{p}_t = p_t Q \tag{1}$$

where p_t is a vector whose i-th element is the probability that the system is in the i-th state at time t, $\dot{p}$ represents the time derivative of p, and Q is a matrix of constants called the transition intensity matrix of the chain. The matrix Q is assumed to have the properties that

$$\sum_j q_{ij} = 0 \tag{2a}$$

for all states i, and

$$q_{ij} \geq 0 \tag{2b}$$

for all $i \neq j$. An equilibrium probability distribution is, by definition, a solution to the equation

$$p\,Q = 0 . \tag{3}$$

However, any solution to Eq. (3) is also a solution to

$$p = p(\Delta Q + I) \tag{4}$$

where Δ is a scalar constant and I is the identity matrix. The purpose of RQA-1 is to solve Eq. (4).

For a discrete-time Markov chain with a finite state space, one can always write the Chapman-Kolmogoroff equation

$$p_{n+1} = p_n A \tag{5}$$

where p_n is the vector whose i-th element is the probability that the system is in the i-th state at time n, and A is a matrix of constants called the transition matrix of the chain. The matrix A is assumed to have the properties that

$$\sum_j a_{ij} = 1 \tag{6a}$$

for all states i, and

$$1 \geq a_{ij} \geq 0 \tag{6b}$$

for all i and j. A real matrix satisfying these properties is called a stochastic matrix. An equilibrium probability distribution in this case is a solution to the equation

$$p = p\,A . \tag{7}$$

Because of the obvious similarity between Eqs. (4) and (7), RQA-1 can conveniently solve either equation through a suitable specification of the input. The most common application, however, is expected to be the continuous-time case, and, hence, a major part of the theoretical discussion and examples is devoted to that case.

The state space will usually originate as a multidimensional vector space with components corresponding to lengths of queues, numbers of occupied servers of a particular type, and similar quantities. However, since the number of such states is assumed finite, it is always possible to represent the state as a single variable i whose values are in correspondence to distinct values of the vector. The variable i is the state variable described in the preceding paragraphs. Nevertheless, the elements p_i of the solution to Eqs. (4) and (7) can be interpreted as joint probabilities of the variables represented in the original multidimensional state space, and appropriate marginal and conditional distributions can be calculated from them as desired.

This report will first treat, in Section 2, the numerical methods which are applied to the solution of Eqs. (3) and (7). This will be followed in Section 3, by a treatment of the technique used for the storage of the transition (intensity) matrix information in the form of what we call a "transition table" and several illustrations of the technique needed for the modeling of systems and the determination of transition tables. Section 4 and Appendix A constitute a description of the structure of the program and the function of its component routines, while Section 5 explains in detail how the program is applied. A person whose only purpose in using RQA-1 is to obtain a set of equilibrium joint probability distributions should be able to skip Sections 2 and 4 without major loss, while a person desiring to alter or to transliterate the program for other computers or purposes will, of course, be most interested in those sections.

The conclusion, Section 6, will discuss the performance of the program and some of the problems associated with its use.

2. NUMERICAL ANALYSIS FOR RQA-1

The RQA-1 employs an iterative procedure to determine the equilibrium probabilities for homogeneous continuous-time and discrete-time Markov chains. An equilibrium probability is found even when the chain is cyclic or when the chain does not have a unique equilibrium probability distribution.

This section is concerned with the conditions for convergence and for uniqueness of solutions. It details the mathematical justification for the procedures used.

2.1 CONDITIONS FOR CONVERGENCE

If G is an arbitrary m-th degree complex matrix, we will say that the iteration process

$$p^{(n+1)} = p^{(n)} G , \tag{8}$$

where $p^{(n)}$ is an m-dimensional row vector, converges if a nonzero vector

$$\lim_{n\to\infty} p^{(n)} = \pi \tag{9}$$

exists. We will say that the iteration process converges uniquely if it converges and if the vector, π, is unique (i.e., independent of the choice of $p^{(0)}$, the initial iterate). Clearly, convergence of the process to π implies that $\pi = \pi G$.

A set of sufficient conditions for the iteration process (Eq. (8)) to converge is that there be an eigenvalue of G at unity, that all other eigenvalues of G which are not equal to unity have modulus less than unity, and that all the unity eigenvalues of G have elementary divisors[2] of first degree. For, from Eq. (8) we may clearly write that

$$p^{(n)} = p^{(0)} G^n . \tag{10}$$

Then from the confluent form of Sylvester's Theorem,[3] we may also write (making use of the above conditions) that

$$p^{(0)} G^n = \lambda_1^n (a_1 y_1 + a_2 y_2 + \ldots + a_r y_r) + O(\lambda_{r+1}^n) \qquad (11)$$

where λ_i represents the i-th eigenvalue (assumed ordered so that $\lambda_1 = \lambda_2 = \ldots = \lambda_r > |\lambda_{r+1}| \geq \ldots \geq |\lambda_m|$), r is the multiplicity of the dominant eigenvalue λ_1, the a_i are scalar constants, and the y_i are m-dimensional row-vectors. If $\lambda_1 = 1$, Eq. (11) evidently converges to $(a_1 y_1 + \ldots + a_r y_r)$, a row-vector. Note also that the rate of convergence is determined by λ_{r+1}, the subdominant eigenvalue.

2.2 CONVERGENCE FOR CONTINUOUS-TIME MARKOV CHAINS

For the continuous-time Markov chains, we are concerned with the convergence of the process

$$p^{(n+1)} = p^{(n)} (\Delta Q + I) \qquad (12)$$

where Q is a transition intensity matrix.

Let $R = \max_i |q_{ii}|$. Then, if $\Delta \leq 1/R$, the matrix $(\Delta Q + I)$ is found to be a stochastic matrix, and, hence, it can be shown[4] that every elementary divisor corresponding to a unity eigenvalue is of the first degree. Since the degree of these elementary divisors is not dependent upon the value of Δ, this implies that they are of first degree regardless of Δ as long as $\Delta \leq \frac{1}{R}$.

Moreover, it is a direct consequence of a theorem of Gershgorin[5] that the eigenvalues of a transition intensity matrix Q will lie on the disc of complex values γ such that

$$|\gamma - R| \leq R \qquad (13)$$

where $R = \max_i |q_{ii}|$. The eigenvalues of $\Delta Q + I$ must, therefore, lie on a disc of complex values such that

$$|\lambda - (\Delta R + 1)| \leq \Delta R . \qquad (14)$$

Thus, if Δ is chosen so that $\Delta < 1/R$, then there can be no eigenvalues of $(\Delta Q + I)$ having unit modulus which are not of unity value, and the iteration process (Eq. (12)) is convergent to a solution of Eq. (4).

On the other hand, while a choice of $\Delta < 1/R$ is sufficient for convergence, it is not always necessary. Indeed, a choice of Δ slightly greater than $1/R$ can often lead to faster convergence than one slightly less, but one cannot know by any simple test when this is so. Hence, in RQA-1, Δ is chosen equal to $.99/R$ to guarantee convergence for continuous-time models.

2.3 CONVERGENCE FOR DISCRETE-TIME MARKOV CHAINS

For discrete-time Markov chains, we are concerned with the convergence of

$$p^{(n+1)} = p^{(n)} A \tag{15}$$

where A is a stochastic matrix. As a result of the theorems cited above, A is known to have only elementary divisors of first degree associated with its unity eigenvalues and to have eigenvalues lying on the unit disc. In order to guarante convergence, one may transform Eq. (15) into the iteration process

$$p^{(n+1)} = p^{(n)} (.99A + .01I) , \tag{16}$$

noting that

$$p = p(.99A + .01I) \tag{17}$$

has the same set of solutions as Eq. (7). As a result, the eigenvalues of $(.99A + .01I)$ lie on the disc of complex values λ' such that

$$|\lambda' - .01| \leq .99 \tag{18}$$

which does not admit values of unit modulus not equal to unity. The iteration process (Eq. (16)) is used in RQA-1 for discrete-time models.

2.4 UNIQUE CONVERGENCE

Under certain circumstances, a Markov chain may not have a unique equilibrium probability distribution. These are the cases in which the limiting probability distribution ($\lim_{t\to\infty} p_t$ or $\lim_{n\to\infty} p_n$) is dependent upon the initial distribution, p_0. It is in those circumstances that the RQA-1 algorithm will not con-

verge uniquely. It converges, rather, to an equilibrium distribution which is consistent with the initial iterate chosen.

This should cause no difficulty since such cases are rare in practice, and the user of RQA-1 will usually be aware of the dependence upon initial conditions through knowledge of his model. In general, a sufficient condition for the convergent iteration processes above to converge uniquely is that it be possible (with nonzero probability) to reach each state from every other state in a finite time. A transition matrix having this property (known as irreducibility) is called a primitive matrix. Necessary conditions for uniqueness are somewhat stronger.

If nonuniqueness of a solution is suspected, a simple technique can be used to test the model. The solution can be found by starting with an initial iterate having unit value for some (arbitrary) state and zero for all others. If the resulting solution has no states with a calculated zero probability, the solution is unique. If it has states with a calculated zero probability, then the solution process should be repeated using as an initial iterate a probability distribution which is nonzero wherever the previous solution was zero (within computational error) and zero wherever the previous solution was nonzero. If the new solution is within the calculational error of the previous solution, it is the unique solution. If not, there is no unique solution.

For more detail on the conditions leading to nonunique equilibrium distributions, Parzen,[6] Gantmacher,[4] or Chung[7] are useful references.

2.5 ESTIMATION OF ERROR

It can be shown that, if the elementary divisors of the subdominant eigenvalues λ_{r+1} of a stochastic matrix G are of a degree m which is small compared to n, and if n is sufficiently large, then the n-th iterate can be approximated roughly by

$$p^{(n)} \triangleq G^n p^{(0)} \approx \pi + \lambda_{r+1}^n n^{m-1} x \qquad (19)$$

where x is a constant vector and π is the convergent solution. Also, it can be shown that the difference between successive iterates can be approximated roughly by

$$p^{(n+1)} - p^{(n)} \approx \lambda_{r+1}^n (\lambda_{r+1} - 1) n^{m-1} x \, . \qquad (20)$$

A fairly crude, but useful, estimate $\hat{\lambda}_{r+1}$ of $|\lambda_{r+1}|$ can be obtained from

$$\hat{\lambda}_{r+1} \triangleq \left| p^{(n+1)} - p^{(n)} \right|^{\frac{1}{n}} \approx |\lambda_{r+1}| \; |\lambda_{r+1} - 1|^{\frac{1}{n}} \; |n^{\frac{m-1}{n}}|^{\frac{1}{n}} \quad x \quad | \qquad (21)$$

since, for n sufficiently large, all but the first multiplicative term (λ_{r+1}) approach unity. Using this estimate, it can be argued from Eq. (20) that

$$\frac{p^{(n+1)} - p^{(n)}}{1 - \hat{\lambda}_{r+1}} \approx \lambda_{r+1}^{n} \; n^{m-1} \; x \qquad (22)$$

which is an approximation to the error term in Eq. (19).

Provided that the $\hat{\lambda}_{r+1}$ calculated is less than unity and greater than about .5, $(\lambda_{r+1} - 1)^{1/R}$ can be assumed to be near enough to unity, and the estimate so provided is, at the very least, a more realistic estimate of error than $(p^{(n+1)} - p^{(n)})$.

2.6 REPRESENTATION OF MATRICES

In order to store information completely describing matrices A or Q, with A or Q having the requisite high degree, a special approach is required. Moreover, because it is desirable to have the matrix elements expressed as algebraic functions of a set of parameters which may assume different values at different executions of the program, this approach must allow expression of the matrix elements in a literal form in the program. More will be said about the latter requirement in Section 3. For now, we will consider the problem of compressing the information needed to describe A or Q in such a way that computations are not appreciably slowed by the decoding required.

Clearly, a 5,000-degree matrix, when stored as a two-index array, requires 25,000,000 locations of storage which is unreasonable for a "fast" program. However, both A and Q are generally sparse matrices (have mostly zero-valued elements) and will usually have a high degree of repetition of equal element values. Hence, a scheme of storage which lists location information along with value information is a plausible starting point.

We construct a set of four vectors, together called a transition table, which implicitly define the matrix. Let us call them α, β, γ, and B* and denote their i-th elements by α_i, β_i, γ_i, and B_i, respectively. The quadruple $(\alpha_i, \beta_i, \gamma_i, B_i)$ specifies one or more elements of a matrix in the following manner.

---

*In the program listing, these correspond to the variables named "TRANS," "FIRST," "DEST," and "LAST," respectively.

The value of the element is α_i, and its matrix co-ordinates are (β_i, γ_i). Due to the repetitive nature of these matrices, the element α_i may occur in other locations of the matrix with co-ordinates $(\beta_i + r\delta, \gamma_i + r\delta)$ where δ is a constant (fixed throughout the transition table) and r takes values $0, 1, 2, \ldots, (B_i - \beta_i)/\delta$. In other words, the quadruple $(\alpha_i, \beta_i, \gamma_i, B_i)$ specifies the occurrence in the matrix of an element with value α_i at co-ordinates (β_i, γ_i), $(\beta_i + \delta, \gamma_i + \delta), \ldots, (B_i, \gamma_i + (B_i - \beta_i))$. This set of co-ordinates will be called the range of the quadruple. The set $\{\beta_i, \beta_i + \delta, \ldots, B_i\}$ will be called the β-range of the quadruple. The constant δ will be called the period of the table.

The choice of a value for δ will be discussed in detail in Section 3. Also in Section 3, we will carry out the construction of transition tables for three systems of varying degrees of complexity. These systems will be characterized as consisting of a single dominant queueing facility and a complex service facility. The degree of the Q-matrix for such a system will depend on the maximum number of jobs allowed in the queue. This number will be a parameter in the transition table, and, in this way, a transition table of fixed size can represent a Q matrix of arbitrary order.

2.7 TIMING

Each iteration (Eqs. (12) or (16)) is carried out in the following sequence:

(1) The vector to contain p^{n+1} is initially zeroed.

(2) α_1 is multiplied by $p^{(n)}_{\beta_1}$ and accumulated into $p^{(n+1)}_{\gamma_1}$.

(3) α_1 is multiplied by $p^{(n)}_{\beta_1+\delta}$ and accumulated into $p^{(n+1)}_{\gamma_1+\delta}$ if $\beta_i + \delta \leq B_i$.

(4) Step (3) is repeated until the range of the quadruple is exhausted.

(5) Steps (2) through (4) are repeated using α_i, β_i γ_i, B for $i = 2, 3, \ldots$ until all quadruples have been treated.

As a result of this procedure, the number of multiplications per iteration is equal to the number of nonzero elements in the transition matrix. In general, if the average number of different events which can occur at any time is E and the number of states is S, then the number of multiplications per iteration is ES. The convergence error after n iterates is $O(\lambda^n_{r+1})$. Thus, if ϵ is the desired error, the number of iterations needed will be $I \approx \log\epsilon/\log|\lambda_{r+1}|$, and the total number of multiplications required in the main loop is

$$M = ES \log\epsilon / \log|\lambda_{r+1}| \ . \tag{23}$$

In contrast, a rule of thumb estimate for the standard deviation of a measurement of a probability p in a continuous-time Markov chain using simulation is

$$S \approx \frac{2p}{|\gamma|T} \tag{24}$$

where T is the length of the simulation (in system-time) and γ is the nonzero eigenvalue of the matrix Q having smallest modulus. For ϵ to be equal to two standard deviations, this means that T should be

$$T = \frac{8p}{|\gamma|\epsilon^2} \ , \tag{25}$$

and the number of random numbers generated would need to be

$$K = \frac{8 \ v \ p}{|\gamma|\epsilon^2} \tag{26}$$

where v is the mean rate of occurrence of events.

It can be shown that $|\gamma|/v$ is usually of the order of $-\log|\lambda_{r+1}|$ which, in turn, is normally much smaller than unity. Hence, a relatively crude measure of the computation time of simulation and RQA-1 calculation is obtained from Eqs. (23) and (26) by approximating the ratio K/M by

$$\frac{K}{M} = O\left(\frac{8p}{\epsilon^2 \log\epsilon} \cdot \frac{1}{ES}\right) \ . \tag{27}$$

Since $\epsilon^2 \log\epsilon$ is very small for $\epsilon < .01$,

ϵ	$\frac{1}{\epsilon^2 \log \epsilon}$
.1	43.5
.01	2,170
.001	145,000 ,

we conclude that RQA-1 has a large advantage over simulation if desired accuracy is moderately high. This conclusion is strengthened considerably by the fact that a much greater amount of "overhead" calculation is required in simulations than with the RQA-1 algorithm.

3. EXAMPLES OF TRANSITION TABLES

The preparation of the transition table is the most technical aspect of RQA-1 use. As stated in the previous section, the aim is to code a representation for a transition matrix which has the flexibility to include a wide class of problems and which makes efficient use of storage. The general format of the transition table has already been discussed. The purpose of this section is to illustrate the procedure by which the set of quadruples can be developed when a statement of the control rules and the structure of a Markovian queueing system is given. The examples are obviously not exhaustive, but it is hoped that they are sufficiently representative and that a user of RQA-1 familiar with them will have a minimum of trouble deriving transition tables for a wide variety of problems. Future reports will expand upon the process of modeling general, large queueing systems by Markov chains and by RQA-1 transition tables.

The procedure illustrated here is capable of being mechanized by computer programs to some degree once the principles are understood. In general, however, each queueing model represents a distinct programming problem, and it appears difficult to generalize to automatic techniques. For this reason, the present report will limit itself to essentials. More sophisticated general methods are the object of current research by the Laboratory.

All three examples discussed in this section represent continuous-time Markov chains containing one main queue of jobs to be served by a service facility. In each succeeding example, the service facility has an increasingly complex structure. The derived transition table in each case represents a transition intensity matrix Q.

The pattern to be followed in transcribing a Markovian model for RQA-1 is:

(1) Determine the state space and a suitable mapping of the states (which are almost always vectors) to a set of integers.

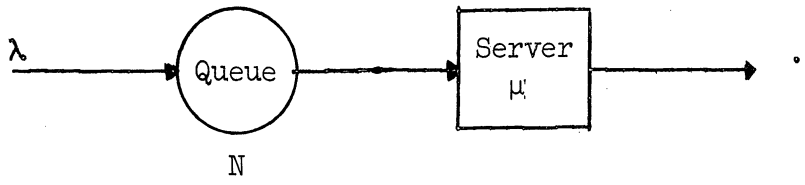
(2) Determine the most useful value of δ. This will generally be the value of δ which allows the transitions to be described with the least number of quadruples. The choice of state-space mapping strongly influences what is possible here, and steps (1) and (2) may have to be attempted several times for a complex model. Computation time does not depend upon the number of quadruples used. Thus, the primary necessity here is to remain within available storage where space for only 2,000 quadruples is allotted. A secondary necessity is to keep the transition table preparation from becoming unnecessarily unwiedly.

(3) Choose a sequence for generating the quadruples. This sequence must insure that every possible event of the model is considered for every

possible state of the model and that no event is considered more than once for any state. It must also insure that, when an event-state pair is considered, it is incorporated by extending the range of an already prepared quadruple whenever possible rather than creating a new quadruple.

3.1 SIMPLE SERVER SYSTEM

The first and simplest example is illustrated in the diagram below:



Jobs arrive as a Poisson process with mean rate λ. The duration of job processing time is independent of arrival and is exponentially distributed with mean time $1/\mu$. Jobs are executed on a first-come first-served basis. If more than N jobs are present in the queue, arrivals are simply rejected.

The first step is to define the state space for the system. Because of the simplicity of the example, the definition of state is artificial in this instance but provides a pattern for future use. The system state is defined by the number of jobs in the waiting line and by whether or not the service facility is occupied. Thus, a state is designated by the ordered pair (i,j) where i takes values in the set $I = \{0,1,2,\ldots,N\}$ and j takes values in the set $J = \{0,1\}$. In this case, the cardinality C of the set J is 2.

It is apparent that not all ordered pairs (i,j) in the Cartesian product I x J represent states. In fact, the set $\{(i,0) \mid i > 0\}$ is not a set of states, since, physically, a queue can exist only when there is a job in service. In general, the Cartesian product I x J can be partitioned into two sets: the set of states of the system, and the set of ordered pairs which are not states which we will call points. Thus, for this example, the set of states of the system is $\{(0,0), (0,1), (1,1), (2,1),\ldots,(N,1)\}$, and the set of points is $\{(1,0), (2,0),\ldots,(N,0)\}$.

As stated previously, the state description within the program is one-dimensional. A natural mapping, T, from the two-dimensional description (i,j) to the single dimension, k, is

$$k = T(i,j) = Ci + j + 1 . \qquad (28)$$

Thus, the one-dimensional state description k runs from k = 1 corresponding to (i,j) = (0,0) to k = C(N + 1) corresponding to (i,j) = (N,C - 1) which is equal to (N,1), in this case. In this example, the set of states is {1,2,4,6,...,2N + 2}.

At any time, the system can change state due to a job arrival or a service completion or can remain in the same state if neither of these occurs. The transition table can be partitioned into three classes of quadruples: those associated with job arrivals, those associated with service completions, and those associated with no change, i.e.,

(1) Job arrivals,
(2) service completions, and
(3) no change.

We can subdivide each of the above classes by the first dimension

(a) i = 0;
(b) i = 1;
(c) i = 2, etc.,

and further subdivide them by the second dimension

(i) j = 0
(ii) j = 1 .

A subclass is interpreted as a collection of transitions from a particular state caused by a particular event. For example, subclass (1) (b) (ii) represents the transitions from state (1,1) caused by job arrivals.

One can then create quadruples of the form (α,β,γ,B) by sequencing through each subclass, making certain that the transition rate associated with each subclass corresponds to the α of a quadruple, and that the state associated with the subclass is in the β-range of the same quadruple (see Section 2.4). The state corresponding to the destination of the transition must be the γ corresponding to that β. The following detailed construction should clarify the above discussion.

In this example, δ, the period of the transition table, is chosen to be equal to 2, since, as will be seen, the states to which equal rates apply differ most often by 2. It will usually be the case, when the model contains a main queue, that δ is most conveniently chosen to be equal to C, the cardinality of the set J of all possible combinations of values of all state variables other than the queue length. However, in general, δ may be any other value (constant throughout the transition table) which allows the matrix to be described with a smaller number of quadruples.

Beginning with the subclass (1) (a) (i), the first two parts of a quadruple can immediately be written,

$$\alpha = \lambda$$

$$\beta = 1 .$$

The fact that $\alpha = \lambda$ is obvious since class (1) is concerned with arrivals and jobs arrive at the system with a rate λ. $\beta = 1$ is the result of applying the mapping T to the state designated by (a) (i), namely (0,0). To determine γ, we note that an arrival in state (0,0) produces a transition to state (0,1) and, therefore, $\gamma = T(0,1) = 2$. To determine B, assume $B = r\delta + \beta$ for $r \neq 0$; then if λ applies for the transition

$$1 \rightarrow 2 ,$$

it should next apply for the transition

$$1 + \delta \rightarrow 2 + \delta$$

$$\text{i.e.,} \qquad 3 \rightarrow 4 .$$

But in the listing of states, 3 does not occur and hence $B = \beta = 1$. Thus, the first quadruple $(\lambda,1,2,1)$ applies only for the transition from state 1 to state 2 and, therefore, refers only to state 1, or subclass (1) (a) (i).

The next subclass is (1) (a) (ii). We can immediately write

$$\alpha = \lambda$$

$$\beta = T(0,1) = 2 .$$

An arrival in state (0,1) produces a transition to state (1,1) and, therefore, $\gamma = T(1,1) = 4$. Again, assume $B = r\delta + \beta$ for $r \neq 0$; then λ will apply for the transition

$$2 + \delta \rightarrow 4 + \delta$$

$$\text{i.e.,} \qquad 4 \rightarrow 6 .$$

To determine B, it is necessary to find the maximum r such that

$$r\,\delta + \gamma \leq \delta(N + 1)$$

and such that the corresponding transition is a correct description of an arrival, i.e., in this case, $r = N + 1 - 4/2 = N - 1$. In general, the maximum r is

desired for which the transition intensity α for states $\beta + r\delta$ to $\gamma + r\delta$ is still valid. For all the examples to be treated, this will correspond to the maximum r for which $\gamma + r\delta$ is in the state space. Therefore, $B = 2(N-1)+2 = 2N$. Thus, the second quadruple $(\lambda,2,4,2N)$ has the range $\{(2,4),(4,6),\ldots,(2N,2N+2)\}$ and, therefore, refers to the subclasses (1) (a) (ii), (1) (b) (ii), (1) (c) (ii), etc. Since (b) (i), (c) (i), etc., are not states, all the subclasses of class (1) have been referred to. A similar procedure now begins for class (2).

The β for the quadruple associated with subdivision (2) (a) (i) would be 1, but, since there can be no service completions in the 1 state, the corresponding α is zero and requires no quadruple. We next turn to (2) (a) (ii) for which $\beta = 2$ and $\alpha = \mu$, the service rate. A service completion in state 2 produces a transition to state 1, i.e., $\gamma = 1$. Assuming $B = r\delta + \beta$, for $r \neq 0$, the next transition should be

$$4 \rightarrow 3 .$$

But this is impossible since a completion does not take state 4 into state 3 (which in fact, is not a state). Hence, the resulting quadruple is $(\mu,2,1,2)$ and refers only to subclass (2) (a) (ii).

Since (b) (i), (c) (i), etc., are not states, the next subclass is (2) (b) (ii) for which $\alpha = \mu$ and $\beta = 4$. The corresponding γ is 2. B will be determined from the maximum r such that

$$r\delta + \gamma \leq \delta (N + 1)$$

$$\text{i.e.,} \quad r = N + 1 - 2/2 = N .$$

Therefore, $B = r\delta + \beta = 2N + 2$. The quadruple $(\mu,4,2,2N + 2)$ has the range $\{(4,2),(6,4),\ldots,(2N + 2, 2N)\}$ which refers to subclasses (2) (b) (ii), (2) (c) (ii), etc., and thereby completes class (2).

The quadruples for class (3) can easily be constructed by using the property of Q that the row sums are zero. For all these quadruples, it is also the case that $\beta = \gamma$. For class (3) (a) (i), it can be readily determined that

$$\alpha = -\lambda$$

$$\beta = 1$$

$$\gamma = 1$$

since the only way to leave state 1 is through an arrival with transition intensity λ. Thus, the practical rule for determing the α's of quadruples for

class (3) is that α equals minus the sum of all transition intensities associated with leaving that particular state. For the above quadruple, it is apparent that B = 1 since state 1 is the only state for which there will be no change if there are no arrivals.

The next subclass is (3) (a) (ii) for which we write

$$\alpha = -(\lambda + \mu)$$

$$\beta = 2$$

$$\gamma = 2 .$$

The fact that $\alpha = -(\lambda + \mu)$ is an illustration of the above rule where the sum of the transition intensities is $\lambda + \mu$. This quadruple will refer to all states which have the same properties as state 2, i.e., states 4,6,...,2N. Hence, B is 2N, and the quadruple is $(-(\lambda + \mu), 2,2,2N)$.

For the final state, 2N + 2, the only transition intensity associated with leaving is μ; therefore, the necessary quadruple is $(-\mu,2N + 2, 2N + 2, 2N + 2)$.

This completes class (3), thereby completing references to all subclasses. The transition table is listed in its entirety in Table I. We will now go on to consider a more complex problem in considerably less detail.

TABLE I

TRANSITION TABLE—"SIMPLE SERVER" MODEL

	α	β	γ	B
Job	λ	1	2	1
Arrivals	λ	2	4	2N
Service	μ	2	1	2
Completions	μ	4	2	2N + 2
No	$-\lambda$	1	1	1
Changes	$-\lambda-\mu$	2	2	2N
	$-\mu$	2N + 2	2N + 2	2N + 2

3.2 FIXED-SIZE MEMORY-SHARED COMPUTER MODEL ("FIXMEM" MODEL)

This model was discussed in the paper by Fife and Rosenberg.[8] It is depicted in Fig. 1.

Jobs arrive at the loading queue as a Poisson process with mean rate λ. Each job finds some number n of jobs in queue ahead of it. The queue is cleared on a first-come first-served basis, and the job enters the first stage of "service" where the mean rate of loading of jobs is μ_1 and the number of jobs being loaded is n_1(n_1 = 0 or 1). If more than N jobs are in queue, the arriving job is rejected.

After loading is completed, a job enters the second state of service, the processing phase. There are 5 "channels" in this stage, and n_2 is the number which are occupied. The mean rate at which jobs are processed in a single block or channel is μ_2. A job leaves the system when its processing time ends.

The principal complexity of the model is introduced by the following constraint:

$$n_1 + n_2 \leq 5 \ .$$

This implies that loading for no job can be begin while all five blocks are processing jobs.

The state description consists of the ordered triple (n,n_1,n_2) where n takes values $0,1,\ldots,N$, n_1 takes values zero or 1, and n_2 takes values $0,1,2,3,4,5$, with the restriction that $n_1 + n_2 \leq 5$. Table II lists all the 11 possible occupancy configurations (n_1,n_2) and associates a value j to each.

TABLE II

INTERNAL-STATE DESIGNATION—"FIXMEM" MODEL

n_1	n_2	Internal State j
0	0	1
0	1	2
0	2	3
0	3	4
0	4	5
0	5	6
1	0	7
1	1	8
1	2	9
1	3	10
1	4	11

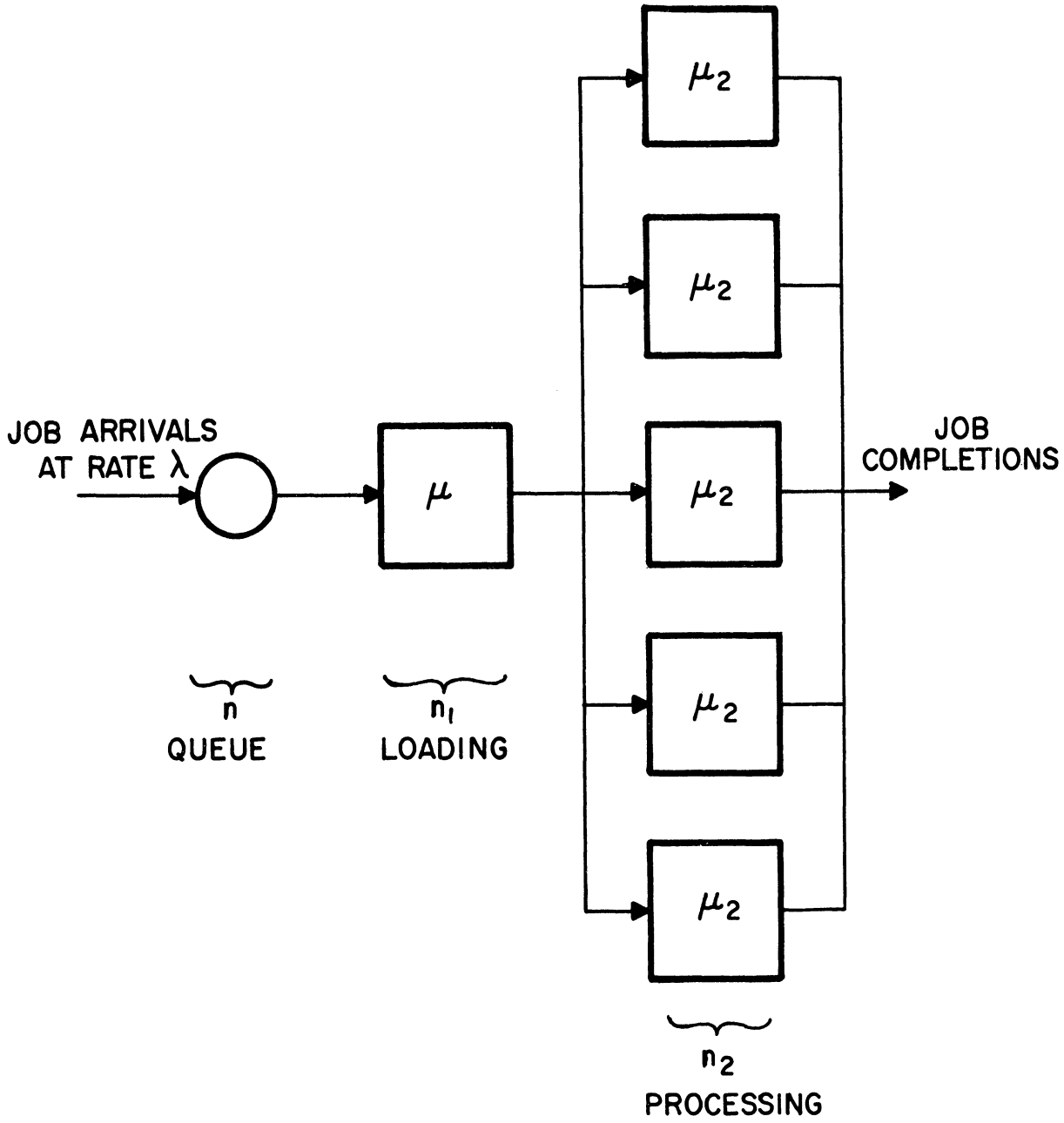


Fig. 1. Fixed-size memory-shared computer model.

The values of j are here called "internal states." Thus, for this example, we have the set $I = \{1,2,\ldots,N\}$ and the set $J = \{1,2,\ldots,11\}$ and, again, we take $\delta = C = 11$ where C is the cardinality of J. The mapping T for this example reduces to

$$k = T(i,j) = 11\,i + j \ .$$

The set of "points" is $\{(i,j) \mid i > 0,\ j = 1,2,3,4,5\}$.

Table III lists the 47 quadruples which are necessary to refer to all the subclasses of this system. There are more subclasses for the example than for the simple server system since there are 11 internal states rather than 2. We will not carry out the complete derivation of this table but, rather, we will consider a few illustrative cases.

We will derive the quadruples which refer to state (0,6) for all three classes. For class (1), we can write

$$\alpha = \lambda$$

$$\beta = 6 \ .$$

An arrival at state 6 will remain in the queue resulting in state (1,6), i.e., state 17. Thus $\gamma = 17$. It is obvious that later arrivals will also remain in the queue. Thus, we are looking for a $B = w\delta + \beta$ such that $w\delta + \gamma \leq \delta(N+1)$ where w is the largest integer for which this inequality holds. The above yields

$$w \leq N + 1 - 17/11 \ .$$

Therefore,

$$w = N - 1 \ .$$

This results in $B = 11(N-1) + 6$. The quadruple $(\lambda,6,17,11(N-1) + 6)$, which is number (6) in the listing, has the range $\{(6,17),(17,28),\ldots,(11(N-1) + 6,11N + 6)\}$.

For class (2), we write

$$\alpha = 5\mu_2$$

$$\beta = 6$$

since state 6 represents the configuration in which all 5 processing blocks are occupied, and jobs are processed at a mean rate of μ_1 per block, the rate for

TABLE III

"FIXMEM" MODEL QUADRUPLES

No.	α	β	γ	B
		(a) Job Arrivals		
1	λ	1	7	1
2	λ	2	8	2
3	λ	3	9	3
4	λ	4	10	4
5	λ	5	11	5
6	λ	6	17	11(N-1) + 6
7	λ	7	18	11(N-1) + 7
8	λ	8	19	11(N-1) + 8
9	λ	9	20	11(N-1) + 9
10	λ	10	21	11(N-1) + 10
11	λ	11	22	11(N-1) + 11
		(b) Service Completions		
12	μ_2	2	1	2
13	$2\mu_2$	3	2	3
14	$3\mu_2$	4	3	4
15	$4\mu_2$	5	4	5
16	$5\mu_2$	6	5	6
17	$5\mu_2$	17	11	11N + 6
18	μ_1	7	2	7
19	μ_1	18	8	11N + 7
20	μ_2	8	7	11N + 8
21	μ_1	8	3	8
22	μ_1	19	9	11N + 8
23	$2\mu_2$	9	8	11N + 9
24	μ_1	9	4	9
25	μ_1	20	10	11N + 9
26	$3\mu_2$	10	9	11N + 10
27	μ_1	10	5	10
28	μ_1	21	11	11N + 10
29	$4\mu_2$	11	10	11N + 11
30	μ_1	11	6	11N + 11
		(c) No Changes		
31	$-\lambda$	1	1	1
32	$-\lambda-\mu_2$	2	2	2
33	$-\lambda-2\mu_2$	3	3	3
34	$-\lambda-3\mu_2$	4	4	4
35	$-\lambda-4\mu_2$	5	5	5
36	$-\lambda-5\mu_2$	6	6	11(N-1) + 6
37	$-5\mu_2$	11N + 6	11N + 6	11N + 6
38	$-\lambda-\mu_1$	7	7	11(N-1) + 7
39	$-\mu_1$	11N + 7	11N + 7	11N + 7
40	$-\lambda-\mu_1-\mu_2$	8	8	11(N-1) + 8
41	$-\mu_1-\mu_2$	11N + 8	11N + 8	11N + 8
42	$-\lambda-\mu_1-2\mu_2$	9	9	11(N-1) + 9
43	$-\mu_1-2\mu_2$	11N + 9	11N + 9	11N + 9
44	$-\lambda-\mu_1-3\mu_2$	10	10	11(N-1) + 10
45	$-\mu_1-3\mu_2$	11N + 10	11N + 10	11N + 10
46	$-\lambda-\mu_1-4\mu_2$	11	11	11(N-1) + 11
47	$-\mu_1-4\mu_2$	11N + 11	11N + 11	11N + 11

ℓ blocks in use, where $\ell \leq 5$ is $\ell\mu_1$. In this instance, $\ell = 5$, and, therefore, $\alpha = 5\mu_2$. A single service completion will result in a state in which $\ell = 4$, and, therefore, $\gamma = 5$. If there is a queue, then a service completion in the processing block will result in the job at the head of the queue moving into the loading phase. Thus, two quadruples, namely $(5\mu_2,6,5,6)$ and $(5\mu_2,17,11,11N+6)$, will be necessary to form the range $\{(6,5),(17,11),(28,22),\ldots,(11N + 6, 11N)\}$.

Class (3) will require two quadruples for the same range: for the first $\alpha = -(\lambda + 5\mu_2)$, and for the second $\alpha = -5\mu_2$. These quadruples are numbered (36) and (37), respectively. For the states $6,17,28,\ldots,11(N-1) + 6$, transitions are possible in two ways: an arrival with transition intensity λ, and a service completion with transition intensity $5\mu_2$. For state $11N + 6$, a transition is only possible through a service completion.

For a final case, we will derive the quadruples which refer to state (0,9) for all three classes. State 9 is the service configuration consisting of 2 jobs being processed and 1 job being loaded. Thus, there are two ways for service completions to occur.

The quadruple associated with class (1) is number (9) in the listing, and, at this point, its derivation should be straightforward.

For class (2), the relevant quadruples are numbered (23),(24) and (25). The α for quadruple (23) is $2\mu_2$ which is the transition intensity associated with a service completion in the processing blocks. The range of this quadruple is $\{(9,8),(20,19),\ldots,(11N + 9, 11N + 8)\}$. The service configuration for state 8 is shown in Table II. There are two quadruples necessary to describe service completions in the loading block. When there are no jobs in the queue, a service completion in the loading block results in an additional processing block being employed, i.e., state 4 results. When there are jobs in the queue, the job at the head of the queue moves into the recently vacated loading block and state 10 results.

The class (3) quadruples are numbered (42) and (43). The range of quadruple (42) is $\{(9,9), (20,20),\ldots,(11(N-1) + 9, 11(N-1) + 9)\}$, and its α is $-(\lambda + \mu_1 + 2\mu_2)$, as expected. Quadruple (42) refers to state $11N + 9$ with an α of $-(\mu_1 + 2\mu_2)$.

These few cases should be sufficient to indicate the structure of the entire transition table. The next example will introduce an additional complexity in that service completions will depend more closely upon the nature of job arrivals.

3.3 RANDOM-SIZE MEMORY-SHARED COMPUTER MODEL ("RANMEM" MODEL)*

The model depicted in Fig. 2 bears a resemblance to the previous model discussed, but it is only a superficial one. As before, the queue will be cleared on a first-come first-served basis. There will be three types of jobs arriving, distinguishable by the number of service blocks or storage units they require. The jobs will arrive as a Poisson process with mean arrival rates λ_1, λ_2, λ_3 for one, two, and three storage unit jobs, respectively. These rates are independent of time and state of the system. For convenience, we set $\lambda = \lambda_1 + \lambda_2 + \lambda_3$.

We assume that the system is nonpreemptive, i.e., a job once begun is completed. Each type of job will have a basic mean service rate when it is being processed alone. These service rates are μ_1, μ_2, μ_3 for one, two, and three storage unit jobs, respectively. However, when two jobs are being processed simultaneously, the mean service rate at which each job is completed is assumed to be one-half the basic mean service rate. For example, if a one-unit and a two-unit job are being processed, the mean service rate for the one-unit job is $(1/2)\mu_1$ and for the two-unit job $(1/2)\mu_2$. Similarly, for three one-unit jobs, each will have a mean service rate of $(1/3)\mu_1$. Upon completion of any type of job, all blocks being used for that job become free simultaneously.

This model is particularly interesting because it requires the evaluation of conditional probabilities (conditional upon the state of the system) in order to determine the correct transition intensities. This is true because jobs having different properties are allowed to be mixed in the queue, and the nature of the first job in the queue is dependent statistically upon the condition of the system.

The state description consists of the ordered quadruple (n,n_1,n_2,n_3) where n is the number of jobs in the waiting line, and n_1,n_2,n_3 are the numbers of one-, two-, and three-unit jobs in service, respectively. Table IV lists all

TABLE IV

INTERNAL STATE DESIGNATION—"RANMEM" MODEL

n_1	n_2	n_3	Internal State, j
0	0	0	1
0	0	1	2
0	1	0	3
1	1	0	4
1	0	0	5
2	0	0	6
3	0	0	7

*This model corresponds to one suggested by Dr. R. V. Evans, now of UCLA, for multiprogrammed time-shared computer operation.[9]

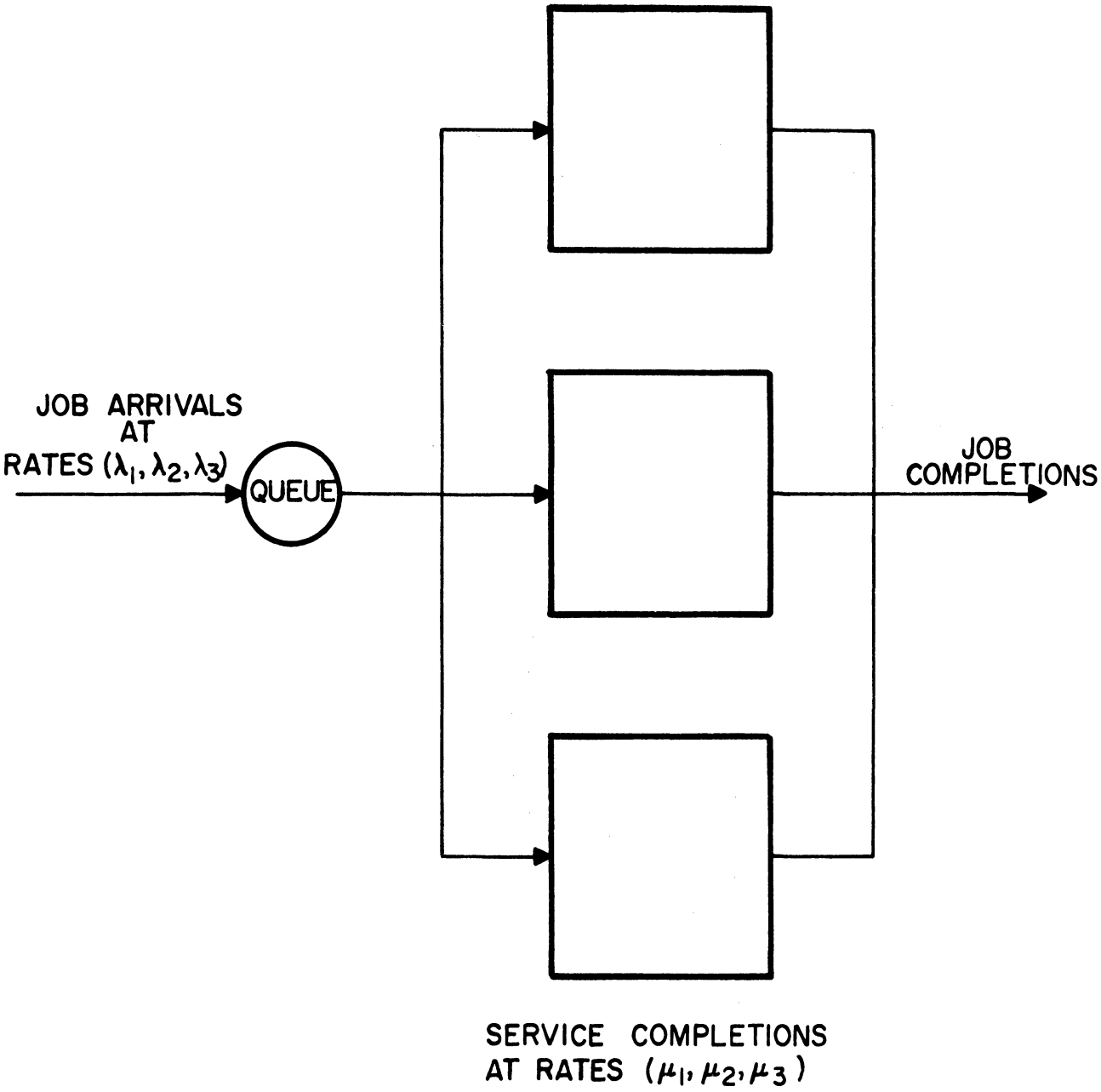


Fig. 2. Random-size memory-shared computer model (RANMEM)

the possible occupancy configurations of which there are 7. For this example, the set J is $\{1,2,\ldots,7\}$ with $\delta = C = 7$, and the set I is $\{0,1,\ldots,N\}$. The mapping T reduces to $k = T(i,j) = 7\,i + j$. We will again call the value of j the "internal state" of the system. The set of "points" is $\{(i,j) \mid i > 0, j = 1\}$.

Table V lists the 60 quadruples which are necessary to refer to all the subclasses of the state space. We will consider only one illustrative case.

We will derive the quadruples which refer to state $(n,3), n = 0,1,\ldots,N$, for all three classes. State 3 represents a two-unit job in service. The arrival of a one-unit job with transition intensity λ_1 produces a transition to state 4. The arrivals of either a two-unit or a three-unit job with intensity $\lambda_2 + \lambda_3$ do not change the internal state but do increase the queue length. Both these quadruples (5) and (6) refer only to the single state 3, since for state 10, the arrival of any type of job (with intensity $\lambda = \lambda_1 + \lambda_2 + \lambda_3$) produces a transition to state 17. Then quadruple (14) will have the range $\{(10,17),(17,24),\ldots,(7(N-1) + 3,\ 7N + 3)\}$. Thus, three quadruples are necessary for the class (1), subclasses $(n,3)$, $n = 0,1,\ldots,(N)$.

In state 3, a service completion will produce a transition to state 1 with intensity μ_2. Quadruple (18) will have only the single element (3,1) in the range, because the transitions resulting from service completion of a two-unit job are quite different for state 10. In state 10, the single job in queue can only be a two-unit or a three-unit job (a one-unit job cannot remain in the queue when there is space for it in service. If, for example, only a two-unit job in service, the one-unit job cannot remain in the queue).

We have here the situation in which the identity of the first job in the queue depends partially on the type of job in service. For example, if there is a two-unit job in service, the first job in the queue cannot be a one-unit job. If there is a one-unit job in service, the first in queue cannot be a one-unit or a two-unit job but must be a three-unit job. This dependency will introduce complexities into the form of the transition intensities. If there is a two-unit job in the queue, a service completion will result in a transition with intensity $\mu_2\lambda_2/(\lambda_2 + \lambda_3)$; similarly, for a three-unit job in the queue, the transition intensity will be $\mu_2\lambda_3/(\lambda_2 + \lambda_3)$. In the first case, the $\lambda_2/(\lambda_2 + \lambda_3)$ factor is the probability of a two-unit job being in queue given that there is a two-unit or a three-unit job in queue.

Quadruples (27) and (28) account for these cases. Quadruple (2) has a complete range because, given a three-unit job at the head of the queue, the remaining composition of the queue is unimportant. But quadruple (27) has only the single element (10,3) in its range, since transitions from state 17 depend on the nature of the first two jobs in the queue if the first in queue is a two-unit job.

The only two cases to consider for state 17 are that the first in queue is a two-unit job and the second in queue is not a one-unit job, and that the

TABLE V

"RANMEN" MODEL QUADRUPLES

No.	α	β	γ	B
		(a) Job Arrivals		
1	λ_1	1	5	1
2	λ_2	1	3	1
3	λ_3	1	2	1
4	λ	2	9	$7(N-1) + 2$
5	λ_1	3	4	3
6	$\lambda_2 + \lambda_3$	3	10	3
7	λ	4	11	$7(N-1) + 4$
8	λ_1	5	6	5
9	λ_2	5	4	5
10	λ_3	5	12	5
11	λ_1	6	7	6
12	$\lambda_2 + \lambda_3$	6	13	6
13	λ	7	14	$7(N-1) + 7$
14	λ	10	17	$7(N-1) + 3$
15	λ	12	19	$7(N-1) + 5$
16	λ	13	20	$7(N-1) + 6$
		(b) Service Completions		
17	μ_3	2	1	2
18	μ_2	3	1	3
19	$\mu_1/2$	4	3	4
20	$\mu_2/2$	4	5	4
21	μ_1	5	1	5
22	μ_1	6	5	6
23	μ_1	7	6	7
24	$\mu_3\lambda_3/\lambda$	9	2	$7N + 2$
25	$\mu_3\lambda_2/\lambda$	9	3	9
26	$\mu_3\lambda_1/\lambda$	9	5	9
27	$\mu_2\lambda_2/(\lambda_2 + \lambda_3)$	10	3	10
28	$\mu_2\lambda_3/(\lambda_2 + \lambda_3)$	10	2	$7N + 3$
29	$\mu_1(\lambda_3 + \lambda_2)/2\lambda$	11	10	$7N + 4$
30	$\mu_2\lambda_3/2\lambda$	11	12	$7N + 4$
31	$(\mu_1\lambda_1 + \mu_2\lambda_2)/2\lambda$	11	4	$7N + 4$
32	$\mu_2\lambda_1/2\lambda$	11	6	11
33	$\mu_1\lambda_3/(\lambda_2 + \lambda_3)$	13	12	$7N + 6$
34	$\mu_1\lambda_2/(\lambda_2 + \lambda_3)$	13	4	$7N + 6$
35	μ_1	12	2	$7N + 5$
36	$\mu_1(\lambda_2 + \lambda_3)/\lambda$	14	13	$7N + 7$
37	$\mu_1\lambda_1/\lambda$	14	7	$7N + 7$
38	$\mu_3\lambda_2(\lambda_2 + \lambda_3)/\lambda^2$	16	10	$7N + 2$
39	$2\mu_3\lambda_2\lambda_1/\lambda^2$	16	4	$7N + 2$
40	$\mu_3\lambda_1\lambda_3/\lambda^2$	16	12	$7N + 2$
41	$\mu_3\lambda_1^2/\lambda$	16	16	16
42	$\mu_2\lambda_2/\lambda$	17	10	$7N + 3$
43	$\mu_2\lambda_2\lambda_1/(\lambda(\lambda_2+\lambda_3))$	17	4	$7N + 3$
44	$\mu_2\lambda_1(\lambda_2 + \lambda_3)/2\lambda^2$	18	13	$7N + 4$
45	$\mu_2\lambda_1^2/2\lambda^2$	18	7	$7N + 4$
46	$\mu_3\lambda_1^2(\lambda_2 + \lambda_3)/\lambda_3$	23	13	$7N + 2$
47	$\mu_3\lambda_1^3/\lambda^3$	23	7	$7N + 2$
		(c) No Changes		
48	$-\lambda$	1	1	1
49	$-\mu_3$	$7N + 2$	$7N + 2$	$7N + 2$
50	$-\lambda-\mu_3$	2	2	$7(N-1) + 2$
51	$-\mu_2$	$7N + 3$	$7N + 3$	$7N + 3$
52	$-\lambda-\mu_2$	3	3	$7(N-1) + 3$
53	$-\mu_1/2-\mu_2/2$	$7N + 4$	$7N + 4$	$7N + 4$
54	$-\lambda-\mu_1/2-\mu_2/2$	4	4	$7(N-1) + 4$
55	$-\mu_1$	$7N + 5$	$7N + 5$	$7N + 5$
56	$-\lambda-\mu_1$	5	5	$7(N-1) + 5$
57	$-\mu_1$	$7N + 6$	$7N + 6$	$7N + 6$
58	$-\lambda-\mu_1$	6	6	$7(N-1) + 6$
59	$-\mu_1$	$7N + 7$	$7N + 7$	$7N + 7$
60	$-\lambda-\mu_1$	7	7	$7(N-1) + 7$

second in queue is a one-unit job. In the first case, the transition intensity will be

$$\mu_2\lambda_2/\lambda = \mu_2 \frac{\lambda_2}{\lambda_2 + \lambda_3} \quad \frac{\lambda_2 + \lambda_3}{\lambda} ,$$

i.e., μ_2 multiplied by the probability that the first job in queue is a two-unit job and the second is either a two-unit or a three-unit job. The transition will be made to state 10. The complete quadruple is number (42). In the second case, the transition intensity will be

$$\mu_2 \frac{\lambda_2}{\lambda_2 + \lambda_3} \quad \frac{\lambda_1}{\lambda} ,$$

i.e., μ_2 multiplied by the probability that the first job in queue is a two-unit job and the second is a one-unit job. Thus, after the service completion two jobs move into service resulting in state 4. Number (43) in Table Vb is the complete quadruple.

For this problem, it will be advisable to determine the class (3) quadruples by examiniation of those classes (1) and (2). The sum of the transition intensities referring to state 3 is $\mu_2 + \lambda_1 + \lambda_2 + \lambda_3$ resulting in the transition intensity $-(\lambda + \mu_2)$. To determine the range of the quadruple with this α, we will consider α's for states 10 and 17. For state 10, the following is the sum of the transition intensities.

$$\lambda + \mu_2\lambda_2/(\lambda_2 + \lambda_3) + \mu_2\lambda_3(\lambda_2 + \lambda_3) .$$

This also reduces to $\lambda + \mu_2$. For state 17, the following is the sum of the transition intensities:

$$\lambda + \mu_2\lambda_3/(\lambda_2 + \lambda_3) + \mu_2\lambda_2/\lambda + \mu_2\lambda_2\lambda_1/(\lambda(\lambda_2 + \lambda_3) .$$

This again reduces to $\lambda + \mu_2$. Thus, quadruple (52) will refer to all states except the final state. For the final state, the α will be simply $-\mu_2$ since quadruple (14) with $\alpha = \lambda$ does not refer to the final state.

A few additional observations may be useful. In quadruples (19) and (20), the transition intensities μ_1 and μ_2 are each divided by 2. This is a result of an assumption made earlier in which it was stated that, if two jobs were run simultaneously, the mean rate at which each would be completed would be 1/2 the basic mean rate. When two like jobs (i.e., one-unit jobs) are being run, the mean rate for each is $(1/2)\mu_1$ but the mean rate for a service completion will be the sum, namely μ_1. Quadruple (39) has a factor 2 associated with the transition intensity. This arises from the fact that two like transition intensities have been added in order to arrive at this result.

3.4 SUMMARY

It should be noted that all three transition tables have a common feature. Each represents a system with the size of the state space limited by the maximum number of allowed jobs in the queue. As we stated previously, this is only one kind of system for which a transition table may be usefully employed. Additional complexities can be introduced by having elements of the quadruples expressed as functions of some or all of the parameters of the queueing model.

As illustrated, the method for deriving a transition table provides an orderly procedure for determining a complete set of quadruples. What is actually done is to enumerate all events which can occur for each state. The division of events between arrivals and service completions in the examples is artificial and done only for convenience in ordering tasks. For problems in which some events cannot be identified as either an arrival or a service completion, some other division will be suitable. In any case, one will continue to enumerate the transitions for each event and each state. Of course, sequences other than those suggested in the examples could be followed in systematic evaluation of the quadruples.

In some instances, the construction of a partial state diagram will greatly aid the process of determining the transition table.[8] For complex systems, the state diagram may prove very difficult to draw and may not provide much insight into the regularities of the state space.

Once again, we stress the point that examples have been presented to suggest possible forms for transition tables. For most problems, it will be the user's responsibility to devise transition tables with those properties most useful to his special purposes.

4. PROGRAM DESCRIPTION

The purpose of this section is to discuss programming details of the RQA-1 program. For the most part, the block diagram (Fig. 3) and the MAD listings (Appendix A) of the program provide full documentation of the program. However, a few remarks will be made here to aid in their interpretation. We shall begin these remarks by discussing the roles of the major subroutines.

More detail of the procedures to be followed by a user, of definitions of user specified parameters, and of diagnostic printouts is provided in Section 5 which acts as a sort of "user's manual" for RQA-1.

4.1 THE DISCIPLINE SUBROUTINE (DISCPL)

The function of the Discipline Subroutine is to prepare the transition table which describes the transition probabilities or intensities of the problem to be solved. This subroutine is unique to each application or model solved by RQA-1, and hence, must be written by the user.

The form of the transition table internal to RQA-1 is a set of four vectors called TRANS, FIRST, DEST, and LAST corresponding to the α, β, γ, and B respectively, of the quadruples discussed in Section 3. The i-th element of each of the four vectors together form the i-th quadruple. The DISCPL subroutine is basically a program for the evaluation ofthese vectors in terms of parameters which may be altered at execution time. These parameters are the counterparts of the λ, μ, N, etc., of the examples in Section 3 and appear only in DISCPL.

In addition to preparing the transition table, DISCPL has the responsibility of reading all parameters supplied by the user at execution and of providing essential control information descriptive of the queueing model being analyzed.

Sample DISCPL routines for the three models used as examples in Section 3 can be found in Section 5 (Table VI, VII, and VIII).

4.2 STOCHASTIC CONVERSION SUBROUTINE (STOCAL)

The role of this subroutine is to determine whether the transition table in memory represents a transition intensity matrix Q (hence, a continuous-time model) or a stochastic matrix A (hence, a discrete-time model). If it

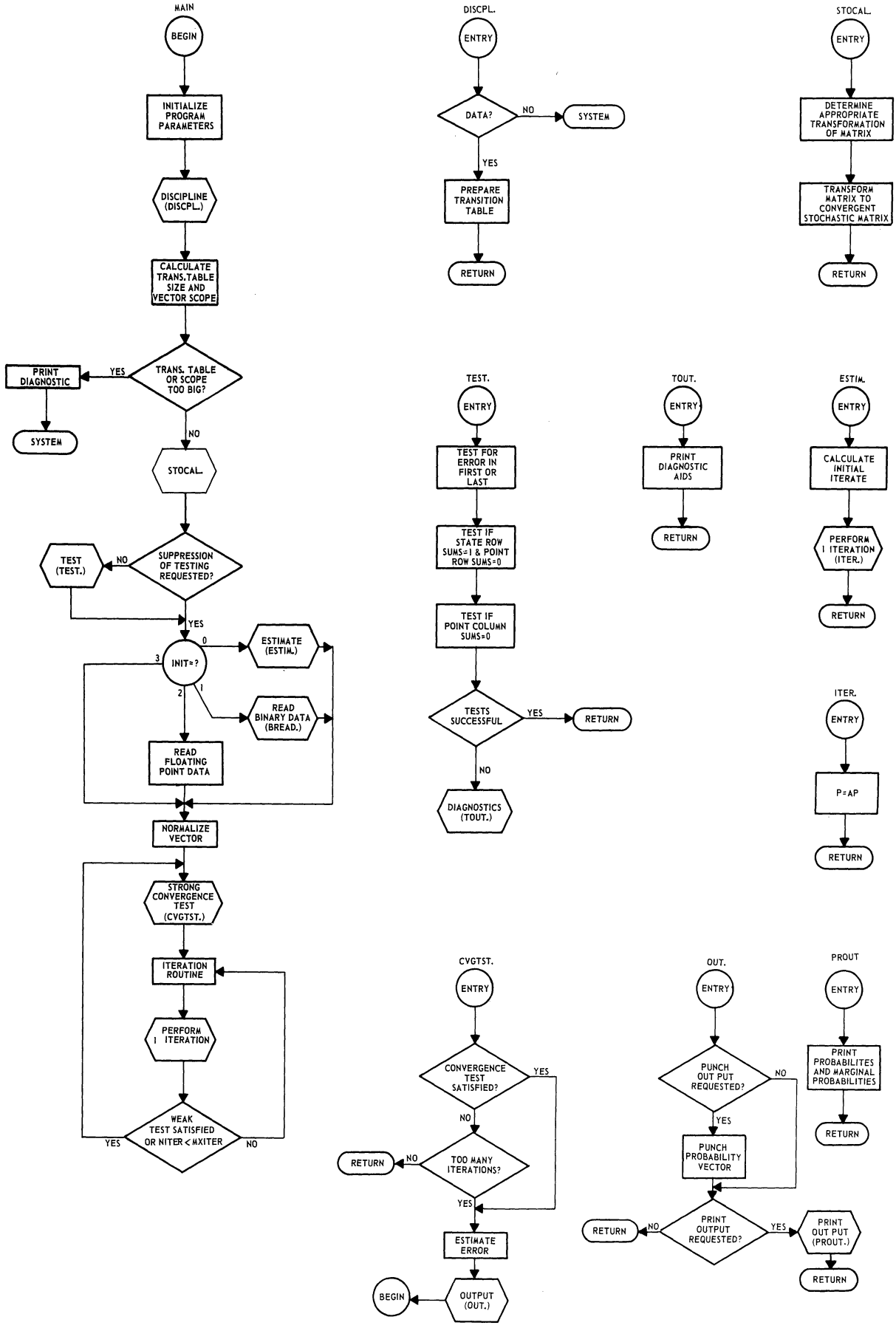


Fig. 3. Flow diagram for RQA-1.

is the former, STOCAL converts the transition table to represent the stochastic matrix((.99/R)Q + I), where $R = \max_i |q_{ii}|$. If it is the latter, STOCAL converts the transition table to represent the new stochastic matrix (.99A + .01I). These conversions assure the convergence of the resulting iteration processes to an equilibrium distribution of the specified model (cf., Sections 2.2 and 2.3).

The determination of the type of matrix represented is accomplished by determination of the element of the TRANS vector having least value. If it is equal to or greater than zero, the transition table represents a stochastic matrix. If, on the other hand, it is negative, the transition table represents a transition intensity matrix, and the value of the element having least value is equal to -R.

4.3 THE TEST SUBROUTINES TEST AND TOUT

The purpose of TEST is to determine whether the current transition table does indeed represent a stochastic matrix, and whether it adheres to the necessary properties of a valid transition table.

Using the transition table, the elements corresponding to each row of the matrix are summed and successively stored in a prezeroed vector. If the matrix is stochastic, the rows corresponding to states will have a one in the associated position of the vector and these corresponding to points will have a zero. In addition, all elements in columns corresponding to points must be zero. In the TEST subroutine, it is determined whether or not the above conditions exists. If they do not exist, the subroutine TOUT is called, and various diagnostics are printed to help determine the error(s) in the transition table.

For each quadruple in a valid transition table, the following equation must hold:

$$B - \beta = r\delta \qquad \text{for } r \in \{0,1,2,\ldots\} . \qquad (29)$$

If this condition is not met for all the quadruples of the transition table, the subroutine TOUT is called, and appropriate diagnostics are printed.

4.4 THE ESTIMATION SUBROUTINE ESTIM

Before beginning the iterations, an initial iterate must be selected. The RQA-1 main program allows this iterate to be selected in four different ways by the user: it can be read in as input data either in binary or floating point form, it can be taken to be the final iterate of the previous run (which has remained in storage), or it can be calculated by the subroutine called ESTIM.

As its name suggests, ESTIM is a subroutine for estimating a satisfactory initial iterate and is based on a generalization of the simple server system discussed in detail in Section 3.1. For a simple server model, the probability of queue length i is proportional to ρ^i, where $\rho = \lambda/\mu$ (e.g., ref. 10). ESTIM prepares an initial iterate so that the k-th element of the iterate is proportional to $(\rho^*)^k$, where ρ^* is a constant. If $G = \{g_{ij}\}$ is the matrix represented by the transition table, then ρ^* is calculated as

$$\rho^* = \left(\frac{\lambda^*}{\mu^*}\right)^{\frac{1}{\nu}} \tag{30}$$

where

$$\lambda^* = \sum_{i,j:\, i>j} g_{ij} \tag{31}$$

$$\mu^* = \sum_{i,j:\, i>j} g_{ij} \tag{32}$$

$$\nu = \frac{\sum_{\substack{i,j: \\ g_{ij} \neq 0}} |i - j|}{\eta} \tag{33}$$

and η is the total number of nonzero, off-diagonal elements of the matrix. For the simple queue, these expressions reduce to $\rho^* = \lambda/\mu$. The only real justification for applying this calculation to more complex models is that it has been found to be quite effective, particularly when the model being solved originally had a multivariate state space and the mapping to the linear state space had a major queue of the system as the most significant variable in a Cartesian mapping.

ESTIM calls the iteration subroutine once before returning to the calling program, producing a vector having nonzero values only in states. The vector supplied to the calling program will not have a unit sum. Hence, subsequent normalization by the calling program is required.

4.5 THE ITERATION SUBROUTINE, ITER

Whereas the main program and all the other subroutines are written in the procedure-oriented language, MAD, the ITER subroutine is written in the machine-oriented language, UMAP.* The purpose of this is to make this subroutine as efficient as possible.

The only argument of this subroutine is the number of iterations to be carried out. Thus, a typical call of the subroutine would be EXECUTE ITER.(N) where N is the number of iterations. One iteration consits of the multiplication of the current iterate by the matrix. The subroutine requires two vectors in erasable store: one for the last iterate, and one for the iterate being formed. A switch is set which indicates which is the current iterate, thus removing the necessity for continually transposing newly computed iterates.

Because of the importance of this subroutine, we would like to discuss its operation in more detail. The two vectors mentioned above are actually stored in one vector which is represented in the program by the symbol V. This vector has 10,000 locations, thereby allotting 5,000 to each vector. To indicate which half of V contains the current iterate and which half will be used to store the newly computed iterate, two symbols, VFLAG and CVFLAG, are defined in the program. These symbols can only assume the values 0 or 1 and are constrained to have complementary values, i.e., when VFLAG = 1, CVFLAG = 0.

The current iterate is always referred to by calling V(VFLAG,I). When VFLAG = 0, the first half of V contains the current iterate. When VFLAG = 1, the second half of V contains the current iterate. To illustrate how a typical iteration would proceed, assume that VFLAG = 1. This means that the current iterate occupies the second half of V. During the process of multiplying this vector by the matrix, values of the newly calculated iterate are stored in the first half of V which had been zeroed previously. When this single iteration is complete, VFLAG is set equal to 0, and the second half of V is zeroed. Thus, the current iterate occupies the first half of V and will be correctly referenced by calling V(VFLAG,I).

As mentioned above, the multiplication process is carried out by sequencing through the transition table. For a particular quadruple $(\alpha_i,\beta_i,\gamma_i,B_i)$, α_i is multiplied by the value of the current iterate at location β_i, and the result is added to the value at location γ_i of the iterate being formed. This process is continued for the range of the quadruple, i.e., until α_i is multiplied by the value of the current iterate at location B_i with the result being added to the value at location $\gamma_i + B_i - \beta_i$ of the iterate being

---

*For users who do not have a UMAP assembler available, a MAD version of ITER is also included in the listings of Appendix A. Its use in place of the UMAP version will approximately quadruple the execution time of RQA-1.

formed. The program then proceeds to $(\alpha_{i+1},\beta_{i+1},\gamma_{i+1},B_{i+1})$ and repeats the process until all quadruples are exhausted.

This type of vector-matrix multiplication is quite efficient because only nonzero elements of the matrix are referenced. Also, no significant penalty in computing time results from the manner of describing the matrix.

4.6 THE ITERATION STRATEGY (MAIN AND CVGTST)

Upon completion of the calculation and normalization of the initial iterate, the main program produces a first call to the convergence testing subroutine CVGTST. This first call of CVGTST simply sets the underflow traps so that underflows produce normal zeros, and returns to the calling program (MAIN) with an index corresponding to a nonzero element of the initial iterate.

The main program continues to iterate until successive values of the specified element have an absolute difference which is less than the convergence criterion ϵ, or until the specified maximum number of iterations has been taken. At that point CVGTST is called again.

If we define $p^{(n)}$ to be the n-th iterate and $p_i^{(n)}$ to be the value of the i-th element of the n-th iterate, the maximum difference between successive iterates $e^{(n)}$ is given by

$$e^{(n)} = \max_i \left| p_i^{(n)} - p_i^{(n-1)} \right| .$$

The iteration process is said to have converged after n iterations for error criterion ϵ if $e^{(n)} \leq \epsilon$. For every call of CVGTST after the first, CVGTST calculates $e^{(n)}$ and tests for convergence. If the process has not converged, this subroutine returns to the main program with the integer i_M corresponding to the state for which the absolute value between successive iterates is largest. The idea is not to calculate i_M for each iteration but to use an initially calculated i_M as a coarse test and to recalculate only when this coarse test is satisfied. Then the main program iterates until the absolute difference of the i_M-th element is less than ϵ or the specified number of iterations has been taken. Another call of CVGTST then results, and the process is repeated.

On the other hand, whenever the tests within CVGTST are satisfied, CVGTST prints appropriate comments and error estimates, resets the underflow error trap, calls the output subroutine OUT2, and transfers out (via an "error return") to the beginning of the main program for a calculation of another model (if data are available).

The purpose of this two-level testing procedure is to avoid the expense of calculating i_M for each iteration by using a value of i_M in a coarse test for convergence and recalculating i_M (using CVGTST) only when the coarse test has been satisfied.

4.7 MISCELLANEOUS SYSTEM ROUTINES

For the sake of completeness, the library routines explicitly called by RQA-1 will be listed here with a brief mention of their function.

(a) BNBCD.(X) converts its argument X from binary to binary coded decimal (Hollerith) form suitable for punching or printing characters.

(b) BPUNCH.($\mathcal{L}$) punches the values of the list $\mathcal{L}$ in column binary format.*

(c) BREAD.($\mathcal{L}$) reads the values of the list $\mathcal{L}$ from column binary cards.

(d) ELOG.(X) computes the natural logarithm of X.

(e) ERROR. produces a call to the monitor system with a dump of memory if requested.

(f) FTRAP. sets the underflow trapping procedure so that underflows are replaced by normal zeros until NTRAP is executed.

(g) NSDBCD.(X,Y) provides for punching the BCD values X,Y in columns 72-80 of the next card punched. (Sequencing of the last characters of this field is accomplished by The University of Michigan Executive System.)

(h) NTRAP. resets the underflow trapping procedure to its normal condition so that an underflow produces an exit to the monitor system (as in ERROR).

(i) SYSTEM. produces a call to the monitor system.

(j) TODAY.(DATE,YEAR) gives to DATE and YEAR BCD values corresponding to the day, month, and year that the program is executed.

(k) ZERO.($\mathcal{L}$) sprays normal zeros into all locations specified by the list $\mathcal{L}$.

---

*Actually, in The University of Michigan system, it writes binary card images on the "punch output" tape, for offline punching.

4.8 CONCLUSION

We would like to make some remarks on the program as a whole. Except for the two subroutines DISCPL and PROUT,* the program operates on a state space which is one-dimensional. Thus part of the program needs no information with respect to the structure of the states, i.e., their multidimensional nature. The transition table created by DISCPL describes a one-dimensional process, but DISCPL also contains some information about the structure of the states which is passed on to PROUT. For example, if the state designation were two-dimensional, DISCPL would contain values for the upper bounds of each dimension. In the examples of Section 3, the upper bound for the internal state variable is C which is fixed, and the upper bound for the queue state variable is N which is put in as data. The upper bounds are needed only in the subroutine PROUT to prepare a suitable output format. PROUT was written specifically for a two-dimensional state description and presents the final iterate in terms of such a state description.

Thus, the two subroutines DISCPL and PROUT can be considered user-controlled. DISCPL must be supplied by the user, while PROUT may be replaced by the user if he desires output formats other than those built into RQA-1 or if he wishes to perform some additional caculation before printing results.

*PROUT is the program,called by OUT2, which prepares and prints the printed state probabilities and marginal probabilities.

5. PROCEDURES FOR THE USE OF THE PROGRAM

This section describes procedures for the use of RQA-1. It is concerned primarily with the data which must be supplied by the user and with the interpretation of the output produced by the program. It is assumed here that the program is used with The University of Michigan Executive System.

In normal use, the object deck of RQA-1 is followed by the DISCPL subroutine, a "$DATA" card, and the input data. Since the DISCPL subroutine must be prepared by the user, we will give a detailed discussion of its preparation. The data are in five classes, divided into two distinct groups of cards. In order of their appearance following the $DATA card, these are

GROUP 1

(1) Program control parameters
(2) Structural parameters
(3) Discipline parameters
(4) Transition table parameter

GROUP 2

(1) Initial Iterate.

Following the description of the programming of a DISCPL subroutine will be a discussion of the input data, the form of both normal and abnormal output, the structure of the program card deck, and the format of binary cards occurring as punched output.

5.1 THE DISCIPLINE SUBROUTINE, DISCPL

The chief input to RQA-1 provided by the user is the discipline subroutine. This user-generated program provides RQA-1 with a complete description of the model to be solved in the form of code which generates the transition table. In addition to correctly describing his model, the user must insure, while writing this subroutine, that the number of quadruples generated does not exceed 2,000 and that the linear index of every state does not exceed 5,000.*

The discipline subroutines must carry out certain necessary functions for RQA-1:

---

*Failure to do the former will usually result in a violation of low-core storage protection in the University of Michigan system and probably worse difficulties in other systems. Failure to do the latter will simply result in an error return to system with suitable diagnostics printed out.

(1) Specifying the complete transition table
(2) Reading or specifying the Group 1 parameters
(3) Specifying a name for the model.

The transition table consists of four vectors called TRANS, FIRST, DEST, and LAST corresponding respectively to the α, β, γ, and B of Section 3. The values of elements of these vectors will usually be dependent upon the values of discipline parameters which are supplied at execution time and, hence, cannot be treated as simple numerical vectors to be read as data. Rather, they are computed by the execution of program segments contained in the discipline subroutine. This facilitates examination of the behavior of a model as parameters are varied, since the program may be simply rerun with different numerical values of the parameters supplied.

Normally, the discipline subroutine must contain all the statements within the dashed blocks in Fig. 4. The first block contains the instruction to read the Group 1 data, and in this section, it will be assumed that this is done by the simplified input/output statements in MAD. The second block defines the model name ("name" should be replaced by any desired twelve characters, including spaces) and sets up necessary communication to the remainder of RQA-1 through PROGRAM COMMON.

Group 1 parameters which are not read as data may be declared or calculated in DISCPL. The Group 1 parameters will be defined later.

The vectors TRANS, FIRST, DEST, and LAST of the transition table can be set up in several ways, only one of which is exhibited in the listings of the DISCPL subroutines shown in Tables VI, VII, and VIII for the examples of Section 3. This method involves the use of two list-manipulating statements in MAD, namely: SET LIST TO and SAVE DATA.[1] SET LIST TO TRANS sets up a vector named TRANS, and SAVE DATA $\alpha_1, \alpha_2, \ldots, \alpha_n$ causes $\alpha_1, \alpha_2, \ldots, \alpha_n$ to be the n entries of this vector where the α's can be any arithmetic expression acceptable to MAD.

While the illustrated technique is quite useful, other methods will also be found suitable to particular problems. For example, the quadruples may be generated in a series of THROUGH loops when some systematic evaluation technique is available or the number of quadruples to be generated depends on data. Appendix B contains a discussion of a group of subroutines, contributed by J. H. Jackson, which have been found to be useful in the generation of quadruples under quite general circumstances. On the whole, the degree of innovation which is possible is limited chiefly by the user's desires and needs.

In determining the quadruples of the transition table, the period DELTA (δ) can be treated as a constant or a parameter. If it is a constant, as is the case in the examples of Section 3, it is set in DISCPL, but, if it is to be treated as a parameter, it can be defined in DISCPL or read in as data.

TABLE VI

"SIMPLE SERVER" DISCIPLINE SUBROUTINE

```
$COMPILE MAD                                   DISC0000        000787   09/09/65      2 09 11.6 PM

MAD (09 AUG 1965 VERSION) PROGRAM LISTING ... ... ...

    EXTERNAL FUNCTION                                                          DISC0010   *001
    ENTRY TO DISCPL.                                                           DISC0020   *002
  INPUT PARAMETERS ARE L, MU, EPSI, NQ
    PRINT COMMENT $8                              INPUT PARAMETER              DISC0030   *003
  1 S $                                                                        DISC0040   *003
    READ AND PRINT DATA                                                        DISC0050   *004

    MAXJ = 2                                                                   DISC0062   *005
    DELTA = 2                                                                  DISC0063   *006

    SET LIST TO TRANS                                                          DISC0080   *007
    SAVE DATA L,L,MU,MU,-L,-L-MU,-MU                                           DISC0090   *008

    SET LIST TO FIRST                                                          DISC0100   *009
    SAVE DATA 1,2,2,4,1,2,2*NQ+2                                               DISC0110   *010

    SET LIST TO DEST                                                           DISC0120   *011
    SAVE DATA 2,4,1,2,1,2,2*NQ+2                                               DISC0130   *012

    SET LIST TO LAST                                                           DISC0140   *013
    SAVE DATA 1,2*NQ  ,2,2*NQ+2,1,2*NQ,2*NQ+2                                  DISC0150   *014

    VECTOR VALUES NAME = $SIMPLE MODEL$                                        DISC0160   *015

    INTEGER FIRST,DELT,LAST,INIT,PCH,PRSW,DELTA,MAXJ,MXITER,NQ                 DISC0200   *016
    PROGRAM COMMON TRANS(2001),FIRST(2000),DEST(2000),LAST(2000),              DISC0220   *017
  1 PCH,PRSW,MXITER,EPSI,DELTA,INIT,RUN,TSW,NAME(1),MAXJ                       DISC0230   *017
    FUNCTION RETURN                                                            DISC0270   *018
    END OF FUNCTION                                                            DISC0280   *019
```

TABLE VII

"FIXMEM" DISCIPLINE SUBROUTINE

```
$COMPILE MAD,EXECUTE,DUMP,I/O DUMP,PRINT OBJECT,PUNCH OBJECT    DISC0000    030791  09/09/65  4 40 21.8 PM

MAD (09 AUG 1965 VERSION) PROGRAM LISTING ... ... ...

   EXTERNAL FUNCTION                                                          DISC0010  *001
   ENTRY TO DISCPL.                                                           DISC0020  *002
   INPUT PARAMETERS ARE L, NU1, NU2, EPSI, NQ
   PRINT COMMENT $8                                  INPUT PARAMETER          DISC0030  *003
 1 S $                                                                        DISC0040  *003

   READ AND PRINT DATA                                                        DISC0050  *004

   MAXJ = 11                                                                  DISC0062  *005
   DELTA = 11                                                                 DISC0063  *006

   SET LIST TO TRANS                                                          DISC0080  *007
   SAVE DATA L,L,L,L,L,L,L,L,L,L,L,                                           DISC0090  *008
 1          NU2,2*NU2,3*NU2,4*NU2,5*NU2,5*NU2,NU1,NU1,NU2,NU1,               DISC0100  *008
 2 NU1,2*NU2,NU1,NU1,3*NU2,NU1,NU1,4*NU2,NU1,                               DISC0110  *008
 3                                -L,-L-NU2,-L-2*NU2,-L-3*NU2,              DISC0120  *008
 4 -L-4*NU2,-L-5*NU2,-5*NU2,-L-NU1,-NU1,-L-NU1-NU2,-NU1-NU2,                DISC0130  *008
 5 -L-NU1-2*NU2,-NU1-2*NU2,-L-NU1-3*NU2,-NU1-3*NU2,-L-NU1-4*NU2,            DISC0140  *008
 6 -NU1-4*NU2                                                               DISC0150  *008

   SET LIST TO FIRST                                                          DISC0160  *009
   SAVE DATA 1,2,3,4,5,6,7,8,9,10,11,2,3,4,5,6,17,7,18,8,8,19,9,              DISC0170  *010
 1 9,20,10,10,21,11,11,1,2,3,4,5,6,11*NQ+6,7,11*NQ+7,8,11*NQ+8,9            DISC0180  *010
 2 ,11*NQ+9,10,11*NQ+10,11,11*NQ+11                                         DISC0190  *010

   SET LIST TO DEST                                                           DISC0200  *011
   SAVE DATA 7,8,9,10,11,17,18,19,20,21,22,1,2,3,4,5,11,2,8,7,3,              DISC0210  *012
 1 9,8,4,10,9,5,11,10,6,1,2,3,4,5,6,11*NQ+6,7,11*NQ+7,8,11*NQ+8,            DISC0220  *012
 2 9,11*NQ+9,10,11*NQ+10,11,11*NQ+11                                        DISC0230  *012

   SET LIST TO LAST                                                           DISC0240  *013
   SAVE DATA 1,2,3,4,5,11*(NQ-1)+6,11*(NQ-1)+7,11*(NQ-1)+8,                   DISC0250  *014
 1 11*(NQ-1)+9,11*(NQ-1)+10,11*(NQ-1)+11,2,3,4,5,6,11*NQ+6,7,               DISC0260  *014
 2 11*NQ+7,11*NQ+8,8,11*NQ+8,11*NQ+9,9,11*NQ+9,11*NQ+10,10,                 DISC0270  *014
 3 11*NQ+10,11*NQ+11,11*NQ+11,1,2,3,4,5,11*(NQ-1)+6,11*NQ+6,                DISC0280  *014
 4 11*(NQ-1)+7,11*NQ+7,11*(NQ-1)+8,11*NQ+8,11*(NQ-1)+9,11*NQ+9,             DISC0290  *014
 5 11*(NQ-1)+10,11*NQ+10,11*(NQ-1)+11,11*NQ+11                              DISC0300  *014

   VECTOR VALUES NAME = $FIXMEM MODEL$                                        DISC0310  *015

   INTEGER FIRST,DEST,LAST,INIT,PCH,PRSW,DELTA,MAXJ,MXITER,NQ                 DISC0350  *016
   PROGRAM COMMON TRANS(2001),FIRST(2000),DEST(2000),LAST(2000),              DISC0370  *017
 1 PCH,PRSW,MXITER,EPSI,DELTA,INIT,RUN,TSW,NAME(1),MAXJ                       DISC0380  *017
   FUNCTION RETURN                                                            DISC0420  *018
   END OF FUNCTION                                                            DISC0430  *019
```

TABLE VIII

"RANMEM" DISCIPLINE SUBROUTINE

```
$COMPILE MAD                                        DISC0000        001195   09/13/65     12 58  6.7 PM

MAD (09 AUG 1965 VERSION) PROGRAM LISTING ... ... ...


      EXTERNAL FUNCTION                                                    DISC0000      *001
      ENTRY TO DISCPL.                                                     DISC0010      *002
      INPUT PARAMETERS ARE L1, L2, L3, L, NU1, NU2, NU3, EPSI, NQ
      PRINT COMMENT $8                                  INPUT PARAMETER    DISC0020      *003
    1 S    $                                                              DISC0040      *003
      READ AND PRINT DATA                                                  DISC0040      *004

      MAXJ = 7                                                             DISC0062      *005
      DELTA = 7                                                            DISC0063      *006


      SET LIST TO TRANS                                                    DISC0070      *007
      SAVE DATA L1,L2,L3,L,L1,L2+L3,L,L1,L2,L3,L1,L2+L3,L,L,L,L,           DISC0080      *008
    1                                 NU3,NU2,NU1/2.,NU2/2.,NU1,NU1,       DISC0090      *008
    2 NU1,NU3*L3/L,NU3*L2/L,NU3*L1/L,NU2*L2/(L2+L3),NU2*L3/(L              DISC0100      *008
    3 2+L3),NU1*(L3+L2)/(2.*L),NU2*L3/(2.*L),NU1*L1/(2.*L)+NU2*L2/(       DISC0110      *008
    4 2.*L),NU2*L1/(2.*L),NU1*L3/(L2+L3),NU1*L2/(L2+L3),NU1,NU1*(L2       DISC0120      *008
    5 +L3)/L,NU1*L1/L,NU3*L2*(L2+L3)/(L*L),2.*NU3*L2*L1/(L*L),NU3*L       DISC0130      *008
    6 1*L3/(L*L),NU3*L1*L1/(L*L),NU2*L2/L,NU2*L2*L1/((L2+L3)*L),NU2       DISC0140      *008
    7 *L1*(L2+L3)/(2.*L*L),NU2*L1*L1/(2.*L*L),NU3*L1*L1*(L3+L2)/(L*       DISC0150      *008
    8 L*L),NU3*L1*L1*L1/(L*L*L)                                           DISC0160      *008
      SAVE DATA     -L,                                                    DISC0170      *009
    1 -NU3,-L-NU3,-NU2,-L-NU2,-NU1/2.-NU2/2.,-L-NU1/2.-NU2/2.,-NU1,      DISC0180      *009
    2 -L-NU1,-NU1,-L-NU1,-NU1,-L-NU1                                       DISC0190      *009


      SET LIST TO FIRST                                                    DISC0200      *010
      SAVE DATA 1,1,1,2,3,3,4,5,5,5,6,6,7,10,12,13,2,3,4,4,5,6,7,9,        DISC0210      *011
    1 9,9,10,10,11,11,11,11,13,13,12,14,14,16,16,16,16,17,17,18,18,      DISC0220      *011
    2 23,23,1,7*NQ+2,2,7*NQ+3,3,7*NQ+4,4,7*NQ+5,5,7*NQ+6,6,7*NQ+7,7       DISC0230      *011


      SET LIST TO DEST                                                     DISC0240      *012
      SAVE DATA 5,3,2,9,4,10,11,6,4,12,7,13,14,17,19,20,1,1,3,5,1,5        DISC0250      *013
    1 ,6,2,3,5,3,2,10,12,4,6,12,4,2,13,7,10,4,12,6,10,4,13,7,13,7,1      DISC0260      *013
    2 ,7*NQ+2,2,7*NQ+3,3,7*NQ+4,4,7*NQ+5,5,7*NQ+6,6,7*NQ+7,7             DISC0280      *013
```

TABLE VIII (Concluded)

```
  SET LIST TO LAST                                                    DISC0290    *014
  SAVE DATA 1,1,1,7*(NQ-1)+2,3,3,7*(NQ-1)+4,5,5,5,6,6,7*(NQ-1)+       DISC0300    *015
1 7,7*(NQ-1)+3,7*(NQ-1)+5,7*(NQ-1)+6,2,3,4,4,5,6,7,7*(NQ  )+2,9       DISC0310    *015
2 ,9,10,7*(NQ  )+3,7*(NQ  )+4,7*(NQ  )+4,7*(NQ  )+4,11,7*(NQ  )      DISC0330    *015
3 +6,7*(NQ  )+6,7*(NQ  )+5,7*(NQ  )+7,7*(NQ  )+7,7*(NQ  )+2,         DISC0340    *015
4 7*(NQ  )+2,7*(NQ  )+2,           16,7*(NQ  )+3,7*(NQ  )+3,         DISC0345    *015
5 7*(NQ  )+4,7*(NQ  )+4,7*(NQ  )+2,7*(NQ  )+2                       DISC0350    *015
  SAVE DATA 1,7*NQ+2,7*(NQ-1)+2,7*NQ+3,7*(NQ-1)+3,7*NQ+4,7*(NQ-      DISC0351    *016
1 1)+4,7*NQ+5,7*(NQ-1)+5,7*NQ+6,7*(NQ-1)+6,7*NQ+7,7*(NQ-1)+7         DISC0352    *016


  VECTOR VALUES NAME = $RANMEM MODEL$                                 DISC0360    *017

  INTEGER FIRST,DEST,LAST,INIT,PCH,PRSW,DELTA,MAXJ,MXITER,NQ          DISC0400    *018
  PROGRAM COMMON TRANS(2001),FIRST(2000),DEST(2000),LAST(2000),       DISC0420    *019


1 PCH,PRSW,MXITER,EPSI,DELTA,INIT,RUN,TSW,NAME(1),MAXJ                DISC0430    *019
  FUNCTION RETURN                                                     DISC0470    *020
  END OF FUNCTION                                                     DISC0480    *021
```

```
EXTERNAL FUNCTION
ENTRY TO DISCPL
PRINT COMMENT $8                    INPUT PARAMETERS $
READ AND PRINT DATA
```

. . . . . .

Portion of program defining Group 1 parameters which are not read as data

. . . . . .

Portion of program defining transition table in terms of
DELTA, TRANS, FIRST, DEST, LAST

. . . . . .

```
VECTOR VALUES NAME  =  $ "Name"$
INTEGER FIRST, DEST, LAST, INIT, PCH, PRSW, DELTA, MAXJ, MXITER
PROGRAM COMMON TRANS(2001), FIRST(2000), DEST(2000), LAST(2000), PCH, PRSW,
   MXITER, EPSI, DELTA, INIT, RUN, TSW, NAME(1), MAXJ
FUNCTION RETURN
END OF FUNCTION
```

Fig. 4 DISCPL subroutine.

We will complete this discussion with a brief reference to the three DISCPL subroutines listed in Tables VI, VII, and VIII. In all these cases, the SET LIST TO and SAVE DATA statements have been used. All these examples have been formulated in terms of a single queue and a complex server. The parameter NQ is used in the transition tables to represent the maximum queue length. Of course, any integer-valued variables introduced by the user must be declared. For these examples, DELTA was chosen equal to C (the cardinality of the set J) and so the following pair of statements is used:

$$\begin{aligned} \text{DELTA} &= x \\ \text{MAXJ} &= x \end{aligned}$$

where x is 2,7,11 for the SIMPLE QUEUE, RANMEM MODEL and FIXMEM MODEL respectively. (MAXJ is a structural parameter (Group 1) which, in these cases, is also equal to the cardinality of J.)

Group 1 parameters are defined in the next three subsections.

5.2 PROGRAM CONTROL PARAMETERS

The program control parameters are named PCH, PRSW, MXITER, EPSI, INIT, and TSW. The mode, preset value, and range of permitted values of each of these variables are given in Table IX. If these parameters are not specified on any particular run, they have, for that run, the value last supplied on a previous run. If no previous run has specified their value, they have the "preset" value shown.

TABLE IX

PROGRAM CONTROL DATA

Name	Mode	Preset Value	Other Values
PCH	INTEGER	1	0,1
PRSW	INTEGER	1	0,1
MXITER	INTEGER	100	$\geq 0, < 10^5$
EPSI	FLOATING POINT	10^{-4}	$\geq 0, \leq 1.$
INIT	INTEGER	0	0,1,2,3
TSW	INTEGER	1	0,1

EPSI is the value of the convergence test criterion. Thus, when the maximum absolute difference between corresponding elements of successive iterates is less than or equal to EPSI, the convergence test is satisfied. MXITER is the maximum number of iterations to be allowed in the calculation. If this number of iterations has not been satisfied, the calculation is terminated and output supplied.

Table X lists the significance of each of the values of the variables PCH, PRSW, INIT, and TSW.

TABLE X

THE FUNCTIONS OF PCH, PRSW, INIT, and TSW

Name	Value	Function
PCH	1	- Final iterate is punched in binary format.
	0	- Final iterate is not punched.
PRSW	1	- Final iterate is printed by execution of PROUT.
	0	- Final iterate is not printed. PROUT is not called.
INIT	0	- The initial iterate is calculated by the subroutine ESTIM.
	1	- The library subroutine BREAD is called and an initial iterate in binary format is read in. This input is the punched output of a previous run.
	2	- The initial iterate is read in by READ DATA statement and is in floating point format.
	3	- The final iterate from the previous run is used as the initial iterate for the present run.
TSW	1	- The transition table is tested to assure its stochastic character and other properties. (TEST is called).
	0	- The transition table is not tested.

5.3 STRUCTURAL PARAMETER

Structural parameters convey dimensionality information to be output printing subroutines. For the output printing subroutine provided (PROUT), there is only one such parameter, MAXJ. This parameter indicates the maximum extent of the least significant coordinate J of the two-dimensional output array of proba-

bilities. In PROUT, a state with index K will be treated as a state with vector value (I,J) as determined by the mapping

$$K = MAXJ * I + J + 1$$

corresponding to Eq. (28). Thus, MAXJ corresponds to the cardinality of the set {J} described in Section 3.

The preset value and the range of MAXJ are given in Table XI. If the preset value is used (by not specifying MAXJ) or if a value greater than 2500 is specified, the printed output will supply the results ten-to-a-row and fifty-to-a-block.

TABLE XI

STRUCTURAL PARAMETER

Name	Mode	Preset Value	Restriction
MAXJ	INTEGER	10	MAXJ $\leq$ 2500

5.4 DISCIPLINE PARAMETERS

These data consist of the required parameters for the particular queue discipline being run. They are defined by DISCPL only. If they are not floating point constants, their mode must be declared. For the examples discussed in Section 3, such parameters are the L's and NU's. For the FIXMEM model (Table VII), a data card with discipline parameters would be

$$L = 1.0, \quad NU1 = 1.5, \quad NU2 = .25, \ \ldots$$

In general, these parameters are the independent variables involved in the quadruples of the transition table.

5.5 TRANSITION TABLE PARAMETER

This parameter is DELTA, which is an integer. Its use was discussed in Section 5.1. It is treated essentially like a structural parameter, but, while structural parameters refer to the state-space structure, DELTA is actually a part of the description of the transition table. It has a preset value of 1.

5.6 INITIAL ITERATE

As seen in Table X, when INIT is 1 or 2, the initial iterate is read in as data. For INIT = 1, the initial iterate must be presented in column binary format. It is usually the set of cards containing a final iterate which was punched in binary mode in a previous run as the result of setting PCH = 1.

The form of this iterate is described in Section 5.11. The entire set of output cards as described there must be used as input for the initial iterate, i.e., the header card followed by the binary data.

For INIT = 2, the initial iterate must be supplied in floating point form to be read by a READ DATA type statement. Unspecified values are automatically zero. Before being used, the initial iterate will be normalized to a probability vector. Therefore, at least one element must be specified. Typical data cards would be

$$V(1) = .25,\quad V(3) = .50,\quad V(10) = .25*$$

or

$$V(1) = .20,\ .15,\ .10,\ .05,\quad V(50) = .20,\ .15,\ .10,\ .05* \quad .$$

Values can be given for any V(I) where I runs from 1 to the largest integer denoting a state.

These data are Group 2 data and must appear after the Group 1 data of the run.

5.7 FORMAT OF INPUT DATA

There is no specified order for the Group 1 data. These data are in the format of cards to be read by a READ DATA statement in MAD and must be followed (according to that format) by "*".

Following the Group 1 data will be the Group 2 data, if required (i.e., if INIT = 1 or INIT = 2). Following this will be the data for the next run, if further runs of the model with different data values are desired. Any number of sets of data may be stacked in this way.

The run number sequence which identifies all output of RQA-1 can be started (restarted) at a user-specified value on any run by insertion of a card just ahead of the Group 1 data for the run with the format

THE INITIAL RUN NUMBER IS XXX .

This statement must start in the first column of the card, and words must be separated by a single space. The XXX may be any non-negative integer of three or less digits. If this is not done, runs will be numbered starting with 1 for the first run after loading.

The following may be a typical set of data cards for one run:

```
EPSI  =  .0005, MXITER  =  200, INIT  =  2
L  =  1,
MU1  =  1.5, MU2  =  .25*
V(1)  =  .5, .3, .1, .1*
```

Alternatively, another set may be:

```
THE INITIAL RUN NUMBER IS 75
INIT  =  1
L  =  1, MXITER  =  50, MU1  =  1.5, MU2  =  .25*
.....
(Binary card deck as described in Section 5.11)
.....
```

Note that, in this case, EPSI will be the same value as it was on the previous run (or it will be .0001 if this is the first set of data) and that, in both cases, PCH, PRSW, and TSW are the same as they were in previous runs (or their preset values if this is the first set of data).

5.8 NORMAL OUTPUT

The normal output of RQA-1, illustrated in Table XII, should be self-explanatory.

However, it should be noted that the probabilities are printed in terms of a two-dimensional state designation where PROB(I,J) is the limiting state

TABLE XII

TYPICAL OUTPUT OF RQA-1

```
 RQA RESULTS FOR FIXMEM MODEL DISCIPLINE                    SEPT 1, 1965 VERSION OF RQA-1

 RUN   1, 16 SEP 1965


                    INPUT PARAMETERS

   L=1.,NU1=1.5,NU2=0.5,

   NQ=15*

 THE INITIAL ITERATE WAS DETERMINED BY THE SUBROUTINE ESTIM.


******************************** SUCCESSFUL RUN ********************************

     CONVERGENCE TEST CRITERION (EPSI) =  .00010000
   MAXIMUM ALLOWED ITERATIONS (MXITER) =  100
                  ITERATIONS TAKEN =   89
MAXIMUM DIFFERENCE BETWEEN LAST TWO ITERATES = 1.1E-04
               ESTIMATED ORDER OF ERROR =  .1E-03   (APPROXIMATE CONVERGENCE FACTOR = .9016573)

THE SOLUTION VECTOR HAS BEEN PUNCHED
```

TABLE XII (Continued)

THE FOLLOWING ARE THE LIMITING STATE PROBABILITIES, PROB(I,J).
PROB(I) IS THE SUM OF PROB(I,J) OVER ALL STATES WITH THE VALUE I FOR THE FIRST VARIABLE

I	PROB(I)	PROB(I,J)									
		J = 0	1	2	3	4	5	6	7	8	9
0	.52348530	.04460908 .02029071	.08872072	.08792652	.05749297	.02716861	.00869784	.02980799	.05949457	.05947242	.03980388
1	.15267809	.00000000 .01496318	.00000000	.00000000	.00000000	.00000000	.00889547	.02000391	.04019641	.04069529	.02792383
2	.10608881	.00000000 .01070170	.00000000	.00000000	.00000000	.00000000	.00712090	.01348743	.02730851	.02796113	.01950914
3	.07300190	.00000000 .00744561	.00000000	.00000000	.00000000	.00000000	.00521510	.00912379	.01858519	.01916128	.01347094
4	.04970614	.00000000 .00506128	.00000000	.00000000	.00000000	.00000000	.00364714	.00617654	.01261922	.01303470	.00916726
5	.03345351	.00000000 .00337126	.00000000	.00000000	.00000000	.00000000	.00247474	.00417181	.00851817	.00877507	.00614247
6	.02223266	.00000000 .00220401	.00000000	.00000000	.00000000	.00000000	.00164044	.00280297	.00569988	.00583454	.00405082
7	.01457771	.00000000 .00141568	.00000000	.00000000	.00000000	.00000000	.00106571	.00186844	.00377258	.00382645	.00262885
8	.00942488	.00000000 .00089424	.00000000	.00000000	.00000000	.00000000	.00067974	.00123296	.00246581	.00247318	.00167895
9	.00600726	.00000000 .00055633	.00000000	.00000000	.00000000	.00000000	.00042634	.00080406	.00158983	.00157486	.00105586
10	.00377696	.00000000 .00034182	.00000000	.00000000	.00000000	.00000000	.00026350	.00051756	.00101066	.00098847	.00065495
11	.00234679	.00000000 .00020801	.00000000	.00000000	.00000000	.00000000	.00016089	.00032868	.00063387	.00061297	.00040237
12	.00144560	.00000000 .00012387	.00000000	.00000000	.00000000	.00000000	.00009656	.00020615	.00039372	.00037850	.00024682
13	.00088689	.00000000 .00006910	.00000000	.00000000	.00000000	.00000000	.00005556	.00012857	.00024633	.00023900	.00014833
14	.00054715	.00000000 .00003438	.00000000	.00000000	.00000000	.00000000	.00002964	.00008311	.00017104	.00014868	.00008031
15	.00033959	.00000000 .00000949	.00000000	.00000000	.00000000	.00000000	.00001674	.00008897	.00011723	.00007539	.00003186

TABLE XII (Concluded)

PROB(J) IS THE SUM OF PROB(I,J) OVER ALL STATES WITH THE VALUE J FOR THE SECOND VARIABLE

CHECKSUM	PROB(J)									
	J = 0	1	2	3	4	5	6	7	8	9
.99999920	.04460908 .06769066	.08872072	.08792652	.05749297	.02716861	.04048632	.09083293	.18282300	.18525183	.12699661

probability of state (I,J). The marginal probabilities PROB(I) and PROB(J), which are the sums of PROB(I,J) over J and I respectively, are also listed. Thus, for a fixed value of I, PROB(I) is printed and followed on the same line by the values of PROB(I,J) as J runs over its allowed values. CHECKSUM, the sum of all the elements of the probability vector, may not be exactly equal to 1 due to roundoff.

5.9 DIAGNOSTICS OF ABNORMAL OUTPUT

A run can terminate without supplying answers to the given problem for several reasons; there may be no input data, the number of allowable states or entries in the transition table may be exceeded, or there may be an error in the transition table. In each of these instances, appropriate statements an diagnostic aids are printed. This termination is caused by execution of the MAD instruction

EXECUTE ERROR.

which causes a return to the operating system proceded by a dump of storage if requested.

The following is a further explanation of a few of these diagnostics which may not be self-explanatory:

(1) THE FOLLOWING QUADRUPLES HAVE AN ERROR IN FIRST OR LAST.

This is printed if LAST(I)-FIRST(I) is not an integer multiple of DELTA. This is inconsistent with the rules for forming a transition table.

(2) THE FOLLOWING QUADRUPLES HAVE AN ERROR IN DEST.

This is printed if there is a transition specified into a point, and there is no other transition with that point as a source.

(3) THE FOLLOWING ROW SUMS, SUM(I), OF THE STOCHASTIC MATRIX ARE INCORRECT.

The appearance of this statement indicates that the matrix represented by the transition table (at this point, it should be stochastic) is not stochastic. If a row sum corresponding to a point is nonzero, then a transition is being defined for something other than a state. If a row sum corresponding to a state is less than one, then a possible transition is being overlooked. If it is greater than one, an extraneous transition has been included.

5.10 FORMAT OF PROGRAM DECK

Two formats will be described; one for which the DISCPL subroutine is to be compiled, and the other for which it has been compiled and, similar to the rest of the program, is now on binary cards. For the first case, the following is the deck format:

2 yellow ID cards
"$ COMPILE MAD, EXECUTE, (etc.)" card
DISCPL MAD deck
"$ BINARY" card
RQA-1 binary deck
"$DATA" card
DATA

For the second case, the following is the deck format:

2 yellow ID cards
"$ BINARY, EXECUTE (etc.)" card
RQA-1 binary deck including DISCPL binary deck
"$DATA" card
Data

5.11 FORMAT OF THE PUNCHED FINAL ITERATE

As discussed in Section 5.2, when PCH = 1, the final iterate will be punched on binary cards. To illustrate the form of this output, we will assume that the name of the model is SIMPLE QUEUE, the date is MAY 25, 1965, and this is the 15th run.

The first card punched, called the header card, will have the following form:

SIMPLE QUEUE RUN 15 25 MAY 1965 SIMPLE Q.

Since, in the University of Michigan system, binary cards have only columns 73-80 interpreted (printed on the top of card), this card will be identified by the first 8 characters in the model name appearing in columns 73-80. The next card will contain the linear index corresponding to the maximum state, called SCOPE, punched in binary mode. Columns 73-80 of this card will contain

25 MAY 01

i.e., the day, the month, and the number 01. The succeeding cards will contain the values of the final iterate in binary mode. The number of such values will be SCOPE. Columns 73-75 of these cards will contain

(i.e., the run number). Columns 78-80 will contain the numbers 001, 002, etc. (i.e., a sequential numbering of the cards).

Each final iterate punched by RQA-1 will automatically have this output card format.

6. CONCLUSION

The purpose of the RQA-1 program is to evaluate equilibrium joint-probability distributions of queue lengths and system conditions for very large, finite Markov queueing systems. Although only these probabilities are obtained by the program, it is possible to estimate from them a wide variety of other performance statistics. The expected waiting and throughput times are easily determined as well as the probability distributions of any of the state variables (e.g., queue lengths, channel occupancies) by taking appropriate sums of state probabilities. Special programs can be written to determine the above statistics which will have as their input the probability distribution calculated by the RQA-1 program.

As part of this section, we will present some timing data for computer runs of the queueing models discussed previously.

6.1 TIMING EXPERIENCE

Results are shown in Table XIII(a) for the FIXMEM model with λ and μ_1 fixed and μ_2 varied. Table XIII(b) gives the results for one run of the model with 611 states. It is interesting to note that for each model the number of iterations per second varies inversely as the number of states. Table XIV presents some results for the RANMEM model for which the discipline parameters are fixed but EPSI and the number of states are varied. For the problem with 77 states and EPSI = 10^{-4}, the number of iterations per second is 48.3. For the problem with 357 states and EPSI = 10^{-4}, this number is 10. The ratio of states is 357/77 = 4.6, and the ratio of iterations per second is 48.3/10 = 4.8.

It should be noted that these results have been achieved for very sparse matrices. In fact, for the FIXMEM model, there is an average of four nonzero elements per row. Thus, for a 611 state model, the ratio of the number of nonzero elements to the total number of elements is $(4 \times 611)/(611 \times 611) = .0065$.

Using the equation developed in Section 2, we can compare the number of multiplications required for a solution of a typical FIXMEM model by simulation with the number of multiplications required by the numerical methods employed in RQA-1. For the FIXMEM model with $\mu_2 = .3333$, the EPSI satisfied was $.4058 \times 10^{-4}$ which corresponds to an ϵ, in the equation, of about 10^{-3}. A simulation for a model with these parameters would require at least 100 times as many multiplications as the numerical procedure or about one minute as compared to one-half second. For the FIXMEM model with $\mu_2 = .5$ and EPSI = $.6429 \times 10^{-4}$, a simulation would require about 150 times as many multiplications or about 4-1/2 minutes as compared with 2 seconds for an $\epsilon = 10^{-3}$.

TABLE XIII

"FIXMEM" MODEL
(The initial vector is estimated for each run)

$\lambda = 1.0$, $\mu_1 = 1.5$

μ_2	EPSI ($\times 10^{-4}$)	Number of Iterations	Iteration of (Seconds)	Number of Iterations per Second	Total Time (Seconds)	Maximum Probability
			(a) Number of States = 101			
1.0	0.4884	73	2.4	30.4	3.6	0.1251
0.5	0.6429	64	1.8	35.5	3.6	0.0801
0.3333	0.4058	23	0.6	38.4	2.4	0.0525
0.2857	0.9508	63	1.8	35.0	3.6	0.0411
TOTALS		223	6.6	33.8	13.2	
			(b) Number of States = 611			
0.5	1.0	54	8.4	6.4	17.4	0.0783

TABLE XIV

"RANMEM" MODEL
(The initial vector is estimated for each run)

$\lambda_1 = 1.5$, $\lambda_2 = .3$, $\lambda_3 = .2$;
$\mu_1 = 2$, $\mu_2 = 1.5$, $\mu_3 = 1.2$

Number of States	EPSI x 10^{-4}	Number of Iterations	Iteration of (Seconds)	Number of Iterations per Second	Total Time (Seconds)	Maximum Probability
77	1.0	58	1.2	48.3	3.6	0.3828
77	0.1	102	2.4	42.5	4.8	0.3844
77	0.0478	115	3.0	38.3	5.4	0.3845
TOTALS		275	6.6	41.7	13.8	
357	1.0	60	6.0	10.0	13.8	0.3812

6.2 FINAL REMARKS

This work represents the first stage in exploiting numerical solution of stochastic models for the analysis of large systems. The major disadvantage of the technique is currently the relatively great difficulty of preparing correct discipline subroutines for complex models. It is hoped that this disadvantage can be overcome by the development of automatic programs to do this work from a very formalized description of the model to be solved (i.e., a translator for a problem-oriented language). With that accomplished, we believe the RQA-1 and its successors represent a very powerful tool indeed for system analysis.

The programs listed have used the May 1, 1965, version of MAD and the May 5, 1965, version of UMAP. These programs are known as the September 1, 1965, version of RQA-1. They are designed to operate with The University of Michigan Executive System, although an effort was made to make the program as independent as possible. We believe that with the information presented here a potential user could adapt the program to any comparable system.

6.3 ACKNOWLEDGMENT

The authors are particularly indebted to Professor R. V. Evans whose original suggestions prompted this work, to Mr. D. W. Fife who made many valuable suggestions on its application, to Dr. K. B. Irani for his critical examination of this manuscript, to J. H. Jackson for contributing the work in Appendix B, and to E. S. Walter and D. L. Mason for help to prepare portions of the final RQA-1 program.

REFERENCES

1. Arden, B., B. Galler, and R. Graham, MAD—The Michigan Algorithmic Decoder, University of Michigan Press, Ann Arbor, 1964.

2. Gantmacher, F. R., Matrix Theory, Vol. 1, Chelsea, New York, 1959, p. 142.

3. Frazer, R. A., W. J. Duncan, and A. R. Collar, Elementary Matrices, Cambridge, University Press, New York, 1960, p. 83. (It follows from the definition in [3] and [2] that a root of multiplicity r has degeneracy r if its elementary divisors are of first degree.)

4. Gantmacher, R. F., Applications of the Theory of Matrices, Interscience Publishers, New York, 1959, Theorem 10, p. 102.

5. Varga, R. S., Matrix Iterative Analysis, Prentice-Hall, New York, 1963, pp. 16-17.

6. Parzen, E., Stochastic Processes, Holden-Day, San Francisco, 1964, pp. 248-258.

7. Chung, K. L., Markov Chains with Stationary Transition Probabilities, Springer-Verlag, Hamburg, 1960.

8. Fife, D., and R. Rosenberg, "Queueing in a Memory-Shared Computer," Proceedings of the 19th National Conference of the ACM, Philadelphia, 1964.

9. Evans, R. V., "Several Queueing Systems Suggested by Computer Organization Problems," Internal Memo.

10. Saaty, T. L., Elements of Queueing Theory, McGraw-Hill, New York, 1961, pp. 38-40.

APPENDIX A

LISTING OF RQA-1

This program was written in the MAD and UMAP languages and processed in The University of Michigan Executive System. The listings are complete for the September 1, 1965, version of the RQA-1 program. The listings for the DISCPL subroutines of the examples of Section 3 are given in Section 5. Two versions of ITER are supplied, one in UMAP and the other in MAD.

MAD (09 AUG 1965 VERSION) PROGRAM LISTING ... ...

```
      REFERENCES ON                                                    *001
      ***************************************************************
           INITIALIZATION OF PARAMETERS
      ***************************************************************
      VFLAG=1                                                  MAIN0005  *001
      CWFLAG=0                                                 MAIN0006  *002
      VECTOR VALUES INIT=0                                     MAIN0010  *003
      VECTOR VALUES PCH=1                                      MAIN0020  *004
      VECTOR VALUES PRSW=1                                     MAIN0025  *005
      VECTOR VALUES EPSI=0.0001                                MAIN0030  *006
      VECTOR VALUES MXITER=100                                 MAIN0040  *007
      VECTOR VALUES DELTA=1                                    MAIN0045  *008
      VECTOR VALUES TSW=1                                      MAIN0060  *009
      VECTOR VALUES RUN=1                                      MAIN0061  *010
      VECTOR VALUES MAXJ=10                                    MAIN0067  *011
      VECTOR VALUES CHK=-1                                     MAIN0068  *012
      ***************************************************************
           PROBLEM TITLE
           THE INITIAL RUN NUMBER MAY BE SPECIFIED, IF DESIRED, BY
           PUNCHING THE STATEMENT... THE INITIAL RUN NUMBER IS XXX,
           STARTING IN COLUMN ONE.
      ***************************************************************
BEGIN      PRINT FORMAT $1H1,S14,16HRQA RESULTS FOR ,2C6,11H DISCIPLINE,  MAIN0070  *013
     1 S20,29HSEPT 1, 1965 VERSION OF RQA-1/1HQ*$,NAME,NAME(1)          MAIN0071  *013
           EXECUTE TODAY.(DATE,YEAR)                                   MAIN0072  *014
           LOOK AT FORMAT $2C6*$,RUNP,RUNP(1)                          MAIN0073  *015
           WHENEVER RUNP.E.$THE IN$.AND.RUNP(1).E.$ITIAL $,READ FORMAT MAIN0074  *016
     1 $S25,I47*$,RUN                                                  MAIN0075  *016
           WHENEVER RUN.GE.1000, RUN=0                                 MAIN0076  *017
           PRINT FORMAT $1HQ,S14,4HRUN ,I3,1H,,S1,2C6*$,RUN,DATE,YEAR  MAIN0077  *018
           ZERO.(TRANS...TRANS(2000),FIRST...FIRST(2000),DEST...DEST   MAIN0095  *019
     1 (2000),LAST...LAST(2000))                                       MAIN0096  *019
      ***************************************************************
           DATA READ IN AND RATE TABLE PREPARED
      ***************************************************************
           EXECUTE DISCPL.                                             MAIN0110  *020
      ***************************************************************
```

```
C
C       CALCULATION AND TEST OF THE NUMBER OF ENTRIES IN THE
C       TRANSITION TABLE.  ERROR. IS A SYSTEM POST MORTEM DUMP
C       ROUTINE.
C
C*****************************************************************
C
      THROUGH MAIN3, FOR I=1,1,I.G.2000                                 MAIN0111    *021
MAIN3 WHENEVER TRANS(I).NE.0., NTRANS = I                               MAIN0112    *022    01
      WHENEVER CHK.E.-1,TRANSFER TO MAIN4                               MAIN*112    *023
      PRINT COMMENT $6     ****TRANSITION TABLE EXCEEDS 2000 ENTRIES    MAIN0113    *024
     1 $                                                                MAIN0114    *024
      EXECUTE ERROR.                                                    MAIN0115    *025
C
C*****************************************************************
C
C       CALCULATION AND TEST OF THE SCOPE OF THE PROB. VECTOR
C
C*****************************************************************
C
MAIN4 SCOPE=0                                                           MAIN0116    *026
      THROUGH MAIN5, FOR I=1,1,I.G.NTRANS                               MAIN0117    *027
      SCOPET = DEST(I) + LAST(I) - FIRST(I)                             MAIN0118    *028    01
      WHENEVER LAST(I).G.SCOPE,SCOPE=LAST(I)                            MAIN*118    *029    01
MAIN5 WHENEVER SCOPET.G.SCOPE,SCOPE = SCOPET                            MAIN0119    *030    01
      WHENEVER SCOPE.LE.5000, TRANSFER TO MAIN6                         MAIN0120    *031
      PRINT COMMENT $6     ****THE LINEAR INDEX OF AT LEAST ONE STAT    MAIN0121    *032
     1 E EXCEEDS 5000$                                                  MAIN0122    *032
      PRINT FORMAT $1HO,S14,H+ONE INDEX IS +,I11*$,SCOPE                MAIN0123    *033
      ARATEM=1.                                                         MAIN0124    *034
      DIADD=0.                                                          MAIN0125    *035
      EXECUTE TOUT3.                                                    MAIN0126    *036
C
C*****************************************************************
C
C       CONVERGENT STOCHASTIC MATRIX PREPARED AND TESTED IF
C       REQUIRED
C
C*****************************************************************
C
MAIN6 EXECUTE STOCAL.                                                   MAIN0140    *037
      WHENEVER TSW.E.1, EXECUTE TEST.                                   MAIN0150    *038
C
C*****************************************************************
C
C       CHOICE OF INITIAL VECTOR
C
C*****************************************************************
C
      WHENEVER INIT.E.0                                                 MAIN0170    *039
      PRINT COMMENT $8              THE INITIAL ITERATE WAS DETERMI     MAIN0171    *040    01
     1 NED BY THE SUBROUTINE ESTIM.$                                    MAIN0172    *040
      EXECUTE ESTIM.                                                    MAIN0180    *041    01
C
      OR WHENEVER INIT.E.1                                              MAIN0190    *042    01
      READ FORMAT INIT1R,TEMP(1)...TEMP(6)                              MAIN0191    *043    01
      PRINT FORMAT INIT1,TEMP(1)...TEMP(6)                              MAIN0192    *044    01
      DIMENSION TEMP(6)                                                 MAIN0193    *045    01
      VECTOR VALUES INIT1=$1H8,S14,38HTHE INITIAL ITERATE WAS THE R     MAIN0195    *046    01
     1 ESULT OF ,6C6*$                                                  MAIN0196    *046
      VECTOR VALUES INIT1R=$S15,6C6*$                                   MAIN0197    *047    01
```

```
      EXECUTE BREAD.(SCOPEP)                                          MAIN0202   *048   01
      EXECUTE BREAD.(V(VFLAG,1)...V(VFLAG,SCOPEP))                    MAIN0203   *049   01

      OR WHENEVER INIT.E.2                                            MAIN0210   *050   01
      EXECUTE ZERO.(V(VFLAG,1)...V(VFLAG,SCOPE),V(CVFLAG,1)...V(CVF   MAIN0220   *051   01
     1 LAG,SCOPE))                                                    MAIN0221   *051
      VFLAG = 0                                                       MAIN0222   *052   01
      CVFLAG = 1                                                      MAIN0223   *053   01
      PRINT FORMAT $1H8,S14,H*THE INITIAL ITERATE IS GIVEN BELOW*/1H-*$ MAIN0224 *054   01
      READ AND PRINT DATA                                             MAIN0230   *055   01

      OTHERWISE                                                       MAIN0231   *056   01
      PRINT COMMENT $8              THE INITIAL ITERATE WAS THE RES   MAIN0232   *057   01
     1 ULT OF THE PREVIOUS RUN$                                       MAIN0234   *057
      END OF CONDITIONAL                                              MAIN0240   *058   01

      ***************************************************************

          INITIAL VECTOR NORMALIZED

      ***************************************************************

      VECSUM=0.                                                       MAIN0260   *059
      THROUGH VEC1, FOR I=1,1,I.G.SCOPE                               MAIN0270   *060
VEC1  VECSUM=VECSUM+V(VFLAG,I)                                        MAIN0280   *061   01
      THROUGH VEC2, FOR I=1,1,I.G.SCOPE                               MAIN0290   *062
VEC2  V(VFLAG,I)=V(VFLAG,I)/VECSUM                                    MAIN0300   *063   01

      ***************************************************************

          ITERATION ROUTINE AND CONVERGENCE TESTING

      ***************************************************************

      NITER=0                                                         MAIN0440   *064
MAIN1 MAXIX=CVGTST.(BEGIN)                                            MAIN0450   *065
      SAVE1=V(VFLAG,MAXIX)                                            MAIN0460   *066
MAIN2 SAVE2=SAVE1                                                     MAIN0470   *067
      EXECUTE ITER.(1)                                                MAIN0480   *068
      SAVE1=V(VFLAG,MAXIX)                                            MAIN0490   *069
      WHENEVER.ABS.(SAVE2-SAVE1).LE.EPSI.OR.MXITER.LE.NITER, TRANSFER TO MAIN1 MAIN0500 *070
      TRANSFER TO MAIN2                                               MAIN0510   *071

      INTEGER INIT,MXITER,I,SCOPE,VFLAG,NITER,MAXIX,MAXJ,             MAIN0520   *072
     1 CVFLAG,DELTA,FIRST,DEST,LAST,EXIT1,EXIT2,J,CVGTST,,RUN,DATE,   MAIN0530   *072
     2 YEAR,SCOPEP,NAME,TEMP,SCOPET,NTRANS,CHK,PCH,TSW,PRSW,RUNP(1)   MAIN0535   *072
      EQUIVALENCE (TRANS,NTRANS),(TRANS(2001),CHK)                    MAIN0545   *073
      PROGRAM COMMON TRANS(2001),FIRST(2000),DEST(2000),LAST(2000),   MAIN0550   *074
     1 PCH,PRSW,MXITER,EPSI,DELTA,INIT,RUN,TSW,NAME(1),MAXJ           MAIN0560   *074
      ERASABLE DUMMY(100),VFLAG,CVFLAG,SCOPE,NITER,V((0...1)*5000),   MAIN0580   *075
     1 ARATEM,VECSUM,EXIT1,EXIT2,EXIT3,I,J,DATE,YEAR,DIADD            MAIN0585   *075
      END OF PROGRAM                                                  MAIN0600   *076
```

```
$COMPILE MAD,EXECUTE,DUMP,I/O DUMP,PRINT OBJECT,PUNCH OBJECT    STOC0000    000791    09/09/65    4 39 31.8 PM

MAD (09 AUG 1965 VERSION) PROGRAM LISTING ... ... ...


            REFERENCES ON                                                                   *001

            EXTERNAL FUNCTION                                           STOC0010    *001
            ENTRY TO STOCAL.                                            STOC0020    *002
            RATEM=0.                                                    STOC0030    *003
            THROUGH STOC1, FOR I=1,1,I.G.NTRANS                         STOC0040    *004
            WHENEVER RATEM.G.TRANS(I),RATEM=TRANS(I)                    STOC0050    *005          01
STOC1       CONTINUE                                                    STOC0060    *006          01

            WHENEVER RATEM.GE.0.                                        STOC0070    *007
            ARATEM =1./.99                                              STOC0090    *008    01
            DIADD=.01                                                   STOC0100    *009    01
            TRANSFER TO STOC3                                           STOC0105    *010    01
            END OF CONDITIONAL                                          STOC0106    *011    01

            ARATEM=(.ABS.RATEM)/.99                                     STOC0110    *012
            DIADD=1.0                                                   STOC0115    *013
STOC3       THROUGH STOC2, FOR I=1,1,I.G.NTRANS                         STOC0120    *014
            WHENEVER FIRST(I).NE.DEST(I)                                STOC0130    *015          01
            TRANS(I)=TRANS(I)/ARATEM                                    STOC0140    *016    01    01
            OTHERWISE                                                   STOC0150    *017    01    01
            TRANS(I)=(TRANS(I)/ARATEM)+DIADD                            STOC0160    *018    01    01
            END OF CONDITIONAL                                          STOC0170    *019    01    01
STOC2       CONTINUE                                                    STOC0180    *020          01

            INTEGER I,NTRANS,VFLAG,CVFLAG,DELTA,SCOPE,NITER,FIRST,DEST,LA    STOC0190    *021
          1 ST,MXITER,INIT,PCH,MAXJ,SCOPET                              STOC0200    *021
            EQUIVALENCE (TRANS,NTRANS)                                  STOC0210    *022
            PROGRAM COMMON TRANS(2001),FIRST(2000),DEST(2000),LAST(2000),    STOC0220    *023
          1 PCH,PRSW,MXITER,EPSI,DELTA,INIT,RUN,TSW,NAME(1),MAXJ        STOC0230    *023
            ERASABLE DUMMY(100),VFLAG,CVFLAG,SCOPE,NITER,V((0...1)*5000),    STOC0245    *024
          1 ARATEM,VECSUM,EXIT1,EXIT2,EXIT3,I,J,DATE,YEAR,DIADD         STOC0246    *024
            FUNCTION RETURN                                             STOC0260    *025
            END OF FUNCTION                                             STOC0270    *026
```

MAD (09 AUG 1965 VERSION) PROGRAM LISTING ... ...

```
                    REFERENCES ON                                                    *001
                    EXTERNAL FUNCTION                                    TEST0010    *001
                    ENTRY TO TEST.                                       TEST0020    *002
                    ***************************************************************
                        TEST FOR LAST(I)-FIRST(I)=R*DELTA, WHERE R IS AN INTEGER
                    ***************************************************************
                    ZERO.(V(CVFLAG,1)...V(CVFLAG,SCOPE))                 TEST0030    *003
                    EXIT1=0                                              TEST0040    *004
                    THROUGH TES1, FOR I=1,1,I.G.NTRANS                   TEST0050    *005
                    THROUGH TES2, FOR J=FIRST(I),DELTA,J.G.LAST(I)       TEST0060    *006    01
TES2                V(CVFLAG,J)=V(CVFLAG,J)+TRANS(I)                     TEST0070    *007    02
                    WHENEVER J.E.LAST(I)+DELTA.AND.LAST(I).LE.SCOPE.AND.FIRST(I)  TEST0090  *008  01
                   1 .G.0, TRANSFER TO TES1                              TEST0095    *008
                    EXIT1=EXIT1+1                                        TEST0096    *009    01
                    WHENEVER EXIT1.E.1,EXECUTE TOUT1.                    TEST0100    *010    01
                    EXECUTE TOUT1.                                       TEST0110    *011    01
TES1                CONTINUE                                             TEST0120    *012    01
                    ***************************************************************
                        TEST FOR STATE ROW SUMS = 1. AND POINT ROW SUMS = 0.
                    ***************************************************************
                    EXIT2=0                                              TEST0130    *013
                    THROUGH TES3, FOR I=1,1,I.G.SCOPE                    TEST0140    *014
                    WHENEVER V(CVFLAG,I).E.0.                            TEST0150    *015    01
                    TRANSFER TO TES3                                     TEST0160    *016    01
                    OR WHENEVER .ABS.(1-V(CVFLAG,I)).LE.1.E-7            TEST0170    *017    01
                    TRANSFER TO TES3                                     TEST0180    *018    01
                    OTHERWISE                                            TEST0190    *019    01
                    EXIT2=EXIT2+1                                        TEST0200    *020    01
                    WHENEVER EXIT2.E.1,EXECUTE TOUT2.                    TEST0210    *021    01
                    EXECUTE TOUT2.                                       TEST0220    *022    01
                    END OF CONDITIONAL                                   TEST0230    *023    01
TES3                CONTINUE                                             TEST0240    *024    01
                    ***************************************************************
                        TEST FOR POINT COLUMNS = 0
                    ***************************************************************
                    EXIT3=0                                              TEST0251    *025
                    THROUGH TES5, FOR I=1,1,I.G.NTRANS                   TEST0252    *026
                    THROUGH TES4, FOR J=DEST(I),DELTA,J.G.LAST(I)-FIRST(I)+DEST(I  TEST0253  *027  01
                   1 )                                                   TEST0254    *027
```

```
      WHENEVER V(CVFLAG,J).NE.0,TRANSFER TO TES4                        TEST0255   *028   02
      EXIT3 = EXIT3+1                                                   TEST0256   *029   02
      WHENEVER EXIT3.E.1,EXECUTE TOUTM.                                 TEST0257   *030   02
      EXECUTE TOUT1.                                                    TEST0258   *031   02
      TRANSFER TO TES5                                                  TEST0259   *032   02
TES4  CONTINUE                                                          TEST0260   *033   02
TES5  CONTINUE                                                          TEST0261   *034   01

      WHENEVER EXIT1.NE.0.OR.EXIT2.NE.0.OR.EXIT3.NE.0, EXECUTE          TEST0262   *035
     1 TOUT3.                                                           TEST0263   *035
      FUNCTION RETURN                                                   TEST0264   *036

      INTEGER CVFLAG,EXIT1,I,NTRANS,J,FIRST,DELTA,LAST,EXIT2,SCOPE,     TEST0280   *037
     1 VFLAG,NITER,DEST,MXITER,INIT,PCH,MAXJ,EXIT3                      TEST0290   *037
      EQUIVALENCE (TRANS,NTRANS)                                        TEST0300   *038
      PROGRAM COMMON TRANS(2001),FIRST(2000),DEST(2000),LAST(2000),     TEST0310   *039
     1 PCH,PRSW,MXITER,EPSI,DELTA,INIT,RUN,TSW,NAME(1),MAXJ             TEST0320   *039
      ERASABLE DUMMY(100),VFLAG,CVFLAG,SCOPE,NITER,V((0...1)*5000),     TEST0340   *040
     1 ARATEM,VECSUM,EXIT1,EXIT2,EXIT3,I,J,DATE,YEAR,DIADD              TEST0345   *040
      END OF FUNCTION                                                   TEST0360   *041
```

MAD (09 AUG 1965 VERSION) PROGRAM LISTING ... ...

```
      REFERENCES ON                                                          *001

      EXTERNAL FUNCTION                                          TOUT0010    *001
      ENTRY TO TOUTT.                                            TOUT0020    *002
      PRINT COMMENT  $1                THE FOLLOWING QUADRUPLES HAVE         TOUT0030    *003
     1 AN ERROR IN FIRST OR LAST.  J IS FIRST + N*DELTA, WHERE N IS          TOUT0040    *003
     2 $                                                                     TOUT0045    *003
      PRINT COMMENT $                 AN INTEGER.$                           TOUT0046    *004
      PRINT FORMAT TITLET                                                    TOUT0050    *005
      FUNCTION RETURN                                                        TOUT0060    *006

      ENTRY TO TOUTM.                                            TOUT0061    *007
      PRINT COMMENT  $1                THE FOLLOWING QUADRUPLES HAVE         TOUT0062    *008
     1 AN ERROR IN DEST.  J IS A POINT INTO WHICH TRANSITION IS INDI          TOUT0063    *008
     2 CATED.$                                                               TOUT0064    *008
      PRINT FORMAT TITLET                                                    TOUT0065    *009
      FUNCTION RETURN                                                        TOUT0066    *010

      ENTRY TO TOUT1.                                            TOUT0070    *011
      PRINT FORMAT OUT1,I,TRANS(I),FIRST(I),DEST(I),LAST(I),J                TOUT0080    *012
      FUNCTION RETURN                                                        TOUT0090    *013

      ENTRY TO TOUTC.                                            TOUT0100    *014
      PRINT COMMENT $1                 THE FOLLOWING ROW SUMS, SUM (I)       TOUT0110    *015
     1 , OF THE STOCHASTIC MATRIX ARE INCORRECT.$                            TOUT0120    *015
      PRINT FORMAT TITLEC                                                    TOUT0130    *016
      FUNCTION RETURN                                                        TOUT0140    *017

      ENTRY TO TOUT2.                                            TOUT0150    *018
      PRINT FORMAT OUT2,I,V(CVFLAG,I)                            TOUT0160    *019
      FUNCTION RETURN                                            TOUT0170    *020

      ENTRY TO TOUT3.                                            TOUT0180    *021
      PRINT COMMENT $1                 THE ABOVE IS A FATAL ERROR IN T       TOUT0190    *022
     1 HE TRANSITION TABLE$                                                  TOUT0195    *022
      PRINT FORMAT $1HC,S14,H+IF ((U)) IS THE MATRIX DESCRIBED BY D          TOUT0200    *023
     1 ISCPL., THEN THIS TABLE DESCRIBES+,E14.7,H'*((U)) +',E14.7,           TOUT0201    *023
     2 H'*(((I))'*$,1./ARATEM,DIADD                                          TOUT0202    *023
      PRINT FORMAT TITLED                                                    TOUT0220    *024
      THROUGH TU, FOR I=1,1,I.G.NTRANS                                       TOUT0230    *025
TU    PRINT FORMAT OUT1,I,TRANS(I),FIRST(I),DEST(I),LAST(I)                  TOUT0240    *026   01
      WHENEVER SCOPE .G.500,TRANSFER TO TUO                                  TOUT0245    *027

      PRINT COMMENT $1                 THE FOLLOWING ARE THE ROW SUMS        TOUT0250    *028
     1 OF THE STOCHASTIC MATRIX $                                            TOUT0260    *028
      PRINT COMMENT$8$                                                       TOUT0261    *029
      THROUGH TUT, FOR I=1,8,I+7.G.SCOPE                                     TOUT0270    *030
TUT   PRINT FORMAT OUT4,I,I+7,V(CVFLAG,I)...V(CVFLAG,I+7)                    TOUT0280    *031   01
      WHENEVER I.LE.SCOPE,PRINT FORMAT OUT4,I,SCOPE,V(CVFLAG,I)...V          TOUT0285    *032
     1 (CVFLAG,SCOPE)                                                        TOUT0286    *032

TUO   EXECUTE ERROR.                                             TOUT0290    *033
```

```
      VECTOR VALUES TITLET=$1H8,S15,1HI,S8,8HTRANS(I),S11,8HFIRST(I      TOUT0380    *034
     1 ),S8,7HDEST(I),S9,7HLAST(I),S11,1HJ/1H *$                         TOUT0390    *034
      VECTOR VALUES OUT1=$I17,F15.8,4I16*$                               TOUT0400    *035
      VECTOR VALUES TITLEC=$1H8,S15,1HI,S9,6HSUM(I)/1H *$                TOUT0410    *036
      VECTOR VALUES OUT2=$S14,I4,S4,F11.8*$                              TOUT0420    *037
      VECTOR VALUES TITLED=$1H8,S15,1HI,S8,8HTRANS(I),S11,8HFIRST(I      TOUT0430    *038
     1 ),S8,7HDEST(I),S9,7HLAST(I)/1H *$                                 TOUT0440    *038
      VECTOR VALUES OUT4=$S14,I4,3H - ,I4,S3,8(F8.6,S3)/1H *$            TOUT0460    *039

      INTEGER FIRST,DEST,LAST,I,J,CVFLAG,NTRANS,VFLAG,DELTA,SCOPE,       TOUT0490    *040
     1 NITER,MXITER,INIT,PCH,MAXJ,EXIT1,EXIT2,EXIT3                      TOUT0500    *040
      EQUIVALENCE (TRANS,NTRANS)                                         TOUT0510    *041
      PROGRAM COMMON TRANS(2001),FIRST(2000),DEST(2000),LAST(2000),      TOUT0520    *042
     1 PCH,PRSW,MXITER,EPSI,DELTA,INIT,RUN,TSW,NAME(1),MAXJ              TOUT0530    *042
      ERASABLE DUMMY(100),VFLAG,CVFLAG,SCOPE,NITER,V((0...1)*5000),      TOUT0540    *043
     1 ARATEM,VECSUM,EXIT1,EXIT2,EXIT3,I,J,DATE,YEAR,DIADD               TOUT0550    *043
      FUNCTION RETURN                                                    TOUT0590    *044
      END OF FUNCTION                                                    TOUT0600    *045
```

MAD (03 JAN 1966 VERSION) PROGRAM LISTING ... ...

```
                REFERENCES ON                                                   *C01

                EXTERNAL FUNCTION                                   ESTM0010    *001
                ENTRY TO ESTIM.                                     ESTM0020    *CC2
                LAMSTR=0.                                           ESTM0030    *CC3
                MUSTR=0.                                            ESTM0040    *C04
                T2=0                                                ESTM0050    *005
                T3=0                                                ESTM0060    *006
                THROUGH ES1, FOR I=1,1,I.G.NTRANS                   ESTM0070    *007
                DIFSTR=FIRST(I)-DEST(I)                             ESTM0080    *008    01
                T1=((LAST(I)-FIRST(I))/DELTA)+1                     ESTM0090    *C09    01
                WHENEVER DIFSTR.E.0,TRANSFER TO ES1                 ESTM0100    *010    01
                WHENEVER DIFSTR.G.0,MUSTR=MUSTR+TRANS(I)*T1         ESTM0110    *C11    01
                WHENEVER DIFSTR.L.0,LAMSTR=LAMSTR+TRANS(I)*T1       ESTM0120    *012    01
                T2=T2+.ABS.DIFSTR*T1                                ESTM0130    *013    01
                T3=T3+T1                                            ESTM0140    *014    01
ES1             CONTINUE                                            ESTM0150    *C15    01
                RHOSTR=(LAMSTR/MUSTR).P.((1.*T3)/T2)                ESTM0160    *016

               ************************************************************

                    ESTIMATE ONLY VALUES WHICH ARE LARGER THAN DESIRED
                    ACCURACY WHEN NORMALIZED.

               ************************************************************

                T1=1                                                ESTM0170    *017
                T2=SCOPE                                            ESTM0180    *018
                ESCOPE=.ABS.(ELOG.(EPSI)/ELOG.(RHOSTR))             ESTM0190    *019
                WHENEVER ESCOPE.GE.SCOPE,TRANSFER TO ES2            ESTM0200    *C20
                WHENEVER RHOSTR.GE.1.,T1=SCOPE-ESCOPE               ESTM0210    *021
                WHENEVER RHOSTR.L.1.,T2=ESCOPE                      ESTM0220    *022
                EXECUTE ZERO.(V(1)...V(SCOPE))                      ESTM0230    *023
ES2             RHON=1.                                             ESTM0240    *024
                CVFLAG=1                                            ESTM0250    *025
                VFLAG=0                                             ESTM0260    *026
                THROUGH ES3, FOR I=T1,1,I.G.T2                      ESTM0270    *027
                RHON=RHON*RHOSTR                                    ESTM0280    *028    01
ES3             V(I)=RHON                                           ESTM0290    *029    01

               ************************************************************

                    ZERO OUT VALUES AT POINTS BY PERFORMING AN ITERATION

               ************************************************************

                EXECUTE ITER.(1)                                    ESTM0300    *030
                FUNCTION RETURN                                     ESTM0310    *C31

                INTEGER T1,T2,T3,ESCOPE,DIFSTR,I,NTRANS,FIRST,DEST,LAST,    ESTM0350    *032
              1 DELTA,K,J,CVFLAG,VFLAG,SCOPE,NITER,MXITER,INIT,PCH,MAXJ     ESTM0360    *032
                EQUIVALENCE (TRANS,NTRANS)                          ESTM0370    *C33
```

```
PROGRAM COMMON TRANS(2001),FIRST(2000),DEST(2000),LAST(2000),          ESTM0380      *034
PCH,PRSW,MXITER,EPSI,DELTA,INIT,RUN,TSW,NAME(1),MAXJ                   ESTM0390      *034
ERASABLE DUMMY(100),VFLAG,CVFLAG,SCOPE,NITER,V((0...1)*5000),          ESTM0410      *035
ARATEM,VECSUM,EXIT1,EXIT2,EXIT3,I,J,DATE,YEAR,DIADD                    ESTM0420      *035
END OF FUNCTION                                                        ESTM0430      *036
```

PROGRAM LENGTH (OCTAL) 00066 PROGRAM COMMON LENGTH 17520 LOWEST ERASABLE DEFINED 54206 ID ITER0000
INITIAL TRANSFER VECTOR 00000 TRANSFER VECTOR LENGTH 00000

PROGRAM ENTRY POINT LOCATIONS AND NAMES
00000 ITER

PROGRAM LISTING (NUMBERS IN OCTAL)

```
                                                  ENTRY    ITER
00000  0634 00 4 00060  010   ITER    SXA      XR4,4          ID ITER0001
00001  0500 60 4 00001  00            CLA*     1,4
00002  0601 00 0 00064  010           STO      N
00003  0500 00 0 27511  00            CLA      DELTA
00004  0767 00 0 00022  00            ALS      18
00005  0622 00 0 00044  010           STD      D1
00006  0622 00 0 00045  010           STD      D2
                                *
00007  0534 00 1 77632  010   START   LXA      VFLAG,1
00010  0500 00 1 00063  010           CLA      AD2,1
00011  0621 00 0 00042  010           STA      RV1
00012  0621 00 0 00043  010           STA      RV2
00013  0621 00 0 00025  010           STA      RV3
00014  0500 00 0 77632  010           CLA      VFLAG
00015  0322 00 0 00065  010           ERA      =1
00016  0734 00 1 00000  00            PAX      ,1
00017  0500 00 1 00063  010           CLA      AD2,1
00020  0621 00 0 00040  010           STA      OPV1
00021  0500 00 0 77627  010           CLA      NITER
00022  0400 00 0 00065  010           ADD      =1
00023  0601 00 0 77627  010           STO      NITER
00024  0534 00 1 77630  010           LXA      SCOPE,1        ID ITER0002
00025  0600 00 1 00000  00    RV3     STZ      **,1
00026  2 00001 1 00025  010           TIX      *-1,1,1
                                *
00027  0534 00 4 13721  00            LXA      TRANS,4
00030  0500 00 4 27504  00    K1      CLA      LAST,4
00031  0767 00 0 00022  00            ALS      18
00032  0622 00 0 00037  010           STD      L1
00033  0500 00 4 17642  00            CLA      FIRST,4
00034  0734 00 2 00000  00            PAX      ,2
00035  0500 00 4 23563  00            CLA      DEST,4
00036  0734 00 1 00000  00            PAX      ,1
                                *
00037  3 00000 2 00046  010   L1      TXH      L2,2,**
00040  0560 00 2 00000  00    OPV1    LDQ      **,2
00041  0260 00 4 13721  00    F1      FMP      TRANS,4
00042  0300 00 1 00000  00    RV1     FAD      **,1
00043  0601 00 1 00000  00    RV2     STO      **,1
00044  1 00000 1 00045  010   D1      TXI      *+1,1,**
00045  1 00000 2 00037  010   D2      TXI      L1,2,**
                                *
00046  2 00001 4 00030  010   L2      TIX      K1,4,1
                                *
00047  0500 00 0 77632  010           CLA      VFLAG
00050  0601 00 0 77631  010           STO      CVFLAG         ID ITER0003
00051  0322 00 0 00065  010           ERA      =1
```

```
      00052  0601 00 0 77632  010           STO     VFLAG
      00053  0500 00 0 00064  010           CLA     N
      00054  0402 00 0 00065  010           SUB     =1
      00055  0601 00 0 00064  010           STO     N
      00056  0520 00 0 00064  010           ZET     N
      00057  0020 00 0 00007  010           TRA     START
      00060  0774 00 4 00000  00    XR4     AXT     **,4
      00061  0020 00 4 00002  00            TRA     2,4
      00062  0 00000 0 77626  010    AD1     PZE     V
      00063  0 00000 0 66016  010    AD2     PZE     V-5000
      00064  0 00000 0 00000  00     N       PZE
                          13721             TRANS   PGMCOM  2001
                          17642             FIRST   PGMCOM  2000
                          23563             DEST    PGMCOM  2000
                          27504             LAST    PGMCOM  2000
                          27510                     PGMCOM  3
                                                    PCLIST  DELTA
                          27511             DELTA
                          27517                     PGMCOM  5
                                            *
                          77632                     ERAS    100
                                                    ERLIST  VFLAG,CVFLAG,SCOPE,NITER
                          77632             VFLAG
                          77631             CVFLAG
                          77630             SCOPE
                          77627             NITER
                          77626             V       ERAS    1000
                                                    END

PROGRAM LITERALS
      00065  000000000001      00

**** 00066 IS THE FIRST LOCATION NOT USED BY THIS PROGRAM.

REFERENCES TO SYMBOLS DEFINED IN PROGRAM.
TYPE SYMBOL VALUE REFERENCES TO SYMBOL   (SYMBOL TYPES -- R - RELOCATABLE, A - ABSOLUTE, E - ERASABLE, T - TRANSFER VECTOR)
 R   AD1    00062 00062
 R   AD2    00063 00010,00017,00063
 E   CVFLAG 77631 00050,00065
 R   D1     00044 00005,00044
 R   D2     00045 00006,00045
 A   DELTA  27511 00003,00065
 A   DEST   23563 00035,00065
 R   FI     00041 00041
 A   FIRST  17642 00033,00065
 R   ITER   00000 00000,00000,00000
 R   K1     00030 00030,00046
 R   L1     00037 00032,00037,00045
 R   L2     00046 00037,00046
 A   LAST   27504 00030,00065
 R   N      00064 00002,00053,00055,00056,00064
 E   NITER  77627 00021,00023,00065
 R   OPV1   00040 00020,00040
 R   RV1    00042 00011,00042
 R   RV2    00043 00012,00043
 R   RV3    00025 00013,00025
 E   SCOPE  77630 00024,00065
 R   START  00007 00007,00057
```

```
 A    TRANS   13721 00027,00041,00065
 E    V       77626 00062,00063,00065
 E    VFLAG   77632 00007,00014,00047,00052,00065
 R    XR4     00060 00000,00060

REFERENCES TO PROGRAM LITERALS
  LOC  REFERENCES
00065 00015,00022,00051,00054

FLAGGED LINES  FATAL   0      NON-FATAL 0      ****  DEFINITIONS    SYMBOL 26     OPERATION 0     MACRO 0      RMT 0
MACRO CALLS    CORRECT 0      INCORRECT 0      ****  CARD COUNTS    INPUT  71     OUTPUT    4
```

MAD (09 AUG 1965 VERSION) PROGRAM LISTING ... ...

```
                REFERENCES ON                                                   *001

                EXTERNAL FUNCTION                                   CVGT0001    *001
                ENTRY TO CVGTST.                                    CVGT0002    *002

                **************************************************************

                    IF NITER = 0, SET TRAP PROCEDURE SO THAT UNDERFLOWS
                    PRODUCE ZEROS.  ALSO FIND SUITABLE INDEX FOR WEAK TEST.

                **************************************************************

                VECTOR VALUES FTSW=0                                CVGT0003    *003
                WHENEVER FTSW.E.0                                   CVGT0004    *004
                WHENEVER NITER .GE. MXITER                          CVGT0005    *005    01
                MAXDIF=0.                                           CVGT0006    *006    02
                TRANSFER TO CV3                                     CVGT0007    *007    02
                END OF CONDITIONAL                                  CVGT0008    *008    02
                FTSW=1                                              CVGT0014    *009    01
                EXECUTE FTRAP.                                      CVGT0015    *010    01
CV0             THROUGH CV0, FOR I=1,1,V(VFLAG,I).G.0..OR.I.E.SCOPE CVGT0016    *011    01
                FUNCTION RETURN I                                   CVGT0017    *012    01
                END OF CONDITIONAL                                  CVGT0018    *013    01

                **************************************************************

                    FIND MAXIMUM DIFFERENCE AND TEST AGAINST EPSI.

                **************************************************************

                MAXDIF=0.                                           CVGT0030    *014
                THROUGH CV2, FOR I=1,1,I.G.SCOPE                    CVGT0040    *015
                DIF=.ABS.(V(VFLAG,I)-V(CVFLAG,I))                   CVGT0050    *016            01
                WHENEVER MAXDIF.G.DIF, TRANSFER TO CV2              CVGT0060    *017            01
                MAXDIF=DIF                                          CVGT0070    *018            01
                MIX=I                                               CVGT0080    *019            01
CV2             CONTINUE                                            CVGT0090    *020            01

                WHENEVER MAXDIF .LE. EPSI                           CVGT0100    *021
                PRINT COMMENT $8          ****************************** SUCC  CVGT0105    *022    01
              1 ESSFUL RUN ******************************$                      CVGT0110    *022
                TRANSFER TO CV1                                     CVGT0115    *023    01

                OR WHENEVER NITER .GE. MXITER                       CVGT0120    *024    01
CV3             PRINT COMMENT $8********** THE SPECIFIED NUMBER OF ITERATIONS CVGT0125    *025    01
                1  HAS BEEN SATISFIED ********** $                  CVGT0130    *025

                **************************************************************

                    ESTIMATE SUBDOMINANT EIGENVALUE AND ERROR.

                **************************************************************
```

```
CV1     LDIF=MAXDIF.P.(1./NITER)                                        CVGT0131    *026    01
        WHENEVER LDIF.GE.1..OR.LDIF.L..5                                CVGT0132    *027    01
        PRINT FORMAT CVFL,EPSI,MXITER,NITER,MAXDIF,MAXDIF,0.            CVGT0133    *028    02
        OR WHENEVER (1.-LDIF).P.(1./NITER).L.LDIF                       CVGT0134    *029    02
        PRINT FORMAT CVFL,EPSI,MXITER,NITER,MAXDIF,0.,LDIF              CVGT0135    *030    02
        OTHERWISE                                                       CVGT0136    *031    02
        PRINT FORMAT CVFL,EPSI,MXITER,NITER,MAXDIF,MAXDIF/LDIF,LDIF     CVGT0137    *032    02
        END OF CONDITIONAL                                              CVGT0138    *033    02

        ***************************************************************

            RESET TRAP PROCEDURE SO THAT UNDERFLOWS AGAIN CAUSE
            ERROR RETURNS.

        ***************************************************************

        EXECUTE NTRAP.                                                  CVGT0139    *034    01
        FTSW=0                                                          CVGT0140    *035    01

        EXECUTE OUT2.                                                   CVGT0145    *036    01

        ***************************************************************

            THE ERROR RETURN TRANSFERS TO 'BEGIN' IN THE MAIN PROGRAM

        ***************************************************************

        ERROR RETURN                                                    CVGT0150    *037    01

        OTHERWISE                                                       CVGT0155    *038    01
        FUNCTION RETURN MIX                                             CVGT0160    *039    01
        END OF CONDITIONAL                                              CVGT0165    *040    01

        VECTOR VALUES CVFL=$1H-,S14,35HCONVERGENCE TEST CRITERION (EP   CVGT0170    *041
      1 SI) =F11.8/1HO,S12,37HMAXIMUM ALLOWED ITERATIONS (MXITER) =I5   CVGT0180    *041
      2 /1HO,S31,18HITERATIONS TAKEN =I5/1HO, S3,46HMAXIMUM DIFFERENC   CVGT0182    *041
      3 E BETWEEN LAST TWO ITERATES =E8.1/1HO, S23,26HESTIMATED ORDER    CVGT0184    *041
      4  OF ERROR =E8.1,S3,33H(APPROXIMATE CONVERGENCE FACTOR =F9.7,     CVGT0186    *041
      5 1H)*$                                                           CVGT0180    *041

        INTEGER I,SCOPE,VFLAG,CVFLAG,MIX,DELTA,NITER,FIRST,DEST,LAST,   CVGT0185    *042
      1 MXITER,INIT                                                     CVGT0186    *042
        PROGRAM COMMON TRANS(2001),FIRST(2000),DEST(2000),LAST(2000),   CVGT0190    *043
      1 PCH,PRSW,MXITER,EPSI,DELTA,INIT,RUN,TSW,NAME(1),MAXJ            CVGT0200    *043
        ERASABLE DUMMY(100),VFLAG,CVFLAG,SCOPE,NITER,V((0...1)*5000),   CVGT0205    *044
      1 ARATEM,VECSUM,EXIT1,EXIT2,EXIT3,I,J,DATE,YEAR,DIADD             CVGT0210    *044
        END OF FUNCTION                                                 CVGT0290    *045
```

```
$COMPILE MAD,EXECUTE,DUMP,I/O DUMP,PRINT OBJECT,PUNCH OBJECT    OUTT0000    001632  09/15/65    6 18 16.2 PM

MAD (09 AUG 1965 VERSION) PROGRAM LISTING ... ... ...


            REFERENCES ON                                                          *001

            EXTERNAL FUNCTION                                          OJTT0010    *001
            ENTRY TO OUT2.                                             OJTT0020    *002
            WHENEVER PCH.NE.1,TRANSFER TO UT9                          OJTT0300    *003
            PRINT COMMENT $-   THE SOLUTION VECTOR HAS BEEN PUNCHED$   OJTT0320    *004

            **************************************************************

                OUTPUT VECTOR CARDS WILL HAVE THE FOLLOWING ID IN
                COLUMNS 73-80 .
               (1) MODEL NAME
               (2) DATE
               (3) RUN NO. XX 001
               ALL SUCCEEDING CARDS WILL HAVE RUN NO. XX VECTOR CARD NO.

            **************************************************************

            PUNCH FORMAT $1H ,S14,2C6,S1,4HRUN ,I3,2H, ,2C6,S23,C6,C2*$,   OJTT0351    *005
          1 NAME,NAME(1),RUN,DATE,YEAR,NAME,NAME(1)                        OJTT0352    *005
            EXECUTE NSDBCD.(DATE,$  $)                                     OUTT0354    *006
            EXECUTE BPUNCH.(SCOPE)                                         OJTT0355    *007
            EXECUTE NSDBCD.((BNBCD.(RUN).LS.18).V.$000  0$,$01$)           OJTT0356    *008
            EXECUTE BPUNCH.(V(VFLAG,1)...V(VFLAG,SCOPE))                   OUTT0357    *009

UT9         WHENEVER PRSW.E.1,EXECUTE PROUT.                               OJTT0370    *010

            RUN = RUN + 1                                                  OJTT0380    *011

            INTEGER NITER,MXITER,I,SCOPE,J,K,DELTA,VFLAG,CVFLAG,FIRST,DES  OUTT0430    *012
          1 T,LAST,INIT,MAXJ,S,RUN,DATE,YEAR,NAME,BNBCD.,PCH,PRSW          OJTT0440    *012
            PROGRAM COMMON TRANS(2001),FIRST(2000),DEST(2000),LAST(2000),  OJTT0470    *013
          1 PCH,PRSW,MXITER,EPSI,DELTA,INIT,RUN,TSW,NAME(1),MAXJ           OJTT0480    *013
            ERASABLE DUMMY(100),VFLAG,CVFLAG,SCOPE,NITER,V((0...1)*5000),  OJTT0495    *014
          1 ARATEM,VECSUM,EXIT1,EXIT2,EXIT3,I,J,DATE,YEAR,DIADD            OJTT0496    *014
            FUNCTION RETURN                                                OUTT0510    *015
            END OF FUNCTION                                                OUTT0520    *016
```

```
                REFERENCES ON                                                    *001

                EXTERNAL FUNCTION                                  POUT0010      *001
                ENTRY TO PROUT.                                    POUT0020      *002

               ***************************************************************

                           PROBABILITY VECTOR PRINTED OUT

               ***************************************************************

                PRINT FORMAT $1H1,S20,H+THE FOLLOWING ARE THE LIMITING STATE       POUT0060      *003
              1 PROBABILITIES, PROB(I,J).+/S20,   H+ PROB(I) IS THE SUM OF PR      POUT0070      *003
              2 OB(I,J) OVER ALL STATES WITH THE VALUE I FOR THE FIRST VARIAB      POUT0080      *003
              3 LE+/1H-*$                                                          POUT0090      *003
                WHENEVER (SCOPE/MAXJ)+1.G.2500.OR.MAXJ.G.2500                      POUT0095      *004
                PRINT COMMENT $      *** MAXJ, THE MAXIMUM COLUMN INDEX HAS B     POUT0100      *005      01
              1 EEN SET TO FIFTY TO ALLOW MORE EFFICIENT PRINTING OF THE PROB      POUT0105      *005
              2 ABILITY VECTOR.$                                                   POUT0106      *005
                MAXJ=50                                                            POUT0110      *006      01
                END OF CONDITIONAL                                                 POUT0115      *007      01
                PRINT FORMAT $1H-,S3,1HI,S4,8H PROB(I),S40,9HPROB(I,J)/1H-,        POUT0120      *008
              1 S19,3HJ =,S2,10(I1,S10)*$,(I=0,1,I.G.9.OR.I.G.(MAXJ-1),I)          POUT0121      *008
                VECSUM=0.                                                          POUT0122      *009
                EXECUTE ZERO.(MARG1...MARG1((SCOPE/MAXJ)+1),MARG2...MARG2(MAX      POUT0127      *010
              1 J))                                                                POUT0128      *010
                THROUGH UTTA, FOR K=1,MAXJ,K.G.SCOPE                               POUT0130      *011
                THROUGH UTTB, FOR S=K,1,S.G.(K+MAXJ-1).OR.S.G.SCOPE                POUT0140      *012                01
                I=(S-1)/MAXJ                                                       POUT0150      *013                02
                J=S-1-I*MAXJ                                                       POUT0160      *014                02
                MARG1(I)=MARG1(I)+V(VFLAG,S)                                       POUT0170      *015                02
UTTB            MARG2(J)=MARG2(J)+V(VFLAG,S)                                       POUT0180      *016                02
                VECSUM=VECSUM+MARG1(I)                                             POUT0190      *017                01
                WHENEVER (K+MAXJ-1).G.SCOPE                                        POUT0200      *018                01
                PRINT FORMAT LINE,I,MARG1(I),V(VFLAG,K)...V(VFLAG,SCOPE)           POUT0210      *019      01        01
                OTHERWISE                                                          POUT0220      *020      01        01
                PRINT FORMAT LINE,I,MARG1(I),V(VFLAG,K)...V(VFLAG,K+MAXJ-1)        POUT0230      *021      01        01
UTTA            END OF CONDITIONAL                                                 POUT0240      *022      01        01
                VECTOR VALUES LINE=$1HO,I4,S2,F10.8,S2,10(F10.8,S1)/(              POUT0250      *023
              1 S19,10(F10.8,S1))*$                                                POUT0260      *023
                PRINT FORMAT $1H8,S15,H+ PROB(J) IS THE SUM OF PROB(I,J) OVER      POUT0270      *024
              1  ALL STATES WITH THE VALUE J FOR THE SECOND VARIABLE+/1H-,          POUT0280      *024
              2 S8,8HCHECKSUM,S40,7HPROB(J)/1HO,S19,3HJ =,S2,10(I1,S10)*$,         POUT0290      *024
              3 (I=0,1,I.G.9.OR.I.G.(MAXJ-1),I)                                    POUT0291      *024
                PRINT FORMAT $1HO,S6,F10.8,S2,10(F10.8,S1)/(S19,10(F10.8,          POUT0310      *025
              1 S1))*$,VECSUM,MARG2(0)...MARG2(MAXJ-1)                             POUT0320      *025

                INTEGER NITER,MXITER,MAXJ,SCOPE,K,I,J,VFLAG,CVFLAG,S               POUT0330      *026
                DIMENSION MARG1(2500),MARG2(2500)                                  POUT0340      *027
                EQUIVALENCE (MARG1,FIRST),(MARG1(2001),TRANS),(MARG2,LAST)         POUT0345      *028
              1 ,(MARG2(2001),DEST)                                                POUT0346      *028
                PROGRAM COMMON TRANS(2001),FIRST(2000),DEST(2000),LAST(2000),      POUT0350      *029
```

```
1 PCH,PRSW,MXITER,EPSI,DELTA,INIT,RUN,TSW,NAME(1),MAXJ
  ERASABLE DUMMY(100),VFLAG,CVFLAG,SCOPE,NITER,V((0...1)*5000),
1 ARATEM,VECSUM,EXIT1,EXIT2,EXIT3,I,J,DATE,YEAR,DIADD
  FUNCTION RETURN
  END OF FUNCTION
```

```
POUT0360        *029
POUT0380        *030
POUT0390        *030
POUT0400        *031
POUT0410        *032
```

MAC (09 AUG 1965 VERSION) PROGRAM LISTING ... ...

```
                REFERENCES ON                                                   ITER0010    *001
                EXTERNAL FUNCTION (N)                                           ITER0020    *001
                ENTRY TO ITER.                                                  ITER0030    *002
                NORMAL MODE IS INTEGER                                          ITER0040    *003
                FLOATING POINT V,TRANS                                          ITER0050    *004
                THROUGH END, FOR CNTR=1,1,CNTR.G.N                              ITER0060    *005
                EXECUTE ZERO.(V(CVFLAG,1)...V(CVFLAG,SCOPE))                    ITER0070    *006    01
                NITER = NITER+1                                                 ITER0080    *007    01
                THROUGH KLOOP, FOR K=1,1,K.G.NTRANS                             ITER0090    *008    01
                J=FIRST(K)                                                      ITER0100    *009    02
                I=DEST(K)                                                       ITER0110    *010    02
COMPLP          V(CVFLAG,I)=V(VFLAG,J)*TRANS(K)+V(CVFLAG,I)                     ITER0120    *011    02
                I=I+DELTA                                                       ITER0130    *012    02
                J=J+DELTA                                                       ITER0140    *013    02
KLOOP           WHENEVER J.LE.LAST(K),TRANSFER TO COMPLP                        ITER0150    *014    02
                VFLAG=VFLAG.EV.1                                                ITER0160    *015    01
END             CVFLAG=CVFLAG.EV.1                                              ITER0170    *016    01
                PROGRAM COMMON TRANS(2001),FIRST(2000),DEST(2000),LAST(2000),   ITER0180    *017
              1 PCH,PRSW,MXITER,EPSI,DELTA,INIT,RUN,TSW,NAME(1),MAXJ            ITER0190    *017
              , ERASABLE DUMMY(100),VFLAG,CVFLAG,SCOPE,NITER,V((0...1)*5000),   ITER0210    *018
              1 ARATEM,VECSUM,EXIT1,EXIT2,EXIT3,I,J,DATE,YEAR,DIADD             ITER0220    *018
                EQUIVALENCE (NTRANS,TRANS)                                      ITER0230    *019
                FUNCTION RETURN                                                 ITER0240    *020
                END OF FUNCTION                                                 ITER0250    *021
```

APPENDIX B

SOME SUBROUTINES FOR USE IN GENERATING QUADRUPLES IN RQA-1

James H. Jackson

B.1 PURPOSE

A three-entry subroutine has been written in an attempt to simplify the programming of discipline subroutines for use with RQA-1. The entries and their functions are as follows:

SETUP - Initializes parameters internal to the subroutine

QUAD - Produces the quadruples which represent transitions between states (the off-diagonal elements of the Q matrix)

DIAG - Produces the quadruples which represent the diagonal elements of the Q matrix.

B.2 CALLING SEQUENCES (MAD)

SETUP. (STATE.)

STATE is the name of an integer-valued internal function whose value is the state number represented by the current values of the state variables. Its values may range from 1 through 5,000.

QUAD. (RATE, CHANGE, REPEAT)

RATE is a floating point expression whose value is the transition rate for the generated quadruple.

CHANGE is an integer expression whose value is the amount by which the state number is changed when a transition indicated by the generated quadruple occurs.

REPEAT is an integer expression chosen such that the generated quadruple will apply to STATE., STATE. +DELTA,..., STATE.+REPEAT*DELTA.

DIAG.

This entry has no arguments.

B.3 RESTRICTIONS

SETUP must be called before either QUAD or DIAG is called, otherwise an error comment will be printed and control will be returned to the system.

MAXJ must be initialized before SETUP is called.

QUAD must be called as many times as needed to produce all of the quadruples which represent transitions between states. The last call to QUAD must occur before DIAG is called.

After DIAG is called, no more calls should be made to the subroutine until RQA-1 has processed the generated data.

B.4 DISCIPLINE PROGRAM STRUCTURE

The discipline program which calls SETUP, QUAD, and DIAG may have the following structure:

```
EXTERNAL FUNCTION
ENTRY TO DISCPL.
READ AND PRINT DATA
MAXJ = ----
DELTA = ----
SETUP.(STATE.)
THROUGH S, FOR ----
----
THROUGH S, FOR ----
WHENEVER ----, TRANSFER TO S
WHENEVER ----, QUAD.(----,----,----)
----
WHENEVER ----, QUAD.(----,----,----)
CONTINUE
DIAG.
FUNCTION RETURN
FLOATING POINT EPSI, ----
NORMAL MODE IS INTEGER
PROGRAM COMMON DUMMY (8004), PCH, PRSW, MXITER, EPSI,
1DELTA, INIT, RUN, TSW, NAME(1), MAXJ
VECTOR VALUES NAME = ----
INTERNAL FUNCTION STATE. (X) = ----
END OF FUNCTION
```

When the discipline subroutine is called, it reads a set of data. This data should include values of the parameters which describe the model as well as the values of PCH, PRSW, MXITER, EPSI, INIT, RUN, and TSW which are to be different from the preset values determined by RQA-1.

After the data is read, MAXJ and DELTA are initialized. Then SETUP is called to initialize internal parameters to the subroutine.

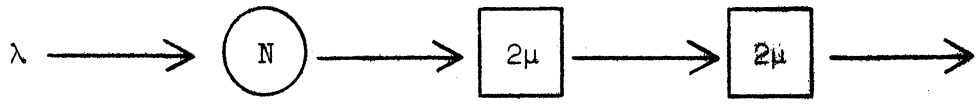
A set of nested interations is then begun. Each iteration variable is a state variable. Hence, within the innermost loop, all possible combinations of state variables will be produced.

Since impossible combinations of state variables may not always be eliminated by constraining the set of values each variable may assume, the first conditional statement shown specifies a transfer to the end of the loop when any constraint of the model is violated.

The remaining conditional statements generate quadruples which represent transitions between states whenever transitions may occur from the state represented by the current values of the state variables. When all of these quadruples have been generated, the diagonal elements of the Q matrix are generated, and control is returned to RQA-1.

B.5 EXAMPLE

As an illustration of the programming of a discipline routine using SETUP, QUAD, and DIAG, one might consider a second-order Erlang channel with a finite queue:



N = Maximum length of queue
λ = Arrival rate
μ = Processing rate

He may choose to define the state variables as follows:

A = Number of Erlang channels busy
B = Number of Erlang channels in second phase
C = Queue Length

The ranges of these state variables are the following:

$0 \leq A \leq 1$
$0 \leq B \leq A$
$0 \leq C \leq N$.

Since RQA-1 will print a two-dimensional probability array, the state space must be described in terms of two state variables to facilitate interpretation of the results. One may choose one of these two variables to be the state of

the service and the other to be the queue length. Since the state of the service is completely defined by A and B, it will be denoted by S(A,B), whereas the queue length is simply C.

S(A,B) may be defined by the following table:

A	B	S(A,B)
0	0	0
1	0	1
1	1	2

from this table, one may derive the following algebraic expression for S(A,B):

$$S(A,B) = A + B \quad .$$

Since S(A,B) has three possible values, and since the minimum values of the function STATE must be at least one, the value of STATE , S(A,B,C), may be expressed as follows:

$$S(A,B,C) = S(A,B) + 3C + 1 \ ,$$

or

$$S(A,B,C) = A + B + 3C + 1 \ .$$

We may now define all possible transitions for the model by the following table:

Condition	Event	Change	Rate
$0 \leq C \leq N$ $A = 1$ $B = 0$	B ↑	1	2μ
$1 \leq C \leq N$ $B = 1$	C ↓ B ↓	-4	2μ
$C = 0$ $B = 1$	A ↓ B ↓	-2	2μ
$C = 0$ $A = 0$	A ↑	1	λ
$0 \leq C \leq N-1$ $A = 1$	C ↑	3	λ

From this table, one can infer that many of the transitions are repeated for the various values of the queue length C. Consequently, C should be chosen to be the first index for the probability array, and DELTA should be chosen to be the number of values of S(A,B), or 3. MAXJ is then 3.

By applying the above table of possible transitions and the program form of Section 4, one may obtain the following discipline program for the model:

```
      EXTERNAL FUNCTION
      ENTRY TO DISCPL.
      READ AND PRINT DATA
      MAXJ = 3
      DELTA = 3
      SETUP.(STATE.)
      THROUGH S, FOR A = 0, 1, A. G. 1
      THROUGH S, FOR B = 0, 1, B. G. A.
      THROUGH S, FOR C = 0, 1, C. G. N. OR. C. G. 1
      WHENEVER C. G. 0. AND. A. E. 0, TRANSFER TO S
      WHENEVER C. E. 0. AND. A. E. 1. AND. B. E. 0, QUAD.(2.*MU,1,N)
      WHENEVER C. E. 1. AND. B. E. 1, QUAD.(2.*MU,-4,N-1)
      WHENEVER C. E. 0. AND. B. E. 1, QUAD.(2.*MU,-2,0)
      WHENEVER C. E. 0. AND. A. E. 0, QUAD.(LAMBDA,1,0)
      WHENEVER C. E. 0. AND. A. E. 1, QUAD.(LAMBDA,3,N-1)
S     CONTINUE
      DIAG.
      FUNCTION RETURN
      FLOATING POINT EPSI, MU, LAMBDA
      NORMAL MODE IS INTEGER
      PROGRAM COMMON DUMMY (8004), PCH, PRSW, MXITER, EPSI,
     1DELTA, INIT, RUN, TSW, NAME(1),MAXJ
      VECTOR VALUES NAME = $EXAMPLES$
      INTERNAL FUNCTION STATE.(X) = A + B + 3* C + 1
      END OF FUNCTION
```

PAGE 1 (MOD 05 MAY 1965) UNIVERSITY OF MICHIGAN 20 SEP 1965

PROGRAM LENGTH (OCTAL) 00352 PROGRAM COMMON LENGTH 17520 LOWEST ERASABLE DEFINED 77626 ID QUAD 000
INITIAL TRANSFER VECTOR 00000 TRANSFER VECTOR LENGTH 00002

PROGRAM ENTRY POINT LOCATIONS AND NAMES

00002	SETUP
00031	QUAD
00105	DIAG

EXTERNAL ROUTINES CALLED BY THIS PROGRAM

00000	334751314563	00	.PRINT
00001	255151465160	00	ERROR

ID QUAD 001

PROGRAM LISTING (NUMBERS IN OCTAL)

					ENTRY	SETUP,QUAD,DIAG
00002	0634 00 1 00025	010	SETUP	SXA		XRA,1
00003	0534 00 1 77631	010		LXA		Z,1
00004	0500 00 1 00250	010		CLA		ADR,1
00005	0621 00 0 00023	010		STA		VA
00006	0621 00 0 00075	010		STA		VB
00007	0621 00 0 00076	010		STA		VC
00010	0621 00 0 00122	010		STA		COND
00011	0621 00 0 00131	010		STA		VD
00012	0621 00 0 00132	010		STA		VE
00013	0621 00 0 00140	010		STA		HIGH
00014	0621 00 0 00142	010		STA		VF
00015	0400 00 0 00351	010		ADD		=1
00016	0621 00 0 00160	010		STA		VG
00017	0774 00 1 00001	00		AXT		1,1
00020	0634 00 1 13721	00		SXA		TRANS,1
00021	0634 00 1 00251	010		SXA		SW,1
00022	0774 00 1 11610	00		AXT		5000,1
00023	0600 00 1 00000	00	VA	STZ		**,1
00024	2 00001 1 00023	010		TIX		VA,1,1
00025	0774 00 1 00000	00	XRA	AXT		**,1
00026	0500 60 4 00001	00		CLA*		1,4
00027	0621 00 0 00044	010		STA		FN
00030	0020 00 4 00001	00		TRA		1,4
00031	0500 00 0 00251	010	QUAD	CLA		SW
00032	0402 00 0 00351	010		SUB		=1
00033	-0100 00 0 00171	010		TNZ		ERSETA
00034	0634 00 4 00045	010		SXA		XRB,4
00035	0634 00 2 00102	010		SXA		XRC,2
00036	0634 00 1 00103	010		SXA		XRD,1
00037	0534 00 2 13721	00		LXA		TRANS,2
00040	3 03720 2 00201	010		TXH		ERC,2,2000
00041	0500 60 4 00001	00		CLA*		1,4
00042	0601 00 2 13721	00		STO		TRANS,2
00043	0634 00 2 00046	010		SXA		XRB+1,2
00044	0074 00 4 00000	00	FN	TSX		**,4
00045	0774 00 4 00000	00	XRB	AXT		**,4
00046	0774 00 2 00000	00		AXT		**,2
00047	0340 00 0 00350	010		CAS		=5000
00050	0020 00 0 00212	010		TRA		ERD
00051	0020 00 0 00053	010		TRA		*+2
00052	-0120 00 0 00220	010		TMI		ERE

ID QUAD 002

ID QUAD 003

```
00053  0601 00 2 17642  00          STO     FIRST,2
00054  0400 60 4 00002  00          ADD*    2,4
00055  0601 00 2 23563  00          STO     DEST,2
00056  0402 60 4 00002  00          SUB*    2,4
00057  0601 00 0 77777  010         STO     TMP
00060  0560 60 4 00003  00          LDQ*    3,4
00061  0200 00 0 27511  00          MPY     DELTA
00062  0131 00 0 00000  00          XCA
00063  0400 00 0 77777  010         ADD     TMP
00064  0601 00 2 27504  00          STO     LAST,2
00065  0734 00 1 00000  00          PAX     ,1
00066 -0634 00 1 00073  010         SXD     THRU,1
00067  0534 00 1 27511  00          LXA     DELTA,1
00070 -0634 00 1 00077  010         SXD     INC,1
00071  0500 00 2 17642  00          CLA     FIRST,2
00072  0734 00 1 00000  00          PAX     ,1
00073  3 00000 1 00100  010  THRU    TXH     BQ,1,**
00074  0502 00 2 13721  00          CLS     TRANS,2
00075  0300 00 1 00000  00  VB      FAD     **,1
00076  0601 00 1 00000  00  VC      STO     **,1
00077  1 00000 1 00073  010  INC     TXI     THRU,1,**
00100  1 00001 2 00101  010  BQ      TXI     *+1,2,1
00101  0634 00 2 13721  00          SXA     TRANS,2
00102  0774 00 2 00000  00  XRC     AXT     **,2
00103  0774 00 1 00000  00  XRD     AXT     **,1
00104  0020 00 4 00001  00          TRA     1,4
00105  0500 00 0 00251  010 DIAG    CLA     SW
00106  0402 00 0 00351  010         SUB     =1
00107 -0100 00 0 00175  010         TNZ     ERSETB
00110  0634 00 1 00164  010         SXA     XRE,1
00111  0634 00 2 00165  010         SXA     XRF,2
00112  0634 00 4 00166  010         SXA     XRG,4
00113  0774 00 1 00001  00          AXT     1,1
00114  0534 00 4 27511  00          LXA     DELTA,4
00115 -0634 00 4 00127  010         SXD     BKD,4
00116 -0634 00 4 00156  010         SXD     BKZ,4
00117  0534 00 4 13721  00          LXA     TRANS,4
00120  3 03720 4 00201  010         TXH     ERC;4,2000
00121  0622 00 0 00130  010         STD     BKD+1
00122 -0520 00 1 00000  00  COND    NZT     **,1
00123  1 00001 1 00122  010         TXI     *-1,1,1
00124  0754 00 1 00000  00          PXA     ,1
00125  0734 00 2 00000  00          PAX     ,2
00126  3 11610 1 00164  010         TXH     XRE,1,5000
00127  1 00000 2 00130  010  BKD     TXI     *+1,2,**
00130  3 00000 2 00140  010         TXH     HIGH,2,**
00131  0500 00 1 00000  00  VD      CLA     **,1
00132  0302 00 2 00000  00  VE      FSB     **,2
00133  0760 00 0 00003  00          SSP
00134  0340 00 0 27510  00          CAS     EPSI
00135  0020 00 0 00140  010         TRA     HIGH
00136  0020 00 0 00127  010         TRA     BKD
00137  0020 00 0 00127  010         TRA     BKD
00140  0500 00 1 00000  00  HIGH    CLA     **,1
00141  0601 00 4 13721  00          STO     TRANS,4
00142  0600 00 1 00000  00  VF      STZ     **,1
00143  0754 00 1 00000  00          PXA     ,1
00144  0601 00 4 17642  00          STO     FIRST,4
```

ID QUAD 004

ID QUAD 005

ID QUAD 006

```
00145  0601 00 4 23563  00          STO     DEST,4
00146  0754 00 2 00000  00          PXA     ,2
00147  0402 00 0 27511  00          SUB     DELTA
00150  0601 00 4 27504  00          STO     LAST,4
00151  1 00001 4 00152  010         TXI     *+1,4,1
00152  3 03720 4 00201  010         TXH     ERC,4,2000
00153  1 00001 1 00154  010         TXI     *+1,1,1
00154 -0634 00 2 00157  010         SXD     STPZ,2
00155  0634 00 1 00162  010         SXA     ENDZ,1
00156  1 00000 1 00157  010  BKZ    TXI     STPZ,1,**
00157  3 00000 1 00162  010  STPZ   TXH     ENDZ,1,**
00160  0600 00 1 00000  00   VG     STZ     **,1
00161  0020 00 0 00156  010         TRA     BKZ
00162  0774 00 1 00000  00   ENDZ   AXT     **,1
00163  0020 00 0 00122  010         TRA     COND
00164  0774 00 1 00000  00   XRE    AXT     **,1
00165  0774 00 2 00000  00   XRF    AXT     **,2
00166  0774 00 4 00000  00   XRG    AXT     **,4
00167  0600 00 0 00251  010         STZ     SW
00170  0020 00 4 00001  00          TRA     1,4
00171  0074 00 4 00000  010  ERSETA PRINT   FA,0
00172 -1 00000 0 00252  010
00173 -1 00000 0 00000  00
00174  0074 00 4 00001  010         CALL    ERROR
00175  0074 00 4 00000  010  ERSETB PRINT   FB,0
00176 -1 00000 0 00261  010
00177 -1 00000 0 00000  00
00200  0074 00 4 00001  010         CALL    ERROR
00201  0774 00 1 03720  00   ERC    AXT     2000,1
00202  0074 00 4 00000  010         PRINT   FC
00203 -1 00000 0 00270  010
00204 -1 00000 1 13721  00          IOP     TRANS,1
00205 -1 00000 1 17642  00          IOP     FIRST,1
00206 -1 00000 1 23563  00          IOP     DEST,1
00207 -1 00000 1 27504  00          IOP     LAST,1
00210 -1 00000 0 00000  00          ENDIO
00211  0074 00 4 00001  010         CALL    ERROR
00212  0601 00 0 77777  010  ERD    STO     TMP
00213  0074 00 4 00000  010         PRINT   FD,TMP,0
00214 -1 00000 0 00306  010
00215 -1 00000 0 77777  010
00216 -1 00000 0 00000  00
00217  0020 00 0 00225  010         TRA     TBL
00220  0601 00 0 77777  010  ERE    STO     TMP
00221  0074 00 4 00000  010         PRINT   FE,TMP,0
00222 -1 00000 0 00315  010
00223 -1 00000 0 77777  010
00224 -1 00000 0 00000  00
00225  0534 00 1 13721  00   TBL    LXA     TRANS,1
00226 -3 00001 1 00246  010         TXL     END+1,1,1
00227  1 77777 1 00230  010         TXI     *+1,1,-1
00230 -0634 00 1 00235  010         SXD     MAX,1
00231  0074 00 4 00000  010         PRINT   FTBL
00232 -1 00000 0 00324  010
00233  0600 00 0 77777  010         STZ     TMP
00234  0774 00 1 00001  00          AXT     1,1
00235  3 00000 1 00245  010  MAX    TXH     END,1,**
00236 -1 00000 1 13721  00          IOP     TRANS,1
```

ID QUAD 007

ID QUAD 008

```
00237 -1 00000 1 17642  00          IOP     FIRST,1
00240 -1 00000 1 23563  00          IOP     DEST,1                                                    ID QUAD 009
00241 -1 00000 1 27504  00          IOP     LAST,1
00242  0634 00 1 77777  010         SXA     TMP,1
00243 -1 00000 0 77777  010         IOP     TMP
00244  1 00001 1 00235  010         TXI     MAX,1,1
00245 -1 00000 0 00000  00    END   ENDIO
00246  0074 00 4 00001  010         CALL    ERROR
00247  0 00000 0 66016  010         PZE     V-5000
00250  0 00000 0 77626  010   ADR   PZE     V
00251  0 00000 0 00000  00    SW    PZE
00252  030430105454     00    FA    BCI     7,34H8**** SETUP NOT CALLED BEFORE QUAD*
00253  545460622563     00
00254  644760454663     00
00255  602321434325     00
00256  246022252646     00
00257  512560506421     00
00260  245460606060     00
00261  030430105454     00    FB    BCI     7,34H8**** SETUP NOT CALLED BEFORE DIAG*
00262  545460622563     00
00263  644760454663     00
00264  602321434325     00                                                                            ID QUAD 010
00265  246022252646     00
00266  512560243121     00
00267  275460606060     00
00270  030130105454     00    FC    BCI     9,31H8**** TRANSITION TABLE EXCEEDED/28H0**** LAST QUADR
00271  545460635121     00
00272  456231633146     00
00273  456063212243     00
00274  256025672325     00
00275  252425246102     00
00276  103000545454     00
00277  546043216263     00
00300  605064212451     00
00301  644743256066     00          BCI     5,UPLE WAS .../F45.5,3(1H,I19)*
00302  216260333333     00
00303  612604053305     00
00304  730374013073     00
00305  310111345460     00
00306  011030105454     00    FD    BCI     7,18H8**** STATE NUMBERI12,13H EXCEEDS 5000*
00307  545460626321     00
00310  632560456444     00                                                                            ID QUAD 011
00311  222551310102     00
00312  730103306025     00
00313  672325252462     00
00314  600500000054     00
00315  011030105454     00    FE    BCI     7,18H8**** STATE NUMBERI12,12H IS NEGATIVE*
00316  545460626321     00
00317  632560456444     00
00320  222551310102     00
00321  730102306031     00
00322  626045252721     00
00323  633165255460     00
00324  020530012364     00    FTBL  BCI     9,25H1CURRENT TRANSITION TABLE/1H0S39,5HTRANSS15,5HFIRST
00325  515125456360     00
00326  635121456231     00
00327  633146456063     00
00330  212243256101     00
```

```
     00331  300062031173      00
     00332  053063512145      00
     00333  626201057305      00
     00334  302631516263      00                                                    ID QUAD 012
     00335  620106730430      00          BCI     9,S16,4HDESTS16,4HLASTS10,16HQUADRUPLE NUMBER/(1HC,F45.5
     00336  242562636201      00
     00337  067304304321      00
     00340  626362010073      00
     00341  010630506421      00
     00342  245164474325      00
     00343  604564442225      00
     00344  516174013000      00
     00345  732604053305      00
     00346  730431020034      00          BCI     2,,4I20)*
     00347  546060606060      00
                      77777         TMP     ERAS    101
                      77631         Z       ERAS    2
                      77626         V       ERAS
                      13721         TRANS   PGMCOM  2001
                      17642         FIRST   PGMCOM  2000
                      23563         DEST    PGMCOM  2000
                      27504         LAST    PGMCOM  2000
                      27510         EPSI    PGMCOM  3
                      27511         DELTA   PGMCOM
                      27517         MAXJ    PGMCOM  5
                                    END

PROGRAM LITERALS
     00350  000000011610      00
     00351  000000000001      00

**** 00352 IS THE FIRST LOCATION NOT USED BY THIS PROGRAM.

REFERENCES TO SYMBOLS DEFINED IN PROGRAM.
TYPE SYMBOL VALUE REFERENCES TO SYMBOL   (SYMBOL TYPES -- R - RELOCATABLE, A - ABSOLUTE, E - ERASABLE, T - TRANSFER VECTOR)
 R    ADR    00250 00004,00250
 R    BKD    00127 00115,00121,00127,00136,00137
 R    BKZ    00156 00116,00156,00161
 R    BQ     00100 00073,00100
 R    COND   00122 00010,00122,00163
 A    DELTA  27511 00061,00067,00114,00147,00350
 A    DEST   23563 00055,00145,00206,00240,00350
 R    DIAG   00105 00000,00002,00105
 R    END    00245 00226,00235,00245
 R    ENDZ   00162 00155,00157,00162
 A    EPSI   27510 00134,00350
 R    ERC    00201 00040,00120,00152,00201
 R    ERD    00212 00050,00212
 R    ERE    00220 00052,00220
 T    ERROR  00001 00174,00200,00211,00246
 R    ERSETA 00171 00033,00171
 R    ERSETB 00175 00107,00175
 R    FA     00252 00172,00252
 R    FB     00261 00176,00261
 R    FC     00270 00203,00270
 R    FD     00306 00214,00306
 R    FE     00315 00222,00315
 A    FIRST  17642 00053,00071,00144,00205,00237,00350
```

```
R   FN      00044 00027,00044
R   FTBL    00324 00232,00324
R   HIGH    00140 00013,00130,00135,00140
R   INC     00077 00070,00077
I   .PRINT  00000 00171,00175,00202,00213,00221,00231
A   LAST    27504 00064,00150,00207,00241,00350
R   MAX     00235 00230,00235,00244
A   MAXJ    27517 00350
R   QUAD    00031 00000,00002,00031
R   SETUP   00002 00000,00002,00002
R   STPZ    00157 00154,00156,00157
R   SW      00251 00021,00031,00105,00167,00251
R   TBL     00225 00217,00225
R   THRU    00073 00066,00073,00077
E   TMP     77777 00057,00063,00212,00215,00220,00223,00233,00242,00243,00350
A   TRANS   13721 00020,00037,00042,00074,00101,00117,00141,00204,00225,00236,00350
E   V       77626 00247,00250,00350
R   VA      00023 00005,00023,00024
R   VB      00075 00006,00075
R   VC      00076 00007,00076
R   VD      00131 00011,00131
R   VE      00132 00012,00132
R   VF      00142 00014,00142
R   VG      00160 00016,00160
R   XRA     00025 00002,00025
R   XRB     00045 00034,00043,00045
R   XRC     00102 00035,00102
R   XRD     00103 00036,00103
R   XRE     00164 00110,00126,00164
R   XRF     00165 00111,00165
R   XRG     00166 00112,00166
E   Z       77631 00003,00350

REFERENCES TO PROGRAM LITERALS
 LOC  REFERENCES
00350 00047
00351 00015,00032,00106

FLAGGED LINES  FATAL  0    NON-FATAL 0    ****  DEFINITIONS  SYMBOL 55    OPERATION 0    MACRO 0    RMT 0
MACRO CALLS    CORRECT 0   INCORRECT 0   ****  CARD COUNTS  INPUT  177   OUTPUT    13
```