AN EXACT SOLUTION TO A CLASS OF STORAGE-LIMITED EXPONENTIAL QUEUEING SYSTEMS

by

R. M. Bryant

Computer Science Technical Report #426

March 1981

AN EXACT SOLUTION TO A CLASS OF STORAGE-LIMITED EXPONENTIAL QUEUEING SYSTEMS

R. M. Bryant*
Madison Academic Computing Center
and
Computer Sciences Department
1210 W. Dayton St.
Madison, Wisconsin 53706

ABSTRACT

A storage-limited (SL-type) exponential queueing system is a generalization of an M/M-type queueing system in that each customer has a storage requirement as well as a service time. There is a fixed amount of storage associated with the server. The system is called storage-limited because only those customers who can jointly fit into the server's storage are eligible for service. Such queueing systems can form the basis of analytic computer-system models that explicitly depict the size of main memory and the memory-request size distribution.

This paper gives a numerical method for solving

---

*Supported in part by NSF grant MCS-8003341 at the University of Wisconsin-Madison. This work originates in the author's Ph. D. dissertation which was supported in part by NASA contracts NAS5-22878 and NAS5-24092 and EPA grant R805478-01-0 at the University of Maryland.

limited-arrival-process SL-M/M/-type queueing systems (i. e. systems with $\lambda(n) = 0$ for $n > N_p$). The method requires a number of states proportional to N_p^2; the number of states is essentially independent of the size of the server's storage. This is a considerable improvement over straightforward solution techniques, where the number of states increases exponentially with storage size.

This exact solution is compared with a solution to a finite memory size model described by R. Brown, J. Browne and K. M. Chandy. The latter is shown to be an approximate solution based on a certain independence assumption. For the finite population cases, the accuracy of this approximate method is good with relative errors being less than 5%. For the infinite population, limited waiting room cases, the error is somewhat higher. The maximum error observed in these cases is on the order of 10%. Considering the small cost of the approximate solution technique, it is to be highly recommended.

Keywords and Phrases: multiple-resource queueing systems, computer system modeling, multiprogramming level distribution, finite population models, equilibrium equations.

AN EXACT SOLUTION TO A CLASS OF STORAGE-LIMITED EXPONENTIAL QUEUEING SYSTEMS

1 . INTRODUCTION

This paper describes a numerical technique that provides an exact solution to a class of dual-resource queueing models. We refer to this class of models as storage limited (SL-type) exponential queueing systems because they are immediate generalizations of M/M-type queueing systems. The difference is that in an SL-M/M-type queueing system, each customer's resource requirement is specified by a pair of random variables representing the customer's storage and service requirements, respectively. The system is called "storage limited" because there is a finite amount of storage associated with the server and only those customers who can jointly fit into the storage are eligible for service. The class of SL-M/M/-type queueing systems is important because they provide a method of creating analytic computer system models that explicitly include the size of memory and the memory-request size distribution.

It is possible to represent an SL-M/M-type queueing system as a Markov chain with a very large state space. Unfortunately, the size of the state space grows too rapidly with the size of the storage to use this method of solution for realistic storage sizes. For example, for a system with 100 blocks of memory, there are more than 190,000,000 states [6]. The primary result of this paper is a method for finding an exact solution to limited-arrival-process SL-M/M-type queueing systems (i. e. systems with $\lambda_n=0$ for all $n>N_P$). This method requires $O(N_P^2)$ states, and the number of states is essentially independent of the storage size. The storage size and the memory-request size distribution enter the solution through a peripheral calculation used to determine transition probabilities among the $O(N_P^2)$ states. This peripheral calculation needs to be done only once per storage size and is independent of the system arrival and service rates. With this technique SL-M/M/-type processor-sharing queueing systems with hundreds of units of main storage and up to 30 customers have been solved exactly in a few minutes of computer time.

Related Work. This paper follows the approach of Bard [2] in calculating the distribution of the MPL; a more efficient variation of Bard's algorithm for calculating $P\{L_k=i \mid L_{k-1}=j\}$ is derived here and is crucial to the solution of the problem. A similar analysis has been performed by Buzen and Rubin [8], but their primary result deals with the distribution of unused memory (when all possible jobs have been loaded from a never empty queue) instead of the distribution of the MPL. Betteridge [3] used a Markov chain approach with each state containing not only the memory sizes of all jobs in system but the starting block address of each loaded job. He then proposed using numerical methods to study the behavior of the chain, with the goal being to compare such memory allocation methods as first-fit and best-fit. However, the state space of the Markov chain grows so rapidly with increasing memory size that no realistic comparisons appear to be possible.

R. Brown _et al_ [4] appear to be the first researchers to incorporate the memory size distribution of user tasks in an analytic computer-system model. Their solution depends on the assumption that the MPL, when observed at job departure instants, may be modeled as an independent and identically distributed sequence of random variables, subject to the restriction that the maximum number of jobs that can be loaded at any instant is the total number of ready jobs. The examples of Section 5 in this paper provide a numerical estimate of the error this independence assumption introduces in the solution of some typical SL-M/M-type queueing systems. Brown _et al_'s solution can be used to incorporate memory size constraints in product form queueing network in a very straightforward way. This has apparently been done in at least one general network of queues solution package [16].

Konheim and Reiser [12; 13] solve a slightly different type of finite memory-size model where the MPL is assumed to have a fixed upper bound. Hence the memory-request size distribution enters only indirectly into their model. In [13] they also explore the accuracy of the decomposition argument used in [4; 13] to reduce the solution of a network of queues model of an entire computer system to the solution of a simple finite-capacity queueing system. Exactly the same techniques could be used to incorporate an SL-M/M-type queueing system as part of a computer system model. For further details the reader is directed to [4; 13].

Omahen and Marathe [15; 14], discuss the problems of calculating maximum service rates of general multiresource queueing systems. This author [7] has developed effective computational algorithms for finding maximum service rates of dual resource systems consisting of CPU's and main memory.

Recently, Green [9; 10] has published an analysis of queueing systems where each customer requires service from a random number of servers. If one identifies servers with units of main memory and the server request distribution with the memory request size distribution, then it is clear that Green's model is very similar to the one considered here. However, the solution of [9] assumes that servers are freed one by one, even though they must be acquired together. In [10] she considers a model in which servers are held throughout a customer's service, but no solution to this model is derived. Instead qualitative results are shown that compare performance measures of the models of [9] and [10]. This author [5; 6] has discussed approximate solutions to similar models, based on the assumption that the numbers of servers in use at each job departure and arrival instant are independent random variables. These models have been found to be less accurate [5] than those of [4].

Model Assumptions and Notation. A storage-limited queueing system is a simple model of a computer system with a processor and M independently allocatable units (blocks) of main storage. Since M is in general quite large, we will assume throughout that $N_P << M$. In this model we assume that no external storage fragmentation takes place. This is equivalent to assuming that memory is compacted at each job departure time [8] or that paging hardware is used to eliminate external storage fragmentation [2]. We require, however, that the entire storage requirement of a job must fit into main memory before the job can be loaded. Finally, we assume that jobs are loaded strictly in the order of their arrival (first-come first-loaded or FCFL). If when a job arrives there are no jobs waiting to be loaded and there are enough free blocks of memory the job is loaded. Otherwise the job enters the memory queue. On the other hand, if the memory queue is non-empty when the job arrives, the job always joins the memory queue. When a job's service requirement has been satisfied, its memory space is freed and as many jobs as possible are loaded from the memory queue. In all cases we assume that the loading process is instantaneous.

Jobs arrive at the system according to a Poisson process of rate λ_n when there are n jobs in system. We restrict our attention to arrival processes with $\lambda_n=0$ for all $n > N_p$. Two important queueing systems that can be studied in this framework are SL-M/M/K//N_p and SL-M/M/K/N_p.

The resource requirements of each job are represented by the pair of independent random variables (X_i,S_i). X_i denotes the storage requirement (in blocks) of the i^{th} job; the sequence $\{X_i\}$ is assumed to be i. i. d. with distribution $F(x)$ and density $f(x)$. The X_i's are required to be integer valued with $1 \leq X_i \leq m$ where $m \leq M$. X_i can be interpreted as either the program size of the the i^{th} job in a swapping system or as the working set size of the i^{th} job in a paging system. S_i represents the service time requirement of the i^{th} job. The sequence $\{S_i\}$ is i. i. d. with an exponential distribution of parameter μ.

Let $L(t)$ denote the number of loaded jobs (the multiprogramming level or MPL) at time t and let $N(t)$ denote the number of jobs in system at time t. By a solution to an SL-M/M-type queueing system we mean a method of evaluating the stationary distribution $\pi(j,k)$ of the process $(L(t),N(t))$.

Let $r_n(k)$ be the system service rate given $N(t)=n$ and $L(t)=k$. We assume, without loss of generality, that $r_1(1)=1.0$. It follows from our exponential service time assumption that the average job interdeparture time given $N(t)=n$ and $L(t)=k$ is $(\mu \, r_n(k))^{-1}$. This quantity does not depend on the service discipline among the loaded jobs so long as the overall service rate is $r_n(k)$. To simplify notation we will put $\mu_{n,k} = \mu\, r_n(k)$.

We define $t_\emptyset=\emptyset$ and for $k\geq 1$ we let t_k denote the departure time of the kth job to leave the system; we note that jobs need not depart in the same order that they arrive. Similarly, define e_k as the time of the kth change of state of the process $(L(t),N(t))$. We let $T(e_k)$ be the type of event that occurred at time e_k. Then $T(e_k)$ is either an arrival (which we shall denote by $T(e_k)=A$) or a departure $(T(e_k)=D)$. Finally, we put $N_k = N(e_k)$; $L_k = L(e_k)$.

Throughout this paper, we will use subscripts to indicate particular members of a sequence, e. g. X_k. Capital letters are (usually) used to indicate random variables, lower case letters indicate values for the associated random variables. Underbars are used to indicate vector quantities and superscripts are used to indicate elements of a vector. Thus Z_k^j is the jth element of $\underline{Z}_k$, the latter being a random vector. Square brackets are used to indicate events (such as $[X=x]$). The probability of this event is denoted by $\Pr\{X=x\}$; the expected value of the random variable X is denoted by $E\{X\}$ or $\overline{X}$. A box ($\Box$) is used to indicate the end of a proof or the end of a theorem with no formal proof.

<u>Summary of paper</u>. We begin by studying the process $(L(t),N(t))$. While this process is non-Markovian, it can be expressed as a function of an underlying Markov chain $(\underline{Z}(t),N(t))$. In Section 2, we defined and discuss the

process $(\underline{Z}(t),N(t))$ and derive a set of equations for $\pi(j,n)$. Then in Section 3, we discuss calculation of the coefficients in these equations. Next, in Section 4, we discuss the approximate solution of R. Brown *et al* [4]. In Section 5 we present some example solutions to an SL-M/M/∞ queueing system and compare the exact and approximate methods of solution.

2. THE PROCESSES $(L(t),N(t))$ AND $(\underline{Z}(t),N(t))$

Our goal is to determine steady state occupancy probabilities for the process $(L(t),N(t))$. However, this process is non-Markovian because $L(t)$ depends on the memory sizes of all the jobs in system and this information is not part of the state description. $(L(t),N(t))$ can be expressed as a function of an underlying Markov chain as follows.

Let the vector $\underline{Z}(t) = (Z^1(t),Z^2(t), \ldots ,Z^M(t))$ represent the memory sizes of the $N(t)$ jobs currently in system and of the next $M-N(t)$ jobs to arrive in system at time t. Here $Z^1(t)$ is the memory requirement of the oldest job in system at time t, $Z^2(t)$ is the memory requirement of the second oldest job, and so forth. We define $\underline{Z}_k = \underline{Z}(e_k)$.

Given $\underline{Z}(t)$, we can calculate $L(t)$ as $\min(R(\underline{Z}(t)),N(t))$ where $R(\underline{z})$ is defined as:

$$R(\underline{z}) = \begin{cases} M \text{ if } \sum_{i=1}^{M} z^i = M \\ \\ n \text{ if } \sum_{i=1}^{n} z^i \leq M \text{ and } \sum_{i=1}^{n+1} z^i > M \end{cases}$$

where $\underline{z}=(z^1,z^2, \ldots ,z^M)$.

Thus $R(\underline{Z}(t))$ gives the number of jobs that could be loaded if the memory queue were never allowed to become empty (i. e. if the workload was "infinite"); the relation $L(t) = \min(R(\underline{Z}(t)),N(t))$ merely says that the number of loaded jobs can at most be the number of jobs in system.

In [7] we studied the process $\underline{Z}(t)$ at job departure instances subject to the constraint that the memory queue was never empty. Let $\{\underline{Z}_k^{\infty}\}$ denote this modified sequence. We summarize the following results about $\{\underline{Z}_k^{\infty}\}$ from [7]:

Lemma 2.1: $\{\underline{Z}_k^{\infty}\}$ is a positive recurrent Markov chain with a unique stationary distribution. $\square$

Theorem 2.2: The stationary distribution π of the Markov chain $\{\underline{Z}_k^{\infty}\}$ is given by

$$\pi(\underline{z}) = \prod_{i=1}^{M} f(z^i). \qquad (2.1)$$

Furthermore, this distribution is the same regardless of how the next departing job is chosen from the set of loaded jobs, provided only that this choice is made independently of memory size. $\square$

It follows that equation 2.1 also gives the steady state distribution for $\{\underline{Z}_k^\infty\}$.

We recall that $\{\underline{Z}(t_k)\}$ is the sequence generated by observing $\underline{Z}(t)$ at job departure times. The only difference between $\{\underline{Z}(t_k)\}$ and $\{\underline{Z}_k^\infty\}$ is that for $\{\underline{Z}(t_k)\}$ the choice of departing job is made from the memory sizes of the first $L(t_k)$ entries in the vector $\underline{Z}(t_k)$, while for $\{\underline{Z}_k^\infty\}$ the choice is made from the first $R(\underline{Z}_k^\infty)$ entries. In either case, the choice is made independently of memory size. It follows that $\{\underline{Z}(t_k)\}$ has the same stationary distribution as does $\{\underline{Z}_k^\infty\}$.

If we put $L_k^\infty = R(\underline{Z}_k^\infty)$, we have the following result [7]:

<u>Theorem 2.3</u>: The marginal distribution of L_k^∞ is given by

$$\Pr\{L_k^\infty = n\} = F^{(n)}(M) - F^{(n+1)}(M)$$

where $F^{(n)}$ denotes the n-fold convolution of F with itself. $\square$

Since $L_k = \min(R(\underline{Z}_k), N_k)$ it follows that:

<u>Corollary 2.4</u>: The marginal distribution of L_k given $N_k = n$ is

$$\Pr\{L_k=j \mid N_k=n\}=\begin{cases} F^{(j)}(M)-F^{(j+1)}(M) & j<n \\ F^{(n)}(M) & j=n. \end{cases} \quad \square$$

Since $\underline{Z}(t)$ contains the memory sizes of all jobs in system at time t, since $\underline{Z}(t)$ and $N(t)$ determine $L(t)$, and since the state residency times of $(\underline{Z}(t),N(t))$ depend only on $L(t)$ and $N(t)$, it follows that $(\underline{Z}(t),N(t))$ is a continuous time Markov chain. The infinitesimal parameters of $(\underline{Z}(t),N(t))$ are determined by the distribution of $e_{k+1}-e_k$ and the transition probabilities of the embedded (discrete time) Markov chain $(\underline{Z}_k,N_k)$. It is convenient to specify the latter conditioned on $T(e_{k+1})$.

From our exponential distribution assumptions for service and interarrival times it follows that:

$$P\{e_{k+1}-e_k\leq s \mid (\underline{Z}_k,N_k)=(\underline{z},n)\} = 1-\exp(-[\lambda_n+\mu_{n,j}]s) \tag{2.2}$$

$$P\{T(e_{k+1})=A \mid (\underline{Z}_k,N_k)=(\underline{z},n)\} = \frac{\lambda_n}{\lambda_n+\mu_{n,j}} \tag{2.3}$$

$$P\{T(e_{k+1})=D \mid (\underline{Z}_k,N_k)=(\underline{z},n)\} = \frac{\mu_{n,j}}{\lambda_n+\mu_{n,j}} \tag{2.4}$$

where $j=\min(R(\underline{z}),n)$.

Consider first the case $T(e_{k+1})=A$. Clearly $N_{k+1}=N_k+1$. When an arrival occurs $\underline{Z}_k$ does not change, since $\underline{Z}_k$ already contains the memory size for the newly arriving job. (Recall that by assumption $N_P << M$.) Therefore

$$P\{(\underline{Z}_{k+1},N_{k+1})= (\underline{z},n+1) \mid (\underline{Z}_k,N_k)=(\underline{z},n),\ T(e_{k+1})=A\}=1. \tag{2.5}$$

When $T(e_{k+1})=D$, clearly $N_{k+1}=N_k-1$. Now put $j=\min(R(\underline{z}_1),n)$ and let $N_D(\underline{z}_1,\underline{z}_2,j)$ be the number of ways that $\underline{z}_1$ can be changed to the $M-1$ vector

$$(z_2^1,\ z_2^2,\ z_2^3,\ .\ .\ .\ ,z_2^{M-1})$$

by deleting one of the first j elements of the vector $\underline{z}_1$. Then:

$$P\{(\underline{Z}_{k+1},N_{k+1})=(\underline{z}_2,n-1) \mid (\underline{Z}_k,N_k)=(\underline{z}_1,n),\ T(e_{k+1})=D\}$$

$$=\frac{N_D(\underline{z}_1,\underline{z}_2,j)}{j}\ f(z_2^M),\quad n>1 \tag{2.6}$$

From equations (2.2) through (2.6) it follows that the infinitesimal parameters of the process $(\underline{Z}(t),N(t))$ are given by

$$q_{(\underline{z}_1,n)}=\lambda_n+\mu_{n,j}$$

$$q_{(\underline{z}_1,n),(\underline{z}_1,n+1)}=\lambda_n$$

$$q_{(\underline{z}_1,n),(\underline{z}_2,n-1)}=\frac{\mu_{n,j}N_D(\underline{z}_1,\underline{z}_2,j)}{j}\ f(z_2^M)$$

where $j=\min(R(\underline{z}_1),n)$ and all other $q_{(i,j),(k,l)}$ are zero.

From these infinitesimal parameters it is theoretically possible to determine the steady state distribution of $(\underline{Z}(t),L(t))$ (and hence $(L(t),N(t))$). However, there are far too many states for this to be done for all but the smallest values of M. To reduce the state space to a manageable size, we use the following aggregation Lemma [6]:

Lemma 2.5: Let A(t) be a continuous time Markov chain with state space S, time homogeneous transition probabilities

$$P_{i,j}(s)=P\{A(t+s)=j \mid A(t)=i\},$$

infinitesimal parameters defined by

$$q_i=\lim_{h\to 0^+} \frac{1-P_{i,i}(h)}{h}$$

$$q_{i,j}=\lim_{h\to 0^+} \frac{P_{i,j}(h)}{h} \qquad i\neq j,$$

and a unique steady state distribution π_A.

Let $S_1,\ldots,S_k,\ldots$ be a partition of S such that each S_k has finitely many members and $q_{i,j}=0$ whenever i and j are in the same S_k. Suppose that

$$\pi(i \mid k)=P\{A(t)=i \mid A(t)\in S_k\}$$

is constant in t and known *a priori*.

Define B(t)=k iff $A(t)\in S_k$, and

$$\pi_B(k)=\sum_{s\in S_k} \pi_A(s).$$

Let

$$q_{i,S_k}=\sum_{j\in S_k} q_{i,j}, \qquad i\neq k,$$

$$Q_i=\sum_{s\in S_i} q_s\, \pi(s \mid i),$$

and $$Q_{i,k} = \sum_{s \in S_i} q_{s,S_k} \pi(s|i), \quad i \neq k.$$

Then π_B satisfies the equation

$$\pi_B(k)\, Q_k = \sum_{i \neq k} \pi_B(i)\, Q_{i,k}. \tag{2.7}$$

Proof: The equilibrium equations for π_A are of the form

$$\pi_A(s)\, q_s = \sum_{t \neq s} q_{t,s}\, \pi_A(t).$$

Summing these equations over all $s \in S_k$ and using the facts that $\pi(s|k)=\pi_A(s)/\pi_B(k)$ for $s \in S_k$ and $q_{t,s}=0$ whenever $t,s \in S_k$ gives the desired result. $\square$

Note that B(t) in the Lemma need not be a Markov chain. However we may define a new Markov chain $\hat{B}(t)$ with the same state space as B(t) and with infinitesimal parameters Q_k, $Q_{i,k}$. Then in general B(t) is not the same as $\hat{B}(t)$. Suppose, however, that $\hat{B}(t)$ possesses a unique steady state distribution $\hat{\pi}_B$. Then $\pi_B=\hat{\pi}_B$ since there is only one solution to the set of equations. Thus if the quantities Q_k and $Q_{i,k}$, when taken as the infinitesimal parameters of a Markov chain, define a Markov chain with a unique steady state distribution, then the limiting distribution of B(t) can be found from equation (2.7), even if B(t) is non-markovian. This idea allows us to derive "equilibrium" equations for the process $(L(t),N(t))$ and from them to determine the steady state distribution for this process.

We apply Lemma 2.5 to the process $(\underline{Z}(t),N(t))$ by partitioning its state space S into the sets

$$S(j,n)=\{(\underline{z},n)\mid j=\min(R(\underline{z}),n)\}.$$

Thus members of the set $S(j,n)$ are those states $(\underline{z},n)$ that correspond to a MPL of j and $(L(t),N(t))=(j,n)$ iff $(\underline{Z}(t),N(t)) \in S(j,n)$.

Not all ordered pairs (j,n) are feasible states of the process $(L(t),N(t))$. Let L_Q be the integer part of M/m. L_Q is the minimum number of jobs that must be loaded before a memory queue can form. If $n\le L_Q$, states of the form (j,n) with $j<n$ cannot occur with non-zero probability. The feasible states of the process $(L(t),N(t))$ are therefore:

$$\begin{array}{lllll}
(\emptyset,\emptyset) & (1,1) & (2,2) & \cdots & (L_Q,L_Q) \\
 & & & & \\
(L_Q,L_Q+1) & & (L_Q,L_Q+2) & \cdots & (L_Q,N_P) \\
(L_Q+1,L_Q+1) & & (L_Q+1,L_Q+2) & \cdots & (L_Q+1,N_P) \\
 & & (L_Q+2,L_Q+2) & \cdots & \cdots \\
 & & & & (N_P,N_P)
\end{array}$$

Because of the two dimensional state space of the process $(L(t),N(t))$ we redefine the quantities $\pi(i|k)$, q_{i,S_j}, Q_i, and $Q_{i,j}$ of Lemma 2.5 in the following way:

$$\pi((\underline{z},n)\mid(j,n))=P\{(\underline{Z}(t),N(t))=(\underline{z},n)\mid(\underline{z},n)\in S(j,n)\}$$

$$q_{(\underline{z},n),S(j,k)}=\sum_{(\underline{y},k)\in S(j,k)} q_{(\underline{z},n),(\underline{y},k)} \qquad (\underline{z},n)\notin S(j,k)$$

$$Q_{(j,k)} = \sum_{(\underline{y},k)\in S(j,k)} q_{(\underline{y},k)} \, \pi((\underline{y},k)\,|\,(j,k))$$

$$Q_{(p,t),(r,s)} = \sum_{(\underline{y},t)\in S(p,t)} q_{(\underline{y},t),S(r,s)} \, \pi((\underline{y},t)\,|\,(p,t))$$

To apply the Lemma to the process $(L(t),N(t))$, we must be able to evaluate $\pi((\underline{z},n)\,|\,(j,n))$. Now since the state residency times of all states in $S(j,n)$ are exponentially distributed with parameter $(\lambda(n) + \mu_{n,j})$, $\pi((\underline{z},n)\,|\,(j,n))$ does not depend on $\lambda(n)$ or $\mu_{n,j}$. Since $\{\underline{Z}(t_k)\}$ has the same marginal distribution as $\{\underline{Z}_k^{\infty}\}$ we can use Corollary 2.4 to show that:

$$\pi((\underline{z},n)\,|\,(j,n)) = \begin{cases} \dfrac{1}{F^{(j)}(M) - F^{(j+1)}(M)} \displaystyle\prod_{i=1}^{M} f(z^i) & j<n \\[2em] \dfrac{1}{F^{(j)}(M)} \displaystyle\prod_{i=1}^{M} f(z^i) & j=n \end{cases}$$

The parameters $Q_{(j,k)}$ and $Q_{(j,k),(m,n)}$ take several different forms depending on the values of j, k, m, and n. To give the flavor of this analysis, we consider one case here (Appendix A discusses the other cases.):

<u>Case</u> <u>iv</u>: $Q_{(j,n),(k,n-1)}$ with $j<n$. Clearly if $(\underline{z},n) \in Q_{(j,n)}$, $j<n$, then $R(\underline{z})<n$. Thus:

$$Q_{(j,n),(k,n-1)} = \sum_{(\underline{z},n)\in S(j,n)} q_{(\underline{z},n),S(k,n-1)} \pi((\underline{z},n)\,|\,(j,n))$$

$$= \sum_{(\underline{z},n)} \sum_{(\underline{y},n-1)\in S(k,n-1)} \frac{\mu_{n,j} N_D(\underline{z},\underline{y},j)}{j} f(y^M) \pi((\underline{z},n)\,|\,(j,n))$$

Here j=min(R($\underline{z}$),n). (Recall that y^M refers to the m^{th} element of the vector $\underline{y}$.) To simplify this further, we need to consider the cases k<n-1 and k=n-1 separately. In the case k<n-1 we have

$$Q_{(j,n),(k,n-1)} = \mu_{n,j} \sum_{\underline{z} \ni R(\underline{z})=j} \sum_{\underline{y} \ni R(\underline{y})=k} \frac{N_D(\underline{z},\underline{y},R(\underline{z}))}{R(\underline{z})} f(y^M)\ \pi((\underline{z},n)\mid(j,n)).$$

But the term

$$\frac{N_D(\underline{z},\underline{y},R(\underline{z}))}{R(\underline{z})} f(y^M)$$

in this equation is merely the probability of changing from memory state $\underline{y}$ to $\underline{z}$ given that $N(t) \geq R(\underline{z})$. But this is equivalent to assuming that $N(t) = \infty$ so that the above can be rewritten in terms of transition probabilities for the sequence $\{\underline{Z}_k^{\infty}\}$:

$$Q_{(j,n),(k,n-1)} = \mu_{n,j} \sum_{\underline{z}} \sum_{\underline{y}} \frac{P\{Z_2^{\infty}=\underline{y} \mid Z_1^{\infty}=\underline{z}\}\ P\{Z_1^{\infty}=\underline{z}\}}{P\{L_1^{\infty}=j\}}$$

Examining the limits of summation, the above can be reduced to:

$$\mu_{n,j}\ P\{L_2^{\infty}=k \mid L_1^{\infty}=j\}.$$

In the case k=n-1 we have, by similar arguments:

$$Q_{(j,n),(n-1,n-1)} = \mu_{n,j} \sum_{\underline{z} \ni R(\underline{z})=j} \sum_{\underline{y} \ni R(\underline{y}) \geq n-1} \frac{N_D(\underline{z},\underline{y},R(\underline{z}))}{R(\underline{z})} f(y^M)\pi((\underline{z},n)\mid(j,n))$$

$=\mu_{n,j}\ P\{L_2^{\infty} \geq n-1 \mid L_1^{\infty} = j\}.\ \square$

The resulting equilibrium equations for the process $(L(t),N(t))$ are summarized in equation (2.8). Since there is one equation for each feasible state, a total of $L_Q+(N_P-L_Q)(N_P-L_Q+1)/2 = O(N_P^2)$ equations must be solved. We will discuss methods of solving this set of equations in Section 5 .

Because state $(\emptyset,\emptyset)$ is reachable from all states in the process $(L(t),N(t))$, it follows that the infinitesimal parameters $Q_{(i,j)}$ and $Q_{(p,q),(r,s)}$ define an irreducible Markov process on a finite state space. Therefore there is a unique probability measure $\pi(j,k)$ that satisfies equation (2.8). Thus we can use the equilibrium equations (2.8) to solve for $\pi(j,k)$ even though $(L(t),N(t))$ is not a Markov Chain.

3 . CALCULATION OF THE COEFFICIENTS IN EQUATION (2.8)

We now show how to calculate the probabilities $P\{L_2^{\infty}=v \mid L_1^{\infty}=u\}$ and $P\{L_2^{\infty}\geq v \mid L_1^{\infty}=u\}$. These quantities appear as coefficients in equation (2.8). Our analysis follows that of [2] who develops a similar but much less efficient formula for $P\{L_2^{\infty} \mid L_1^{\infty}\}$.

To simplify our discussion, we only consider the case $u>1$, $v>u$. The rest of the cases are analyzed in Appendix B. As a notational convenience, let

$$\pi(\emptyset,\emptyset)\lambda_\emptyset = \mu_{1,1}\ \pi(1,1)$$

$$\pi(j,j)(\lambda_j+\mu_{j,j}) = \lambda_{j-1}\pi(j-1,j-1)+\mu_{j+1,j+1}\pi(j+1,j+1) \qquad 1 \le j < L_Q$$

$$\pi(j,j)(\lambda_j+\mu_{j,j}) = \lambda_{j-1}P\{L_1^\infty > j-1 \mid L_1^\infty \ge j-1\}\pi(j-1,j-1)+\sum_{k=L_Q}^{j+1}\mu_{k,j+1}P\{L_2^\infty \ge j \mid L_1^\infty = k\}\pi(k,j+1) \qquad L_Q \le j < N_P$$

$$\pi(N_P,N_P)\mu_{N_P,N_P} = \lambda_{N_P-1}P\{L_1^\infty > N_P-1 \mid L_1^\infty \ge N_P-1\}\pi(N_P-1,N_P-1)$$

$$\pi(i,j)(\lambda_j+\mu_{i,j}) = \lambda_{j-1}\pi(i,j-1)+\sum_{k=L_Q}^{i+1}\mu_{k,j+1}P\{L_2^\infty = i \mid L_1^\infty = k\}\pi(k,j+1) \qquad L_Q \le i < j-1,\ L_Q+2 \le j < N_P$$

$$\pi(j-1,j)(\lambda_j+\mu_{j-1,j}) = \lambda_{j-1}P\{L_1^\infty = j-1 \mid L_1^\infty \ge j-1\}\pi(j-1,j-1)+\sum_{k=L_Q}^{j}\mu_{k,j+1}P\{L_2^\infty = j-1 \mid L_1^\infty = k\}\pi(k,j+1) \qquad L_Q < j < N_P$$

$$\pi(N_P-1,N_P)\mu_{N_P-1,N_P-1} = \lambda_{N_P-1}P\{L_1^\infty = N_P-1 \mid L_1^\infty \ge N_P-1\}\pi(N_P-1,N_P-1)$$

$$\pi(j,N_P)\mu_{j,N_P} = \lambda_{N_P-1}\pi(j,N_P-1) \qquad L_Q \le j < N_P-1$$

Equation(s) (2.8)

Equilibrium Equations for $(L(t),N(t))$

$$S_i^j = \sum_{k=i}^{j} X_k .$$

Since the value of $P\{L_1^{\infty}=u\}$ is known, it is sufficient to determine $P\{L_1^{\infty}=u, L_2^{\infty}=v\}$.

Case VI: $u>1$, $v>u$. Here the event $E=[L_1^{\infty}=u, L_2^{\infty}=v]$ is equivalent to

$$[S_1^u \leq M,\ S_1^{u+1}>M,\ S_2^{v+1}\leq M,\ S_2^{v+2}>M].$$

This can be rewritten in terms of independent random variables as

$$[X_1+S_2^u \leq M,\ X_1+S_2^u+X_{u+1}>M,\ S_2^u+X_{u+1}+S_{u+2}^{v+1}\leq M,\ S_2^u+X_{u+1}+S_{u+2}^{v+1}+X_{v+2}>M].$$

If $X_1+S_2^u+X_{u+1}>M$, then $S_2^u \geq M-2m+1$. Similarly if $S_2^u+X_{u+1}+S_{u+2}^{v+1}\leq M$ then $S_2^u \leq M-(v-u+1)$. But we know that $m(u-1)\geq S_2^u \geq u-1$. Combining these conditions gives us

$$\max(M-2m+1, u-1) \leq S_2^u \leq \min(M-(v-u+1), m(u-1)). \qquad (3.1)$$

Conditioning on the event $[S_2^u=s]$ gives us

$$P\{E\} = \sum_s f^{(u-1)}(s)\ P\{E_1\}$$

where

$$E_1 = [X_1 \leq M-s,\ X_1+X_{u+1}>M-s,\ X_{u+1}+S_{u+2}^{v+1}\leq M-s,\ X_{u+1}+S_{u+2}^{v+1}+X_{v+2}>M-s]$$

and the limits on s are given by equation (3.1).

Now if $X_1+X_{u+1}>M-s$, then $X_{u+1}>M-(s+m)$. Similarly if $X_{u+1}+S_{u+2}^{v+1}\leq M-s$, then $X_{u+1}\leq M-(s+v-u)$. Since $1\leq X_{u+1}\leq m$ we therefore have

$$\max(1, M-(s+m)) \leq X_{u+1} \leq \min(m, M-(s+v-u)). \qquad (3.2)$$

Conditioning on the event $[X_{u+1}=x]$ gives us

$$P\{L_1^{\infty}=u,L_2^{\infty}=v\}=\sum_s f^{(u-1)}(s)\sum_x f(x)\ P\{E_2\}\ P\{E_3\} \tag{3.3}$$

where

$$E_2=[X_1\le M-s,\ X_1>M-(s+x)],$$
$$E_3=[S_{u+2}^{v+1}\le M-(s+x),\ S_{u+2}^{v+1}+X_{v+2}>M-(s+x)],$$

and the limits on s and x are given by equations (3.1) and (3.2) respectively. However $P\{E_3\}$ is merely the probability that the MPL is v-u in a main memory of size M-(s+x). Therefore

$$P\{E_2\}=F(M-s)-F(M-(s+x)) \text{ and}$$
$$P\{E_3\}=F^{(v-u)}(M-(s+x))-F^{(v-u+1)}(M-(s+x)). \ \square$$

$P\{L_2^{\infty}\ge v|L_1^{\infty}=u\}$ can be calculated in a similar way. For details see Appendix B.

4 . AN APPROXIMATE SOLUTION TECHNIQUE

In this section we present the approximate solution of Brown et al [4] as it applies to solving a SL-M/M-type queueing system.

Brown et al consider two types of memory schedules: first-fit with skip and first-fit without skip. Since the two schedules are handled much the same way and since the second corresponds to the FCFL policy that we have been using in this paper, we will skip discussion of the first policy.

The basic result of Brown et al is a recursive method of evaluating the probability $P\{i|j\}$ of having exactly i jobs loaded given j jobs in system:

Let $g(n,y|k)$ be the conditional probability that n jobs are assigned y units of memory given that k jobs have been examined. There are three cases:

(i) $n = k$, i. e. all ready jobs examined are loaded:

$$g(k,y|k) = \sum_{x=\emptyset}^{y} g(k-1,x|k-1)\ f(y-x)$$

(ii) $n = k-1$, i. e. the last job examined did not fit:

$$g(k-1,y|k) = g(k-1,y|k-1)\ (1 - F(M-y))$$

(iii) $n < k-1$, i. e. the scheduler has ceased to examine jobs because some previously examined job did not fit into avaliable memory:

$$g(n,y|k) = g(n,y|k-1)$$

Given $g(n,y|k)$, $P\{i|j\}$ can be evaluated from

$$P\{i|j\} = \sum_{y=1}^{M} g(i,y|j).$$

It can be shown that the above reduces to the formula of Corollary 2.4.

Given $P\{i|j\}$ Brown et al solve the SL-M/M-type queue by assuming that it can be modeled as an M/G/1-PS (processor sharing) queueing system where the service distribution given j jobs in system is:

$$B_j(t) = \sum_{i=1}^{j} (1-\exp(\mu_{i,j}\ t))\ P\{i|j\}$$

$B_j(t)$ is seen to be the inter-departure time distribution given j jobs in system. From the well-known relationship between M/G/1-PS and M/M/1 queueing systems, it follows that one may find an approximate solution to an SL-M/M-type queueing system by solving the M/M/1 queueing system with load dependent service rate $\bar{\mu}_j$ where

$$\bar{\mu}_j = \sum_{i=1}^{j} \mu_{i,j}\ P\{i|j\}.$$

We wish to carefully point out the approximation assumption that has been made in this solution. Note that g(n,y|k) (and hence P{i|j}) are calculated under the assumption that memory is initially empty. Thus the MPL given j jobs in system is distributed according to P{i|j} <u>and is assumed to be independent of the MPL before the last job departure</u>. One way to think of the model is that at each job departure instant, all loaded jobs are removed from memory, given new memory sizes, and returned to the memory queue. They are then reloaded into memory in order until either memory becomes full or all jobs in system are loaded. This independence assumption simplifies considerably the task of solving an SL-M/M-type queueing system.

This author has also considered models where the MPL at each arrival instant is also assumed to be independent of the MPL just before the arrival. While this model can be solved exactly without the artifact of the PS assumption introduced here, experience shows that the model is less accurate than the one of Brown et al [5].

5 . SOME EXAMPLE SOLUTIONS

Equation (2.8) can be written in compact form as the eigenvector problem $\underline{Q}\,\underline{\Pi}=\emptyset$ where $\underline{Q}$ is a matrix of order $(N_P-L_Q)^2/2$. If (N_P-L_Q) is not too large, one may determine $\underline{\Pi}$ by the inverse power method (see, for example, [17,p. 343]). Since the storage requirement of this approach increases as $(N_P-L_Q)^4$, some other technique must be used when (N_P-L_Q) is large.

By exploiting the structure of the matrix $\underline{Q}$, we can reduce the problem of solving $\underline{Q}\,\underline{\Pi}=\emptyset$ to the problem of solving $\underline{R}\,\underline{y}=\emptyset$ where the order of $\underline{R}$ is (N_P-L_Q) and a simple relation exists between the vectors $\underline{y}$ and $\underline{\Pi}$. To do this, we use the "recursive" technique of [11]. The details of this approach are given in Appendix C.

In the rest of this section we discuss two examples solved by these methods and compare the exact solution values to those obtained by the approximate method of [4]. We restrict our attention to the SL-M/M/∞ queue. Because the system service rate is $\mu_{n,k} = k\,\mu$ for the SL-M/M/∞ system, this system exhibits the strongest dependence of

$\mu_{n,k}$ on k among all SL-M/M-type queueing systems likely to be encountered in practice. Hence this should be a worst case comparison for the approximate solution method.

In these examples we have used two distinct memory size distributions. One memory size distribution is the Univac 1100/82 program size distribution shown in Figure 5.1. This distribution is taken from data observed at the Madison Academic Computing Center during the summer of 1980. As is typical of such observed distributions, this distribution has a large tail (mean = 29 blocks, min = 4, max = 150, standard deviation $\cong$25, squared coefficient of variation 0.7). The other memory size distribution we will use is the discrete uniform distribution on the integers 1,. . .,150. While this is admittedly an artificial choice, it provides a contrast to the other memory size distribution since the coefficient of variation for uniform $(CV_2 \cong 0.3)$ is smaller than for that of the Univac 1100/82 memory size distribution given above.

<u>Example</u> <u>5.1</u>: The SL-M/M/∞//N_P queue. We first consider the case where the customer population is finite. The arrival rate given n customers in system is therefore $\lambda_n=(N_P-n)\lambda$; $\lambda_n=0$ for $n\geq N_P$. To fix the rest of the parameters we have chosen $\mu=1.0$, $r_n(k) = k$, $N_P=20$, $m=150$, and $M=300$.

We first consider the uniform memory size distribution case. Graphs of the $\overline{L}$ and $\overline{N}$ values versus λ from both the exact and approximate solutions are given in Figures 5.1.1

65-74.

[11] U. Herzog, L. Woo., and K. M. Chandy, Solution of queueing problems by a recursive technique, IBM Journal of Research and Development 19, 3 (May 1975) 295-300.

[12] A. G. Konheim and M. Reiser, A queueing model with finite waiting room and blocking, Journal of the ACM 23, 2 (April 1976) 328-341.

[13] A. G. Konheim and M. Reiser, Finite Capacity Queueing Systems with Applications in Computer Modelling, SIAM J. Computing 7, 2 (May 1978) 210-229.

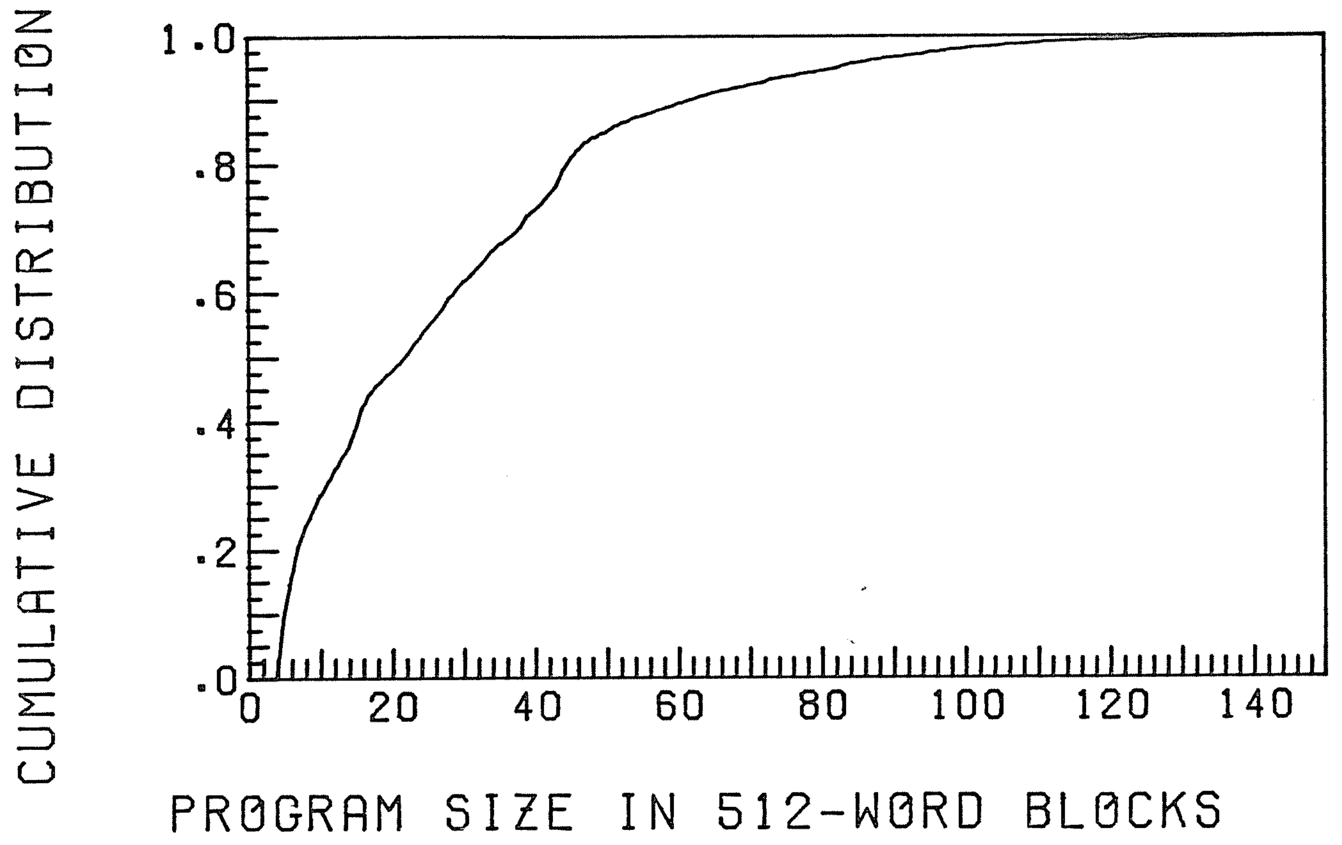
[14] K. J. Omahen, Capacity bounds for multiresource queues, Journal of the ACM 24, 4 (October 1977) 646-663.

[15] K. Omahen and V. Marathe, A queueing model for a multiprocessor system with partitioned memory, Computer Science Technical Report 132, Purdue University, (January 1975).

[16] C. H. Sauer, Confidence intervals for queueing simulations of computer systems, Performance Evaluation Review 8, 1 and 2 (Spring-Summer 1979) 45-55.

[17] G. W. Stewart, Introduction to Matrix Computations, (Academic Press, New York, 1973).

FIGURE 5.1
UNIVAC 1100/82 MEMORY SIZE DISTRIBUTION



and 5.1.2. Figure 5.1.3 gives a plot of the relative error in the approximate solution. The "exact solution" values were calculated using the recursive method of Herzog *et al* [11] and the boundary set B_1 (see Appendix C).

Figures 5.1.4, 5.1.5, and 5.1.6 provide the same information for the Univac 1100/82 distribution case.

We see that the accuracy of the approximate method of Brown *et al* is good. The maximum difference in $\bar{L}$ values is 2.4%; the maximum difference in $\bar{N}$ values is 4.0%. The average relative error in the approximate solution case is less than 0.5% for the uniform memory size distribution and approximately one per cent for the Univac 1100/82 memory size distribution. The $\bar{L}$ values from the approximate model are not exactly correct, but the differences are so small that they would probably be undetectable in a computer system model.

Each exact solution required about 1 second of CPU time on the Univac 1100/82. This, of course, was after the the probabilities $P\{L_2^\infty=v|L_1^\infty=u\}$ and $P\{L_2^\infty\geq v|L_1^\infty=u\}$ had been calculated. This calculation took about 200 seconds; since $P\{L_2^\infty=v|L_1^\infty=u\}$ does not depend on λ this calculation only had to be done once for each memory size distribution. These times could have been reduced by taking advantage of the fact that $N_P = 20$. We would have then had to repeat this calculation for the next example. We thus based the calculations on $N_P = 30$. The times required for the approximate solution were negligible once $Pr\{L_k=j\}$ was

FIGURE 5.1.1
FINITE POPULATION, UNIFORM MEMORY SIZE CASE

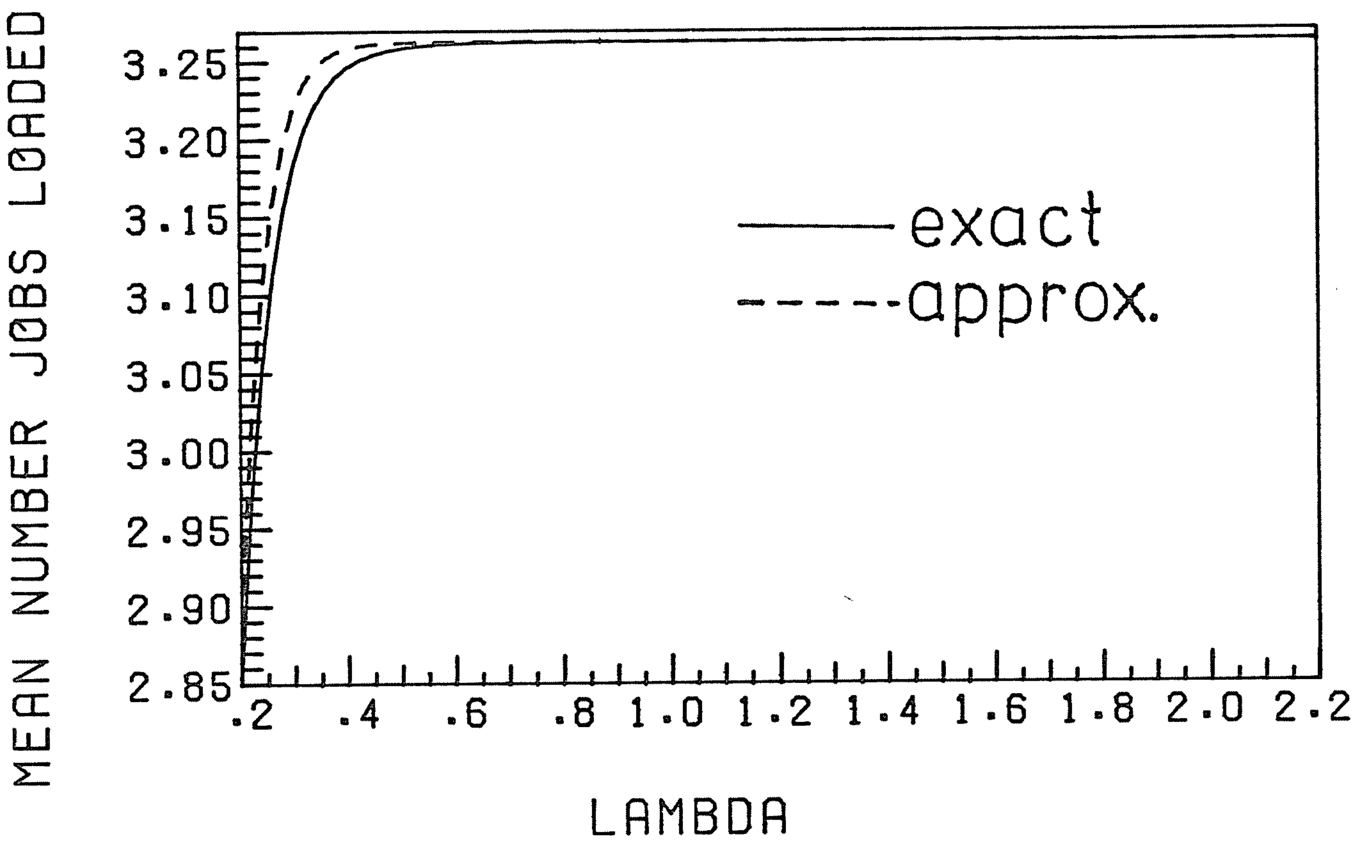


FIGURE 5.1.2
FINITE POPULATION, UNIFORM MEMORY SIZE CASE

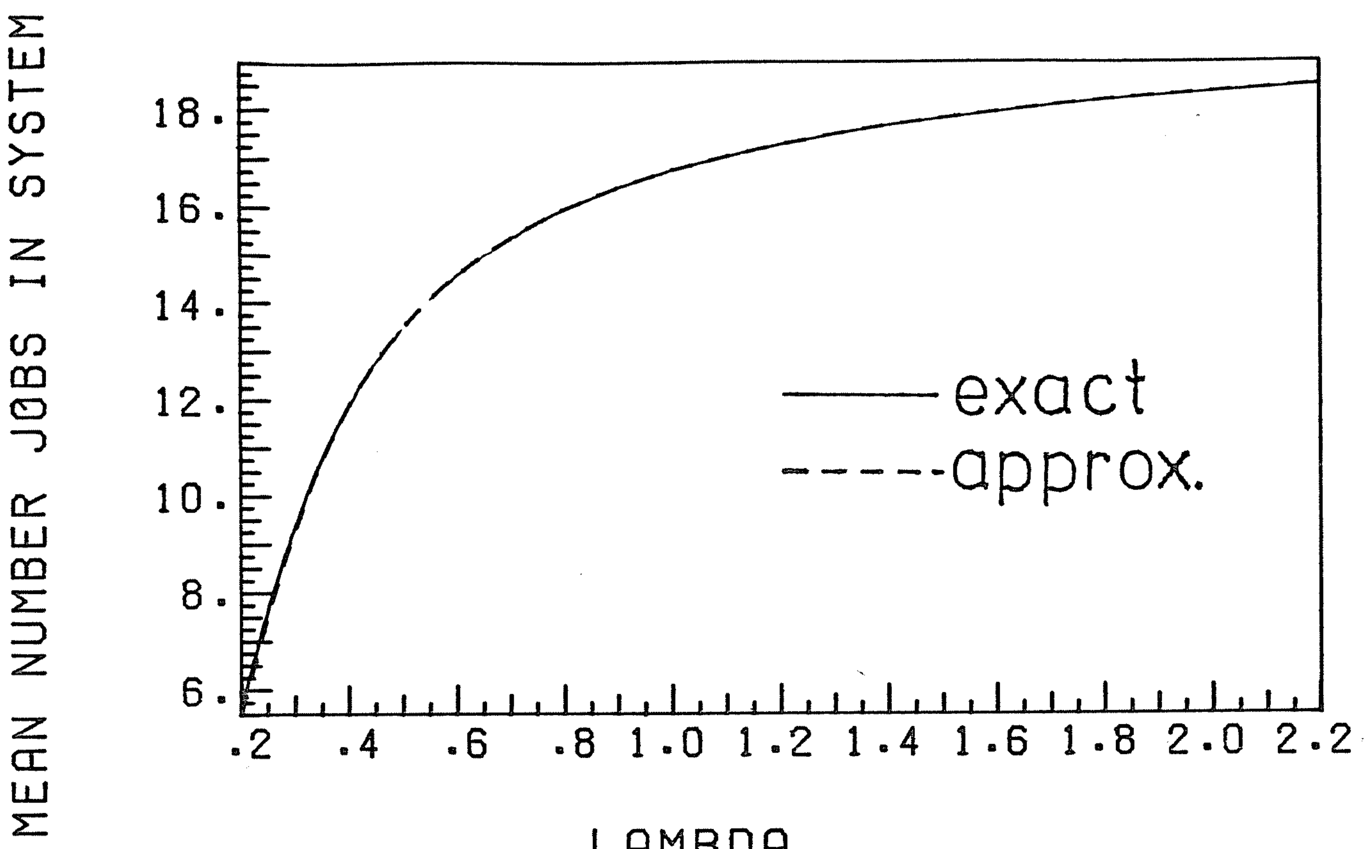


FIGURE 5.1.3
FINITE POPULATION, UNIFORM MEMORY SIZE CASE

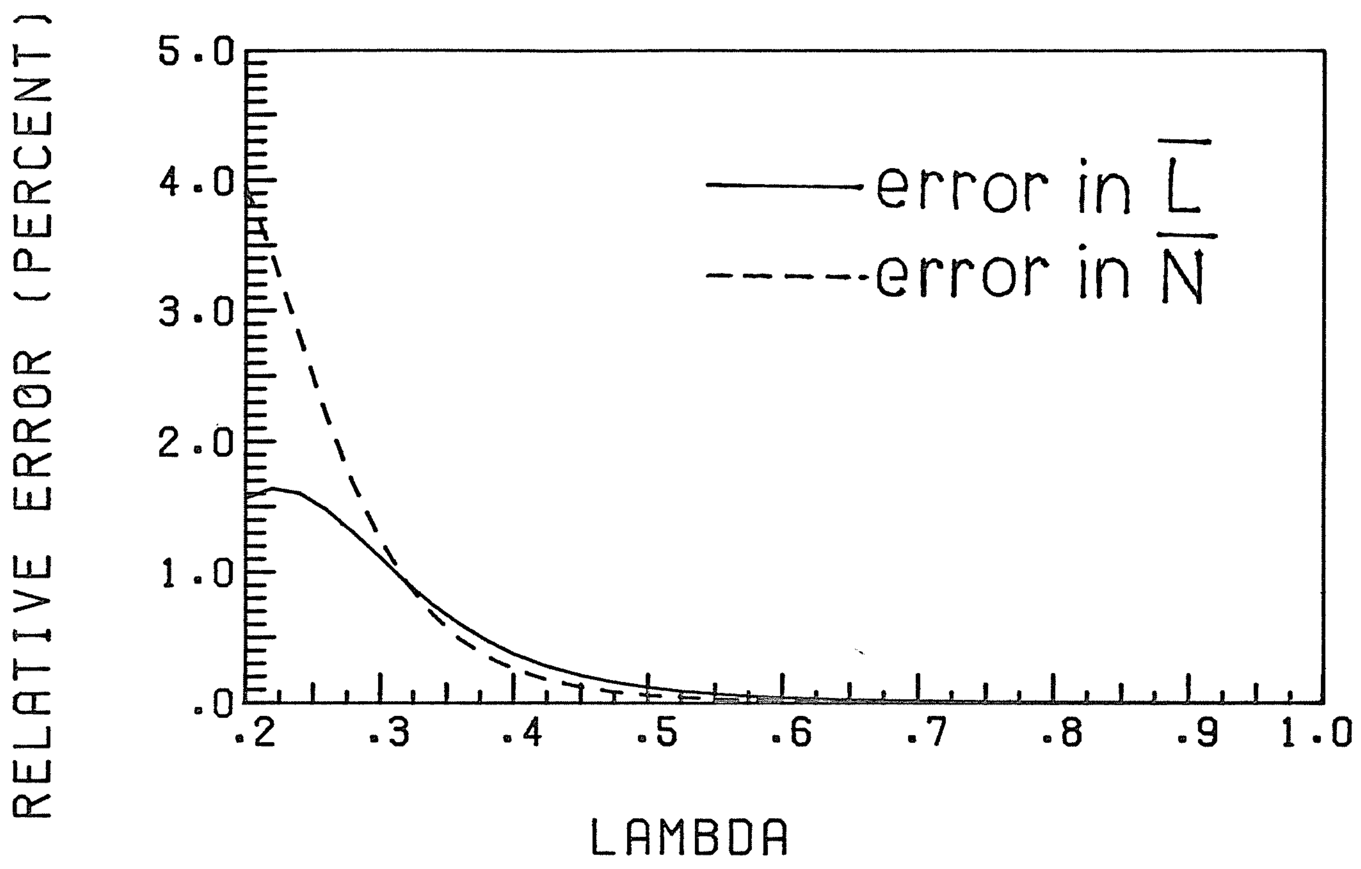


FIGURE 5.1.4
FINITE POPULATION, 1100/82 MEMORY SIZE CASE

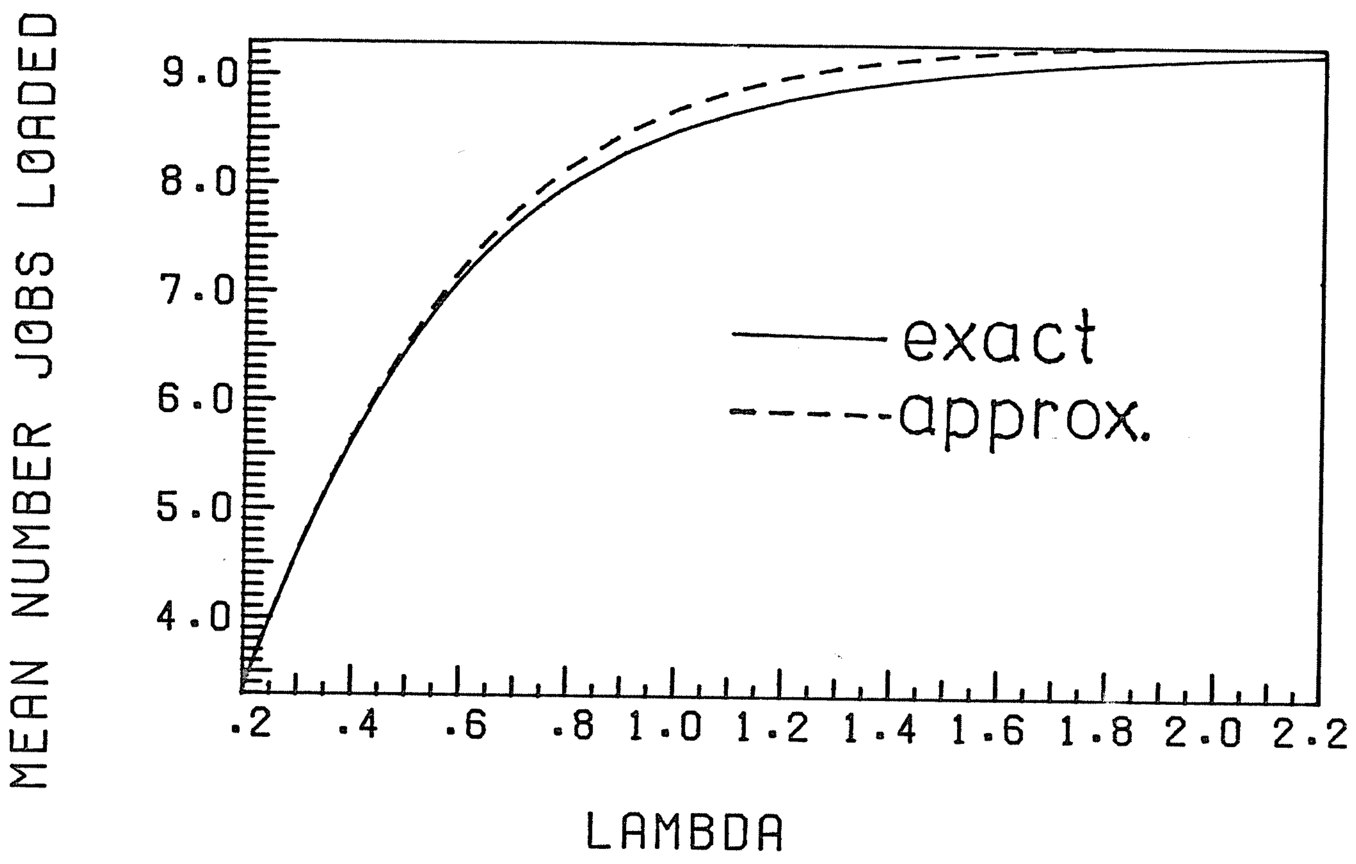


FIGURE 5.1.5
FINITE POPULATION, 1100/82 MEMORY SIZE CASE

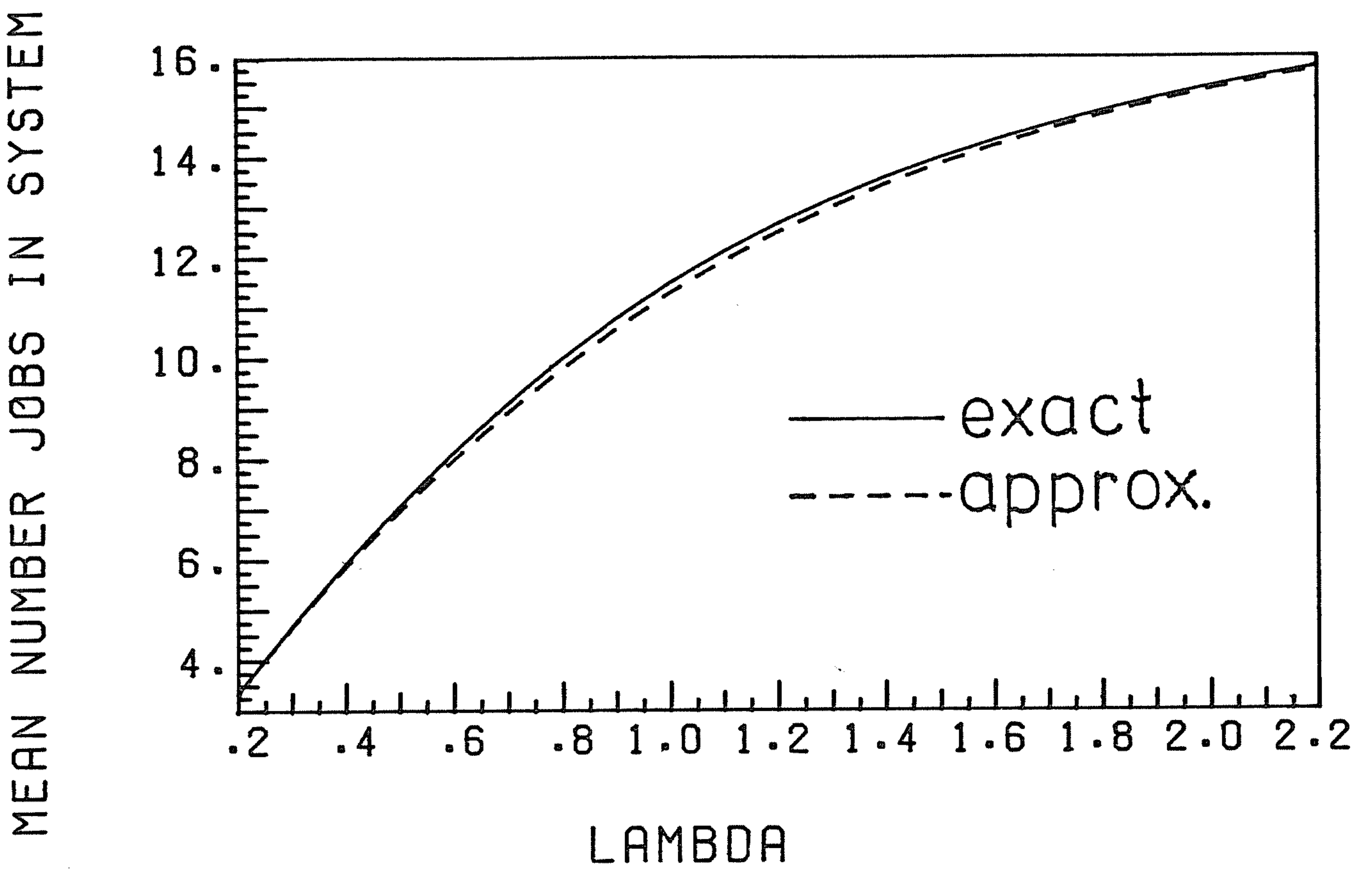
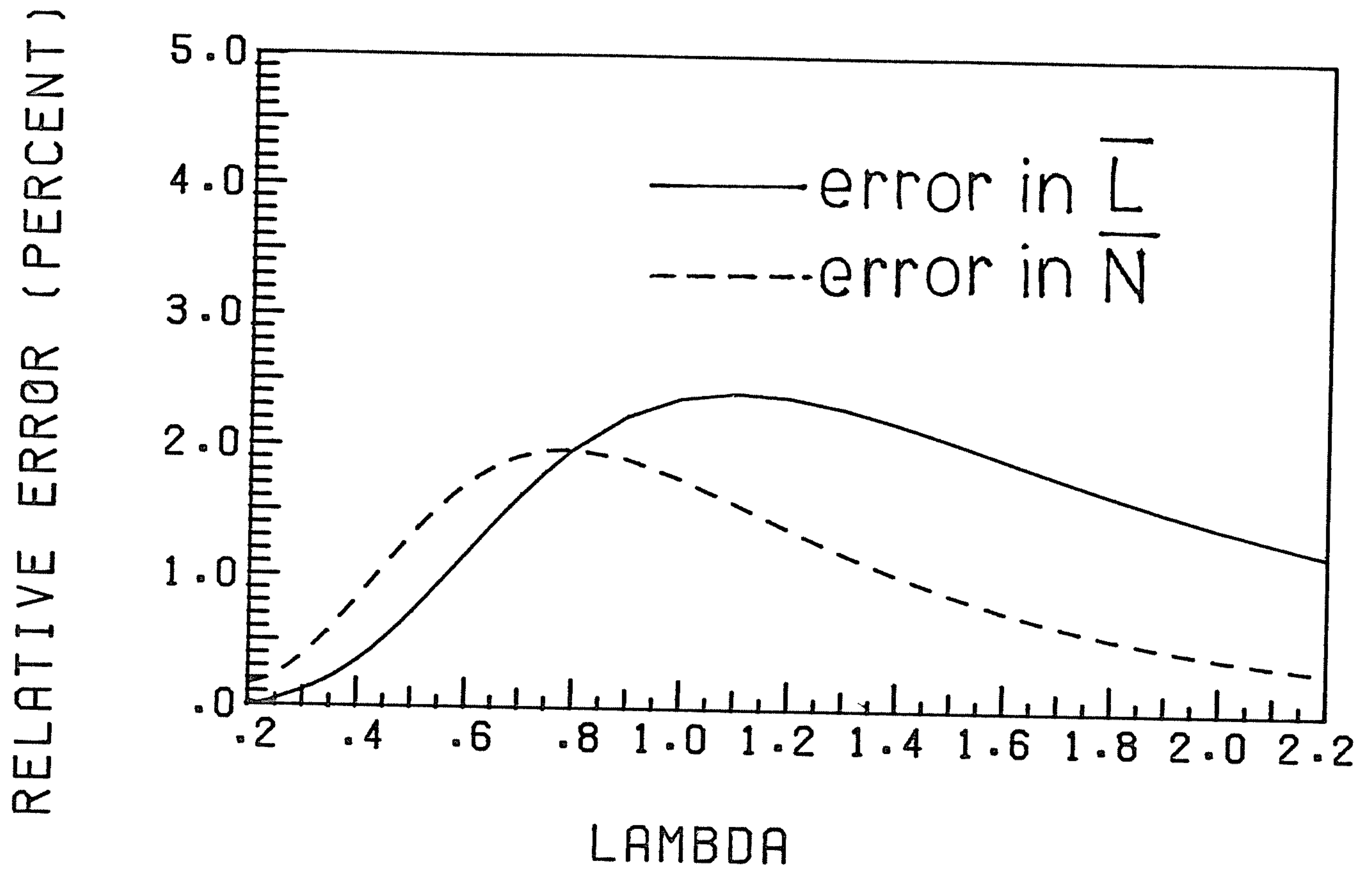


FIGURE 5.1.6
FINITE POPULATION, 1100/82 MEMORY SIZE CASE



known. Calculation of $Pr\{L_k=j\}$ required less than one minute of CPU time for all of the examples considered here. $\square$

Example 5.2: The SL-M/M/∞/N_P queue. We now consider the case where the customer population is infinite ($\lambda_n=\lambda$), but the waiting room at the server is finite ($\lambda_n=0$ for $n\geq N_P$). This system is a "loss" system in the sense that if a customer arrives to find N_P other customers already in system, then that customer departs without being served. Our primary interest in this system is that it may be used as an approximate model of the SL-M/M/∞ queue (where there is sufficient waiting room) when the probability of loss is small. The other parameters of this system are the same as the last example except that we used $N_P=30$.

Figures 5.2.1 through 5.2.3 give plots of $\bar{L}$ and $\bar{N}$ versus λ for this system in the uniform distribution case. Figures 5.2.4 through 5.2.6 show the results in the Univac 1100/82 memory size distribution case. The "exact solution" values were calculated using the recursive method and the boundary set B_2 (See Appendix C). However, double precision arithmetic had to be used throughout to keep negative probabilities from appearing in the π vectors. This is an example of the numerical problems often associated with the recursive method of solution. (See Appendix C for discussion of this problem.) Each of these solutions required about 7 seconds of CPU time. As before,

FIGURE 5.2.1
INFINITE POPULATION, UNIFORM MEMORY SIZE CASE

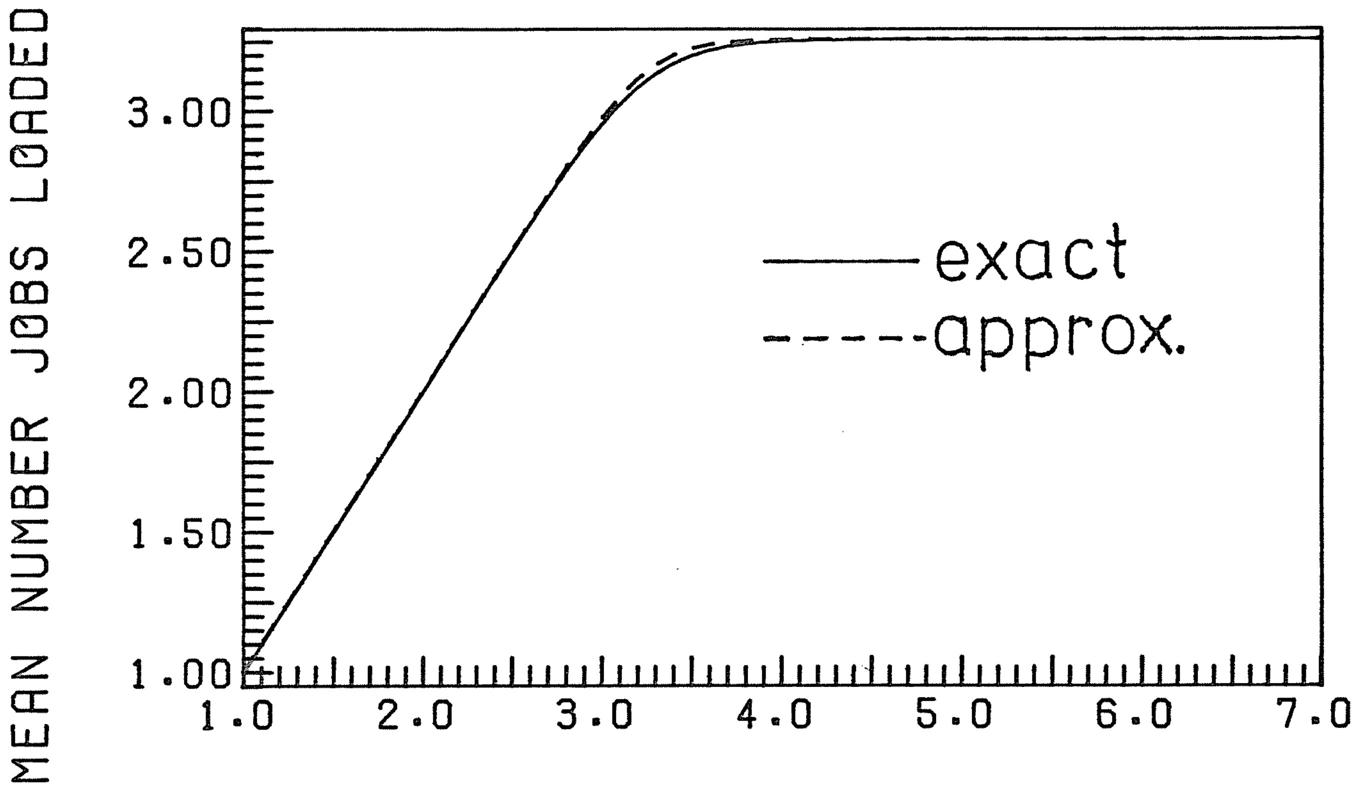


FIGURE 5.2.2
INFINITE POPULATION, UNIFORM MEMORY SIZE CASE

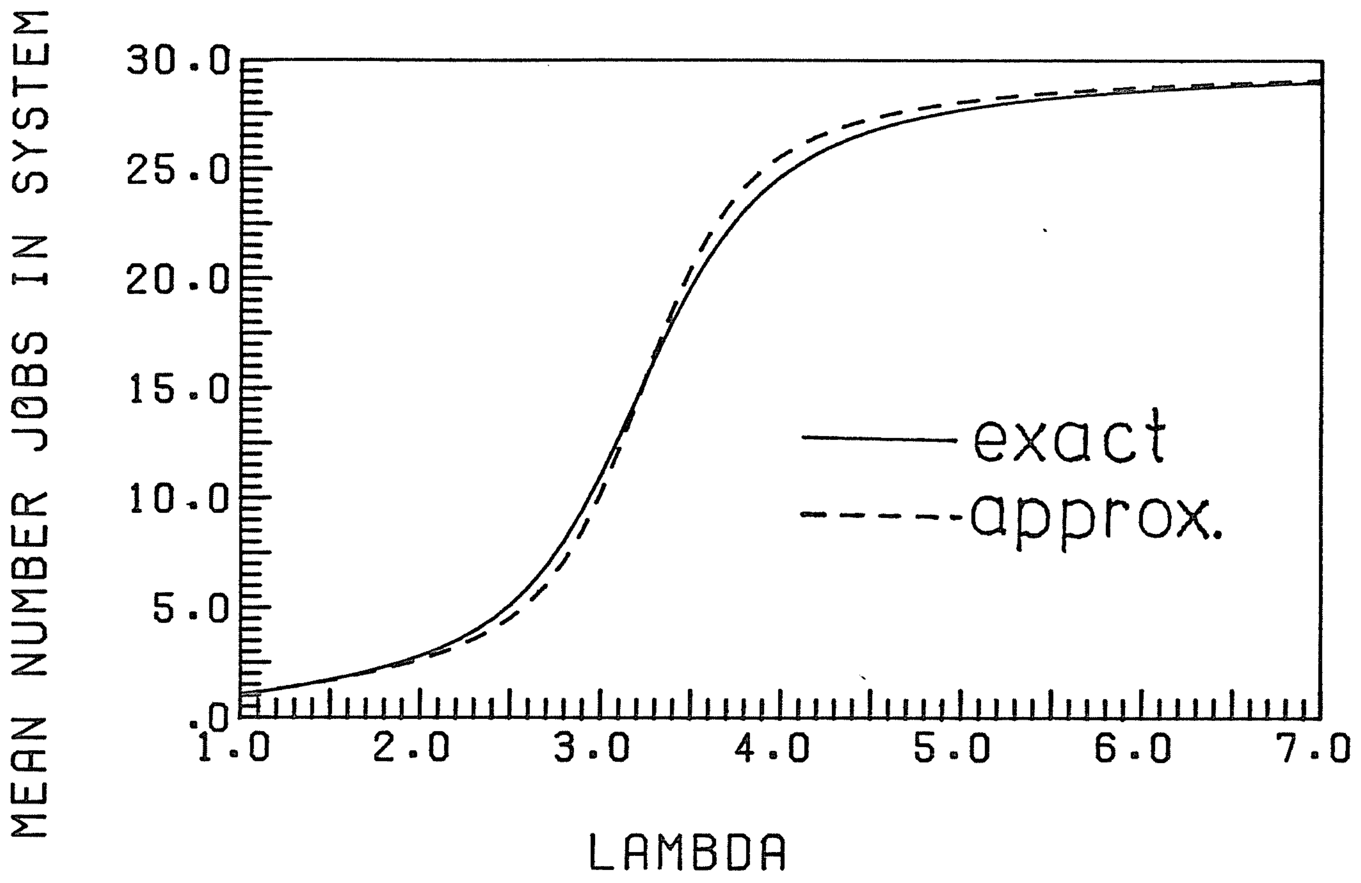


FIGURE 5 .2.3

INFINITE POPULATION, UNIFORM MEMORY SIZE CASE

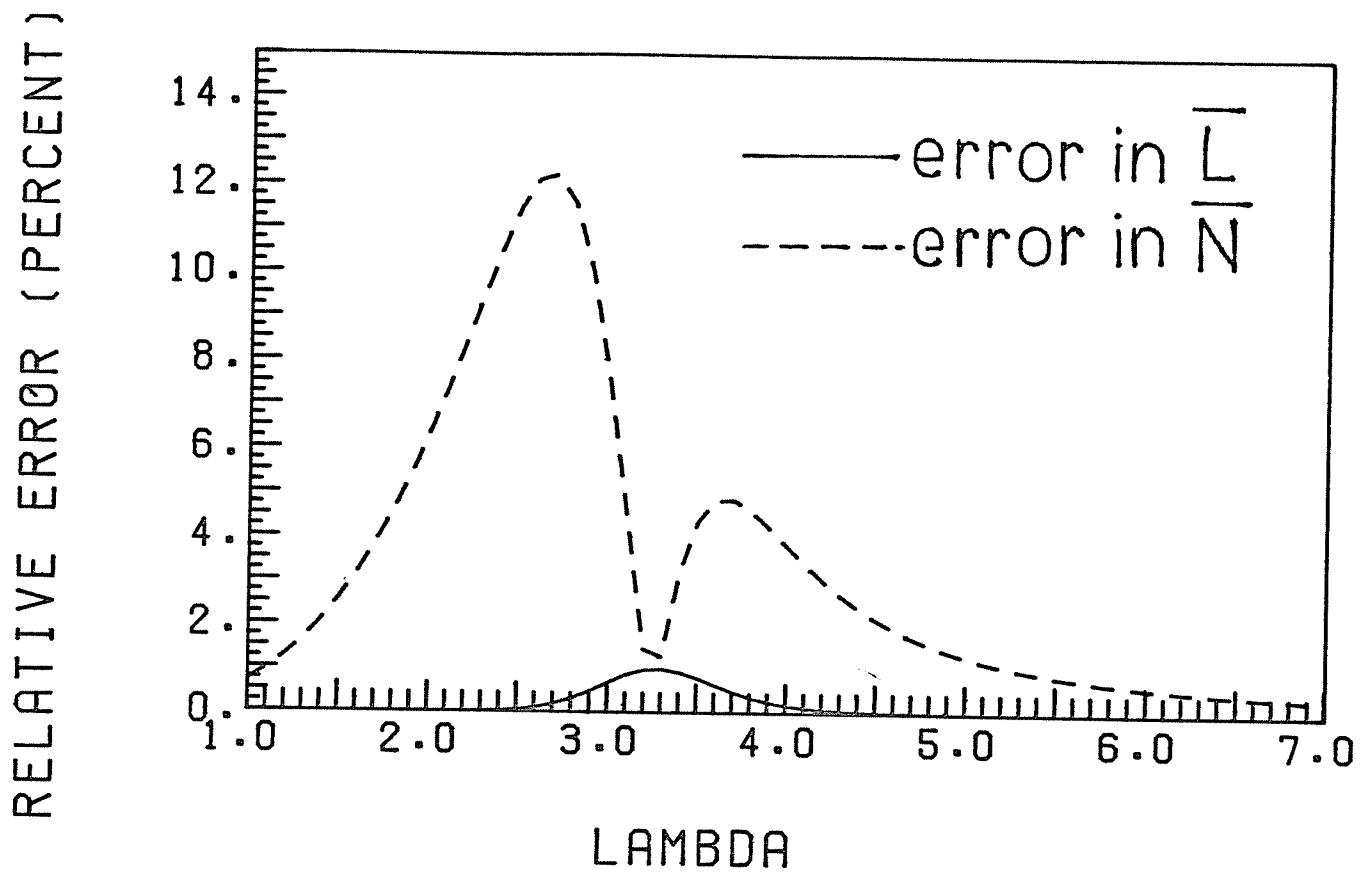


FIGURE 5.2.4
INFINITE POPULATION, 1100/82 MEMORY SIZE CASE

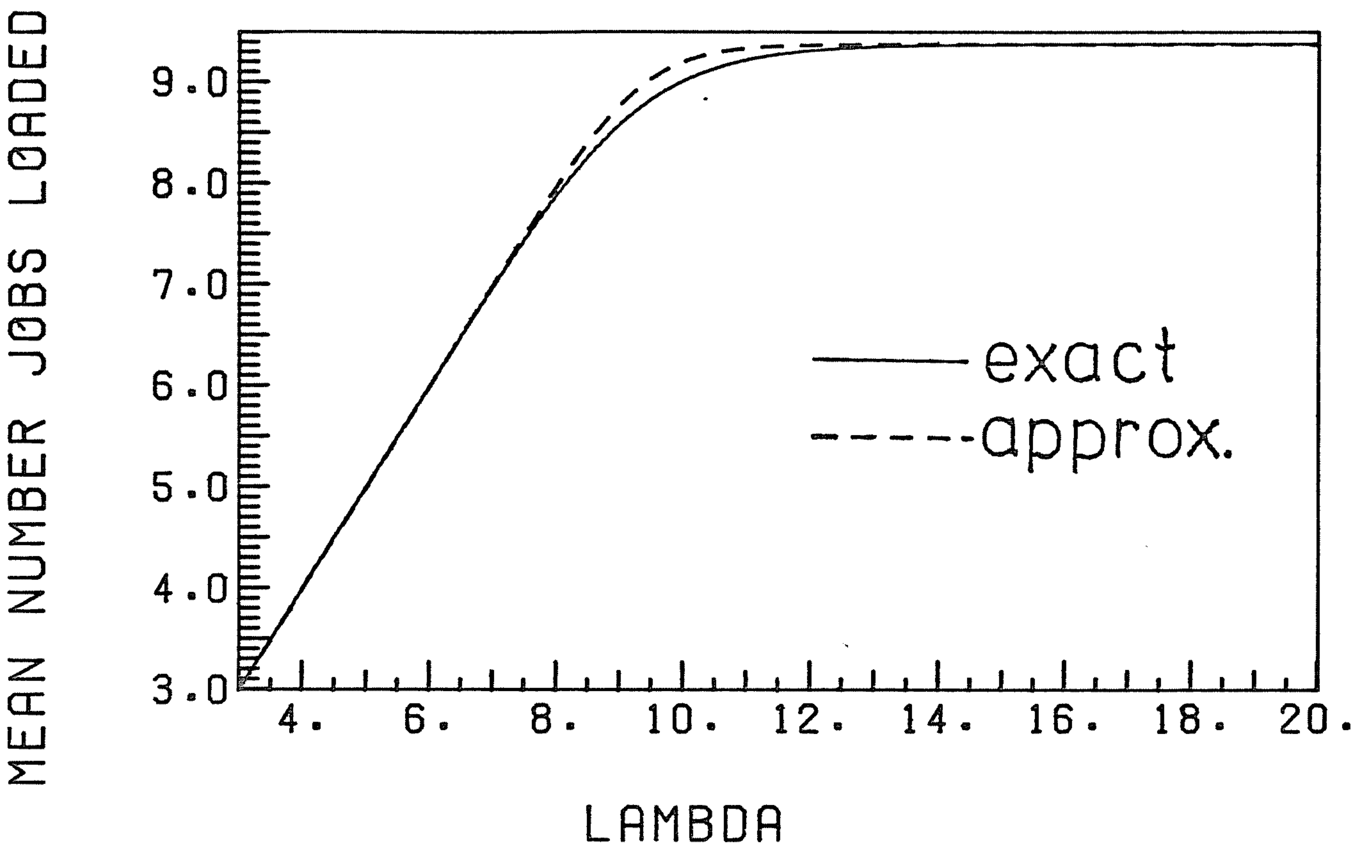


FIGURE 5.2.5
INFINITE POPULATION, 1100/82 MEMORY SIZE CASE

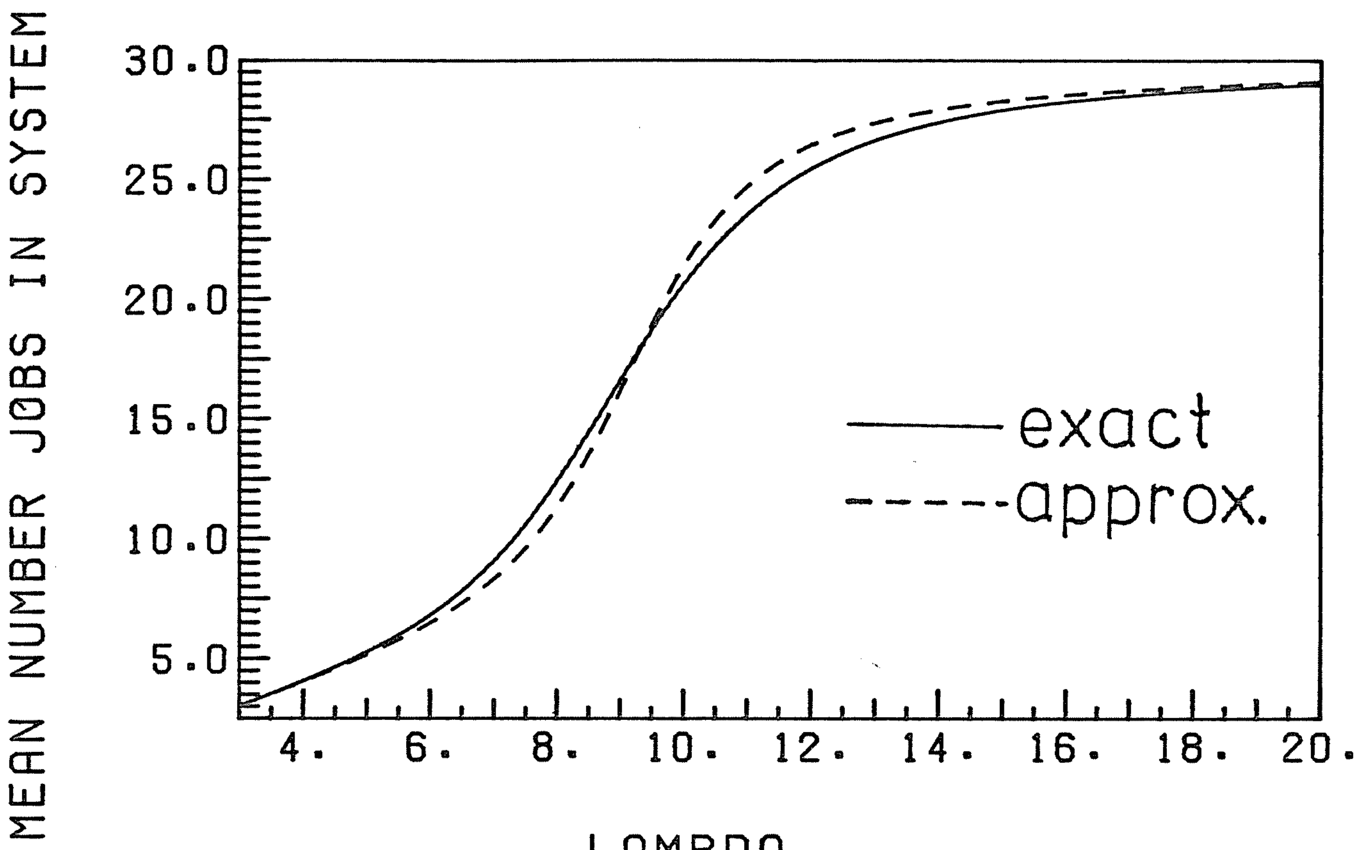
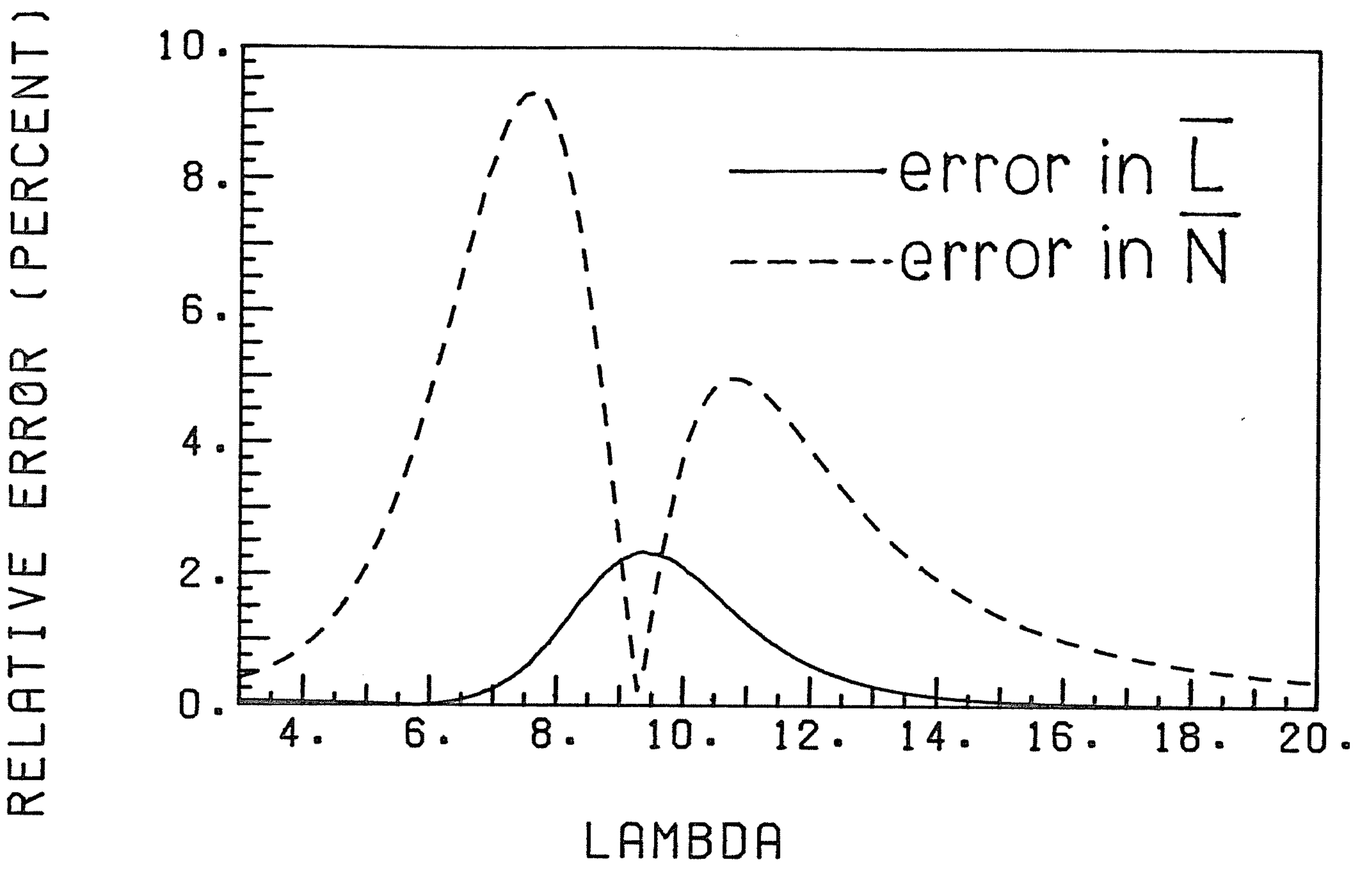


FIGURE 5.2.6
INFINITE POPULATION, 1100/82 MEMORY SIZE CASE



times for the approximate solutions were negligible.

Examining Figures 5.2.5 and 5.2.6, we note that the maximum difference between the two solutions is higher in this case than it was in Example 5.1. The difference in $\overline{N}$ values reaches 12.2%; the difference in $\overline{I}$ values is always less than 3%.

Graphs of the loss probability ($Pr\{N(t) = N_E\}$) versus λ are given in Figures 5.2.7 and 5.2.8. These figures show that even if the loss probability is small, the difference between the exact and approximate solution values of $\overline{N}$ can be large.

Thus, in general we would conclude that the approximate solution of [4] is more suitable to the finite than the infinite population case. Since the former is the more common case in computer-system modeling, the approximate solution is to be highly recommended.

6 . CONCLUDING REMARKS

We have derived an efficient numerical procedure for the exact solution of a class of storage limited exponential queueing systems and used this solution to evaluate the accuracy of a well-known approximate solution for the same problem [4]. While the approximate solution is more accurate in the finite-population rather than the infinite-population cases, the maximum relative error we encountered was 12.2%. Since the approximate solution

FIGURE 5.2.7
INFINITE POPULATION, UNIFORM MEMORY SIZE CASE

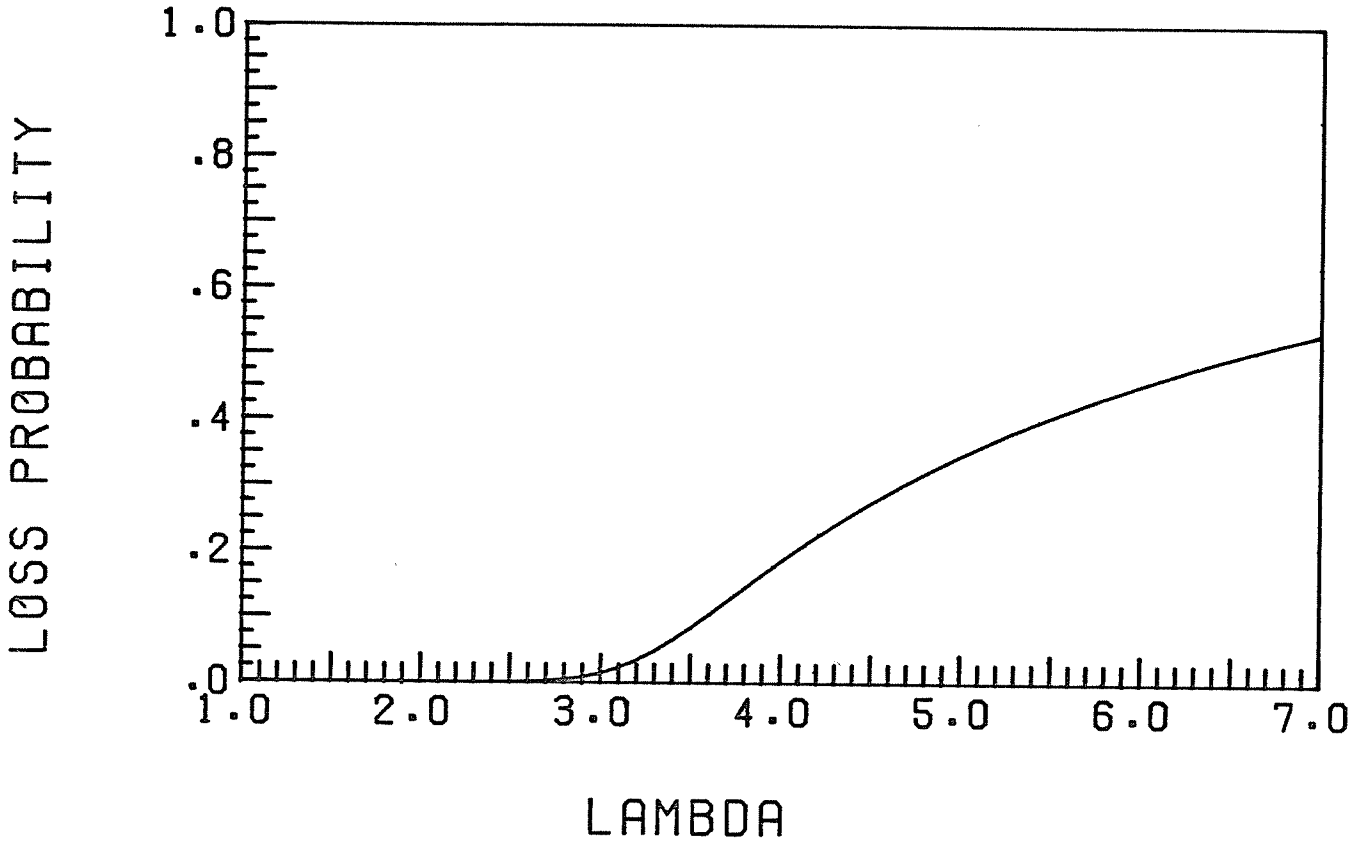
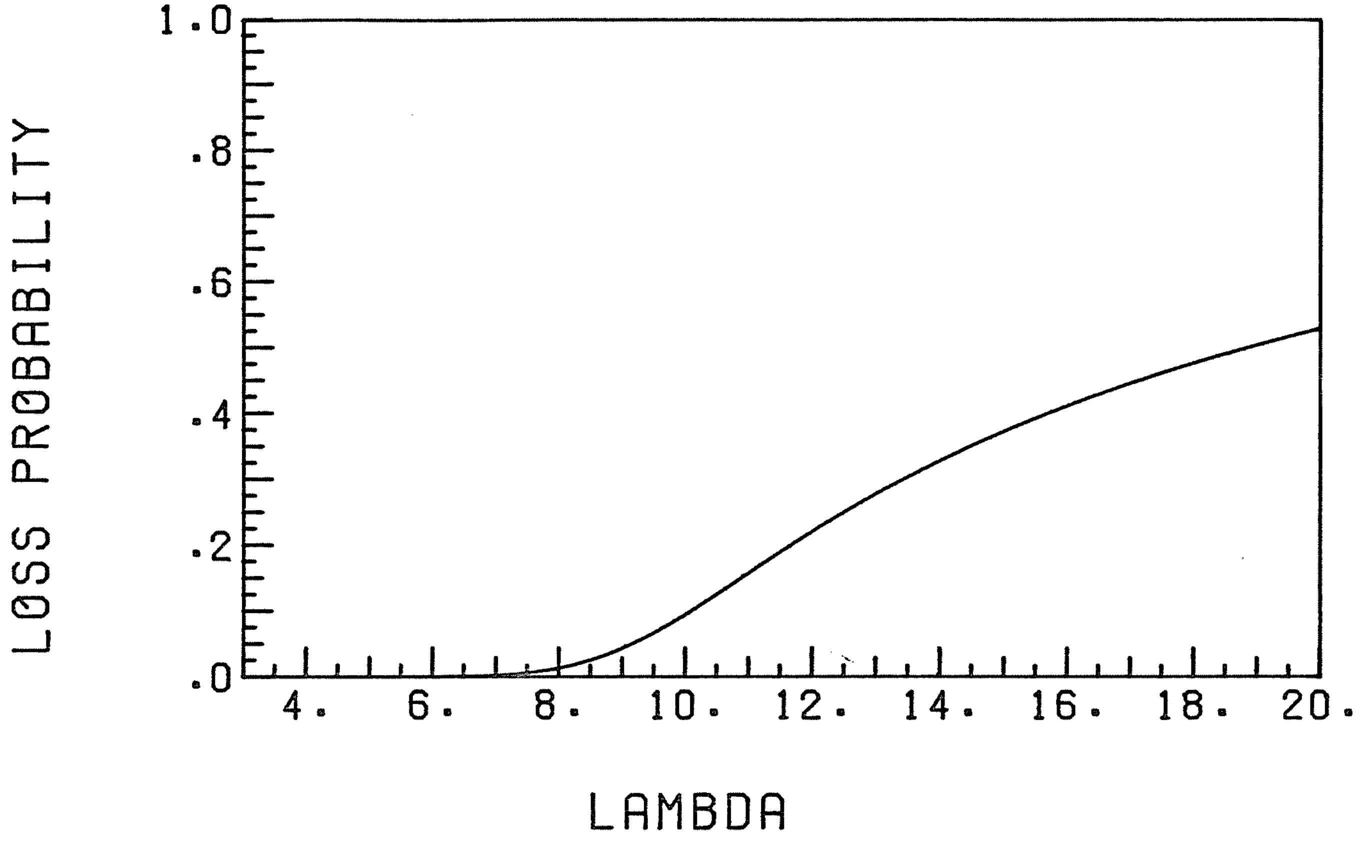


FIGURE 5.2.8
INFINITE POPULATION, 1100/82 MEMORY SIZE CASE



requires trivial computational effort, it is to be highly recommended, especially in the finite-population case. Since this case is the most common in computer-system models, the approximate solution should be sufficient for most applications.

Acknowledgements: This work originated in the author's Ph. D. thesis [6] which was written at the University of Maryland under the supervision of Dr. A. K. Agrawala, whose support and guidance I gratefully acknowledge. The current presentation and examples were completed at the University of Wisconsin-Madison.

APPENDIX A
DERIVATION OF INFINITESIMAL PARAMETERS FOR (L(t),N(t))

We know that that $q_{(\underline{z},n),S(j,k)}=\emptyset$ unless $k=n+1$ or $k=n-1$. Considering the former case first, suppose that an arrival occurs while $(\underline{Z}(t),N(t))=(\underline{z},n)$. If $R(\underline{z})\leq n$, then $(\underline{z},n)$ represents a system state in which either memory is full or the memory queue is non-empty. In either case the arriving job cannot be loaded and therefore the MPL does not change. Thus, if $R(\underline{z})\leq n$, then $q_{(\underline{z},n),S(j,n+1)}=\emptyset$ unless $j=R(\underline{z})$. Similarly if $R(\underline{z})>n$ then $R(\underline{z})\geq n+1$ and hence $q_{(\underline{z},n),S(j,n+1)}=\emptyset$ unless $j=\min(R(\underline{z}),n+1)$. Combining these conditions we have:

$$q_{(\underline{z},n),S(j,n+1)}=\begin{cases}\emptyset & R(\underline{z})\leq n,\ j\neq R(\underline{z})\\ \emptyset & R(\underline{z})>n,\ j\neq \min(R(\underline{z}),n+1)\\ & \\ \lambda_n & R(\underline{z})\leq n,\ j=R(\underline{z})\\ \lambda_n & R(\underline{z})>n,\ j=\min(R(\underline{z}),n+1)\end{cases}$$

To evaluate $q_{(\underline{z},n),S(j,n-1)}$, let $\underline{y}\in S(j,n-1)$. Then if $R(\underline{z})\geq n$ and $\underline{y}$ is reachable in one transition from $\underline{z}$, we know that $R(\underline{y})\geq n-1$ since the multiprogramming level can decrease by at most one when a job departs. On the other hand if $R(\underline{z})<n$ and $\underline{y}$ is reachable from $\underline{z}$, then $R(\underline{y})$ can take any

value from $R(\underline{z})-1$ to M. Thus:

$$q_{(\underline{z},n),S(j,n-1)} = \begin{cases} \mu_{n,j} & R(\underline{z})\geq n,\ j=n-1 \\ \displaystyle\sum_{(\underline{y},n-1)\in S(j,n-1)} \frac{\mu_{n,k}N_D(\underline{z},\underline{y},k)}{k}\, f(y^M) & \text{when } R(\underline{z})<n \ (k=\min(R(\underline{z}),n)) \\ \emptyset & \text{otherwise} \end{cases}$$

Next we clearly have

$$Q_{(j,n)}=\lambda_n+\mu_{n,j}$$

since $q_{(\underline{z},n)}$ is constant for all states in the set $S(j,n)$.

Because $q_{(\underline{z},n),S(j,k)}=\emptyset$ unless $k=n+1$ or $k=n-1$ it follows that only parameters of the form $Q_{(j,n),(k,n+1)}$ and $Q_{(j,n),(k,n-1)}$ are non-zero. The values of these parameters take several different forms depending on j,n and k.

<u>Case I</u>: $Q_{(j,n),(k,n+1)}$ with $j<n$. If $j<n$ then $(\underline{z},n)\in S(j,n)$ implies $R(\underline{z})<n$. Therefore $q_{(\underline{z},n),S(j,n+1)} = \lambda_n$ for all $(\underline{z},n)\in S(j,n)$, and hence $Q_{(j,n),(j,n+1)} = \lambda_n$, $j<n$. This equation merely reflects the fact that if a job arrives to find a non-empty memory queue, then the job always joins the end of the memory queue.

<u>Case II</u>: $Q_{(j,n),(k,n+1)}$ with $j=n$. The set $S(n,n)$ consists of the two subsets:

$$S^1_{(n,n)}=\{(\underline{z},n)\in S(n,n)\ |\ R(\underline{z})=n\}$$
$$S^2_{(n,n)}=\{(\underline{z},n)\in S(n,n)\ |\ R(\underline{z})>n\}$$

If an arrival occurs while $(\underline{Z}(t),N(t))\in S^1_{(n,n)}$ then the arrival does not get loaded, but instead joins the queue. On the other hand, if an arrival occurs while $(\underline{Z}(t),N(t))\in$

$S^2_{(n,n)}$, then the arrival is loaded and no memory queue forms. Thus:

$$Q_{(n,n),(n,n+1)} = \lambda_n \sum_{(\underline{z},n) \in S^1_{(n,n)}} \pi((\underline{z},n) \mid (n,n))$$

$$Q_{(n,n),(n+1,n+1)} = \lambda_n \sum_{(\underline{z},n) \in S^2_{(n,n)}} \pi((\underline{z},n) \mid (n,n))$$

The sums represent the probability that $R(\underline{Z})=n$ and $R(\underline{Z})>n$ respectively, given that $R(\underline{Z}) \geq n$, where $\underline{Z}$ is an M-vector of i. i. d. random variables with distribution $F(x)$. Recalling that $L^\infty_k = R(\underline{Z}^\infty_k)$ and using the known stationary distribution of $\{\underline{Z}^\infty_k\}$ we see that

$$P\{\ R(\underline{Z}) = n \mid R(\underline{Z}) \geq n\} = P\{\ L^\infty_1 = n \mid L^\infty_1 \geq n\}$$

and

$$P\{\ R(\underline{Z}) > n \mid R(\underline{Z}) \geq n\} = P\{\ L^\infty_1 = n \mid L^\infty_1 \geq n\}.$$

Therefore:

$$Q_{(n,n),(n,n+1)} = \lambda_n\ P\{L^\infty_1 = n \mid L^\infty_1 \geq n\}$$
$$Q_{(n,n),(n+1,n+1)} = \lambda_n\ P\{L^\infty_1 > n \mid L^\infty_1 \geq n\}$$

From Theorem 2.3 we have

$$P\{L^\infty_1 = n \mid L^\infty_1 \geq n\} = \frac{P\{L^\infty_1 = n\}}{P\{L^\infty_1 \geq n\}} = \frac{F^{(n)}(M) - F^{(n+1)}(M)}{F^{(n)}(M)}$$

and similarly

$$P\{L^\infty_1 > n \mid L^\infty_1 \geq n\} = \frac{F^{(n+1)}(M)}{F^{(n)}(M)}.$$

<u>Case</u> <u>III</u>: $Q_{(j,n),(k,n-1)}$ with $j=n$. If $(\underline{z},n) \in S(n,n)$ we have observed that $R(\underline{z}) \geq n$. Hence $q_{(\underline{z},n),S(n-1,n-1)} = \mu_{n,n}$

for all $(\underline{z},n) \in S(n,n)$ and

$Q_{(n,n),(n-1,n-1)} = \mu_{n,n}$.

<u>Case</u> <u>IV</u>: $Q_{(j,n),(k,n-1)}$ with $j<n$. This case was discussed in Section 2 of the paper.

The details of evaluating $P\{L_2^{\infty}=j \mid L_1^{\infty}=k\}$ and $P\{L_2^{\infty} \geq j \mid L_1^{\infty}=k\}$ are given in Appendix B.

APPENDIX B

EVALUATION OF COEFFICIENTS FOR EQUATION 2.8

B.1 EVALUATION OF $P\{L_2^\infty=v|L_1^\infty=u\}$

Since we know $P\{L_1^\infty= u\}$ from Theorem 2.3 it is sufficient to evaluate $P\{L_1^\infty= u, L_2^\infty= v\}$. Recall that as a notational convenience we have defined

$$S_i^j=\sum_{k=i}^{j} X_k.$$

Case I: $v<u-1$. Because jobs depart the system one at a time, the maximum decrease in the MPL from one loader activation to the next is one. Therefore $P\{L_1^\infty=u,L_2^\infty=v\}=0$ whenever $v<u-1$.

Case II: $v=u-1$. This case is trivial when $u=1$ so we assume that $u\geq 2$. The event $E=[L_1^\infty=u,L_2^\infty=u-1]$ is equivalent to

$$[S_1^u\leq M,\ S_1^{u+1}>M,\ S_2^{u+1}>M].$$

The latter may be rewritten using independent random variables as

$$[X_1+S_2^u\leq M,\ X_1+S_2^u+X_{u+1}>M,\ S_2^u+X_{u+1}>M].$$

The middle condition is redundant because $X_1>0$. Therefore

$$P\{E\}=P\{X_1+S_2^u\leq M,\ S_2^u+X_{u+1}>M\}.$$

If $S_2^u+X_{u+1}>M$ then $S_2^u\geq M-m+1$. However $1\leq X_i\leq m$ which implies that $u-1 \leq S_2^u \leq m(u-1)$. Combining these conditions we get:

$$\max(u-1, M-m+1) \leq S_2^u \leq \min(M-1, m(u-1)) \qquad (B.1)$$

Conditioning on the value of S_2^u we obtain

$$P\{L_1^\infty = u, L_2^\infty = u-1\} = \sum_s P\{S_2^u = s\} \; P\{X_1 + s \leq M, s + X_{u+1} > M\}$$

$$= \sum_s f^{(u-1)}(s) F(M-s)(1-F(M-s)),$$

where the limits on s are given by equation (B.1).

Case III: $u=1, v=1$. In this case $E=[L_1^\infty = 1, L_2^\infty = 1]$ is equivalent to

$$[X_1 \leq M, \; X_1 + X_2 > M, \; X_2 \leq M, \; X_2 + X_3 > M].$$

Therefore

$$P\{L_1^\infty = 1, L_2^\infty = 1\} = P\{X_1 + X_2 > M, \; X_2 + X_3 > M\}$$

$$= \sum_{x=1}^{m} f(x) \; P\{X_1 > M-x, \; X_3 > M-x\}$$

$$= \sum_{x=1}^{m} f(x) \; (1-F(M-x))^2.$$

Case IV: $u=1$, $v>1$. In this case the event $E=[L_1^\infty = 1, L_2^\infty = v]$ is equivalent to

$$[S_1^1 \leq M, \; S_1^2 > M, \; S_2^{v+1} \leq M, \; S_2^{v+2} > M].$$

This can be rewritten using independent random variables as

$$[X_1 \leq M, \; X_1 + X_2 > M, \; X_2 + S_3^{v+1} \leq M, \; X_2 + S_3^{v+1} + X_{v+2} > M].$$

Conditioning on the value of X_2 we obtain

$$P\{E\}=\sum_{x=1}^{m} f(x)\ P\{X_1>M-x,\ S_3^{v+1}\leq M-x,\ S_3^{v+1}+X_{v+2}>M-x\}.$$

But the event $[S_3^{v+1}\leq M-x,\ S_3^{v+1}+X_{v+2}>M-x]$ is independent of X_1 and is equivalent to the event that the MPL is v-1 in a main memory of size M-x. Therefore

$$P\{L_1^{\infty}=1,L_2^{\infty}=v\}=\sum_{x=1}^{m} f(x)(1-F(M-x))\ (F^{(v-1)}(M-x)-F^{(v)}(M-x)).$$

<u>Case</u> <u>V</u>: u=v, v>1. The event $E=[L_1^{\infty}=u,L_2^{\infty}=u]$ is equivalent to

$$[S_1^u\leq M,\ S_1^{u+1}>M,\ S_2^{u+1}\leq M,\ S_2^{u+2}>M],$$

or

$$[X_1+S_2^u\leq M,\ X_1+S_2^u+X_{u+1}>M,\ S_2^u+X_{u+1}\leq M,\ S_2^u+X_{u+1}+X_{u+2}>M].$$

If $X_1+S_2^u+X_{u+1}>M$ then $S_2^u\geq M-2m+1$. Similarly if $S_2^u+X_{u+1}\leq M$ then $S_2^u\leq M-1$. But we know that $u-1\leq S_2^u\leq m(u-1)$. Combining these conditions we get:

$$\max(u-1,M-2m+1)\ \leq\ S_2^u\ \leq\ \min(m(u-1),M-1) \qquad (B.2)$$

Conditioning on the event $[S_2^u=s]$ gives us

$$P\{E\}=\sum_{s} f^{(u-1)}(s)P\{E_1\},$$

where

$$E_1=\ [X_1\leq M-s,\ X_1+X_{u+1}>M-s,\ X_{u+1}\leq M-s,\ X_{u+1}+X_{u+2}>M-s].$$

In order for this event to occur we must have $M-(s+m)+1\leq X_{u+1}\leq M-s$. Since $1\leq X_{u+1}\leq m$ it follows that

$$\max(1,M-(s+m)+1)\leq X_{u+1}\leq\min(m,M-s) \qquad (B.3)$$

Therefore

$$P\{E\}=\sum_{s} f^{(u-1)}(s)\sum_{x} f(x)P\{M-(s+x)<X_1\leq M-s,\ X_{u+2}>M-(s+x)\}$$

$$=\sum_{s} f^{(u-1)}(s)\sum_{x} f(x)[F(M-s)-F(M-(s+x))][1-F(M-(s+x))]$$

where the limits on s and x are given by equations (B.2) and (B.3) respectively.

Case VI: $u>1$, $v>u$. This case is discussed in Section 3 in the paper.

B.2 EVALUATION OF $P\{L_2^\infty \geq v \mid L_1^\infty=u\}$

To calculate $P\{L_2^\infty \geq v \mid L_1^\infty=u\}$ it is only necessary to consider the case $v>u$ since other values can easily be obtained by adding the appropriate values of $P\{L_2^\infty=v \mid L_1^\infty=u\}$. Of course the former could be calculated from a complete set of the latter values, but this approach would be enormously expensive. As above it is sufficient to calculate $P\{L_1^\infty=u, L_2^\infty \geq v\}$.

Case I: $u=1$. $E=[L_1^\infty=u, L_2^\infty \geq v]$ is equivalent to

$$[S_1^1\leq M,\ S_1^2>M,\ S_2^{v+1}\leq M]$$

or

$$[X_1\leq m,\ X_1+X_2>M,\ X_2+S_3^{v+1}\leq M].$$

Conditioning on the value of X_2 we obtain

$$P\{E\}=\sum_{x-1}^{m} f(x)P\{X_1\leq M,\ X_1>M-x,\ S_3^{v+1}\leq M-x\}$$

$$=\sum_{x=1}^{m} f(x)(1-F(m-x))F^{(v-1)}(M-x).$$

Case II: $u>1$. $E=[L_1^\infty=u, L_2^\infty \geq v]$ is equivalent to

$$[S_1^u \leq M,\ S_1^{u+1} > M,\ S_2^{v+1} \leq M]$$

or

$$[X_1 + S_2^u \leq M,\ X_1 + S_2^u + X_{u+1} > M,\ S_2^u + X_{u+1} + S_{u+2}^{v+1} \leq M].$$

Conditioning on the event $[S_2^u = s]$ and then on the event $[X_{u+1} = x]$ we obtain

$$P\{E\} = \sum_s f^{(u-1)}(s) \sum_x f(x) P\{E_1\}$$

where

$$E_1 = [X_1 \leq M-s,\ X_1 > M-(s+x),\ S_{u+2}^{v+1} \leq M-(s+x)]$$

and the limits on s and x are given by equations (3.1) and (3.2) respectively. Clearly

$$P\{E_1\} = [F(m-s) - F(m-(s+x))] F^{(v-u)}(M-(s+x)).\ \square$$

APPENDIX C

THE RECURSIVE METHOD OF SOLVING EQUATION (2.8)

The "recursive" method of [11] can be used to convert the problem of solving $\underline{Q}\ \underline{\Pi}=\emptyset$ to the problem of solving $\underline{R}\ \underline{y}=\emptyset$ where $\underline{R}$ is of order (N_P-L_Q) and a simple relation exists between $\underline{y}$ and $\underline{\Pi}$. In general terms this recursive method proceeds as follows.

One begins by identifying a set of "boundary" states that has the property that all state occupancy probabilities can be expressed in terms of the occupancy probabilities of the boundary states. By substituting the expressions for the non-boundary state occupancy probabilities into the balance equations for the boundary states, one obtains a set of simultaneous equations in the boundary-state probabilities alone, which when taken together with the law of total probability can then be solved for the boundary-state probabilities. Once these probabilities are known, all state occupancy probabilities can then be found. (In a birth-death process, for example, the occupancy probabilities of all states can be expressed in terms of the probability that the system is in state zero. This state is the boundary state for a birth-death process. Once all probabilities have been defined in terms of the residency probability of this state, the law of total probability is invoked to evaluate the boundary state probability.) The advantage of this technique is that one only need solve a small set of simultaneous equations. The disadvantage is that unless a non-trivial portion of the total probability is concentrated on the boundary states, then the overall computation is numerically ill-posed (see below).

We now describe the recursive method for determining the steady state probabilities for the process (L(t),N(t)). To simplify the discussion let us assume that the states of the process (L(t),N(t)) are numbered by a single index i=1,2,. . . . Let B represent the set of boundary states; we may assume without loss of generality that $B=\{1,2,...,n_B\}$. We select B so that if π^i, $i \in B$, are known, then the rest of the π^j's can be easily determined by requiring all but n_B of the equations (2.8) to be satisfied. Let $\underline{v}_i$, $i \in B$, be any linearly independent set of vectors, each of length n_B. Then let $\underline{\pi}_j$, $j \in B$, be the vectors generated by setting $\pi_j^i=v_j^i$, $i \in B$, and calculating the rest of the vector $\underline{\pi}_j$ according the procedure outlined above.

Then each $\underline{\pi}_i$ satisfies an equation of the form

$$\underline{Q}\ \underline{\pi}_i=(\emptyset,...,\emptyset,r^{1,i},...,r^{n_B,i})^T \qquad (C.1)$$

where not all of the $r^{i,j}$'s are zero. Because the process (L(t),N(t)) is positive recurrent, the vector $\underline{\pi}$ is in the subspace spanned by the $\underline{\pi}_i$'s. Let

$$\underline{\pi} = \sum_{i \in B} y^i\ \underline{\pi}_i .$$

If $\underline{Q}\ \underline{\pi}=\emptyset$ then $\underline{y}=(y^1,\ldots,y^{n_p})^T \neq \emptyset$ satisfies the equation $\underline{R}\ \underline{y}=\emptyset$ where $\underline{R}=\{r^{i,j}\}$ and the $r^{i,j}$ are given by equation (C.1). This gives a method of finding $\underline{y}$ and hence $\underline{\pi}$. To simplify the calculation of $\underline{\pi}$ from $\underline{y}$ it is convenient to take $\underline{v}_i=(\emptyset,\ldots,\emptyset,1,\emptyset,\ldots,\emptyset)$ with the one appearing in the ith position. Then $\pi^i=y^i$, $i \in B$.

The choice of the "boundary states" B is influenced by three factors. First, the smaller n_B is the more efficient the solution process will be. Second, the calculation of $\underline{\pi}_i$ from $\underline{v}_i$ should be a simple task. Clearly if we have to solve a large set of linear equations to determine $\underline{\pi}_i$ from $\underline{v}_i$ we might as well try to solve $\underline{Q}\ \underline{\pi}=\emptyset$ directly. Third, one should choose B so that a non-trivial proportion of the probability $\underline{\pi}$ is concentrated on B. To see why we note that $\underline{y}$ is merely the projection of $\underline{\pi}$ down onto the subspace spanned by $\underline{v}_i$, $i \in B$. Because the process $(L(t),N(t))$ is positive recurrent, we know $\underline{\pi}$ cannot be orthogonal to this subspace. However if $\sum_{i\in B}\pi(i)\cong\emptyset$, then from a computational standpoint $\underline{\pi}$ is indistinguishable from a vector that is orthogonal to all of the $\underline{v}_i$. Therefore, the calculation of $\underline{R}$ from $\underline{Q}$ as well as the calculation of $\underline{\pi}$ from $\underline{y}$ will be poorly posed. Thus B must be selected so that $\sum_{i\in B}\pi(i)$ is not negligibly small.

Examination of equation (2.8) shows that there are several choices for the set B. One choice is $B_1=\{(i,N_P-1):L_Q\leq i<N_P\}$. A recursive method based on B_1 was used to do the calculations in Example 5.1.

Another choice for the boundary states is $B_2 = \{(L_Q-1, L_Q-1), (L_Q, i) : L_Q \leq i < N_P\}$. Unless $P\{L_k = L_Q\}$ is very small, this choice of boundary values is suitable for all levels of system utilization. A recursive method based on B_2 was used for the calculations of Example 5.2. It was necessary to use double precision arithmetic in the calculation of $\underline{\pi}$ in order to make $\|\underline{Q}\ \underline{\pi}\|$ as small as 10^{-6}.

A detailed error analysis of this recursive method is beyond the scope of this paper. Empirical evidence, based on solving $\underline{Q}\ \underline{\pi} = 0$ directly and by the recursive method, shows that $\bar{N}$ and $\bar{L}$ values calculated by the recursive method are accurate to between four and five significant digits, provided that $\|\underline{Q}\ \underline{\pi}\|$ was small. These comparisons have only been done for $(N_P - L_Q) \leq 20$. However, as $(N_P - L_Q)$ increases, $\|\underline{Q}\ \underline{\pi}\|$ does not appreciably change. Therefore it seems likely that the recursive method has about the same precision for larger N_P values.

BIBLIOGRAPHY

[1] A. K. Agrawala and R .M. Bryant, Models of memory scheduling, Proceedings of Fifth Symposium on Operating System Principles, Austin, Texas, (1975) 217-222.

[2] Y Bard, An analytic model of CP-67 and VM/370, IBM Research Report G320-2101, Cambridge Scientific Center, Cambridge, Massachusetts, (1974).

[3] T. Betteridge, An analytic storage allocation model, Acta Informatica 3 (1974) 101-122.

[4] R. M. Brown, J. C. Browne, and K. M. Chandy, Memory management and response time, Communications of the ACM 20, 3 (March 1977) 153-165.

[5] R. M. Bryant and A. K. Agrawala, An evaluation of M/M/R queues as finite memory size models of computer systems, Proceedings of the 1977 Sigmetrics/CMG VIII Conference on Computer Performance Modelling, Measurement, and Management, Washington, D. C., (November 28-December 1, 1977).

[6] R. M. Bryant, The storage limited exponential queue and applications to finite memory size models of computer systems, (Ph. D. Thesis, Applied Mathematics, University of Maryland, College Park, May 1978).

[7] R. M. Bryant, Maximum processing rates of memory bound systems, Computer Sciences Department Techical Report, University of Wisconsin-Madison, (August 1979). Submitted to Journal of the ACM.

[8] J. P. Buzen and D. B. Rubin, Effects of compaction on memory utilization in multiprogramming systems, Proceedings of the International IRIA Workshop on Modeling and Evaluation of Computer Architectures and Networks, Rocquencourt, France, (1974) 113-124.

[9] L. Green, A queueing system in which customers require a random number of servers, Operations Research 28, 6 (November-Decmber 1980) 1335-1346.

[10] L. Green, Comparing Operating Characteristics of Queues in Which Customers Require a Random Number of Servers, Management Science 22, 1 (January 1981)