NUMERICAL METHODS FOR MULTISERVER DISCRETE TIME QUEUES WITH BATCH MARKOVIAN ARRIVALS

by

Stanley Chinwan Tang

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Industrial and Systems Engineering

APPROVED:

[illegible]
Dr. John N. Daigle, Co-chairperson

[illegible]
Dr. Hanif D. Sherali, Co-chairperson

John E Kobza
Dr. John E. Kobza

[illegible]
Dr. Richard E. Nance

Subhash C Sarin
Dr. Subhash C. Sarin

January, 1995

Blacksburg, Virginia

NUMERICAL METHODS FOR MULTISERVER DISCRETE TIME QUEUES WITH BATCH MARKOVIAN ARRIVALS

by

Stanley Chinwan Tang

Chairpersons: John N. Daigle and Hanif D. Sherali

Industrial and Systems Engineering

Abstract

Many physical systems are well-modeled by queueing systems in which time is slotted, the distribution of the number of entitites that arrive during a slot is dependent upon the evolution of a discrete time, discrete state Markov chain, and the number of entities that may be served during a slot is limited to some number, say R. Techniques for analyzing systems in this, or closely related, class have appeared in the literature, but distributions have been presented in only rare instances, limited to the case $R = 1$. Yet, distributions are very important, not only in performance evaluation, but in design, especially for sizing buffers in integrated (BISDN) communications systems and intermediate storage space designs in manufacturing systems.

In this dissertation, a numerically stable methodology based on eigenanalysis and probability generating function technique has been developed for producing both occupancy and delay moments and distributions for the equilibrium process described above. Feasibility of the methodology is demonstrated through numerical results for two examples of an important subclass. Special attention is paid to obtaining accurate numerical values; and wherever available, numerical values are compared to those previously obtained in the literature. Furthermore, additional important models amenable to analysis by the same methodology are discussed and numerically feasible approaches for obtaining important performance measures are suggested.

ACKNOWLEDGEMENT

My greatest gratitude to Dr. John Daigle, for his expert advice and guidance, and for believing in my capability as a student and a research associate. Special thanks to Dr. Sherali for picking up the position as my committee co-chair. Thank you to Dr. Kobza, Dr. Nance and Dr. Sarin for serving on my committee. Special thanks to Ms. Lovedia Cole at the Student Record Office of the ISE Department of Viginia Tech.

I am greatly indebted to my parents, and my brothers and sisters for their steadfast spiritual as well as financial support. Thanks to all friends and colleagues at Virginia Tech and elsewhere for brightening up my life and making my work meaningful.

CONTENTS

FIGURES

TABLES

CHAPTER 1
INTRODUCTION

1.1 Statement of Problem

The behavior of many physical systems is well modeled by a queueing system in which time is slotted, the distribution of the number of entitites that arrive during a slot is dependent upon the evolution of a discrete time, discrete state Markov chain, and the number of entities that may be served during a slot is limited to some number. As an example, consider the problem of buffering characters, or packets, which arrive to a statistical multiplexer from independently operating terminals (Chu [1969]). In this case, there is a finite number of individual sources that alternate between active and idle periods (Rickard [1972]). During idle periods, no characters are generated, but during each successive time slot of an active period, a terminal generates one character, which is multiplexed along with the characters generated by other terminals. The number of slots in periods of each type is assumed to have a geometric distribution. Thus, the number of terminals that are active during a slot, and therefore the number of characters that arrive during the slot, is equivalent to the state of a discrete time Markov chain. As a second example, consider the sequence of water inflows into a Moran reservoir (Moran [1954]). Because water that enters the reservoir during a time period may actually come from rainfalls of some previous time periods, it is more realistic to take account of the correlation of the units of water infl ws among different time periods. A discrete-time Markov chain is usually employed to take this type of correlation into effect (for example, Lloyd [1963]).

The type of arrival process mentioned above is known as the batch Markovian arrival process (BMAP). Coined in Lucantoni [1991], BMAP includes many familiar arrival process as subclasses, for example, the Markovian arrival process (MAP) (Lucantoni et al [1990]) which includes the Poisson process, the PH-renewal process, Markov-modulated Poisson process (MMPP), alternating PH-renewal process, a sequence of PH interarrival times selected via a Markov chain, a superposition of PH-renewal process, and the superposition of independent MAP's, a MAP with i.i.d. batch arrivals, and a batch Poisson process with correlated batch arrivals. In particular, BMAP is statistically equivalent to a special class of point process known as the versatile Markovian point process (VMPP).

The VMPP was introduced by Neuts [1979]. Using probabilistic arguments, Neuts and his colleagues have been able to develop a system of algorithmic techniques for analyzing and obtaining useful performance measures for the type of queueing processes mentioned above. The method initially championed by Neuts but contributed to by numerous other researchers has become known as the matrix analytical method (Neuts [1989]). The matrix analytical method has long been recognized as a powerful tool not only in obtaining important measures, but also in gaining informative insight into the probabilistic structure of many queueing models. The disadvantages of the matrix analytical method, however, are its incomprehensibility to many a nonpractitioner of probability theory, and its unpredictable and sometimes excessive computation time in finding some of the critical parameters for obtaining important performance measures (for the second argument, see Daigle [1991]). In fact, among the very few numerical examples that use the matrix analytical method as the analytical tool in the literature, most of them are very simple and are usually limited to some special cases of the general model. Moments are presented most of the time; distributions are presented only in rare instances and are usually limited to cases of single server. Yet distributions are very important, not only in performance evaluation, but also in design, especially in sizing buffers, for many queueing systems.

The purpose of our present investigation is to develop an alternative algorithmic approach to finding important distributions as well as moments for the type of queueing models introduced above. Our objective is to develop an algorithm which is efficient in terms of computation speed and effective in terms of stability and accuracy. Our approach is based on classical eigenanalysis and partial fraction expansion, and can be applied to the analysis of a large number of queueing models that arise in many physical environments such as telecommunications, storage systems and manufacturing systems. In the remainder of this chapter, we will describe briefly the development of the matrix analytical method, discuss our transform/eigenanalysis approach, discuss the scope of research, and, finally, outline the remainder of this dissertation.

1.2 The Matrix Analytical Method

Motivated by the immense computational power afforded by modern computing technology, researchers have been developing algorithm-oriented methodologies for analyzing the behavior of some large and complicated queueing models, an example being the matrix analytical method. In Winsten [1959], the occurrence of geometric terms in the stationary queue length distributions of some queueing problems, namely the $G/M/1$ queues, was given renewed recognition and new derivations were provided. Evans [1965] took advantage of this special property by reducing the problem of finding the stationary queue length distribution into one of finding the term ratio, and an iterative scheme (in this case, the successive substitution method) was developed for this purpose. Inspired by Vic Wallace, a student of Evans, Neuts [1981] generalized the results obtained by Evans into an algorithmic method for the analysis of a new class of queueing models which we hencefore call queueing models of the $G/M/1$ type, and the new algorithmic method was known as the matrix geometric method.

A queueing model of the $G/M/1$ type differs from an ordinary $G/M/1$ queue in the fact that the distribution of the service time in the $G/M/1$ paradigm is dependent upon the evolution of a Markovian phase process. This phase process was formalized and developed into the renewal process of phase type (PH-renewal process) in Neuts [1978]. Using the PH-renewal process as substratum, Neuts [1979] introduced a new class of point process known as the versatile Markovian point process (VMPP). The VMPP has three types of events (Neuts [1989]):

a. The Markov-modulated Poisson arrivals, which are of rate λ_j during sojourns in the phase j,

b. The transitions from a phase j to a phase j', $j' \neq j$, which do not occur at a renewal epoch of the PH-renewal process, and

c. The transitions from a phase j to a phase j' at a renewal epoch in the PH-renewal process. Such transitions involve an instantaneous absorption and restarting of the Markov process.

In Ramaswami [1980], the VMPP was renamed the N-process and was the main feature in the analysis of another new class of queueing models called the $N/G/1$ queue. In a $N/G/1$ queue, as indicated by the special characteristics of the VMPP, the arrival process is modulated by an underlying phase process. The $N/G/1$ queue has a transition matrix of the $M/G/1$ type (see, for example, Daigle [1991]).

Central to the solution of the stationary queue length distributions for the $M/G/1$ and $G/M/1$ paradigms is the computation of the entrywise minimal nonnegative solution of a nonlinear matrix equation of the form $G = \sum A_n G^n$ or $R = \sum R^n A_n$. The G matrix, which appears in the $M/G/1$ paradigm, is a matrix in which the entries are transition probabilities of a Markov chain. The R matrix, which appears in the $G/M/1$ paradigm, is known as the rate matrix and its entries represent the expected number of visits to states of the immediately higher level before the first return to the current level (Neuts[1981]). A duality theorem which establishes the relationship between the G matrix and the R matrix was presented and discussed in detail in Ramaswami [1990]. and a complete probabilistic interpretation of the theorem was given by Asmussen and Ramaswami [1990]. In practice, computation of the G matrix or the R matrix is difficult. It is consistently cited in the literature that computation times of these matrices are unpredictable and possibly excessive (Daigle [1991]). In one special study, Cao and Stewart [1987] concluded that it was more computationally efficient to solve the equilibrium balance equations directly than to compute the rate matrix.

Five iterative algorithms for the computation of the G or R matrix were reviewed in Gün [1989]. and an extension was proposed to improve their efficiency. The five iterative schemes are as follows:

(i) Successive Substitution: direct iteration on the original matrix polynomial, starting with $G = 0$ or $R = 0$.

(ii) Modified Successive Substitution: similar to successive substitution, except that the coefficient matrix of the first power of G or R is grouped and inverted before iteration begins,

(iii) Newton-Kantorovich Method: an iterative scheme based on the Gateau derivative of the matrix polynomial.

(iv) Modified Newton-Kantorovich Method: a modified scheme proposed by Ramaswami [1988] to improve efficiency in the original Newton-Kantorovich method by avoiding the solution of a linear system, and

(v) Uniformization Algorithm: based on the "randomization" technique originally due to Jensen [1953]. this method develops a numerical stable way of computing the G or R matrix without having to compute the matrices A_n (see Lucantoni and Ramaswami [1985]).

The extension suggested by Gün is based on the fact that the matrix G is stochastic and therefore that extention does not apply to the computation of the R matrix. Gün suggested that, in each iteration, the most recent results could be used to obtain an approximation to the stochastic matrix by means of linear extrapolation. It was reported that savings of up to 50-70% in the number of iterations and CPU times could be obtained using the extrapolation technique.

Once the R matrix is obtained. computation of the stationary queue length distribution of the queueing system of the $G/M/1$ type is straight forward because of the matrix-geometric property of the probability vectors. Computation of the stationary queue length distribution of the queueing system pf the $G/M/1$ type requires a more elaborate iterative scheme. A stable recursive method which avoids subtractions in the iterations was reported in Ramaswami [1988]. Recently, Sengupta [1989] discovered that stationary joint probability distribution of queue length and phase of the queueing system of the $G/M/1$ type had a matrix-exponential form, i.e. $\phi(n) = \phi(0)e^{Rn}$. Extension of the matrix-exponential result to the $M/G/1$ paradigm was given by Lucantoni [1991]. A great deal of simplification in the analysis of subclasses of the $G/M/1$ and $M/G/1$ paradigms has been achieved using these new discoveries. However, no numerical study on the new method has been reported to date.

1.3 The Transform/Eigenanlysis Approach

The transform method has always been one major tool for the analysis of various queueing systems. In this section, we shall review some literatures that use transform techniques for studying multiserver queues and queues with correlated batch arrivals. Then we shall discuss our transform/eigenanalysis approach and present several relevant references on numerical eigenvalue-eigenvector analysis methods.

Early studies of multiserver Markovian queues were due to Erlang, and Molina obtained some approximate results on multiserver queues with constant holding time (see Crommelin [1932]). The most extensive and lucid results on $M/D/c$ queues, however, were obtained by C. D. Crommelin. In Crommelin [1932], formulae for distributions of queue length and virtual waiting time were derived. In a follow-up paper (Crommelin [1934]), these results were compared with those obtained earlier by Pollaczek. In addition, discussions were made on two of Pollacsek's formulae, namely, probability of no delay and average delay, which were not obtained previously by Crommelin. A detailed study on $G/G/s$ systems came much later and was provided in Kiefer and Wolfowitz [1955]. It was found in that paper that the waiting time distribution approaches an integral function which is satisfied by a unique distribution function when the traffic intensity is less than 1.

An $M/G/s$ system was studied in Bailey [1954]. Bailey used the imbedded Markov chain method introduced by Kendall [1951] to obtain the equilibrium distribution of queue length. Expressions for the mean and variance of queue length and the mean waiting time were also obtained. Tables of delay distributions for the $M/D/c$ queue were obtained by Kühn [1976], who computed the probabilities by truncation of infinite stochastic matrices.

In Bodreau et al. [1962], the queue length distribution of a multiserver, constant service-time queueing system was studied using the probability generating function approach. In that paper, the numbers of arrivals in a time slot are independent, identically distributed random variables. Studies on queues with correlated batch arrivals were initiated by research on the stochastic theory of reservoirs (or dams). In Lloyd [1963], the sequence of water infl ws into a finite reservoir was approximated by a Markov chain. In a series of papers (Odoom and Lloyd [1965], Ali Khan and Gani [1968], Ali Khan [1970], Herbert [1972, 75]) that followed, researchers dabbled in a wide range of related topics, which included modeling the sequence of correlated infl ws by a moving average, studies on stationary and transient behaviors, and studies on finite and infinite cases. A more general moving average model for correlated inputs was presented in Gopinath and Morrison [1977]. In Morris [1981], analysis of the performance of a class of packet switching communications networks which led to consideration of a queueing model with input given by a function of a Markov chain was presented in detail. There are numerous published papers in the communications area that deal with correlated batch inputs. We defer discussion of these papers to a later chapter.

The solution methodology under consideration in this dissertation is based upon classical eigenanalysis and partial fraction expansion. In this methodology, the quantity of interest, namely, the queue length distribution, is modeled in a transformed compact form using the probability generating function technique, an expression in the form of an integral-difference equation is formed, expanded in a spectral series, and finally inverse transformed to obtained the distribution. This approach to the analysis of a phase-dependent queue was first carried out in Daigle and Lucantoni [1991]. In that paper, the transition matrix of the queueing model under investigation is of the $G/M/1$ type. Joint and marginal queue length distributions, and virtual and stochastic equilibrium waiting time distributions were obtained. The authors also made comparison of their transform/eigenanalysis approach with the matrix-geometric approach of Neuts. It was concluded that the transform/eigenanalysis approach was "blindingly fast" in comparison to the matrix analytical approach.

One main step in the computation scheme proposed in this dissertation requires the inversion of a polynomial matrix of the form

$$\mathcal{A}(z) = \sum_{i=0}^{l} \mathcal{A}_i z^i,$$

where $\mathcal{A}_i, i = 0, 1, \cdots, l$ are MxM square matrices. While each element of the original matrix is a polynomial, each element of the inverted matrix is a ratio of two polynomials, with the denominator being given by the same determinant of the original matrix. The zeros of this denominator polynomial are the null values of the matrix $\mathcal{A}(z)$, where the null values and vectors of a polynomial matrix are the set of values and vectors, $\{z_i\}$ and $\{X_{z_i}\}$, such that $\mathcal{A}(z_i)X_{z_i} = 0$ with X_{z_i} nontrivial.

Several techniques for obtaining null values and null vectors of polynomial matrices have been developed over the last 40 years. In Tarnove [1958], an iterative root-finding technique is developed, whereas in Guderley [1959] and Dimsdale [1960] the nonlinear problem is converted into a first-degree linear problem of solving a matrix equation of the form

$$AX = zBX,$$

where A and B are $l$Mx$l$M matrices, z is a scalar and X is a nontrivial vector. Problems of this type are known as generalized eigenvalue-eigenvector problems (see, for example, Peters and Wilkinson [1970a]). If either A or B is nonsingular, the problem can be further reduced to a standard eigenvalue-eigenvector problem. For cases where all eigenvectors are distinct, canned eigenvalue-eigenvector algorithms are available; for example, the QZ algorithm (Moler and Stewart [1973] and Ward [1973]) for the generalized eigenvalue-eigenvector problem, and the QR method (Francis [1961, 62], and Peters and Wilkinson [1970b]) for the standard eigenvalue-eigenvector problem. For the computation of eigensystems with repeated eigenvalues, an algorithm developed by Kagström and Ruhe [1980] is suggested.

Hitherto we have mentioned methods for obtaining the zeros of the denominator polynomial of an inverted polynomial matrix. Once the zeros are obtained, the denominator polynomial can then be factorized and the inverted matrix expressed as the sum of partial fractions. For non-algorithmic treatment of polynomial matrix inversion, the readers are referred to Lancaster [1966, 75, 77a, 77b], Gohberg, Lancaster and Rodman [1982], and Markus [1988].

1.4 Scope of Research

In this dissertation, we consider a discrete time, multiserver queue with batch Markovian arrivals. Our solution methodology is a transform/eigenanalysis approach which is based upon classical eigenanalysis and partial fraction expansion. Since the type of queueing models considered herein has great potential for applications in the analysis of packet switching communications systems, we will study two communications models in detail and present some numerical examples in order to illustrate the viability and applicability of our method. In particular, we will obtain the stationary distributions and the means of queue length and actual waiting time at departure points. Special attention will be given to the accuracy of the results. In cases where data is available, we will compare our results with those published in the literature. Finally, we will discuss approaches to the solutions of some additional models which have more general sturctures than the two communications models have.

1.5 Organization of Dissertation

The organization of the remainder of this dissertation is now described. In Chapter 2, we will present a brief survey on Queueing Theory. Chapter 3 is a brief chronicle on the developments in Telecommunications. In Chapter 4, we will illustrate our methodology through detailed ananlysis of a communications model whose queueing process has a one-to-one correspondence between the phase of the arrival process and the number of entities that arrive during a time slot. In Chapter 5, we will extend our approach in Chapter 4 to the analysis of a more complicated communications model. Specifically, while in the model of Chapter 4 each source alternates between active and idle periods, and each source generates one packet each time slot during an active period, in the model of Chapter 5 each source generates one packet every, say, T slots of time during an active period. Additional models which have some additonal features to those in the models of Chapters 4 and 5 will be presented in Chapter 6. In Chapter 7, we conclude our discussions.

CHAPTER 2
A SURVEY ON QUEUEING THEORY

2.1 Introduction

Since its conception about a century ago, queueing theory has developed into a very sophisticated analytical tool which can be readily applied for the study of many service systems. Among them, telecommunications still commands a major percentage of articles contributing to the theory. Other common application areas of queueing theory include transportation systems, computer systems, manufacturing systems and storage systems. Queueing theory deals with the analysis, design and operations of such service systems characterized by their input processes, service mechanisms and queue disciplines. In this chapter, we will briefly describe some of the major developments in queueing theory from the beginning of this century when the theory evolved from studies on telephone traffic congestion through the present day. In discussing various classes of queueing systems, we will use the notation developed by Kendall in 1953.

2.2 Early Developments

The earliest work in queueing theory is usually credited to G. T. Blood who in 1898 attemped to mathematically describe telephone traffic flctuation in an unpublished memorandum. Probabilistic studies on telephone traffic were further pursued by M. C. Rorty in 1903, and Johannsen and Grinsted in 1907. The foundation of queueing theory was laid in 1909 when A. K. Erlang published his Theory of Probability and Telephone Conversation. Erlang introduced the concept of statistical equilibrium, developed the balance equation approach, formulated the famous loss and delay equations, and first attempted at optimizing a queueing system.

Congestion Theory prior to 1920 was summarized by G. F. O'Dell who in 1927 published his classic paper on grading. In the same year, Molina published his Applications of the Theory of Probability to Telephone Trunking Problems. The first comprehensive treatment on congestion problems was provided by T. C. Fry in his Probability and Its Engineering Uses in 1928. In the 30's, many significant developments occured. Notable among contributors of the 30's include Pollaczek, Kolmogorov, Khintchine, Palm, Crommelin, Feller, Vaudot, Lubbeger, Langer, Kosten and Wilkinson.

2.3 Basic Concepts and Techniques

The Poisson distribution plays a special role in queueing theory. Originally obtained by S. D. Poisson in 1827, it was rediscovered independently by E. C. Molina in 1908 and by W. H. Grinsted in 1909. Poissonian input was called "pure chance" input by O'Dell (1927), T. C. Fry (1928) and C. Palm (1937).

Statistical equilibrium was developed out of the concept of ergodicity in statistics. Erlang was first in applying statistical equilibrium in queueing theory. In 1955, Kiefer and Wolfowitz showed that the waiting time distribution of a $G/G/s$ queue did not exist in the steady state if the traffic intensity was not less than unity.

An important result in queueing theory is known by the acronym PASTA which stands for Poisson Arrivals See Time Average. This result states that the fraction of arrivals that sees the process in some state is equal to the fraction of time the process is in that state. This result was proved by Wolff in 1982. The reverse problem of finding under what conditions must the arrival process be Poisson given that arrivals do see time average was explored by Konig, Miyazawa, and Schmidt (1983) and Green and Melamed (1990). Melamed and Whitt (1990) droped the Poissonian assumption and investigated ASTA or Arrivals See Time Average. They also discussed Departures See Time Average.

Another useful result was obtained by Little (1961) who established a general relationship between mean queue length and mean waiting time. A comprehensive survey on Little's result was given by Ramalhoto, Amaral and Cochito (1983). Keilson and Servi (1988) enlarged on the "distributional form of Little's law" and established a relationship between the distribution of number of customers in system and the distribution of time customers spend in system.

Early approaches to queueing analysis were highly analytical and relied heavily on complex variables techniques and transformation. F. Pollaczek initiated the analytical method in 1934. His approach was further pursued by J. W. Cohen (1969, 82) and J. H. A. deSmit (1971, 75). Other methods which rely on the use of transforms include the integro-differential equation approach of Takács (1955) and Beneš (1956), the supplementary variable techniques of Cox (1955), Kielson and Kooharian (1960) and the renewal theoretic approach of Gaver (1959), Takács (1962) and Bhat (1964, 68).

One major development in queueing theory was the introduction of regeneration points for the description of process of the renewal type by Palm (1943).

Kendall (1951) extended the regeneration point technique and developed the imbedded Markov chain method. With the imbedded Markov chain structure, a wide variety of queueing systems can be analyzed using the semi-Markov process technique. This later approach was pursued by Fabens (1961) and Neuts (1966, 71). In 1953, Jensen introduced the concept of uniformization.

There are many other useful techniques. Another first was that of Champernowne (1956) who considered the $M/M/1$ queue as a random walk. Takàs (1965, 67) developed the combinatorial method using the classical ballot theorem. Runnenburg(1965) used the method of collective marks to derive generating functions by means of probabilistic arguments. Spitzer (1957, 60) and Prabhu (1965) introduced the Weiner-Hopf technique. Keilson (1965) studied the Green's function method.

The most popular process in queueing analysis is the birth-death process. Kendall and Bartlett studied the birth process independently in 1949. Further results in birth-death process were obtained by McGregor (1957). Multidimensional birth-death process was studied extensively by Cooper (1972).

In order to render analyses of complex queueing models more tractable, some approximated methods were developed. Important works on heavy traffic limit theorem was given by Kendall (1957), Kingman (1961, 62) and others. Marshall (1966) and Marchal (1978) gave inequalities for some queueing models. Gaver (1966) studied diffusion approximation, and Newell (1971) gave flid approximation results.

Numerical analysis of queueing models is recently gaining more and more attention. Numerical inversions of transforms were studied by Gaver (1960) and Bellman et al (1966). In the past two decades, algorithmic and numerical methods for queueing analysis were developed by Bhat (1971, 73), Bagchi and Templeton (1972), Wallace (1973), Brandwajn (1977), Neuts (1977, 81, 89), Daigle (1986, 89, 90) and others.

Simulation is used either as a supplement to theoretical results or as a tool for the analysis of analytically intractable models. In 1955, G. Neovius used pseudo-random numbers generated by a digital computer to evaluate congestion in telephone systems. Specific problems arose in simulating queueing systems were investigated by Motchell (1973). Fishman (1974) and others.

2.4 Queueing Models

The simplest stochastic model in queueing analysis is the $M/M/1$ queue. The equilibrium behavior of $M/M/1$ queue can be derived from Erlang's study on $M/M/s$ queue. Transition probabilities were obtained by Clarke (1956), Bailey (1954), Ledermann and Reuter (1954), Champernowne (1956) and Conolly (1958). Autocorrelation functions of queue length were obtained by Morse (1955). Transient behavior and waiting time distribution were studied by Palm (1943), Kendal (1951), Bailey (1956), and Prabhu (1969). Prabhu (1960) studied the busy period distribution. Burke (1956) and Reich (1957) showed that the departure process of this queue is also Poisson.

Initial studies on $M/G/1$ queue is due to Pollaczek (1930) and Khintchine (1932). Waiting time distribution was studied by Takás (1955), Prabhu (1965) and Bhat (1965). Extensive studies on queue length distribution were done by Takás, Cox (1955), Gaver (1959), and Keilson (1960). Kendall's imbedded Markov chain method (1951, 53) gave the limiting queue length distribution of this model as well as that of the $GI/M/1$ queue.

The $GI/M/1$ queue is the dual of the $M/G/1$ queue. Many results developed for the $M/G/1$ queue can be used on $GI/M/1$ queue. Smith (1953) discovered that the limiting waiting time distribution of $GI/M/1$ queue is always exponential.

Neuts (1981, 89) and others extended the concepts of $M/G/1$ and $GI/M/1$ queues into 2-dimensional $M/G/1$ and $G/M/1$ paradigms.

Pollaczek was first in considering $GI/G/1$ queue. Lindley (1952) derived an integral equation for waiting time distribution. Spitzer (1956) expressed the waiting time as the maximum of a partial sum process. Kingman (1962), Rice (1962) and Prabhu (1965) studied the busy period,

Studies on multiserver Markovian queue were due to Erlang. $M/G/s$ queue was studied by Crommelin (1932, 34). $GI/G/s$ queue was studied by Kiefer and Wolfowitz (1955), and Kingman (1966). Tables of performance measures for $GI/G/s$ queue were provided by Seelem, Tijms and van Hoorn (1985). Whitt (1985) and Tijm (1986) gave some approximation results. A multiserver queueing model with versatile Markovian point arrivals and arbitrary service time distrubtion was studied by Neuts (1981).

The first transient solution for a birth-death model with constant coefficients was obtained by Clarke (1952). Bailey (1956) and Lederman and Reuter (1956) solved some time dependent problems using the method of generating function and spectral theory respectively. Significant results for transient behavior of queues were due to Karlin and McGregor (1955), Cox (1955), Clarke (1956), Luchak (1956), Conolly (1958), Gaver (1959), Keilson (1960), Saaty (1960), Takács (1961, 62), Prabhu (1963, 65), and Luchak (1964, 68). Recently, Boxma (1984), Towsley (1987), Syski (1988), Whitt (1987, 88) and others had made significant contributions.

Mellor (1942) first attempted at getting the delay distribution for random selection for service. The waiting time distribution of $M/M/1$ queue with random selection for service was studied by Palm (1957) and Riordan (1962). Kingman (1962) studied the limiting behavior of $M/G/1$ queue with random selection for service.

Cobham published the first paper on priority queue in 1954. Extensive works on priority queue were done by Phipps (1956), Barry (1956), Kesten and Runnenburg (1957), Morse (1958), White and Christies (1958), Jackson (1959), Miller (1960), Fuhrman (1984) and Cooper (1986).

Queues with customer impatient behaviors such as balking, reneging and jockeying were studied by Haight (1957), Barrer (1957), Finch (1959), Ancker and Gafarian (1963), Rao (1965) and others.

Queues with customers arrive in gropus and/or served in groups are called bulk queues. Bailey (1954) first investigated queues with batch service. Miller (1959) first studied queueus with batch arrivals. Significant works on bulk queues were done by Keilson (1962), Kinney (1962), Bhat (1964), Cohen (1969), Neuts (1971, 81, 89) and others.

Extensive investigations into queueing systems with vacations were done by Gaver (1962), Heyman (1968, 69, 77), Doshi (1983, 85), Shanthikumar (1981, 82, 83), Neuts (1979, 84), Fuhrmann (1981, 85, 86), Keilson (1962, 86), Levy (1976, 86), Cooper (1969, 90) and others.

In 1954, O'Brien studied the problem of two queues in series. Studies on queueing networks were made by Hunt (1956), Marshall and Reich (1956), Jackson (1954, 57, 63), Klinrock (1964), Gordon and Newell (1967), Whittle (1967, 68), Burke (1956, 76), Daduna (1982), Kelly (1979, 83) and Gordon (1990). The time-reversal approach frequently used in queueing network analysis was due to Kolmogorov (1936).

In 1937, Palm studied holding time given by the sum of exponential terms. An extension of this type of Erlangian distribution was done by Luchak (1956) who showed the wider applicability of the weighted-sum-Erlangian distribution. Schessberger (1973) proved that any nonnegative random variable can be represented by a compound sum of independent, identical exponential variables. A mixture of exponentials is called the hyperexponential distribution. Hyperexponential distribution was studied by Morse (1958) and Botta, Harris and Marchal (1987).

Another general distribution which has many simple distributions as subsets is the class of PH-distri-bution investigated by Neuts (1978), and Takahasi and Takami

(1976). The PH-distribtion provides the substratum for Neuts' versatile Markovian arrival process and Lucantoni's batch Markovian arrival process.

2.5 Applications

Early studies on queueing theory such as those by Blood, Rorty, Johnannson, Grinsted and Erlang were centered around applications in solving telephone traffic congestion problems. In 1952, Bailey published a study on appointment systems in hospitals. Edie looked into traffic delays at toll booths in 1956.

In 1957, Alison Doig published a bibliography on queueing theory, in which papers on eight application categories and two theoretical categories were included. The eight application categories are: storage problems such as the optimal size of dams; problems relating to flows through a network; inventory control and production scheduling problems; problems arising in servicing automatic machines; counter problems; road traffic and related topics; problems in telephone traffic; and miscellaneous topics which include the scheduling of air traffic and the design of appointment systems.

In 1958, Morse's book with applications of queueing results to realistic problems was published. In 1962, Bather attempted to extend some methods developed for inventory theory to the solution of queueing problems.

In 1963, Hillier introduced standard optimization techniques to queueing problems in his paper on economic models for industrial waiting line problems. In the same year, Brosh and Naor considered attaching priorities to individual arrival streams when cost functions are assigned to the expected queue size of these streams. Additional optimization problems in priority queues were investigated by Etschmaier (1966) and Jaiswal (1968).

Some of the design and control problems studied under the context of Markov decision process were carried out by Miller (1967), Shaler (1968), Kumin (1968) and Evan (1968). Today, papers deals with queueing systems for specific applications can be found in various journals dedicated specificlly to those applications.

2.6 Conclusions

We have presented in this chapter an overview of some of the major results in queueing theory as the theory has developed over the past ninety some years. Emphasis has been placed on providing a historical perspective on the subject. For extensive bibliographies and detailed discussions on major developments and results in queueing theory, the reader is referred to the survey papers by Bhat [1969, 78], and Cooper [1990], and the textbooks by Saaty [1961], Cohen [1982] and Syski [1986]. An excellent survey paper on queueing systems with vacations is given by Doshi [1986]. Walrand [1988] provides a comprehensive discussion on queueing networks. Kleinrock [1976] specifically deals with queueing systems in computer applications. Daigle [1991] treats extensively queueing applications in computer communications, and Viswanadham and Naraham [1992] in performance modeling of automated manufacturing systems.

CHAPTER 3
DEVELOPMENTS IN TELECOMMUNICATIONS

3.1 Introduction

In ancient times, men transmitted messages by fire and smoke signals, drum beats, runners, pigeons, boats, postriders and sentinels who relayed shouted signals. In the last decades of the eighteenth century, the future James II of England designed a set of flag-signals which was later systematized by Kempenfelt and Earl Howe. The invention of telescope greatly improved the possibilities of such visual systems, but telegraphy, in the modern sense of the term, was not developed until the 1790's. In 1794, Claude Chappe developed a visual system which he called an ocular, or semaphore, telegraph. Claude Chappe's system consisted of a line of towers 6 to 10 miles apart. On top of each tower was a semaphore which could be bent into different shapes. Claude Chappe's system was expensive to operate because it required men to be stationed at each tower. Dicoveries in electricity during the eighteenth century quickly changed the way telecommunications was defined. We will present in the following a brief chronicle on major developments in modern telecommunications technology. We will trace through early experiments on conduction of electricity, the inventions of teletypewriters and telephones, the invention of integrated circuits and digital computers, and finally the development of packet switching and digital computer communications systems.

3.2 Early Developments in Electric Telegraphy

Static electricity was known to the anciect Greek as early as 600 B.C. During the reign of Queen Elizabeth I of England, William Gilbert wrote a comprehensive treatise on magnetism which he called De Magnete. In 1660 von Guericke invented a frictional machine to generate electricity. Shortly afterward, Francis Hauksbee demonstrated that charged bodies might repel as well as attract each other. In 1728 Grey and Wheeler showed that electricity could be conducted to a great distance. In 1729, Stephen Gray distinguished conductors and non-conductors. In the folowing year, Charles Du Fay discovered that electricity could be of two kinds: positive and negative. In 1745, Cunens discovered the Leyden jar, a device which stored electric charges. In 1746, Winkler discharged a Leyden jar through a wire of considerable length. In 1747, Watson extended the experiments over a distance of 4 miles. In 1748. Franklin identified lightning as an electric discharge. In 1754, John Canton used pith balls to measure electric charge. In 1774, George LeSage tested a short-distance telegraph systems in which signals were detected by the deflection of 24 pith balls. In 1784, Lomond sent signals to a neighboring room using a pith ball electrometer. There were other experiments conducted by Reiser, and Cavalo. In 1787, the gold-leaf electroscope was invented by Bennet. In 1798, Betancourt established a 26 mile telegraph between Madrid and Aranguez.

In 1786, Luigi Galvani discovered the presence of electricity in frog legs. The first battery was invented by Volta in 1800. Electrolysis was discovered by Humphrey Davy, who also noted that a brilliant light was emitted between two pieces of carbon when they were struck with a spark. In 1809, Söemering constructed the first galvanic telegraph at Munich. There were other proposals by John Coxe, and Francis Ronalds.

In 1819, Oerstead discovered electro-magnetism. In 1920, Ampere suggested the construction of an electro-magnetic telegraph. The first electro-magnet was constructed by Sturgeon in 1825. In 1827, Georg Ohm discovered the famed Ohm's Law. In 1831, Joseph Henry discovered a method of forming magnets of intensity and of quantity. In the same year, Michael Faraday demonstrated electro-magnetic induction. The next year, Baron Schilling contrived a deflective magnetic telegraph. In 1833, Gauss and Weber constructed a simplified electro-magnetic telegraph. June 1837, the deflective Electro-Magnetic Telegraph of Cook and Wheatstone was patented. July 1837, Stienhiel constructed his Registering Electro-Magnetic Telegraph betwen Munich and Bogenhausen. October 1837, Samuel F. B. Morse entered his first caveat for an "American Electro-Magnetic Telegraph". His invention was patented in France in 1838 and in the U.S. in 1840. Edward Davy patented a chemical telegraph, and Alexander Bain obtained a patent for his Electro-Magnetic clock in 1838. In 1844, Morse completed a line between Baltimore and Washington, D.C., using Morse code to transmit messages. Cooke and Wheatstone formed the Electric Telegraph Company in 1846. Bain obtained the English patent for his improved Electo-Chemical Telegraph in 1846 and the U.S. patent in 1849. In 1848, Royal E. House obtained his patent for a Printing Electric Telegraph. In 1848, Zook and Barnes invented a modification of the Electro-Magnetic Telegraph. In 1849, Horn invented his Igniting Telegraph. There were also works by Johnson, and Daniel Davis.

3.3 Commercialization of Electric Telegraph and Telex

In 1850, the first submarine cable for telegraphy was laid under the English Channel. In 1851, a greatly improved cable was opened for service in November. In 1856, the Atlantic Telegraph Company was organized and Western Union was formed. The first successful Atlantic cable began operation in 1866. In 1871, Emile Baudot invented a time multiplexing system. The basic design for the modern teletypewriter was invented by Baudot in 1874. In 1878, the first public exchange in the U.S. was opened in New Haven, Connecticut. In 1907, the Morkrum Company was formed. Morton and Krum designed the first practical teletypewriter in 1920. In 1925, the Morkrum Company became the Teletype Coorporation. In 1927, switched public teletypewriter services, known as Telex, were introduced in Europe. Later, telex service was provided in the U.S. by Western Union. In 1928, E. E. Kleinschmidt invented the modern teletypewriter. In 1931, AT & T began a manually switched teletypewriter service similar to Telex, calling it Teletypewriter Exchange Service (TWX). TWX was merged into the telephone network using a device called a modem in 1962.

3.4 Telephone

In 1837, Charles Page observed that pieces of iron gave out musical notes when they were subjected to rapidly changing magnetic fields. In 1860, Phillipp Reis constructed an apparatus by which a melody could be transmitted electrically to a distance. In 1875, Elisha Gray developed an instrument very similar to Reis' and filed a U.S. patent in 1876. In 1876, Bell successfully transmitted a sound through a telephone and filed a patent application for "improvement in telegraph." January 1877, Bell filed a patent on an electro-magnetic telephone. It was in the same year the first commercial telephone was constructed. In July, the Bell Telephone Company was incorporated. In 1880, Bell Telephone Company was reorganized as the American Bell Company. In 1885, the American Telephone and Telegraph Company was incorporated. In 1887, Heaviside predicted that the addition of inductance to telephone lines could help compensate for attenuation and distortion. In 1888, AT & T purchased the Western Electric Company. Automatic switching was introduced by Almon Strowger in 1890. In 1907, DeForest invented the triode vacuum tube. A telephone repeater with a vacuum tube amplifier became availbale in 1912. In 1914, carrier systems using a frequency translation technique were used to carry a voice signal at a frequency other than its natural one. Telephone repeaters were tested across the width of the U.S. in 1914, and transcontinental service was inaugurated in 1915. In 1920, the Bell System's C-Carrier system added three voice channels to an open-wire voice line. In 1925, the Bell Telephone Laboratories were formed.

3.5 Wireless Communications

The existence of electro-magnetic waves was discovered by James Clark Maxwell in the 1860's. In 1887, Heinrich Hertz succeeded in producing electro-magnetic waves experimentally. Oliver Lodge first demonstrated "wireless" telegraph in 1894. In 1898, Lodge invented the selective tuner, and John A. Fleming invented the diode vacuum tube. In 1901, Guglielmo Marconi made the first trans-Atlantic transmission using Morse code. Reginald first produced a continuous wave with an alternator in 1903 and later invented the heterodyne circuit for receivers. In 1914, Edwin Howard Armstrong invented the feedback circuit. In 1918, Armstrong invented the super-heterodyne circuit which is basic for modern radio and radar receptors. Commercial broadcasting began in the U.S. in 1920. In 1921, the Detroit Police Department first used land mobile radio. In 1926, J. L. Baird demonstrated the first practical television system. In 1932, television broadcast began in London. New York Police Department used land mobile radio in the same year. In 1941, television broadcast began in the U.S. In 1947, mobile communications in U.K. began. In 1948, commercial long distance microwave telephone and television transmission began between New York and Boston. In 1951, coast-to-coast U.S. TV service via microwave began. The first passive communications satellite was launched in 1960. The first communications satellite with an active repeater known as Telestar was launched two years later. In 1963, the first geostationary communications satellite Syncom II was launched. In 1969, INTELSAT 3F4 was launched, completing the global reach of INTELSAT's network. CB radio became popular in the U.S. in the 1970's. In 1981, The Nordic Mobile Telephone system was launched in Sweden. Today, more and more new mobile telephone systems are being established around the world.

3.6 Digital Computer Communications

A mechanical calculator using binary components called Z1 was constructed by Konrad Zuse in Germany in 1931. In 1936, Alan M. Turing published a paper called "Can a Machine Think!" In late 1930's, John V. Atanasoff and Clifford Berry built the first electronic digital computer. In 1939, George R. Stibitz, S. B. Williams and others completed their "complex computer". In 1944, a report by John von Neuman, Herman H. Goldstein and Arthur W. Burks first described the design of an electronic stored-program digital computer. In 1945, the Whirlwind computer was constructed at MIT under the direction of Jay W. Forrester. In 1946, the valve-based Electronic Numerator, Integrator and Computer (ENIAC) was built at the University of Pennsylvania. In the same year, Alan Turing presented to U.K.'s National Physical Laboratory a detailed design for a stored-program electronic digital computer. In 1948, John Bardeen, Walter Brattain and William Shockey published the results of their work on solid-state electronic devices. In the same year, Claude Shannon developed information theory. In 1949, Forrester invented magnetic-core storage for the Whirlwind. In 1950, the Electronic Discrete Variable Automatic Computer (EDVAC) was built. The Universal Automatic Computer (UNIVAC) was built for use by the U.S. Census Bureau in 1951. In the 1950's, medium to large-scale computer systems known as mainframes were introduced for business use by IBM, UNIVAC and others. In 1954, John Bachus of IBM published the first version of FORTRAN. In 1957, the first patent for a micro-electronic (or integrated) circuit was granted to J. S. Kilby of the U.S. In 1959, a standardized high-level language known as COBOL was developed for business use. In the same year, Christopher Strachey in the U.K. proposed the idea of a time-sharing mainframe computer.

In the year of 1960, the RS-232 physical layer interface standard was published, laser was invented by T. H. Maiman, and the first time-sharing computer operating system was introduced by Corbato and his colleagues. In 1961, MIT demonstrated a model time-sharing system. In 1962, Steven R. Hofstein and Frederick P. Heiman made the first metal-oxide semiconductor (MOS) chip. In 1964, Bryant Rogers at Fairchild invented the dual-in-line package (DIP). In the same year, Paul Baran of the Rand Corporation published an 11-volumn report describing what we now call packet switching. In 1965, Donald Davies of the NPL in the U.K. coined the term "packet". In 1966, the use of glass fiber as waveguide was proposed by Kao and Hockham. In 1967, the first published document on a packet switching network, the ARPANET, was presented. The System Network Architecture (SNA) was developed by IBM in the late 1960's. In 1969, ARPANET became operational. In 1969, the Societe Internationale de Telecommunications Aeronautiques (SITA) reorganized its switching network to act like a packet switching network. In 1969, a time-sharing service bureau, Tymshare Coorporation started installing a network based on minicomputers. In 1972, Bolt Beranek and Newman, Inc. formed Telenet Communications Coorporation. In 1973, the CYCLADES network linked serveral major computing centers throughout France. In 1974, RCP (Reseau a Commutation par Paquets) started by the French PTT Administration became operational. In 1976, the European Informatics Network (EIN) became operational. In the same year, a three-layered interface architecture for packet switching called X.25 was recommended by the International Telephone and Telegraph Consultative Committee (CCITT). In the year of 1977, both EPSS in the U.K. and DATAPAC in Canada became operational, and TYMNET was approved as a carrier in the U.S. In 1983, the seven-layered communications architecture called Reference Model for Open Systems Interconnection (OSI) was approved by the International Organization for Standardization (ISO). In 1984, a series of recommendations on Integrated Service Digital Network (ISDN) called the I-series was adopted by CCITT. Recently, Asynchronous Transfer Mode (ATM) was recommended as the target transfer mode for implementing broadband ISDN (B-ISDN).

3.7 Conclusions

We have presented in a short passage major developments in telecommunications technology of the last three centuries. We observe that in a period of approximately three hundred years, telecommunications has grown from some sporadic experiments in electricity to a colossal edifice of modern technology. This chapter has drawn on materials from many sources. The first four sections and part of section 5 are mainly from Jones [1852], Kirby et al [1956], Derry and Williams [1961] and Daumas [1979]. Pratt and Bostian [1986] and Parsons and Gardiner [1989] provide most materials for the wireless communications section. The digital computer communications section mainly comes from Powers and Stair [1990]. The two textbooks by Schwartz [1988] and Spragins [1990] also provide valuable information. Materials on packet switching are mainly due to Roberts [1978].

CHAPTER 4
A PACKETIZED VOICE COMMUNICATIONS MODEL WITH ON-OFF SOURCES

4.1 Introduction

In this chapter, we illustrate our approach through actually solving the model for the special case in which there is a one-to-one correspondence between the phase of the arrival process and the number of entities that arrive during a slot. We present a numerically stable solution methodology that yields both moments and distributions for the equilibrium process. We use classical eigenvalue/eigenvector techniques, which were found to be extremely fast and reliable by Daigle and Lucantoni [1991].

In digital commnications of the future, information will be sent in chunks of data bits known as packets. Packets are units of data which are transmitted through a myriad of electro-optical networks like letters through the mail system. Unlike dedicated communications lines in the past, a packet is stored and then forwarded at various switching nodes in the networks. This stored-and-forward mode of transmission engenders delay for the packet during its course of transmission, and buffers are required to stored it at various nodes. Studies on packet delay and buffer requirement for packet switching networks are numerous. Early studies include Chu [1969] and Rickard [1972]. Packetized voice systems are studied by Jenq [1984], Stern [1983] and Daigle and Langford [1986] etc. Sriram et al. [1983], Konheim and Reiser [1986], and Rubin and Zhang [1988, 90] study the special case where voice and data are integrated using the movable-boundary method.

Many researchers, for example Stern [1983], have pointed out that packets tend to arrive at input nodes in bursts; that is, they arrive during distinct periods of activity, which are alternated by periods of idleness during which no packets arrive. This unique arrival pattern is captured by the Markov-modulated model in Burman and Smith [1984], Hefffes and Lucantoni [1986], Stern and Elwalid [1991] and Liao and Mason [1989], and by the switched batch Bernoulli model in Hashida and Takahashi [1991]. In this chapter, we will study the queueing behavior of a packetised voice system with bursty arrivals using a batch Markovian arrivals model.

As stated in Chapter 1, the batch Markovian arrival process (BMAP) is statistically equivalent to the versatile Markovian point process (VMPP) developed by Neuts

[1979]. In the VMPP, entities arrive at a queueing system in accordance with a distribution whose parameters evolve with the state of an underlying Morkov chain known as the phase process. In Ramaswami [1980], a detailed analysis of a single-server queueing system whose input is the versatile Markovian point process is presented. A recursive algorithm for computing the steady state queue length distribution vector for the same type of queueing system is formulated in Ramaswami [1988]. Lucantoni [1991], however, points out that Ramaswami's recursive algorithm is in practice infeasible to be implemented in its full generality. Lucantoni proceeds to constuct a new batch arrival process which is notationally much simpler than but statistically equivalent to the VMPP and calls it the batch Markovian arrival process. Lucantoni also shows that the G matrix, which is essential for the computation procedures in the matrix analytic approach to queues of $BMAP/G/1$ type, has an exponential form. Other studies on single-sever queues with BMAP inputs include Bruneel [1988] and Stavrakakis [1990]; in both articles, results on average queue length are presented.

Neuts [1981] analyzes a R server queue with constant service times and a vesatile Markovian arrival process using the same matrix analytical technique as in Ramaswami [1980]. Neuts approaches the problem by partitioning the transition probability matrix into square blocks of R rows and R columns of matrices. In this way, the analysis of the R server queue is analogous to that of the single server queue but with huge matrices. Neuts' approach is theoretically sound but creates tremendous computational difficulties. In more recent work, van Arem [1990] and Li [1990] respectively present results for special cases of discrete time R server queues with BMAP arrivals. van Arem considers a slotted transmission systems in which packets are generated by N sources which alternate between passive and active periods; but in each time slot, only one change is allowed. Li [1990] considers a similiar communications system. Li writes the one step transition probability matrix as the Kronecker product of 2×2 matrices and approaches the problem via a spectral decomposition technique.

The organization of the remainder of this chapter is now described. In Section 2, we formalize the model, provide the general framework for the resolution of occupancy distribution, describe our proposed approach through an example in which there is a one-to-one correspondence between the state of the Markov chain governing the arrival process and the number of arrivals that occur during a slot. The PGF will be specified in Section 3, the zeros of the denominator of the PGF will be specified in terms of the eigenvalues of a matrix in Section 4, and the resolution of the unknown probabilities, using both eigenvalues and eigenvectors of the previously mentioned matrix, will be addressed in Section 5. Inversion of the PGF, again using eigenvalues and eigenvectors of the same matrix, will be addressed in Section 6. In Section 7, we formulate the equations for obtaining delay distribution. In Section 8, we present some numerical results. Our example demonstrates the potential of our eigenanalysis approach by showing that we can reproduce mean occupancy statistics for larger instances of the most complicated case previously presented in the literature, and that, in addition, we obtain occupancy distribution, mean delay and delay distribution. In section 9, we draw conclusions.

4.2 The Queueing Model

For the remainder of this chapter, we shall refer to the entities that are multiplexed as *packets* and the state of the Markov chain described in the previous section as the phase. A time slot will be thought of as representing the amount of time to transmit a frame. Thus, the state of the system at the end of the j-th time slot, which we shall refer to as *time* j, will be given by the point (n, m) where the first component specifies the number of packets in the system and the second component specifies the phase, both quantities being observed at the end of a slot. Thus, the process is a vector, discrete valued, discrete parameter Markov chain on the state space $\{(n, m), n \geq 0, 0 \leq m \leq N\}$, where N is the number of sources generating packets and the total number of phases is therefore $N + 1$. Let $\tilde{n}(j)$ and $\tilde{m}(j)$ denote the number of packets in the system and the state of the phase process, respectively, at the end of the j-th time slot, the evolution of the queueing system can be described by

$$\tilde{n}(j+1) = (\tilde{n}(j) - R)^+ + \tilde{a}(j+1),$$

where $(\cdot)^+ = \max\{\cdot, 0\}$, and $\tilde{a}(j+1)$ is the number of packets that arrive in the half open interval $(j, j+1]$. Let

$$o_{n,m}(j) = P\{\tilde{n}(j) = n, \tilde{m}(j) = m\}. \tag{1}$$

Also, define $o_n(j)$ to be the row vector of probabilities such that the number of packets in the system is n at time j. That is,

$$o_n(j) = \begin{bmatrix} o_{n,0}(j) & o_{n,1}(j) & \cdots & o_{n,N}(j) \end{bmatrix}. \tag{2}$$

With regard to the phase process, define $\mathcal{P}$ to be the one step state transition probability matrix. That is, define

$$p_{ik} = P\{\tilde{m}(j+1) = k | \tilde{m}(j) = i\}.$$

then

$$\mathcal{P} = \begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0N} \\ p_{10} & p_{11} & \cdots & p_{1N} \\ \cdots & \cdots & \cdots & \cdots \\ p_{N0} & p_{N1} & \cdots & p_{NN} \end{bmatrix}. \tag{3}$$

Now, there is a random number of packet arrivals to the system during any slot, and this random number is dependent upon the phase of the system. Therefore, let $\tilde{a}_m$, independent of j, denote the number of arrivals to the system during a slot in which the phase process is in state m, $0 \le m \le N$. In addition, let $P_{m,i} = P\{\tilde{a}_m = i\}$, and let

$$\begin{aligned} P_i &= \text{diag}\,(P\{\tilde{a}_0 = i\}, P\{\tilde{a}_1 = i\}, \cdots, P\{\tilde{a}_N = i\}) \\ &= \text{diag}\,(P_{0,i}, P_{1,i}, \cdots, P_{N,i}). \end{aligned} \tag{4}$$

This system is readily analyzed in a manner analogous to the ordinary *M/D/R* system. In particular, in order that the system have, say, n packets at the end of the $(j+1)$-st time slot, there must have been i packets, $0 \le i \le n+R$, present at the end of the j-th time slot and $n-(i-R)^+$ arrivals during slot $j+1$. Thus,

$$\phi_n(j+1) = \sum_{i=0}^{n+R} \phi_i(j)\mathcal{P}P_{n-(i-R)^+}. \tag{5}$$

This expression can be rewritten in the matrix form $\tilde{\Phi}(j+1) = \tilde{\Phi}(j)\tilde{\mathcal{P}}$, where

$$\tilde{\Phi}(j) = [\phi_0(j) \quad \phi_1(j) \quad \phi_2(j) \quad \phi_3(j) \quad \cdots \quad \cdots],$$

and

$$\tilde{\mathcal{P}} = \begin{bmatrix} \mathcal{P}P_0 & \mathcal{P}P_1 & \mathcal{P}P_2 & \mathcal{P}P_3 & \cdots & \cdots \\ \mathcal{P}P_0 & \mathcal{P}P_1 & \mathcal{P}P_2 & \mathcal{P}P_3 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathcal{P}P_0 & \mathcal{P}P_1 & \mathcal{P}P_2 & \mathcal{P}P_3 & \cdots & \cdots \\ 0 & \mathcal{P}P_0 & \mathcal{P}P_1 & \mathcal{P}P_2 & \cdots & \cdots \\ 0 & 0 & \mathcal{P}P_0 & \mathcal{P}P_1 & \cdots & \cdots \\ 0 & 0 & 0 & \mathcal{P}P_0 & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & \cdots \end{bmatrix},$$

where the first row of the matrix is repeated R times if there is a capacity of R packets per slot.

Thus, we see that for the special case of $R = 1$, the queueing system under discussion satisfies the *M/G/1* paradigm of Ramaswami [1980] which, as we have pointed out, has been analyzed extensively in Lucantoni [1991] and the references therein. For general values of R, methodology analogous to that of the single server case has been presented in Neuts [1981b]. Neuts's approach is to simply create larger blocks by grouping together R rows and columns of blocks of the matrix $\tilde{\mathcal{P}}$. The overall size of the blocks is then $(N+1)R \times (N+1)R$, but the new block structure is identical to that of the ordinary *M/G/1* system.

We shall first specify the probability generating function (PGF) for the joint phase, occupancy probability masses in a manner analogous to the analysis of the ordinary $M/G/1$ system. Although the analysis is straightforward, we include intermediate steps in order to discuss the notation needed later in a natural way. The resulting PGF will be specified in terms of a matrix of unknown probabilities, which will, in turn, be resolved by formulating a linear system of equations based on the zeros of the denominator of the PGF.

4.3 The Probability Generating Function

We define

$$\mathcal{G}(z,j) = \sum_{n=0}^{\infty} z^n \phi_n(j). \tag{6}$$

Then upon substitution of (6) into (5), we find after a little algebra that

$$\mathcal{G}(z,j+1) = \left\{ \sum_{i=0}^{R-1} \phi_i(j) + z^{-R} \left[\mathcal{G}(z,j) - \sum_{i=0}^{R+i-1} \phi_i(j) \right] \right\} \mathcal{P}\mathcal{F}_{\tilde{a}}(z), \tag{7}$$

where $\mathcal{F}_{\tilde{a}}(z)$ is the diagonal matrix of PGFs representing the distributions of the number of arrivals during a slot in which the system is in phase m, $0 \leq m \leq N$. That is,

$$\mathcal{F}_{\tilde{a}_m}(z) = \sum_{i=0}^{\infty} z^i P\{\tilde{a}_m = i\},$$

and

$$\mathcal{F}_{\tilde{a}}(z) = \text{diag}\ \left(\mathcal{F}_{\tilde{a}_0}(z), \mathcal{F}_{\tilde{a}_1}(z), \cdots, \mathcal{F}_{\tilde{a}_N}(z) \right). \tag{8}$$

Upon taking limits in (7) as $j \to \infty$, we find

$$\mathcal{G}(z) = \left\{ \sum_{i=0}^{R-1} \phi_i + z^{-R} \left[\mathcal{G}(z) - \sum_{i=0}^{R+i-1} \phi_i \right] \right\} \mathcal{P}\mathcal{F}_{\tilde{a}}(z), \tag{9}$$

where

$$\mathcal{G}(z) = \lim_{j \to \infty} \mathcal{G}(z,j),$$

and where we have assumed that the implied limits exist (see Hunter [1983] for existence conditions). Upon solving (9), we find

$$\mathcal{G}(z) \left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z) \right] = \sum_{i=0}^{R-1} (z^R - z^i) \phi_i \mathcal{P}\mathcal{F}_{\tilde{a}}(z). \tag{10}$$

By letting $\mathcal{B}(z) = [z^R - 1 \quad z^R - z \quad \cdots \quad z^R - z^{R-1}]$ and $\Phi = [\phi_0^T \quad \phi_1^T \quad \cdots \quad \phi_{R-1}^T]^T$, we may readily rewrite (10) as

$$\mathcal{G}(z)\left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z)\right] = \mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z). \tag{11}$$

Thus,

$$\mathcal{G}(z) = \frac{1}{\det\left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z)\right]}\mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z)\text{adj}\left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z)\right]. \tag{12}$$

where we note that $\mathcal{B}(z)$ is a row vector of dimension R and Φ is a matrix of dimension $R \times (N+1)$ and we have used the fact that $A^{-1} = (1/\det A)\text{adj } A$.
Upon postmultiplication of both sides of (11) by $\mathbf{e}$ which, as usual, denotes the column vector in which each element is unity, we find that the probability generating function for the marginal occupancy distribution is given by

$$\mathcal{F}_{\tilde{n}}(z) = \mathcal{G}(z)\mathbf{e} = \frac{1}{\det\left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z)\right]}\mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z)\text{adj}\left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z)\right]\mathbf{e}. \tag{13}$$

Equation (12) is a vector-valued version of equation (5.16) of Hayes [1986], which gives the probability generating function for the queue length distribution in a time division multiplexing system. Analogous to the scalar case in which there are R unknown probabilities, the matrix Φ is a vector of R row vectors of dimension $N+1$ of unknown probabilities, amounting to $R \times (N+1)$ unknown probabilities in the present case. For the general case, this vector of unknown coefficients can, in principle, be determined by formulating and solving a linear system of equations which result from the fact that $\mathcal{F}_{\tilde{n}}(z)$ is a PGF, and is therefore, bounded inside and on the unit circle of the complex plane (Hunter [1983]).

In the general case, a formulation of the proper linear system of equations is difficult, and alternate techniques are required. An algorithmic approach for computing the unknown vector in the BMAP case with $R = 1$ is provided in Lucantoni, Meier-Hellstern, and Neuts [1990] and Lucantoni [1991] and with arbitrary values of R in Neuts [1981,1989]. Their algorithmic approach, which appears applicable to the current case with minor modification, requires the method of successive approximations and its computational complexity is not polynomially bounded. In addition, once the unknown probabilities are known, the transform equation must still be inverted. Inversion methods developed to date, even for the case $R = 1$ require iterative computation of the unknown probabilities, thus leading to increased roundoff errors as the number of probabilities needed increases.

In the next three sections, we first address the problem of finding the null values of the matrix $[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z)]$. We then address the resolution of the unknown coefficient matrix, Φ, of (13), and next we address the inversion of (13) to obtain the probability masses.

4.4 Eigenanalysis Approach to Determining the Required Null Values

In this section, we shall restrict our attention to the case in which $P\{\tilde{a}_m = i\} = 0$ for $i \neq m$ and $P\{\tilde{a}_m = m\} = 1$, where $\tilde{a}_m$ represents the number of arrivals during a time slot in which the phase is m. In this case, (8) becomes

$$\mathcal{F}_{\tilde{a}}(z) = \Delta_{N+1}(z^i). \tag{14}$$

This is exactly the case in which there is a one-to-one correspondence between the phase of the arrival process at the end of the slot and the number of arrivals that occur during the same slot.

For this particular case, a total of $(1/2)R(R+1)$ unknown probability masses must be determined in order to complete a specification of the probability generating function for the queue occupancy. As previously pointed out, these unknown probability masses may be determined by forming a linear system of equations based on the null values of the matrix $[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z)]$ inside and on the unit circle of the complex plane. In this section, we demonstrate that these null values may be obtained by formulating the problem as an eigenvalue problem. In addition to obtaining the null values, we also obtain the null vectors which will also be instrumental in forming the required linear system of equations.

For the general case, let

$$\mathcal{A}(z) = z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z). \tag{15}$$

Then, for the case under consideration, we find that

$$\mathcal{A}(z) = z^R I - \mathcal{P}\Delta_{N+1}(z^i). \tag{16}$$

Now, $\mathcal{A}(z)$ can be factored into the form

$$\mathcal{A}(z) = \mathcal{A}_L(z)\mathcal{A}_R(z), \tag{17}$$

where

$$\mathcal{A}_R(z) = \begin{bmatrix} \Delta_{R+1}(z^i) & 0 \\ 0 & \Delta_{N-R}(z^R) \end{bmatrix} = \mathrm{diag}(\, 1 \quad z \quad \cdots \quad z^R \quad z^R \quad \cdots \quad z^R \,) \tag{18}$$

and

$$\mathcal{A}_L(z) = \begin{bmatrix} \Delta_{R+1}(z^{R-i}) & 0 \\ 0 & I_{N-R} \end{bmatrix} - \mathcal{P} \begin{bmatrix} I_{R+1} & 0 \\ 0 & \Delta_{N-R}(z^{i+1}) \end{bmatrix} = \sum_{i=0}^{N_L} \mathcal{A}_i z^i, \tag{19}$$

with

$$N_L = \max\{R, N-R\}. \tag{20}$$

The $\mathcal{A}_i$, $i = 0, 1, \cdots, N_L$, are $(N+1)$-square matrices of constants whose values follow directly from (19). We now introduce some additional notation to aid in specifying the $\mathcal{A}_i$ in compact form. Define e_i to be the column vector whose i-th element is 1 and all other elements are 0, and we define E_i to be the matrix whose i-th column is e_i and all other columns are zero. If $i < 0$ or $i > N+1$, then E_i is defined to be the $(N+1)$-square null matrix. In addition, we define I and O to be the i-square identity and null matrices, respectively. With these definitions, we are now ready to define

$$\mathcal{A}_0 = \begin{bmatrix} O_R & 0 \\ 0 & I_{N+1-R} \end{bmatrix} - \mathcal{P} \begin{bmatrix} I_{R+1} & 0 \\ 0 & O_{N-R} \end{bmatrix}, \tag{21}$$

and for $1 \le i \le N_L$,

$$\mathcal{A}_i = E_{R-i} - \mathcal{P}E_{R+i}. \tag{22}$$

In expanded form, these definitions result in the following:

$$\begin{array}{lcllllllllll}
\mathcal{A}_0 & = & [& -\mathcal{P}_0 & -\mathcal{P}_1 & \cdots & -\mathcal{P}_{R-2} & -\mathcal{P}_{R-1} & (e_R - \mathcal{P}_R) & e_{R+1} & \cdots & e_N \;] \\
\mathcal{A}_1 & = & [& 0 & 0 & \cdots & 0 & -e_{R-1} & 0 & -\mathcal{P}_{R+1} & \cdots & 0 \;] \\
\mathcal{A}_2 & = & [& 0 & 0 & \cdots & -e_{R-2} & 0 & 0 & -\mathcal{P}_{R+2} & \cdots & 0 \;] \\
\cdots & & & \cdots & & & \cdots & & & \cdots & & \cdots
\end{array}$$

For example, suppose $N = 5$ and $R = 2$. Then $N_L = 3$, and $\mathcal{A}_R(z) = \text{diag}\,(\,1 \quad z \quad z^2 \quad z^2 \quad z^2 \quad z^2\,)$, and

$$\mathcal{A}_L(z) = \sum_{i=0}^{3} \mathcal{A}_i z^i$$

with

$$\begin{array}{lclllllll}
\mathcal{A}_0 & = & [& -\mathcal{P}_0 & -\mathcal{P}_1 & (e_2 - \mathcal{P}_2) & e_3 & e_4 & e_5 \;] \\
\mathcal{A}_1 & = & [& 0 & e_1 & 0 & -\mathcal{P}_3 & 0 & 0 \;] \\
\mathcal{A}_2 & = & [& e_0 & 0 & 0 & 0 & -\mathcal{P}_4 & 0 \;] \\
\mathcal{A}_3 & = & [& 0 & 0 & 0 & 0 & 0 & -\mathcal{P}_5 \;]
\end{array}$$

where $\mathcal{P}_i$ denotes the i-th column of the matrix $\mathcal{P}$.

Now, suppose that

$$z^* \in \mathcal{Z} = \{z|\det \mathcal{A}(z) = 0\},$$

and that $\psi^*(z^*)$ is a null vector of $\mathcal{A}(z)$ corresponding to the null value z^* of $\mathcal{A}(z)$. Then,

$$\mathcal{A}_L(z^*)\mathcal{A}_R(z^*)\psi^*(z^*) = 0. \tag{23}$$

For $z^* \neq 0$, define $\psi(z^*) = \mathcal{A}_R(z^*)\psi^*(z^*)$. We then find

$$\mathcal{A}_L(z^*)\psi(z^*) = 0. \tag{24}$$

Thus, for $z^* \neq 0$, we may determine $\psi^*(z^*)$ by first finding $\psi(z^*)$, the null vector of $\mathcal{A}_L(z)$ corresponding to the null value z^* of $\mathcal{A}_L(z)$, and then premultiplying by $\mathcal{A}_R(z^*)^{-1}$, where it is obvious that the required inverse exists since all null values of $\mathcal{A}_R(z)$ are equal to zero; that is,$\det \mathcal{A}_R(z) = 0$ if and only if $z = 0$.

In order to find the null vectors of $\mathcal{A}_L(z)$, we shall convert the set of equations $\mathcal{A}_L(z)\psi(z) = 0$ into a larger set of equations of the form

$$[C_0 - zC_1]X(z) = 0, \tag{25}$$

where C_0 and C_1 are square scalar matrices and $X(z)$ is a column vector conforming to the dimensions of C_0 and C_1. The null values of $\mathcal{A}_L(z)$ will then be the same as the null values of $[C_0 - zC_1]$ and the null vectors of $\mathcal{A}_L(z)$ will be obtained by partitioning the null vectors of $[C_0 - zC_1]$. Provided that C_0^{-1} exists, the null values of $[C_0 - zC_1]$ are simply the inverses of the eigenvalues of the matrix $C_0^{-1}C_1$, and the null vectors of $\mathcal{A}_L(z)$ are simply the corresponding eigenvectors of $C_0^{-1}C_1$. Alternate procedures are readily specified for the case in which C_0^{-1} does not exist.

To facilitate completing definitions of C_0 and C_1, we first define the following transformation matrices:

$\bar{I}_1$ is defined as the matrix formed by removing rows $(R-1)$ through $(R+1)$ from the $(N+1)$-square identity matrix.

$\bar{I}_i$ For $N_L > i > 1$, $\bar{I}_i$ is defined as the matrix formed by removing rows $(R-i)^+$ through $(R+i)^-$ and columns $(R+1-i)^+$ through $(R+i-1)^-$ from the $(N+1)$-square identity matrix, where the notation $(a)^+$ denotes $\max\{a, 0\}$, and the notation $(a)^-$ denotes $\min\{a, N+1\}$.

With the above definitions, we now define

$$X(z) = \begin{bmatrix} \psi(z) \\ X_2(z) \\ X_3(z) \\ \cdots \\ X_{N_L-1}(z) \\ X_{N_L}(z) \end{bmatrix} = \begin{bmatrix} I & 0 & 0 & \cdots & 0 & 0 \\ z\bar{I}_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & z\bar{I}_2 & 0 & \cdots & 0 & 0 \\ \cdots \\ 0 & 0 & 0 & \cdots & z\bar{I}_{N_L-1} & 0 \end{bmatrix} \begin{bmatrix} \psi(z) \\ X_2(z) \\ X_3(z) \\ \cdots \\ X_{N_L-1}(z) \\ X_{N_L}(z) \end{bmatrix} \tag{26}$$

Based on the above definitions, it is then readily verified that

$$C_0 = \begin{bmatrix} \mathcal{A}_0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & I & 0 & \cdots & 0 & 0 \\ 0 & 0 & I & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & I & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}. \tag{27}$$

and

$$C_1 = \begin{bmatrix} -\mathcal{A}_1 & -\bar{\mathcal{A}}_2 & -\bar{\mathcal{A}}_3 & \cdots & -\bar{\mathcal{A}}_{N_L-1} & -\bar{\mathcal{A}}_{N_L} \\ \bar{I}_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \bar{I}_2 & 0 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & \bar{I}_{N_L-1} & 0 \end{bmatrix}. \tag{28}$$

where

$$\bar{\mathcal{A}}_i = \mathcal{A}_i \prod_{j=1}^{i-1} \bar{I}_j{}^T \quad \text{for} \quad 2 \le i \le N_L, \tag{29}$$

and the superscript T denotes the transpose operator. We note in passing that the dimension of the matrices C_0 and C_1 is

$$K = (1/2)R(R+1) + (1/2)(N-R+1) + 1. \tag{30}$$

Upon comparing the definitions of C_0 and $\mathcal{A}_0$, it is readily verified that C_0 is nonsingular if and only if $\mathcal{A}_0$ is nonsingular. Again, from the definition of $\mathcal{A}_0$, we see that the condition for nonsingularity is that the matrix

$$G = \begin{bmatrix} -p_{00} & p_{01} & \cdots & -p_{0,R-1} & -p_{0R} \\ -p_{10} & -p_{11} & \cdots & -p_{1,R-1} & -p_{1R} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ -p_{R-1,0} & -p_{R-1,1} & \cdots & -p_{R-1,R-1} & -p_{R-1,R} \\ -p_{R,0} & -p_{R,1} & \cdots & -p_{R,R-1} & 1-p_{RR} \end{bmatrix} \tag{31}$$

is nonsingular. A discussion of why this matrix is nonsingular for most cases of practical interest is given in van Arem [1990].

We offer the following proposition by the matrix generalization of Rouché's theorem (de Smit [1983]):

Proposition 4.1: The matrix $\mathcal{A}(z)$ has

(i) $(N+1)R-1$ null values inside the unit circle of the complex plane,

(ii) a null value of zero with multiplicity at least $(1/2)R(R+1)+(N-R)R$,

(iii) a null value of one with multiplicity exactly one,

(iv) at most $(1/2)(N-R+1)(N-R)$ finite null values outside the unit circle of the complex plane.

As mentioned above, the null vectors and null values of $[C_0 - zC_1]$ are readily obtained from the eigenvalues and eigenvectors of $C_0^{-1}C_1$, which may be obtained by using standard library routines. Let

$$\sigma \in \mathcal{S} = \left\{\sigma | \det \left[\sigma I - C_0^{-1}C_1\right] = 0\right\},$$

denote an eigenvalue of $C_0^{-1}C_1$, and $W(\sigma)$ its corresponding eigenvector. Also, let

$$\mathcal{S}^{[0,1)} = \{\sigma \in \mathcal{S}, |\sigma| \in [0,1)\} \quad \text{and} \quad \mathcal{S}^{(1,\infty)} = \{\sigma \in \mathcal{S}, |\sigma| \in (1,\infty)\}.$$

Then, as we have observed above, the null values of $[C_0 - zC_1]$ are given by $\mathcal{Z}^{(0,1)} \cup \mathcal{Z}^{(1,\infty)\cup\{\infty\}} \cup \{1\}$, where

$$\mathcal{Z}^{(0,1)} = \left\{z = 1/\sigma | \sigma \in \mathcal{S}^{(1,\infty)}\right\} \quad \text{and} \quad \mathcal{Z}^{(1,\infty)\cup\{\infty\}} = \left\{z = 1/\sigma | \sigma \in \mathcal{S}^{[0,1)}\right\}.$$

Additionally,

$$X(z) = W(\sigma) \quad \text{for} \quad z \in \mathcal{Z}^{(0,1)} \cup \mathcal{Z}^{(1,\infty)\cup\{\infty\}} \cup \{1\}.$$

Noting the latter, we shall use $X(z)$, $W(\sigma)$, and $X(\sigma)$ and $\psi(z)$ and $\psi(\sigma)$ interchangeably throughout the remainder of this chapter.

4.5 Resolution of the Unknown Probabilities

As we pointed out in the previous section, the probability generating function for the queue occupancy has a total of $(1/2)R(R+1)$ unknown probabilities. We shall use the null values and null vectors determined in the previous section to form a linear system of equations from which to determine the unknown probabilities.

For continuity, we repeat (11)

$$\mathcal{G}(z)\left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z)\right] = \mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z). \tag{11}$$

By substituting $\mathcal{A}_L(z)\mathcal{A}_R(z)$ for $\left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z)\right]$ into (11) and solving for $\mathcal{G}(z)$, we find

$$\mathcal{G}(z) = \mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z)\mathcal{A}_R^{-1}(z)\mathcal{A}_L^{-1}(z). \tag{32}$$

Since $\mathcal{A}_R(z)$ is diagonal, its inverse is trivially obtained. The inverse of $\mathcal{A}_L(z)$ is more difficult to obtain, but it can easily be shown that $\mathcal{A}_L^{-1}(z)$ is equal to the upper left $(N+1)$-square submatrix of $(C_0 - zC_1)^{-1}$. In order to express $\mathcal{A}_L^{-1}(z)$ in a convenient form, we first obtain a convenient expression for $(C_0 - zC_1)^{-1}$.

Let σ_i, $0 \le i \le K-1$, denote an ordering of the eigenvalues of $C_0^{-1}C_1$. With $\sigma_i = 1/z_i$ and $W = [W(\sigma_0) \quad W(\sigma_1) \quad \cdots \quad W(\sigma_{K-1})]$, we readily find from (25) that

$$W\Delta(\sigma_i) = C_0^{-1}C_1W. \tag{33}$$

By multiplying both sides of (33) by z, subtracting the result from W, premultiplying by C_0, and then solving for $(C_0 - zC_1)^{-1}$, we find

$$(C_0 - zC_1)^{-1} = W\,[I - z\Delta(\sigma_i)]^{-1}\,(C_0W)^{-1}. \tag{34}$$

If we define

$$Y = (C_0W)^{-1} = \begin{bmatrix} Y^0 \\ Y^1 \\ \cdots \\ Y^{K-1} \end{bmatrix}; \tag{35}$$

we readily find

$$\begin{aligned}(C_0 - zC_1)^{-1} &= W\,[I - z\Delta(\sigma_i)]^{-1}\,Y \\ &= \sum_{i=0}^{K-1} \frac{1}{1 - z\sigma_i} W(\sigma_i)Y^i.\end{aligned} \tag{36}$$

If we now recall the definition of $X(z_i)$ as given in (26) and the equivalence of $W(\sigma_i)$ and $X(z_i)$ for $z_i = 1/\sigma_i$, and set $Y^i = [\xi^i \quad Y_2^i \quad \cdots \quad Y_{N_L}^i]$, where the subvectors of Y^i have the same dimension as the subvectors of $W(z_i)$, we find that the upper left $(N+1)$-square submatrix of $(C_0 - zC_1)^{-1}$ can be expressed as

$$\mathcal{A}_L^{-1}(z) = \sum_{i=0}^{K-1} \frac{1}{1 - z\sigma_i} \psi(\sigma_i)\xi^i = \sum_{\{i|\sigma_i \in S\}} \frac{1}{1 - z\sigma_i} \psi(\sigma_i)\xi^i. \tag{37}$$

Upon substitution of this result into (32) and rearranging terms, we find

$$\mathcal{F}_{\tilde{n}}(z) - \mathcal{G}(z)\mathbf{e} - \sum_{\{i|\sigma_i \in S\}} \frac{1}{1 - z\sigma_i} \mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z)\mathcal{A}_R^{-1}(z)\psi(\sigma_i)\xi^i\mathbf{e}. \tag{38}$$

Since $\mathcal{F}_{\tilde{n}}(z)$ is bounded within the unit circle, we readily see that

$$\begin{aligned} 0 &= \mathcal{B}(z_i)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z_i)\mathcal{A}_R^{-1}(z_i)\psi(z_i)\xi^i\mathbf{e} \\ &= \mathcal{B}(z_i)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z_i)\mathcal{A}_R^{-1}(z_i)\psi(z_i) \quad \text{for} \quad z_i \in \mathcal{Z}^{(0,1)}. \end{aligned} \tag{39}$$

where we have used the fact that $\psi(z_i) = \psi(\sigma_i)$ for $\sigma_i = 1/z_i$, $z_i \in \mathcal{Z}^{(0,1)}$. But, since

$$\left[z_i^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z_i)\right] \mathcal{A}_R^{-1}(z_i)\psi(z_i) = 0$$

it follows that $z_i^R \mathcal{A}_R^{-1}(z_i)\psi(z_i) = \mathcal{P}\mathcal{F}_{\tilde{a}}(z)\mathcal{A}_R^{-1}(z_i)\psi(z_i)$ so that (39) becomes

$$0 = \mathcal{B}(z_i)\Phi\mathcal{A}_R^{-1}(z_i)\psi(z_i) \quad \text{for} \quad z_i \in \mathcal{Z}^{(0,1)}. \tag{40}$$

Upon substitution of the definitions of $\mathcal{B}(z_i)$and Φ, (40) becomes

$$0 = \sum_{j=0}^{R-1}(z_i^R - z_i^j)\sum_{\ell=0}^{j}\frac{1}{z_i^\ell}\Phi_{j\ell}\psi_\ell(z_i) \quad \text{for} \quad z_i \in \mathcal{Z}^{(0,1)}. \tag{41}$$

Since there are $(1/2)R(R+1) - 1$ elements of $\mathcal{Z}^{(0,1)}$, only one additional equation is required to completly specify the unknown probabilities.

The required additional equation is based on the null value of $\mathcal{A}(z)$ at $z = 1$. Differentiating both sides of (11) yields

$$\mathcal{G}'(z)\left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z)\right] + \mathcal{G}(z)\left[Rz^{R-1}I - \mathcal{P}\mathcal{F}_{\tilde{a}}'(z)\right] = \mathcal{B}'(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z) + \mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}'(z).$$

Upon postmultiplying both sides of the above equations by $\mathbf{e}$ and taking limits as $z \to 1$, we find

$$R - \mathcal{G}(1)\mathcal{F}_{\tilde{a}}'(1)\mathbf{e} = \mathcal{B}'(1)\Phi\mathbf{e}, \tag{42}$$

where we have used the fact that $\mathcal{F}_{\tilde{a}}(1)$ is an identity matrix so that $\mathcal{F}_{\tilde{a}}(1)\mathbf{e} = \mathbf{e}$ and that $\mathcal{G}(1)$ is the stationary probability vector for $\mathcal{P}$ so that $\mathcal{G}(1)\mathcal{P} = \mathcal{G}(1)$ and $\mathcal{G}(1)\mathbf{e} = 1$. Again, since $\mathcal{G}(1)$ is the stationary probability vector for $\mathcal{P}$, the quantity $\mathcal{G}(1)\mathcal{F}_{\tilde{a}}'(1)\mathbf{e}$ represents the overall average number of arrivals per slot in stochastic equilibrium. We therefore define

$$E[\tilde{a}] = \mathcal{G}(1)\mathcal{F}_{\tilde{a}}'(1)\mathbf{e}.$$

Finally, by substituting the appropriate definitions for $\mathcal{B}(z)$ and Φ into (42), we have

$$R - E[\tilde{a}] = \sum_{i=0}^{R-1}(R-i)\sum_{j=0}^{i}\Phi_{ij}. \tag{43}$$

This last equation then completes the specification of the $(1/2)R(R+1)$ linear equations that are needed to solve for the unknown probabilities in the occupancy distribution probability generating function.

4.6 Inversion of the Probability Generating Function

The expression for the probability generating function for the occupancy distribution was presented in (38). In this section, we invert the probability generating function to obtain the occupancy probabilities. For continuity, we repeat (38).

$$\mathcal{F}_{\tilde{n}}(z) = \sum_{\{i|\sigma_i \in \mathcal{S}\}} \frac{1}{1 - z\sigma_i} \mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z)\mathcal{A}_R^{-1}(z)\psi(\sigma_i)\xi^i\mathbf{e}. \tag{38}$$

A more convenient form for (38) may be obtained by recognizing that $\mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z)$ $\mathcal{A}_R^{-1}(z)\psi(\sigma_i)\xi^i\mathbf{e}$ is simply a polynomial. If we let $b_i(z)$ represent this polynomial, then from the definitions of $\mathcal{B}(z)$, $\mathcal{F}_{\tilde{a}}(z)$, and $\mathcal{A}_R^{-1}(z)$, we find

$$b_i(z) = \mathbf{e}^T\Delta_{R+1}(z^j)\begin{bmatrix} -\Phi\mathcal{P} \\ \mathbf{e}^T\Phi\mathcal{P}\end{bmatrix}\begin{bmatrix} I_{R+1} & 0 \\ 0 & \Delta_{N-R}(z^{j+1})\end{bmatrix}\psi(\sigma_i)\xi^i\mathbf{e}, \tag{44}$$

and

$$\mathcal{F}_{\tilde{n}}(z) = \sum_{\{i|\sigma_i \in \mathcal{S}\}} \frac{1}{1 - z\sigma_i} b_i(z). \tag{45}$$

If we denote the coefficient of z^j in (44) by b_{ij}, we find

$$b_i(z) = \sum_{j=0}^{N} b_{ij}z^j, \tag{46}$$

where the coefficients are easily determined by examination of (44).

To facilitate further discussion, we expand (45) according to the location of the elements of $\mathcal{S}$ as follows:

$$\mathcal{F}_{\tilde{n}}(z) = b_0(z) + \sum_{\{i|\sigma_i \in \mathcal{S}^{[1,\infty)}\}} \frac{1}{1 - z\sigma_i} b_i(z) + \sum_{\{i|\sigma_i \in \mathcal{S}^{(0,1)}\}} \frac{1}{1 - z\sigma_i} b_i(z), \tag{47}$$

where $\sigma_0 = 0$. Now, from the definition of C_1, it is readily seen that $\psi(0) = e_R$. Thus, by examination of (44), we see that $b_0(z)$ is a polynomial of degree R. Also, since $\mathcal{F}_{\tilde{n}}(z)$ contains no singularities in the unit disc of the complex plane, $(1 - \sigma_i z)$ is a factor of $b_i(z)$ for $\sigma_i \in \mathcal{S}^{[1,\infty)}$. Therefore,

$$\sum_{\{i|\sigma_i \in \mathcal{S}^{[1,\infty)}\}} \frac{1}{1 - z\sigma_i} b_i(z) = \sum_{\{i|\sigma_i \in \mathcal{S}^{[1,\infty)}\}} \gamma_i(z), \tag{48}$$

where

$$\gamma_i(z) = \frac{b_i(z)}{1 - \sigma_i z} \tag{49}$$

and $\gamma_i(z)$ is a polynomial of order $N-1$. Finally,

$$\sum_{\{i|\sigma_i\in\mathcal{S}^{(0,1)}\}} \frac{1}{1-z\sigma_i} b_i(z) = \sum_{\{i|\sigma_i\in\mathcal{S}^{(0,1)}\}} \gamma_i(z) + z^N \sum_{\{i|\sigma_i\in\mathcal{S}^{(0,1)}\}} \frac{r_i}{1-\sigma_i z}, \tag{50}$$

where again $\gamma_i(z)$ is a polynomial of order $N-1$, and

$$r_i = \sigma_i^N b_i(\sigma_i^{-1}). \tag{51}$$

If we write

$$\gamma_i(z) = \sum_{j=0}^{N-1} \gamma_{ij} z^j, \tag{52}$$

it is readily seen that

$$\gamma_{ij} = \sum_{k=0}^{j} b_{ik}\sigma_i^{j-k}, \tag{53}$$

and

$$r_i = \sum_{k=0}^{N} b_{ik}\sigma_i^{N-k}. \tag{54}$$

Upon substitution of (48)-(54) into (47), we find

$$\mathcal{F}_{\tilde{n}}(z) = \sum_{j=0}^{N-1} z^j \left[b_{0j} + \sum_{\{i|\sigma_i\in\mathcal{S}^{(0,\infty)}\}} \sum_{k=0}^{j} b_{ik}\sigma_i^{j-k} \right] + z^N \sum_{\{i|\sigma_i\in\mathcal{S}^{(0,1)}\}} \sum_{k=0}^{N} b_{ik}\sigma_i^{N-k} \frac{1}{1-\sigma_i z}. \tag{55}$$

The occupancy probabilities, $P\{\tilde{n}=n\}$ are now easily obtained from (55) by simply taking the coefficient of z^n. We find

$$P\{\tilde{n}=n\} = \begin{cases} b_{0n} + \sum_{\{i|\sigma_i\in\mathcal{S}^{(0,\infty)}\}} \sum_{k=0}^{n} b_{ik}\sigma_i^{n-k} & \text{for } n < N \\ \sum_{\{i|\sigma_i\in\mathcal{S}^{(0,1)}\}} \sigma_i^{n-N} \sum_{k=0}^{N} b_{ik}\sigma_i^{N-k} & \text{for } n \geq N. \end{cases} \tag{56}$$

This concludes our specification of the occupancy probability masses. A procedure to obtain the joint phase-occupancy probability masses could have been similarly specified. The primary difference in the approach is that all operations would have been performed without postmultiplication by $\mathbf{e}$ with the result that the $b_i(z)$ and $\gamma_i(z)$ would have been vectors of polynomials rather than scalar polynomials.

The mean occupancy can be obtained by differentiating (55) and taking limits as $z \to 1$, that is $E[\tilde{n}] = \mathcal{F}'_{\tilde{n}}(1)$.

4.7 Computation of the Delay Distribution

Once the occupancy probabilities are found, computation of the delay mass probabilities is straightforward. Define ϕ_n to be the $N+1$ order row vector whose m-th element is the stationary joint probability such that the number of packets in the system is n and the phase process is in state m, and $\mathrm{a}_k = \mathcal{P}P_k\mathbf{e}$ to be the $N+1$ order column vector whose m-th element is the transition probability such that the number of arrivals in the $(j+1)$-st time slot is k, given the phase process at the end of the j-th time slot is m. The expected number of arrivals which are delayed for d units of time is then given by

$$E[\tilde{a}, d] = \sum_{n=0}^{R} \phi_n \left[\sum_{k=(d-1)R+1}^{\min\{dR,N\}} \mathrm{a}_k(k-(d-1)R) + \sum_{k=\min\{dR,N\}+1}^{N} \mathrm{a}_k R \right] \quad \text{for} \quad d=1, \tag{57}$$

$$E[\tilde{a}, d] = RHS(57) + \sum_{n=(d-1)R+1}^{dR-1} \phi_n \left[\sum_{k=1}^{dR-n} \mathrm{a}_k k + \sum_{k=dR-n+1}^{N} \mathrm{a}_k(dR-n) \right] \quad \text{for} \quad d=2, \tag{58}$$

$$E[\tilde{a}, d] = RHS(58) + \sum_{n=R+1}^{(d-1)R} \phi_n \left[\sum_{k=(d-1)R-n+1}^{\min\{dR-n,N\}} \mathrm{a}_k(k+n-(d-1)R) + \sum_{k=\min\{dR-n,N\}+1}^{N} \mathrm{a}_k R \right] \quad \text{for} \quad d \geq 3, \tag{59}$$

where RHS(i) represent the right hand side of equation (i).

Let $\tilde{d}$ denote the time of delay, the delay probabilities are readily obtained by

$$P\{\tilde{d} = d\} = \frac{E[\tilde{a}, d]}{E[\tilde{a}]} \quad \text{for} \quad d = 1, 2, 3, \cdots. \tag{60}$$

Here $E[\tilde{a}]$ is the expected number of arrivals given by

$$E[\tilde{a}] = \nu \left[\sum_{k=1}^{N} \mathrm{a}_k k \right],$$

where ν is the row vector of stationary phase probabilities obtained from $\nu\mathcal{P} = \nu$, and $\nu\mathbf{e} = 1$.

The mean delay can be computed using Little's result

$$E[\tilde{d}] = \frac{E[\tilde{n}]}{E[\tilde{a}]} \tag{61}$$

or directly from the delay probability masses

$$E[\tilde{d}] = 1 + \sum_{d=1}^{\infty} \left(1 - \sum_{i=1}^{d} P\{\tilde{d} = i\} \right). \tag{62}$$

The extra 1 unit of delay on the right hand side of (62) derives from the fact that newly arrived packets must wait for the departing packets to be cleared before they can be processed.

4.8 Numerical Examples

In this section, we present numerical examples of stationary distributions and mean values for occupancy and delay. In particular, there is a finite number, N, of identical sources. Define α to be the probability of remaining active and β the probability of remaining idle, then the lengths of active periods are geometrically distributed with parameter $(1-\alpha)$ and the lengths of idle periods geometrically distributed with parameter $(1-\beta)$. The expected length of an active period is then given by $(1-\alpha)^{-1}$ and the expected length of an idle period by $(1-\beta)^{-1}$. Denote the expected length of an active period and the expected length of an idle period by EA and EP respectively, and the traffic intensity by ρ. From the theory of alternating renewal processes (Ross [1983, 85]), we obtain the traffic intensity by

$$\rho = \frac{EA}{EA+EP} \times \frac{N}{R}. \tag{63}$$

We note that the number of active sources in an arbitrary time slot is binomially distributed with parameters N and ρ. Define the state of the phase process to be the number of active sources; that is, there are m, $0 \le m \le N$, active sources when the phase process is in state m. The phase transition probabilities are computed by

$$P\{\tilde{m}(j+1)=k|\tilde{m}(j)=i\} = \sum_{r=(k-(N-i))^+}^{\min\{k,i\}} \binom{i}{r}\binom{N-i}{k-r} \alpha^r(1-\alpha)^{i-r}\beta^{[(N-i)-(k-r)]}(1-\beta)^{k-r}. \tag{64}$$

Figures 4.1 and 4.2 provide an example of the type of numerical results that may be obtained from our method. The input parameters for the system are those used by van Arem [1990] to model voice traffic on a slotted ring. Figure 4.1 shows the complementary occupancy distribution function for several values of R, which represents the number of slots, with the number of users, N, fixed at 18. van Arem [1990] did not present distributions, and the largest example therein is for $N = 10$. Thus, we could not compare our results for a model of this size. Figure 4.2 shows the complementary delay distribution for several values of R.

In addition to the results shown in Figures 4.1 and 4.2, we ran all numerical examples presented by van Arem, and the results are presented in Tables 4.1 and 4.2. Furthermore, we present in Table 4.3 the mean values for van Arem's representative voice traffic model with 22 phases for several feasible values of R. In all these tables, the traffic intensity or utilization is given by $E[\tilde{a}]/R$, the mean occupancy is computed from $\mathcal{F}'_{\tilde{n}}(1)$, the absolute error which measures the degree to which the resulting probabilities sum to unity is obtained by $|1.0 - \mathcal{F}_{\tilde{n}}(1)|$ and the mean delay is calculated from Little's result. From the tables, it is readily seen that our techniques yield extremely accurate results.

4.9 Conclusions

We demonstrated in this chapter the feasibility of our approach through the solution of a special but nontrivial subclass of the class of queueing systems we have in mind, namely, the BMAP/D/R queueing system. In particular, we address the subclass in which there is a one-to-one correspondence between the number of arrivals during a slot and the state of the Markov chain governing the arrival process. We obtained both the occupancy distribution and the mean occupancy for this model for which, to the best of our knowledge, only mean values have been presented in the literature to date. In addition, while we have not exhausted our capabilities, we analyzed systems having as many as up to 22 phases and 20 servers while the best techniques to date seem to have failed with more than 7 phases. Extension to obtaining mean delay and delay distribution is also investigated, and numerical results are obtained.

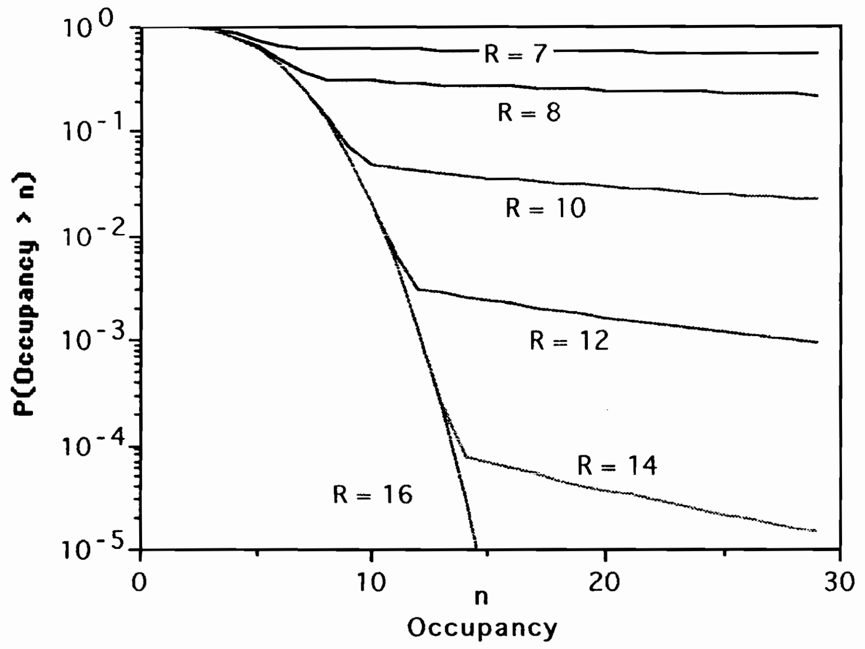

Figure 4.1. Residual distribution functions of occupancy (no speed conversion, N = 18).

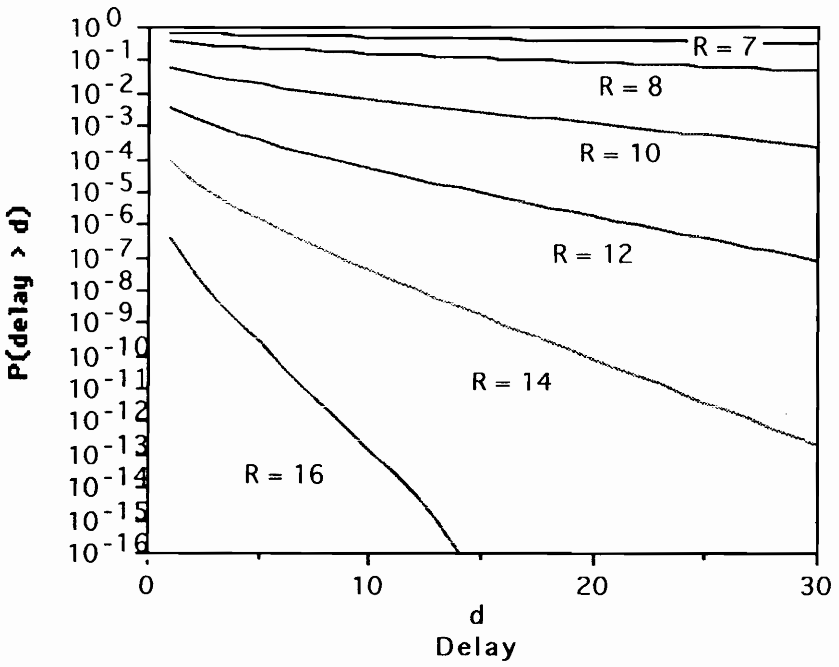

Figure 4.2. Residual distribution functions of delay (no speed conversion, N = 18).

Table 4.1.

Mean occupancy, mean delay and absolute error for van Arem's representative traffic with 6 sources for all feasible values of R.

R	Utilization %	Mean Occupancy	Mean Delay	Absolute Error
3	69.2	2.719264e+1	1.309275e+1	6.055269e-17
4	51.9	3.957621e+0	1.905521e+0	5.442695e-17
5	41.6	2.162381e+0	1.041147e+0	9.595189e-18
6	34.6	2.076923e+0	1.000000e+0	———

Table 4.2.

Mean occupancy, mean delay and absolute error for van Arem's nice traffic with 10 sources for all feasible values of R.

R	Utilization %	Mean Occupancy	Mean Delay	Absolute Error
4	93.8	9.316806e+1	2.484482e+1	1.375148e-15
5	75.0	9.782579e+0	2.608688e+0	3.127923e-17
6	62.5	4.583619e+0	1.222298e+0	1.149254e-17
7	53.6	3.849530e+0	1.026541e+0	5.377643e-17
8	46.9	3.758128e+0	1.002167e+0	1.084202e-19
9	41.7	3.750328e+0	1.000087e+0	1.734723e-17
10	37.5	3.750000e+0	1.000000e+0	———

Table 4.3.

Mean occupancy, mean delay and absolute error for van Arem's representative traffic with 22 sources for selected values of R.

R	Utilization %	Mean Occupancy	Mean Delay	Absolute Error
8	95.2	5.979745e+2	7.85219e+1	2.442436e-15
10	76.2	2.735915e+1	3.592616e+0	1.436026e-16
12	63.5	8.910448e+0	1.615385e+0	3.014082e-17
14	54.4	7.673494e+0	1.007631e+0	3.666772e-16
16	47.6	7.616792e+0	1.000185e+0	2.059984e-17
18	42.3	7.615399e+0	1.000002e+0	1.105886e-17
20	38.1	7.615385e+0	1.000000e+0	2.688821e-17
22	34.6	7.615385e+0	1.000000e+0	———

CHAPTER 5
A PACKETIZED VOICE COMMUNICATIONS MODEL WITH LINE SPEED CONVERSION

5.1 Introduction

In this chapter, we will extend our methodology developed in Chapter 4 to the analysis of a more complicated communications model. Specifically, while in the model of Chapter 4 each source alternates between active and idle periods, and each source generates one packet each time slot during an active period, in the model of this chapter each source generates one packet every, say, T slots of time during an active period. We present a numerically stable solution method based on classical eigenvalue/eigenvector techniques that yields both moments and distributions for the aforementioned equilibrium queueing process.

The bursty characteristics of the input streams arriving at the multiplexers of asynchronous digital communications systems are well noted in the literature. Yet another salient feature is quite often ignored; it is not unusual that the input lines of the multiplexers have a speed lower than the ouput chunk, thus creating bursts of packet arrivals that are periodic in nature. In this chapter, we will investigate the behavior of a class of queueing models that exhibit the distinct charateristics of this sort.

It is noted in Rickard [1972] that a buffered data network has the capability of providing speed conversion in addition to reducing bandwidth requirements and accepting messages quickly. Unfortunately, Rickard does not elaborate on this special feature of the buffered data network. Eckberg [1979] presents an algorithm for computing the delay distribution of a single server queue whose arrival process is the supperposition of a finite number of independent equivalent deterministic streams of equally spaced packets of fixed message lengths. Eckberg's models does not take account of the burstiness of the arrival streams, nor the randomness of the messages lengths.

A similar model is considered in Roberts and Virtamo [1991]. In that paper, a closed form expression for the queue length distribution for cases where all streams have the same period is derived, and upper and lower bounds for cases where the periods are different are obtained. Robert and Virtamo's model considers bursts of periodic streams of packets or cells. Specifically, they consider a single server queue handling N sources, each transmitting one cell every D units of time, and the N cell arrival epochs are independently and uniformly distributed in an interval $[t-D, t)$.

Descloux [1991] describes a more sophisticated model of a multiserver queue in which the cells of a given burst arrive over randomly spaced time slots, the leading cells arrive according to a discretized Poisson process, and the number of cells in a given burst has a modified geometric distribution. Descloux presents results in the form of graphs which show cell loss probabilities and some queue length averages in terms of average burst sizes and average spacings. Recently, Landry and Stavrakakis [1994a, b] studied the queueing behavior of a finite-capacity multiplexer with periodic Markovian sources. They present numerical results for both queue length distribution and packet loss distribution.

Our present investigation considers an infinite-capacity multiserver discrete time queue in which N sources generate bursts of cells which arrive periodically during active periods. However, the length of the idle period in units of time and the number of cells in a burst are geometrically distributed with parametes $(1-\beta)$ and $(1-\alpha)$ respectively. We capture both burstiness of the arrival streams and the randomness of the message lengths in a finite state Markov chain. We derive expressions for both distributions and averages for queue length and delay.

The organization of the remainder of this chapter is now described. In Section 2, we formalize the model and provide the general framework for the resolution of occupancy distribution. The PGF will be specified in Section 3, the zeros of the denominator of the PGF will be specified in terms of the eigenvalues of a matrix in Section 4, and the resolution of the unknown probabilities, using both eigenvalues and eigenvectors of the previously mentioned matrix, will be addressed in Section 5. Inversion of the PGF, again using eigenvalues and eigenvectors of the same matrix, will be addressed in Section 6. In Section 7, we formulate the equations for obtaining the delay distribution. In Section 8, we present some numerical results. Our example demonstrates the potential of our eigenanalysis approach by showing that we can reproduce both mean statistics and distributions. In section 9, we draw conclusions.

5.2 The Queueing Model

For the remainder of this chapter, we shall refer to the entities that are multiplexed as *packets* and the state of the Markov chain described in the previous section as the phase. A time slot will be thought of as representing the amount of time to transmit a frame. Thus, the state of the system at the end of the j-th time slot, which we shall refer to as *time* j, will be given by the point (n, m) where the first component specifies the number of packets in the system and the second component specifies the phase, both quantities being observed at the end of a slot. Thus, the process is a vector, discrete valued, discrete parameter Markov chain on the state space $\{(n, m), n \geq 0, 1 \leq m \leq M\}$, where $M = \binom{N+T}{N}$ is the total number of phases in the arrival process, N is the number of sources generating packets and there are $T+1$ states in each individual source's phase process. Let $\tilde{n}(j)$ and $\tilde{m}(j)$ denote the number of packets in the system and the state of the phase process, respectively, at the end of the j-th time slot, the evolution of the queueing system can be described by

$$\tilde{n}(j+1) = (\tilde{n}(j) - R)^+ + \tilde{a}(j+1),$$

where $(\cdot)^+ = \max\{\cdot, 0\}$, and $\tilde{a}(j+1)$ is the number of packets that arrive in the half open interval $(j, j+1]$. Let

$$\phi_{n,m}(j) = P\{\tilde{n}(j) = n, \tilde{m}(j) = m\}. \tag{1}$$

Also, define $\phi_n(j)$ to be the row vector of probabilities such that the number of packets in the system is n at time j. That is,

$$\phi_n(j) = [\phi_{n,1}(j) \quad \phi_{n,1}(j) \quad \cdots \quad \phi_{n,M}(j)]. \tag{2}$$

With regard to the phase process, define $\mathcal{P}$ to be the one step state transition probability matrix. That is, define

$$p_{ik} = P\{\tilde{m}(j+1) = k | \tilde{m}(j) = i\}.$$

Then

$$\mathcal{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1M} \\ p_{21} & p_{22} & \cdots & p_{2M} \\ \cdots & \cdots & \cdots & \cdots \\ p_{M1} & p_{M2} & \cdots & p_{MM} \end{bmatrix}. \tag{3}$$

Now, there is a random number of packet arrivals to the system during any slot, and this random number is dependent upon the phase of the system. Therefore, let $\tilde{a}_m$, independent of j, denote the number of arrivals to the system during a slot in which the phase process is in state m, $1 \leq m \leq M$. In addition, let $P_{m,i} = P\{\tilde{a}_m = i\}$, and let

$$\begin{aligned} P_i &= \text{diag}\,(P\{\tilde{a}_1 = i\}, P\{\tilde{a}_2 = i\}, \cdots, P\{\tilde{a}_M = i\}) \\ &= \text{diag}\,(P_{1,i}, P_{2,i}, \cdots, P_{M,i}). \end{aligned} \tag{4}$$

This system is readily analyzed in a manner analogous to the ordinary *M/D/R* system. In particular, in order that the system have, say, n packets at the end of the $(j+1)$-st time slot, there must have been i packets, $0 \le i \le n+R$, present at the end of the j-th time slot and $n-(i-R)^+$ arrivals during slot $j+1$. Thus,

$$\phi_n(j+1) = \sum_{i=0}^{n+R} \phi_i(j)\mathcal{P}P_{n-(i-R)^+}. \tag{5}$$

This expression can be rewritten in the matrix form $\tilde{\Phi}(j+1) = \tilde{\Phi}(j)\tilde{\mathcal{P}}$, where

$$\tilde{\Phi}(j) = [\phi_0(j) \quad \phi_1(j) \quad \phi_2(j) \quad \phi_3(j) \quad \cdots \quad \cdots],$$

and

$$\tilde{\mathcal{P}} = \begin{bmatrix} \mathcal{P}P_0 & \mathcal{P}P_1 & \mathcal{P}P_2 & \mathcal{P}P_3 & \cdots & \cdots \\ \mathcal{P}P_0 & \mathcal{P}P_1 & \mathcal{P}P_2 & \mathcal{P}P_3 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathcal{P}P_0 & \mathcal{P}P_1 & \mathcal{P}P_2 & \mathcal{P}P_3 & \cdots & \cdots \\ 0 & \mathcal{P}P_0 & \mathcal{P}P_1 & \mathcal{P}P_2 & \cdots & \cdots \\ 0 & 0 & \mathcal{P}P_0 & \mathcal{P}P_1 & \cdots & \cdots \\ 0 & 0 & 0 & \mathcal{P}P_0 & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & \cdots \end{bmatrix},$$

where the first row of the matrix is repeated R times if there is a capacity of R packets per slot.

Thus, we see that for the special case of $R=1$, the queueing system under discussion satisfies the *M/G/1* paradigm of Ramaswami [1980] which, as we have pointed out, has been analyzed extensively in Lucantoni [1991] and the references therein. For general values of R, methodology analogous to that of the single server case has been presented in Neuts [1981b]. Neuts's approach is to simply create larger blocks by grouping together R rows and columns of blocks of the matrix $\tilde{\mathcal{P}}$. The overall size of the blocks is then $(M \times R) \times (M \times R)$, but the new block structure is identical to that of the ordinary *M/G/1* system.

We shall first specify the probability generating function (PGF) for the joint phase, occupancy probability masses in a manner analogous to the analysis of the ordinary $M/G/1$ system. Although the analysis is straightforward, we include intermediate steps in order to discuss the notation needed later in a natural way. The resulting PGF will be specified in terms of a matrix of unknown probabilities, which will, in turn, be resolved by formulating a linear system of equations based on the zeros of the denominator of the PGF.

5.3 The Probability Generating Function

We define

$$\mathcal{G}(z,j) = \sum_{n=0}^{\infty} z^n \phi_n(j). \tag{6}$$

Then upon substitution of (6) into (5), we find after a little algebra that

$$\mathcal{G}(z,j+1) = \left\{ \sum_{i=0}^{R-1} \phi_i(j) + z^{-R} \quad \mathcal{G}(z,j) - \sum_{i=0}^{R+i-1} \phi_i(j) \right\} \mathcal{P}\mathcal{F}_{\tilde{a}}(z), \tag{7}$$

where $\mathcal{F}_{\tilde{a}}(z)$ is the diagonal matrix of PGFs representing the distributions of the number of arrivals during a slot in which the system is in phase m, $1 \le m \le M$. That is,

$$\mathcal{F}_{\tilde{a}_m}(z) = \sum_{i=0}^{\infty} z^i P\{\tilde{a}_m = i\},$$

and

$$\mathcal{F}_{\tilde{a}}(z) = \text{diag } \left(\mathcal{F}_{\tilde{a}_1}(z), \mathcal{F}_{\tilde{a}_2}(z), \cdots, \mathcal{F}_{\tilde{a}_M}(z) \right). \tag{8}$$

Upon taking limits in (7) as $j \to \infty$, we find

$$\mathcal{G}(z) = \left\{ \sum_{i=0}^{R-1} \phi_i + z^{-R} \quad \mathcal{G}(z) - \sum_{i=0}^{R+i-1} \phi_i \right\} \mathcal{P}\mathcal{F}_{\tilde{a}}(z), \tag{9}$$

where

$$\mathcal{G}(z) = \lim_{j \to \infty} \mathcal{G}(z,j),$$

and where we have assumed that the implied limits exist (see Hunter [1983] for existence conditions). Upon solving (9), we find

$$\mathcal{G}(z) \left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z) \right] = \sum_{i=0}^{R-1} (z^R - z^i) \phi_i \mathcal{P}\mathcal{F}_{\tilde{a}}(z). \tag{10}$$

By letting $\mathcal{B}(z) = [z^R - 1 \quad z^R - z \quad \cdots \quad z^R - z^{R-1}]$ and $\Phi = [\phi_0^T \quad \phi_1^T \quad \cdots \quad \phi_{R-1}^T]^T$, we may readily rewrite (10) as

$$\mathcal{G}(z) \left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z) \right] = \mathcal{B}(z) \Phi \mathcal{P}\mathcal{F}_{\tilde{a}}(z). \tag{11}$$

Thus,

$$\mathcal{G}(z) = \frac{1}{\det \left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z) \right]} \mathcal{B}(z) \Phi \mathcal{P}\mathcal{F}_{\tilde{a}}(z) \text{adj } \left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z) \right], \tag{12}$$

where we note that $\mathcal{B}(z)$ is a row vector of dimension R and Φ is a matrix of dimension $R \times M$ and we have used the fact that $A^{-1} = (1/\det A) \text{adj } A$.

Upon postmultiplication of both sides of (11) by e which, as usual, denotes the column vector in which each element is unity, we find that the probability generating function for the marginal occupancy distribution is given by

$$\mathcal{F}_{\tilde{n}}(z) = \mathcal{G}(z)\mathbf{e} = \frac{1}{\det\left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z)\right]}\mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z)\text{adj}\left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z)\right]\mathbf{e}. \tag{13}$$

Equation (12) is a vector-valued version of equation (5.16) of Hayes [1986], which gives the probability generating function for the queue length distribution in a time division multiplexing system. Analogous to the scalar case in which there are R unknown probabilities, the matrix Φ is a vector of R row vectors of dimension M of unknown probabilities, amounting to $R \times M$ unknown probabilities in the present case. For the general case, this vector of unknown coefficients can, in principle, be determined by formulating and solving a linear system of equations which result from the fact that $\mathcal{F}_{\tilde{n}}(z)$ is a PGF, and is therefore, bounded inside and on the unit circle of the complex plane (Hunter [1983]), and some other conditions.

In the general case, formulation of the proper linear system of equations is difficult, and alternate techniques are required. An algorithmic approach for computing the unknown vector in the BMAP case with $R = 1$ is provided in Lucantoni, Meier-Hellstern, and Neuts [1990] and Lucantoni [1991] and with arbitrary values of R in Neuts [1981b,1989]. Their algorithmic approach, which appears applicable to the current case with minor modification, requires the method of successive approximations and its computational complexity is not polynomially bounded. In addition, once the unknown probabilities are known, the transform equation still must be inverted. Inversion methods developed to date, even for the case $R = 1$ require iterative computation of the unknown probabilities, thus leading to increasing roundoff error as the number of probabilities needed is increased.

In the next three sections, we first address the problem of finding the null values of the matrix $\left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z)\right]$. We then address resolution of the unknown coefficient matrix, Φ, of (13), and next we address inversion of (13) to obtain the probability masses.

5.4 Eigenanalysis Approach to Determining the Required Null Values

Define M_i, $0 \le i \le N$, to be the number of phases in which i packets arrive at the system. By a simple combination argument,

$$M_i = \binom{N - i + T - 1}{N - i},$$

and the total number of phases, M, in the arrival process equals to $\sum_{i=0}^{N} M_i$. In this section, we shall restrict our attention to the case in which $P\{\tilde{a}_m = i\} = 1$ for $1 \le m \le M_0$ for $i = 0$ and $\sum_{k=0}^{i-1} M_k < m \le \sum_{k=0}^{i} M_k$ for $i \ge 1$, and $P\{\tilde{a}_m = i\} = 0$ otherwise; where $\tilde{a}_m$ represents the number of arrivals during a time slot in which the phase is m. In this case, (8) becomes

$$\mathcal{F}_{\tilde{a}}(z) = \text{diag}\ (\text{diag}_{M_0}(1), \text{diag}_{M_1}(z), \cdots, \text{diag}_{M_N}(z^N)), \tag{14}$$

where $\text{diag}_n(x)$ is an $n \times n$ diagonal matrix whose diagonal elements are all x.

For this particular case, a total of $\sum_{i=0}^{R-1} M_i$ unknown probability masses must be determined in order to complete specification of the the probability generating function for the queue occupancy. As previously pointed out, these unknown probability masses may be determined by forming a linear system of equations based on the null values of the matrix $[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z)]$ inside and on the unit circle of the complex plane, plus some other conditions. In this section, we demonstrate that these null values may be obtained by formulating the problem as an eigenvalue problem. In addition to obtaining the null values, we also obtain the null vectors which will also be instrumental in forming the required linear system of equations.

For the general case, let

$$\mathcal{A}(z) = z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z). \tag{15}$$

Then, for the case under consideration, we find that

$$\mathcal{A}(z) = z^R I - \mathcal{P}\text{diag}\ (\text{diag}_{M_0}(1), \text{diag}_{M_1}(z), \cdots, \text{diag}_{M_N}(z^N)). \tag{16}$$

Now, $\mathcal{A}(z)$ can be factored into the form

$$\mathcal{A}(z) = \mathcal{A}_L(z)\mathcal{A}_C(z)\mathcal{A}_R(z), \tag{17}$$

where

$$\mathcal{A}_R(z) = \text{diag}\ (\ \text{diag}_{M_0}(1) \quad \text{diag}_{M_1}(z) \quad \cdots \quad \text{diag}_{\sum_{j=R}^{N} M_j}(z^R)\ ), \tag{18}$$

$$\mathcal{A}_C(z) = \sum_{i=0}^{N_L} \mathcal{A}_i z^i, \tag{19}$$

and

$$\mathcal{A}_L(z) = \text{diag}\ (z_i), \tag{20}$$

where z_i, $i = 1, 2, \cdots, M$, are powered terms of z with nonnegative powers less than or equal to $\max\{R, N - R\}$. The definitions of N_L and $\mathcal{A}_i$, $i = 0, 1, \cdots, N_L$, follow directly from (16)-(20).

Now, suppose that

$$z^* \in \mathcal{Z} = \{z|\det \mathcal{A}(z) = 0\},$$

and that $\psi^*(z^*)$ is a null vector of $\mathcal{A}(z)$ corresponding to the null value z^* of $\mathcal{A}(z)$. Then,

$$\mathcal{A}_L(z^*)\mathcal{A}_C(z^*)\mathcal{A}_R(z^*)\psi^*(z^*) = 0. \tag{23}$$

For $z^* \neq 0$, define $\psi(z^*) = \mathcal{A}_R(z^*)\psi^*(z^*)$. We then find

$$\mathcal{A}_C(z^*)\psi(z^*) = 0. \tag{24}$$

Thus, for $z^* \neq 0$, we may determine $\psi^*(z^*)$ by first finding $\psi(z^*)$, the null vector of $\mathcal{A}_C(z)$ corresponding to the null value z^* of $\mathcal{A}_C(z)$, and then premultiplying by $\mathcal{A}_R(z^*)^{-1}$, where it is obvious that the required inverse exists since all null values of $\mathcal{A}_R(z)$ are equal to zero; that is,det $\mathcal{A}_R(z) = 0$ if and only if $z = 0$.

In order to find the null vectors of $\mathcal{A}_C(z)$, we shall convert the set of equations $\mathcal{A}_C(z)\psi(z) = 0$ into a larger set of equations of the form

$$[C_0 - zC_1]X(z) = 0, \tag{25}$$

where C_0 and C_1 are square scalar matrices and $X(z)$ is a column vector conforming to the dimensions of C_0 and C_1. The null values of $\mathcal{A}_C(z)$ will then be the same as the null values of $[C_0 - zC_1]$ and the null vectors of $\mathcal{A}_C(z)$ will be obtained by partitioning the null vectors of $[C_0 - zC_1]$. Provided that C_0^{-1} exists, the null values of $[C_0 - zC_1]$ are simply the inverses of the eigenvalues of the matrix $C_0^{-1}C_1$, and the null vectors of $\mathcal{A}_C(z)$ are simply the corresponding eigenvectors of $C_0^{-1}C_1$. Alternate procedures are readily specified for the case in which C_0^{-1} does not exist.

Let $\bar{I}_i$ be a square matrix formed by removing some rows and columns from the M-square identity matrix, we first define $\bar{I}_i$, $1 \leq i \leq N_L - 1$, to be the transformation matrices obtained by eliminating some of the redundant elements from $z^i\psi(z)$, $1 \leq i \leq N_L - 1$. With the above definitions, we now define

$$X(z) = \begin{bmatrix} \psi(z) \\ X_2(z) \\ X_3(z) \\ \cdots \\ X_{N_L-1}(z) \\ X_{N_L}(z) \end{bmatrix} = \begin{bmatrix} I & 0 & 0 & \cdots & 0 & 0 \\ z\bar{I}_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & z\bar{I}_2 & 0 & \cdots & 0 & 0 \\ \cdots \\ 0 & 0 & 0 & \cdots & z\bar{I}_{N_L-1} & 0 \end{bmatrix} \begin{bmatrix} \psi(z) \\ X_2(z) \\ X_3(z) \\ \cdots \\ X_{N_L-1}(z) \\ X_{N_L}(z) \end{bmatrix} \tag{26}$$

Based on the above definitions, it is then readily verified that

$$C_0 = \begin{bmatrix} \mathcal{A}_0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & I & 0 & \cdots & 0 & 0 \\ 0 & 0 & I & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & I & 0 \\ 0 & 0 & 0 & \cdots & 0 & I \end{bmatrix}, \tag{27}$$

and

$$C_1 = \begin{bmatrix} -\mathcal{A}_1 & -\bar{\mathcal{A}}_2 & -\bar{\mathcal{A}}_3 & \cdots & -\bar{\mathcal{A}}_{N_L-1} & -\bar{\mathcal{A}}_{N_L} \\ \bar{I}_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \bar{I}_2 & 0 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & \bar{I}_{N_L-1} & 0 \end{bmatrix}, \tag{28}$$

where

$$\bar{\mathcal{A}}_i = \mathcal{A}_i \prod_{j=1}^{i-1} \bar{I}_j{}^T \quad \text{for} \quad 2 \le i \le N_L, \tag{29}$$

and the superscript T denotes the transpose operator. We note that the dimension of the matrices C_0 and C_1 is K. Upon comparing the definitions of C_0 and $\mathcal{A}_0$, it is readily verified that C_0 is nonsingular if and only if $\mathcal{A}_0$ is nonsingular.

We offer the following proposition by the matrix generalization of Rouché's theorem (de Smit [1983]):

Proposition (5.1): The matrix $\mathcal{A}(z)$ has

(i) $M \times R - 1$ null values inside the unit circle of the complex plane,

(ii) at most $\sum_{i=0}^{R-1}(R-i)M_i - 1$ non-zero null values inside the unit circle of the complex plane,

(iii) a null value of one with multiplicity exactly one,

(iv) at most $\sum_{i=R+1}^{N}(i-R)M_i$ finite null values outside the unit circle of the complex plane.

As mentioned above, the null vectors and null values of $[C_0 - zC_1]$ are readily obtained from the eigenvalues and eigenvectors of $C_0^{-1}C_1$, which may be obtained by using standard library routines. Let

$$\sigma \in \mathcal{S} = \left\{\sigma|\det\left[\sigma I - C_0^{-1}C_1\right] = 0\right\},$$

denote an eigenvalue of $C_0^{-1}C_1$, and $W(\sigma)$ its corresponding eigenvector. Also, let

$$\mathcal{S}^{[0,1)} = \{\sigma \in \mathcal{S}, |\sigma| \in [0,1)\} \quad \text{and} \quad \mathcal{S}^{(1,\infty)} = \{\sigma \in \mathcal{S}, |\sigma| \in (1,\infty)\}.$$

Then, as we have observed above, the null values of $[C_0 - zC_1]$ are given by $\mathcal{Z}^{(0,1)} \cup \mathcal{Z}^{(1,\infty)\cup\{\infty\}} \cup \{1\}$, where

$$\mathcal{Z}^{(0,1)} = \left\{ z = 1/\sigma | \sigma \in \mathcal{S}^{(1,\infty)} \right\} \quad \text{and} \quad \mathcal{Z}^{(1,\infty)\cup\{\infty\}} = \left\{ z = 1/\sigma | \sigma \in \mathcal{S}^{[0,1)} \right\}.$$

Additionally,

$$X(z) = W(\sigma) \quad \text{for} \quad z \in \mathcal{Z}^{(0,1)} \cup \mathcal{Z}^{(1,\infty)\cup\{\infty\}} \cup \{1\}.$$

Noting the latter, we shall use $X(z)$, $W(\sigma)$, and $X(\sigma)$ and $\psi(z)$ and $\psi(\sigma)$ interchangeably throughout the remainder of this chapter.

5.5 Resolution of the Unknown Probabilities

As we pointed out in the previous section, the probability generating function for the queue occupancy has a total of $\sum_{i=0}^{R-1} M_i$ unknown probabilities. We shall use the null values and null vectors determined in the previous section, plus some other conditions, to form a linear system of equations from which to determine the unknown probabilities.

For continuity, we repeat (11):

$$\mathcal{G}(z)\left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z)\right] = \mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z). \tag{11}$$

By substituting $\mathcal{A}_L(z)\mathcal{A}_C(z)\mathcal{A}_R(z)$ for $[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z)]$ into (11) and solving for $\mathcal{G}(z)$, we find

$$\mathcal{G}(z) = \mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z)\mathcal{A}_R^{-1}(z)\mathcal{A}_C^{-1}(z)\mathcal{A}_L^{-1}(z). \tag{32}$$

Since $\mathcal{A}_R(z)$ $\mathcal{A}_L(z)$ are diagonal, their inverses are trivially obtained. The inverse of $\mathcal{A}_C(z)$ is more difficult to obtain.

Let K', with $K' \leq K$, be the number of distinct eigenvalues and σ_i, $1 \leq i \leq K'$, denote an ordering of the eigenvalues of $C_0^{-1}C_1$. With $\sigma_i = 1/z_i$ and $W = [W'(\sigma_1) \quad W'(\sigma_2) \quad \cdots \quad W'(\sigma_K')]$, where $W'(\sigma_i)$, $1 \leq i \leq K'$, are rectangular matrices whose columns are the principal vectors (see, for example, Golub and Wilkinson [1976] for the definition of principal vectors) associated with the eigenvalues σ_i, $1 \leq i \leq K'$, respectively, we readily find that

$$WJ = C_0^{-1}C_1W, \tag{33}$$

where J is the Jordan canonical form of $C_0^{-1}C_1$. That is

$$J = \text{diag}\ (\ J(\sigma_1) \quad J(\sigma_2) \quad \cdots \quad J(\sigma_{K'})\ ),$$

and $J(\sigma_i)$ is the Jordan block associated with the eigenvalue σ_i.

It can easily be shown that $\mathcal{A}_C^{-1}(z)$ is equal to the upper left M-square submatrix of $(C_0 - zC_1)^{-1}$. A closed form expression in terms of the eigenvalues and the principal vectors of $C_0^{-1}C_1$ for $(C_0 - zC_1)^{-1}$ is provided in Gohberg [1982]. Gohberg's solution, however, requires the inversion of another polynomial matrix, and therefore is not suitable for numerical computation. In this chapter, we will present a numerical approach to finding the inverse of $\mathcal{A}_C(z)$.

In our numerical examples, we have found that all eigenvalues of $C_0^{-1}C_1$ except that at 0 have multiplicities of 1; we will make this assumption in the remainder of this chapter and obtain the inverse of $\mathcal{A}_C(z)$ by means of the partial fraction method. Define h to be the **index** of the eigenvalue at 0, that is, h is the dimension of the largest *elementary* Jordan block in $J(0)$. Using the fact that $\det \mathcal{A}_C = \det (C_0 - zC_1)$, we find by partial fraction

$$\mathcal{A}_C^{-1}(z) = (1/\det \mathcal{A}_C(z))\mathrm{adj}\ \mathcal{A}_C(z) = \sum_{i=0}^{h-1} D_i z^i + \sum_{\{j|\sigma_j \in \mathcal{S}^{(0,\infty)}\}} \frac{1}{1 - z\sigma_j} E_j, \tag{36}$$

where D_i, $0 \le i < h$, and E_j for $\{j|\sigma_j \in \mathcal{S}^{(0,\infty)}\}$ are M-square matrices of constant elements. Define $F_i = D_i + \sum_{\{j|\sigma_j \in \mathcal{S}^{(0,\infty)}\}} E_j\sigma_j^i$ for $0 \le i < h$. We obtain

$$\mathcal{A}_C^{-1}(z) = \sum_{i=0}^{h-1} F_i z^i + \sum_{\{j|\sigma_j \in \mathcal{S}^{(0,\infty)}\}} \frac{1}{1 - z\sigma_j} E_j \sigma_j^h z^h. \tag{37}$$

It can easily be shown that $F_0 = \mathcal{A}_0^{-1}$, $F_i = -\mathcal{A}_0^{-1}\left(\sum_{k=1}^{i} A_i F_{i-k}\right)$ for $0 < i < h$, and the columns of E_j for $\{j|\sigma_j \in \mathcal{S}^{(0,\infty)}\}$ are multiples of $\psi(\sigma_j)$ and hence can be represented by $\psi(\sigma_j)\xi^j$, where ξ^j for $\{j|\sigma_j \in \mathcal{S}^{(0,\infty)}\}$ are row vectors of dimension M. The elements of ξ^j can be obtained by first finding the corresponding E_j matrix and then dividing one element in each column of E_j by the corresponding element in $\psi(\sigma_j)$. The matrix principal part E_j is obtained by multiplying $(1 - \sigma_j z)$ to $\mathcal{A}_C^{-1}(z)$ and taking limits as $z \to 1/\sigma_j$. A numerical technique for computing E_j is discussed in Daigle [1991] for distinct null values (also see illustrative problem 5.3 in Brogan [1982]).

Upon substitution of (37) into (32) and replacement of E_j by $\psi(\sigma_j)\xi^j$, we find

$$\mathcal{F}_{\tilde{n}}(z) = \mathcal{G}(z)\mathrm{e} = \mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{n}}(z)\mathcal{A}_R^{-1}(z)\left(\sum_{i=0}^{h-1} F_i z^i + \sum_{\{j|\sigma_j \in \mathcal{S}^{(0,\infty)}\}} \frac{1}{1 - z\sigma_j}\psi(z_j)\xi^j\sigma_j^h z^h\right)\mathcal{A}_L^{-1}(z)\mathrm{e}. \tag{38}$$

Since $\mathcal{F}_{\tilde{n}}(z)$ is bounded within the unit circle, we readily see that

$$\begin{aligned} 0 &= \mathcal{B}(z_i)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z_i)\mathcal{A}_R^{-1}(z_i)\psi(z_i)\xi^i\sigma_i^h z^h \mathcal{A}_L^{-1}\mathbf{e} \\ &= \mathcal{B}(z_i)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z_i)\mathcal{A}_R^{-1}(z_i)\psi(z_i) \quad \text{for} \quad z_i \in \mathcal{Z}^{(0,1)}, \end{aligned} \tag{39}$$

where we have used the fact that $\psi(z_i) = \psi(\sigma_i)$ for $\sigma_i = 1/z_i$, $z_i \in \mathcal{Z}^{(0,1)}$. But, since

$$\left[z_i^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z_i)\right] \mathcal{A}_R^{-1}(z_i)\psi(z_i) = 0$$

it follows that $z_i^R \mathcal{A}_R^{-1}(z_i)\psi(z_i) = \mathcal{P}\mathcal{F}_{\tilde{a}}(z)\mathcal{A}_R^{-1}(z_i)\psi(z_i)$ so that (39) becomes

$$0 = \mathcal{B}(z_i)\Phi\mathcal{A}_R^{-1}(z_i)\psi(z_i) \quad \text{for} \quad z_i \in \mathcal{Z}^{(0,1)}. \tag{40}$$

Upon substitution of the definitions of $\mathcal{B}(z_i)$and Φ, (40) becomes

$$0 = \sum_{j=0}^{R-1}(z_i^R - z_i^j)\sum_{\ell=0}^{j}\frac{1}{z_i^\ell}\Phi_{j\ell}\psi_\ell(z_i) \quad \text{for} \quad z_i \in \mathcal{Z}^{(0,1)}. \tag{41}$$

Since there are at most $\sum_{i=0}^{R-1}(R-i)M_i - 1$ elements of $\mathcal{Z}^{(0,1)}$, at least one additional equation is required to completely specify the unknown probabilities.

An additional equation is based on the null value of $\mathcal{A}(z)$ at $z = 1$. Differentiating both sides of (11) yields

$$\mathcal{G}'(z)\left[z^R I - \mathcal{P}\mathcal{F}_{\tilde{a}}(z)\right] + \mathcal{G}(z)\left[Rz^{R-1}I - \mathcal{P}\mathcal{F}_{\tilde{a}}'(z)\right] = \mathcal{B}'(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z) + \mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}'(z).$$

Upon postmultiplying both sides of the above equations by $\mathbf{e}$ and taking limits as $z \to 1$, we find

$$R - \mathcal{G}(1)\mathcal{F}_{\tilde{a}}'(1)\mathbf{e} = \mathcal{B}'(1)\Phi\mathbf{e}, \tag{42}$$

where we have used the fact that $\mathcal{F}_{\tilde{a}}(1)$ is an identity matrix so that $\mathcal{F}_{\tilde{a}}(1)\mathbf{e} = \mathbf{e}$ and that $\mathcal{G}(1)$ is the stationary probability vector for $\mathcal{P}$ so that $\mathcal{G}(1)\mathcal{P} = \mathcal{G}(1)$ and $\mathcal{G}(1)\mathbf{e} = 1$. Again, since $\mathcal{G}(1)$ is the stationary probability vector for $\mathcal{P}$, the quantity $\mathcal{G}(1)\mathcal{F}_{\tilde{a}}'(1)\mathbf{e}$ represents the overall average number of arrivals per slot in stochastic equilibrium. We therefore define

$$E[\tilde{a}] = \mathcal{G}(1)\mathcal{F}_{\tilde{a}}'(1)\mathbf{e}.$$

Finally, by substituting the appropriate definitions for $\mathcal{B}(z)$ and Φ into (42), we have

$$R - E[\tilde{a}] = \sum_{i=0}^{R-1}(R-i)\sum_{j=0}^{i}\Phi_{ij}. \tag{43}$$

This last equation then completes the specification of at most $\sum_{i=0}^{R-1}(R-i)M_i$ linear equations that are needed to solve for the unknown probabilities in the occupancy distribution probability generating function.

In cases where the total number of equations obtained from $\mathcal{Z}^{(0,1)} \cup \{1\}$ is less than $\sum_{i=0}^{R-1}(R-i)M_i$, additional equations are required. We note from the structure of $\tilde{\mathcal{P}}$ that $\phi_{n,m} = 0$ for $\{(n,m)|0 \leq n < N \text{and} \sum_{k=0}^{n} M_k < m \leq N\}$. A linear expression of $\phi_{n,m}$ in terms of the unknown $\sum_{i=0}^{R-1}(R-i)M_i$ probability masses can be obtained from (38). By equating the linear expressions of $\phi_{n,m}$ for $\{(n,m)|R \leq n < N \text{and} \sum_{k=0}^{n} M_k < m \leq N\}$ to 0, we can obtain $\sum_{i=R+1}^{N}(i-R)M_i$ additional equations. By adding these $\sum_{i=R+1}^{N}(i-R)M_i$ additional equations to the set of equations obtained above, we have more equations than unknown probability masses. Redundant equations are then eliminated by means of row reduction or other computation procedures (see Brogan [1982]).

Let us represent $\mathcal{P}\mathcal{F}_{\tilde{a}}(z)\mathcal{A}_R^{-1}(z)\left(\sum_{i=0}^{h-1} F_i z^i + \sum_{\{j|\sigma_j \in \mathcal{S}^{(0,\infty)}\}} \frac{1}{1-z\sigma_j}\psi(z_j)\xi^j\sigma_j^h z^h\right)\mathcal{A}_L^{-1}(z)$ by a matrix polynomial $\sum_{i=0}^{\infty} G_i z^i$, where G_i, $i \geq 0$, are M-square matrices. We can rewrite $\mathcal{G}(z)$ as follows

$$\mathcal{G}(z) = \left[-\sum_{i=0}^{R-1} \phi_i z^i + (\sum_{j=0}^{R-1} \phi_j) z^R\right]\left[\sum_{i=0}^{\infty} G_i z^i\right],$$

where we have substituted $\left[-\sum_{i=0}^{R-1} \phi_i z^i + (\sum_{j=0}^{R-1} \phi_j) z^R\right]$ for $\mathcal{B}(z)\Phi$.

Now, ϕ_n is the coefficient of the z^n term in $\mathcal{G}(z)$. For $n \geq R$, we have

$$\phi_n = \sum_{i=0}^{R-1} \phi_i (G_{n-R} - G_{n-i}).$$

Let G_i^m be the m-th column of G_i, we find

$$\phi_{n,m} = \sum_{i=0}^{R-1} \phi_i (G_{n-R}^m - G_{n-i}^m).$$

By equating $\phi_{n,m}$ for $\{(n,m)|R \leq n < N \text{and} \sum_{k=0}^{n} M_k < m \leq N\}$ to 0, we obtain a set of $\sum_{i=R+1}^{N}(i-R)M_i$ additional linear equations in terms of the unknown probability masses.

5.6 Inversion of the Probability Generating Function

The expression for the probability generating function for the occupancy distribution was presented in (38). In this section, we invert the probability generating function to obtain the occupancy probabilities. We rewrite (38).

$$\mathcal{F}_{\tilde{n}}(z) = \mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z)\mathcal{A}_R^{-1}(z)\left(\sum_{i=0}^{h-1} F_i z^i + \sum_{\{j|\sigma_j\in\mathcal{S}^{(0,\infty)}\}} \frac{1}{1-z\sigma_j}\psi(z_j)\xi^j\sigma_j^h z^h\right)\mathcal{A}_L^{-1}(z)\mathbf{e}. \tag{38}$$

A more convenient form for (38) may be obtained by recognizing that $\mathcal{B}(z)\Phi\mathcal{P}$ $\mathcal{F}_{\tilde{a}}(z)\mathcal{A}_R^{-1}(z)\left(\sum_{i=0}^{h-1} F_i z^i\right)\mathcal{A}_L^{-1}(z)\mathbf{e}$ and $\mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z)\mathcal{A}_R^{-1}(z)\left(\sum_{\{j|\sigma_j\in\mathcal{S}^{(0,\infty)}\}}\frac{1}{1-z\sigma_j}\psi(z_j)\xi^j\sigma_j^h z^h\right)$ $\mathcal{A}_L^{-1}(z)\mathbf{e}$ are simply polynomials. If we let $b_0(z)$ represent $\mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z)\mathcal{A}_R^{-1}(z)\left(\sum_{i=0}^{h-1} F_i z^i\right)$ $\mathcal{A}_L^{-1}(z)\mathbf{e}$ and $b_i(z)$ represent $\mathcal{B}(z)\Phi\mathcal{P}\mathcal{F}_{\tilde{a}}(z)\mathcal{A}_R^{-1}(z)\left(\sum_{\{j|\sigma_j\in\mathcal{S}^{(0,\infty)}\}}\frac{1}{1-z\sigma_j}\psi(z_j)\xi^j\sigma_j^h z^h\right)\mathcal{A}_L^{-1}(z)\mathbf{e}$, then we find

$$\mathcal{F}_{\tilde{n}}(z) = b_0(z) + \sum_{\{i|\sigma_i\in\mathcal{S}^{(0,\infty)}\}} \frac{1}{1-z\sigma_i} b_i(z). \tag{45}$$

If we denote the coefficient of z^j in $b_i(z)$ by b_{ij}, we find

$$b_i(z) = \sum_{j=0}^{N+h} b_{ij} z^j \quad \text{for} \quad \{i|\sigma_i \in \mathcal{S}^{[0,\infty)}\}. \tag{46}$$

To facilitate further discussion, we expand (45) according to the location of the elements of $\mathcal{S}$ as follows:

$$\mathcal{F}_{\tilde{n}}(z) = b_0(z) + \sum_{\{i|\sigma_i\in\mathcal{S}^{[1,\infty)}\}} \frac{1}{1-z\sigma_i} b_i(z) + \sum_{\{i|\sigma_i\in\mathcal{S}^{(0,1)}\}} \frac{1}{1-z\sigma_i} b_i(z). \tag{47}$$

Now, since $\mathcal{F}_{\tilde{n}}(z)$ contains no singularities in the unit disc of the complex plane, $(1-\sigma_i z)$ is a factor of $b_i(z)$ for $\sigma_i \in \mathcal{S}^{[1,\infty)}$. Therefore,

$$\sum_{\{i|\sigma_i\in\mathcal{S}^{[1,\infty)}\}} \frac{1}{1-z\sigma_i} b_i(z) = \sum_{\{i|\sigma_i\in\mathcal{S}^{[1,\infty)}\}} \gamma_i(z), \tag{48}$$

where

$$\gamma_i(z) = \frac{b_i(z)}{1-\sigma_i z} \quad \text{for} \quad \{i|\sigma_i \in \mathcal{S}^{[1,\infty)}\}, \tag{49}$$

and $\gamma_i(z)$ is a polynomial of order $N+h-1$. Finally,

$$\sum_{\{i|\sigma_i\in\mathcal{S}^{(0,1)}\}} \frac{1}{1-z\sigma_i} b_i(z) = \sum_{\{i|\sigma_i\in\mathcal{S}^{(0,1)}\}} \gamma_i(z) + z^{N+h} \sum_{\{i|\sigma_i\in\mathcal{S}^{(0,1)}\}} \frac{r_i}{1-\sigma_i z}, \tag{50}$$

where again $\gamma_i(z)$ is a polynomial of order $N+h-1$, and

$$r_i = \sigma_i^{N+h} b_i(\sigma_i^{-1}). \tag{51}$$

If we write

$$\gamma_i(z) = \sum_{j=0}^{N+h-1} \gamma_{ij} z^j, \tag{52}$$

it is readily seen that

$$\gamma_{ij} = \sum_{k=0}^{j} b_{ik} \sigma_i^{j-k}, \tag{53}$$

and

$$r_i = \sum_{k=0}^{N+h} b_{ik} \sigma_i^{N+h-k}. \tag{54}$$

Upon substitution of (48)-(54) into (47), we find

$$\mathcal{F}_{\tilde{n}}(z) = \sum_{j=0}^{N+h-1} z^j \left[b_{0j} + \sum_{\{i|\sigma_i \in \mathcal{S}^{(0,\infty)}\}} \sum_{k=0}^{j} b_{ik}\sigma_i^{j-k} \right] + z^{N+h} \sum_{\{i|\sigma_i \in \mathcal{S}^{(0,1)}\}} \sum_{k=0}^{N+h} b_{ik}\sigma_i^{N+h-k} \frac{1}{1-\sigma_i z}. \tag{55}$$

The occupancy probabilities, $P\{\tilde{n} = n\}$ are now easily obtained from (55) by simply taking the coefficient of z^n. We find

$$P\{\tilde{n} = n\} = \begin{cases} b_{0n} + \sum_{\{i|\sigma_i \in \mathcal{S}^{(0,\infty)}\}} \sum_{k=0}^{n} b_{ik}\sigma_i^{n-k} & \text{for } n < N+h \\ \sum_{\{i|\sigma_i \in \mathcal{S}^{(0,1)}\}} \sigma_i^{n-N-h} \sum_{k=0}^{N} b_{ik}\sigma_i^{N+h-k} & \text{for } n \geq N+h. \end{cases} \tag{56}$$

This concludes our specification of the occupancy probability masses. A procedure to obtain the joint phase-occupancy probability masses could have been similarly specified. The primary difference in the approach is that all operations would have been performed without postmultiplication by $\mathbf{e}$ with the result that the $b_i(z)$ and $\gamma_i(z)$ would have been vectors of polynomials rather than scalar polynomials.

The mean occupancy can be obtained by differentiating (55) and taking limits as $z \to 1$, that is $E[\tilde{n}] = \mathcal{F}'_{\tilde{n}}(1)$.

5.7 Computation of the Delay Distribution

Once the occupancy probabilities are found, computation of the delay mass probabilities is straightforward. Define ϕ_n to be the $N+1$ order row vector whose m-th element is the stationary joint probability such that the number of packets in the system is n and the phase process is in state m, and $\mathbf{a}_k = \mathcal{P}P_k\mathbf{e}$ to be the $N+1$ order column vector whose m-th element is the transition probability such that the number of arrivals in the $(j+1)$-st time slot is k, given the phase process at the end of the j-th time slot is m. The expected number of arrivals which are delayed for d units of time is then given by

$$E[\tilde{a},d] = \sum_{n=0}^{R} \phi_n \left[\sum_{k=(d-1)R+1}^{\min\{dR,N\}} \mathrm{a}_k(k-(d-1)R) + \sum_{k=\min\{dR,N\}+1}^{N} \mathrm{a}_k R \right] \quad \text{for} \quad d=1, \tag{57}$$

$$E[\tilde{a},d] = RHS(57) + \sum_{n=(d-1)R+1}^{dR-1} \phi_n \left[\sum_{k=1}^{dR-n} \mathrm{a}_k k + \sum_{k=dR-n+1}^{N} \mathrm{a}_k(dR-n) \right] \quad \text{for} \quad d=2, \tag{58}$$

$$E[\tilde{a},d] = RHS(58) + \sum_{n=R+1}^{(d-1)R} \phi_n \left[\sum_{k=(d-1)R-n+1}^{\min\{dR-n,N\}} \mathrm{a}_k(k+n-(d-1)R) + \sum_{k=\min\{dR-n,N\}+1}^{N} \mathrm{a}_k R \right] \quad \text{for} \quad d \geq 3, \tag{59}$$

where RHS(i) represent the right hand side of equation (i).

Let $\tilde{d}$ denote the time of delay, the delay probabilities are readily obtained by

$$P\{\tilde{d}=d\} = \frac{E[\tilde{a},d]}{E[\tilde{a}]} \quad \text{for} \quad d=1,2,3,\cdots. \tag{60}$$

Here $E[\tilde{a}]$ is the expected number of arrivals given by

$$E[\tilde{a}] = \nu \left[\sum_{k=1}^{N} \mathrm{a}_k k \right].$$

where ν is the row vector of stationary phase probabilities obtained from $\nu\mathcal{P} = \nu$, and $\nu\mathbf{e} = 1$.

The mean delay can be computed using Little's result

$$E[\tilde{d}] = \frac{E[\tilde{n}]}{E[\tilde{a}]} \tag{61}$$

or directly from the delay probability masses

$$E[\tilde{d}] = 1 + \sum_{d=1}^{\infty} \left(1 - \sum_{i=1}^{d} P\{\tilde{d}=i\} \right). \tag{62}$$

The extra 1 unit of delay on the right hand side of (62) comes from the fact that newly arrived packets must wait for the departing packets to be cleared before they can be processed.

5.8 Numerical Examples

In this section, we present numerical examples of stationary distributions and mean values for occupancy and delay. In particular, there is a finite number, N, of identical sources, and each source generates a packet every T units of time during active periods. The lengths of messages in units of packets are geometrically distributed with parameter $(1-\alpha)$ and the lengths of idle periods in units of time are geometrically distributed with parameter $(1-\beta)$. That is, each input source evolves in accordance with the $T+1$ state Markov chain

$$\begin{bmatrix} \beta & 1-\beta & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 \\ 1-\alpha & \alpha & 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}.$$

The expected length of a message is then given by $(1-\alpha)^{-1}$ and the expected length of an idle period by $(1-\beta)^{-1}$.

Denote the expected length of a message and the expected length of an idle period by EM and EP respectively, and the traffic intensity by ρ. From the theory of alternating renewal process (Ross [1983, 85]), we obtain the traffic intensity by

$$\rho = \frac{T \times EM}{T \times EM + EP} \times \frac{N}{T \times R}. \tag{63}$$

We note that $T \times EM$ is the expected length of an active period.

Denote a state of the aggregate phase process by m. A state of the aggregate phase process corresponds to a phase vector $\underline{m} = (m_1 m_2 \cdots m_{T+1})$, where m_k, $1 \le k \le T+1$, is the number of sources in state k of the individual phase process. Let i and k be respectively the *from* state and the *to* state in an aggregate phase transition, and $\underline{i}$ and $\underline{k}$ the corresponding phase vectors. The phase transition probabilities are then given by

$$P\{\tilde{m}(j+1) = k | \tilde{m}(j) = i\} = \\ \sum_{r=(k_2-i_1)^+}^{\min\{i_{T+1},k_2\}} \binom{i_{T+1}}{r} \binom{i_1}{k_2-r} \alpha^r (1-\alpha)^{i_{T+1}-r} \beta^{i_1-k_2+r} (1-\beta)^{k_2-r}, \tag{64}$$

for $k_r = i_{r-1}$, $3 \le r \le T+1$, and $i_1 + i_{T+1} = k_1 + k_2$; and 0 otherwise.

Figures 5.1 and 5.2 provides an example of the type of numerical results that may be obtained from our method. The input parameters for the system are $T = 2$, $EM = 8$, and $\rho = 0.9$. Figure 5.1 shows the complementary occupancy distribution function for several values of R, which represents the number of slots, with the number of users, N, fixed at 8. Figure 5.2 shows the complementary delay distribution for several values of R.

In addition to the results shown in Figures 5.1 and 5.2, we ran numerical examples for 6 users with the same values for T, EM and ρ, and the results are presented in Tables 5.1 and 5.2. Table 5.1 presents the mean values for 6 users and Table 5.2 for 8 users. In all these tables, the traffic intensity or utilization is given by $E[\tilde{a}]/R$, the mean occupancy is computed from $\mathcal{F}'_{\tilde{n}}(1)$, the absolute error which measures the degree to which the resulting probabilities sum to unity is obtained by $|1.0 - \mathcal{F}_{\tilde{n}}(1)|$, and the mean delay is calculated from Little's result. From the tables, it is readily seen that our techniques yield extremely accurate results.

5.9 Conclusions

We demonstrate in this chapter the feasibility of our approach through the solution of an important subclass of the class of queueing systems we have in mind, namely a R server queue with batch Markovian arrival process (BMAP). In particular, we address the class of communications models in which packets are gernerated by each individual sources every T units of time during active periods. We obtain both distributions and mean values for occupancy and delay. Our methodology is computationally stable and yields extremely accurate results.

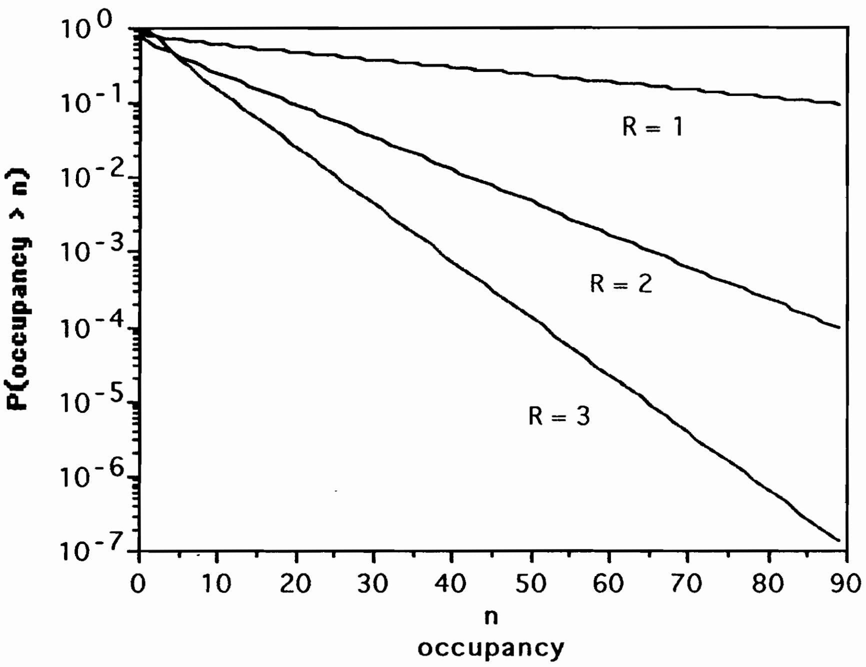

Figure 5.1. Residual distribution functions of occupancy (speed conversion 1:2, N = 8).

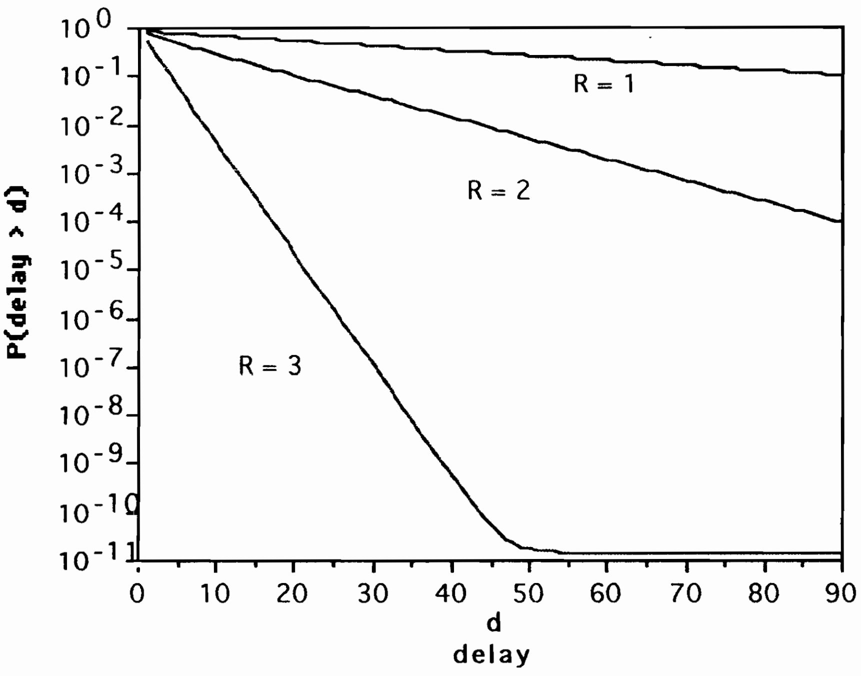

Figure 5.2. Residual distribution funtions of delay (speed conversion 1:2, N = 8).

Table 5.1.

Mean occupancy, mean delay and absolute error for a 1:2 speed-conversion multiplexer with 6 sources for several values of R.

R	Utilization %	Mean Occupancy	Mean Delay	Absolute Error
1	90.0	2.693575e+1	2.992861e+1	7.041893e-17
2	90.0	8.010574e+0	4.450319e+0	3.827776e-16
3	90.0	3.032765e+0	1.123246e+0	4.753330e-08

Table 5.2.

Mean occupancy, mean delay and absolute error for a 1:2 speed-conversion multiplexer with 8 sources for several values of R.

R	Utilization %	Mean Occupancy	Mean Delay	Absolute Error
1	90.0	3.357426e+1	3.730474e+1	5.854692e-17
2	90.0	1.510279e+1	8.390437e+0	2.093260e-12
3	90.0	6.029186e+0	2.233032e+0	4.953675e-12

CHAPTER 6
ADDITIONAL MODELS

6.1 Introduction

The behavior of many physical systems is well modeled by a queueing system in which time is slotted, the distribution of the number of entitites that arrive during a slot is dependent upon the evolution of a discrete time, discrete state Markov chain, and the number of entities that may be served during a slot is limited to some number. This new class of queueing models is known as queues of the $M/G/R$ type, whereas the kind of arrival process mentioned above is known as the batch Markovian arrival process or BMAP (Lucantoni [1991]). In the previous two chapters, we developed a straightforward algebraic algorithmic methodology for analyzing the behavior of queueing systems of this sort, and presented detailed descriptions for obtaining certain important performance measures for two models of a very important subclass in the telecommunications area. In this chapter, we will demonstrate the versatility of our methodology by outlining the general approach for obtaining the performance measures of several additional queueing models.

Reservoir or dam models are perhaps the earliest class of queueing models that are amenable to analyses along the lines of queues of the $M/G/R$ type. Lloyd [1963], in recognition of the limitations of the reservoir theory of independent and identically distributed inflows developed by Moran [1954], extends the theory by taking account of the serial correlation in the sequence of inflows. He approximates the structure of the correlated sequence by a Morkov chain with a finite number of states and works with a bivariate Markov process involving the joint distribution of levels and inflows. Lloyd outlines the general form of the transition matrix for Markov inputs and Moran withdrawals. Odoom and Lloyd [1965] obtains the equilibrium probability of emptiness and the generating function of the distribution of levels for a semi-infinite reservoir subject to Markovian inputs and unit withdrawals. Ali Khan and Gani [1968] outline the time-dependent solution for the same models as in Odoom and Lloyd [1965] and show that their solution approaches that of Odoom and Lloyd at equilibrium. Finite dams with inputs forming a Markov chain is presented by Ali Khan [1970]. He derives the probabilities of first emptiness before overflow and with overflow. An equation for the probability that the reservoir ever dries up before overflow and a time dependent formula for the probability distribution of the dam content are also obtained. The probability generating function for the distribution of the time to first emptiness for semi-infinite storage systems with a Markov chain input and unit outflow is obtained by Herbert [1975]. Herbert also considers the probability that the dam ever empties and computes the mean time to first emptiness.

The two-state breakdown model presented in Towsley [1980] is an excellent example of those queueing models with non-deterministic servers to which our methodology can still be applied. In this paper, a statistical multuplexer which operates in a two state Markovian environment is analyzed and the resulting generating function for the equilibrium queue length distribution is obtained. The breakdown model is used to analyze a multiplexer with a head of the line fixed priority premptive queueing discipline.

The meteor burst communications system suggested in Robert, Mitrani and King [1988] present another queueing model with intermittent service which is amenable to analysis by our methodology. The meteor burst communications system is a rather unusual communications system which makes use of the reflective power of the ionization layers in the earth's atmosphere created by showers of meteors for the transmission of data. Robert, Mitran and King construct a probabilistic model with a two-dimensional Markov chain for obtaining the steady-state joint distribution of the number of packets awaiting transmission at a station and the remaining duration of the current operative or inoperative period. Chandramouli, Neuts and Ramaswami [1989] take account of different stages in transmission as well as data loss to construct a finite state Markov chain model for the analysis of the metor burst communications system. They analyze the model using the matrix analytical method (Neuts [1989] etc.) developed for the analysis of queues of the $M/G/1$ type.

A queueing model with multiple Markovian arrival and service processes is considered in Li [1990] as an extention to the analysis of a ATM multiplexer whose arrival process is the sum of a finite number of two-state Markov chains. A simple Kronecker product technique is used to decompose the analysis of the queueing system into an evaluation of individual sources.

The organization of this chapter is now described. In the next section, we will outline the approach to obtaining the probability generating function of occupancy for the general $M/G/R$ queue. In Section 3, we will describe in detail the procedures in constructing sets of linear equations for finding the unknown boundary probabilities for the dam/storage model, the breakdown model of Towsley, and the meteor burst communications model. We also discuss an approach for finding the probability generating function of occupancy for a queueing model with multiple Markovian arrival and service processes. In Section 4, we draw conclusions.

6.2. The Approach

In this section, we will outline an approach for obtaining the probability generating function of occupancy for the general $M/G/R$ queue.

Let $\tilde{n}$ and $\tilde{m}$ denote, respectively, the number of entities in the system and the state of the *phase process* at equilibrium, and let

$$\phi_{n,m} = P\{\tilde{n} = n, \tilde{m} = m\}. \tag{1}$$

Define ϕ_n to be the row vector of probabilities such that the number of entities in the system is n at equilibrium. That is,

$$\phi_n = [\phi_{n,1} \quad \phi_{n,2} \quad \cdots \quad \phi_{n,M}], \tag{2}$$

where M is the number of phases in the phase process.

Denote the one-step transition probability matrix by $\tilde{\mathcal{P}}$. Then, at equilibrium, $\tilde{\Phi} = \tilde{\Phi}\tilde{\mathcal{P}}$, where

$$\tilde{\Phi} = [\phi_0 \quad \phi_1 \quad \phi_2 \quad \phi_3 \quad \cdots \quad \cdots],$$

$$\tilde{\mathcal{P}} = \begin{bmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} & \cdots & \cdots \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ B_{R-1,0} & B_{R-1,1} & B_{R-1,2} & B_{R-1,3} & \cdots & \cdots \\ A_0 & A_1 & A_2 & A_3 & \cdots & \cdots \\ 0 & A_0 & A_1 & A_2 & \cdots & \cdots \\ 0 & 0 & A_0 & A_1 & \cdots & \cdots \\ 0 & 0 & 0 & A_0 & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & \cdots \end{bmatrix}.$$

A_i, for $i = 0, 1, \cdots$, and $B_{i,j}$, for $i = 0, 1, \cdots$, $j = 0, 1, \cdots$, are M-square matrices of probability masses. In detail, $\tilde{\Phi} = \tilde{\Phi}\tilde{\mathcal{P}}$ can be rewritten in the form

$$\phi_j = \sum_{i=0}^{R-1} \phi_i B_{i,j} + \sum_{i=R}^{j+R} \phi_i A_{j+R-i}, \quad \text{for} \quad j = 0, 1, \cdots. \tag{3}$$

Now, let $\mathcal{G}(z) = \sum_{j=0}^{\infty} \phi_j z^j$, $A(z) = \sum_{j=0}^{\infty} A_j z^j$, and $B_i(z) = \sum_{j=0}^{\infty} B_{i,j} z^j$. Upon substituting these expressions into (3), we have

$$\mathcal{G}(z)\left[z^R I - A(z)\right] = \sum_{i=0}^{R-1} \phi_i \left[z^R B_i(z) - z^i A(z)\right]. \tag{4}$$

where ϕ_i, for $i = 0, 1, \cdots, R-1$, are vectors of unknown probabilities.

The probability generating function of occupancy, $\mathcal{F}_{\tilde{n}}(z)$, is given by $\mathcal{G}(z)\mathbf{e}$, where $\mathbf{e}$ is the column vector in which each element is unity. That is

$$\mathcal{F}_{\tilde{n}}(z) = \frac{1}{\det\left[z^R I - A(z)\right]} \sum_{i=o}^{R-1} \phi_i \left[z^R B_i(z) - z^i A(z)\right] \operatorname{adj}\left[z^R I - A(z)\right]\mathbf{e}. \tag{5}$$

Define the null values and vectors of a matrix, $\mathcal{A}(z)$, as the set of values and vectors, $\{z_i\}$ and $\{X_{z_i}\}$, such that $\mathcal{A}(z_i)X_{z_i} = 0$ with X_{z_i} nontrivial. The inverse of $\left[z^R I - A(z)\right]$, then, can be expressed in terms of its null values and vectors. In order to find the null values and vectors of $\left[z^R I - A(z)\right]$, we decompose $\left[z^R I - A(z)\right]$ into three M-square polynomial matrices, that is $\left[z^R I - A(z)\right] = \mathcal{A}_L \mathcal{A}_C \mathcal{A}_R$, where $\mathcal{A}_L$ and $\mathcal{A}_R$ are respectively the products of some elementary matrices and diagonal matrices of z powers. In other words, $\mathcal{A}_L$ and $\mathcal{A}_R$ together contain all the trivial null values of $\left[z^R I - A(z)\right]$ whereas $\mathcal{A}_C$ contains the rest.

Let $\mathcal{Z}$ denote the set of null values of $\left[z^R I - A(z)\right]$. That is,

$$\mathcal{Z} = \left\{z \middle| \det\left[z^R I - A(z)\right] = 0\right\}.$$

Now, for $z^* \in \mathcal{Z}$, let $\psi^*(z^*)$ be the null vector of $\left[z^R I - A(z)\right]$ corresponding to the null value z^* so that

$$\mathcal{A}_L(z^*)\mathcal{A}_C(z^*)\mathcal{A}_R(z^*)\psi^*(z^*) = 0.$$

For $z^* \neq 0$, define $\psi(z^*) = \mathcal{A}_R(z^*)\psi^*(z^*)$. We then find

$$\mathcal{A}_C(z^*)\psi(z^*) = 0. \tag{6}$$

Since $\det\left[z^R I - A(z)\right]$ if and only if $z = 0$, once having determined $\psi(z^*)$, we may calculate $\psi^*(z^*) = \mathcal{A}_R(z^*)^{-1}\psi(z^*)$ for $z^* \neq 0$.

In order to find the null vectors of $\mathcal{A}_C(z)$, we convert the set of equations $\mathcal{A}_C(z)\psi(z) = 0$ into a larger set of equations of the form

$$[C_0 - zC_1]X(z) = 0,$$

where C_0 and C_1 are square scalar matrices and $X(z)$ is a column vector conforming to the dimensions of C_0 and C_1.

The null values of $\mathcal{A}_C(z)$ will then be the same as the null values of $[C_0 - zC_1]$ and the null vectors of $\mathcal{A}_C(z)$ will be obtained by partitioning the null vectors of $[C_0 - zC_1]$. Provided that C_0^{-1} exists, the null values of $[C_0 - zC_1]$ are simply the inverses of the eigenvalues of the matrix $C_0^{-1}C_1$, and the null vectors of $\mathcal{A}_C(z)$ are simply the corresponding eigenvectors of $C_0^{-1}C_1$.

Denote $\mathcal{A}_C(z) = \sum_{i=0}^{N_L} \mathcal{A}_i z^i$. Let $\bar{I}_i$ be a square matrix formed by removing some rows and columns from the M-square identity matrix, we first define $\bar{I}_i$, $1 \le i \le N_L - 1$, to be the transformation matrices obtained by eliminating some of the redundant elements from $z^i v(z)$, $1 \le i \le N_L - 1$. With the above definitions, we now define

$$X(z) = \begin{bmatrix} v(z) \\ X_2(z) \\ X_3(z) \\ \cdots \\ X_{N_L-1}(z) \\ X_{N_L}(z) \end{bmatrix} = \begin{bmatrix} I & 0 & 0 & \cdots & 0 & 0 \\ z\bar{I}_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & z\bar{I}_2 & 0 & \cdots & 0 & 0 \\ \cdots \\ 0 & 0 & 0 & \cdots & z\bar{I}_{N_L-1} & 0 \end{bmatrix} \begin{bmatrix} \psi(z) \\ X_2(z) \\ X_3(z) \\ \cdots \\ X_{N_L-1}(z) \\ X_{N_L}(z) \end{bmatrix} \tag{7}$$

It is then readily verified that

$$C_0 = \begin{bmatrix} \mathcal{A}_0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & I & 0 & \cdots & 0 & 0 \\ 0 & 0 & I & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & I & 0 \\ 0 & 0 & 0 & \cdots & 0 & I \end{bmatrix}, \tag{8}$$

and

$$C_1 = \begin{bmatrix} -\mathcal{A}_1 & -\bar{\mathcal{A}}_2 & -\bar{\mathcal{A}}_3 & \cdots & -\bar{\mathcal{A}}_{N_L-1} & -\bar{\mathcal{A}}_{N_L} \\ \bar{I}_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \bar{I}_2 & 0 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & \bar{I}_{N_L-1} & 0 \end{bmatrix}, \tag{9}$$

where

$$\bar{\mathcal{A}}_i = \mathcal{A}_i \prod_{j=1}^{i-1} \bar{I}_j{}^T \quad \text{for} \quad 2 \le i \le N_L,$$

and the superscript T denotes the transpose operator.

We note that the dimension of the matrices C_0 and C_1 is K. Upon comparing the definitions of C_0 and $\mathcal{A}_0$, it is readily verified that C_0 is nonsingular if and only if $\mathcal{A}_0$ is nonsingular.

As mentioned above, the null vectors and null values of $[C_0 - zC_1]$ are readily obtained from the eigenvalues and eigenvectors of $C_0^{-1}C_1$, which may be obtained by using standard library routines. Let

$$\sigma \in \mathcal{S} = \left\{\sigma | \det \left[\sigma I - C_0^{-1}C_1\right] = 0\right\}.$$

denote an eigenvalue of $C_0^{-1}C_1$, and $W(\sigma)$ its corresponding eigenvector. Also, let

$$\mathcal{S}^{[0,1)} = \{\sigma \in \mathcal{S}, |\sigma| \in [0,1)\} \quad \text{and} \quad \mathcal{S}^{(1,\infty)} = \{\sigma \in \mathcal{S}, |\sigma| \in (1,\infty)\}.$$

Then, as we have observed above, the null values of $[C_0 - zC_1]$ are given by $\mathcal{Z}^{(0,1)} \cup \mathcal{Z}^{(1,\infty)\cup\{\infty\}} \cup \{1\}$, where

$$\mathcal{Z}^{(0,1)} = \left\{ z = 1/\sigma | \sigma \in \mathcal{S}^{(1,\infty)} \right\} \quad \text{and} \quad \mathcal{Z}^{(1,\infty)\cup\{\infty\}} = \left\{ z = 1/\sigma | \sigma \in \mathcal{S}^{[0,1)} \right\}.$$

Additionally,

$$X(z) = W(\sigma) \quad \text{for} \quad z \in \mathcal{Z}^{(0,1)} \cup \mathcal{Z}^{(1,\infty)\cup\{\infty\}} \cup \{1\}.$$

Let K', with $K' \leq K$, be the number of distinct eigenvalues and σ_i, $1 \leq i \leq K'$, denote an ordering of the eigenvalues of $C_0^{-1}C_1$. With $\sigma_i = 1/z_i$ and $W = [\,W'(\sigma_1) \quad W'(\sigma_2) \quad \cdots \quad W'(\sigma'_K)]$, where $W'(\sigma_i)$, $1 \leq i \leq K'$, are rectangular matrices whose columns are the principal vectors (see, for example, Golub and Wilkinson [1976] for the definition of principal vectors) associated with the eigenvalues σ_i, $1 \leq i \leq K'$, respectively, we readily find that

$$WJ = C_0^{-1}C_1W, \tag{10}$$

where J is the Jordan canonical form of $C_0^{-1}C_1$. That is

$$J = \text{diag}\,(\,J(\sigma_1) \quad J(\sigma_2) \quad \cdots \quad J(\sigma_K)\,),$$

and $J(\sigma_i)$ is the Jordan block associated with the eigenvalue σ_i.

It can be easily shown that $\mathcal{A}_C^{-1}(z)$ is equal to the upper left M-square submatrix of $(C_0 - zC_1)^{-1}$. In practice, all eigenvalues of $C_0^{-1}C_1$ except that at 0 have multiplicities of 1; we will make this assumption in the remainder of this chapter and obtain the inverse of $\mathcal{A}_C(z)$ by means of the partial fraction method.

Define h to be the **index** of the eigenvalue at 0, that is, h is the dimension of the largest *elementary* Jordan block in $J(0)$. Using the fact that $\det \mathcal{A}_C(z) = \det\,(C_0 - zC_1)$, we find by partial fraction

$$\mathcal{A}_C^{-1}(z) = (1/\det \mathcal{A}_C(z))\text{adj } \mathcal{A}_C(z) = \sum_{i=0}^{h-1} D_i z^i + \sum_{\{j|\sigma_j \in \mathcal{S}^{(0,\infty)}\}} \frac{1}{1 - z\sigma_j} E_j, \tag{11}$$

where D_i, $0 \leq i < h$, and E_j for $j|\sigma_j \in \mathcal{S}^{(0,\infty)}$ are M-square matrices of constant elements.

Define $F_i = D_i + \sum_{\{j|\sigma_j \in \mathcal{S}^{(0,\infty)}\}} E_j \sigma_j^i$ for $0 \leq i < h$. We obtain

$$\mathcal{A}_C^{-1}(z) = \sum_{i=0}^{h-1} F_i z^i + \sum_{\{j|\sigma_j \in \mathcal{S}^{(0,\infty)}\}} \frac{1}{1 - z\sigma_j} E_j \sigma_j^h z^h. \tag{12}$$

It can easily be shown that $F_0 = \mathcal{A}_0^{-1}$, $F_i = -\mathcal{A}_0^{-1}\left(\sum_{k=1}^{i} \mathcal{A}_i F_{i-k}\right)$ for $0 < i < h$, and that the columns of E_j for $\{j|\sigma_j \in \mathcal{S}^{(0,\infty)}\}$ are multiples of $\psi(\sigma_j)$ and hence can be represented by $\psi(\sigma_j)\xi^j$, where ξ^j for $\{j|\sigma_j \in \mathcal{S}^{(0,\infty)}\}$ are row vectors of dimension M. The elements of ξ^j can be obtained by first finding the corresponding E_j matrix and then dividing one element in each column of E_j by the corresponding element in $\psi(\sigma_j)$. The matrix principal part E_j is obtained by multiplying $(1 - \sigma_j z)$ to $(1/\det \mathcal{A}_C(z))\text{adj}\,\mathcal{A}_C(z)$ and taking limits as $z \to 1/\sigma_j$. A numerical technique for computing E_j is discussed in Daigle [1991] for distinct null values.

With the definftions above, equation (5) can be rewritten in the form

$$\mathcal{F}_{\tilde{n}}(z) = \sum_{i=o}^{R-1} \phi_i \left[z^R B_i(z) - z^i A(z)\right] \mathcal{A}_R^{-1}(z)$$
$$\times \left(\sum_{i=0}^{h-1} F_i z^i + \sum_{\{j|\sigma_j \in \mathcal{S}^{(0,\infty)}\}} \frac{1}{1 - z\sigma_j} \psi(z_j)\xi^j \sigma_j^h z^h \right) \mathcal{A}_L^{-1}(z)\mathrm{e}, \tag{13}$$

where $\mathcal{F}_{\tilde{n}}(z)$ is the probability generating function for the occupancy distribution. The occupancy probabilities, $P\{\tilde{n} = n\}$, $n = 0, 1, \cdots$, are now easily obtained by expanding (13) and using the coefficients of z^n, $n = 0, 1, \cdots$.

We note that there are $M \times R$ unknown probabilities, that is $\phi_{n,m}$ for $0 \le n \le R-1$ and $1 \le m \le M$, in (13). Some of these unknown probabilities are readily obtainable from (3), while others must be obtained through the resolution of a set of linear equations. In the remainder of this section, we will obtain some of the linear equations using the null values and vectors of $\mathcal{A}_C(z)$. We will set up the rest of the set of linear equations for individual models in the next section.

The probability generating function, $\mathcal{F}_{\tilde{n}}(z)$, is bounded within the unit cirle of the complex plane. From (13), we readily see that

$$0 = \sum_{i=o}^{R-1} \phi_i \left[z_k^R B_i(z_k) - z_k^i A(z_k)\right] \mathcal{A}_R^{-1}(z_k)\psi(z_k)\xi^k \sigma_k^h z_k^h \mathcal{A}_L^{-1}\mathrm{e}$$
$$= \sum_{i=o}^{R-1} \phi_i \left[z_k^R B_i(z_k) - z_k^i A(z_k)\right] \mathcal{A}_R^{-1}(z_k)\psi(z_k) \quad \text{for} \quad z_k \in \mathcal{Z}^{(0,1)}, \tag{14}$$

where we have used the fact that $\psi(z_k) = \psi(\sigma_k)$ for $\sigma_k = 1/z_k$, $z_k \in \mathcal{Z}^{(0,1)}$. But, since

$$\left[z_k^R I - A(z_k)\right] \mathcal{A}_R^{-1}(z_k)\psi(z_k) = 0$$

it follows that $z_k^R \mathcal{A}_R^{-1}(z_k)\psi(z_k) = A(z_k)\mathcal{A}_R^{-1}(z_k)\psi(z_k)$ so that (14) becomes

$$0 = \sum_{i=o}^{R-1} \phi_i \left[z_k^R B_i(z_k) - z_k^{R+i} I\right] \mathcal{A}_R^{-1}(z_k)\psi(z_k)$$

$$= \sum_{i=o}^{R-1} \phi_i \left[B_i(z_k) - z_k^i I\right] \mathcal{A}_R^{-1}(z_k)\psi(z_k) \quad \text{for} \quad z_k \in \mathcal{Z}^{(0,1)}, \tag{15}$$

Equation (15) gives us at most $R \times M - 1$ equations; the proof of this statement can be obtained from a matrix generalization of Rouché's theorem (see de Smit [1983]). One additional equation can be obtained from the null value at $z = 1$.

Differentiating both sides of (4) yields

$$\mathcal{G}'(z)\left[z^R I - A(z)\right] + \mathcal{G}(z)\left[Rz^{R-1} I - A'(z)\right]$$
$$= \sum_{i=0}^{R-1} \phi_i [Rz^{R-1} B_i(z) - iz^{i-1} A(z)] + \sum_{i=0}^{R-1} \phi_i [z^R B_i'(z) - z^i A'(z)].$$

Upon postmultiplying both sides of the above equations by $\mathbf{e}$ and taking limits as $z \to 1$, we find

$$R - \mathcal{G}(1)A'(1)\mathbf{e} = \sum_{i=0}^{R-1}(R-i)\phi_i \mathbf{e} + \sum_{i=0}^{R-1} \phi_i[B_i'(1) - A'(1)]\mathbf{e}, \tag{16}$$

where we have used the fact that $A(1)\mathbf{e} = \mathbf{e}$, $B(1)\mathbf{e} = \mathbf{e}$ and $\mathcal{G}(1)\mathbf{e} = 1$. Again, since $\mathcal{G}(1)$ is the stationary probability vector for the phase process, the quantity $\mathcal{G}(1)A'(1)\mathbf{e}$ represents the overall average number of arrivals per slot in stochastic equilibrium. We therefore define

$$E[\tilde{a}] = \mathcal{G}(1)A'(1)\mathbf{e}.$$

Finally, by substituting the definition of $E[\tilde{a}]$ into (16), we have

$$R - E[\tilde{a}] = \sum_{i=0}^{R-1}(R-i)\phi_i \mathbf{e} + \sum_{i=0}^{R-1} \phi_i[B_i'(1) - A'(1)]\mathbf{e}. \tag{17}$$

This last equation then completes the specification of at most $R \times M$ linear equations for the resolution of the unknown probabilitieses in the probability generating function for the occupancy distribution. In general, the number of equations obtained from the null values and vectors of $\mathcal{A}_{\mathcal{C}}$ is less than the number of unknown probabilities in $\mathcal{F}_{\tilde{n}}(z)$. Additional equations are obtained in Section 3.

6.3. The Models

In the previous section, we set up a linear system of equations for the resolution of the unknown probabilities for use in the probabblity generating function of the occupancy distribution using the fact that $\mathcal{F}_{\tilde{n}}(x)$ is bounded within the unit circle of the complex plane. The number of equations thus obtained, however, is usually less than the number of unknowns. In this section, we continue setting up additional equations for the resolution of the unknown probabilities using special relationships afforded by the algebraic structure of each specific model.

6.3.1. Infinite Dam/Storage Model with Modulated Arrivals

The dam is considered at equally spaced times and the amount of water removed per time slot is assumed to be constant at R units. The inflows into the dam form a M-state Markov chain $\{\tilde{a}(j), j \geq 1\}$ with transition probability matrix $\mathcal{P}$. That is

$$\mathcal{P} = \begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0N} \\ p_{10} & p_{11} & \cdots & p_{1N} \\ \cdots & \cdots & \cdots & \cdots \\ p_{N0} & p_{N1} & \cdots & p_{NN} \end{bmatrix}, \tag{18}$$

where $N = M - 1$ is the maximum amount of infl w into the dam per unit of time, and $p_{ik} = P\{\tilde{a}(j+1) = k | \tilde{a}(j) = i\}$.

The contents of the dam forms a stochastic process $\{\tilde{n}(j), j \geq 0\}$ by virtue of the relation

$$\tilde{n}(j+1) = (\tilde{n}(j) + \tilde{a}(j+1) - R)^{+},$$

where $(\cdot)^{+} = \max\{\cdot, 0\}$. The process $\{(\tilde{n}(j), \tilde{a}(j), j \geq 0\}$ is a vector, discrete valued, discrete parameter Markov chain on the state space $\{(n, m), n \geq 0, 0 \leq m \leq N\}$. This system is readily analyzed in a manner analogous to the *M/D/R* paradigm.

At stochastic equilibrium, let $\phi_{n,m} = P\{\tilde{n} = n, \tilde{a} = m\}$. Define ϕ_n to be the row vector of probabilities such that the amount of water in the dam is n units at equilibrium. That is, $\phi_n = [\phi_{n,0} \quad \phi_{n,1} \quad \cdots, \phi_{n,N}]$. Also, define A_k, $k = 0, 2, \cdots, N$, to be a M-square matrix whose elements are zeros except that its kth column is the kth column of $\mathcal{P}$. That is

$$A_k = \begin{bmatrix} 0 & \cdots & 0 & p_{0k} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & p_{1k} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & 0 & p_{Nk} & 0 & \cdots & 0 \end{bmatrix}. \tag{19}$$

Then $\tilde{\Phi} = \tilde{\Phi}\tilde{\mathcal{P}}$ and $\tilde{\Phi}\mathbf{e} = 1$, where

$$\tilde{\Phi} = [\phi_0 \quad \phi_1 \quad \phi_2 \quad \phi_3 \quad \cdots \quad \cdots],$$

$$\tilde{\mathcal{P}} = \begin{bmatrix} \sum_{k=0}^{R} A_k & A_{R+1} & A_{R+2} & A_{R+3} & \cdots & \cdots \\ \sum_{k=0}^{R-1} A_k & A_R & A_{R+1} & A_{R+2} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ A_0 + A_1 & A_2 & A_3 & A_4 & \cdots & \cdots \\ A_0 & A_1 & A_2 & A_3 & \cdots & \cdots \\ 0 & A_0 & A_1 & A_2 & \cdots & \cdots \\ 0 & 0 & A_0 & A_1 & \cdots & \cdots \\ 0 & 0 & 0 & A_0 & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & \cdots \end{bmatrix},$$

and $\tilde{\mathcal{P}}$ is the one-step transition probability matrix. Upon substitution of appropriate definitions, the probability generating function of occupancy, $\mathcal{F}_{\tilde{n}}(z)$, is then given by

$$\mathcal{F}_{\tilde{n}}(z) = \sum_{i=o}^{R-1} \phi_i \left[z^R B_i(z) - z^i A(z)\right] \mathcal{A}_R^{-1}(z)$$
$$\times \left(\sum_{i=0}^{h-1} F_i z^i + \sum_{\{j|\sigma_j \in \mathcal{S}^{(0,\infty)}\}} \frac{1}{1 - z\sigma_j} \psi(z_j) \xi^j \sigma_j^h z^h \right) \mathcal{A}_L^{-1}(z)\mathbf{e}, \qquad (13)$$

where $B_i(z) = \sum_{k=0}^{R-i} A_k + \sum_{j=1}^{\infty} A_{R-i+j} z^j$ for $i = 0, 1, \cdots, R-1$.

We notice from $\tilde{\mathcal{P}}$ that the right-hand-side of (13) has a total of $(1/2)R(3R+1)$ unknown probabilitites, whereas $[z^R I - A(z)]$ can provide only $(1/2)R(R+1)$ linear equations from its null values $z \in \mathcal{Z}^{(0,1)} \cup \{1\}$. The remaining $R \times R$ equations required for the resolution of the unknown probabilities can be obtained directly from $\tilde{\mathcal{P}}$. They are

$$\phi_{0,k} = \left(\sum_{i=0}^{R-k} \phi_i \right) \mathcal{P}_k, \quad \text{for} \quad k = 1, 2, \cdots, R,$$
$$\phi_{1,k} = \phi_{R+1-k} \mathcal{P}_k, \quad \text{for} \quad k = 2, 3, \cdots, R+1,$$
$$\cdots\cdots\cdots$$
$$\phi_{R-1,k} = \phi_{2R-1-k} \mathcal{P}_k, \quad \text{for} \quad k = R, R+1, \cdots, 2R-1, \qquad (20)$$

where $\mathcal{P}_i$, $i = 0, 1, \cdots, N$, is the ith column of $\mathcal{P}$.

6.3.2. Towsley's Model with Server Breakdown

Towsley's breakdown model is a statistical multiplexer which operates in a two state Markovian environment. The multiplexer transmits one unit of data, ***packet***, during one time slot when it is in the operative state. The environment state of the multiplexer, $\{\tilde{m}(j), j \geq 0\}$, is a Markov chain with transition probability $p_{ik} = P\{\tilde{m}(j+1) = k | \tilde{m}(j) = i\}$.

Let $\tilde{a}_m$, independent of time, denote the number of packets arriving during one time slot in which the environment is in state m, $m = 0, 1$. In addition, let $P_{m,i} = P\{\tilde{a}_m = i\}$, and

$$\mathcal{F}_{\tilde{a}_m}(z) = \sum_{i=0}^{\infty} z^i P\{\tilde{a}_m = i\}.$$

Define q_m, $m = 0, 1$, to be the probability that a packet cannot be transmitted during a slot while the environment is in state m. and $\bar{q}_m = 1 - q_m$ the probability of successful transmission. Define the function

$$\mathcal{F}_{\bar{q}_m}(z) = \bar{q}_m + q_m z.$$

Denote the number of packets in the buffer at the beginning of the $(j+1)$-st time slot by $\tilde{n}(j)$, and the number of arrivals during the $(j+1)$-st time slot by $\tilde{a}(j+1)$. The stochastic process $\{\tilde{n}(j), j \geq 0\}$ is governed by the relation

$$\tilde{n}(j+1) = (\tilde{n}(j) - R)^{+} + \tilde{a}(j+1),$$

where $(\cdot)^{+} = \max\{\cdot, 0\}$. The state of the system is represented by the vector, discrete valued, discrete parameter Markov process $\{(\tilde{n}(j), \tilde{m}(j), j \geq 0\}$ on the state space $\{(n,m), n \geq 0, m = 0, 1\}$. The one-step transition probability matrix is then given by

$$\tilde{\mathcal{P}} = \begin{bmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} & \cdots & \cdots \\ A_0 & A_1 & A_2 & A_3 & \cdots & \cdots \\ 0 & A_0 & A_1 & A_2 & \cdots & \cdots \\ 0 & 0 & A_0 & A_1 & \cdots & \cdots \\ 0 & 0 & 0 & A_0 & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & \cdots \end{bmatrix},$$

where

$$B_{0,k} = \begin{bmatrix} P_{0,k}p_{00} & P_{0,k}p_{01} \\ P_{1,k}p_{10} & P_{1,k}p_{11} \end{bmatrix} = \begin{bmatrix} P_{0,k} & 0 \\ 0 & P_{1,k} \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix}, \quad \text{for} \quad k \geq 0,$$

$$A_0 = \begin{bmatrix} (P_{0,0}\bar{q}_0)p_{00} & (P_{0,0}\bar{q}_0)p_{01} \\ (P_{1,0}\bar{q}_1)p_{10} & (P_{1,0}\bar{q}_1)p_{11} \end{bmatrix} = \begin{bmatrix} P_{0,0}\bar{q}_0 & 0 \\ 0 & P_{1,0}\bar{q}_1) \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix},$$

and

$$A_k = \begin{bmatrix} (P_{0,k}\bar{q}_0 + P_{0,k-1}q_0)p_{00} & (P_{0,k}\bar{q}_0 + P_{0,k-1}q_0)p_{01} \\ (P_{1,k}\bar{q}_1 + P_{1,k-1}q_1)p_{10} & (P_{1,k}\bar{q}_1 + P_{1,k-1}q_1)p_{11} \end{bmatrix}$$
$$= \begin{bmatrix} P_{0,k}\bar{q}_0 + P_{0,k-1}q_0 & 0 \\ 0 & P_{1,k}\bar{q}_1 + P_{1,k-1}q_1 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix}, \quad \text{for} \quad k \geq 1.$$

Let $A(z) = \sum_{j=0}^{\infty} A_j z^j$, and $B_i(z) = \sum_{j=0}^{\infty} B_{i,j} z^j$, we have

$$B_0(z) = \begin{bmatrix} \mathcal{F}_{\tilde{a}_0}(z) & 0 \\ 0 & \mathcal{F}_{\tilde{a}_1}(z) \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix},$$

$$A(z) = \begin{bmatrix} \mathcal{F}_{\tilde{a}_0}(z)\mathcal{F}_{\tilde{q}_0}(z) & 0 \\ 0 & \mathcal{F}_{\tilde{a}_1}(z)\mathcal{F}_{\tilde{q}_1}(z) \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix},$$

and

$$zB_0(z) - A(z) = \begin{bmatrix} \mathcal{F}_{\tilde{a}_0}(z)(z - \mathcal{F}_{\tilde{q}_0}(z)) & 0 \\ 0 & \mathcal{F}_{\tilde{a}_1}(z)(z - \mathcal{F}_{\tilde{q}_1}(z)) \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix}. \quad (21)$$

Now, Towsley's multiplexer breaks down at state 1, that is $q_1 = 0$. Therefore, $\mathcal{F}_{\tilde{q}_1}(z) = \bar{q}_1 + q_1 z = z$. Consequently,

$$zB_0(z) - A(z) = \begin{bmatrix} \mathcal{F}_{\tilde{a}_0}(z)(z - \mathcal{F}_{\tilde{q}_0}(z)) & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix}. \quad (22)$$

Let $\phi_{n,m} = P\{\tilde{n} = n, \tilde{m} = m\}$ at stochastic equilibrium and substitute (22) into (13), we have

$$\mathcal{F}_{\tilde{n}}(z) = [\,\phi_{0,0} \quad \phi_{0,1}\,]\,[zB_0(z) - A(z)]\,\mathcal{A}_R^{-1}(z) \times \left(\sum_{i=0}^{h-1} F_i z^i + \sum_{\{j|\sigma_j \in \mathcal{S}(0,\infty)\}} \frac{1}{1 - z\sigma_j} \psi(z_j)\xi^j \sigma_j^h z^h \right) \mathcal{A}_L^{-1}(z)\mathrm{e}. \tag{23}$$

We note that there are two unkown probabilities in (23). But since all the elements in the second row of $zB_0(z) - A(z)$ are 0, the unknown probability, $\phi_{0,1}$, disappears from the probability generating function for the occupancy distribution. Therefore, we can substitute $\phi_{0,1}$ with 0 in (23) and still be able to find the occupancy distribution.

The unknown probability, $\phi_{0,0}$, is determined by (16). That is

$$1 - \pi_0(E[\tilde{a}_0] + q_0) - \pi_1(E[\tilde{a}_1] + q_1) = \phi_{0,0}\bar{q}_0, \tag{24}$$

where

$$\pi_m = p_{\bar{m}m}/(p_{01} + p_{10}), \quad \text{for} \quad m = 0, 1,$$

$$\bar{m} = 1 - m, \quad \text{for} \quad m = 0, 1,$$

and

$$E[\tilde{a}_m] = \mathcal{F}'_{\tilde{a}_m}(1), \quad \text{for} \quad m = 0, 1.$$

We now extend Towsley's model to considering a multiplexer in which R packets are transmitted during one time slot. The one-step transition probability matrix is then

$$\tilde{\mathcal{P}} = \begin{bmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} & \cdots & \cdots \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ B_{R-1,0} & B_{R-1,1} & B_{R-1,2} & B_{R-1,3} & \cdots & \cdots \\ A_0 & A_1 & A_2 & A_3 & \cdots & \cdots \\ 0 & A_0 & A_1 & A_2 & \cdots & \cdots \\ 0 & 0 & A_0 & A_1 & \cdots & \cdots \\ 0 & 0 & 0 & A_0 & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & \cdots \end{bmatrix},$$

where

$$B_{i,k} = \begin{bmatrix} P_{0,k}\bar{q}_0 + P_{0,k-i}q_0 & 0 \\ 0 & P_{1,k}\bar{q}_1 + P_{1,k-i}q_1 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix}, \quad \text{for} \quad i = 1, 2, \cdots, R-1, \quad k \geq 0,$$

and

$$A_k = \begin{bmatrix} P_{0,k}\bar{q}_0 + P_{0,k-R}q_0 & 0 \\ 0 & P_{1,k}\bar{q}_1 + P_{1,k-R}q_1 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix}, \quad \text{for} \quad k \geq 1.$$

We have used the fact that $P_{i,k} = 0$ for $k < 0$ above. Defining $A(z) = \sum_{j=0}^{\infty} A_j z^j$, and $B_i(z) = \sum_{j=0}^{\infty} B_{i,j} z^j$, we find

$$B_i(z) = \begin{bmatrix} \mathcal{F}_{\bar{a}_0}(z)(\bar{q}_0 + q_0 z^i) & 0 \\ 0 & \mathcal{F}_{\bar{a}_1}(z)(\bar{q}_1 + q_1 z^i) \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix}, \quad \text{for} \quad i = 1, 2, \cdots, R-1,$$

$$A(z) = \begin{bmatrix} \mathcal{F}_{\bar{a}_0}(z)(\bar{q}_0 + q_0 z^R) & 0 \\ 0 & \mathcal{F}_{\bar{a}_1}(z)(\bar{q}_1 + q_1 z^R) \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix},$$

and

$$z^R B_i(z) - z^i A(z) = \begin{bmatrix} \mathcal{F}_{\bar{a}_0}(z)(z^r - z^i)\bar{q}_0 & 0 \\ 0 & \mathcal{F}_{\bar{a}_1}(z)(z^r - z^i)\bar{q}_1 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix}. \tag{25}$$

Now, Towsley's multiplexer breaks down at state 1, that is $\bar{q}_1 = 0$. From (25), we have

$$z^R B_i(z) - z^i A(z) = \begin{bmatrix} \mathcal{F}_{\bar{a}_0}(z)(z^r - z^i)\bar{q}_0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix}. \tag{26}$$

The probability generating function for the occupancy distribution is given by (13). We note that there are $2R$ unkown probabilities in $\mathcal{F}_{\bar{n}}(z)$ for this special case. But since all the elements in the second row of $z^R B_i(z) - z^i A(z)$, $i = 0, 1, \cdots, R-1$, are 0, the unknown probabilities, $\phi_{i,1}$, $i = 0, 1, \cdots, R-1$, disappear completely from $\mathcal{F}_{\bar{n}}(z)$. Therefore, we can substitute $\phi_{i,1}$, $i = 0, 1, \cdots, R-1$, with 0 in (13) and still be able to find the occupancy distribution. The unknown probabilities, $\phi_{i,0}$, $i = 0, 1, \cdots, R-1$, are then determined by the null values of $[z^R I - A(z)]$ in $\mathcal{Z}^{[0,1)} \cup \{1\}$.

From the definition of $A(z)$, we have

$$z^R I - A(z) = \begin{bmatrix} z^R & 0 \\ 0 & z^R \end{bmatrix} - \begin{bmatrix} \mathcal{F}_{\bar{a}_0}(z)(\bar{q}_0 + q_0 z^R) & 0 \\ 0 & \mathcal{F}_{\bar{a}_1}(z)(\bar{q}_1 + q_1 z^R) \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix}. \tag{27}$$

Upon substituting $\bar{q}_1 = 0$ into (27), we have

$$z^R I - A(z) = \begin{bmatrix} 1 & 0 \\ 0 & z^R \end{bmatrix} \begin{bmatrix} z^R - \mathcal{F}_{\bar{a}_0}(z)(\bar{q}_0 + q_0 z^R)p_{00} & -\mathcal{F}_{\bar{a}_0}(z)(\bar{q}_0 + q_0 z^R)p_{01} \\ -\mathcal{F}_{\bar{a}_1}(z)p_{10} & 1 - \mathcal{F}_{\bar{a}_1}(z)p_{11} \end{bmatrix}. \tag{28}$$

With R trivial null values factored out, $[z^R I - A(z)]$ provides R null values in $\mathcal{Z}^{(0,1)} \cup \{1\}$ for the resolution of the R unknown probabilities, $\phi_{i,0}$, $i = 0, 1, \cdots, R-1$.

6.3.3. A Meteor Burst Communications Model

The meteor burst communications system transmits data via an intermittently available ionization layer created by the arrival of meteorite showers in the upper atmosphere. The communications link, therefore, alternates between an operative period and an inoperative period which are of $\tilde{m}_Q$ and $\tilde{m}_S$ time slots respectively. During inoperative periods, the receiving station generates a probing signal, which when received by the sender, indicates the arrival of operative periods. The sending station starts sending R packets per time slot until acknowlegement signals are no longer received. This indicates that the reflecting layer has become inoperative, and packets sent during the last M_E time slots are lost and must be retransmitted in the next operative period.

Assume that a packet sent during the first $\tilde{m}_T = \tilde{m}_Q - M_E$ slots of an operative period has a probaility $\bar{e} = (1-r)^d$ of being received, where r is the bit error rate and d is the number of bits in a packet. The inoperative period has a *phase type distribution* $PH(\beta, S)$. That is, the inoperative periods are i.i.d. with probability mass

$$P\{\tilde{m}_S = k\} = \beta S^{k-1} S^0, \quad \text{for} \quad k \geq 1.$$

Such a distribution is obtained as the distribution of the absorption time in a discrete Markov chain with initial probability vector $(\, \beta \quad 0 \,)$ and transition matrix $\begin{bmatrix} S & S^0 \\ 0 & 1 \end{bmatrix}$, where $S^0 = \mathbf{e} - S\mathbf{e}$, and $\mathbf{e}$ is the column vector in which all elements are unity. The operative periods are also i.i.d. and have probability mass

$$P\{\tilde{m}_Q = k\} = \begin{cases} b_k & \text{for} \quad k \leq M_E \\ \bar{b}_{M_E} \alpha T^{i-1} T^0 & \text{for} \quad k = M_E + i, \quad i \geq 1, \end{cases}$$

where b_k, $k = 1, 2, \cdots, M_E$, and $\bar{b}_{M_E} = 1 - \sum_{i=1}^{M_E} b_i$ are nonnegative probabilities, the length of the operative period in excess of M_E slots has distribution $PH(\alpha, T)$, and $T^0 = \mathbf{e} - T\mathbf{e}$.

The states, $\{\tilde{m}(j), j \geq 0\}$, of the communications medium at successive epochs are described by an irreducible $M = M_T + M_E + M_S$ state, where $M_T + M_E$ is the total number of operative states and M_S is the number of inoperative states. Markov chain with transition probability matrix

$$\mathcal{P} = \begin{bmatrix} T & T^0 & 0 & 0 \\ 0 & 0 & I_{M_E - 1} & 0 \\ 0 & 0 & 0 & \beta \\ \bar{b}_{M_E} S^0 \alpha & b_{M_E} S^0 & S^0 b & S \end{bmatrix}. \tag{29}$$

where I_k denote the k-square identity matrix, and $b = (\, b_{M_E - 1} \quad \cdots \quad b_1 \,)$ is a vector of nonnegative probability masses.

Suppose that the number of packets arriving in successfive time interval $[j, j+1)$ is i.i.d. with common distribution $\{P\{\tilde{a} = i\} = P_i, i \geq 0\}$ and probability generating function

$$\mathcal{F}_{\tilde{a}}(z) = \sum_{i=0}^{\infty} z^i P\{\tilde{a} = i\}.$$

Define $\tilde{n}(j)$ as the number of packets in the system at time j^+, and $\tilde{m}(j)$ the state of the medium at time j^-. The process $\{(\tilde{n}(j), \tilde{m}(j), j \geq 0\}$ is a discrete time Markov chain with state space $\{(n,m), n \geq 0, m = 1, 2, \cdots, M\}$ and transition probability matrix

$$\tilde{\mathcal{P}} = \begin{bmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ B_{R-1,0} & B_{R-1,1} & B_{R-1,2} & B_{R-1,3} & \cdots & \cdots \\ A_0 & A_1 & A_2 & A_3 & \cdots & \cdots \\ 0 & A_0 & A_1 & A_2 & \cdots & \cdots \\ 0 & 0 & A_0 & A_1 & \cdots & \cdots \\ 0 & 0 & 0 & A_0 & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & \cdots \end{bmatrix},$$

where

$$B_{i,k} = \mathcal{P}\begin{bmatrix} \sum_{j=0}^{i} c_{i,j} P_{k-j} I_{M_T} & 0 \\ 0 & P_{k-i} I_{M_E+M_S} \end{bmatrix}, \quad \text{for} \quad i = 0, 1, \cdots, R-1, \quad k \geq 0,$$

$$A_k = \mathcal{P}\begin{bmatrix} \sum_{j=0}^{R} c_{R,j} P_{k-j} I_{M_T} & 0 \\ 0 & P_{k-R} I_{M_E+M_S} \end{bmatrix}, \quad \text{for} \quad k \geq 0,$$

$$c_{i,j} = \binom{i}{j} \bar{e}^{i-j}(1-\bar{e})^j, \quad \text{for} \quad i = 0, 1, \cdots, R-1, \quad 0 \leq j \leq i,$$

and

$$P_k = 0, \quad \text{for} \quad k < 0.$$

Let $\phi_{n,m} = P\{\tilde{n} = n, \tilde{m} = m\}$, $\phi_n = (\,\phi_{n,1} \quad \phi_{n,2} \quad \cdots \quad \phi_{n,M}\,)$, $\mathcal{G}(z) = \sum_{n=0}^{\infty} \phi_n z^n$, $A(z) = \sum_{k=0}^{\infty} A_k z^k$, and $B_i(z) = \sum_{k=0}^{\infty} B_{i,k} z^k$. We find

$$\mathcal{F}_{\tilde{n}}(z) = \sum_{i=o}^{R-1} \phi_i \left[z^R B_i(z) - z^i A(z)\right] \mathcal{A}_R^{-1}(z)$$
$$\times \left(\sum_{i=0}^{h-1} F_i z^i + \sum_{\{j | \sigma_j \in \mathcal{S}(0,\infty)\}} \frac{1}{1 - z\sigma_j} \psi(z_j) \xi^j \sigma_j^h z^h \right) \mathcal{A}_L^{-1}(z) \mathbf{e}, \tag{13}$$

where $\mathcal{F}_{\tilde{n}}(z) = \sum_{i=0}^{\infty} z^i P\{\tilde{n} = i\}$ is the probability generating function for the occupancy distribution.

By using the definition of $A(z)$, we have

$$z^R - A(z) = \left(\begin{bmatrix} z^R I_{M_T} & 0 \\ 0 & I_{M_E+M_S} \end{bmatrix} - \mathcal{P} \begin{bmatrix} \mathcal{F}_{\tilde{a}}(z) \;{}_{j=0}^{R} c_{R,j} z^j I_{M_T} & 0 \\ 0 & \mathcal{F}_{\tilde{a}}(z) I_{M_E+M_S} \end{bmatrix} \right)$$
$$\times \begin{bmatrix} I_{M_T} & 0 \\ 0 & z^R I_{M_E+M_S} \end{bmatrix}. \tag{30}$$

The polynomial matrix $\left[z^R - A(z)\right]$ provides $R \times M_T$ nontrivial null values for the resolution of the $R \times M$ unknown probabilities in (13). Notice that column $M_T + 1$

through column M in A_k, $k = 0, 1, \cdots, R-1$, are all columns of trivial elements. Let $\tilde{\Phi} = [o_0 \quad o_1 \quad o_2 \quad \cdots \quad \cdots]$. From $\tilde{\Phi} = \tilde{\Phi}\tilde{\mathcal{P}}$, we obtain the remaining $R \times (M_E + M_S)$ equations using the equations for ϕ_{k,M_T+i}, $k = 0, 1, \cdots, R-1$ and $i = 1, 2, \cdots, M_E + M_S$, in terms of o_k, $k = 0, 1, \cdots, R-1$.

6.3.4. A Queueing Model with Multiple Markovian Arrivals and Servers

We consider a queueing system in which both the arrival and service processes are composed of multiple Markovian processes. Let $\tilde{n}(j)$ be the number of entities in the system at the beginning of the j-th time slot, $\tilde{b}(j)$ the number of arrivals during the j-th time slot, and $\tilde{c}(j)$ the number of entities served during the j-th time slot. The occupancy of the queueing system evolves according to the following relation

$$\tilde{n}(j+1) = \left(\tilde{n}(j) + \tilde{b}(j) - \tilde{c}(j)\right)^+, \tag{31}$$

where $(\cdot)^+ = \max\{\cdot, 0\}$.

Let R be the maximum number of entities that the server can serve during one time slot. We convert the service process into a pseudo-arrival process $\tilde{d}(j) = R - c(j)$. Upon substituting this relation into (31), we get

$$\tilde{n}(j+1) = \left(\tilde{n}(j) + \tilde{b}(j) + \tilde{d}(j) - R\right)^+. \tag{32}$$

Now $\tilde{b}(j)$ is composed of a number of arrival streams, $\tilde{b}(j) = \sum_k \tilde{b}^k(j)$, and $\tilde{c}(j)$ is of several small servers, $\tilde{c}(j) = \sum_k \tilde{c}^k(j)$. Let R^k be the maximum number of entities that the k-th small server can serve during one time slot. We have $\tilde{d}(j) = \sum_k \tilde{d}^k(j) = \sum_k R^k - \tilde{c}^k(j)$. Defining $\tilde{a}(j) = \sum_k \tilde{a}^k(j) = \sum_l \tilde{b}^l(j) + \sum_i \tilde{d}^i(j)$, we have

$$\tilde{n}(j+1) = (\tilde{n}(j) + \tilde{a}(j) - R)^+. \tag{33}$$

Let $\tilde{m}(j)$ denote the state of the phase process at the beginning of the j-th time slot. The process $\{(\tilde{n}(j), \tilde{m}(j)), j \geq 0\}$ is a vector, discrete valued, discrete parameter Markov chain on the state space $\{(n, m), n \geq 0, m = 1, 2, \cdots, M\}$, where M is the total number of states in the Markov chain, and the one-step transition probability matrix for the Markov chain is given by

$$\tilde{\mathcal{P}} = \begin{bmatrix} \sum_{i=0}^{R} A_i & A_{R+1} & A_{R+2} & A_{R+3} & \cdots & \cdots \\ \sum_{i=0}^{R-1} A_i & A_R & A_{R+1} & A_{R+2} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ A_0 + A_1 & A_2 & A_3 & A_4 & \cdots & \cdots \\ A_0 & A_1 & A_2 & A_3 & \cdots & \cdots \\ 0 & A_0 & A_1 & A_2 & \cdots & \cdots \\ 0 & 0 & A_0 & A_1 & \cdots & \cdots \\ 0 & 0 & 0 & A_0 & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & \cdots \end{bmatrix},$$

where A_i, $i \geq 0$, is a M-square matrix of probability masses for $\tilde{a}(j) = i$. Now, let

$$B_i(z) = \sum_{j=0}^{R-1-i} A_j + \sum_{k=0}^{\infty} A_{R-i+k} z^k. \tag{34}$$

The probabability generating function for the occupancy distribution is then given by (13). Upon substituting (34) into (13), we have

$$\mathcal{F}_{\tilde{n}}(z) = \sum_{i=o}^{R-1} \phi_i \left[\sum_{k=0}^{R-1-i} A_k z^k (z^{R-k} - z^i) \right] \mathcal{A}_R^{-1}(z) \\ \times \left(\sum_{i=0}^{h-1} F_i z^i + \sum_{\{j|\sigma_j \in \mathcal{S}^{(0,\infty)}\}} \frac{1}{1 - z\sigma_j} \psi(z_j) \xi^j \sigma_j^h z^h \right) \mathcal{A}_L^{-1}(z) \mathrm{e}. \tag{35}$$

Let $\tilde{n}$ and $\tilde{m}$ denote, respectively, the number of entities in the system and the state of the phase process at equilibrium, $\phi_{n,m} = P\{\tilde{n} = n, \tilde{m} = m\}$, $\phi_n = [\phi_{n,1} \quad \phi_{n,2} \quad \cdots \quad \phi_{n,M}]$, and $\tilde{\Phi} = [\phi_0 \quad \phi_1 \quad \phi_2 \quad \cdots \quad \cdots]$. Then, at stochastic equilibrium, $\tilde{\Phi} = \tilde{\Phi}\tilde{\mathcal{P}}$. Let $\tilde{a}$ represent the arrival process at equilibrium and $A(z) = \sum_{i=0}^{\infty} A_i z^i$ denote the probability generating matrix for $\tilde{a}$. Now, $\tilde{a} = \sum_k \tilde{a}^k$ at equilibrium. It is easy to verify that

$$A(z) = \bigotimes_k A^k(z) = A^1(z) \otimes A^2(z) \otimes \cdots, \tag{36}$$

where $A^k(z)$ is the probability generating matrix for $\tilde{a}^k$, and $\otimes$ is the matrix Kronecker product operator.

As an example, we consider the special case in which there is a one-to-one correspondence between the phase of the arrival process and the number entities that arrive during a slot. Denote the phase transition probability of the k-th arrival stream by $\mathcal{P}^k$ and the number of phases by M^k. Define $A^k(z) = \Delta_{M^k}(z^i)\mathcal{P}^k$, where $\Delta_m(x_i)$ is a $m \times m$ diagonal matrix in which the i-th diagonal element is x_i for $i = 0, 1, \cdots, m-1$. Let $\underline{j}$ be a row vector in which the k-th element is j_k, and $|j| = \underline{j}\mathrm{e} = \sum_k j_k$. Define the $\underline{j}$-th element of the row vector $\bar{\mathcal{B}}(z)$ to be

$$\bar{\mathcal{B}}_{\underline{j}}(z) = \begin{cases} \sum_{i=0}^{R-1-|j|} \phi_{i,\underline{j}}(z^{R-|j|} - z^i) & \text{for } 0 \leq |j| < R \\ 0 & \text{otherwise.} \end{cases} \tag{37}$$

Denoting the probability generating function of $\tilde{n}$ by $\mathcal{F}_{\tilde{n}}(z)$, we have

$$\mathcal{F}_{\tilde{n}}(z) = \bar{\mathcal{B}}(z) A(z) \mathcal{A}_R^{-1}(z) \left(\sum_{i=0}^{h-1} F_i z^i + \sum_{\{j|\sigma_j \in \mathcal{S}^{(0,\infty)}\}} \frac{1}{1 - z\sigma_j} \psi(z_j) \xi^j \sigma_j^h z^h \right) \mathcal{A}_L^{-1}(z) \mathrm{e}. \tag{38}$$

Equation (38) is equivalent to equation (2.20) in Li [1990]. The unknown probabilities, $\phi_{i,\underline{j}}$ for $0 \leq |j| < R$ and $0 \leq i \leq R-1-|j|$, are then solved using the set of linear equations developed in Section 2.

6.4. Conclusions

In this chapter, we have outlined a general approach for obtaining the probability generating function of occupancy for additional queueing models of the $M/G/R$ type. In Sections 2, we discussed the general approach for constructing a set of linear equations using the null values and vectors of $[z^R I - A(z)]$ for finding the unknown boundary probabilities in the probability generating function. In section 3, we described in detail the procedures for constructing sets of linear equations for finding the unknown boundary probabilities for the dam/storage model, the breakdown model of Towsley, and the meteor burst communications model. In addition, we discussed an approach for finding the probability generating function of occupancy for a queueing model with multiple Markovian arrival and service processes. Once the probability generating function is found, the occupancy distribution is readily obtained by inverting the probability generating function, and the delay distribution follows.

CHAPTER 7
CONCLUSIONS

In this dissertation, we develop a computational methodology for finding important measures of a queueing system in which time is slotted, the distribution of the number of entitites that arrive during a slot is dependent upon the evolution of a discrete time, discrete state Markov chain, and the number of entities that may be served during a slot is limited to some number. The behavior of many physical systems, such as telecommunications systems, storage systems and manufacturing systems, is well modeled by this type of queueing models. Using probabilistic argruments, Neuts (for example, Neuts [1989]) and his colleagues have been able to develop a system of algorithmic techniques for analyzing and obtaining useful performance measures for this type of queueing systems. The method developed by Neuts and his colleagues has become known as the matrix analytical method. The matrix analytical method has long been recognized as a powerful tool not only in obtaining important measures, but also in gaining informative insights into the probabilistic structure of many queueing models. The disadvantages of the matrix analytical method, however, are its incomprehensibility to many a nonpractitioner of probability theory, and its unpredictability and sometimes excessive computational effort in determining some of the critical parameters for obtaining important performance measures. In fact, among the very few numerical examples that use the matrix analytical method as the analytical tool in the literature, most of them are very simple and are usually limited to some special cases of the general model. Moments are presented most of the time; distributions are presented only in rare instances and are limited to cases of a single server. Yet distributions are very important, not only in performance evaluation, but also in the design of many queueing systems. The purpose of our present investigation is to develop an alternative algorithmic approach for finding important distributions as well as moments for the type of queueing models mentioned above.

Our solution methodology is a transform/eigenanalysis approach which is based upon classical eigenanalysis and partial fraction expansion, and can be applied for the analysis of a large number of queueing models that arise in many physical environments. In this methodology, the queue length distribution is modeled in a transformed compact form using the probability generating function technique, an expression in the form of an integral-difference equation is formed, expanded in a spectral series, and finally inverse transformed to obtained the distribution. Our algorithm is efficient in terms of computation speed and effective in terms of stability and accuracy. Once the queue length distribution is obtained, the delay distribution and mean statistics are readily found using simple probabilistic argruments. Since the type of queueing models considered herein has great potential for applications in the analysis of packet switching communications systems, we study two communications models in detail and present some numerical examples in order to illustrate the viability and applicability of our method. In particular, we obtain the stationary distributions and the means of queue length and actual waiting time at departure points. Special attention is given to the accuracy of the results. In cases where data is available, we compare our results with those published in the literature. Finally, we discuss approaches to the solution of some additional models that have more general sturctures than the two communications models have.

We now recapitulate the main topics presented in this dissertation. In Chapter 1, we stated the problem considered in this deissertation, discussed briefly the matrix analytical method and our transform/eigenanalysis approach, defined the scope of our investigation, and decribed the organization of this dissertation. In Chapter 2, we presented a brief survey on Queueing Theory. Chapter 3 gave a brief chronicle on the developments in Telecommunications. In Chapter 4, we illustrated our methodology through a detailed ananlysis of a communications model whose queueing process has a one-to-one correspondence between the phase of the arrival process and the number of entities that arrive during a time slot. In Chapter 5, we extended our approach in Chapter 4 to the analysis of a more complicated communications model. Specifically, while in the model of Chapter 4 each source alternates between active and idle periods, and each source generates one packet for each time slot during an active period, in the model of Chapter 5, each source generates one packet every, say, T slots of time during an active period. Additional models that have some new features as compared with those in the models of Chapters 4 and 5 are presented in Chapter 6. In this chapter, we conclude our discussions.

The major steps in our transform/eigenanalysis methodology for finding the occupancy and delay distributions for queues of the $M/G/R$ type are as follows:

1. Obtain the probability generating function in matrix form using the z-transformation technique.

2. Factor out all the trivial null values from $[z^R I - A(z)]$, find the other null values and the corresponding null vectors in $[z^R I - A(z)]$ by first converting the polynomial matrix into an eigenanalysis problem of the form $[zI - C]$, and then solve for the eigenvalues and eigenvectors of C.
3. Construct a set of linear equations to solve for the unknown boundary probabilities in the probability generating function $\mathcal{F}_{\bar{n}}(z)$ using the null values and vectors obtained in Step 2.
4. Construct additional equations to solve for the unknown boundary probabilities using the algebraic structure of each specific queueing model.
5. Compute the principal parts of the partial fractions and invert the probability generating function to obtain the occupancy probability masses.
6. Compute the delay probability masses using the occupancy distribution obtained in Step 5.

We recommend the following for further investigation:

1. In the commuications model with speed conversion, we constructed a set of linear equations for solving the unknown boundary probabilities in the probability generating function. The number of equations thus constructed is usually greater than the number of unknowns for this particular model. A further investigation into the algebraic structure of this model to curtail computation inefficiency due to the presence of redundant equations is recommended.
2. We have been fortunate in not including repeated eigenvalues in our numerical examples. However, we have no reason to dismiss the possibility of repeated eigenvalues occurring in some other models or in cases that we have not investigated. Therefore, procedures that deal with repeated eigenvalues are needed. A plausible path is through finding efficient ways to invert elementary Jordan blocks. In addition, an efficient computational procedure for finding repeated eigenvalues and the corresponding principal vectors for a general matrix is needed.
3. In our examples, we solve for the null values and vectors of polynomial matrices by converting them into standard eigenvalue-eigenvector problems of bigger sizes. We recommend further study to develop an efficient computational procedure for finding null values and vectors using the original polynomial matrices.
4. There are some other interesting queueing properties of the $M/G/R$ queueing system that one may wish to pursue; for example, the output distribution, the busy period distribution, and the blocking probability distribution for a $M/G/R$

queueing system with finite capacity.

5. One may wish to continue investigation into multi-layered queueing systems with embedded micro-systems of the $M/G/R$ type; for example, an ATM communications system in which packet arrivals vary with time periods.
6. Networks of queueing nodes of the $M/G/R$ type are also interesting research topics.

BIBLIOGRAPHY

Ahmadi, Hamid. 1993. Analysis of a Rate-Based Access Control Mechanism for High-Speed Networks. *IEEE Trans. on Comm.*, **41:6**, pp. 940-950.

Ali Khan, M. S. 1970. Finite Dams with Inputs Forming a Markov Chain. *J. Appl. Prob.*, **7**, pp. 291-303.

Ali Khan, M. S. and J. Gani. 1968. Infinite Dams with Inputs Forming a Markov Chain. *J. Appl. Prob.*, **5**, pp. 72-83.

Anick, D., D. Mitra and M. M. Sondhi. 1982. Stochastic Theory of a Data-Handling System with Multiple Sources. *The Bell Technical J.*, **61:8**, pp. 1871-1895.

Asmussen, S. and V. Ramaswami. 1990. Probabilistic Interpretations of Some Duality Results for the Matrix Paradigms in Queueing Theory. *Stochastic Models*, **6:4**, pp. 715-733.

Bagchi, T. P. and J. G. C. Templeton. 1973. A Note on the $M^X/G^X/1,K$ Bulk Queueing System. *J. Appl. Prob.*, **10**, pp. 901-906.

Bagchi, T. P. and J. G. C. Templeton. 1973. Finite Waiting Space Bulk Queueing Systems. *J. of Engineering Mathematics*, **7:4**, pp. 313-317.

Bailey, N. T. J. 1954. On Queueing Processes with Bulk Service. *J. Royal Statistical Society, Ser. B*, **16**, pp. 80-87.

Blondia, C. 1989. The N/G/1 Finite Capacity Queue. *Stochastic Models*, **5:2**, pp. 273-294.

Blondia, C. and O. Casals. 1992. Performance Analysis of Statistical Multiplexing of VBR sources. *Proc. IEEE INFOCOM'92*, pp. 828-837.

Boudreau, P. E., J. S. Griffin and M. Kac. 1962. An Elementary Queueing Problem. *American Math. Monthly*, **69**, pp. 713-724.

Brady, T. 1968. A Statistical Analysis of On-Off Patterns in 16 Conversations. *The Bell System Technical J.*, pp.73-91.

Brogan, W. L. 1982. *Modern Control Theory*, Englewood Cliffs, N.J.

Bruneel, H. 1985. Some Remarks on Discrete-Time Buffers with Correlated Arrivals. *Comput. & Ops. Res.*, **12:5**, pp. 445-458.

Bruneel, H. 1988. Queueing Behavior of Statistical Multiplexers with Correlated Inputs. *IEEE Trans. on Commun.*, **36:12**, pp. 1339-1341.

Burman, D. and D. R. Smith. 1986. An Asymtotic Analysis of a Queueing System with Markov-Modulated Arrivals. *Operations Research*, **34:1**, pp. 105-119.

Cao, W.-L. and W. J. Stewart. 1987. Queueing Models, Block Hessenberg Matrices and the Method of Neuts. *Annals of Operations Research*, **8**, pp. 265-284.

Chandramouli, Y., M. F. Neuts and V. Ramaswami. 1987. A Queueing Model for Meteor Burst Packet Communications Systems. *IEEE Trans. on Commun.*, **37:10**, pp. 1024-1030.

Chu, W. W. 1969. A Study of Asynchronous Time Division Multiplexing for Time-Sharing Computers. *Fall Joint Comput. Conf., AFIPS Conf. Proc.*, Montvale, NJ. **35**, pp. 669-678, AFIPS Press.

Crommelin, C. D. 1932. Delay Probability Formulae when the Holding Times are Constant. *The Post Office Electrical Engineers' Journal*, **25**, pp.41-50.

Crommelin, C. D. 1934. Delay Probability Formulae. *The Post Office Electrical Engineers' Journal*, **26**, pp. 266-274.

Dagsvik, J. 1975. The General Bulk Queue as a Matrix Factorisation Problem of the Wiener-Hopf Type, Part I. *Adv. Appl. Prob.*, **7**, pp. 636-646.

Dagsvik, J. 1975. The General Bulk Queue as a Matrix Factorisation Problem of the Wiener-Hopf Type, Part II. *Adv. Appl. Prob.*, **7**, pp. 647-655.

Daigle, J. N. 1989. Queue Length Distributions from Probability Generating Functions via Discrete Fourier Transforms. *Operations Research Letters*, **8**, pp. 229-236.

Daigle, J. N. 1991. *Queueing Theory for Computer Communications*, Addison-Wesley, MA.

Daigle, J. N. and J. D. Langford. 1986. Queueing Analysis of Packet Voice Communication Systems. *IEEE Journal on Selected Areas in Communications*, **SAC-4:6**, pp. 847-855.

Daigle, J. N., Y. Lee, and M. N. Magalhães. 1990. Discrete Time Queues with Phase Dependent Arrivals, *Proceedings of IEEE INFOCOM'90*, pp. 728-732, San Francisco.

Daigle, J. N. and D. M. Lucantoni. 1990. Queueing Systems Having Phase Dependent Arrival and Service Rates. *First International Workshop on the Numerical Solutions of Markov Chains*, pp. 375-395, Raleigh, N. C.

Daigle, J. N. and S. C. Tang. 1992. The Queue Length Distribution for Multiserver Discrete Time Queues with Batch Markovian Arrivals. *Stochastic Models*, **8:4**, pp. 665-683.

Daumas, Maurice (ed.). 1979. *A History of Technology and Invention: Progress Through the Ages*, **Vol. III**, Crown Publishing, N. Y.

Derry, T. K., and T. I. Williams. 1961. *A Short History of Technology: From the Earliest Time to A.D. 1990*, Oxford University Press, N.Y.

Descloux, A. 1991. Stochastic Models for ATM Switching Networks. *IEEE J. on Selected Areas in Communications*, **9:3**, pp. 450-457.

de Smit, J. H. A. 1983. The Queue GI/M/s with Customers of Different Types or The Queue GI/Hm/s. *Adv. Appl. Prob.*, **15**, pp. 392-419.

Dimsdale, B. 1960. Characteristic Roots of a Matric Polynomial. *J. Soc. Indust. Appl. Math.*, **8:1**, pp. 218-223.

Doshi, B. T. 1986. Queueing Systems with Vacations-A Survey. *Queueing Systems*, **1**, pp. 29-66.

Eckberg, A. E. 1979. The Single Server Queue with Periodic Arrival Process and Deterministic Service Times. *IEEE Trans. on Commun.*, **27:3**, pp. 556-562.

Evans, Richard V. 1967. Geometric Distribution in Some Two-dimensional Queueing Systems. *Operations Research*, **15:5**, pp. 830-846.

Fonseca and Silvester. 1994. Modelling the Output Process of an ATM Multiplexer with Markov Modulated Arrivals. *Proc. IEEE ICC'94*, **2**, pp.721-725.

Francis, J. G. F. 1961. The QR Transformation: A Unitary Analogue to the LR Transformation-Part 1. *Comp. J.*, **4**, pp. 265-271.

Francis, J. G. F. 1961. The QR Transformation: A Unitary Analogue to the LR Transformation-Part 2. *Comp. J.*, **4**, pp. 265-271.

Gail, H. R., S. L. Hantler and B. A. Taylor. 1994. Solutions of the Basic Matrix Equation for M/G/1 and G/M/1 type Markov Chain. *Stochastic Models*, **10:1**, pp. 1-43.

Gohberg, I., P. Lancaster and L. Rodman. 1982. *Matrix Polynomials*. Academic Press, NY.

Gohberg, I. C. and E. I. Sigal. 1971. An Operator Generalization of the Logarithm Residue Theorem and The Theorem of Rouche. *Math USSR Sbornik*, **13:4**, pp. 603-625.

Gopinath, B. and J. A. Morrison. 1977. Discrete-Time Single Server Queues with Correlated Inputs. *The Bell System Technical Journal*, **56:9**, pp. 1743-1768.

Golub, Gene H. and C. F. van Loan. 1989. *Matrix Computation* (2nd ed.), The Johns Hopkin Univ. Press, Baltimore.

Golub, G. H. and J. H. Wilkinson. 1976. Ill-Conditioned Eigensystems and the Computation of the Jordan Canonical Form. *SIAM Review*, **18:4**, pp. 578-619.

Guderley, K. G. 1958. On Nonlinear Eigenvalue Problems for Matrices. *J. Soc. Appl. Math.*, **6:4**, pp. 335-353.

Gün, L. 1989. Experimental Results on Matrix-Analytic Solution Techniques - Extentions and Comparisons. *Stochastic Models*, **5:4**, pp. 669-682.

Hashida, O., Y. Takahashi and S. Shimogawa. 1991. Switched Batch Bernoulli Process (SBBP) and the Discrete-Time SBBP/G/1 Queue with Application to Statistical Multiplexer Performance. *IEEE J. on Selected Areas in Commun.*, **9:3**, pp. 394-401.

Hayes, J. F. 1986. *Modeling and Analysis of Computer Communication Networks*. Plenum Press, New York.

Heffes, H. and D. M. Lucantoni. 1986. A Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performances. *IEEE J. on Selected Areas in Commun.*, **4:6**, pp. 856-868.

Herbert, H. G. 1972. An Infinite Discrete Dam with Dependent Inputs. *J. Appl. Prob.*, **9**, pp. 404-413.

Herbert, H. G. 1975. A Note on First Emptiness in a Discrete Storage System with Markovian Inputs. *SIAM J. Appl. Math.*, **28:3**, pp. 657-661.

Hunter, J. J. 1983. *Mathematical Techniques in Applied Probability, Vol. 1, Discrete Time Models: Basic Theory*. Academic Press, New York.

Jenq, Y. C. 1984. Approximations for Packetized Voice Traffic in Statistical Multiplexer. *Proc. IEEE Infocom'84*, San Francisco, pp. 256-259.

Jensen, A. 1953. Markov Chains as an Aid in the Study of Markoff Process. *Akturavictidskrift*, pp. 87-91.

Jones, Alexander. 1852. *Historical Sketch of the Electric Telegraph: Including Its Rise and Progress in the United States.* Putnam, N. Y.

Kagström, B. and A. Ruhe. 1980. An Algorithm for Numerical Computation of the Jordan Normal Form of a Complex Matrix. *ACM Transactions on Mathematical Software*, **6:3**, pp.399-443.

Khamisy, A. and M. Sidi. 1991. Discrete-Time Priority Queueing Systems with Two-State Markov Modulated Arrival Process. Manuscript submitted for publication in *Stochastic Models.*

Kiefer, J. and J. Wolfowitz. 1955. On the Theory of Queues with Many Servers. *Transactions of the American Mathematical Society*, **78:1**, pp. 1-18.

Kirby, R. S., S. Withington, A. B. Darling, and F.G. Kilgour. 1956. *Engineering History*, Mcgraw-Hill, N.Y.

Konheim, A. G. and M. Reiser. 1986. The Movable-Boundary Multiplexer Stability and Decomposability, in *Teletraffic Analysis and Computer Performance Evaluation.* O. J. Boxma, J. W. Cohn and H. C. Tijms (editors), Elservier Science Publishers (North Holland).

Kroner, H. 1991. Statistical Multiplexing of Sporadic Sources-Exact and Approximate Performance Analysis. *ITC*, **13**.

Kublanovskaya, V. N. 1966. On a Method of Solving the Complete Eigenvalue Problem for a Degenerate Matrix. *U.S.S.R. Computaional Mathematics and Mathematical Physics*, **6:4**, pp. 1-14.

Kühn, P. 1976. *Tables on Delay Systems*, Inst. of Switching and Data Techniques, Univ. of Stuttgart, Stuttgart, Germany.

Lancaster, P. 1960. *Lamdsa-Matrices and Vibrating Systems.* Pergamon Press, NY.

Lancaster, P. and H. K. Wimmer. 1975. Zur Theorie der Lambda-Matrizen. *Mathenatische Nachrichten*, **BD 68**, pp. 325-330.

Lancaster, P. 1977 A Fundamental Theorem on Lambda-Matrices with Applications-I. Ordinary Differential Equations with Constant Coefficients. *Linear Algebra and Its Applications*, **18**, pp. 189-211.

Lancaster, P. 1977 A Fundamental Theorem on Lambda-Matrices with Applications-II. Difference Equations with Constant Coefficients. *Linear Algebra and Its Applications*, **18**, pp. 213-222.

Landry, R. and I. Stavrakakis. 1994a. Multiplexing Generalized Periodic Markovian Sources with an Application to the Study of VBR Video. *Proc. IEEE ICC'94*, **2**, pp. 1014-1018.

Landry, R. and I. Stavrakakis. 1994b. Non-Deterministic Periodic Packet Streams and Their Impact on a Finite-Capacity Multiplexer. *Proc. IEEE INFOCOM'94*, **1**, pp. 224-231.

Latouche, G. 1982. A Phase-Type Semi-Markov Point Process. *SIAM J. Alg. Sisc. Meth.*, **3:1**, pp. 77-90.

Latouche, G. 1987. A Note on Two Matrices Occurring in the Solution of Quasi-Birth-Death Process. *Stochastic Models*, **3:2**, pp. 252-257.

Latouche, G. and V. Ramaswami. 1992. A Unified Stochastic Model for the Packet Stream from Periodic Sources. *Performance Eval.*, **14**, pp. 103-121.

Li, S.-Q. 1990. A General Solution Technique for Discrete Queueing Analysis of Multi-Media Traffic on ATM. *Proceedings of IEEE INFOCOM'90*, pp. 1144-1155, San Francisco.

Li, San-Qi. 1993. Generating Function Approach for Discrete Queueing Analysis with Decomposable Arrival and Service Markov Chains. *Stochastic Models*, **9:3**, pp. 401-420.

Liao, K. and L. G. Mason. 1989. A Discrete-Time Single Server Queue with a Two-Level Modulated Input and Its Applications. *Proc. IEEE ICC'89*, pp. 913-918.

Lloyd, E. H. 1963. Reservoirs with Serially Correlated Infl ws. *Technometrics*, **5:1**, pp. 85-93.

Lucantoni, D. M. 1991. New Results on the Single Server Queue with A Batch Markovian Arrival Process. *Stochastic Models*, **7:1**, pp. 1-46.

Lucantoni, D. M., G. C. Choudhury and W. Whitt. 1994. The Transient BMAP/G/1 Queue. *Stochastic Models*, **10:1**, pp. 145-182.

Lucantoni, D. M., K. S. Meier-Hellstern, and M. F. Neuts. 1990. A Single Server Queue with Server Vacations and a Class of Non-Renewal Arrival Processes. *Adv. in Appl. Prob.*, **22:3**, pp. 676-705.

Lucantoni, D. M.and M. F. Neuts. 1978. *Numerical Methods for a Class of Markov Chains Arising in Queueing Theory*. Tech. Rept. No. 78/10, Department of Statistics and Computer Science, University of Delaware, May 1978.

Lucantoni, D. M. and V. Ramaswami. 1985. Efficient Algorithms for Solving the Nonlinear Matrix Equations Arising in Phase Type Queues. *Stochastic Models*, **1:1**, pp. 29-51.

Markus, A. S. 1988. *Introduction to the Spectral Theory of Polynomial Operator Pencils*, American Mathematical Society, Rhode Island.

McNeil, Ian (ed.) 1990. *An Encyclopedia of the HIstory of Technology*, Routledge, N.Y.

Moler, C. B. and G. W. Stewart. 1973. An Algorithm for Generalized Matrix Eigenvalue Problems. *SIAM J. Numer. Anal.*, **10:2**, pp. 241-256.

Morris, R. J. T. 1981. An Algorithmic Technique for a Class of Queueing Models with Packet Switching Applications. *Proc. IEEE ICC*, **2**, pp. 41.2.1-41.2.8.
Moran, P. A. P. 1954. A Probability Theory of Dams and Storage Systems. *Aust. J. Appl. Sci.*, **5**, pp. 116-124.

Neuts, M. F. 1976. Moment Formulas for the Markov Renewal Branching Process. *Adv. Appl. Prob.*, **8**, pp. 690-711.

Neuts, M. F. 1978. Renewal Processes of Phase Type. *Naval Research Quarterly*, **25**, pp. 445-454.

Neuts, M. F. 1978. The M/M/1 Queue with Random Varying Arrival and Service Rates. *OPSEARCH*. **15:4**. pp. 139-168.

Neuts, M. F. 1979. Markovian Point Process. *J. Appl. Prob.*, **16**, pp. 764-779.

Neuts. M. F. 1981a. *Matrix Geometric Solutions in Stochastic Models*. The Johns Hopkins University Press. Baltimore.

Neuts, M. F. 1981b. The c-Server Queue with Constant Service Times and a Versatile Markovian Arrival Process. in *Applied Probability - Computer Science: The Interface*, **1**, pp. 31-67.

Neuts, M. F. 1986. A New Informative Embedded Markov Renewal Process for the PH/G/1 Queue. *Adv. Appl. Prob.*, **18**, pp.553-557.

Neuts, M. F. 1989. *Structured Stochastic Matrices of M/G/1 Type and Their Applications.* Marcel Dekker, New York.

Neuts, M. F. 1993. The Burstiness of Point Process. *Stochastic Models*, **9:3**, pp. 445-466.

Odoom, S. and E. H. Lloyd. 1965. A Note on the Equilibrium Distribution of Levels in a Semi-Infinite Reservoir Subject to Markovian Inputs and Unit Withdrawals. *J. Appl. Prob.*, **26**, pp. 215-222.

Parsons, J. D., and J. D. Gardiner. 1989. *Mobile Communication Systems*, John Wiley & Sons, N.Y.

Peters, G. and J. H. Wilkinson. 1970a. $Ax = \lambda Bx$ and the Generalized Eigenproblem. *SIAM J. Numer. Anal.*, **7:4**, pp. 479-492.

Peters, G. and J. H. Wilkinson. 1970b. Eigenvectors of Real and Complex Matrices by LR and QR Triangularizations. *Numer. Math.*, **16**, pp. 181-204.

Powers, John T. and Henry H. Stair. 1990. *Megabit Data Communications: A Guide for Professionals.* Prentice Hall, New Jersey.

Pratt, T. and C. W. Bostian. 1986. *Satellite Communications*, John Wiley & Sons, N.Y.

Press, W. H., B. P. Flanner, S.A. Teukolsky and W. T. Vetterling. 1988. *Numerical Recipes in C - The Art of Scientific Computing.* Marcel Dekker, New York.

Ramaswami, V. 1980. The N/G/1 Queue and Its Detailed Analysis. *Adv. Appl. Prob.*, **12**, pp. 222-261.

Ramaswami, V. 1988a. Stable Recursion for the Steady-State Vector for Markov Chains of the M/G/1 Type. *Stochastic Models*, **4**, pp. 183-188.

Ramaswami, V. 1988b. Nonlinear Matrix Equations in Applied Probability- Solution Techniques and Open Problems. *SIAM Review*, **30:2**, pp. 256-263.

Ramaswami, V. and G. Latouche. 1989. An Experimental Evaluation of the Matrix-Geometric Method for the GI/PH/1 Queue. *Stochastic Models*, **5:4**, pp. 629-667.

Ramaswami, V. 1990. A Duality Theorem for the Matrix Paradigms in Queueing Theory. *Stochastic Models*, **6:1**, pp. 151-161.

Rickard, W. I. 1972. Delay Performance of a Buffered Communication Network. *IEEE Trans. on Commun.*, **20:5**, pp. 1008-1015.

Robert, P., I. Mitrani and P. J. B. King. 1988. Analysis of a Meteor Scatter Communication Protocol, in P. J. Courtois and G. Latouche (eds.) *Performance'87*, Elsevier Science Publisher B. V. (North Holland), pp. 469-479.

Roberts. J. W. and J. T. Virtamo. 1991 The Superposition of Periodic Cell Arrival Streams in an ATM Multiplexer. *IEEE Trans. on Commun.*, **39:2**, pp. 298-303.

Roberts, Lawence G. 1978. The Evolution of Packet Switching. *Proceedings of the IEEE*, **66:11**, pp. 1307-1313.

Ross. S. M. 1983. *Stochastic Processes.* John Wiley, New York.

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

Rubin, I. and Z. Zhang. 1988. Message Delay Analysis for TDMA Schemes Using Contiguous-Slot Assignments. *Proc. IEEE ICC'88*, pp. 419-422.

Ruhe, Axel. 1970. An Algorithm for Numerical Determination of the Structure of a General Matrix. *BIT*, **10**, pp. 196-216.

Saito. 1990. The Departure Process of an N/G/1 Queue. *Performance Eval.*, **11**, pp. 241-251.

Schellhaas, H. 1990. On Ramaswami's Algorithm for the Generation of the Steady State Vector in Markov Chains of the M/G/1 type. *Stochastic Models*, **6:3**, pp.541-550.

Schwartz, Mischa, J. W. 1988. *Telecommunications Networks: Protocols, Modeling and Analysis.* Assison-Wesley, MA.

Sengupta, B. 1989. Markov Process whose Steady State Distribution is Matrix-Exponential with an Application to the GI/PH/1 Queue. *Adv. Appl. Prob.*, **21**, pp. 159-180.

Spragins, John D., J. L. Hammond and K. Pawlikowski. 1990. *Telecommunications: Protocols and Design.* Addison-Wesley, New York.

Sriram, K., P. K. Varshney and J. G. Shanthikumar. 1983. Discrete-Time Analysis of Integrated Voice/Data Multiplexers with and without Speech Activity Detectors. *IEEE J. on Selected Areas in Commun.*, **1:6**, pp. 1124-1132.

Stavrakakis, I. 1990. Analysis of a Statistical Multiplexer under a General Input Traffic Model. *Proceedings of IEEE INFOCOM'90*, pp. 1220-1225, San Francisco.

Stavrakakis, I. 1992. Statistical Multiplexing of Correlated Slow Traffic Sources. *Proc. IEEE GLOBCOM'92.*

Stern, T. E. 1983. A Queueing Analysis of Packet Voice. *Proc. Global Tele. Conf.* **1**, pp. 71-76. San Diego.

Takine, Suda and Hasegana. 1993. Cell Loss and Output Process Analyses of Finite Buffer Discrete Time Queueing System with Correlated Arrivals. *Proc. IEEE INFOCOM'93*, **3**, pp. 1259-1268.

Tarnove, Ivin. 1958. Determination of Eigenvalues of Matrices having Polynomial Elements. J. Soc. Appl. Math., **6:2**, pp. 163-171.

Towsley, D. 1980. The Analysis of a Statistical Multiplexer with Nonindependent Arrivals and Errors. *IEEE Trans. on Commun.*, **28:1**, pp. 65-72.

van Arem, B. 1990. *Queueing Models for Slotted Transmission Systems.* Ph. D. dissertation, Twente University, The Netherlands.

Viswanadham, N. and Y. Narahari. 1992. *Performance Modeling of Automated Manufacturing Systems*, Prentice Hall, NJ.

Walrand, J. 1988. *An Introduction to Queueing Networks*, Prentice Hall, NJ.

Ward, R. C. 1973. An Extension of the QZ Algorithm for solving the Generalized Matrix Eigenvalue Problem. *Technical Note. NASA*, **TN D-7305**.

Wilkinson, J. H. and C. Reinsch. 1971. *Handbook for Automatic Computation, Vol. II, Linear Algebra.* Springer-Verlag.

Winstend, C. B. 1959. Geometric Distributions in the Theory of Queues. *Journal of the Royal Statistical Society, Ser. B*, **21:1**, pp. 1-35.

Zhang, J. and E. J. Coyle. 1988. Matrix Recursive Solutions for the Transient Behavior of QBD-Process and Its Application to Random Access Networks. *Proc. CISS'88*, Princeton, NJ.

Zhang, J. and E. J. Coyle. 1989. Transient Analysis of Quasi-Birth Death Processes,. *Stochastic Models*, **5:3**, pp. 459-496.

Zhang, Z. and I. Rubin. 1990. Bounds on the Mean System-Size and Delay for a Movable-Boundary Integrated Circuit and Packet Switched Communications Channel. *Proc. IEEE Infocom'90*, pp.866-873.

APPENDIX A
Source code for numerical examples in Chapter 4 and Chapter 5

```
#define _MC68881_
/*****
 * Discrete-time analysis of a slotted transmission system:
 * Time is divided into frames. In each frame R slots are available for the
 * transmission of packets generated by N sources. Each source alternates between
 * passive and active periods. During an active period. A source generates a packet
 * every ms-1 frames. The aggregate arrival of packets is governed by a transition
 * probability matrix P.
 *****/
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "util.h"
#include "matrix_util.h"
#include "complex_util.h"
#include "eigen_util.h"
#include "geigen_util.h"
#include "svdcmp.h"
#define TINY (double) 1.0e-30
#define SMALL (double) 1.0e-15
#define BIG (double) 1.18973e+4932

/* Functions declaration */
void get_model_parameters(int*,int*,int*,double*,double*);
void assign_P(int,int,double,double,double**,int);
void erg_arrival(int,int,double**,int*,double*,double**,double*);
void factorize_sc_A(int,int,int,double**,int*,int*,int*,int*);
void elimin_redun_var(int,int,int,double**,int*,int*,int*,int*);
void assign_C(int,int,int,double**,int*,int*,int*,int*,int,double**,double**);
void pre_eigen(int,int,int,int*,double**,double**,double**,double*,int*);
void compute_index(int,double**,int*);
void eigen_anal(int,double**,double*,double*,double**);
void norm_null_vec(int,double*,double*,double**);
void assign_X(int,double*,double*,double**,c_rect**);
void eigen_right(int,int,double*,double*,double**,int,double**,double*,int*);
void eigen_left0(int,int,int,double*,double*,double**,c_rect**,double**);
void eigen_left(int,int,int,int,int,double,int*,double*,double**,double**,int*,int*,int*,
        double**);
void compute_F(int,int,int,int,double**,double**,int*,int*,int*,double***);
void solve_phi_0(int,int,int*,int*,int*);
void solve_phi_1(int,int,int,int*,int,double*,int*,double**,double,double**,double*,int*,
        int*);
void solve_phi_2(int,int,int,int*,int,double**,int,double*,int*,double**,double**,
        double***,double**,double*,int*,int*,int*,int,int);
void solve_phi_3(int,int,double**,double*);
void solve_phi_4(int,int,double**,double*);
void assign_B_phi_A(int,int,int,int*,double**,double**,int*,double**);
void partial_fract(int,double*,int*,double**,double**,double***,c_rect**,int,int,int,int,
        double**,double**);
void exp_queue(int,int,int,double**,int,double*,int*,c_rect**,int*,double*);
void check_accuracy(int,int,int,double**,int,double*,int*,c_rect**,int*,double*);
void pgf_of_queue(double**,int,double*,int*,c_rect**,int,int,int,int,double*,int*,
        double**,double);

FILE *ifp, *ofp;
unsigned long aaa;

main()
{
  c_rect **mu,**c_X;
  double alpha,beta,**P,*v,**a_m,Ea,**A0i,det0,**C0,**C1,**null_vec,*null_re,
        *null_im;
  double EM,EP,rho,**S,*s,Ed,**right_vec,**left_vec,*null_val,***F,**Phi,**gamma;
  double **B_Phi_A,Eq,error;
  int   i,j,k,ms,M,*bn,N,R,nC,*skip,degree,*Rindex,*Lindex,n_eigen,*zero_bp,ind0,
        *compx;
  int   r_size,c_size,row;

  time(&aaa); printf("%s\n",ctime(&aaa));
  ofp = fopen("slotted_out_data","w");
  time(&aaa); fprintf(ofp,"%s\n",ctime(&aaa));

/* Assign transition probability matrices of arrival-and-phase. */
  ifp = fopen("slotted_in_data","r");
  get_model_parameters(&N,&ms,&R,&EM,&EP);
  alpha = 1.0-1.0/EM;    /* alpha = prob. of remaining active after transmission */
  beta = 1.0-1.0/EP;     /* beta = prob. of remaining idle */
  fclose(ifp);
  fprintf(ofp,"No. of sources = %d, no. of phases per source = %d, no. of servers =
        %d\n",N,ms,R);
  fprintf(ofp,"message length = %5.1f, length of idle period = %9.6f\n",EM,EP);
  fprintf(ofp,"prob. of remaining active = %9.6f, prob. of remaining idle =
        %9.6f\n",alpha,beta);
  M = ncr(N+ms-1,N); P = dmatrix(1,M,1,M);
  assign_P(N,ms,alpha,beta,P,M);
  fprintf(ofp,"\nMk[]: ");
  bn = ivector(0,N+1); bn[0] = 1;
  for (i=0; i<=N; i++) {bn[i+1] = ncr(N-i+ms-2,N-i)+bn[i]; fprintf(ofp,"%3d",bn[i+1]-
        bn[i]);}
  fprintf(ofp,",  No. of phases = %3d\n",M);

/* Find ergodic prob. and exp. no. of packet arrivals in a frame. */
  v = dvector(1,M); /* ergodic phase probabilities */
  a_m = dmatrix(1,M,1,N); /*  arrival probability conditioned on initial phase */
  erg_arrival(M,N,P,bn,v,a_m,&Ea); rho = Ea/R;
  fprintf(ofp,"\nExp. no. of arrivals = %9.6f, Server utilization = %9.6f\n",Ea,rho);
  if(Ea >= R) {printf("Exp. no. of arrivals exceeds no. of slots.\n");
```

```
    printf("Exiting program....\n"); goto Done;}

/* Find null values and null vectors. */
  Rindex = ivector(1,M); Lindex = ivector(1,M); n_eigen = N*M;
  factorize_sc_A(M,N,R,P,bn,Rindex,Lindex,&n_eigen);
  fprintf(ofp,"\nn_eigen = %3d\n",n_eigen);
  skip = ivector(2*M+1,(N+1)*M);
  elimin_redun_var(M,N,R,P,bn,Rindex,Lindex,skip);
  nC = M; for (i=2*M+1;i<=(N+1)*M;i++) if (skip[i] != 1) nC++; fprintf(ofp,"\nnC =
        %d\n",nC);
  C0 = dmatrix(1,M,1,M); C1 = dmatrix(1,nC,1,nC);
  assign_C(M,N,R,P,bn,Rindex,Lindex,skip,nC,C0,C1);
  time(&aaa); printf("Assign C finished. Time: %s\n",ctime(&aaa));
  A0i = dmatrix(1,M,1,M);
  pre_eigen(M,N,nC,skip,C0,C1,A0i,&det0,&degree);
  fprintf(ofp,"\nDeterminant of sc_A0 = %14.6e\n",det0);
  if (fabs(det0) < TINY) {printf("Singular C0. Exiting program....\n"); goto Done;}
  time(&aaa); printf("C0 inverted. Time: %s\n",ctime(&aaa));
  fprintf(ofp,"\nDegree of sc_A after factorization = %3d\n",degree);
  free_ivector(skip,2*M+1,(N+1)*M); free_dmatrix(C0,1,M,1,M);
  compute_index(nC,C1,&ind0);
  fprintf(ofp,"\nIndex of null-value 0 = %2d\n",ind0);
  null_re = dvector(1,nC); null_im = dvector(1,nC);
  null_vec = dmatrix(1,nC,1,nC);
  eigen_anal(nC,C1,null_re,null_im,null_vec);
  free_dmatrix(C1,1,nC,1,nC);
  norm_null_vec(nC,null_re,null_im,null_vec);
  if (ind0 == 1) {
        c_X = c_rect_matrix(1,nC,1,nC);
        assign_X(nC,null_re,null_im,null_vec,c_X);
  }
  right_vec = dmatrix(1,M,1,n_eigen); null_val = dvector(1,n_eigen);
  compx = ivector(1,n_eigen);
  eigen_right(n_eigen,nC,null_re,null_im,null_vec,M,right_vec,null_val,compx);
  time(&aaa); printf("Right Eigenanalysis finished. Time: %s\n",ctime(&aaa));
  free_dmatrix(null_vec,1,nC,1,nC);
  fprintf(ofp,"\nEigenvalues:\n");
  left_vec = dmatrix(1,n_eigen,1,M);
  if (ind0 == 1) {
    eigen_left0(M,nC,n_eigen,null_re,null_im,A0i,c_X,left_vec);
    free_c_rect_matrix(c_X,1,nC,1,nC);
  } else
  eigen_left(M,N,R,n_eigen,degree,det0,compx,null_val,right_vec,P,bn,Lindex,Rindex,
        left_vec);
  time(&aaa); printf("Left Eigenanalysis finished. Time: %s\n",ctime(&aaa));
  free_dvector(null_re,1,nC); free_dvector(null_im,1,nC);
  F = (double ***) calloc(ind0,sizeof(double**));
  for (i=0;i<ind0;i++) F[i] = dmatrix(1,M,1,M);
  compute_F(M,N,R,ind0,A0i,P,bn,Lindex,Rindex,F);
  time(&aaa); printf("Compute F finished. Time: %s\n",ctime(&aaa));
  free_dmatrix(A0i,1,M,1,M);
```

```
/* Resolution of boundary probabilities. */
  zero_bp = ivector(1,R*M);
  solve_phi_0(M,R,Rindex,&c_size,zero_bp);
  r_size = c_size; for (i=1;i<=M;i++) r_size -= Lindex[i];
  if (r_size < c_size) for (k=N;k>R;k--) r_size += (bn[k+1]-bn[k])*(k-R);
  S = dmatrix(1,r_size,1,c_size); s = dvector(1,r_size);
  solve_phi_1(M,N,R,Rindex,n_eigen,null_val,compx,right_vec,Ea,S,s,&row,zero_bp);
  time(&aaa); printf("Solve 1 finished. Time: %s\n",ctime(&aaa));
  fprintf(ofp,"\nNo. of z's such that (0 < z <= 1) = %4d\n",row-1);
  if (row <= c_size) {
    solve_phi_2(M,N,R,Rindex,ind0,P,n_eigen,null_val,compx,right_vec,
    left_vec,F,S,s,&row,zero_bp,bn,r_size,c_size);
    time(&aaa); printf("Solve 2 finished. Time: %s\n",ctime(&aaa));
    solve_phi_3(r_size,c_size,S,s);
    time(&aaa); printf("Solve 3 finished. Time: %s\n",ctime(&aaa));
    r_size = c_size;
  }
  solve_phi_4(r_size,c_size,S,s);
  time(&aaa); printf("Unknown probabilities found. Time: %s\n",ctime(&aaa));
  free_dmatrix(S,1,r_size,1,c_size);
  /* Put unknown probabilities in a matrix. */
  Phi = dmatrix(0,R-1,1,M);
  k = 1; for (i = 0; i < R; i++) for (j = 1; j <= M; j++) {
    if (zero_bp[i*M+j]) continue; Phi[i][j] = s[k]; k++;
  }
  free_dvector(s,1,r_size); free_ivector(zero_bp,1,R*M);
  fprintf(ofp,"\nUnknown boundary probabilities:\n");
  for (i = 0; i < R; i++) {for (j = 1; j <= M; j++)
    fprintf(ofp,"%14.6e",Phi[i][j]); fprintf(ofp,"\n");}

/* Dertermination of joint occupancy-phase distribution, and delay distribution. */
  B_Phi_A = dmatrix(0,N,1,M);
  assign_B_phi_A(M,N,R,Rindex,Phi,P,bn,B_Phi_A);
  free_dmatrix(P,1,M,1,M); free_ivector(bn,0,N+1);
  free_dmatrix(Phi,0,R-1,1,M); free_ivector(Rindex,1,M);
  mu = c_rect_matrix(1,n_eigen,1,M);
  gamma = dmatrix(0,N+ind0-1,1,M);
  partial_fract(n_eigen,null_val,compx,left_vec,right_vec,F,mu,N,M,R,ind0,gamma,
        B_Phi_A);
  time(&aaa); printf("Partial fraction done. Time: %s\n",ctime(&aaa));
  for (i=0;i<ind0;i++) free_dmatrix(F[i],1,M,1,M); free((char *) F);
  free_dmatrix(left_vec,1,n_eigen,1,M);
  free_dmatrix(right_vec,1,M,1,n_eigen);
  free_dmatrix(B_Phi_A,0,N,1,M);
  exp_queue(M,N,ind0,gamma,n_eigen,null_val,compx,mu,Lindex,&Eq);
  time(&aaa); printf("Expected queue length computed. Time: %s\n",ctime(&aaa));
  fprintf(ofp,"\nExpected system occupancy (from derivative) = %14.6e\n",Eq);
  Ed = Eq/Ea;
  fprintf(ofp,"\nExpected delay (Little's formula) = %14.6e\n",Ed);
  check_accuracy(M,N,ind0,gamma,n_eigen,null_val,compx,mu,Lindex,&error);
  time(&aaa); printf("Error computed. Time: %s\n",ctime(&aaa));
  fprintf(ofp,"\nError = %14.6e\n",error);
  fprintf(ofp,"\n"); fprintf(ofp,"'*'\n");
  pgf_of_queue(gamma,n_eigen,null_val,compx,mu,M,N,R,ind0,v,Lindex,a_m,Ea);
  free_dmatrix(gamma,0,N+ind0-1,1,M);
  free_c_rect_matrix(mu,1,n_eigen,1,M);
  free_ivector(compx,1,n_eigen);
  free_dvector(null_val,1,n_eigen);
  free_ivector(Lindex,1,M);
Done:
  free_dvector(v,1,M);
  free_dmatrix(a_m,1,M,1,N);
  time(&aaa); fprintf(ofp,"\n%s\n",ctime(&aaa)); fclose(ofp);
  time(&aaa); printf("%s\n",ctime(&aaa));
}

void pgf_of_queue(gamma,n_eigen,null_val,compx,mu,M,N,R,ind0,v,Lindex,a_m,Ea)
int n_eigen,M,N,R,ind0,*Lindex,*compx; double **gamma,*null_val,*v,**a_m,Ea;
c_rect **mu;
{
  int j, k, n, m, no_pt, q_max, d, ub, d_max;
  double mag_sq, q_prob, *q_rprob, r_n, *d_prob, d_rprob, Ed, res, mean_q;
  c_rect sigma, c_temp, **s_pow_vec;

  time(&aaa); printf("Finding distributions and first moments. Time:
        %s\n",ctime(&aaa));
  d_max = 150; q_max = (d_max+1)*R-1; no_pt = 90; mean_q = 0.0;
  q_rprob = dvector(1,M);
  d_prob = dvector(1,d_max);
  for (k=1; k<=M; k++) q_rprob[k] = v[k];
  s_pow_vec = c_rect_matrix(1,n_eigen,1,M);
  for (k=1; k<=n_eigen; k++) for (j=1;j<=M;j++)
    {s_pow_vec[k][j].re = 1.0; s_pow_vec[k][j].im = 0.0;}
  res = 1.0;
  fprintf(ofp,"\nOccupancy \t Res. prob.\n");
  for (n=0; n<=q_max; n++) {
    if (n < no_pt) fprintf(ofp,"%4d",n);
    for (j=1;j<=M;j++) {
      m = n+Lindex[j];
      if (m < N+ind0) {
        q_prob = gamma[m][j];
        q_rprob[j] -= q_prob;
      } else {
        q_prob = 0.0;
        for (k=1; k<=n_eigen; k++) {
          mag_sq = null_val[k]*null_val[k];
          if (compx[k]) mag_sq += null_val[k+1]*null_val[k+1];
          if (mag_sq < (1.0-SMALL)) {
            sigma.re = null_val[k];
            if (compx[k]) sigma.im = null_val[k+1]; else sigma.im = 0.0;
            c_temp = c_mul(mu[k][j],s_pow_vec[k][j]);
            s_pow_vec[k][j] = c_mul(s_pow_vec[k][j],sigma);
            if (compx[k]) {q_prob += 2.0*c_temp.re; k++;}
            else q_prob += c_temp.re;
          } else if (compx[k]) k++;
        } /* end k */
        q_rprob[j] -= q_prob;
      } /* end if-else */
      res -= q_prob;
      if (n <= R) for (d=1;d<=d_max;d++) {
        ub = (d*R < N) ? d*R : N;
        for (k=(d-1)*R+1;k<=ub;k++)
          d_prob[d] += q_prob*a_m[j][k]*(k-(d-1)*R);
        for (k=ub+1;k<=N;k++)
          d_prob[d] += q_prob*a_m[j][k]*R;
      } else {
        d = (n-1)/R;
        ub = (d+1)*R-n;
        for (k=1;k<=ub;k++)
          d_prob[d] += q_prob*a_m[j][k]*k;
        for (k=ub+1;k<=N;k++)
          d_prob[d] += q_prob*a_m[j][k]*ub;
        for (d=(n-1)/R+1;d<=d_max;d++) {
          ub = (((d+1)*R-n) < N) ? ((d+1)*R-n) : N;
          for (k=d*R-n+1;k<=ub;k++)
            d_prob[d] += q_prob*a_m[j][k]*(k+n-d*R);
          for (k=ub+1;k<=N;k++)
            d_prob[d] += q_prob*a_m[j][k]*R;
        } /* end d */
      } /* end if-else */
    } /* end j */
    mean_q += res;
    if (n < no_pt) fprintf(ofp,"\t %14.6e\n",res);
  } /* end n */
  free_dvector(q_rprob,1,M);
  free_c_rect_matrix(s_pow_vec,1,n_eigen,1,M);
  fprintf(ofp,"\nMean occupancy = %14.6e\n",mean_q);
  fprintf(ofp,"\nDelay \t residual prob.\n");
  Ed = 1.0; d_rprob = 1.0;
  for (d=1;d<=d_max;d++) {
    d_prob[d] /= Ea; d_rprob -= d_prob[d];
    if (d <= no_pt) fprintf(ofp,"%3d \t %14.6e\n",d,d_rprob);
    Ed += d_rprob;
  }
  fprintf(ofp,"\nExpected delay = %14.6e\n",Ed);
  free_dvector(d_prob,1,d_max);
}

void check_accuracy(M,N,ind0,gamma,n_eigen,null_val,compx,mu,Lindex,error)
int M,N,ind0,n_eigen,*Lindex,*compx; double **gamma,*null_val,*error; c_rect **mu;
{
  int j, k, n;
  double mag_sq, check_sum;
  c_rect c_one, sigma, c_temp;

  c_one.re = 1.0; c_one.im = 0.0;
  check_sum = 0.0;
  for (j=1;j<=M;j++)
    for (n=Lindex[j];n<N+ind0;n++) check_sum += gamma[n][j];
  for (k=1;k<=n_eigen;k++) {
    mag_sq = null_val[k]*null_val[k];
    if (compx[k]) mag_sq += null_val[k+1]*null_val[k+1];
    if (mag_sq < (1.0-SMALL)) {
      if (compx[k]) {
        sigma.re = null_val[k]; sigma.im = null_val[k+1];
        for (j=1;j<=M;j++) {
          c_temp = c_div(mu[k][j],c_sub(c_one,sigma));
          check_sum += 2.0*c_temp.re;
        } /* end for */
        k++;
      } else for (j=1;j<=M;j++) check_sum += mu[k][j].re/(1.0-null_val[k]);
    } else if (compx[k]) k++;
  } /* end k */
  *error = 1.0 - check_sum;
}

void exp_queue(M,N,ind0,gamma,n_eigen,null_val,compx,mu,Lindex,Eq)
int M,N,ind0,n_eigen,*Lindex,*compx; double **gamma,*null_val,*Eq; c_rect **mu;
{
  int j, k, n;
  double mag_sq, r_temp;
  c_rect c_one, sigma, c_temp1, c_temp2;

  c_one.re = 1.0; c_one.im = 0.0;
  *Eq = 0.0;
  for (j=1;j<=M;j++)
    for (n=Lindex[j]+1;n<N+ind0;n++) *Eq += ((double) (n-Lindex[j]))*gamma[n][j];
  for (k=1;k<=n_eigen;k++) {
    mag_sq = null_val[k]*null_val[k];
    if (compx[k]) mag_sq = null_val[k]*null_val[k]+null_val[k+1]*null_val[k+1];
    if (mag_sq < (1.0-SMALL)) {
      if (compx[k]) {
        sigma.re = null_val[k]; sigma.im = null_val[k+1];
        c_temp1 = c_sub(c_one,sigma);
        for (j=1;j<=M;j++) {
          c_temp2 = rc_mul(N+ind0-Lindex[j]-1,c_temp1);
          c_temp2 = c_add(c_temp2,c_one);
          c_temp2 = c_div(c_temp2,c_temp1);
          c_temp2 = c_div(c_temp2,c_temp1);
          c_temp2 = c_mul(mu[k][j],c_temp2);
          *Eq += 2.0*c_temp2.re;
        } /* end j */
        k++;
      } else {
        r_temp = 1.0-null_val[k];
        for (j=1;j<=M;j++)
          *Eq += mu[k][j].re*((N+ind0-Lindex[j]-1)*r_temp+1.0)/r_temp/r_temp;
      } /* end if-then-else */
    } else if (compx[k]) k++;
  } /* end k */
}

void partial_fract(n_eigen,null_val,compx,left_vec,right_vec,F,mu,N,M,R,ind0,gamma,
        B_Phi_A)
int n_eigen,N,M,R,ind0,*compx;
double *null_val,**gamma,**B_Phi_A,**left_vec,**right_vec,***F; c_rect **mu;
{
  int i, j, k, n;
  c_rect *beta, sigma, ctemp, *sigma_pow;

  beta = c_rect_vector(0,N); sigma_pow = c_rect_vector(0,ind0);
  for (n=1;n<=n_eigen;n++) {
    /* Multiply the coeff. vectors by the right null-vector */
    for (i=0;i<=N;i++) {
      beta[i].re = 0.0; beta[i].im= 0.0;
      for (j=1;j<=M;j++) {
        ctemp.re = right_vec[j][n];
        if (compx[n]) ctemp.im = right_vec[j][n+1]; else ctemp.im =0.0;
        beta[i] = c_add(beta[i],rc_mul(B_Phi_A[i][j],ctemp));
      }
    } /* end i */
    /* Synthetic division of the N order polynomial by 1-sigma*z */
    sigma.re = null_val[n];
    if (compx[n]) sigma.im = null_val[n+1]; else sigma.im = 0.0;
    for (i=1;i<=N;i++) {
      beta[i] = c_add(beta[i],c_mul(beta[i-1],sigma));
    }
    complex_all_power(sigma,ind0,sigma_pow);
    /* save coefficients */
    for (i=0;i<=N;i++) for (j=1;j<=M;j++) {
      ctemp.re = left_vec[n][j];
      if (compx[n]) ctemp.im = left_vec[n+1][j]; else ctemp.im = 0.0;
      ctemp = c_mul(sigma_pow[ind0],c_mul(ctemp,beta[i]));
      if (i < N) {
        if (compx[n]) gamma[i+ind0][j] += 2.0*ctemp.re;
        else gamma[i+ind0][j] += ctemp.re;
      } else {
        mu[n][j].re = ctemp.re; mu[n][j].im = ctemp.im;
      }
    } /* end j and i */
    if (compx[n]) n++;
  } /* end n */
  free_c_rect_vector(sigma_pow,0,ind0); free_c_rect_vector(beta,0,N);
  for (n=0;n<ind0;n++) for (j=1;j<=M;j++) for (i=0;i<=N;i++) for (k=1;k<=M;k++)
    gamma[i+n][j] += B_Phi_A[i][k]*F[n][k][j];
}

void assign_B_phi_A(M,N,R,Rindex,Phi,P,bn,B_Phi_A)
int M,N,R,*Rindex,*bn; double **Phi,**P,**B_Phi_A;
/* B_Phi_A = B(z)*Phi*A(z), coeff. vector of ith power of z stored at the ith row */
{
  int n, i, j, k, h;
  double temp, Ahij;

  for (n=0;n<R;n++) for (k=0;k<=(N-R);k++) for (j=1;j<=M;j++) {
    h = k + Rindex[j];
    for (i=1;i<=M;i++) {
      if ((bn[h] <= j) && (j < bn[h+1])) Ahij = P[i][j]; else Ahij = 0.0;
      temp = Phi[n][i]*Ahij;
      B_Phi_A[k+n][j] -= temp;
      B_Phi_A[k+R][j] += temp;
    } /* end i */
  } /* end j, k, and n */
}

void solve_phi_4(r_size,c_size,S,s) int r_size,c_size; double **S,*s;
/* Step 4 for solving the unknown boundary probabilities */
{
        int j;
  double wmin,wmax,*x,*w,**V;

        w = dvector(1,c_size);
        V = dmatrix(1,c_size,1,c_size);
        x = dvector(1,c_size);

  svdcmp(S,r_size,c_size,w,V);
  wmax = 0.0;
  for (j=1;j<=c_size;j++) if (w[j] > wmax) wmax = w[j];
  wmin = wmax*1.0e-15;
  for (j=1;j<=c_size;j++) if (w[j] < wmin) w[j] = 0.0;
  svbksb(S,w,V,r_size,c_size,s,x);
  for (j=1;j<=c_size;j++) s[j] = x[j];

  free_dvector(x,1,c_size);
  free_dvector(w,1,c_size);
  free_dmatrix(V,1,c_size,1,c_size);
}

void solve_phi_3(r_size,c_size,S,s) int r_size,c_size; double **S,*s;
/* Step 3 for solving the unknown boundary probabilities */
{
  int   i,j,col,nrow,indx;
  double pivot,temp;
  /* Row reduce S */
  nrow = 1;
  for (col=1;col<=c_size;col++) {
    indx = nrow; pivot = S[nrow][col];
    for (i=nrow+1;i<=r_size;i++)
      if (fabs(S[i][col]) > fabs(pivot)) {pivot = S[i][col]; indx = i;}
    if (fabs(pivot) > TINY) {
      if (indx != nrow) {
        for (j=col;j<=c_size;j++) {
                temp = S[nrow][j]; S[nrow][j] = S[indx][j]; S[indx][j] = temp;
        }
                        temp = s[nrow]; s[nrow] = s[indx]; s[indx] = temp;
      }
      for (i=nrow+1;i<=r_size;i++) if (fabs(S[i][col]) > TINY) {
        temp = S[i][col]/pivot; S[i][col] = 0.0;
        for (j=col+1;j<=c_size;j++) S[i][j] -= S[nrow][j]*temp;
        s[i] -= s[nrow]*temp;
      }
    }
    nrow++;
  }
  temp = s[nrow]; for (i=nrow+1;i<=r_size;i++) if (fabs(s[i]) > fabs(temp)) temp = s[i];
  s[nrow] = temp; for (i=nrow+1;i<=r_size;i++) s[i] = 0.0;
  fprintf(ofp,"\ns_[c_size+1] = %14.6e\n",temp);
}

void solve_phi_2(M,N,R,Rindex,ind0,P,n_eigen,null_val,compx,right_vec,left_vec,F,S,s,
        row,zero_bp,bn,r_size,c_size)
int M,N,R,*Rindex,ind0,n_eigen,*compx,*row,*zero_bp,*bn,r_size,c_size;
double **P,*null_val,**right_vec,**left_vec,***F,**S,*s;
/* Set up remaining equations. */
{
  int c,i,j,h,k,m,n,p;
  double **f,**g,itemp,ub,Amic;
  c_rect ctemp1,ctemp2,sigma,*sigma_pow;

  j = M;
  for (k=N;k>R;k--) for (j=(bn[k+1]-1);j>=bn[k];j--) {
    f = dmatrix(1,M,0,k-1); sigma_pow = c_rect_vector(0,k-1);
    for (p=0;p<ind0;p++) for (i=1;i<=M;i++) f[i][p] = F[p][i][j];
    for (n=1;n<=n_eigen;n++) {
      sigma.re = null_val[n];
      if (compx[n]) sigma.im = null_val[n+1]; else sigma.im = 0.0;
      complex_all_power(sigma,k-1,sigma_pow);
      if (compx[n]) {
        ctemp2.re = left_vec[n][j]; ctemp2.im = left_vec[n+1][j];
        for (p=ind0;p<k;p++) for (i=1;i<=M;i++) {
          ctemp1.re = right_vec[i][n]; ctemp1.im = right_vec[i][n+1];
          ctemp1 = c_mul(ctemp1,c_mul(ctemp2,sigma_pow[p]));
          f[i][p] += 2.0*ctemp1.re;
        }
        n++;
      } else for (p=ind0;p<k;p++) for (i=1;i<=M;i++)
        f[i][p] += right_vec[i][n]*left_vec[n][j]*sigma_pow[p].re;
    } /* end n */
    free_c_rect_vector(sigma_pow,0,k-1); g = dmatrix(1,M,0,k-1);
    for (p=0;p<k;p++) for (i=1;i<=M;i++) {
      itemp = ((N-R) > R) ? (N-R) : R; ub = (p < itemp) ? p : itemp;
      for (n=0;n<=ub;n++) for (c=1;c<=M;c++) {
        m = (n+Rindex[c]) % (N+1);
        if ((bn[m] <= c) && (c < bn[m+1])) Amic = P[i][c]; else Amic = 0.0;
        g[i][p] += Amic*f[c][p-n];
      } /* end n */
    } /* end p */
    free_dmatrix(f,1,M,1,k-1);
    for (m=R;m<k;m++) {
      c = 1;
      for (i=0;i<R;i++) for (n=1;n<=M;n++) {
        if (zero_bp[i*M+n]) continue;
        S[*row][c] = g[n][m-R]-g[n][m-i];
        c++;
      }
      *row += 1;
    }
    free_dmatrix(g,1,M,1,k-1);
  } /* end k */
}

void solve_phi_1(M,N,R,Rindex,n_eigen,null_val,compx,right_vec,Ea,S,s,row,zero_bp)
int M,N,R,*Rindex,n_eigen,*row,*zero_bp,*compx; double
*null_val,Ea,**S,*s,**right_vec;
/* first step in solving the boundary probabilities */
{
  int   i, j, k, n, col;
  double mag_sq;
  c_rect z, *z_pow, *psi, ctemp;

  /* Set up the first row of equation using the null-value at 1 */
  col = 1;
  for (k = 0; k < R; k++) for (j=1;j<=M;j++)
    if (!zero_bp[k*M+j]) {S[1][col] = R-k; col++;}
  s[1] = R-Ea; *row = 2;

  /* Set up the equations using null-values 0 < |z| < 1.0. */
  z_pow = c_rect_vector(0,R);
  psi = c_rect_vector(1,M);
  for (n = 1; n <= n_eigen; n++) {
    if (compx[n]) mag_sq = null_val[n]*null_val[n]+null_val[n+1]*null_val[n+1];
    else mag_sq = null_val[n]*null_val[n];
    if (mag_sq > (1.0+SMALL)) {
      if (compx[n]) {z.re = null_val[n]/mag_sq; z.im = -null_val[n+1]/mag_sq;}
      else {z.re = 1.0/null_val[n]; z.im = 0.0;}
```

```
        complex_all_power(z,R,z_pow);
        /* pre-multiplication by inv(AR) */
        for (i = 1; i <= M; i++) {
          k = Rindex[i]; psi[i].re = right_vec[i][n];
          if (compx[n]) psi[i].im = right_vec[i][n+1]; else psi[i].im = 0.0;
          psi[i] = c_div(psi[i],z_pow[k]);
        }
        col = 1;
        for (k = 0; k < R; k++) for (i = 1; i <= M; i++)
          if (!zero_bp[k*M+i]) {
            ctemp = c_mul(psi[i],c_sub(z_pow[R],z_pow[k]));
            S[*row][col] = ctemp.re;
            if (compx[n]) S[*row+1][col] = ctemp.im;
            col++;
          }
        if (compx[n]) {n++; *row += 2;} else *row += 1;
      } else if (compx[n]) n++;
    } /* end n */
    free_c_rect_vector(z_pow,0,R);
    free_c_rect_vector(psi,1,M);
}

void solve_phi_0(M,R,Rindex,c_size,zero_bp) int M,R,*Rindex,*c_size,*zero_bp;
/* Step 0 for solving the unknown boundary probabilities, determine the size of S */
{
  int i,k;

  /* determine which boundary values are zeros */
  for (i=1;i<=M;i++) for (k=0;k<Rindex[i];k++) zero_bp[k*M+i] = 1;
  *c_size = R*M; for (i=1;i<=R*M;i++) *c_size -= zero_bp[i];
}

void compute_F(M,N,R,ind0,A0i,P,bn,Lindex,Rindex,F)
int M,N,R,ind0,*Lindex,*Rindex,*bn;
double **A0i,**P,***F;
{
  int i,j,k,h,d,m; double **Matx,Ahim;

  for (i=1;i<=M;i++) for (j=1;j<=M;j++) F[0][i][j] = A0i[i][j];
  Matx = dmatrix(1,M,1,M);
  for (d=1;d<ind0;d++) {
    for (i=1;i<=M;i++) for (j=1;j<=M;j++) Matx[i][j] = 0.0;
    for (k=d;k>=1;k--) {
      for (i=1;i<=M;i++) for (j=1;j<=M;j++) for (m=1;m<=M;m++) {
        h = (k+Lindex[i]+Rindex[m]) % (N+1);
        if ((bn[h] <= m) && (m < bn[h+1])) Ahim = P[i][m]; else Ahim = 0.0;
        if ((h == R) && (i == m)) Matx[i][j] += (1.0-Ahim)*F[d-k][m][j];
        else Matx[i][j] -= Ahim*F[d-k][m][j];
      } /* end i */
    } /* end k */
    for (i=1;i<=M;i++) for (j=1;j<=M;j++)
```

```
      for (m=1;m<=M;m++) F[d][i][j] -= A0i[i][m]*Matx[m][j];
  } /* end d */
  free_dmatrix(Matx,1,M,1,M);
}

void eigen_left(M,N,R,n_eigen,degree,det0,compx,null_val,right_vec,P,bn,Lindex,
        Rindex,left_vec)
int M,N,R,n_eigen,degree,*compx,*Lindex,*Rindex,*bn;
double det0,*null_val,**right_vec,**P,**left_vec;
{
  int   n,i,j,k,h,*indx;
  double z,temp,coeff,sum,d,**Matx,*vec,Ahij;
  c_rect c_one,c_z,c_temp,c_coeff, c_sum,sigma,**c_Matx,*c_vec,c_d,*detA;

  /* find the determinant of sc_ALR */
  detA = c_rect_vector(0,n_eigen); detA[0].re = det0;
  printf("n_eigen = %3d\n",n_eigen);
  for (n=1;n<=n_eigen;n++) if (compx[n] == 1) {
    sigma.re = null_val[n]; sigma.im = null_val[n+1];
    for (i=n;i>=1;i--) detA[i] = c_sub(detA[i],c_mul(detA[i-1],sigma));
    sigma.im *= -1.0; n++;
    for (i=n;i>=1;i--) detA[i] = c_sub(detA[i],c_mul(detA[i-1],sigma));
  } else for (i=n;i>=1;i--) detA[i].re -= detA[i-1].re*null_val[n];

  /* find the principal parts of the partial fractions */
  indx = ivector(1,M); c_one.re = 1.0; c_one.im = 0.0;
  for (n=1;n<=n_eigen;n++) {
    printf("n = %2d",n);
    if (compx[n]) {
      printf(", compx\n");
      /* find inverse matrix and determinant */
      c_Matx = c_rect_matrix(1,M,1,M); c_vec = c_rect_vector(1,M);
      sigma.re = null_val[n]; sigma.im = null_val[n+1]; c_z = c_div(c_one,sigma);
      c_temp.re = 1.0; c_temp.im = 0.0;
      for (k=0;k<=degree;k++) {
        for (i=1;i<=M;i++) for (j=1;j<=M;j++) {
          h = (k+Lindex[i]+Rindex[j]) % (N+1);
          if ((bn[h] <= j) && (j < bn[h+1])) Ahij = P[i][j]; else Ahij = 0.0;
          if ((h == R) && (i == j))
            c_Matx[i][j] = c_add(c_Matx[i][j],rc_mul(1.0-Ahij,c_temp));
          else c_Matx[i][j] = c_sub(c_Matx[i][j],rc_mul(Ahij,c_temp));
        }
        c_temp = c_mul(c_temp,c_z);
      }
      complex_ludcmp(c_Matx,M,indx,&d);
      c_d.re = d; c_d.im = 0.0; for (i=1;i<=M;i++) c_d = c_mul(c_d,c_Matx[i][i]);
      /* divide det(sc_ALR) by (1-sigma*z) and then substitute in z */
      c_temp.re = 1.0; c_temp.im = 0.0;
      c_coeff.re = c_sum.re = det0; c_coeff.im = c_sum.im = 0.0;
      for (i=1;i<n_eigen;i++) {
        c_temp = c_mul(c_temp,c_z);
      c_coeff = c_add(detA[i],c_mul(sigma,c_coeff));
      c_sum = c_add(c_sum,c_mul(c_coeff,c_temp));
    }
    /* find left null-vectors */
    c_d = c_div(c_d,c_sum);
    for (j=1;j<=M;j++) {
      for (i=1;i<=M;i++) {c_vec[i].re = 0.0; c_vec[i].im = 0.0;}
      c_vec[j].re = c_d.re; c_vec[j].im = c_d.im;
      complex_lubksb(c_Matx,M,indx,c_vec);
      for (i=1;i<=M;i++) if ((fabs(right_vec[i][n])+fabs(right_vec[i][n+1])) > TINY) {
        c_temp.re = right_vec[i][n]; c_temp.im = right_vec[i][n+1];
        c_temp = c_div(c_vec[i],c_temp);
        break;
      } /* end i */
      left_vec[n][j] = c_temp.re; left_vec[n+1][j] = c_temp.im;
    } /* end j */
    free_c_rect_matrix(c_Matx,1,M,1,M); free_c_rect_vector(c_vec,1,M);
    n++;
  } else {
    printf(", real\n");
    /* find inverse matrix and determinant */
    Matx = dmatrix(1,M,1,M); vec = dvector(1,M);
    temp = 1.0; z = 1.0/null_val[n];
    for (k=0;k<=degree;k++) {
      for (i=1;i<=M;i++) for (j=1;j<=M;j++) {
        h = (k+Lindex[i]+Rindex[j]) % (N+1);
        if ((bn[h] <= j) && (j < bn[h+1])) Ahij = P[i][j]; else Ahij = 0.0;
        if ((h == R) && (i == j))
          Matx[i][j] += (1.0-Ahij)*temp;
        else Matx[i][j] -= Ahij*temp;
      }
      temp *= z;
    }
    ludcmp(Matx,M,indx,&d); for (i=1;i<=M;i++) d *= Matx[i][i];
    /* divide det(sc_ALR) by (1-sigma*z) and then substitute in z */
    temp = 1.0; coeff = sum = det0;
    for (i=1;i<n_eigen;i++) {
      temp *= z;
      coeff = detA[i].re+null_val[n]*coeff;
      sum += temp*coeff;
    }
    /* find left null-vectors */
    d /= sum;
    for (j=1;j<=M;j++) {
      for (i=1;i<=M;i++) vec[i] = 0.0; vec[j] = d;
      lubksb(Matx,M,indx,vec);
      for (i=1;i<=M;i++) if (fabs(right_vec[i][n]) > TINY) {
        left_vec[n][j] = vec[i]/right_vec[i][n];
        break;
      } /* end i */
    } /* end j */
```

```
      free_dmatrix(Matx,1,M,1,M); free_dvector(vec,1,M);
    } /* end if-else */
  } /* end n */
  free_c_rect_vector(detA,0,n_eigen);
  free_ivector(indx,1,M);
}

void eigen_left0(M,nC,n_eigen,null_re,null_im,A0i,c_X,left_vec)
int M,nC,n_eigen; double *null_re,*null_im,**A0i,**left_vec; c_rect **c_X;
{
  int h, i, j, k, *indx; double d; c_rect **c_Xi;

  c_Xi = c_rect_matrix(1,nC,1,nC); for (i=1;i<=M;i++) c_Xi[i][i].re = 1.0;
  indx = ivector(1,nC); complex_ludcmp(c_X,nC,indx,&d);
  complex_lubksb_matrix(c_X,nC,M,indx,c_Xi); free_ivector(indx,1,nC);
  k = 1;
  for (i=1;i<=nC;i++) {
    if (k > n_eigen) break;
    if ((null_re[i]*null_re[i]+null_im[i]*null_im[i]) > TINY) {
            if (fabs(null_im[i]) > TINY) {
              for (j=1;j<=M;j++) {
                for (h=1;h<=M;h++) left_vec[k][j] += c_Xi[i][h].re*A0i[h][j];
                for (h=1;h<=M;h++) left_vec[k+1][j] += c_Xi[i][h].im*A0i[h][j];
              }
              i++; k++;
            } else for (j=1;j<=M;j++) for (h=1;h<=M;h++)
              left_vec[k][j] += c_Xi[i][h].re*A0i[h][j];
     k++;
    } /* end if not 0.0 */
  } /* end i */
  free_c_rect_matrix(c_Xi,1,nC,1,nC);
}

void eigen_right(n_eigen,nC,null_re,null_im,null_vec,M,right_vec,null_val,compx)
int n_eigen,nC,M,*compx; double *null_re,*null_im,**null_vec,**right_vec,*null_val;
/* Store the right null-vectors in a compact form */
{
  int i, j, k;

  k = 1;
  for (j=1;j<=nC;j++) {
    if (k > n_eigen) break;
    if ((null_re[j]*null_re[j]+null_im[j]*null_im[j]) > TINY) {
            if (fabs(null_im[j]) > TINY) {
              compx[k] = 1; compx[k+1] = 1;
              null_val[k] = null_re[j]; null_val[k+1] = fabs(null_im[j]);
              for (i=1;i<=M;i++) {
                right_vec[i][k] = null_vec[i][j]; right_vec[i][k+1] = null_vec[i][j+1];
              }
              j++; k++;
            } else {
              null_val[k] = null_re[j];
              for (i=1;i<=M;i++) right_vec[i][k] = null_vec[i][j];
      } /* end if-else */
      k++;
    } /* end if not 0.0 */
  } /* end j */
}

void assign_X(nC,null_re,null_im,null_vec,c_X)
int nC; double *null_re,*null_im,**null_vec; c_rect **c_X;
{
  int i,j,COMPLEX;

  for (j=1;j<=nC;j++) {
    if (fabs(null_im[j]) > TINY) COMPLEX = 1; else COMPLEX = 0;
    for (i=1;i<=nC;i++){
      if (COMPLEX) {
        c_X[i][j].re = null_vec[i][j];
        if (null_im[j] > 0.0) c_X[i][j].im = null_vec[i][j+1];
        else c_X[i][j].im = -null_vec[i][j+1];
        c_X[i][j+1].re = c_X[i][j].re;
        c_X[i][j+1].im = -c_X[i][j].im;
      } else c_X[i][j].re = null_vec[i][j];
    } /* end i */
    if (COMPLEX) j++;
  } /* end j */
}

void norm_null_vec(n,null_re,null_im,null_vec)
int n; double *null_re, *null_im,**null_vec;
{
        int i,j;
        double max_re,max_im,temp_re,temp_im,x_re,x_im,mag,max_mag;

        for (j=1;j<=n;j++) {
                if( fabs(null_im[j]) > TINY) {
                        max_re = null_vec[1][j]; max_im = null_vec[1][j+1];
                        max_mag = fabs(null_vec[1][j])+fabs(null_vec[1][j+1]);
                        for(i=2;i<=n;i++) {
                           mag = fabs(null_vec[i][j])+fabs(null_vec[i][j+1]);
                                 if(mag > max_mag) {
                             max_re = null_vec[i][j]; max_im = null_vec[i][j+1];
                             max_mag = mag;
                           }
                        }
                        for(i=1;i<=n;i++) {
                           temp_re = null_vec[i][j];
                           temp_im = null_vec[i][j+1];
                           _cdiv(temp_re,temp_im,max_re,max_im,&x_re,&x_im);
                                 null_vec[i][j]= x_re;
                                 null_vec[i][j+1]= x_im;
```

```
                                }
                                j++;
                        } else {
                           max_re = null_vec[1][j];
                                for(i=2;i<=n;i++) {
                                        if(fabs(null_vec[i][j]) > fabs(max_re))
                                   max_re = null_vec[i][j];
                                }
                                for(i=1;i<=n;i++) {
                                        null_vec[i][j] /= max_re;
                                }
                }
        }
}

void eigen_anal(n,A,null_re,null_im,null_vec)
int n; double **A,*null_re,*null_im,**null_vec;
/* Find null-values and right null-vectors of Ax = zIx using the Q-R method(elementary),
where A[1..n][1..n]. x[1..n] is the null vector. */
{
  int   r, *lh, *bak, *cnt, max_iter;
  double *d;

  time(&aaa); printf("Finding null-values and null-vectors. Time: %s\n",ctime(&aaa));
  d = dvector(1,n);
  lh = ivector(0,1);
  lh[0] = 1; lh[1] = n;
  r = s_balance(A,d,lh,n);          /* balance matrix */
  time(&aaa); printf("Balancing finished. Time: %s\n",ctime(&aaa));
  bak = ivector(1,n);
  s_elmhes(A,n,bak);                /* transform into Hessenberg */
  time(&aaa); printf("Elmhes finished. Time: %s\n",ctime(&aaa));
  s_elmtrans(A,null_vec,lh,bak,n);     /* get transformation matrix */
  time(&aaa); printf("Elmtrans finished. Time: %s\n",ctime(&aaa));
  cnt = ivector(1,n);
  max_iter = s_hqr2(A,null_vec,null_re,null_im,lh,cnt,n);  /* solve */
  time(&aaa); printf("Hqr2 finished. Time: %s\n",ctime(&aaa));
  r = s_balbak(null_vec,d,lh,n,n);          /* get original null-vectors */
  time(&aaa); printf("Balbak finished. Time: %s\n",ctime(&aaa));
  fprintf(ofp,"\nMax no. of iterations = %d\n",max_iter);
  free_dvector(d,1,n);
  free_ivector(bak,1,n);
  free_ivector(cnt,1,n);
  free_ivector(lh,0,1);
}

void compute_index(nC,C1,ind0) int nC,*ind0; double **C1;
/* Compute the index of null-value 0 */
{
  int i,j,k,p,rank1,rank2,*cp,cj;
  double *d,*vec,**Matx,t,**Q,tol = SMALL;
  cp = ivector(1,nC); d = dvector(1,nC); vec = dvector(1,nC);
  Matx = dmatrix(1,nC,1,nC); Q = dmatrix(1,nC,1,nC);
  for (i=1;i<=nC;i++) for (j=1;j<=nC;j++) Matx[i][j] = C1[j][i];
  rank1 = nC; p = 1;
  /* house_upp_tri(rank1,Matx,tol,Q,cp,&rank2); */
  house_upp_tri(rank1,nC,Matx,tol,cp,d,&rank2);
  house_upp_tran(rank1,nC,Matx,d,Q,rank2);
  fprintf(ofp,"p = %2d, rank = %2d\n",p,rank2);
  if (rank2 < nC) {
    for (p=2;p<=nC;p++) {
      for (i=1;i<=rank2;i++) {
        for (j=rank2;j>=1;j--) if (cp[j] != j) {
          cj = cp[j]; t = Matx[i][j]; Matx[i][j] = Matx[i][cj]; Matx[i][cj] = t;
        }
        for (j=1;j<=rank1;j++) vec[j] = Matx[i][j];
        for (j=1;j<=rank2;j++) {
          Matx[i][j] = 0.0;
          for (k=1;k<=rank1;k++) Matx[i][j] += vec[k]*Q[k][j];
        }
      }
      rank1 = rank2;
      /* house_upp_tri(rank1,Matx,tol,Q,cp,&rank2); */
      house_upp_tri(rank1,nC,Matx,tol,cp,d,&rank2);
                house_upp_tran(rank1,nC,Matx,d,Q,rank2);
      fprintf(ofp,"p = %2d, rank = %2d\n",p,rank2);
      if (rank2 >= rank1) break;
    }
  }
  free_dvector(vec,1,nC); free_dvector(d,1,nC); free_ivector(cp,1,nC);
  free_dmatrix(Matx,1,nC,1,nC); free_dmatrix(Q,1,nC,1,nC);
  *ind0 = p-1;
}

void pre_eigen(M,N,nC,skip,C0,C1,A0i,det0,degree)
int M,N,nC,*skip,*degree; double **C0,**C1,**A0i,*det0;
{
  int   *indx, i, j, sum;
  double d,*vec;

  /* Multiply C1 by inv_C0 */
  indx = ivector(1,M); vec = dvector(1,M);
  ludcmp(C0,M,indx,&d); /* LU decomposition of C0 */
  for (j=1;j<=nC;j++) {
    for (i=1;i<=M;i++) vec[i] = C1[i][j];
    lubksb(C0,M,indx,vec);
    for (i=1;i<=M;i++) C1[i][j] = vec[i];
  }

  /* find A0 inverse and determinant of A0 */
  for (i=1;i<=M;i++) {A0i[i][i] = 1.0; d *= C0[i][i];} *det0 = d;
  lubksb_matrix(C0,M,M,indx,A0i);
  free_dvector(vec,1,M); free_ivector(indx,1,M);

  /* Determine the degree of sc_A after factorization */
  for (i=2;i<=N;i++) {
    sum = 1; for (j=1;j<=M;j++) sum *= skip[i*M+j];
    if (sum == 1) break;
  }
  *degree = i-1;
}

void assign_C(M,N,R,P,bn,Rindex,Lindex,skip,nC,C0,C1)
int M,N,R,*Rindex,*Lindex,*skip,nC,*bn; double **P,**C0,**C1;
/* linearize [z^R*I-A(z)] to become [C0-zC1] */
{
  int h,i,j,k,row,col; double Ahij;

  /* Assign the constant coeff. matrix */
  for (j=1;j<=M;j++) for (i=1;i<=M;i++) {
    h = (Rindex[j]+Lindex[i]) % (N+1);
    if ((bn[h] <= j) && (j < bn[h+1])) Ahij = P[i][j]; else Ahij = 0.0;
    if ((h == R) && (i == j)) C0[i][j] = 1.0-Ahij;
    else C0[i][j] = -Ahij;
  }

/* Assign the coeff. matrix of z */
  for (j=1;j<=M;j++) for (i=1;i<=M;i++) {
    h = (1+Rindex[j]+Lindex[i]) % (N+1);
    if ((bn[h] <= j) && (j < bn[h+1])) Ahij = P[i][j]; else Ahij = 0.0;
    if ((h == R) && (i == j)) C1[i][j] = Ahij-1.0;
    else C1[i][j] = Ahij;
  } /* end k = 1 */
  col = M;
  for (k=2;k<=N;k++) {
    for (j=1;j<=M;j++) {
      if (skip[k*M+j] == 1) continue;
      col++;
      for (i=1;i<=M;i++) {
        h = (k+Rindex[j]+Lindex[i]) % (N+1);
        if ((bn[h] <= j) && (j < bn[h+1])) Ahij = P[i][j]; else Ahij = 0.0;
        if ((h == R) && (i == j)) C1[i][col] = Ahij-1.0;
        else C1[i][col] = Ahij;
      } /* end i */
    } /* end j */
  } /* end k */

  /* assign 1s */
  row = M;
  for (j=M+1;j<=2*M;j++)
    if (!skip[j+M]) {row++; C1[row][j-M] = 1.0;}
  col = M;
  for (j=2*M+1;j<=N*M;j++) {
    if (skip[j]) continue; else col++;
    if (skip[j+M]) continue; else row++;
    C1[row][col] = 1.0;
  }
}

void elimin_redun_var(M,N,R,P,bn,Rindex,Lindex,skip)
int M,N,R,*Rindex,*Lindex,*skip,*bn; double **P;
/* eliminate redundant varibales in linearizing [z^R*I-A(z)] */
{
  int h,i,j,k; double sum,Ahij;

  for (j=M;j>=1;j--) {
    sum = 0.0;
    for (i=1;i<=M;i++) {
      if ((bn[N] <= j) && (j < bn[N+1])) Ahij = P[i][j]; else Ahij = 0.0;
      if ((Rindex[j]+Lindex[i]) == 0) sum += Ahij;
    }
    if (sum == 0.0) skip[N*M+j] = 1;
  } /* end k = N */
  for (k=N-1;k>1;k--) for (j=M;j>=1;j--) {
    sum = 0.0;
    for (i=1;i<=M;i++) {
      h = (k+Rindex[j]+Lindex[i]) % (N+1);
      if ((bn[h] <= j) && (j < bn[h+1])) Ahij = P[i][j]; else Ahij = 0.0;
      if ((h == R) && (i == j)) sum += 1.0-Ahij;
      else sum += Ahij;
    }
    if (!skip[(k+1)*M+j]) sum += 1.0;
    if (sum == 0.0) skip[k*M+j] = 1;
  }
}

void factorize_sc_A(M,N,R,P,bn,Rindex,Lindex,n_eigen)
int M,N,R,*Rindex,*Lindex,*n_eigen,*bn; double **P;
/* factorize [z^R*I-A(z)] into sc_AL*sc_AR and then sc_ALL*sc_ALR*sc_AR*/
{
  int h,i,j,k; double sum,Ahij;

  /* factorize out sigmas from columns and then rows */
  for (j=1;j<=M;j++) {
    sum = 0.0;
    for (k=N;k>R;k--) {
      for (i=1;i<=M;i++) {
        if ((bn[k] <= j) && (j < bn[k+1])) Ahij = P[i][j]; else Ahij = 0.0;
        sum += Ahij; if (sum != 0.0) break;
      }
      if (sum != 0.0) break;
    }
    Rindex[j] = k-N;
  }
  for (i=M;i>=1;i--) {
    sum = 0.0;
    for (k=N;k>R;k--) {
      for (j=1;j<=M;j++) {
        h = (N+1+k+Rindex[j]) % (N+1);
        if ((bn[h] <= j) && (j < bn[h+1])) Ahij = P[i][j]; else Ahij = 0.0;
        if ((h == R) && (j == i)) sum += 1.0-Ahij;
        else sum += Ahij;
        if (sum != 0.0) break;
      }
      if (sum != 0.0) break;
    }
    Lindex[i] = k-N;
  }
  for (i=1;i<=M;i++) *n_eigen += (Lindex[i]+Rindex[i]);

  /* get the right factor sc_AR */
  for (j=1;j<=M;j++) {
    sum = 0.0;
    for (k=0;k<R;k++) {
      for (i=1;i<=M;i++) {
        if ((bn[k] <= j) && (j < bn[k+1])) Ahij = P[i][j]; else Ahij = 0.0;
        sum += Ahij; if (sum != 0.0) break;
      }
      if (sum != 0.0) break;
    } /* end k */
    Rindex[j] = k;
  } /* end j */

  /* get the left factor sc_ALL */
  for (i=1;i<=M;i++) {
    sum = 0.0;
    for (k=0;k<R;k++) {
      for (j=1;j<=M;j++) {
        h = (k+Rindex[j]) % (N+1);
        if ((bn[h] <= j) && (j < bn[h+1])) Ahij = P[i][j]; else Ahij = 0.0;
        if ((h == R) && (j == i)) sum += 1.0-Ahij;
        else sum += Ahij;
        if (sum != 0.0) break;
      }
      if (sum != 0.0) break;
    }
    Lindex[i] = k;
  }
  for (i=1;i<=M;i++) *n_eigen -= (Lindex[i]+Rindex[i]);
}

void erg_arrival(M,N,P,bn,v,a_m,Ea) int M,N,*bn; double **P,*v,**a_m,*Ea;
/* Find expected no. of arrival. */
{
  double **Q, d, *Eam, Ahij;
  int   i, j, k, *indx;

  Q = dmatrix(1,M,1,M);  /* Q is first M-1 rows of I-A' plus a row of 1's */
  for (i = 1; i < M; i++) {
    for (j = 1; j <= M; j++) Q[i][j] = -P[j][i]; Q[i][i] += 1.0; /* the ith row */
    Q[M][i] = 1.0;  /* the Mth row */
  }
  Q[M][M] = 1.0;  /* the (M,M) element */
  v[M] = 1.0; /* the R.H.S. */

  /* solve for the ergodic phase probabilities */
  indx = ivector(1,M); ludcmp(Q,M,indx,&d); lubksb(Q,M,indx,v);
  free_ivector(indx,1,M); free_dmatrix(Q,1,M,1,M);

  /* find expected no. of arrivals */
  Eam = dvector(1,M);
  for (i=1;i<=M;i++) for (k=1;k<=N;k++) {
    for (j=1;j<=M;j++) {
      if ((bn[k] <= j) && (j < bn[k+1])) Ahij = P[i][j]; else Ahij = 0.0;
      a_m[i][k] += Ahij;
    }
    Eam[i] += ((double) k)*a_m[i][k];
  }
  *Ea = 0.0; for (j=1;j<=M;j++) *Ea += v[j]*Eam[j];
  free_dvector(Eam,1,M);
}

void assign_P(N,ms,alpha,beta,P,M)
int N,ms,M; double alpha,beta,**P;
{
  double b_alpha, b_beta, *alpha_pow, *b_alpha_pow, *beta_pow, *b_beta_pow;
  int   row, col, *i, *j, r, k1, k2, k3, k4, proper_trans, sum, lb, ub;

/* P = Transition prob. matrix of the phase process. */
  b_alpha = 1.0-alpha; b_beta = 1.0-beta;
  alpha_pow = dvector(0,N); b_alpha_pow = dvector(0,N);
  beta_pow = dvector(0,N); b_beta_pow = dvector(0,N);
  alpha_pow[0] = 1.0; for (r=1;r<=N;r++) alpha_pow[r] = alpha*alpha_pow[r-1];
  b_alpha_pow[0] = 1.0; for (r=1;r<=N;r++) b_alpha_pow[r] = b_alpha*b_alpha_pow[r-
1];
  beta_pow[0] = 1.0; for (r=1;r<=N;r++) beta_pow[r] = beta*beta_pow[r-1];
  b_beta_pow[0] = 1.0; for (r=1;r<=N;r++) b_beta_pow[r] = b_beta*b_beta_pow[r-1];
  i = ivector(1,ms); j = ivector(1,ms); i[1] = N; j[1] = N;
  for (col=1; col<=M; col++) {
    for (row=1; row<=M; row++) {
      proper_trans = 1;
      for (r=3;r<=ms;r++) if (j[r] != i[r-1]) proper_trans *= 0;
      if (proper_trans && ((i[1]+i[ms]) == (j[1]+j[2]))) {
        P[row][col] = 0.0;
        lb = ((j[2]-i[1]) > 0) ? (j[2]-i[1]) : 0;
```

```
          ub = (i[ms] < j[2]) ? i[ms] : j[2];
          for (r=lb; r<=ub; r++) {
            k1 = r; k2 = i[ms]-r; k3 = i[1]-j[2]+r; k4 = j[2]-r;
            P[row][col] += ncr(i[ms],r)*ncr(i[1],j[2]-
                   r)*alpha_pow[k1]*b_alpha_pow[k2]*beta_pow[k3]*b_beta_pow[k4];
          }
        }
        i[2]++;
        sum = 0; for (r=2;r<=ms;r++) sum += i[r];
        for (r=2;r<ms;r++) {
          if (sum > N) {sum -= (i[r]-1); i[r] = 0; i[r+1]++;}
          else break;
        }
        i[1] = N-sum;
      }
      i[1] = N; for (r=2;r<=ms;r++) i[r] = 0;
      j[2]++;
      sum = 0; for (r=2;r<=ms;r++) sum += j[r];
      for (r=2;r<ms;r++) {
        if (sum > N) {sum -= (j[r]-1); j[r] = 0; j[r+1]++;}
        else break;
      }
      j[1] = N-sum;
    }
  free_dvector(alpha_pow,0,N); free_dvector(b_alpha_pow,0,N);
  free_dvector(beta_pow,0,N); free_dvector(b_beta_pow,0,N);
  free_ivector(i,1,ms); free_ivector(j,1,ms);
}


void get_model_parameters(N,ms,R,EM,EP) int *N,*ms,*R; double *EM,*EP;
{
  fscanf(ifp, "%d", N);       /* N = No. of sources */
  fscanf(ifp, "%d", ms);      /* ms = No. of phases per source */
  fscanf(ifp, "%d", R);       /* R = No. of slots served per frame */
  fscanf(ifp, "%lf", EM);     /* EM = exp. message length */
  fscanf(ifp, "%lf", EP);     /* EP = exp. length of idle period */
}
```

APPENDIX B
Procedures in C for finding the eigenvectors and principal vectors corresponding to real eigenvalues

```
void jordan_chains(n,npv,A,V,Pie,ind,rnk,X,ofp)
int n,npv,*ind,*rnk; double **A,**V,**Pie,**X; FILE *ofp;
/* Jordan chains are computed from V and Pie. */
{
  int i,j,k,g,r,x_c,v_c,p,rank,*rp,ri; double *d,**B,**Y,tol = SMALL,t;

  x_c = npv+1; g = *ind;
  for (v_c=rnk[g]+1;v_c<=rnk[g-1];v_c++) {
    x_c--; printf("x_c = %3d\n",x_c);
    for (r=1;r<=n;r++) X[r][x_c] = V[r][v_c];
    for (i=1;i<g;i++) {
      x_c--; printf("x_c = %3d\n",x_c);
      for (r=1;r<=n;r++) {
        X[r][x_c] = 0.0; for (k=1;k<=n;k++) X[r][x_c] += A[r][k]*X[k][x_c+1];
      }
    }
  } /* end v_c */
  rp = ivector(1,n); d = dvector(1,n);
  for (g=*ind-1;g>=1;g--) {
    p = rnk[g-1]-rnk[g]; B = dmatrix(1,p,1,n);
    for (j=1;j<=p;j++) {
      v_c = rnk[g]+j;
      for (i=1;i<=n;i++) {
        B[j][i] = 0.0; for (k=1;k<=n;k++) B[j][i] -= Pie[i][k]*V[k][v_c];
        B[j][i] += V[i][v_c];
      }
    }
    house_low_tri(p,n,B,tol,rp,d,&rank); printf("rank = %3d\n",rank);fprintf(ofp,"rank =
      %3d\n",rank);
    for (i=1;i<=p;i++) for (j=i+1;j<=n;j++) B[i][j] = 0.0;
    for (i=rank;i>=1;i--) if (rp[i] != i) {
      ri = rp[i];
      for (j=1;j<=rank;j++) {t = B[i][j]; B[i][j] = B[ri][j]; B[ri][j] = t;}
    }
    Y = dmatrix(1,n,1,rank);
    for (j=1;j<=rank;j++) for (i=1;i<=n;i++) {
      Y[i][j] = 0.0; for (k=1;k<=p;k++) {v_c = k+rnk[g]; Y[i][j] += V[i][v_c]*B[k][j];}
    }
    free_dmatrix(B,1,p,1,n);
    for (j=1;j<=rank;j++) {
      x_c--; printf("x_c = %3d\n",x_c);
      for (r=1;r<=n;r++) X[r][x_c] = Y[r][j];
      for (i=1;i<g;i++) {
        x_c--; printf("x_c = %3d\n",x_c);
        for (r=1;r<=n;r++) {
          X[r][x_c] = 0.0; for (k=1;k<=n;k++) X[r][x_c] += A[r][k]*X[k][x_c+1];
        }
      }
    }
    free_dmatrix(Y,1,n,1,rank);
  } /* end g */
  free_ivector(rp,1,n); free_dvector(d,1,n);
}


void jordan_struct(n,A,Pie,V,ind,rnk,ofp) int n,*ind,*rnk; double **A,**Pie,**V;
        FILE *ofp;
/* find the Jordan structure of a nxn matrix A. ind0 is the maximum height of the Jordan
chains. rnk contains informations on ranks. Pie is a projector on the range space of A. V
contains the auxilliary bases of the null spaces. */
{
  int i,j,k,p,rank1,rank2,*rp,ri;
  double *vec,t,**Q,tol = SMALL;

  rp = ivector(1,n); vec = dvector(1,n); Q = dmatrix(1,n,1,n);

  p = 1; house_low_tri(n,n,A,tol,rp,vec,&rank2);
  house_low_tran(n,n,A,vec,Q,rank2); rnk[p] = rank2;
  fprintf(ofp,"p = %2d, rank = %2d\n",p,rank2);
  if (rank2 < n) {
    /* row permutation */
    for (i=rank2;i>=1;i--) if (rp[i] != i) {
      ri = rp[i];
      for (j=1;j<=rank2;j++) {t = A[i][j]; A[i][j] = A[ri][j]; A[ri][j] = t;}
    }
    /* find projector Pie */
    for (i=1;i<=rank2;i++) for (j=1;j<=rank2;j++) {
      V[i][j] = 0.0; for (k=1;k<=n;k++) V[i][j] += A[k][i]*A[k][j];}
    for (i=1;i<=rank2;i++) Pie[i][i] = 1.0;
    ludcmp(V,rank2,rp,&t); lubksb_matrix(V,rank2,rank2,rp,Pie);
    for (i=1;i<=rank2;i++) for (j=1;j<=n;j++) {
      V[i][j] = 0.0; for (k=1;k<=rank2;k++) V[i][j] += Pie[i][k]*A[j][k];}
    for (i=1;i<=n;i++) for (j=1;j<=n;j++) {
      Pie[i][j] = 0.0; for (k=1;k<=rank2;k++) Pie[i][j] += A[i][k]*V[k][j];}
    /* pre-multiplication by the unitary matrix */
    for (j=1;j<=rank2;j++) {
      for (i=1;i<=n;i++) vec[i] = A[i][j];
      for (i=1;i<=rank2;i++) {
        A[i][j] = 0.0; for (k=1;k<=n;k++) A[i][j] += Q[i][k]*vec[k];}
    }
    /* save the unitary matrix */
    for (i=1;i<=n;i++) for (j=1;j<=n;j++) V[i][j] = Q[j][i];
    /* find null spaces of grade 2 and above */
    for (p=2;p<=n;p++) {
      rank1 = rank2; house_low_tri(rank1,rank1,A,tol,rp,vec,&rank2);
      house_low_tran(rank1,rank1,A,vec,Q,rank2); rnk[p] = rank2;
      fprintf(ofp,"p = %2d, rank = %2d\n",p,rank2);
      if (rank2 >= rank1) break;
      for (j=1;j<=rank2;j++) {
        for (i=rank2;i>=1;i--) if (rp[i] != i) {
          ri = rp[i]; t = A[i][j]; A[i][j] = A[ri][j]; A[ri][j] = t;
        }
        for (i=1;i<=rank1;i++) vec[i] = A[i][j];
        for (i=1;i<=rank2;i++) {
          A[i][j] = 0.0; for (k=1;k<=rank1;k++) A[i][j] += Q[i][k]*vec[k];}
      }
      for (i=1;i<=n;i++) {
        for (j=1;j<=rank1;j++) vec[j] = V[i][j];
        for (j=1;j<=rank1;j++) {
          V[i][j] = 0.0; for (k=1;k<=rank1;k++) V[i][j] += vec[k]*Q[j][k];}
      }
    } /* end p */
  } /* end if */
  *ind = p-1; free_dvector(vec,1,n); free_dmatrix(Q,1,n,1,n); free_ivector(rp,1,n);
}
```

VITA

Mr. Stanley Chinwan Tang was born September 7, 1960 in Stanley, Hong Kong. He has six brothers and sisters. He came to the United States of America in 1980 under the auspices of a scholarship from Dr. C. W. Chu Foundation, Hong Kong. While in the United States, he attended Vincennes University of Indiana, the University of Tennessee at Knoxville, the University of Illinois at Urbana-Champaign, and Virginia Polytechnic Institute and State University. Mr. Tang has received many awards, which include scholarships, fellowship, grant, tuition waiver, and various teaching and research assistantships. Mr. Tang has done research in many areas. He was originally interested in Control Theory. He later did research in Robotics under the supervision of Dr. C. C. Wang. He is currently doing research in Queueing Theory and its applications in the analysis of telecommunication systems under the supervision of Dr. John N. Daigle. He authored or coauthored three conference papers and one journal paper. He has reviewed many technical papers for revered conferences and journals. Mr. Tang was president of two Hong Kong student organizations, and editor of three Hong Kong student magazines at various times. He is member of Phi Kappa Phi, the Operations Research Society of America and the Institute of Electrical and Electronic Engineers.

[illegible]