1987

Some Efficient Computational Algorithms Related to Phase Models

Vernon J. Rego
Purdue University, rego@cs.purdue.edu

Report Number:
87-727

Rego, Vernon J., "Some Efficient Computational Algorithms Related to Phase Models" (1987). *Department of Computer Science Technical Reports*. Paper 628.
<https://docs.lib.purdue.edu/cstech/628>

SOME EFFICIENT COMPUTATIONAL ALGORITHMS
RELATED TO PHASE MODELS

Vernon Rego

CSD-TR-727
December 1987

Some Efficient Computational Algorithms Related to Phase Models

Vernon Rego *

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

Abstract: This paper develops efficient computational algorithms for some models that utilize phase type distributions. As in other instances, such as matrix-geometric methods, where the PH-distribution lends itself to useful and efficient computational methods, we examine how certain probabilities based on these distributions may be efficiently computed. The examples that we consider generally pertain, but are not restricted, to modelling applications such as token ring networks and multiqueues in communication systems. By exploiting the geometric nature of PH-densities, it is shown that certain exponential time computations can be reduced to linear time, if recursive algorithms are used. We demonstrate speedup with the aid of an example, using a recursive as well as a nonrecursive algorithm.

* This research was supported in part by the National Science Foundation under grant NCR-8702115 and in part by a 1987 Purdue XL grant.

1. Introduction and Motivation

Practitioners of stochastic modelling in communications system design are well aware of the tremendous difficulties inherent to multiqueuing systems, the determination of their stability, and analysis of their behaviour. When faced with concerns of dimensionality, or dependence between key random variables, it is not uncommon for a researcher to look for approximate or even asymptotic methods that would yield some insight into system behaviour. It is perhaps often that some useful computational stochastic models are overlooked in favour of simulations or asymptotics, due to a (sometimes cursory) sense of their high intrinsic dimensionality.

Simulations of multidimensional queuing systems such as communication systems are usually not complete and sometimes impossible, without knowledge of the stable operating region of the system under consideration. Further, it is one matter to *have knowledge* of a system's stability condition, and quite another matter to be able to *tell* if a given system is operating in a stable fashion, for given system parameters. In most interesting systems, especially interference systems and systems with mixed (i.e., finite and infinite) buffers, the latter usually requires the computation of certain joint probabilities. Examples of situations where these joint probabilities are required and also examples of (rare) cases where such joint probabilities may be ignored can be seen in [5]. Besides being important for ascertaining system stability, there are several reasons (for example, see [2], [3]) why computational models are important. Albeit dimensionally small, one useful computational example of an otherwise intractable class of token ring problems can be found in [4].

Our work is strongly motivated by the need for efficient algorithmic methods of the kind pioneered by Neuts [3] in stochastic modelling. In this paper, we address computational algorithms for models that use the well known phase-type distributions [3] in multidimensional settings. In each of the following three sections, we give examples of situations where certain probabilities in multidimensional systems can be computed very efficiently through the use of

Figure 1. Single Server Queue

Figure 2. Asymmetric Token Ring

recurrence equations. In order to demonstrate the amazing degree of speedup obtained, we give the run-times of two different algorithms used to solve the same problem, for various parameters. The techniques used to obtain the fast, recursive algorithms, are mainly elementary generating function based methods, of the kind used in a somewhat similar application by Moore [1].

2. From Exponential to Linear Time

In an $m+1$ state Markov chain on states $\{1, 2, \dots, m+1\}$, assume that states $1, 2, \dots, m$ are all transient, so that absorption into state $m+1$ from any initial state, is certain. Assuming the chain operates in continuous (discrete) time and the initial vector is (α, α_{m+1}) , the *time to absorption* in the chain is known [3] as a continuous (discrete) *phase type* random variable.

In the continuous case, the probability distribution $F(\cdot)$ of this random variable is given by

$$F(x) = 1 - \alpha \exp(Tx)e \quad (2.1)$$

for $x \geq 0$, and in the discrete case, the probability density $\{p_k; k \geq 0\}$ of this random variable is given by

$$\begin{aligned} p_0 &= \alpha_{m+1} \\ p_m &= \alpha T^{k-1} \beta, \quad k \geq 1 \end{aligned} \quad (2.2)$$

where in each case (α, α_{m+1}) is an initial vector and T, β are an appropriately defined $m \times m$ matrix and $m \times 1$ vector, respectively. A complete description can be found in the appendix. In each case, (α, T) is said to be a representation of the PH distribution (i.e., phase type distribution) or PH density, respectively.

We next need a useful result relating queues, continuous PH random variables and discrete PH random variables. In order to motivate our discussion, consider the scene depicting a single server queueing situation shown in Figure 1. Assuming Poisson arrivals and continuous PH

service-times, this queue is an M/PH/1 queue. In keeping with the usual departure instant based queueing chain construction for this M/GI/1 queue, define

$$\begin{aligned} a_k &= \text{probability of } k \text{ customer arrivals in a service time} \\ &= \int_0^\infty \frac{e^{-\lambda t} (\lambda t)^k}{k!} dF(t) \end{aligned}$$

for $k \geq 0$, with λ as arrival rate. The following result (see [3]) shows how the probability density $\{a_k; k \geq 0\}$ can be computed *without* numerical integration for a large class of distributions $F(\cdot)$.

Lemma 1

If $F(\cdot)$ is a continuous PH distribution with the representation (κ, K) , then $\{a_k; k \geq 0\}$ is a discrete PH density with representation (α, T) , where $\alpha = \lambda \kappa(\lambda I - K)^{-1}$, $T = \lambda(\lambda I - K)^{-1}$, $\alpha_{m+1} = \kappa_{m+1} + \alpha(\lambda I - K)^{-1} K^0$, and $T^0 = (\lambda I - K)^{-1} K^0$.

In applying the above result, only one computation of the matrix inverse is necessary and this does not pose a problem. In general, it is required that T be substochastic, and thus $(I - T)$ nonsingular.

2.1 An Exponential Algorithm and Two Examples

The extremely helpful result (as we shall see) given as Lemma 1 is our point of departure into some inefficient and efficient computational methods. To begin with, the quantity

$$C(n) = \sum_{D(n)} \prod_{k=1}^N \left[I_{\{n_k = 0\}} \alpha_{k,m+1} + I_{\{n_k > 0\}} \alpha_k T_k^{n_k-1} \beta_k \right] \quad (2.4)$$

defined with $D(n) = \{(n_1, \dots, n_N) | n_j \geq 0, 1 \leq j \leq N, \sum_{j=1}^N n_j = n\}$, is both useful and easy to compute, provided one constructs the computational procedure with some care. To demonstrate what we mean by this, consider the following brute-force algorithm for $C(n)$:

```

C := 0;
for n1 := 0 to n
.
.
.
for nN := 0 to n
begin
    if ((n1 + ... + nN) .eq. (n)) then
        C := C + p(n1) * ... * p(nN);
    end;
end;
.
.
.
end.

```

(2.5)

where

$$p(n_j) = \begin{cases} \alpha_{j,m+1} & n_j = 0 \\ \alpha_j T_j^{n_j-1} \beta_j & n_j \geq 1 \end{cases} \quad (2.6)$$

for $1 \leq j \leq N$. Clearly the above algorithm has a run time complexity that is exponential in N , since it examines all possible decompositions of the number n into N integer summands, taking order into account as well. Before showing how the algorithm in (2.5) can be redesigned to yield a complexity of $O(N)$, we first demonstrate two applications of the quantity $C(n)$ in (2.4).

Example 1: Token Ring Models

Consider an N station token ring (see Figure 2), with stations numbered 1 to N , where station N has an infinite capacity buffer and all other stations have unit capacity buffers. Assume that packets arrive at station j according to a Poisson process of rate λ_j , and arrivals at different stations are independent. Packet transmissions from each station j require a random time X_j , distributed as $B_j(\cdot)$, $1 \leq j \leq N$. The token is passed around the ring unidirectionally, and stations capturing the token are allowed to transmit at most one packet per token capture. We assume that it takes the token a small random time Y_j , with distribution $S_j(\cdot)$, to switch past an empty station

j , $1 \leq j \leq N$. Additionally, the token passing time from station j to station $k = (j \bmod N) + 1$, is a random time W_k , distributed as $V_k(\cdot)$, $1 \leq j \leq N$.

In the M/SM/1 (Poisson arrival, semi-Markov service time) based token ring model developed in [4], an embedded semi-Markov chain is defined as follows. Let t_j^m , $1 \leq j \leq N$ denote the m^{th} token scan-instant at station j for $m \geq 0$. With the convention $t_1^m < t_2^m < \dots < t_N^m < t_1^{m+1} < \dots$, let $Z_j^{(m)}$ denote the number of packets found queued by the token at station j at time t_j^m . Then $\{Z_n\}$, $Z_n \triangleq \langle Z_1^n, Z_2^n, \dots, Z_N^n \rangle$ is an embedded semi-Markov chain (see [4] for details). Standard arguments lead to the M/G/1 block structured matrix transition probability matrix [3],

$$P = \begin{bmatrix} B_0 & B_1 & B_2 & B_3 & \dots \\ A_0 & A_1 & A_2 & A_3 & \dots \\ 0 & A_0 & A_1 & A_2 & \dots \\ 0 & 0 & A_0 & A_1 & \dots \\ 0 & 0 & 0 & A_0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \quad (2.7)$$

describing a (queue length, cycle-type) process seen at token scans of station N . Here $A_k = A_k(\cdot | \cdot)$, and $B_k = B_k(\cdot | \cdot)$ are $2^{N-1} \times 2^{N-1}$ substochastic matrices, given by

$$\begin{aligned} B_k(z' | z) &= P[Z_{n+1} = \langle z', k \rangle | Z_n = \langle z, 0 \rangle] \\ A_{i+k-1}(z' | z) &= P[Z_{n+1} = \langle z', k \rangle | Z_n = \langle z, i \rangle] \end{aligned} \quad (2.8)$$

with $z = \langle z_1, \dots, z_{N-1} \rangle$, $z' = \langle z'_1, \dots, z'_{N-1} \rangle$ as $(N-1)$ bit binary vectors, and $i > 0$. It can be shown that the queue at station N is stable if and only if

$$\rho = \pi \left(\sum_{k=0}^{\infty} k A_k \right) e < 1. \quad (2.9)$$

where π is the left invariant vector of the stochastic matrix $A = \sum_{k=0}^{\infty} A_k$, and e is an m -vector with all entries set to 1. Further, if $B_j(\cdot)$ and $V_j(\cdot)$ are continuous PH distributions with representations (b_j, B_j) and (v_j, V_j) , respectively, for $1 \leq j \leq N$, then it can be shown [4], using Lemma 1, that

$$A_n(<0, \dots, 0|z) = \eta \sum_{E(n)} \left\{ \prod_{k=1}^N \left[I_{\{n_k=0\}} \alpha_{k,m+1} + I_{\{n_k>0\}} \alpha_k T_k^{n_k-1} T_k^0 \right] \right. \\ \left. \left[I_{\{n_{N+k}=0\}} \alpha_{N+k,m+1} + I_{\{n_{N+k}>0\}} \alpha_{N+k} T_{N+k}^{n_{N+k}-1} T_{N+k}^0 \right] \right\} \quad (2.10)$$

where $E(n) = \{(n_1, \dots, n_{2N}) | n_j \geq 0, 1 \leq j \leq 2N, \sum_{j=1}^{2N} n_j = n\}$. From Lemma 1 and straightforward transition based arguments (see [4]), we have

$$\eta = \prod_{k=1}^{N-1} \delta_{k,m+1} \cdot \gamma_{k,m+1} \quad (2.11)$$

where

$$\delta_{k,m+1} = z_k [b_{k,m+1} + b_k B_k^0] + (1 - z_k) [s_{k,m+1} + s_k S_k^0] \\ \gamma_{k,m+1} = [v_{k,m+1} + v_k V_k^0], \text{ for } 1 \leq k \leq N \quad (2.12)$$

and

$$T_k = c_k [\lambda_N (\lambda_N I - z'_k B_k + (1 - z'_k) S_k)^{-1}], \quad c_k = \lambda_N / \left[\sum_{j=k \bmod N + 1}^N \lambda_j \right] \\ \alpha_k = z'_k [\lambda_N b (\lambda_N I - B_k)^{-1}] + (1 - z'_k) [\lambda_N s (\lambda_N I - S_k)^{-1}], \\ \alpha_{k,m+1} = z'_k [b_{k,m+1} + b (\lambda_N I - B_k)^{-1} B_k^0] + (1 - z'_k) [s_{k,m+1} + s (\lambda_N I - S_k)^{-1} S_k^0] \\ T_k^0 = z'_k [\lambda_N I - B_k)^{-1} B_k^0] + (1 - z'_k) [(\lambda_N I - S_k)^{-1} S_k^0] \quad (2.13)$$

for $1 \leq k \leq N - 1$, and finally, $T_N, \alpha_N, \alpha_{N,m+1}, T_N^0$ defined just as in (2.13), but with z_N in place

of z'_N , and z_N set equal to one (since the A submatrices deal with a nonempty queue at station N). Similarly, to account for terms $N+k$, $1 \leq k \leq N$, we have

$$\begin{aligned} T_{N+k} &= c_k \lambda_N (\lambda_N I - V_k)^{-1}, \alpha_{N+k} = \lambda_N v_k (\lambda_N I - V_k)^{-1}, \\ \alpha_{N+k,m+1} &= v_{k,m+1} + v (\lambda_N I - V_k)^{-1} V_k^0, \text{ and} \\ T_{N+k}^0 &= (\lambda_N I - V_k)^{-1} V_k^0 \end{aligned} \quad (2.14)$$

Remarks:

- (1). The notation $\prod_{k=N}^1 L_k$, which will appear later, is used to denote the *downward* product $L_N \cdot L_{N-1} \cdots L_1$. While the differentiation between a downward and an upward product is not important in the scalar case, the noncommutativity of matrices requires this distinction when L_k is a matrix, $1 \leq k \leq N$.
- (2). In general, for such M/GI/1 block structured transition matrices, it is required that each submatrix $A_k, B_k, k \geq 0$, be readily computable, so that P in (2.7) is "known". The usual iterative methods for steady state solutions [3] do not require that all of these submatrices be stored simultaneously. Only one, or at most a small number, is needed during iterative computation. The expression in (2.10) tells us how to compute a single element in each matrix $A_n, n \geq 0$, namely the entry $A_n(<0, \dots, 0>|z)$, for an arbitrary $(N-1)$ bit binary vector z . In [4] it is shown how the entire matrix $A_n(\cdot|\cdot), n \geq 0$, can be generated once this entry is obtained. For completeness, we have included the complete algorithm in the appendix. Observe that the submatrices $B_n(\cdot|\cdot)$ are obtained in identical fashion, except that now, in defining (2.10), the quantities $T_N, \alpha_N, \alpha_{N,m+1}$ and T_N^0 are defined using z_N in place of z'_N , and z_N set equal to zero (since the B submatrices deal with an empty queue at

Figure 3. N Independent Queues

station N).

- (3). The stability condition in (2.9) can be computed in a simpler manner (see [4]). However, the problem of computing expressions like (2.9) or similar expressions (say, higher order moments), for such matrix sequences, still remains. This demonstrates the value of an expression such as $C(n)$.

It should be clear at this stage that the constant $C(n)$ as defined in (2.4) is precisely the expression in (2.10), i.e., a transition probability, where N is replaced by $2N$, and T_k^0 is denoted by β_k . Observe that a way to tell if the queue at station N is stable is to compute the expression for ρ in (2.9). Since it will always be the case that $A_k, k \geq m$, is a zero matrix for some number $m, m > 0$, the summation in (2.9) is accurately given by a finite sum. However, the number m may be large, and this makes the computation of ρ time-consuming, especially for larger values of the index k .

Example 2: N Independent Queues

Consider the scene in Figure 3, where customers from N independent Poisson processes, with rates $\lambda_1, \lambda_2, \dots, \lambda_N$, arrive at N independent queueing stations, respectively. If the service time at queue $k, 1 \leq k \leq N$, is a continuous PH random variable with distribution $F_k(\cdot)$, then Lemma 1 tells us that the density of the random variable describing the number of customer arrivals at queue k in a service time is a discrete PH density, say with representation $\{\alpha_k, T_k\}$.

Let X_k be the random variable denoting the number of customer arrivals at queue $k, 1 \leq k \leq N$, during a service time at queue k . Further, let $X = \sum_{k=1}^N X_k$. Then we have that

$$\begin{aligned} P[X = n] &= \sum_{D(n)} P[X_1 = n_1, X_2 = n_2, \dots, X_N = n_N] \\ &= \sum_{D(n)} \prod_{k=1}^N [I_{\{n_k = 0\}} \alpha_k + I_{\{n_k > 0\}} \alpha_k T_k^{n_k - 1} \beta_k] \end{aligned} \quad (2.15)$$

which is just $C(n)$ as given by (2.4). ■

Having given some motivation for computing $C(n)$ and showing that the run time complexity of a brute force algorithm is (exponentially) bad, we now show, using generating function methods (for example, see [1] or [6]), that $C(n)$ can be computed extremely efficiently with some care. The idea is to exploit the relation between $C(n)$ and its generating function, since $C(n)$ is precisely a sum of terms that is the coefficient of the n^{th} power of the generating function variable in its series representation.

Theorem 1

The quantity

$$C(n) = \sum_{D(n)} \prod_{k=1}^N [I_{\{n_k=0\}} \alpha_{k,m+1} + I_{\{n_k>0\}} \alpha_k T_k^{n_k-1} \beta_k]$$

where (α_k, T_k) is a discrete PH density of order m with initial vector $(\alpha_k, \alpha_{k,m+1})$, and $D(n) = \{(n_1, \dots, n_N) \mid n_j \geq 0, 1 \leq j \leq N, \sum_{j=1}^N n_j = n\}$, is given by $\alpha_N C_N(n) \beta_1$, via the recurrence

$$C_i(j) = T_i C_i(j-1) + d_i A_i C_{i-1}(j) + A_i [I - d_i A_i T_i] C_{i-1}(j-1) \quad (2.16)$$

for $1 \leq i \leq N, 1 \leq j \leq n$, where

$$C_1(j) = T_1^{j-1} \text{ for } j \geq 1 \quad (2.17)$$

$$C_i(0) = d_i A_i \cdots d_1 A_1 \text{ for } 1 \leq i \leq N, \quad (2.18)$$

$A_1 = I, A_i = \beta_i \alpha_{i-1}$ for $2 \leq i \leq N$, and $d_i = \frac{\alpha_{i,m+1}}{\alpha_i \beta_i}$ for $1 \leq i \leq N$.

The run time complexity of this computation is $O(Nn)$, where N is the number of PH densities considered.

Proof:

Define the generating function

$$G(z) = (d_N I + Iz + T_N z^2 + \cdots) A_N (d_{N-1} I + Iz + T_{N-1} z^2 + \cdots) A_{N-1} \cdots (d_1 I + Iz + T_1 z^2 + \cdots) A_1 \quad (2.19)$$

with $A_i = \beta_i \alpha_{i-1}$, $2 \leq i \leq N$, $A_1 = I$, and $d_i = \frac{\alpha_{i,m+1}}{\alpha_i \beta_i}$, $1 \leq i \leq N$. It is instructive to note that the coefficient of z^0 in $G(z)$ is the product $d_N A_N d_{N-1} A_{N-1} \cdots d_1 A_1$, and in general, the coefficient of z^n in $G(z)$, when multiplied on the left by α_N , and on the right by β_1 , is the sum of all terms of the form $\prod_{k=N}^1 [I_{\{n_k=0\}} \alpha_{k,m+1} + I_{\{n_k>0\}} T_k^{n_k-1} A_k]$, where $n_1 + n_2 + \cdots + n_N = n$, with precisely one term for each such configuration of $(n_1, \dots, n_N)$. Clearly, we are interested in the coefficient of z^n in $G(z)$.

Defining $C(k)$ to be the coefficient of z^k in $G(z)$, we see that

$$G(z) = \sum_{k=0}^{\infty} C(k) z^k \quad (2.20)$$

can alternately be viewed as the generating function of the sequence $\{C(k); k \geq 0\}$. Since each T_j is substochastic, we can write

$$(d_j I + Iz + T_j z^2 + \cdots) = d_j I + z(I - zT_j)^{-1} \quad (2.21)$$

for $1 \leq j \leq N$.

Defining the product

$$G_i(z) = [d_i I + z(I - zT_i)^{-1}] A_i \dots [d_1 I + z(I - zT_1)^{-1}] A_1 \quad (2.22)$$

for $1 \leq i \leq N$, we see that $G_N(z) = G(z)$. Using the generating functional form,

$$G_i(z) = \sum_{k=0}^{\infty} C_i(k) z^k \quad (2.23)$$

for $1 \leq i \leq N$, so that $C(n) = C_N(n)$. Using (2.22), one can develop a recurrence

$$G_i(z) = d_i A_i G_{i-1}(z) + z(I - zT_i)^{-1} A_i G_{i-1}(z) \quad (2.24)$$

for $2 \leq i \leq N$, with $G_1(z)$ given by (2.22). Next, left multiplying both sides of (2.24) by $(I - zT_i)$ yields

$$G_i(z) = zT_i G_i(z) + d_i A_i G_{i-1}(z) - zT_i d_i A_i G_{i-1}(z) + zA_i G_{i-1}(z) \quad (2.25)$$

which, when related to (2.23) gives the recurrence

$$C_i(j) = T_i C_i(j-1) + d_i A_i C_{i-1}(j) - d_i T_i A_i C_{i-1}(j-1) + A_i C_{i-1}(j-1) \quad (2.26)$$

for $1 \leq i \leq N$, and $1 \leq j \leq n$. From $G_1(z)$ we obtain $C_1(j) = T_1^{j-1}$ for $j \geq 1$, and since the coefficient of z^0 in $G(z)$ is the product $d_N A_N \dots d_1 A_1$, we obtain $C_i(0) = d_i A_i \dots d_1 A_1$ for $1 \leq i \leq N$. The required quantity $C(n)$ is finally given by $\alpha_N C_N(n) \beta_1$. ■

The recursive scheme outlined in Theorem 1 computes terms of the form $C(i, j)$, where i and j are the indices of a rectangular matrix. The number of such terms is bounded by the quantity $N(n+1)$, since $1 \leq i \leq N$ and $0 \leq j \leq n$. Observe that except for d_i , all quantities involved in the recursion are $m \times m$ matrices. For a given value of n , $n \geq 0$, it should be clear from the above recursive scheme that the run time complexity is $O(Nn)$ (i.e., increases linearly as the product (N, n)).

2.2 Computational Results

In this section, we briefly report on our computational experience with the stability condition in (2.9) and the run-times of the brute-force algorithm (see (2.5)) and the recursive algorithm in Theorem 1. In Table 1, we have shown a comparison of the running-times of the brute-force and the recursive algorithm for 2×2 matrices T_k , $1 \leq k \leq N$, with $N = 3$. The table displays running-times in cpu seconds (i.e., time to compute *each entry* in A_n) as measured by utility *dtime* (on a VAX 8600 running 4.3 BSD Unix) versus n . It should be clear from this that the recursive scheme is vastly superior for the computation of $C(n)$.

n	Brute-force Alg.	Recursive Alg.
1	—	—
5	0.01667	—
10	0.11667	—
15	0.55000	—
20	1.60000	—
30	7.95000	0.03333
40	24.03333	0.03333
50	57.51667	0.050000
60	117.81667	0.050000
70	216.18333	0.050000
80	365.68332	0.050000
90	579.04999	0.06667
100	871.96667	0.06667

Table 1.

In order to study the convergence properties of the sequence $\{A_n; n \geq 0\}$, define a sequence of matrices $A(R)$, with $A(R) = \sum_{n=0}^R A_n$. Noting that the stochastic matrix $A = \lim_{R \rightarrow \infty} A(R)$, define the sequence $\{\chi(R); R \geq 0\}$ of error terms with $\chi(R) = \|A - A(R)\|$. Here, we take $\|B\| = \max\{B(i,j); 1 \leq i,j \leq M\}$, for any order M matrix B . In Figure 4, we show that the relationship between λ_N (i.e., the customer arrival rate at station N , with $N = 3$) and R (i.e., the

Figure 4. Arrival rate versus R for different stochasticity tolerance levels

Figure 5. Convergence of $\chi(\mathcal{R})$
as a function of $\mathcal{R}$

largest possible number of customer arrivals that we need to consider in order that $A(R)$ is a strictly stochastic matrix) is strictly linear. We agree to call a matrix $A(R)$ strictly stochastic if $\chi(R) < \delta$, where delta is some specified tolerance level. Figure 4 shows a linear relationship between R and λ_3 for different values of δ . In Figure 5, we display the manner in which $\chi(R)$ converges to zero as R increases for two different values of λ_N .

Clearly, the amount of work involved in obtaining the stability boundary for station N 's queue is not much (ie., given by (2.9)). However, attempting to tell if station N 's queue will be stable for a given set of system parameters is decidedly a nontrivial task, since ρ must now be computed. We have shown that use of the brute-force algorithm is terribly impractical in view of the convergence behaviour shown in Figures 4 and 5. Additionally, it is easy to choose values of λ_N (i.e., at the high end) for which the cost of using the brute-force algorithm is prohibitive.

2.3 Some Extensions

Suppose now that we wish to compute the probability that $n_N \geq 1$, given that $n_1 + n_2 + \dots + n_N = n$. Being a conditional probability, this is given by

$$P[n_N \geq 1 | n_1 + \dots + n_N = n] = \frac{P[n_N \geq 1, n_1 + \dots + n_N = n]}{P[n_1 + n_2 + \dots + n_N = n]} \quad (2.27)$$

The denominator on the RHS of (2.27) is precisely $C_N(n)$. The numerator, a joint probability, is computed via a small modification of Theorem 1. While $G_i(z)$ in (2.22) remains unchanged for $1 \leq i \leq N-1$, when $i = N$ we obtain

$$G_N(z) = [z(I - zT_N)^{-1}]A_N G_{N-1}(z) \quad (2.28)$$

since $d_N = 0$ in this case. Consequently, the recurrence for $C_i(j)$ [see (2.26)] remains unchanged, i.e., $C_i(j | n_N \geq 1) = C_i(j)$ for $1 \leq j \leq n$, $1 \leq i < N$, while

$$C_N(j | n_N \geq 1) = T_N C_N(j-1 | n_N \geq 1) + A_N C_{N-1}(j-1 | n_N \geq 1) \quad (2.29)$$

for $i \leq j \leq N$. The above modification gives us $C_N(n | n_N \geq 1)$, and consequently

$$P[n_N \geq 1 | n_1 + \dots + n_N = n] = \frac{\alpha_N C_N(n | n_N \geq 1) \beta_1}{\alpha_N C_N(n) \beta_1} \quad (2.30)$$

The same idea can be used to obtain $P[n_N \geq m | n_1 + n_2 + \dots + n_N = n]$ for any $m, n \geq 1$.

The recurrence as given by (2.26) remains unchanged for $C_i(j | n_N \geq m)$, $1 \leq i < N$, $1 \leq j \leq n$.

When $i = N$,

$$\begin{aligned} G_N(z) &= [T_N^{m-1} z^m + T_N^m z^{m+1} + \dots] A_N G_{N-1}(z) \\ &= [I - zT_N]^{-1} z^m T_N^{m-1} A_N G_{N-1}(z) \end{aligned} \quad (2.31)$$

so that

$$C_N(j | n_N \geq m) = T_N C_N(j-1 | n_N \geq m) + T_N^{m-1} A_N C_{N-1}(j-m | n_N \geq m) \quad (2.32)$$

for $m \leq j \leq n$. Finally,

$$P[n_N \geq m | n_1 + \dots + n_N = m] = \frac{\alpha_N C_N(n | n_N \geq m) \beta_1}{\alpha_N C_N(n) \beta_1} \quad (2.33)$$

where $C_N(n | n_N \geq m)$ is defined by (2.32). In each of the cases, the boundary conditions are given by Theorem 1. Note that the same argument can be used to compute $P[n_k \geq m | n_1 + \dots + n_N = n]$ for any k , $1 \leq k \leq N$, and $m \geq 1$. The easiest way to do this is by exchanging indices for (α_k, T_k) and (α_N, T_N) .

As another example, let S be a proper subset of the set $\{1, 2, \dots, N\}$. For given positive integers $m_j, j \in S$, the quantity $P[n_j \geq m_j, j \in S | n_1 + \dots + n_N = n]$ is computed by further extending the above arguments. For $i \notin S$ we use the recurrence in (2.26), and for $i \in S$ we use

$$C_i(j | n_j \geq m_j, j \in S) = T_i C_i(j-1 | n_j \geq m_j, j \in S) + T_i^{m_i-1} A_i C_{i-1}(j - m_i | n_j \geq m_j, j \in S) \quad (2.34)$$

for $m_i \leq j \leq n$, and $1 < i \leq N$. If $1 \notin S$, then $C_1(j | n_j \geq m_j, j \in S) = T_1^{j-1}$, otherwise $C_1(j | n_j \geq m_j, j \in S) = T_1^{m_1+j-2}$, for $j \geq 1$. Similarly $C_1(0 | n_j \geq m_j, j \in S) = d_1 A_1$ if $1 \notin S$, and $C_1(0 | n_j \geq m_j, j \in S) = I$ otherwise. In like fashion, for $i > 1$, $C_i(0 | n_j \geq m_j, j \in S) = d_i A_i C_{i-1}(0 | n_j \geq m_j, j \in S)$ if $i \notin S$, and $C_i(0 | n_j \geq m_j, j \in S) = T_i^{m_i-1} A_i C_{i-1}(0 | n_j \geq m_j, j \in S)$ otherwise.

3. Nonhomogeneous Transitions

The notion of a phase type random variable introduced in the previous section (from [3]), can be generalised to include nonhomogeneous Markov chains. In this section, we consider such a case and then demonstrate how another useful and efficient computational scheme can be developed. For convenience, we restrict our attention to discrete time.

Let $\{X_r; r \geq 0\}$ be a nonhomogeneous Markov chain with one-step transition probability matrix at time r given by $P(r) \triangleq [p_{ij}(r, r+1)]$. Denoting $H(k, r) \triangleq [p_{ij}(k, r)]$ as a multistep transition probability matrix, the Chapman-Kolmogorov equation can be written as

$$H(k, r) = H(k, q) H(q, r) \quad (3.1)$$

for $k \leq q \leq r$, with $H(r, r) = I$. A direct application of the last equation yields

$$H(0, r) = P(0) P(1) \dots P(r-1) \quad (3.2)$$

for $r \geq 1$.

Assume now that the chain $\{X_r; r \geq 0\}$ is defined over the states $\{1, 2, \dots, m+1\}$, where states 1 through m are transient and state $(m+1)$ is absorbing. Also, if absorption into state $m+1$ can occur at any step r , then $P(r)$ must take the form

$$P(r) = \begin{vmatrix} T(r) & \beta(r) \\ 0 & 1 \end{vmatrix} \quad (3.3)$$

where $T(r)$ is $m \times m$ and $\beta(r)$ is $m \times 1$. The density of the time to absorption for this nonhomogeneous Markov chain, is given by

$$\begin{aligned} p_0 &= P[X_0 = m+1] = \alpha_{m+1} \\ p_k &= P[X_k = m+1] = \alpha T(0) T(1) \dots T(k-1) \beta, \quad k \geq 1 \end{aligned} \quad (3.4)$$

where (α, α_{m+1}) is the initial vector, and $T(0) = I_{m \times m}$. Note that in the homogeneous case, $P(r) = P$ for all r , and the density in (3.4) reduces to that given in (2.2).

Example 3: N Independent Nonhomogeneous Chains

Let $\{X_{n,k}; n \geq 0\}$ be a nonhomogeneous Markov chain with initial vector $(\alpha_k, \alpha_{k,m+1})$ and transition matrix sequence $\{P_k(n); n \geq 0\}$, for each k , $1 \leq k \leq N$. Corresponding to each matrix $P_k(n)$, from (3.3) we have a matrix $T_k(n)$ and vector $\beta_k(n)$, $n \geq 0$.

Given N independent nonhomogeneous Markov chains $\{X_{n,k}; n \geq 0, 1 \leq k \leq N\}$, let Y_k be the time to absorption for the chain $\{X_{n,k}\}$ and define $Y = Y_1 + Y_2 + \dots + Y_N$. The probability that all N chains require a total of n steps for their respective absorptions into state $(m+1)$ is given by $P(Y = n) = C(n)$, where

$$C(n) = \sum_{D(n)} \prod_{k=1}^N \left\{ I_{\{n_k=0\}} \alpha_{k,m+1} + I_{\{n_k>0\}} \alpha_k \left[\prod_{j=0}^{n_k} T_k(j) \right] \beta_k \right\} \quad (3.5)$$

which, like (2.4), requires an exponential algorithm if computed using brute force. In the following result, we demonstrate how (3.5) may be computed efficiently. The idea, like the one used in Theorem 1, exploits the generating function of $\{C(n)\}$ and the representations of convolutions.

Theorem 2

The quantity

$$C(n) = \sum_{D(n)} \prod_{k=1}^N \left\{ I_{\{n_k=0\}} \alpha_{k,m+1} + I_{\{n_k>0\}} \alpha_k \left[\prod_{j=0}^{n_k} T_k(j) \right] \beta_k \right\}$$

is given by $C_N(n)$, via the recurrence

$$C_N(n) = \sum_{k=0}^n \alpha_N A_N(n-k) \beta_N C_{N-1}(k) \quad (3.6)$$

with $C_0(0) = I$, $C_0(k) = 0$, $A_k(0) = d_k I$, $A_k(1) = T_k(0) = I$, and $d_k = \frac{\alpha_{k,m+1}}{\alpha_k \beta_k}$, for $k \geq 1$.

Proof:

Define the function

$$\begin{aligned} G(z) = & \alpha_N [d_N I + T_N(0)z + T_N(1)z^2 + \dots] \beta_N \alpha_{N-1} [d_{N-1} I + T_{N-1}(0)z + T_{N-1}(1)z^2 + \\ & \dots] \beta_{N-1} \dots \alpha_1 [d_1 I + T_1(0)z + T_1(1)z^2 + \dots] \beta_1 \end{aligned} \quad (3.7)$$

and observe that just as in Theorem 1, $C_N(n)$ is given by the coefficient of z^n in $G(z)$. Defining $C(k)$ to be the coefficient of z^k in $G(z)$,

$$G(z) = \sum_{k=0}^{\infty} C(k) z^k \quad (3.8)$$

is the generating function of $\{C(k); k \geq 0\}$. Denoting the i -term version of (3.7) by $G_i(z)$, clearly

$$G_i(z) = \alpha_i [d_i I + T_i(0)z + T_i(1)z^2 + \dots] \beta_i G_{i-1}(z) \quad (3.9)$$

for $i \geq 1$, with $G_0(z) = I$. Applying (3.8), we have

$$G_i(z) = \alpha_i \left[\sum_{k=0}^{\infty} A_i(k) z^k \right] \beta_i \left[\sum_{r=0}^{\infty} C_{i-1}(r) z^r \right] \quad (3.10)$$

where $A_i(0) = d_i I$, $A_i(1) = I$, and $A_i(k) = T_i(k)$ for $i \geq 1$, and $k > 1$. Rewriting (3.10) slightly differently,

$$G_i(z) = \left[\sum_{k=0}^{\infty} \alpha_i A_i(k) z^k \right] \left[\sum_{r=0}^{\infty} \beta_i C_{i-1}(r) z^r \right] \quad (3.11)$$

and recognizing the last product to essentially be a convolution, we get

$$G_i(z) = \sum_{r=0}^{\infty} \left\{ \sum_{k=0}^{\infty} \alpha_i A_i(r-k) \beta_i C_{i-1}(k) \right\} z^r, \quad (3.12)$$

and thus

$$C_N(n) = \sum_{k=0}^n \alpha_N A_N(n-k) \beta_N C_{N-1}(k). \quad (3.13)$$

The boundary conditions give us $C_0(0) = I$, $C_0(k) = 0$, $A_k(0) = \frac{\alpha_{k,m+1}}{\alpha_k \beta_k} I$, and $A_k(1) = T_k(0) = I$ for $k \geq 1$.

■

4. The Multivariate Case

In this section, we use arguments similar to the preceding sections to obtain some multivariate probabilities. As before, we motivate the discussion with an example. We are interested in a system of N queues, where each queue has arrivals coming from M different Poisson processes. The fact that the total arrival process will also be Poisson (with rate that is a sum of individual process rates) is not useful for the probability that we wish to compute. Consider the following example for $M = 2$.

Figure 6. Mixed Arrivals

Example 4: Queues with Several Arrival Classes

Consider the following extension of Example 2. At each of the N queues, customers arrive according to a Poisson process of rate λ_1 or a Poisson process of rate λ_2 (see Figure 6). Assume that the service time at queue k , $1 \leq k \leq N$, is a continuous PH random variable with distribution $F_k(\cdot)$ and representation (γ_k, S_k) . Let X_k (Y_k) be the number of customers that arrive in a service time from process 1 (process 2) at queue k , $1 \leq k \leq N$. Then

$$\begin{aligned} P(X_k = m_k, Y_k = n_k) &= \int_0^\infty \frac{e^{-(\lambda_1 + \lambda_2)t} (\lambda_1 t)^{m_k} (\lambda_2 t)^{n_k}}{m_k! n_k!} dF_k \\ &= \frac{(m_k + n_k)!}{m_k! n_k!} p^{m_k} q^{n_k} \int_0^\infty \frac{e^{-(\lambda_1 + \lambda_2)t} [(\lambda_1 + \lambda_2)t]^{m_k + n_k}}{(m_k + n_k)!} dF_k \end{aligned} \quad (4.1)$$

where $p = \frac{\lambda_1}{\lambda_1 + \lambda_2}$ and $q = 1 - p$. Applying Lemma 1 to (4.1), we obtain

$$P(X_k = m_k, Y_k = n_k) = \frac{(m_k + n_k)!}{m_k! n_k!} \left\{ I_{\{m_k + n_k = 0\}} \alpha_{k,m+1} + I_{\{m_k + n_k > 0\}} \alpha_k X^{m_k} Y^{n_k} \beta_k \right\} \quad (4.2)$$

where, with $\lambda = \lambda_1 + \lambda_2$,

$$\alpha_k = \lambda \gamma_k (\lambda I - S_k)^{-1}, \quad \alpha_{k,m+1} = \gamma_{k,m+1} + \gamma_k (\lambda I - S_k)^{-1} S_k^0,$$

$$X_k = \lambda_1 (\lambda I - S_k)^{-1}, \quad Y_k = \lambda_2 (\lambda I - S_k)^{-1} \quad (4.3)$$

$$\beta_k = \frac{1}{\lambda} \cdot S_k^0$$

for each k , $1 \leq k \leq N$. Now defining $X = X_1 + \dots + X_N$, $Y = Y_1 + \dots + Y_N$, and integers $a \geq 0, b \geq 0$, we have $P(X = a, Y = b) = C(a, b)$, where

$$C(a, b) = \sum_{F(m,n)} \prod_{k=1}^N P(X_k = m_k, Y_k = n_k) \quad (4.4)$$

with

$$F(a,b) = \{ (m_k, n_k) \mid m_k \geq 0, n_k \geq 0, 1 \leq k \leq N, \sum_{i=1}^N m_i = a, \sum_{i=1}^N n_i = b \}$$

While one way to obtain $C(a, b)$ is to use the results of section 2, which will work due to independence arguments, an alternate method utilizing multinomial-type expressions is as follows. Define the generating function $G(z_1, z_2) = G_N(z_1, z_2)$ as

$$\begin{aligned} G(z_1, z_2) &= \prod_{k=N}^1 [d_k I + (X_k z_1 + Y_k z_2) + (X_k z_1 + Y_k z_2)^2 + \dots] A_k \\ &= \prod_{k=N}^1 \left[d_k I + \sum_{j=1}^{\infty} (X_k z_1 + Y_k z_2)^j \right] A_k \end{aligned} \quad (4.5)$$

where $d_k = \frac{\alpha_{k,m+1}}{\alpha_k \beta_k}$ for $1 \leq k \leq N$, $A_k = \beta_k \alpha_{k-1}$ for $1 < k \leq N$, and $A_1 = I$. Observing that the quantity we wish to compute, $C(a, b) = P[X = a, Y = b]$, is given by the coefficient of $z_1^a z_2^b$ in $\alpha_N G(z_1, z_2) \beta_1$, and defining $C_N(i, j)$ to be coefficient of $z_1^i z_2^j$ in $G(z_1, z_2)$, we obtain

$$\begin{aligned} C_i(j, k) &= (d_i - 1)[A_i C_{i-1}(j, k) - X_i A_i C_{i-1}(j-1, k) - Y_i A_i C_{i-1}(j, k-1)] \\ &\quad + A_i C_{i-1}(j, k) + X_i C_i(j-1, k) + Y_i C_i(j, k-1) \end{aligned} \quad (4.10)$$

for $1 \leq i \leq N$, $1 \leq j \leq a$, $1 \leq k \leq b$. With the aid of boundary conditions, we obtain the following theorem.

Theorem 3

The quantity

$$C(a, b) = \sum_{F(a, b)} \prod_{k=1}^N \{ I_{\{m_k + n_k = 0\}} \alpha_{k, m+1} + I_{\{m_k + n_k > 0\}} \alpha_k X^{n_k} Y^{n_k} \beta_k \}$$

is given by $\alpha_N C_N(a, b) \beta_1$ via the recurrence

$$C_i(j, k) = X_i [C_i(j-1, k) + (1 - d_i) A_i C_{i-1}(j-1, k)] + d_i A_i C_{i-1}(j, k) \\ + Y_i [C_i(j, k-1) + (1 - d_i) A_i C_{i-1}(j, k-1)] \quad (4.11)$$

for $1 \leq i \leq N$, $1 \leq j \leq a$, $1 \leq k \leq b$. The required quantities are given by $d_n = \frac{\alpha_{n,m+1}}{\alpha_n \beta_n}$ for $1 \leq n \leq N$, $A_1 = I$, and $A_n = \beta_n \alpha_{n-1}$ for $1 < n \leq N$. Finally, boundary conditions give $C_0(0,0) = I$, $C_i(0,0) = (\prod_{k=1}^i d_k) I$, $C_i(0,k) = Y_i^k A_i$, $C_i(k,0) = X_i^k A_i$ for $1 \leq i \leq N$ and $k > 0$.

■

The last result is easily extended to the case where each queue has M Poisson inputs. Defining $X_{i,j}$ to be the number of process i customers arriving at station j in a service time, $1 \leq i \leq M$, $1 \leq j \leq N$, let $Y_i = X_{i,1} + \dots + X_{i,N}$, for each arrival process i . It follows that $P(Y_i = n_i, 1 \leq i \leq M) = C(n_1, \dots, n_M)$, with

$$C(n_1, \dots, n_M) = \sum_{F(n_1, \dots, n_M)} \prod_{k=1}^N \left\{ I_{\Theta_0} \alpha_{k,m+1} + I_{\Theta_1} \alpha_k \left[\prod_{i=1}^M X_i^{n_{k,i}} \right] \beta_k \right\} \quad (4.12)$$

where $F(n_1, \dots, n_M) = \left\{ (n_{k,1}, \dots, n_{k,M}) \mid \sum_{k=1}^N n_{k,i} = n_i, 1 \leq i \leq M \right\}$,

$\Theta_0 = \{n_1 + \dots + n_M = 0\}$, $\Theta_1 = \{n_1 + \dots + n_M > 0\}$. A straightforward application of the arguments given for Theorem 3 yields the recurrence

$$C_i(j_1, \dots, j_M) = d_i A_i C_{i-1}(j_1, \dots, j_M) + f(M) \quad (4.13)$$

where

$$f(M) = \sum_{k=1}^M X_k \left[C(j_1, \dots, j_k - 1, \dots, j_M) + (1 - d_i) A_i C_{i-1}(j_1, \dots, j_k - 1, \dots, j_M) \right] \quad (4.14)$$

for $1 \leq i \leq N$, $1 \leq j_k \leq n_k$, and $1 \leq k \leq M$, so that

$C(n_1, \dots, n_M) = \alpha_N C_N(n_1, \dots, n_N) \beta_1$. As before, the boundary conditions give us

$C_0(0, \dots, 0) = I$, $C_i(0, \dots, 0) = (\prod_{k=1}^i d_k) I$, $C_i(j_1, \dots, j_M) = X_i^{j_i} A_i$ for $j_k = 0$, and $j_n \neq 0, 1 \leq n \leq M, n \neq k$.

REFERENCES

- [1] F. Moore, "Computational Model of a Closed Queueing Network with Exponential Servers," *IBM J. of Res. and Develop.*, pp. 567-572, November 1962.
- [2] M. F. Neuts, "Computer Power or The Liberation of Applied Probability," Purdue Univ. Dept. Of Statistics, Mimeo Series No. 312, January 1973.
- [3] ———, *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*, The Johns Hopkins University Press, 1981.
- [4] V. Rego and W. Szpankowski, "An M/SM/1 View of a Token Ring Queue," Purdue CSD-TR 685, June 1987.
- [5] W. Szpankowski and V. Rego, "Ultimate Stability Conditions for Some Multidimensional Distributed Systems," Purdue CSD-TR 715, October 1987.
- [6] A. Williams and R. Bhandiwad, "A Generating Function Approach to Queueing Network Analysis of Multiprogrammed Computers," *Networks*, Vol. 6, pp. 1-22, 1976.

APPENDIX

1. Phase Type Distributions

Consider a Markov process on the states $\{1, 2, \dots, m+1\}$ with infinitesimal generator

$$Q = \begin{vmatrix} T & T^0 \\ 0 & 0 \end{vmatrix} \quad (\text{A.1.1})$$

where the $m \times m$ matrix T satisfies $T_{ii} < 0$, for $1 \leq i \leq m$ and $T_{ij} \geq 0$ for $i \neq j$. Also $Te + T^0 = 0$, where $e = (1, 1, \dots, 1)$ is an m -vector. The initial probability vector of Q is given by (α, α_{m+1}) , with $\alpha e + \alpha_{m+1} = 1$. We assume that the states $1, 2, \dots, m$ are transient and state $(m+1)$ is an absorbing state. The probability distribution $F(\cdot)$ of the time until absorption in the state $(m+1)$ given an initial probability vector (α, α_{m+1}) is given by

$$F(x) = 1 - \alpha \exp(Tx)e, \quad x \geq 0 \quad (\text{A.1.2})$$

The distribution $F(\cdot)$ is said to be a phase-type distribution with the representation (α, T) .

The probability density $\{p_k\}$, given by

$$\begin{aligned} p_0 &= \alpha_{m+1} \\ p_k &= \alpha T^{k-1} T^0, \quad k \geq 1 \end{aligned} \quad (\text{A.1.3})$$

is said to be a discrete PH density when T is a substochastic matrix, and T^0 has nonnegative entries, such that $Te + T^0 = e$. The density in (A.1.3) is the time until absorption in state $(m+1)$ in a Markov chain

$$P = \begin{vmatrix} T & T^0 \\ 0 & 1 \end{vmatrix} \quad (\text{A.1.4})$$

and with initial vector (α, α_{m+1}) , this density is said to have the representation (α, T) . In the body of the paper we frequently use the notation β in place of T^0 .

2. Transition Probabilities for the Token Ring Example

Let Θ_{N-1} denote the set of all $(N - 1)$ bit binary vectors. Let $z_0, z_1, \dots, z_M$ denote the elements of Θ_{N-1} given in lexicographic order with $M = 2^{N-1} - 1$. That is $z_0 = \langle 0, 0, \dots, 0 \rangle$, $z_1 = \langle 0, 0, \dots, 1 \rangle, \dots, z_M = \langle 1, 1, \dots, 1 \rangle$. Also, let $z_i(k)$ denote the k^{th} bit (counting from left to right) of the vector z_i , $1 \leq k \leq N - 1, z_i \in \Theta_{N-1}$. Finally, define

$$H_n(z_i | z_j) \triangleq A_n^*(\langle 0, 0, \dots, 0 \rangle | z_j) \quad (\text{A.2.1})$$

where $A_n^*(\langle 0, 0, \dots, 0 \rangle | z_j)$ is given by (2.10)–(2.14), with the added restriction that λ_k is replaced by $[1 - z_i(k)]\lambda_k$, $1 \leq k \leq N - 1$, for this computation.

For any given value of n , $n \geq 0$, the substochastic matrix $A_n = A_n(\cdot | \cdot)$ can be computed using the following recursive scheme:

```

begin
  for  $j := 0$  to  $M$  do
    begin
      compute  $A_n^*(z_0 | z_j)$ ;
      for  $i := 1$  to  $M$  do
        begin
          compute  $H_n(z_i | z_j)$ ;
           $A_n(z_i | z_j) := H_n(z_i | z_j) - \sum_{m \in R(i)} A_n(z_m | z_j)$    (A.2.2)
        end;
      end;
    end;
  end.

```

The set $R(i)$ is defined with $R(i) = \{k | 0 \leq k \leq i - 1, z_k \wedge z_i \neq 0\}$, for $1 \leq i \leq N - 1$.

Since the $B_n = B_n(\cdot | \cdot)$ submatrices are computed using a slight variation of (2.10)–(2.14), the above algorithm can also be used to compute the sequence $\{B_n; n \geq 0\}$ (see section 2.2).