A Novel Computational Method for Solving Finite QBD Processes*

Nail Akar
Sprint Corporation
9300 Metcalf Avenue
Overland Park, KS 66212
akar@sprintcorp.com

Nihat C. Oğuz and Khosrow Sohraby†
Computer Science Telecommunications
University of Missouri-Kansas City
5100 Rockhill Road, Kansas City, MO 64110
{ncoguz,sohraby}@cstp.umkc.edu

Abstract: We present a novel numerical method that exploits invariant subspace computations for finding the stationary probability distribution of a finite QBD process. Assuming that the QBD state space is defined in two dimensions with m phases and $K+1$ levels, the solution vector π_k for level k, $0 \leq k \leq K$, is known to be expressible in the mixed matrix-geometric form $\pi_k = v_1 R_1^k + v_2 R_2^{K-k}$, where R_1 and R_2 are certain solutions to two quadratic matrix equations, and v_1 and v_2 are vectors to be determined using the boundary conditions. We show that the matrix-geometric factors R_1 and R_2 can be *simultaneously* obtained irrespective of K via finding arbitrary bases for the left- and right-invariant subspaces of a certain real matrix of size $2m$. To find these bases, we employ either Schur decomposition or a matrix-sign function iteration with quadratic convergence rate. The vectors v_1 and v_2 are obtained by solving a linear matrix equation, which is constructed with a time complexity of $O(m^3 \log_2 K)$. Therefore, the effect of the number of levels on the overall complexity is minimal. Besides its numerical efficiency, the proposed method is numerically stable, and in the limiting case of $K \to \infty$, it is shown to yield the well-known matrix-geometric solution $\pi_k = \pi_0 R_1^k$ for infinite QBD processes. We also extend the method to the cases of level-dependent and non-canonical transitions, and provide a numerical example to demonstrate its computational features in comparison with other well-known methods.

Keywords: Finite QBD process, matrix-geometric approach, invariant subspaces, matrix-sign function, real Schur decomposition, Sylvester matrix equation, numerical linear algebra.

---

*This paper is a revised and extended version of [3].

†This work was supported by DARPA/ITO under grant A0 F316 and by NSF under grant NCR-950814.

1 INTRODUCTION

In this paper, we examine a finite *Quasi-Birth-and-Death* (QBD) process which is a Markov chain with state space $\{(i, j) : 0 \leq i \leq K, 1 \leq j \leq m\}$, and which, in discrete time, has an irreducible transition probability matrix of the canonical block tridiagonal form

$$P = \begin{bmatrix} B_0 & A_0 & & & & \\ B_1 & A_1 & A_0 & & & \\ & A_2 & A_1 & \ddots & & \\ & & A_2 & \ddots & A_0 & \\ & & & \ddots & A_1 & C_0 \\ & & & & A_2 & C_1 \end{bmatrix} \qquad (1)$$

with B_0, B_1, A_0, A_1, A_2, C_0, and C_1 all being $m \times m$ nonnegative matrices. Note that P is of size $m(K + 1) \times m(K + 1)$. Throughout the paper, we use e_k and I to denote a column vector of ones of length mk and an identity matrix of appropriate size, respectively.

Our aim is to find the stationary probability vector π of the finite QBD process, which is the unique solution to the equations

$$\pi = \pi P \text{ and } \pi e_{(K+1)} = 1. \qquad (2)$$

For convenience, we partition π as

$$\pi \triangleq \begin{bmatrix} \pi_0 & \pi_1 & \cdots & \pi_K \end{bmatrix}, \qquad (3)$$

where π_k, of size $1 \times m$, is the solution vector for level k, $0 \leq k \leq K$, as in [43].

The infinite QBD process with the transition probability matrix of the form

$$P = \begin{bmatrix} B_0 & A_0 & & & \\ B_1 & A_1 & A_0 & & \\ & A_2 & A_1 & \ddots & \\ & & A_2 & \ddots & \\ & & & \ddots & \end{bmatrix}, \qquad (4)$$

has been encountered in numerous applications involving infinite queues [42], [35], [36], [41]. In this case, the stationary solution has a matrix-geometric form [36], i.e.,

$$\pi_k = \pi_0 \, R_1^k, \; k \geq 0, \tag{5}$$

where the matrix-geometric rate matrix R_1 is the unique minimal nonnegative solution of the quadratic matrix equation

$$R = A_0 + RA_1 + R^2 A_2. \tag{6}$$

Once R_1 is known, the boundary vector π_0 in (5) can be computed by solving [36]

$$\pi_0 = \pi_0 B[R_1], \; \pi_0 (I - R_1)^{-1} e_1 = 1, \tag{7}$$

where

$$B[R_1] \triangleq B_0 + R_1 B_1. \tag{8}$$

In the case of finite QBD processes with transition probability matrices of the form (1), the stationary solution can be expressed in a variety of matrix-geometric forms [12], [11], [23], [22], [5]. In [24], Hajek has shown in a general setting that the solution could be expressed as the sum of two matrix-geometric terms plus a linear term. The formulation of [24], however, does not lend itself to an efficient numerical algorithm to compute such a solution. Taking a completely different approach, the folding algorithm of Ye and Li [43] solves a finite QBD process directly by employing a particular Markov chain reduction technique along with an appropriate state permutation scheme that significantly reduces the cost of each reduction step.

This paper follows the matrix-geometric approach, and presents a simple and constructive proof for the mixed matrix-geometric form

$$\pi_k = v_1 R_1^k + v_2 R_2^{K-k}, \; 0 \leq k \leq K, \tag{9}$$

of the solution vector for level k. This form of solution has been reported for the analysis of a multi-server delay-loss system with a general Markovian arrival process [34]. In (9), R_1 is as defined before, and R_2 is the unique minimal nonnegative solution of the quadratic matrix equation

$$R = A_2 + RA_1 + R^2 A_0, \tag{10}$$

which is the dual of (6), and v_1 and v_2 are row vectors of length m. Once R_1 and R_2 are known, v_1 and v_2 follow easily as the solution of a linear matrix equation of size $2m$, which is constructed with a time complexity of $O(m^3 \log_2 K)$. The major contribution of this paper lies in that we propose a novel and numerically efficient method to *simultaneously* compute the two matrix-geometric factors R_1 and R_2. Using the methodology developed in [4] for general M/G/1 and G/M/1 paradigms, we show that R_1 and R_2 are related to the *left*- and *right*-invariant subspaces of *one* square real matrix of size $2m$, respectively. More precisely, they are easily obtained once an arbitrary basis for each of the two subspaces is found. As the engine for *simultaneous* computation of these subspaces, which turns out to be the core of the algorithm in terms of execution time, we consider the use of either Schur decomposition [21] or a matrix-sign iteration with quadratic convergence rate [15].

In [1], we have presented a generalized state-space approach for the solution of M/G/1-type Markov chains with infinite or finite levels. This solution too has a matrix-geometric form as $\pi_{k+1} = gF^k H$ for the infinite-level case whereas a two-term version of this form, as in (9), holds for the case of finite levels. The approach of [1] can also be used to analyze QBD chains as effectively as the invariant subspace approach of this paper although the forms of solutions of the two approaches are not identical.

The general features of the proposed method can be summarized as follows.

(i) The method is simple to implement, efficient, and does not require much storage: it is feasible to implement and use it on small computers even for large-scale QBD processes.

(ii) The number of levels, $K + 1$, does not have impact on the accuracy

or numerical stability of the method; in the limiting case of $K \to \infty$, the solution is shown to yield the matrix-geometric form (5) for infinite QBD processes.

(iii) The simple form of the stationary solution makes it easy to obtain certain performance measures of interest like the blocking probabilities and moments of the level distribution.

(iv) Central to the method is the two matrix-geometric factors, the computation of which does not depend on the number of levels. Once these two matrices are obtained, relatively little additional computational effort is needed for finding the stationary probabilities when the number of levels is a variable parameter. This is a significant advantage in design and dimensioning problems.

(v) The extensions to the cases of a non-canonical transition probability matrix and piecewise homogeneous level-dependent transitions (encountered in queueing systems with multilevel overload control) are easy.

Here we note that the matrix-geometric factors R_1 and R_2 can also be obtained by two successive runs of the logarithmic reduction algorithm of Latouche and Ramaswami [29], or of the cyclic reduction algorithm of Bini and Meini [10]. Both of these algorithms have similar performances in terms of execution times and similar memory requirements as enjoyed by our invariant subspace approach implemented via quadratically convergent matrix-sign iterations [4]. The invariant subspace approach through the use of Schur decomposition, on the other hand, proves to be faster than these three (see Section 5).

The paper is organized as follows. In Section 2, we give the mathematical formulation of the proposed method. Section 3 addresses the invariant subspace approach to finding the matrix-geometric factors and the implementation via Schur decomposition and matrix-sign function. In Section 4, we provide a step-by-step outline of the overall method, and discuss its extensions including the one to the case of level-dependent transitions. In Section 5, we study the performance of both Schur decomposition and matrix-sign function implementations of the proposed method extensively, and compare it, in the context of a continuous-time voice-data multiplexer example, with the well-known folding algorithm of Ye and Li [43] for finite QBD processes, as well as with the logarithmic and cyclic reduction algorithms cited above.

2 MATHEMATICAL FORMULATION

In this section, we will show that the stationary solution of the finite QBD process described by (1) can be expressed in the mixed matrix-geometric form (9). We will also address some numerical stability considerations. Defining

$$A(z) \stackrel{\triangle}{=} A_0 + A_1 z + A_2 z^2, \tag{11}$$

we assume that the matrix $A(1)$ is irreducible, stochastic, and has an invariant vector x such that

$$xA(1) = x \text{ and } xe_1 = 1. \tag{12}$$

As in [18], we introduce the following assumption:

$$\text{the matrix } [zI - A(z)] \text{ is nonsingular for } |z| = 1,\ z \neq 1. \tag{13}$$

This hypothesis is shown in [18] to be guaranteed by re-blocking the system introducing no loss of generality. In the mathematical presentation of this section, we assume without loss of generality that the traffic parameter

$$\gamma \stackrel{\triangle}{=} x(A_0 - A_2)e_1 < 0, \tag{14}$$

which makes the variant of (1) with infinite number of levels positive recurrent. This assumption is indeed not necessary for the finite QBD process, and adjustments in the solution procedure for the case of $\gamma > 0$ will be provided later in Section 4.1. The case of $\gamma = 0$ will be omitted from the discussion; see [24] for the solution in terms of pseudo-inverses for this pathological case.

Recalling that R_1 and R_2 are the unique minimal nonnegative solutions to the quadratic matrix equations (6) and (10), respectively, we first show that the matrix

$$B[R_1, R_2] \triangleq \begin{bmatrix} B_0 + R_1 B_1 & R_1^{K-1}(C_0 + R_1 C_1 - R_1) \\ R_2^{K-1}(R_2 B_0 + B_1 - R_2) & C_1 + R_2 C_0 \end{bmatrix} \tag{15}$$

has an eigenvalue at unity.

Theorem 1 *The matrix $B[R_1, R_2]$ defined in (15) satisfies*

$$B[R_1, R_2] e_2 = e_2, \tag{16}$$

i.e., e_2 is a right-invariant eigenvector of $B[R_1, R_2]$.

Proof: The proof is based on direct substitution using

$$P e_{(K+1)} = e_{(K+1)}, \tag{17}$$

and the quadratic equations (6) and (10) that the matrices R_1 and R_2 satisfy, respectively. First note that

$$\begin{aligned} (I - R_1)(B_0 + R_1 B_1) e_1 &= (I - R_1)(I - A_0 + R_1 A_2) e_1, \text{ by (17)} \\ &= e_1 + R_1(A_0 + A_2 - I) e_1 - (A_0 + R_1^2 A_2) e_1, \\ &= e_1 - R_1 A_1 e_1 + (R_1 A_1 - R_1) e_1, \text{ by (17) and (6)} \\ &= (I - R_1) e_1. \end{aligned} \tag{18}$$

Since R_1 is power-summable, $(I - R_1)$ is nonsingular; and, consequently, (18) implies that

$$(B_0 + R_1 B_1) e_1 = e_1. \tag{19}$$

We similarly have

$$\begin{aligned} (I - R_1)(C_0 + R_1 C_1) e_1 &= (I - R_1)(A_0 + R_1 - R_1 A_2) e_1, \text{ by (17)} \\ &= (A_0 + R_1^2 A_2) e_1 + R_1(I - A_2 - A_0) e_1 - R_1^2 e_1, \\ &= (R_1 - R_1 A_1) e_1 + R_1 A_1 e_1 - R_1^2 e_1, \text{ by (17) and (6)} \\ &= (I - R_1) R_1 e_1, \end{aligned} \tag{20}$$

which implies

$$(C_0 + R_1 C_1 - R_1) e_1 = 0 \tag{21}$$

again by non-singularity of $(I - R_1)$. Now note that

$$\begin{aligned} (I - R_2)(R_2 B_0 + B_1 - R_2) e_1 &= (I - R_2)(A_2 - R_2 A_0) e_1, \text{ by (17)} \\ &= 0, \text{ by (17) and (10)} \end{aligned} \tag{22}$$

which implies $R_2^i(R_2B_0 + B_1 - R_2)e_1 = (R_2B_0 + B_1 - R_2)e_1 \ \forall i \geq 0$, and in particular,

$$R_2^{K-1}(R_2B_0 + B_1 - R_2)e_1 = (R_2B_0 + B_1 - R_2)e_1. \tag{23}$$

Finally, one can show by using (17) that

$$(R_2B_0 + B_1 - R_2)e_1 + (C_1 + R_2C_0)e_1 = e_1, \tag{24}$$

and the proof follows from equalities (19), (21), (23), and (24). □

Now defining

$$S_i \triangleq \sum_{k=0}^{K} R_i^k, \quad i = 1, 2, \tag{25}$$

we have the following result.

Theorem 2 *Let $v = [\, v_1 \ \ v_2 \,]$, where v_1 and v_2 are row vectors of length m, be the left-invariant eigenvector of $B[R_1, R_2]$, i.e.,*

$$v = v\, B[R_1, R_2], \tag{26}$$

normalized so that

$$(v_1S_1 + v_2S_2)\, e_1 = 1. \tag{27}$$

The solution vector π_k for level k of the QBD process (1) is then given by

$$\pi_k = v_1R_1^k + v_2R_2^{K-k}, \quad 0 \leq k \leq K. \tag{28}$$

Proof: The proof is by direct substitution. First, observe that

$$\begin{aligned}
\pi_0B_0 + \pi_1B_1 &= (v_1 + v_2R_2^K)B_0 + (v_1R_1 + v_2R_2^{K-1})B_1, \\
&= v_1(B_0 + R_1B_1) + v_2R_2^{K-1}(R_2B_0 + B_1), \\
&= v_1 + v_2R_2^K, \ \text{ by (26)} \\
&= \pi_0.
\end{aligned} \tag{29}$$

In the same way, one can show that

$$\pi_{K-1}C_0 + \pi_KC_1 = v_1R_1^K + v_2 = \pi_K. \tag{30}$$

For $1 \leq k \leq K-1$, we write

$$\begin{aligned}
\pi_{k-1}A_0 + \pi_kA_1 + \pi_{k+1}A_2 &= v_1R_1^{k-1}(A_0 + R_1A_1 + R_1^2A_2), \\
&\quad + v_2R_2^{K-k-1}(A_2 + R_2A_1 + R_2^2A_0), \\
&= v_1R_1^k + v_2R_2^{K-k}, \ \text{ by (6) and (10)} \\
&= \pi_k.
\end{aligned} \tag{31}$$

Finally,

$$\begin{aligned}
(\sum_{k=0}^{K} \pi_k) e_1 &= v_1 (\sum_{k=0}^{K} R_1^k) e_1 + v_2 (\sum_{k=0}^{K} R_2^{K-k}) e_1, \\
&= v_1 S_1 e_1 + v_2 S_2 e_1, \\
&= 1, \text{ by } (27). \qquad (32)
\end{aligned}$$

Using equalities (29), (30), (31), and (32), it is easy to see that

$$\pi = \pi P \text{ and } \pi e_{(K+1)} = 1$$

hold, which concludes the proof. Also note that one cannot find two linearly independent vectors satisfying (26) and (27) since otherwise it would contradict the uniqueness of the stationary solution to the QBD process (1). □

Finally, our aim is to simplify the expressions for S_1 and S_2 defined as in (25) so that they can be computed efficiently. We first have the following definitions based on [18]. A matrix R is *power-bounded* if the set of powers $R^k, k \geq 0$, is bounded, i.e., $\sup_k ||R^k|| < +\infty$. Similarly, a matrix R is *power-summable* if the set of powers $R^k, k \geq 0$, is summable, i.e., $\sum_k ||R^k|| < +\infty$. The eigenvalues of a power-bounded (power-summable) matrix therefore lie in the closed unit disk (open unit disk). Under the assumptions (13) and (14), it can be shown by following the lines of [36] and [18] that the unique minimal nonnegative solution R_1 of equation (6) is the unique power-summable solution of the same equation, and the eigenvalues of R_1 are exactly the m roots of

$$\Delta_1(z) \triangleq \det[zI - A(z)] \qquad (33)$$

in the open unit disk. Similarly, the unique minimal nonnegative solution R_2 of equation (10) is the unique power-bounded solution of the same equation, and the eigenvalues of R_2 are exactly the m roots of the dual determinant

$$\Delta_2(z) \triangleq \det[zI - (A_2 + A_1 z + A_0 z^2)]$$

in the closed unit disk, one being at $z = 1$ and all others lying in the open unit disk. For an extensive study on the spectra of solutions to nonlinear matrix equations of the form (6) and (10) and their generalizations, we refer the reader to [18].

Now since R_1 is power-summable, we have

$$S_1 = (I - R_1^{K+1})(I - R_1)^{-1}. \tag{34}$$

However, R_2 has an eigenvalue at unity, and a simplification of the form (34) for S_2 is not immediate. Let

$$y \triangleq (A_2 - R_2 A_0)e_1, \tag{35}$$

which yields $R_2 y = y$ by using (22), i.e., y is a right-invariant eigenvector of R_2. By using (10), it is also not difficult to show that x, the invariant vector of $A(1)$ as introduced by (12), is a left-invariant eigenvector of R_2 as well, i.e., $xR_2 = x$. We now define the following matrix

$$M \triangleq \frac{yx}{xy}, \tag{36}$$

which is of rank one. Note that the matrix $(I - \hat{R}_2)$ with

$$\hat{R}_2 \triangleq R_2 - M \tag{37}$$

is invertible since the single eigenvalue of R_2 at unity is moved to the origin by this transformation. One can show by induction that the following holds:

$$R_2^k = \begin{cases} \hat{R}_2^k + M & \text{if } k \geq 1, \\ I & \text{if } k = 0. \end{cases} \tag{38}$$

We then have the following explicit expression for S_2:

$$\begin{aligned} S_2 = \sum_{k=0}^{K} R_2^k &= I + \sum_{k=1}^{K} \hat{R}_2^k + K\,M, \\ &= (I - \hat{R}_2^{K+1})(I - \hat{R}_2)^{-1} + K\,M. \end{aligned} \tag{39}$$

Here we note that it is highly recommended for numerical stability purposes to determine powers of R_2 by using expression (38) in numerical implementations, i.e., take the power of $\hat{R}_2$ instead of R_2 and add M to it.

The method presented here unifies the finite and infinite QBD processes in a single framework. In order to see this, let us look at the coefficient vectors v_1

and v_2 in the solution expression (28) when the number of levels is infinite. Since R_1 is power-summable, $\lim_{K\to\infty} R_1^{K-1} = 0$. Similarly, $\hat{R}_2^{K-1} \to 0$ as $K \to \infty$ which yields $\lim_{K\to\infty} R_2^{K-1} = M$. Therefore, as $K \to \infty$, we have

$$B[R1, R2] \to \begin{bmatrix} B_0 + R_1 B_1 & 0 \\ M(R_2 B_0 + B_1 - R_2) & C_1 + R_2 C_0 \end{bmatrix},$$

and the vector

$$v = \begin{bmatrix} \pi_0 & 0 \end{bmatrix}$$

with π_0 defined as in (7) can be shown to be a left-invariant eigenvector of $B[R_1, R_2]$ satisfying the normalization equation (27). This shows that the solution obtained as described by Theorem 2 yields the well-known matrix-geometric form (5) for the limiting case of infinite number of levels. Besides unification, this convergence ensures numerical stability with respect to the number of levels.

3 INVARIANT SUBSPACE APPROACH TO FINDING THE MATRIX-GEOMETRIC FACTORS

This section first focuses on the theory which relates invariant subspaces to matrix-geometric factors R_1 and R_2 as discussed also in [4]. We then describe the use of matrix-sign function and Schur decomposition in *simultaneous* computation of R_1 and R_2.

3.1 THEORY

We first provide definitions and notation concerning invariant subspaces based on [28] and [20]. Let us begin with some notation on linear spaces, subspaces, and matrices. By $\mathbf{R}^m$, we mean the real linear space of column vectors of m real numbers. Similarly, $\mathbf{R}^{m\times n}$ is the linear space of $m \times n$ matrices with real entries. A *subspace* is a subset of $\mathbf{R}^m$ that is closed under the operations of addition and scalar multiplication. For arbitrary subspaces $\mathcal{S}_1$ and $\mathcal{S}_2$, $\mathcal{S}_1 \subset \mathcal{S}_2$ denotes either inclusion or equality. If $A \in \mathbf{R}^{m\times n}$, we define the image of A by

$$\text{Im } A \triangleq \{x \in \mathbf{R}^m \ : \ x = Ay \text{ for some } y \in \mathbf{R}^n\}.$$

If Im $A =$ Im B for two matrices with the same number of columns, then there exists a nonsingular matrix U such that $B = AU$.

The set of all eigenvalues of a matrix $A \in \mathbf{R}^{m\times m}$ is called the *spectrum* of A, and denoted by $\sigma(A)$. Let C be a region in the complex plane. Then, the matrix A is called C-*stable* if $\sigma(A) \subset C$. Let C^l, C^0, and C^r denote the left-half plane, the imaginary axis, and the right half-plane of the complex plane, respectively. We also use the notation C^{cr} to denote closed right-half plane, i.e., $C^{cr} = C^r \cup C^0$. For $A \in \mathbf{R}^{m\times m}$, a subspace $\mathcal{S}$ of $\mathbf{R}^m$ is said to be A-*invariant* if $A\mathcal{S} \subset \mathcal{S}$. Here, $A\mathcal{S} = \{x \in \mathbf{R}^m \ : \ x = Ay \text{ for some } y \in \mathcal{S}\}$. If a k-dimensional subspace $\mathcal{S}$ is A-invariant, then $AS = SA_1$ holds for a $k \times k$ matrix A_1 and an $m \times k$ matrix S whose columns form a basis for $\mathcal{S}$, i.e., $\mathcal{S} =$ Im S. We call the A-invariant subspace $\mathcal{S}$ the C-invariant subspace of A if $\sigma(A_1) \subset C$, and there is no larger A-invariant subspace for which this inclusion holds. The dimension of $\mathcal{S}$ is the number of eigenvalues of A lying in C, counting multiplicities. We call the C^l- and C^r-invariant subspaces respectively as *left*- and *right*-invariant subspaces for convenience of notation.

We now give an explicit characterization of the matrix-geometric factors R_1 and R_2 in terms of left- and right-invariant subspaces of a real matrix. The methodology described below is mainly based on [4] and [2]. We first need the following definitions to state our first result. Let

$$\begin{aligned}
H_0 &\triangleq A_0 + A_1 + A_2 - I, \\
H_1 &\triangleq 2(A_2 - A_0), \\
H_2 &\triangleq A_0 - A_1 + A_2 + I. \qquad (40)
\end{aligned}$$

Also defining

$$Y_1 \triangleq (R_1 + I)^{-1}(R_1 - I), \tag{41}$$

we have the following theorem:

Theorem 3 *The matrix Y_1 is the unique C^l-stable solution to the matrix equation*

$$H_0 + YH_1 + Y^2H_2 = 0. \tag{42}$$

Proof: To show this, we use direct substitution:

$$\begin{aligned}(I+R_1)^2(H_0 + Y_1H_1 + Y_1^2H_2) &= (I+R_1)^2H_0 + (I+R_1)(R_1 - I)H_1 + (R_1 - I)^2H_2, \\ &= (H_0 - H_1 + H_2) + R_1(2H_0 - 2H_2) + R_1^2(H_0 + H_1 + H_2), \\ &= 4(A_0 + R_1A_1 + R_1^2A_2 - R_1), \text{ by } (40) \\ &= 0, \text{ by } (6).\end{aligned}$$

Since $(I + R_1)$ is nonsingular, Y_1 satisfies the matrix equation (42). Note that the transformation

$$z \to \frac{z-1}{z+1} \tag{43}$$

is one-to-one, and it maps the open unit disk to the the open left-half plane. Therefore, Y_1 is C^l-stable, and uniqueness follows from the fact that R_1 is the unique solution to the matrix equation (6) all eigenvalues of which lie in the open unit disk. □

Now defining

$$Y_2 \triangleq (I + R_2)^{-1}(I - R_2), \tag{44}$$

similarly, we have the following dual of Theorem 3, which we state without proof.

Theorem 4 *The matrix Y_2 is the unique C^{cr}-stable solution to the matrix equation (42).*

We now define the following matrix of size $2m \times 2m$:

$$E \triangleq \begin{bmatrix} 0 & -\hat{H}_0 \\ I & -\hat{H}_1 \end{bmatrix}, \tag{45}$$

where

$$\hat{H}_i \triangleq H_iH_2^{-1}, \quad i = 0, 1, \tag{46}$$

and the invertibility of H_2 follows directly from assumption (13) since

$$H_2 = A(-1) + I = -[zI - A(z)]|_{z=-1}.$$

Next, we present our result concerning the relationship between the matrices Y_1 and Y_2 and certain invariant subspaces of the matrix E^T, where M^T denotes the transpose of a matrix M.

Theorem 5 *The columns of the matrices*

$$\begin{bmatrix} I \\ Y_1^T \end{bmatrix} \quad \textit{and} \quad \begin{bmatrix} I \\ Y_2^T \end{bmatrix}$$

form bases for the C^l- and C^{cr}-invariant subspaces of E^T, respectively.

Proof: The proof follows by observing that

$$Y_i \begin{bmatrix} I & Y_i \end{bmatrix} = \begin{bmatrix} I & Y_i \end{bmatrix} E, \quad i = 1, 2, \tag{47}$$

holds by using Theorems 3 and 4 and the definitions on invariant subspaces. □

We note that the spectrum of the matrix E satisfies $\sigma(E) = \sigma(Y_1) \cup \sigma(Y_2)$ by (47). By definitions (41) and (44) of Y_1 and Y_2, the matrix E has m eigenvalues in the open left-half plane, one simple eigenvalue at $\lambda = 0$ and $m-1$ eigenvalues in the open right-half plane.

Recalling from (12) that x is the invariant vector of $A(1)$, let us define two vectors α and β of length $2m$ as follows:

$$\alpha \triangleq \begin{bmatrix} x & 0 \end{bmatrix}, \quad \beta \triangleq \begin{bmatrix} H_1 e \\ H_2 e \end{bmatrix}. \tag{48}$$

It is easy to show that

$$\alpha E = 0, \quad E\beta = 0,$$

i.e., α and β are respectively the left- and right-eigenvectors of E associated with the simple eigenvalue at $\lambda = 0$. Therefore, the matrix E_m defined by

$$E_m \triangleq E + \frac{\beta\alpha}{\alpha\beta} \tag{49}$$

is free of eigenvalues on the imaginary axis since the simple eigenvalue of E at $\lambda = 0$ is now moved to $\lambda = 1$ (to the right-half plane) by this transformation. That is, E_m has m eigenvalues in each open left- and right-half planes. The following theorem is based on [4], and stated here without proof.

Theorem 6 *The columns of the matrices*

$$\begin{bmatrix} I \\ Y_1^T \end{bmatrix} \quad \textit{and} \quad \begin{bmatrix} I \\ Y_2^T \end{bmatrix}$$

form bases for the left- and right-invariant subspaces of E_m^T, respectively.

Once the left- and right-invariant subspaces of the matrix E_m^T are obtained, it is easy to compute the matrix-geometric factors R_1 and R_2. To see this, let

$$T \triangleq \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \tag{50}$$

be an arbitrary basis for the left-invariant subspace of E_m^T. But, since

$$\mathrm{Im}\, T = \mathrm{Im} \begin{bmatrix} I \\ Y_1^T \end{bmatrix},$$

we have $Y_1 = (T_2 T_1^{-1})^T$. By definition (41) of Y_1, the matrix-geometric factor R_1 can be found as follows:

$$R_1 = (I + Y_1)(I - Y_1)^{-1} = (T_1 - T_2)^{-T}(T_1 + T_2)^T, \tag{51}$$

where $M^{-T} \triangleq (M^{-1})^T = (M^T)^{-1}$ for a nonsingular matrix M. Now defining

$$U \triangleq \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \tag{52}$$

as an arbitrary basis for the right-invariant subspace of E_m^T, one can similarly show that the matrix-geometric factor R_2 is given by

$$R_2 = (U_1 + U_2)^{-T}(U_1 - U_2)^T. \tag{53}$$

3.2 IMPLEMENTATION

We focus on two particular methods to compute a basis for each of the left- and right-invariant subspaces of a real matrix: *(i)* matrix-sign function method, and *(ii)* Schur decomposition method. While the latter method is known of its numerical stability [30], [21], the former one is promising especially for large-scale problems as it is amenable to parallel implementation [38]. Both methods have also been successfully used in the past to solve important equations in control theory, such as matrix Riccati and Lyapunov equations [40], [15], [19], [30]. Here, given the matrix E_m^T, we aim at *simultaneous* computation of the matrix-geometric factors R_1 and R_2 through finding either the matrix-sign or the Schur decomposition of E_m^T. The two methods are introduced next in that spirit.

3.2.1 MATRIX-SIGN FUNCTION METHOD

The use of matrix-sign function for solving certain nonlinear matrix equations arising in M/G/1 and G/M/1 paradigms has been recently proposed in [4]. Here, we follow the same approach as in [4], and start with some introductory material regarding the definition and simple properties of the matrix-sign function based on [40] and [31].

Definition 1 *Let* $M \in \mathbf{R}^{n\times n}$ *with no pure imaginary eigenvalues, i.e.,* $\sigma(M) \cap C^0 = \emptyset$. *Let* M *have a Jordan decomposition* $M = T(D+N)T^{-1}$ *where* $D = \mathrm{diag}\,\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ *and* N *is nilpotent and commutes with* D. *Then, the matrix-sign of* M *is given by*

$$Z = sgn(M) \triangleq T \,\mathrm{diag}\,\{sgn(\lambda_1), sgn(\lambda_2), \ldots, sgn(\lambda_n)\}\, T^{-1}, \qquad (54)$$

where for a complex scalar z *with* $\mathrm{Re}\, z \neq 0$, *the sign of* z *is defined by*

$$sgn(z) \triangleq \begin{cases} 1 & \text{if } Re\ z > 0 \\ -1 & \text{if } Re\ z < 0 \end{cases}$$

Here are some elementary and easily obtainable properties of $Z = sgn(M)$ [31]:

- Z is diagonalizable with eigenvalues ± 1, $Z^2 = I$, and $Z = Z^{-1}$,
- Im $(Z - I)$ = left-invariant subspace of M,
- Im $(Z + I)$ = right-invariant subspace of M.

Therefore, once the matrix-sign of a real matrix with no eigenvalues on the imaginary axis is given, finding basis vectors for its left- and right-invariant subspaces is not difficult. Based on [8], we outline below a numerically stable and efficient method for this purpose.

Let us assume r eigenvalues of an $n \times n$ real matrix M lie in the open left-half plane counting multiplicities, and let $Z = sgn(M)$. Also let the rank-revealing QR decomposition [16] of the matrix $Z - I$ be

$$Z_l \triangleq Z - I = Q_l \, R_l \, \Pi_l, \tag{55}$$

where R_l is upper triangular, Q_l is orthogonal, and Π_l is a permutation matrix. Suppose that the permutation Π_l is chosen so that the rank deficiency of Z_l is exhibited in R_l by a smaller lower-right block in norm of size $(n - r) \times (n - r)$. Then, the leading r columns of Q_l span Im Z_l, or equivalently, form an orthogonal basis for the left-invariant subspace of M. It is also clear that the rank-revealing QR decomposition of $Z + I$,

$$Z_r \triangleq Z + I = Q_r \, R_r \, \Pi_r \tag{56}$$

with proper choice of permutation Π_r, similarly yields an orthogonal basis for the right-invariant subspace of M. For a pseudo-code of a particular algorithm on rank-revealing QR decomposition (QR with column pivoting), see [21, pp. 233-236]. We also note that this decomposition is readily available in linear algebra software packages like MATLAB [32] and LAPACK [6].

Turning back to matrix-sign function evaluation, Definition 1 above does not lend itself to an efficient computation; but, there are several ways of this evaluation. The simplest iterative scheme is Newton's method applied to $sgn(M)^2 = I$:

$$Z_0 = M,\ Z_{k+1} = \frac{1}{2}(Z_k + Z_k^{-1}). \tag{57}$$

This iteration is called the classical matrix-sign function algorithm, and it *converges quadratically* to $sgn(M)$ for any matrix M whose matrix-sign is well-defined, i.e., the matrix M does not have any eigenvalue on the imaginary axis [40]. There are also ways of accelerating iterations by scaling. One popular way of scaling is based on [15]:

$$Z_{k+1} = \frac{1}{2}(\gamma_k Z_k + \gamma_k^{-1} Z_k^{-1}),\quad \gamma_k = |\det Z_k|^{-1/n}, \tag{58}$$

where n is the order of matrix M. A comparison of different scaling schemes is presented by Balzer [9] and Kenney and Laub [26]. Furthermore, iterations with higher convergence rates (e.g., cubic convergence) have also been proposed in [25] at the expense of increased computational load at each iteration as compared to the iteration given in (58). We propose the following stopping criterion based on [7]:

$$||Z_{k+1} - Z_k||_1 < \varepsilon\, ||Z_k||_1, \tag{59}$$

where ε is a small, user-specified error bound.

Recalling equations (51) and (53), all above discussion suggests that the matrix-geometric factors R_1 and R_2 can be computed by performing two rank-revealing QR decompositions after finding the matrix-sign of E_m^T. Since the computational cost of matrix-sign iterations here is significantly more than that of one rank-revealing QR decomposition (about 10-to-1 ratio measured in several trials (see Table 2 in Section 5), the overall procedure can be said to yield R_1 and R_2 simultaneously.

3.2.2 SCHUR DECOMPOSITION METHOD

Given a square real matrix M, the Schur decomposition amounts to an orthogonal similarity transformation on M to an upper quasi-triangular matrix, and provides direct information about certain invariant subspaces of M. Below we outline a procedure for computing the left- and right-invariant subspaces of E_m^T simultaneously via the so-called *ordered real Schur decomposition.*

We start by stating without proof the following combined theorem based on [21, Ch. 7].

Theorem 7 *Let $M \in \mathbf{R}^{n\times n}$.*

(i) *Then, there exists an orthogonal matrix $Q \in \mathbf{R}^{n\times n}$ such that*

$$Q^T M Q = D = \begin{bmatrix} D_{11} & D_{12} \\ 0 & D_{22} \end{bmatrix}, \tag{60}$$

where D is upper block triangular with either 1-by-1 or 2-by-2 diagonal blocks (hence the term quasi-triangular) *respectively comprising real and complex conjugate eigenvalues of M in any desired order.* (The matrix D is said to be in real Schur form, and the columns of matrix Q are referred to as Schur vectors.)

(ii) *Furthermore, if partition D_{11} is $r \times r$ and $\sigma(D_{11}) \cap \sigma(D_{22}) = \emptyset$, then the first r columns of Q span the unique invariant subspace of M associated with $\sigma(D_{11}) \subset \sigma(M)$.*

As Theorem 7 suggests, one can find *orthogonal* bases for the left- and right-invariant subspaces of a given real matrix via two ordered real Schur decompositions. However, our formulation is based on finding *arbitrary* bases (not necessarily orthogonal) for the two subspaces of E_m^T. Therefore, we take an alternative approach that requires only one Schur decomposition as described below.

Given any $M \in \mathbf{R}^{n\times n}$ and its Schur decomposition as in (60), suppose that we find a nonsingular $Y \in \mathbf{R}^{n\times n}$ such that

$$Y^{-1}MY = \begin{bmatrix} D_{11} & 0 \\ 0 & D_{22} \end{bmatrix} \tag{61}$$

by eliminating the upper-right partition D_{12} of matrix D. Then, by an immediate corollary to Theorem 7 Part (ii), the first r and at the same time the last $n-r$ columns of Y form bases for the invariant subspaces of M associated respectively with $\sigma(D_{11})$ and $\sigma(D_{22})$, provided that these two spectra are disjoint. To find Y, we first block diagonalize matrix D of (60) as follows:

$$\begin{bmatrix} I & X \\ 0 & I \end{bmatrix} \begin{bmatrix} D_{11} & D_{12} \\ 0 & D_{22} \end{bmatrix} \begin{bmatrix} I & -X \\ 0 & I \end{bmatrix} = \begin{bmatrix} D_{11} & 0 \\ 0 & D_{22} \end{bmatrix}, \tag{62}$$

where $X \in \mathbf{R}^{r\times(n-r)}$ is found by solving the *Sylvester matrix equation* [21, Ch. 7]

$$D_{11}X - XD_{22} = D_{12}. \tag{63}$$

Then, assuming without loss of generality that the matrix Q of (60) is *orthonormal*, Y follows from (60), (61), and (62) as

$$Y = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \begin{bmatrix} I & -X \\ 0 & I \end{bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} - Q_{11}X \\ Q_{21} & Q_{22} - Q_{21}X \end{bmatrix}, \tag{64}$$

where Q is partitioned accordingly. For more information on real Schur decomposition and block diagonalization, we refer the reader to [21, Ch. 7]. We also note that appropriate computational routines are readily available in LAPACK [6] both for ordered real Schur decomposition with orthonormal Q matrix and for solving the Sylvester matrix equation (63) with upper quasi-triangular coefficient matrices D_{11} and D_{22}.

To summarize, given the $2m \times 2m$ matrix E_m^T, we obtain its real Schur decomposition by selecting m eigenvalues with negative real parts to be placed in the leading diagonal blocks. This immediately gives an orthogonal basis for the left-invariant subspace of E_m^T. For the right-invariant subspace, we perform block diagonalization as above. Since the computational cost of this procedure, i.e., solving the Sylvester matrix equation (63), is negligible as compared to that of Schur decomposition (about 20-to-1 ratio measured in several trials (see Table 2 in Section 5), the overall procedure can be said to yield both subspaces, hence both matrix-geometric factors R_1 and R_2 simultaneously.

4 NUMERICAL ALGORITHM AND EXTENSIONS

Table 1 summarizes all the steps for finding the stationary distribution π for the finite QBD process with the transition probability matrix P given as in (1) when the traffic parameter γ defined in (14) is less than zero. Note that the algorithm is mainly based on simple block matrix operations including additions, multiplications, and solving systems of linear equations. Most of the computation is devoted to the second step where the matrix-sign or the Schur decomposition of the matrix E_m^T is obtained.

In the rest of this section, we first provide the necessary adjustments for the case of $\gamma > 0$ in Section 4.1. We then discuss how continuous-time QBD chains can be handled in Section 4.2, and the extension of the algorithm to the case of level-dependent transitions in Section 4.3. Section 4.4 addresses the case of a non-canonical transition probability matrix via a discrete-time PH/PH/1/K queue example.

4.1 EXTENSION TO THE CASE OF $\gamma > 0$

When the traffic parameter γ of (14) is greater than zero, the form of solution is still the same as in (9), but the properties of the matrix-geometric factors R_1 and R_2 are swapped. Based on [18], $\Delta_1(z)$ of (33) now has exactly $m-1$ roots in the open unit disk and a simple root at $z = 1$. Furthermore, R_1 turns out to be the unique power-bounded solution to equation (6), and has a simple eigenvalue at $z = 1$. Similarly, R_2 becomes the unique power-

<table>
<tr><th></th><th>Matrix-sign function method</th><th>Schur decomposition method</th></tr>
<tr><td>1.</td><td colspan="2">Define $H_i,\ i = 0, 1, 2$ as in (40), $\hat{H}_i,\ i = 0, 1$ as in (46), E as in (45), α, β as in (48) and E_m as in (49). (Note that the matrix E_m^T is $2m \times 2m$, and does not have any eigenvalues on the imaginary axis).</td></tr>
<tr><td>2.</td><td>Find Z, the matrix-sign of E_m^T.</td><td>Find Q and D, as in (60), from the ordered real Schur decomposition of E_m^T.</td></tr>
<tr><td>3.</td><td>Find Q_l, as in (55), from the rank-revealing QR decomposition of $Z - I$. The first m columns of Q_l then gives the matrix T of (50).</td><td>Find X by solving the Sylvester matrix equation (63).</td></tr>
<tr><td>4.</td><td>Find Q_r, as in (56), from the rank-revealing QR decomposition of $Z + I$. The first m columns of Q_r then gives the matrix U of (52).</td><td>Find Y as in (64). The first m and the last m columns of Y then give the matrices T of (50) and U of (52), respectively.</td></tr>
<tr><td>5.</td><td colspan="2">Write the matrix-geometric factors R_1 and R_2 as in (51) and (53), respectively.</td></tr>
<tr><td>6.</td><td colspan="2">Define x, y, and M as in (12), (35), and (36), respectively. Also defining $\hat{R}_2 = R_2 - M$, form the matrix $B[R_1, R_2]$ as in (15), and write S_1 and S_2 as in (34) and (39).</td></tr>
<tr><td>7.</td><td colspan="2">Find the left-invariant eigenvector $v = [\ v_1\ \ v_2\ ]$ of $B[R_1, R_2]$ normalized as in (27).</td></tr>
<tr><td>8.</td><td colspan="2">Write $$\pi_k = v_1 R_1^k + v_2 R_2^{K-k},\ 0 \le k \le K,$$ which is the solution vector of level k of the QBD process.</td></tr>
</table>

Table 1: Summary of the overall numerical algorithm for the case of $\gamma < 0$.

summable solution to equation (10), and therefore, it has all its eigenvalues inside the unit disk. Essentially, the eigenvalue at $z = 1$ moves from R_2 to R_1 as γ moves from 0^- to 0^+ on the real line which makes it necessary to make some adjustments in the numerical algorithm. The modifications are simple, and involve only the definition of E_m and the normalization equations.

Instead of obtaining E_m as in (49) in Step 1, this time we define

$$E_m \triangleq E - \frac{\beta\alpha}{\alpha\beta},$$

where the eigenvalue at $\lambda = 0$ is now moved to $\lambda = -1$ (to the left-half plane). The matrix-geometric factors are then obtained in the same way as in Steps 2-5 from the matrix-sign or Schur decomposition of E_m^T.

For the normalization equations, first note that the expression (34) is not valid any more since R_1 is not invertible. Defining x as in (12), and y and N respectively as

$$y \triangleq (A_0 - R_1 A_2) e_1,$$

and

$$N \triangleq \frac{yx}{xy},$$

one can show that x and y are the left- and right-invariant eigenvectors of R_1, respectively. Then, similar to (38), the following holds:

$$R_1^k = \begin{cases} \hat{R}_1^k + N & \text{if } k \geq 1, \\ I & \text{if } k = 0, \end{cases}$$

where $\hat{R}_1 \triangleq R_1 - N$. Based on above, we have

$$S_1 = \sum_{k=0}^{K} R_1^k = (I - \hat{R}_1^{K+1})(I - \hat{R}_1)^{-1} + K\,N.$$

On the other hand, since R_2 is now invertible, S_2 is easier to write:

$$S_2 = \sum_{k=0}^{K} R_2^k = (I - R_2^{K+1})(I - R_2)^{-1}.$$

Except for the modifications involving E_m, S_1, and S_2 everything else in the numerical algorithm remains the same, and we still have the mixed matrix-geometric expression (9) for the solution vectors.

4.2 THE CONTINUOUS-TIME QBD PROCESS

For the sake of completeness, we now present the well-known technique of uniformization to solve continuous-time QBD processes using the method of this paper developed for discrete-time QBD chains.

Continuous-time Markov chains that have an infinitesimal generator matrix of block tridiagonal form

$$Q = \begin{bmatrix} B_0 & A_0 & & & & \\ B_1 & A_1 & A_0 & & & \\ & A_2 & A_1 & \ddots & & \\ & & A_2 & \ddots & A_0 & \\ & & & \ddots & A_1 & C_0 \\ & & & & A_2 & C_1 \end{bmatrix} \tag{65}$$

are referred to as continuous-time QBD processes. Determining the stationary probability vector π of this process, which satisfies

$$\pi Q = 0 \text{ and } \pi e_{(K+1)} = 1,$$

is required in a wide variety of performance analysis problems involving QBD processes [43]. Following the uniformization technique as in [27], we define

$$q \triangleq \max_i(-Q_{ii}).$$

Then, the matrix defined as

$$P \triangleq I + \frac{Q}{q}$$

is a stochastic matrix of the form (1). Moreover, the invariant vector of the discrete-time QBD process thus obtained coincides with the stationary probability vector of the original continuous-time process. Therefore, by this uniformization technique, one can obtain a transition probability matrix of QBD type and use the proposed method to find the stationary solution of continuous-time QBD processes.

4.3 LEVEL-DEPENDENT TRANSITIONS

Markov chains with transition probability matrices of block tridiagonal form for which transition probabilities are allowed to depend on levels are referred to as Level-Dependent QBD (LDQBD) processes; see [13] and [39] for a detailed discussion. We consider a special case of LDQBD processes for which the transition probability matrices are piecewise homogeneous and the method of this paper can be employed to develop a computationally efficient algorithm. This kind of processes arises in some queueing problems that involve overload control policies, e.g., selective packet discarding in a packet multiplexer when the queue length reaches to a pre-specified threshold [44]. In such cases, the transition probability matrix may have a piecewise uniform block tridiagonal form [43]. As an example, the transition probability matrix for a finite QBD process with one level of control has the following form:

$$P = \begin{array}{c} \begin{array}{cccccccccc} 0 & 1 & & L-1 & L & L+1 & L+2 & & K-1 & K \end{array} \\ \left[\begin{array}{cccccccccc} B_0 & A_0 & & & & & & & & \\ B_1 & A_1 & \ddots & & & & & & & \\ & A_2 & \ddots & A_0 & & & & & & \\ & & \ddots & A_1 & E_0 & & & & & \\ & & & A_2 & E_1 & F_0 & & & & \\ & & & & E_2 & F_1 & D_0 & & & \\ & & & & & F_2 & D_1 & \ddots & & \\ & & & & & & D_2 & \ddots & D_0 & \\ & & & & & & & \ddots & D_1 & C_0 \\ & & & & & & & & D_2 & C_1 \end{array}\right] \end{array} \tag{66}$$

In the above structure, L is the particular control level, A_i and D_i, $i = 0, 1, 2$, govern the transitions in two different regimes, E_i and F_i, $i = 0, 1, 2$, model how transitions from one regime to the other take place; and, B_i and C_i, $i = 0, 1$, govern the transitions on the boundaries as before. Multi-level controls can also be imposed on the QBD process, but we only focus on the transition probability matrix given in (66) in the sequel since otherwise notation may become cumbersome although generalization is easy. We outline the approach below without giving detailed proofs since the theoretical treatment is similar to that of Section 2 for the level-independent finite QBD process.

We first write

$$\pi_k = v_{A,1} R_{A,1}^k + v_{A,2} R_{A,2}^{L-k}, \quad 0 \le k \le L, \tag{67}$$

where $R_{A,1}$ ($R_{A,2}$) is the unique power-summable (power-bounded) solution to the matrix equation

$$R = A_0 + RA_1 + R^2 A_2 \qquad (R = A_2 + RA_1 + R^2 A_0).$$

We similarly write

$$\pi_k = v_{D,1} R_{D,1}^{k-(L+1)} + v_{D,2} R_{D,2}^{K-k}, \; L+1 \le k \le K, \tag{68}$$

where $R_{D,1}$ ($R_{D,2}$) is the unique power-summable (power-bounded) solution to the matrix equation

$$R = D_0 + RD_1 + R^2 D_2 \qquad (R = D_2 + RD_1 + R^2 D_0).$$

By this choice of form for π_k, all the block equations of $\pi P = \pi$ are satisfied (can easily be proven following the lines of (31)) except for the four equations below corresponding to the boundary levels 0, L, $L+1$, and K:

$$\begin{aligned}
\text{level } 0 &: \; \pi_0 B_0 + \pi_1 B_1 = \pi_0 \\
\text{level } L &: \; \pi_{L-1} E_0 + \pi_L E_1 + \pi_{L+1} E_2 = \pi_L \\
\text{level } (L+1) &: \; \pi_L F_0 + \pi_{L+1} F_1 + \pi_{L+2} F_2 = \pi_{L+1} \\
\text{level } K &: \; \pi_{K-1} C_0 + \pi_K C_1 = \pi_K
\end{aligned}$$

Substituting expressions (67) and (68) for π_k in the above equations, we obtain the following system of linear equations which is an extension of (26):

$$v = v \, B[R_{A_1}, R_{A,2}, R_{D,1}, R_{D,2}], \tag{69}$$

where the unknown vector v is partitioned as

$$v \triangleq \left[\begin{array}{cccc} v_{A,1} & v_{A,2} & v_{D,1} & v_{D,2} \end{array} \right],$$

and the matrix $B[R_{A_1}, R_{A,2}, R_{D,1}, R_{D,2}]$ on the right-hand side is given by

$$\left[\begin{array}{ccc}
B_0 + R_{A,1} B_1 & R_{A,1}^{L-1}(E_0 + R_{A,1} E_1 - R_{A,1}) & : \\
R_{A,2}^{L-1}(R_{A,2} B_0 + B_1 - R_{A,2}) & R_{A,2} E_0 + E_1 & : \\
0 & E2 & : \\
0 & R_{D,2}^{K-L-1} E_2 & :
\end{array} \right.$$

$$
\left.\begin{array}{ccc}
: & R_{A,1}^{L}F_0 & 0 \\
: & F_0 & 0 \\
: & F1 + R_{D,1}F_2 & R_{D,1}^{K-L-2}(C_0 + R_{D,1}C_1 - R_{D,1}) \\
: & R_{D,2}^{K-L-2}(R_{D,2}F_1 + F_2 - R_{D,2}) & R_{D,2}C_0 + C_1
\end{array}\right] .
$$

By solving (69) for vector v subject to the normalization equation $\sum_{k=0}^{K} \pi_k e_1 = 1$, which can indeed be expressed in a form analogous to (27), we obtain a piecewise uniform mixed matrix-geometric form of solution for the Markov chain (66) through the expressions (67) and (68).

Note that the above treatment of multiple boundary levels can also be used to generalize the proposed method to handle QBD chains that are not in canonical form (1) as demonstrated next.

4.4 NON-CANONICAL TRANSITION PROBABILITY MATRIX

Consider a finite-buffer single-server queue in which inter-arrival times have a discrete-time phase-type distribution characterized by the pair (α_1, S_1) of dimension m_1 with mean λ^{-1}. Let service times similarly have a discrete-time phase-type distribution represented by (α_2, S_2) of dimension m_2 with mean μ^{-1}. In the representation of a phase-type distribution, the first variable is a row vector and the second is a square matrix; see [36] and [37] for details on phase-type distributions. The state-space describing the Markov chain for the PH/PH/1/K queue is given as

$$
\mathcal{S} = \{(0,i) \cup (k,i,j) \ : \ 1 \le k \le K,\ 1 \le i \le m_1,\ 1 \le j \le m_2\},
$$

where $(0,i)$ represents the states when the system is empty and the arrival is in phase i; and, (k,i,j) represents the states when there are k entities in the system, the arrival is in phase i, and the service is in phase j [14]. The transition probability matrix P representing this Markov chain is given by [14]:

$$P = \begin{bmatrix} B_0 & D_0 & & & & \\ B_1 & A_1 & A_0 & & & \\ & A_2 & A_1 & \ddots & & \\ & & A_2 & \ddots & A_0 & \\ & & & \ddots & A_1 & C_0 \\ & & & & A_2 & C_1 \end{bmatrix}, \tag{70}$$

where

$$\begin{array}{lll} B_0 = S_1, & B_1 = S_1 \otimes S_2^e, & D_0 = (S_1^e \alpha_1) \otimes \alpha_2, \\ A_0 = (S_1^e \alpha_1) \otimes S_2, & A_1 = (S_1^e \alpha_1) \otimes (S_2^e \alpha_2) + S_1 \otimes S_2, & A_2 = S_1 \otimes (S_2^e \alpha_2), \\ C_0 = A_0, & C_1 = A_0 + A_1, & S_i^e \triangleq e - S_i e, \quad i = 1, 2, \end{array}$$

e is a column vector of ones of suitable size, and $\otimes$ denotes Kronecker product. Note that B_0 is of size $m_1 \times m_1$, B_1 is $m \times m_1$, D_0 is $m \times m_1$, where $m = m_1 m_2$. The PH/PH/1/K model falls into the QBD paradigm, and can be analyzed using the proposed method although the transition probability matrix (70) is not in canonical form (1). Partitioning the stationary probability vector π as

$$\pi \triangleq \begin{bmatrix} \pi_0 & \pi_1 & \cdots & \pi_K \end{bmatrix},$$

where π_0 is of size $1 \times m_1$ and π_k is $1 \times m$ for $1 \leq k \leq K$, we first write

$$\pi_k = v_1 R_1^{k-1} + v_2 R_2^{K-k}, \quad 1 \leq k \leq K, \tag{71}$$

where R_1 and R_2 are the desired solutions to the quadratic matrix equations (6) and (10), respectively. Following the treatment of Section 4.3 on QBD chains with level-dependent transitions, we substitute expression (71) in the three equations corresponding to levels 0, 1, and K, respectively, of $\pi = \pi P$ to obtain

$$v = v\, B[R_1, R_2], \tag{72}$$

where this time

$$v \triangleq \begin{bmatrix} \pi_0 & v_1 & v_2 \end{bmatrix}$$

and

$$B[R_1, R_2] \triangleq \begin{bmatrix} B_0 & D_0 & 0 \\ B_1 & A_1 + R_1 A_2 & R_1^{K-2}(C_0 + R_1 C_1 - R_1) \\ R_2^{K-1} B_1 & R_2^{K-2}(A_2 + R_2 A_1 - R_2) & R_2 C_0 + C_1 \end{bmatrix}.$$

Solving for the left-invariant eigenvector of $B[R_1, R_2]$ normalized so that the probabilities sum up to unity, we obtain the closed-form expression (71) for the solution of the PH/PH/1/K queue.

5 NUMERICAL STUDY AND DISCUSSION

In this section, the numerical stability and the CPU time requirements of both matrix-sign function and Schur decomposition implementations of the proposed method are examined in comparison with the folding algorithm of Ye and Li [43], the logarithmic reduction algorithm of Latouche and Ramaswami [29], and with the cyclic reduction algorithm of Bini and Meini [10]. The particular numerical example we consider is the continuous-time voice-data multiplexer of Daigle and Lucantoni [17].

5.1 EXAMPLE: A CONTINUOUS-TIME VOICE/DATA MULTIPLEXER

Assume that a communication line is to service both 64 kbps circuit-switched telephone calls and packet-switched data. Each telephone subscriber has exponentially distributed on-hook and off-hook times. Data packets arrive according to a Poisson process, and their lengths are exponentially distributed. The line capacity that is not used in servicing telephone calls is used to transmit data packets. There is, however, a minimum bit-rate guarantee for data traffic as telephone calls are not allowed to consume the whole line capacity. Now assume that the packet buffer is finite. Defining the number of packets that are waiting for or being in service as the level process, and the number of existing telephone calls as the phase process, this system can be modeled as a continuous-time finite QBD process characterized by an infinitesimal generator matrix of the form (65); see [17] for details. Following the uniformization technique described in Section 4.2, this continuous-time QBD process can be transformed into its discrete-time equivalent, and analyzed by the proposed method. Now let M and r^{-1} denote the number of telephone subscribers and the mean off-hook time (in seconds) for each subscriber, respectively. Also let a be the aggregate voice traffic intensity in Erlangs. Fixing M and r^{-1} respectively as 512 and 100 throughout the study, we characterize this system via the following three parameters.

m – *the number of phases.* The line capacity is assumed to be $64m$ kbps, and there can at most be $m-1$ simultaneous voice calls on the line. That is, a transmission capacity of 64 kbps is reserved for data traffic in the worst case.

ρ_v – *the normalized load offered by voice traffic.* Given m and M, a is determined iteratively to get the desired ρ_v.

ρ_d – *the normalized load offered by data traffic.* Fixing the mean packet length as 8,000 bits, the average packet arrival rate becomes $8m\rho_d$ packets/second.

Having defined the parameters, the block matrix elements of the infinitesimal generator matrix Q for this system are obtained as follows; see (65) for the form of Q and [17] for details:

$$\begin{array}{ll} A_0 = 8m\rho_d I, & (A_2)_{ii} = 8(m-i), \;\; 0 \le i < m, \\ (A_1)_{i,i-1} = ir, \;\; 1 \le i < m, & (A_1)_{i,i+1} = ar(1-i/M), \;\; 0 \le i < m-1, \\ B_0 = A_1 + A_2, \quad B_1 = A_2, \quad C_0 = A_0, \quad \text{and} \;\; C_1 = A_0 + A_1. & \end{array}$$

Before presenting the numerical results, we briefly discuss the folding algorithm of Ye and Li [43], and the grounds on which the comparisons with this algorithm were made. This is required because the folding algorithm, with its direct approach to solving a finite QBD chain, differs in spirit from the method of this paper.

5.2 OVERVIEW OF THE FOLDING ALGORITHM

The overall time and space complexity of the folding algorithm is determined by two major phases it goes through: *forward reduction* and *backward expansion* phases. Assuming a K-level chain, forward reduction phase takes $\log_2 K$ steps each eliminating half of the levels. This elimination or *folding* is performed after permuting the levels in a certain way that significantly reduces the complexity of this phase. That is, after this permutation, each folding step requires only two $m \times m$ matrix inversions and finitely many elementary matrix additions and multiplications. In fact, matrix inverses are not explicitly needed, and they simply amount to solving systems of linear equations through LU decomposition. Namely, the time complexity of forward reduction phase is $O(m^3 \log_2 K)$. At the end of this first phase, the original chain is reduced to a one-level Markov chain with only m states, whose stationary distribution (the boundary vector) can be found by classical methods with a negligible time complexity of $O(m^3)$. During backward expansion phase, the stationary solution of the original chain is constructed in $\log_2 K$ *unfolding* steps each doubling the number of known stationary level probability vectors starting from the boundary vector. Since the number of vector-matrix multiplications required is also doubled in each unfolding step, the time complexity of backward expansion phase is $O(Km^2)$. Finally, the overall solution vector needs to be normalized introducing a time complexity of $O(Km)$. As to the space complexity, on the other hand, three $m \times m$ matrices per folding step have to stored in order to allow backward expansion. This means a space complexity of $O(m^2 \log_2 K)$ for forward reduction phase. Since backward expansion phase requires differentiation of individual level probability vectors obtained, it incurs a space complexity of $O(Km)$.

Obviously, the folding algorithm as discussed above is best-suited to the case that the number of levels of the chain is an integral power of 2. It is pointed out in [43] that both time and space requirements may increase by up to 75% for arbitrary number of levels. In case one seeks particular performance measures like the blocking probability or the marginal queue length distribution, a way to perform backward expansion implicitly during forward reduction phase is described in [43]. The $O(m^2 \log_2 K)$ storage requirement of the forward reduction phase can thus be eliminated. For more information on issues regarding implementation of the folding algorithm, we refer the reader to [43] which also brings up the idea of using *off-line* (disk) storage as well as *on-line* (RAM) storage to cope with space complexity that may be excessively high.

5.3 COMPARISONS WITH THE FOLDING ALGORITHM

In the numerical evaluation that follows, we consider two performance measures: *(i)* the overall CPU time required to obtain the entire solution vector π of (2), and *(ii)* the 1-norm of the error vector $\pi - \pi P$. All three algorithms, which we refer in the following by `ISAMSFI`[1], `ISASCHR`[2], and `FOLDING`, are implemented in C, and compiled (by gcc version 2.7.2.1 with suspended debugger and optimizer option `-O3`) and run on a DEC Alpha Server 2100 4/200 supporting DEC OSF/1 V3.0 using IEEE standard double-precision arithmetic with machine precision $\varepsilon \approx 2.2\ 10^{-16}$. Standard CLAPACK library routines (Fortran-to-C translated version of LAPACK [6]) are used for non-elementary matrix operations like LU, QR, and Schur decompositions, solution of system of linear equations, inversion, and solution of Sylvester equation. In the `ISAMSFI` implementation, we use the matrix-sign iteration (58) and the stopping criterion (59) with $\varepsilon = 10^{-8}$. In the `FOLDING` implementation, which is a discrete-time version of the original continuous-time

[1]Invariant Subspace Approach – Matrix-Sign Function Iterations.

[2]Invariant Subspace Approach – SCHuR decomposition.

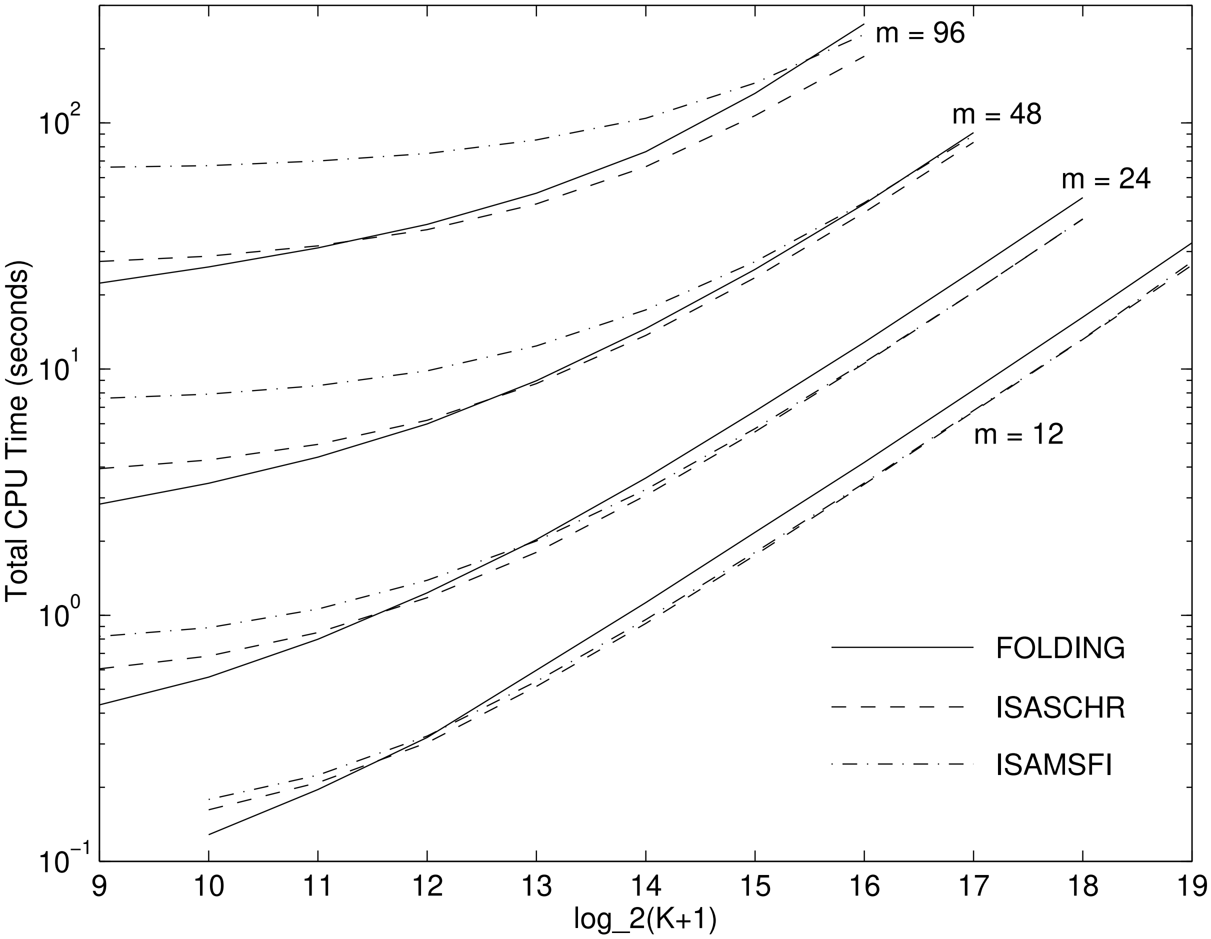

Figure 1: Comparison of total CPU times of the `ISAMSFI`, `ISASCHR`, and `FOLDING` algorithms as a function of the number of levels. System load parameters are fixed as $\rho_v = 0.75$ and $\rho_d = 0.10$. $\varepsilon = 10^{-8}$ is used in stopping criterion for matrix-sign iterations.

algorithm of [43], we do not use off-line storage in order to avoid the CPU overhead associated with disk read/write operations.

Turning back to the voice-data multiplexer example, let K be the packet buffer size. Then, we have a $(K+1)$-level QBD chain to analyze. Fixing ρ_v and ρ_d respectively as 0.75 and 0.10, we first examine the total CPU time as a function of K for different values of m, the number of phases. The results are shown in Figure 1. Note that these results are obtained only at $K+1 = 2^i$ with integer i, and therefore, the `FOLDING` algorithm exhibits a smooth, monotonically increasing CPU time behavior without the degradation mentioned above for non-integral powers of 2. As can be seen,

	`ISAMSFI`		`ISASCHR`		`FOLDING`		
$K+1$	2^9	2^{16}	2^9	2^{16}	2^9	2^{16}	$K+1$
(1) Computing E_m^T	1.108				20.96	39.42	(2) Fwd. Reduction
(1) MSFI/Schur Dec.	44.73		14.76		0.157		(1) Boundary Vect.
(1) 2×QR/Sylvester Eq.	9.227		0.718		1.206	207.9	(3) Bwd. Expansion
(1) Computing R_1/R_2	1.362				0.036	5.158	(4) Normalization
(2) Computing v_1/v_2	8.152	12.56	8.152	12.56			
(3) Computing π_k all k	1.176	157.2	1.176	157.2			
Total CPU Time	65.76	226.2	27.28	187.7	22.36	252.6	Total CPU Time
(1) $O(m^3)$		(2) $O(m^3 \log_2 K)$			(3) $O(Km^2)$		(4) $O(Km)$

Table 2: Detailed CPU time results at $K+1 = 2^9$ and 2^{16} for $m = 96$. System load parameters are fixed as $\rho_v = 0.75$ and $\rho_d = 0.10$. All results are in seconds, and the number of matrix-sign iterations is 11. $K+1 \approx K$ is assumed in complexity figures.

although the CPU times of all three algorithms exhibit the same rate of increase with K, the `ISAMSFI` and `ISASCHR` algorithms become faster than the `FOLDING` algorithm when K exceeds a certain *cross-over* value. For the `ISASCHR` algorithm, this value is $K+1 \approx 2^{12}$, irrespective of the value of m. Such cross-over between the `ISAMSFI` and `FOLDING` algorithms takes place at larger K values as m increases since the number of matrix-sign iterations tend to increase with m (see Table 3). Also note that the `ISASCHR` algorithm is in general faster than the `ISAMSFI` algorithm. Here we point out that all three algorithm implementations are serial in this study although the matrix-sign function computation is amenable to parallel implementation [38].

Another noteworthy observation in Figure 1 is the fixed cost of obtaining matrix-geometric factors R_1 and R_2, which manifests itself as a saturation in the `ISAMSFI` and `ISASCHR` curves for small K. This is the pay-off for the advantages of the simple mixed matrix-geometric form (9) of solution. We further discuss these advantages below by looking at the components of the total CPU time.

Table 2 provides a break-down of CPU time results at two points of Figure 1: $K+1 = 2^9$ and 2^{16} for $m = 96$. As can be seen, the reason for similar asymptotic CPU time behavior of all three algorithms is the domination of the $O(Km^2)$ term. Unlike the FOLDING algorithm for which the backward expansion phase, performed either explicitly or implicitly, is indispensable, the ISAMSFI and ISASCHR algorithms can be relieved of this $O(Km^2)$ time complexity depending on what is sought in the analysis. For example, since R_1^{K-1} is already obtained during the solution, the blocking probability follows simply by

$$P[full] = \pi_K e_1 = (v_2 + v_1 R_1^K) e_1,$$

with a time complexity of only $O(m^2)$ without requiring the computation of π_k for all k. Assuming that necessary powers of R_1 and R_2 required to find v_1 and v_2 are stored, computing moments of the level distribution introduces an additional time complexity of only $O(m^3)$ again without computing π_k for all k. Here, for the first moment for example, one can use the identity

$$\sum_{k=0}^{K} kM^k = (I - M^{K+1})(I - M)^{-2} - (I + KM^{K+1})(I - M)^{-1}$$

which holds for an arbitrary square matrix M provided that it does not have an eigenvalue at unity[3]. From another perspective, once R_1 and R_2 are found, it is not difficult to compute v_1 and v_2 for different values of K. In fact, the overall time of computing v_1 and v_2 for n distinct values of K can be far less than that of n straightforward repetitions. Recalling the equations (15), (34), (39), and (27) from Section 2, it can be seen that, once v_1 and v_2 are found for the smallest K at hand, finding them for larger K requires incremental computations provided that necessary matrices are properly stored. Although the gain achievable by such incremental computations over straightforward repetitions will be influenced by the spread of K values, the proposed method offers a significant advantage for design and dimensioning purposes as the whole solution procedure need not be repeated n times.

Now we fix ρ_v and $K+1$ respectively as 0.5 and 2^{13}, and examine the accuracy of the three algorithms as a function of the total load $\rho = \rho_v + \rho_d$

[3] Otherwise, one can still use it after moving this eigenvalue to the origin by a transformation as discussed in Section 2 for simplifying the expression for S_2.

m	ρ	$P[full]$	ISASCHR	ISAMSFI	(I)	FOLDING
24	0.8	8.14253 10^{-5}	8.5 10^{-17}	1.9 10^{-16}	(9)	1.7 10^{-16}
	0.9	4.35201 10^{-3}	7.0 10^{-17}	9.0 10^{-17}	(9)	1.6 10^{-16}
	0.99	3.92333 10^{-2}	6.8 10^{-17}	2.9 10^{-16}	(12)	2.8 10^{-16}
	0.999	4.65045 10^{-2}	2.8 10^{-16}	3.6 10^{-14}	(15)	5.6 10^{-16}
	0.9999	4.72775 10^{-2}	1.1 10^{-12}	6.1 10^{-13}	(17)	8.2 10^{-17}
	1.0001	4.74504 10^{-2}	7.6 10^{-14}	8.3 10^{-13}	(17)	6.0 10^{-17}
	1.001	4.82336 10^{-2}	8.0 10^{-15}	7.3 10^{-15}	(15)	2.0 10^{-16}
	1.01	5.65168 10^{-2}	4.1 10^{-17}	1.8 10^{-16}	(12)	2.0 10^{-16}
	1.1	1.68075 10^{-1}	1.3 10^{-17}	1.5 10^{-17}	(9)	3.6 10^{-16}
96	0.8	3.85077 10^{-7}	8.4 10^{-17}	1.3 10^{-16}	(11)	3.3 10^{-16}
	0.9	3.20050 10^{-4}	3.0 10^{-17}	4.5 10^{-17}	(10)	2.0 10^{-17}
	0.99	2.17515 10^{-2}	1.8 10^{-17}	4.2 10^{-17}	(12)	2.6 10^{-16}
	0.999	2.91208 10^{-2}	1.4 10^{-15}	2.5 10^{-15}	(15)	3.6 10^{-16}
	0.9999	2.99383 10^{-2}	9.4 10^{-14}	2.4 10^{-13}	(18)	4.4 10^{-16}
	1.0001	3.01219 10^{-2}	6.1 10^{-14}	7.5 10^{-13}	(18)	1.1 10^{-16}
	1.001	3.09573 10^{-2}	8.1 10^{-16}	1.5 10^{-15}	(15)	2.1 10^{-16}
	1.01	4.00973 10^{-2}	6.3 10^{-18}	1.0 10^{-17}	(12)	1.0 10^{-16}
	1.1	1.66739 10^{-1}	6.9 10^{-18}	2.0 10^{-17}	(10)	2.6 10^{-16}

Table 3: Comparison of numerical error, $||\pi - \pi P||_1$, of the ISAMSFI, ISASCHR, and FOLDING algorithms as a function of the system load $\rho = \rho_v + \rho_d$ for $m = 24$ and 96, with ρ_v and $K+1$ fixed as 0.5 and 2^{13}, respectively. $\varepsilon = 10^{-8}$ is used in stopping criterion for matrix-sign iterations, and I in the ISAMSFI results is the number of such iterations.

on the system. The results are given in Table 3 for $m = 24$ and 96. The first to note is that the FOLDING algorithm is highly accurate irrespective of the system load. This is expected due to the direct approach of this algorithm; however, the pay-off for such stability is the lack of flexibility, i.e., strong dependence on K. On the other hand, Table 3 indicates a degradation in the accuracy of the ISAMSFI algorithm for loads extremely close to unity, as well as an increase in the number of matrix-sign iterations. A tendency for an increase in the number of such iterations is also observed with increasing m. In fact, with the load approaching unity, the problem becomes increasingly ill-conditioned, and such accuracy degradations are also reported for infinite QBD processes with available stable algorithms [29]. However, we note that, for the case of a finite QBD system, this constitutes a pathological case: a singularity corresponding to the traffic parameter γ of 0 at which

ρ	$K+1$	$P[full]$	ISASCHR	ISAMSFI	FOLDING
0.8	2^9	$1.21328\ 10^{-2}$	$8.2\ 10^{-17}$	$1.9\ 10^{-16}$	$1.8\ 10^{-16}$
	2^{11}	$4.18946\ 10^{-3}$	$7.9\ 10^{-17}$	$1.9\ 10^{-16}$	$1.7\ 10^{-16}$
	2^{13}	$8.14253\ 10^{-5}$	$8.5\ 10^{-17}$	$1.9\ 10^{-16}$	$1.7\ 10^{-16}$
0.99	2^{11}	$7.15662\ 10^{-2}$	$1.7\ 10^{-16}$	$4.6\ 10^{-16}$	$2.9\ 10^{-16}$
	2^{13}	$3.92333\ 10^{-2}$	$6.8\ 10^{-17}$	$2.9\ 10^{-16}$	$2.8\ 10^{-16}$
	2^{15}	$1.06000\ 10^{-2}$	$2.4\ 10^{-17}$	$1.5\ 10^{-16}$	$2.8\ 10^{-16}$
0.9999	2^{13}	$4.72775\ 10^{-2}$	$1.1\ 10^{-12}$	$6.1\ 10^{-13}$	$8.2\ 10^{-17}$
	2^{15}	$1.83316\ 10^{-2}$	$4.1\ 10^{-13}$	$2.4\ 10^{-13}$	$8.4\ 10^{-17}$
	2^{17}	$5.25339\ 10^{-3}$	$1.2\ 10^{-13}$	$7.0\ 10^{-14}$	$8.4\ 10^{-17}$

Table 4: Numerical error, $||\pi - \pi P||_1$, of the ISAMSFI, ISASCHR, and FOLDING algorithms as a function of the number of levels for $\rho = \rho_v + \rho_d = 0.8$, 0.99, and 0.9999, with ρ_v and m fixed as 0.5 and 24, respectively. $\varepsilon = 10^{-8}$ is used in stopping criterion for matrix-sign iterations.

the solution form (9) does not hold [24] (also see Section 2). Although the ISASCHR algorithm exhibits a similar degradation for loads in the close proximity of unity, it is in general more accurate than the ISAMSFI algorithm. Except when close to this singularity, both ISASCHR and ISAMSFI algorithms are highly accurate, and give errors very close to machine precision without being significantly affected by the number of phases. As for the speed as a function of the system load ρ, although not shown in Table 3, the CPU time of the FOLDING algorithm is not affected by ρ. Neither is that of the ISASCHR algorithm to any significant extent, as opposed to the case of ISAMSFI algorithm where the increase in the number of iterations shows its effect as the load approaches unity.

Finally, Table 4 shows that the accuracy of neither one of the three algorithms is adversely affected by increasing the number of levels. In fact, such numerical stability with respect to K is expected for the ISASCHR and ISAMSFI algorithms since they yield the solution of the infinite QBD process as $K \to \infty$, as was shown in Section 2.

5.4 COMPARISONS WITH THE LOGARITHMIC AND CYCLIC REDUCTION ALGORITHMS

We now compare the `ISAMSFI` and `ISASCHR` algorithms with the `LOGRED` and `CYCRED` algorithms, the logarithmic reduction algorithm of Latouche and Ramaswami [29], and the cyclic reduction algorithm of Bini and Meini [10], respectively. Since these two algorithms belong to the same computational paradigm as ours in that they too are used to find the matrix-geometric factors R_1 and R_2 irrespective of K, and since the rest of the methodology to get the solution of the finite QBD chain is then the same, we only compare the CPU times elapsed until these two matrices are found. As the error measure, on the other hand, we still use the 1-norm of $\pi - \pi P$. The hardware and software platform we made the experiments on, along with the language and compiler options, is the same as described in Section 5.3.

The `LOGRED` and `CYCRED` algorithms are iterative algorithms both with quadratic convergence rates, and are originally introduced in the context of analyzing infinite QBD chains, in particular to solve the quadratic matrix equation

$$G = A_2 + A_1 G + A_0 G^2 \tag{73}$$

for its unique stochastic solution $\hat{G}$. (The exit criterion for both algorithms is based on testing $\hat{G}$ for stochasticity.) The unique minimal nonnegative solution R_1 of (6) then follows as $R_1 = A_0(I - A_1 - A_0\hat{G})^{-1}$ [29]. Recalling the dual nature of the defining equation (10) for R_2 with respect to (6), we use these algorithms to also obtain R_2 through reversing the order of A_0, A_1, and A_2. As for the computational complexity per iteration of these algorithms, both require one inversion of an $m \times m$ matrix whereas `LOGRED` requires eight $m \times m$ matrix multiplications as opposed to six required by `CYCRED`. Here we note that the `ISAMSFI` algorithm requires only one matrix inversion per iteration and no multiplications, but this inversion is of the $2m \times 2m$ matrix E_m^T (see Section 3 and Table 1). The `ISASCHR` algorithm likewise operates on the same $2m \times 2m$ matrix.

m	ρ	ISASCHR	ISAMSFI	(I)	LOGRED	(I_1+I_2)	CYCRED	(I_1+I_2)
24	0.8	0.383	0.600	(9)	0.883	(15+15)	0.817	(16+16)
	0.9	0.400	0.567	(9)	0.967	(17+16)	0.933	(18+17)
	0.99	0.417	0.733	(12)	1.175	(20+20)	1.100	(21+21)
	0.999	0.400	0.850	(15)	1.350	(23+23)	1.250	(24+24)
	0.9999	0.417	0.950	(17)	1.500	(26+26)	1.433	(27+27)
	1.0001	0.400	0.950	(17)	1.500	(26+26)	1.400	(27+27)
	1.001	0.400	0.850	(15)	1.325	(23+23)	1.250	(24+24)
	1.01	0.417	0.750	(12)	1.175	(20+20)	1.100	(21+21)
	1.1	0.433	0.600	(9)	0.950	(16+16)	0.900	(17+17)
96	0.8	17.33	54.70	(11)	64.27	(16+14)	57.75	(17+15)
	0.9	17.42	50.65	(10)	69.95	(17+16)	63.25	(18+17)
	0.99	18.68	59.27	(12)	84.30	(20+20)	75.42	(21+21)
	0.999	18.08	70.95	(15)	96.47	(23+23)	86.87	(24+24)
	0.9999	18.05	82.80	(18)	108.3	(26+26)	96.97	(27+27)
	1.0001	17.95	82.67	(18)	108.3	(26+26)	95.82	(27+27)
	1.001	18.00	70.72	(15)	96.97	(23+23)	85.37	(24+24)
	1.01	18.10	58.75	(12)	83.97	(20+20)	75.22	(21+21)
	1.1	19.42	50.62	(10)	69.97	(16+17)	59.37	(16+17)

Table 5: Comparison of the CPU times of the ISAMSFI, ISASCHR, LOGRED, and CYCRED algorithms as a function of the system load $\rho = \rho_v + \rho_d$ for $m = 24$ and 96, with ρ_v and $K+1$ fixed as 0.5 and 2^9, respectively. All results are in seconds, I indicates the number of matrix-sign iterations, and $I_1 + I_2$ indicates the numbers of iterations in the two successive runs of each of the LOGRED and CYCRED algorithms. $\varepsilon = 10^{-8}$ is used in stopping criteria in all iterations.

The CPU times of the ISASCHR, ISAMSFI, LOGRED, and CYCRED algorithms are compared in Table 5 as functions of the system load $\rho = \rho_v + \rho_d$ for $m = 24$ and 96, where ρ_v and $K + 1$ are fixed as 0.5 and 2^9, respectively. As these results indicate, the number of iterations required by the LOGRED and CYCRED algorithms both increase with the load approaching unity as does the number of matrix-sign iterations. One notable observation is that, unlike the case with the ISAMSFI algorithm, the numbers of LOGRED and CYCRED iterations are virtually insensitive to the number of phases, m. Apart from this, the CYCRED algorithm is consistently faster than the LOGRED algorithm despite the fact that it seems to go through one or two more iterations, which are more than offset by the fewer number of matrix multiplications it has per

ρ_d	`ISASCHR`	`ISAMSFI`	`LOGRED`	`CYCRED`
0.8	1.5e-16	3.1e-17	6.9e-18	6.9e-18
0.9	1.5e-17	1.4e-17	5.2e-18	3.5e-18
0.99	1.0e-16	2.8e-16	1.1e-17	1.6e-17
0.999	7.8e-15	3.0e-14	3.4e-17	1.1e-17
0.9999	4.3e-14	2.0e-12	1.7e-15	1.1e-15
1.0001	4.8e-13	9.9e-13	2.4e-15	2.1e-15
1.001	1.2e-14	1.0e-14	1.7e-17	2.7e-17
1.01	1.7e-16	1.9e-16	9.2e-18	1.2e-17
1.1	2.3e-17	1.3e-17	7.9e-18	1.9e-17

Table 6: Comparison of numerical error, $||\pi - \pi P||_1$, of the `ISAMSFI`, `ISASCHR`, `LOGRED`, and `CYCRED` algorithms as a function of the system load $\rho = \rho_v + \rho_d$ for $m = 24$, with ρ_v and $K+1$ fixed as 0.5 and 2^9, respectively. $\varepsilon = 10^{-8}$ is used in stopping criteria in all iterations.

iteration, six to eight. The `ISAMSFI` algorithm, on the other hand, is faster than both over a wide load range around unity. As for the `ISASCHR` algorithm, not only its speed is largely insensitive to load, but it is significantly faster than the other three algorithms. The speed-up it provides over the others increases with m as well: as m is varied from 24 to 96, the speed-up over the `CYCRED` algorithm changes from about 2 to 3.2 times for low loads, and from about 3.5 to 5.4 times for loads close to unity.

Finally, Table 6 compares the accuracies of the four algorithms as functions of ρ for $m = 24$ only; similar error figures arise for other m values. As these results indicate, the accuracy of the `LOGRED` and `CYCRED` algorithms is not as much sensitive to the system load as that of the `ISASCHR` and `ISAMSFI` algorithms, and all four algorithms are highly accurate except for loads extremely close to unity.

It was argued in Section 5.3 that the accuracy degradation for loads in the close proximity of unity had to do with a singularity in the finite QBD setting. Here, we also point out the example-dependent nature of these numerical stability analyses. See [33], for example, for such analyses in the context of different infinite QBD examples.

5.5 DISCUSSION

Through numerical investigation, it has been observed that the number of levels, $K+1$, of the QBD process plays a minimal role in the performance of the proposed method. First, fixing everything but K results in almost unchanged accuracy in terms of the norm of the error vector. Second, the core of the overall method, which is dedicated to finding R_1 and R_2 via computing the left- and right-invariant subspaces of E_m^T, is independent of K. Finally, the only step where K comes into picture is Step 6 of Table 1, where we construct the matrix $B[R_1, R_2]$ using the matrix powers R_1^{K-1} and R_2^{K-1}. These computations can be done with a time complexity of $O(m^3 \log_2 K)$, and their execution time requirements are typically far less than that of the core, i.e., matrix-sign iterations or Schur decomposition. We therefore conclude that solution of a finite QBD process does not require a significant additional computational effort compared to that of an infinite QBD process, and approximating finite QBD processes by infinite QBD processes to facilitate solutions is generally unnecessary.

References

[1] N. Akar, N. C. Oğuz, and K. Sohraby. Matrix-geometric solutions of M/G/1-type markov chains: a unifying generalized state-space approach. *IEEE J. Select. Areas in Comm.*, 16(5):626–639, 1998.

[2] N. Akar and K. Sohraby. A new paradigm in teletraffic analysis of communication networks. In *Proc. IEEE INFOCOM*, pages 1318–1326, 1996.

[3] N. Akar and K. Sohraby. Finite and infinite QBD chains: a simple and unifying algorithmic approach. In *Proc. IEEE INFOCOM*, 1997.

[4] N. Akar and K. Sohraby. An invariant subspace approach in M/G/1 and G/M/1 type Markov chains. *Commun. Statist. - Stochastic Models*, 13(3), 1997.

[5] F. X. Albores and P. P. Bocharov. Two finite queues with relative priority in a single server system with phase-type distributions. *Avtomatika i Telemeknanika*, 4:96–107, 1993.

[6] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users's Guide*, 1994. Second edition, also available from Web as http://www.netlib.org/lapack.

[7] Z. Bai and J. Demmel. Design of a parallel nonsymmetric eigenroutine toolbox: part I. Computer Science Division Report CSD-92-718, University of California at Berkeley, Dec. 1992.

[8] Z. Bai, J. Demmel, and M. Gu. Inverse free parallel spectral divide and conquer algorithms for nonsymmetric eigenproblems. Computer Science Division Report CSD-94-793, University of California at Berkeley, Feb. 1994.

[9] L. Balzer. Accelerated convergence of the matrix sign function. *Int. Jour. Contr.*, 32:1057–1078, 1980.

[10] D. Bini and B. Meini. On the solution of a nonlinear matrix equation in queueing problems. *SIAM Journal of Matrix Anal. Appl.*, 17:906–926, 1996.

[11] P. P. Bocharov. Analysis of the queueu length and the output flow in single server with finite waiting room and phase-type distributions. *Probl. Control Info. Theory*, 16:211–222, 1987.

[12] P. P. Bocharov and V. A. Naoumov. Matrix-geometric stationary distribution for the PH/PH/1/r queue. *J. Info. Proc. Cyber.*, 22:179–186, 1986.

[13] L. Bright and P. G. Taylor. Calculating the equilibrium distribution in level dependent quasi-birth-and-death processes. *Commun. Statist. - Stochastic Models*, 11(3):497–525, 1995.

[14] J. A. Buzacott and J. G. Shanthikumar. *Stochastic Models of Manufacturing Systems.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[15] R. Byers. Solving the algebraic Riccati equation with the matrix sign function. *Lin. Alg. Appl.*, 85:267–279, 1987.

[16] T. F. Chan. Rank revealing QR factorizations. *Lin. Alg. Appl.*, 88/89:67–82, 1987.

[17] J. N. Daigle and D. M. Lucantoni. Queueing systems having phase-dependent arrival and service rates. In *Numerical Solution of Markov Chains*, pages 223–238. Marcel Dekker, New York, 1991.

[18] H. R. Gail, S. L. Hantler, and B. A. Taylor. Solutions of the basic matrix equation for M/G/1 and G/M/1 type Markov chains. *Commun. Statist. - Stochastic Models*, 10(1):1–43, 1994.

[19] J. D. Gardiner and A. J. Laub. A generalization of the matrix-sign-function solution for algebraic Riccati equations. *Int. Jour. Contr.*, 44:823–832, 1986.

[20] I. C. Gohberg, P. Lancaster, and L. Rodman. *Invariant Subspaces of Matrices with Applications.* John Wiley and Sons, New York, 1986.

[21] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, 1989.

[22] L. Gün. Experimental techniques on matrix-analytical solution techniques - extensions and comparisons. *Commun. Statist. - Stochastic Models*, 5(4):669–682, 1989.

[23] L. Gün and A. A. Makowski. Matrix-geometric solution for finite capacity queues with phase-type distributions. In *Proc. Performance 87*, pages 269–282. North Holland, 1988.

[24] B. Hajek. Birth-and-death processes on the integers with phases and general boundaries. *J. Appl. Prob.*, 19:488–499, 1982.

[25] C. Kenney and A. J. Laub. Rational iterative methods for the matrix sign function. *SIAM Matrix. Anal. Appl.*, 12(2):273–291, 1991.

[26] C. Kenney and A. J. Laub. On scaling Newton's method for polar decomposition and the matrix sign function. *SIAM Jour. Matrix Anal. Appl.*, 13(3):688–706, 1992.

[27] P. R. Kumar and P. Varaiya. *Stochastic Systems: Estimation, Identification, and Adaptive Control.* Prentice-Hall, Englewood Cliffs, NJ, 1986.

[28] P. Lancaster and M. Tismenetsky. *The Theory of Matrices.* Academic Press, New York, 1985.

[29] G. Latouche and V. Ramaswami. A logarithmic reduction algorithm for quasi-birth-death processes. *J. Appl. Prob.*, 30:650–674, 1993.

[30] A. J. Laub. A Schur method for solving algebraic Riccati equations. *IEEE Trans. Auto. Contr.*, 25:913–921, 1979.

[31] A. J. Laub. Invariant subspace methods for the numerical solution of Riccati equations. In S. Bittanti, A. J. Laub, and J. C. Willems, editors, *The Riccati Equation*, chapter 7, pages 163–196. Springer-Verlag, Berlin, 1991.

[32] The MATH WORKS Inc. *MATLAB Users's Guide for UNIX Workstations*, 1991.

[33] B. Meini. Solving QBD problems: the cyclic reduction algorithm versus the invariant subspace method. *Advances in Performance Analysis*, 1:215–225, 1998.

[34] V. A. Naoumov, U. R. Krieger, and D. Wagner. Analysis of a multi-server delay-loss system with a general Markovian arrival process. In S. Chakravarthy and A. Alfa, editors, *Matrix-Analytic Methods in Stochastic Models*, pages 43–66. Marcel Dekker, New York, 1996.

[35] M. F. Neuts. Markov chains with applications in queueing theory, which have a matrix-geometric invariant probability vector. *Adv. Appl. Prob.*, 10:185–212, 1978.

[36] M. F. Neuts. *Matrix-geometric Solutions in Stochastic Models.* Johns Hopkins University Press, Baltimore, MD, 1981.

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

[38] P. Pandey, C. Kenney, and A. J. Laub. A parallel algorithm for the matrix sign function. *International Jour. of High Speed Computing*, 2(2):181–191, 1990.

[39] V. Ramaswami and P. G. Taylor. Some properties of the rate operators in level dependent quasi-birth-and-death processes with a countable number of phases. *Commun. Statist. - Stochastic Models*, 12(1):143–164, 1996.

[40] J. D. Roberts. Linear model reduction and solution of the algebraic Riccati equation by the use of the sign function. *Int. Jour. Control*, 32:677–687, 1980.

[41] R. L. Tweedie. Operator-geometric stationary distributions for Markov chains with applications to queueing models. *Adv. Appl. Prob.*, 14:368–391, 1982.

[42] V. Wallace. *The solution of quasi birth and death processes arising from multiple access computer systems.* PhD thesis, Systems Engineering Laboratory, University of Michigan, 1969.

[43] J. Ye and S. Q. Li. A computational method for finite QBD processes with level-dependent transitions. *IEEE Trans. Commun.*, 42(2):625–639, 1994.

[44] N. Yin, S. Q. Li, and T. E. Stern. Congestion control for packet voice by selective packet discarding. *IEEE Trans. Commun.*, 38(5):674–683, 1990.