A New Algorithm for the Design of Finite Capacity Service Units

Dario Andrea Bini[1]*, Srinivas Chakravarthy[2], and Beatrice Meini[1]

[1] Dipartimento di Matematica
Università di Pisa, 56127 Pisa, Italy.
bini,meini@dm.unipi.it
[2] Department of Science and Mathematics
Kettering University, Flint, USA.
schakrav@kettering.edu

Abstract. In this paper we present a new algorithm for the design of finite capacity service units with phase type arrivals and phase type services. These systems are very useful in manufacturing and production areas. Using cyclic reduction method that exploits both inner and outer structures of the generator of the Markov process governing PH/PH/1/N queue, we develop a new, fast and numerically efficient algorithm for solving the steady state probability vector. This fast algorithm can interactively be used in solving a number of optimization problems that require searching for an optimal value of N. Comparison of our algorithm to that of block Gauss-Seidel iterative procedure and to the matrix sign function approach for selected examples is presented.

1 Introduction

Finite capacity queueing models play an important role in many areas of engineering, notably in production and manufacturing engineering. Jobs (or messages or customers) arrive to a service facility in a random manner and get processed by the server. In this paper, we consider a single server queueing system in which jobs arrive according to a point process governed by a phase type distribution (PH-distribution) with representation $(\boldsymbol{\alpha}, T)$ of order m. The service times are assumed to be of PH-type with representation $(\boldsymbol{\beta}, S)$ of order n. The queueing system has a finite capacity of size N. Any arriving job finding the buffer full is considered lost. Thus, we consider a PH/PH/1/N queue. For details on this queueing model, we refer the reader to [5, 14, 15]. Our focus here is to use the latest development in block tridiagonal Toeplitz systems with structured blocks to develop a new, fast and numerically efficient algorithm for solving the steady-state probability vector of the queueing model under study. This algorithm can interactively be used to solve a number of optimization problems useful in practice. A specific optimization problem that is of interest in practice is mentioned in Section 2. The results of the solution of this and other optimization problems will be reported in a paper sequel to this.

* Supported by the MURST project "Advanced numerical methods for scientific computing".

2 The Model

We consider a single-server queuing system with a finite waiting room of size N. The arrival process is a PH-renewal process where the underlying PH-distribution $F(\cdot)$ has a representation $(\boldsymbol{\alpha}, T)$ of order m. The service times are assumed to be independent, identically distributed with the common PH-distribution $G(\cdot)$ with representation $(\boldsymbol{\beta}, S)$ of order n.

A probability distribution $F(\cdot)$ on $[0, +\infty)$ is a PH-distribution if and only if it is the distribution of the time until absorption in a finite Markov chain of the type

$$\begin{bmatrix} T & \boldsymbol{T}^0 \\ \mathbf{0} & 0 \end{bmatrix}$$

where T is the transition generator, $\boldsymbol{T}^0 = -T\boldsymbol{e}$, $\boldsymbol{e}^T = (1, 1, \ldots, 1)$ and $\mathbf{0}$ denotes the null vector. If $\boldsymbol{\alpha}$ is the initial probability vector, the pair $(\boldsymbol{\alpha}, T)$ is called a representation of $F(\cdot)$. The class of PH-distributions includes many well-known distributions such as generalized Erlang, hyperexponential, etc., as simple special cases and has a number of interesting closure properties. We refer to Neuts [15] for a detailed discussion on this.

Using the PH-formalism as given in Chapter 2 of Neuts [15] efficient algorithms for solving the steady-state probability vectors were developed by Chakravarthy and Neuts [5]. Some interesting relations between various stationary probabilities were derived in that paper. More recently in [9] Latouche and Ramaswami performed a structural analysis in a probabilistic framework that provided a compact representation of the PH/PH/1 queue by means of a QBD process where the block size is reduced to the sum of the number of phases in the arrival and service distributions.

A finite capacity single server queue with phase type arrival and phase type services with a finite buffer size N is denoted by PH/PH/1/N. Any arrivals finding the buffer full will be lost. By keeping track of the phase of the arrivals, the number of customers in the system, and the phase of the service (if the server is busy), this queueing system can be analyzed using matrix formalism [5]. For sake of completeness, we will briefly describe this queueing model. Let $W(t), Y(t)$ and $Z(t)$, respectively, denote the number of customers in the system, the phase of the arrival process, and the phase of the service process (if the server is busy) at time t. Then, $\{W(t), Y(t), Z(t),\ t \geq 0\}$ is a continuous-parameter Markov process with state space given by

$$\Omega = \{(0, j),\ 1 \leq j \leq m\} \cup \{(i, j, k),\ 1 \leq i \leq N+1,\ 1 \leq j \leq m,\ 1 \leq k \leq n\}.$$

The Markov process is in state $(0, j)$ if there are no customers in the system and the arrival process is in phase j. The process is in state (i, j, k) if there are $i > 0$ customers in the system with the arrival process in phase j and the current service is in phase k.

The generator Q of the Markov process is the $(m+(N+1)mn)\times(m+(N+1)mn)$ matrix of the block-tridiagonal Toeplitz-like form

$$Q = \begin{bmatrix} T & \widehat{C} & & & & \bigcirc \\ \widehat{B} & A & C & & & \\ & B & A & C & & \\ & & \ddots & \ddots & \ddots & \\ & & & B & A & C \\ \bigcirc & & & & B & K \end{bmatrix} \tag{1}$$

where

$$\begin{array}{l} \widehat{C} = \boldsymbol{T}^0\boldsymbol{\alpha}\otimes\boldsymbol{\beta}, \quad \widehat{B} = I_m \otimes \boldsymbol{S}^0, \\ A = T\oplus S, \quad C = \boldsymbol{T}^0\boldsymbol{\alpha}\otimes I_n, \quad B = I_m\otimes \boldsymbol{S}^0\boldsymbol{\beta}, \\ K = T\oplus S + \boldsymbol{T}^0\boldsymbol{\alpha}\otimes I_n. \end{array} \tag{2}$$

Here, $\boldsymbol{T}^0$ and $\boldsymbol{S}^0$ are such that $T\boldsymbol{e}+\boldsymbol{T}^0=\boldsymbol{0}$ and $S\boldsymbol{e}+\boldsymbol{S}^0=\boldsymbol{0}$, $\boldsymbol{e}=(1,1,\ldots,1)^T$, the symbols $\otimes$ and $\oplus$ denote the Kronecker product and the Kronecker sum of matrices, respectively and I_r denotes the $r\times r$ identity matrix. We refer to Marcus and Minc [10] for details on Kronecker products and sums.

For the sake of clarity it is useful to specify the size of the blocks involved in the matrix Q, specifically, $T\in\mathbb{R}^{m\times m}$, $S\in\mathbb{R}^{n\times n}$, $\boldsymbol{\alpha}\in\mathbb{R}^{1\times m}$, $\boldsymbol{\beta}\in\mathbb{R}^{1\times n}$, $\boldsymbol{T}^0\in\mathbb{R}^{m\times 1}$, $\boldsymbol{S}^0\in\mathbb{R}^{n\times 1}$ and the matrix Q in (1) has block size $N+2$.

Let $\boldsymbol{y}$ be the steady state probability vector of the state of the Markov process at an arbitrary time. That is, $\boldsymbol{y}$ solves the system $\boldsymbol{y}Q=\boldsymbol{0}$ and $\boldsymbol{y}\boldsymbol{e}=1$.

Observe that the generator Q has a double structure. More precisely an outer block structure (block tridiagonal block Toeplitz-like), and an inner structure given by the Kronecker sums and products, and by the low rank properties of the blocks B and C.

The main purpose of this paper is to exploit the double structure of the generator in order to develop an ad-hoc algorithm for computing the steady state probability vector $\boldsymbol{y}$. This task is mandatory if we want to solve the problem for large values of N. In fact, the customary algorithms would impose a huge computational cost. Moreover, the availability of a fast algorithm would improve the efficiency of the interactive solution of the following optimization problem.

Optimization Problem: Find a value of N so that, given real numbers $\epsilon>0$ and $x^*>0$, we have $P(X>x^*)<\epsilon$, where $X=X(N)$ is the stationary waiting time of an admitted customer into the system.

The task of exploiting both the inner and the outer structure of Q is non-trivial, in fact the algorithms (like iterative methods as Gauss-Seidel iteration) that exploit the inner structure do not fully exploit the outer structure, and the algorithms that exploit the outer structure (like cyclic reduction introduced in [4]) may destroy the inner structure.

In practice, for large size of the blocks and for large block size of the matrix Q, it is necessary to exploit both structures. Moreover in solving the optimization problem we need to solve several systems, with different values of the block dimension $N+2$.

In the next section we propose a method, based on cyclic reduction, that exploits both inner and outer structures of Q, and that is suitable for solving the above mentioned optimization problem. Specifically, following the results of [13] and [12], due to the block tridiagonal block Toeplitz-like structure of Q, we may express the block components of the invariant vector $\boldsymbol{y}$ in terms of the powers of the matrices R and U, which solve two suitable quadratic matrix equations. The matrices R and U do not depend on the block size N of Q, but only depend on the block entries A, B and C. Thus, in the interactive solution of the optimization problem for different values of N, R and U may be computed only once. In order to compute R and U we apply the cyclic reduction algorithm, where the general iteration formulas are expressed in a different form in order to take advantage of the inner structure.

The computational cost of one step of the new algorithm obtained in this way is reduced from $O(m^3n^3)$ to $O((m+n)^3)$ arithmetic operations.

3 Solving a block tridiagonal Toeplitz system with structured blocks

In this section we are concerned with the design of an efficient numerical method for solving the following linear system

$$\boldsymbol{y}Q = \boldsymbol{0}, \quad \boldsymbol{y}\boldsymbol{e} = 1, \tag{3}$$

where Q is the matrix of (1).

By partitioning $\boldsymbol{y}$ as $\boldsymbol{y} = (\boldsymbol{y}_i)_{i=0,N+1}$, according to the block structure of Q, we may rewrite (3) in terms of a homogeneous linear matrix difference equation

$$\boldsymbol{y}_{i-1}C + \boldsymbol{y}_iA + \boldsymbol{y}_{i+1}B = \boldsymbol{0}, \quad i = 2, 3, \ldots, N, \tag{4}$$

with the boundary conditions

$$\begin{cases} \boldsymbol{y}_0T + \boldsymbol{y}_1\widehat{B} = \boldsymbol{0} \\ \boldsymbol{y}_0\widehat{C} + \boldsymbol{y}_1A + \boldsymbol{y}_2B = \boldsymbol{0} \\ \boldsymbol{y}_NC + \boldsymbol{y}_{N+1}K = \boldsymbol{0}. \end{cases} \tag{5}$$

Let the matrices R and U be the minimal nonnegative solutions of

$$\begin{aligned} C + RA + R^2B = 0, \\ U^2C + UA + B = 0. \end{aligned} \tag{6}$$

Then, by a direct inspection it follows that for any $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^{1\times mn}$, the vectors

$$\boldsymbol{y}_i = \boldsymbol{u}R^{i-1} + \boldsymbol{v}U^{N+1-i}, \quad i = 1, \ldots, N+1, \tag{7}$$

solve the equation (4). Furthermore from (5) we have $\boldsymbol{y}_0 = -\boldsymbol{y}_1 \widehat{B} T^{-1}$. Imposing the boundary conditions (5) allows us to compute the vectors $\boldsymbol{u}, \boldsymbol{v}$. In fact, from (7) and (5) we have

$$[\boldsymbol{u}\ \boldsymbol{v}] \begin{bmatrix} A - \widehat{B}T^{-1}\widehat{C} + RB & R^{N-1}(C+RK) \\ U^N(A - \widehat{B}T^{-1}\widehat{C}) + U^{N-1}B & UC + K \end{bmatrix} = \mathbf{0}. \tag{8}$$

This system has a solution since its matrix W is singular (it is a simple matter to verify that $W\boldsymbol{e} = \mathbf{0}$). The same formulas can be found in [1], [12] and [13].

In this way the problem of computing $\boldsymbol{y}$ is reduced to solving the matrix equations (6), the homogeneous system (8) and to applying (7). For the normalization of the vector obtained in this way we may impose the condition

$$\boldsymbol{y}\boldsymbol{e} = \boldsymbol{u}\sum_{i=0}^{N} R^i \boldsymbol{e} + \boldsymbol{v}\sum_{i=0}^{N} U^i \boldsymbol{e} - (\boldsymbol{u} + \boldsymbol{v}U^N)\widehat{B}T^{-1}\boldsymbol{e} = 1.$$

It would be interesting to analyze the sensitivity of the solution $\boldsymbol{y}$ to perturbations of R and U. Even though this problem goes beyond the purpose of this paper, we believe that a conditioning analysis can be carried out by using the tools of [11], more specifically the matrix identity

$$X^i - Y^i = \sum_{j=1}^{i} X^{j-1}(X-Y)Y^{i-j}, \quad i \geq 1,$$

and its translation in terms of tensor products

$$\boldsymbol{vec}(X^i - Y^i) = \left(\sum_{j=1}^{i} (Y^T)^{i-j} \otimes X^{j-1} \right) \boldsymbol{vec}(X - Y),$$

where $\boldsymbol{vec}(A)$ denotes the vector obtained by arranging the entries of the matrix A columnwise. The conditioning analysis can be completed by applying matrix norms, taking into account that the matrices R and U are stochastic or substochastic.

To solve the matrix equations (6) in [13] the authors apply the logarithmic reduction algorithm of Latouche and Ramaswami [8], in [1] the authors apply the matrix sign function approach of [2]. Here, we apply the algorithm of [4],[3] based on cyclic reduction by fully exploiting the inner structure of A, B and C.

We first recall the following result that reports formula (2.15) of [3].

Theorem 1. *The matrices A_j, B_j, C_j, $\widehat{A}_j$, $\widetilde{A}_j$, $j = 0, 1, \ldots$, inductively defined by*

$$\begin{aligned} A_{j+1} &= A_j - B_j A_j^{-1} C_j - C_j A_j^{-1} B_j, \\ B_{j+1} &= -B_j A_j^{-1} B_j, \\ C_{j+1} &= -C_j A_j^{-1} C_j, \\ \widehat{A}_{j+1} &= \widehat{A}_j - C_j A_j^{-1} B_j, \\ \widetilde{A}_{j+1} &= \widetilde{A}_j - B_j A_j^{-1} C_j, \end{aligned} \tag{9}$$

where $A_0 = \widehat{A}_0 = \widetilde{A}_0 = A$, $B_0 = B$, $C_0 = C$, *are such that* $\det A_{j+1}$, $\det \widetilde{A}_{j+1}$, $\det \widehat{A}_{j+1} \neq 0$. *Moreover, the matrices* R *and* U *solve the matrix equations*

$$\begin{aligned} &C + R\widehat{A}_j + R^{2^j+1} B_j = 0, \\ &U^{2^j+1} C_j + U\widetilde{A}_j + B = 0, \end{aligned} \tag{10}$$

and the matrices R^{2^j} *and* U^{2^j} *solve the matrix equations*

$$\begin{aligned} &C_j + R^{2^j} A_j + R^{2\cdot 2^j} B_j = 0, \\ &U^{2\cdot 2^j} C_j + U^{2^j} A_j + B_j = 0. \end{aligned} \tag{11}$$

Proof. The matrices U and R satisfy the equations

$$\left[R\ R^2 \ldots \right] \begin{bmatrix} A & C & & \\ B & A & C & \\ & \ddots & \ddots & \ddots \end{bmatrix} = \left[-C\ 0 \ldots \right],$$

$$\left[U\ U^2 \ldots \right] \begin{bmatrix} A & B & & \\ C & A & B & \\ & \ddots & \ddots & \ddots \end{bmatrix} = \left[-B\ 0 \ldots \right],$$

respectively. Applying j steps of cyclic reduction to the above systems yields

$$\left[R\ R^{2^j+1} \ldots \right] \begin{bmatrix} \widehat{A}_j & C_j & & \\ B_j & A_j & C_j & \\ & \ddots & \ddots & \ddots \end{bmatrix} = \left[-C\ 0 \ldots \right],$$

$$\left[U\ U^{2^j+1} \ldots \right] \begin{bmatrix} \widetilde{A}_j & B_j & & \\ C_j & A_j & B_j & \\ & \ddots & \ddots & \ddots \end{bmatrix} = \left[-B\ 0 \ldots \right],$$

(compare also lemma 3.1 of [4]). By equating the first block component in both sides of the above equations yields (10). Applying the same argument to the systems

$$\left[I\ R\ R^2 \ldots \right] \begin{bmatrix} A & C & & \\ B & A & C & \\ & \ddots & \ddots & \ddots \end{bmatrix} = \left[A + RB\ 0 \ldots \right],$$

$$\left[I\ U\ U^2 \ldots \right] \begin{bmatrix} A & B & & \\ C & A & B & \\ & \ddots & \ddots & \ddots \end{bmatrix} = \left[A + UC\ 0 \ldots \right],$$

and equating the second block entry at the j-th step of cyclic reduction yield (11). $\square$

From the properties of the matrix Q assumed in section 2 and from proposition 1 of [13], it follows that either $\rho(U) < 1$, or $\rho(R) < 1$, provided that the matrix $UC+A+RB$ is nonsingular. If $\rho(U) < 1$ the eigenvalues of U are the nm solutions of the equation $p(\lambda) = \det(B + \lambda A + \lambda^2 C) = 0$ in the open unit disk and the eigenvalues of R are the reciprocal of the remaining solutions (where we assume roots at the infinity if the polynomial $p(\lambda)$ has degree less than $2nm$), hence $\rho(R) = 1$. If otherwise $\rho(R) < 1$ the eigenvalues of R are the reciprocal of the nm solutions of the equation $p(\lambda) = 0$ of modulus greater than 1 and the eigenvalues of U are the solutions of $p(\lambda) = 0$ in the closed unit disk, hence $\rho(U) = 1$. This fact allows us to prove the following convergence property.

Theorem 2. *It holds*

$$\begin{aligned} R &= -C\widehat{A}_j^{-1} + O(r^{2^j}), \\ U &= -B\widetilde{A}_j^{-1} + O(r^{2^j}), \end{aligned}$$

where r is any number such that $\min\{\rho(U), \rho(R)\} < r < 1$.

Proof. Without loss of generality assume that $\rho(R) < 1$. From the first equation of (10) it follows that $||R + C\widehat{A}_j^{-1}|| = ||R^{2^j+1}B_j||$ for any matrix norm. Now observe that, from the definition of A_{j+1}, since its diagonal entries are nonpositive and $B_j \geq 0$, $C_j \geq 0$, it follows that $\mathrm{Diag}(A_{j+1}) \geq \mathrm{Diag}(A_j)$, where $\mathrm{Diag}(A)$ denotes the diagonal matrix having the same diagonal entries of A. This property, together with the relation $A_j\boldsymbol{e} + B_j\boldsymbol{e} + C_j\boldsymbol{e} = \boldsymbol{0}$, implies that $\boldsymbol{0} \leq B_j\boldsymbol{e} + C_j\boldsymbol{e} = -A_j\boldsymbol{e} \leq -\mathrm{Diag}(A_j)\boldsymbol{e} \leq -\mathrm{Diag}(A)\boldsymbol{e}$. That is B_j is bounded by a constant independent of j. From the latter property, since for any $\epsilon > 0$ there exists an operator matrix norm $||\cdot||$ such that $||R|| < \rho(R) + \epsilon$ (compare [7]), it follows the first part of the theorem. For the second part, we observe that $||U^{2^j}||$ is bounded, in fact $\rho(U^{2^j}) = 1$, so that it is sufficient to prove that $||C_j|| = O(r^{2^j})$ for a suitable operator norm. This follows since from (11) it holds $C_j = -R^{2^j}A_j - R^{2\cdot 2^j}B_j$ and A_j, B_j have bounded norm. □

If $UC + A + RB$ is singular then the conditions $\rho(U) < 1$ or $\rho(R) < 1$ may not be satisfied ([13], proposition 1), and the convergence stated in the above theorem is not guaranteed.

From theorem 2 the matrices R and U can be approximated by means of the matrices $-C\widehat{A}_j^{-1}$, $-B\widetilde{A}_j^{-1}$, where j is such that $\min\{||C_j||_\infty, ||B_j||_\infty\} < \epsilon$, for a given input error bound ϵ, where $||\cdot||_\infty$ denotes the infinity norm.

In order to compute the blocks A_j, B_j, C_j, $\widehat{A}_j$, $\widetilde{A}_j$ at each step j, the inner structure of A, B, C can be fully exploited. Indeed, we prove the following result:

Theorem 3. *At each step $j \geq 1$ of cyclic reduction it holds*

$$\begin{aligned} A_j &= A - (I_m \otimes \boldsymbol{S}^0)\sigma_j(\boldsymbol{\alpha} \otimes I_n) - (\boldsymbol{T}^0 \otimes I_n)\theta_j(I_m \otimes \boldsymbol{\beta}), \\ B_j &= -(I_m \otimes \boldsymbol{S}^0)\delta_j(I_m \otimes \boldsymbol{\beta}), \\ C_j &= -(\boldsymbol{T}^0 \otimes I_n)\gamma_j(\boldsymbol{\alpha} \otimes I_n), \\ \widehat{A}_j &= A - (\boldsymbol{T}^0 \otimes I_n)\theta_j(I_m \otimes \boldsymbol{\beta}), \\ \widetilde{A}_j &= A - (I_m \otimes \boldsymbol{S}^0)\sigma_j(\boldsymbol{\alpha} \otimes I_n), \end{aligned} \tag{12}$$

where δ_j, γ_j, σ_j, θ_j are $m \times m$, $n \times n$, $m \times n$, $n \times m$ matrices, respectively. The matrices δ_j, γ_j, σ_j, θ_j, for $j \geq 2$, are defined by the recursions

$$\begin{aligned}
\delta_{j+1} &= \delta_j \left(\delta_1 + \left[\delta_1 \; \sigma_1 \right] D_j V_j^{-1} \begin{bmatrix} \theta_1 \\ \delta_1 \end{bmatrix} \right) \delta_j, \\
\gamma_{j+1} &= \gamma_j \left(\gamma_1 + \left[\theta_1 \; \gamma_1 \right] D_j V_j^{-1} \begin{bmatrix} \gamma_1 \\ \sigma_1 \end{bmatrix} \right) \gamma_j, \\
\sigma_{j+1} &= \sigma_j + \delta_j \left(\sigma_1 + \left[\delta_1 \; \sigma_1 \right] D_j V_j^{-1} \begin{bmatrix} \gamma_1 \\ \sigma_1 \end{bmatrix} \right) \gamma_j, \\
\theta_{j+1} &= \theta_j + \gamma_j \left(\theta_1 + \left[\theta_1 \; \gamma_1 \right] D_j V_j^{-1} \begin{bmatrix} \theta_1 \\ \delta_1 \end{bmatrix} \right) \delta_j,
\end{aligned} \tag{13}$$

for $j = 1, 2, \ldots$, where

$$\begin{aligned}
\delta_1 &= (I_m \otimes \boldsymbol{\beta}) A^{-1} (I_m \otimes \boldsymbol{S}^0), \\
\gamma_1 &= (\boldsymbol{\alpha} \otimes I_n) A^{-1} (\boldsymbol{T}^0 \otimes I_n), \\
\sigma_1 &= (I_m \otimes \boldsymbol{\beta}) A^{-1} (\boldsymbol{T}^0 \otimes I_n), \\
\theta_1 &= (\boldsymbol{\alpha} \otimes I_n) A^{-1} (I_m \otimes \boldsymbol{S}^0),
\end{aligned} \tag{14}$$

and

$$D_j = \begin{bmatrix} \sigma_j & 0 \\ 0 & \theta_j \end{bmatrix}, \quad V_j = I_{m+n} - \begin{bmatrix} \theta_1 & \gamma_1 \\ \delta_1 & \sigma_1 \end{bmatrix} D_j.$$

Proof. Let us observe that, for the properties of Kronecker product we have $B = I_m \otimes \boldsymbol{S}^0 \boldsymbol{\beta} = (I_m \otimes \boldsymbol{S}^0)(I_m \otimes \boldsymbol{\beta})$ and $C = \boldsymbol{T}^0 \boldsymbol{\alpha} \otimes I_n = (\boldsymbol{T}^0 \otimes I_n)(\boldsymbol{\alpha} \otimes I_n)$. Due to this structure, by applying one step of cyclic reduction by means of (9) with $A_0 = A$, $B_0 = B$, $C_0 = C$ we arrive at the formulas (14) for representing A_1, B_1, C_1, $\widehat{A}_1$ and $\widetilde{A}_1$ in terms of δ_1, γ_1, σ_1 and θ_1. For the remaining part, we proceed by induction on j. Let us assume that (12) hold for j and prove them for $j+1$. We need to describe the structure of A_j^{-1}. Let us write $A_j = A - Z_j W_j$, where

$$Z_j = \left[I_m \otimes \boldsymbol{S}^0 \middle| \boldsymbol{T}^0 \otimes I_n \right] D_j, \quad W_j = \left[\frac{\boldsymbol{\alpha} \otimes I_n}{I_m \otimes \boldsymbol{\beta}} \right],$$

and observe that $V_j = I - W_j A^{-1} Z_j$. From the nonsingularity of A and A_j (compare theorem 1), since $\det(A_j) = \det(A)\det(V_j)$, it follows that $\det(V_j) \neq 0$. Whence, applying the Sherman Woodbury Morrison formula $A_j^{-1} = A^{-1} + A^{-1} Z_j (I - W_j A^{-1} Z_j)^{-1} W_j A^{-1}$, we obtain that

$$A_j^{-1} = A^{-1} + A^{-1} \left[I_m \otimes \boldsymbol{S}^0 \middle| \boldsymbol{T}^0 \otimes I_n \right] D_j V_j^{-1} \left[\frac{\boldsymbol{\alpha} \otimes I_n}{I_m \otimes \boldsymbol{\beta}} \right] A^{-1}. \tag{15}$$

By applying the equations (9), in the light of the inductive hypothesis, we may complete the proof. We show this for the matrix B_{j+1}, we leave the details on the remaining part to the reader. We have

$$B_{j+1} = -B_j A_j^{-1} B_j = -(I_m \otimes \boldsymbol{S}^0)\{\delta_j (I_m \otimes \boldsymbol{\beta}) A_j^{-1} (I_m \otimes \boldsymbol{S}^0) \delta_j\}(I_m \otimes \boldsymbol{\beta}).$$

Whence B_{j+1} has the structure (12) where $\delta_{j+1} = \delta_j (I_m \otimes \boldsymbol{\beta}) A_j^{-1} (I_m \otimes \boldsymbol{S}^0) \delta_j$. By replacing A_j^{-1} with (15) and by using the definitions of (14), we obtain the first equation of (13). ◻

In this way, the computation of the $mn \times mn$ blocks A_j, B_j, C_j, $\widetilde{A}_j$, $\widehat{A}_j$, is reduced to computing the blocks of lower dimensions δ_j, γ_j, σ_j, θ_j. Thus the computational cost of one step is reduced from $O(m^3n^3)$ to $O((m+n)^3)$ arithmetic operations.

It is interesting to point out that the matrices $\delta_1, \gamma_1, \sigma_1, \theta_1$ of (14) are the same matrices displayed in formulas (33)–(36) of [9], obtained by means of probabilistic arguments.

Also the matrices R and U are structured. Indeed, it is easy to verify that

$$\begin{aligned} \widehat{A}_j^{-1} &= A^{-1} + A^{-1}(\boldsymbol{T}^0 \otimes I_n)\theta_j(I_m - \sigma_1\theta_j)^{-1}(I_m \otimes \boldsymbol{\beta})A^{-1}, \\ \widetilde{A}_j^{-1} &= A^{-1} + A^{-1}(I_m \otimes \boldsymbol{S}^0)\sigma_j(I_n - \theta_1\sigma_j)^{-1}(\boldsymbol{\alpha} \otimes I_n)A^{-1}. \end{aligned}$$

Hence, R and U are approximated by the matrices $R_j = -C\widehat{A}_j^{-1}$, $U_j = -B\widetilde{A}_j^{-1}$, respectively, for an enough large value of j, such that

$$\begin{aligned} R_j &= -(\boldsymbol{T}^0 \otimes I_n)\left((\boldsymbol{\alpha} \otimes I_n) + \gamma_1\theta_j(I_m - \sigma_1\theta_j)^{-1}(I_m \otimes \boldsymbol{\beta})\right)A^{-1}, \\ U_j &= -(I_m \otimes \boldsymbol{S}^0)\left((I_m \otimes \boldsymbol{\beta}) + \delta_1\sigma_j(I_n - \theta_1\sigma_j)^{-1}(\boldsymbol{\alpha} \otimes I_n)\right)A^{-1}. \end{aligned} \tag{16}$$

Thus the matrices R and U have rank m and n, respectively, and

$$\begin{aligned} R_j^k &= -(\boldsymbol{T}^0 \otimes I_n)\nu_R^{k-1}\left((\boldsymbol{\alpha} \otimes I_n) + \gamma_1\theta_j(I_m - \sigma_1\theta_j)^{-1}(I_m \otimes \boldsymbol{\beta})\right)A^{-1}, \\ U_j^k &= -(I_m \otimes \boldsymbol{S}^0)\nu_U^{k-1}\left((I_m \otimes \boldsymbol{\beta}) + \delta_1\sigma_j(I_n - \theta_1\sigma_j)^{-1}(\boldsymbol{\alpha} \otimes I_n)\right)A^{-1}, \quad k \ge 1, \end{aligned}$$

where ν_R and ν_U are $n \times n$ and $m \times m$ matrices, respectively, given by

$$\begin{aligned} \nu_R &= -\left(\gamma_1 + \gamma_1\theta_j(I_m - \sigma_1\theta_j)^{-1}\sigma_1\right), \\ \nu_U &= -\left(\delta_1 + \delta_1\sigma_j(I_n - \theta_1\sigma_j)^{-1}\theta_1\right). \end{aligned} \tag{17}$$

From (16) it follows that the matrices R and U have a tensor structure as stated by the following

Theorem 4. *The solutions R and U of (6) are such that*

$$\begin{aligned} R &= -(\boldsymbol{T}^0 \otimes I_n)\left((\boldsymbol{\alpha} \otimes I_n) + \gamma_1\theta^*(I_m - \sigma_1\theta^*)^{-1}(I_m \otimes \boldsymbol{\beta})\right)A^{-1} \\ U &= -(I_m \otimes \boldsymbol{S}^0)\left((I_m \otimes \boldsymbol{\beta}) + \delta_1\sigma^*(I_n - \theta_1\sigma^*)^{-1}(\boldsymbol{\alpha} \otimes I_n)\right)A^{-1}. \end{aligned}$$

where $\theta^* = \lim_j \theta_j$, $\sigma^* = \lim_j \sigma_j$ and θ_j, σ_j are defined in (13).

Proof. The limits $\lim_j \theta_j$, $\lim_j \sigma_j$ exist since the sequences θ_j and σ_j are monotone and bounded. The expressions for R and U follow directly from (16). □

The structure of R and U can thus be exploited in the computation of the entries of the matrix in system (8), and in the computation of $\boldsymbol{y}_i$, by means of formula (7).

The resulting algorithm is resumed below:

Algorithm 1. *Computing R and U.*

INPUT: Positive integers n, m, the vectors $\boldsymbol{\alpha} \in \mathbb{R}^m$, $\boldsymbol{\beta} \in \mathbb{R}^n$, the matrices $T \in \mathbb{R}^{m\times m}$, $S \in \mathbb{R}^{n\times n}$, and an error bound $\epsilon > 0$.

Output: Approximations $\tilde{\theta} \in \mathbb{R}^{n\times m}$, $\tilde{\sigma} \in \mathbb{R}^{m\times n}$ to θ^*, σ^* providing approximations $\tilde{R}$, $\tilde{U}$ to the solutions R and U of the equation (6) by means of theorem 4.

Computation: 1. Compute $\boldsymbol{T}^0 = -T\boldsymbol{e}$, $\boldsymbol{S}^0 = -S\boldsymbol{e}$.
2. Compute $A = T \oplus S$ and A^{-1}.
3. Compute the matrices $\delta_1, \gamma_1, \sigma_1, \theta_1$ by means of (14).
4. For $j = 1, 2, \ldots$ compute the matrices $\delta_j, \gamma_j, \sigma_j, \theta_j$ by means of (13) until $\min\{||\gamma_j||_\infty, ||\delta_j||_\infty\} < \epsilon$.
5. Set $\tilde{\theta} = \theta_j, \tilde{\sigma} = \sigma_j$.

For the computation of the vector $\boldsymbol{y}$ we rely on the following:

Algorithm 2. *Computing the vector* $\boldsymbol{y}$.

Input: Positive integer n, m, N, the vectors $\boldsymbol{\alpha} \in \mathbb{R}^m$, $\boldsymbol{\beta} \in \mathbb{R}^n$, the matrices $T \in \mathbb{R}^{m\times m}$, $S \in \mathbb{R}^{n\times n}$, $\theta^* \in \mathbb{R}^{n\times m}, \sigma^* \in \mathbb{R}^{m\times n}$, the matrices $\delta_1, \gamma_1, \sigma_1, \theta_1$.

Output: Approximations of the components $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{N+1}$ of the steady state vector.

Computation: 1. Apply the repeated squaring technique for computing ν_R^{N-2} and ν_U^{N-2} in $\lceil \log_2(N-2) \rceil$ steps (compare (17)).
2. Solve the system (8).
3. By means of successive vector-matrix products, compute $\boldsymbol{u}R^i$ and $\boldsymbol{v}U^i$ for $i = 0, \ldots, N$ and compute $\boldsymbol{y}_i$ by means of (7) for $i = 1, \ldots, N+1$. For this computation exploit the tensor product structure of U and R given by theorem 4.
4. Compute $\boldsymbol{y}_0 = -\boldsymbol{y}_1 \widehat{B} T^{-1}$.
5. Normalize the vector $\boldsymbol{y}$.

The computational costs of algorithms 1 and 2 can easily be estimated. The cost of algorithm 1, dominated by the computation of a matrix inversion of size $m+n$ and a finite number of matrix products of size $m+n$, amounts to $O((m+n)^3 \log\log \epsilon^{-1})$ arithmetic operations (ops) due to the double exponential convergence of the scheme. Concerning algorithm 2, the repeated squaring stage costs $O((\max\{m,n\})^3 \log N)$ ops, the computation of (8) and its solution costs $O((m+n)^3)$ ops, the computation of $\boldsymbol{y}_i$ by means of (7) costs $O((\max\{m,n\})^2 N)$ ops. Thus, the overall computation of algorithm 2 costs $O((m+n)^3 \log N + (m+n)^2 N)$ ops.

The steady state vector could be directly computed by applying the folding algorithm of [16] or the cyclic reduction method as described in [3]. However this approach would require a number of steps in algorithm 1 proportional to $\log N$ and would involve a large memory storage needed for the back substitution stage. In our approach the number of steps, proportional to $\log\log \epsilon^{-1}$, does not depend on N, moreover we do not have to store the matrices obtained in the intermediate steps of the computation.

For PH/PH/1 queues, where the generator matrix is infinite and has the same structure of (1), the components of the steady state vector are given by $\boldsymbol{y}_i = \boldsymbol{w}R^i$, $i = 1, 2, \ldots$, where $\boldsymbol{w}$ is a suitable row vector and R is the solution of the first matrix equation in (6). Thus the approach proposed in this paper can be used for computing the matrix R and the steady state vector $\boldsymbol{y}$.

4 Numerical Results

We implemented our algorithm using Fortran 90 and compared it with block Gauss-Seidel iterative algorithm, for the computation of the vector $\boldsymbol{y}$. We also performed a comparison with the matrix sign function algorithm for the computation of the matrices R and U by using the software TELPACK that can be taken at the url `http://www.cstp.umkc.edu/org/tn/telpack/home.html`. The latter software does not exploit the inner structure of the problem; the specific structure of the matrix Q is exploited when using the Gauss-Seidel method.

The software implementing our algorithm can be downloaded at the url `http://www.dm.unipi.it/pages/meini/public_html`.

We ran our benchmarks on an alpha platform with the 53 bit standard IEEE arithmetic.

We considered the following test problems, where the arrivals and services are taken to be Erlang or mixed Erlang or hyperexponential. As is known, the convergence of the Gauss-Seidel procedure is very slow, especially for the hyperexponential case, and hence it is imperative to have a procedure with faster convergence in all cases.

Problem 1. $n = m = 5$, $\boldsymbol{\alpha} = \boldsymbol{\beta} = (1, 0, 0, 0, 0)$, $T = H(10/3)$, $S = H(5)$, where

$$H(r) = \begin{bmatrix} -r & r & 0 & 0 & 0 \\ 0 & -r & r & 0 & 0 \\ 0 & 0 & -r & r & 0 \\ 0 & 0 & 0 & -r & r \\ 0 & 0 & 0 & 0 & -r \end{bmatrix}.$$

Problem 2. $n = m = 5$, $\boldsymbol{\alpha} = (1, 0, 0, 0, 0)$, $\boldsymbol{\beta} = (0.7, 0, 0, 0.3, 0)$, $T = H(10/3)$,

$$S = \begin{bmatrix} -8 & 8 & 0 & 0 & 0 \\ 0 & -8 & 8 & 0 & 0 \\ 0 & 0 & -8 & 0 & 0 \\ 0 & 0 & 0 & -0.8135593 & 0.8135593 \\ 0 & 0 & 0 & 0 & -0.8135593 \end{bmatrix}.$$

Problem 3. $m = 5$, $n = 2$, $\boldsymbol{\beta} = (1, 0, 0, 0, 0)$, $\boldsymbol{\alpha} = (0.9, 0.1)$, $S = H(5)$,

$$T = \begin{bmatrix} -2 & 0 \\ 0 & -0.095238 \end{bmatrix}.$$

Problem 4. $m = 5$, $n = 2$, $\boldsymbol{\alpha} = (0.9, 0.1)$, $\boldsymbol{\beta} = (0.7, 0, 0, 0.3, 0)$,

$$S = \begin{bmatrix} -8 & 8 & 0 & 0 & 0 \\ 0 & -8 & 8 & 0 & 0 \\ 0 & 0 & -8 & 0 & 0 \\ 0 & 0 & 0 & -0.8135593 & 0.8135593 \\ 0 & 0 & 0 & 0 & -0.8135593 \end{bmatrix},$$

$$T = \begin{bmatrix} -2 & 0 \\ 0 & -0.095238 \end{bmatrix}.$$

	CR			GS		
N	Time	Iter.	Res.	Time	Iter	Res.
50	0.02	5	6.6e-16	0.71	287	2.9e-15
100	0.02	5	6.6e-16	1.85	430	3.5e-15
200	0.26	5	6.6e-16	5.84	693	3.3e-15
400	0.48	5	6.6e-16	20.19	1181	4.3e-15
800	0.85	5	6.6e-16	73.5	2115	3.7e-15
1600	1.61	5	6.6e-16	270.80	3891	4.1e-15

Table 1. Problem 1

	CR			GS		
N	Time	Iter.	Res.	Time	Iter	Res.
50	0.01	7	3.0e-16	1.45	662	2.4e-15
100	0.01	7	3.0e-16	3.62	850	2.3e-15
200	0.02	7	3.0e-16	9.89	1173	2.6e-15
400	0.41	7	3.0e-16	29.81	1752	2.5e-15
800	0.65	7	3.0e-16	96.8	2794	2.4e-15
1600	1.26	7	3.0e-16	328.76	4755	2.5e-15

Table 2. Problem 2

The CPU time in seconds, the number of iterations and the residual errors of the Gauss-Seidel (GS) algorithm and our algorithm (CR) in the computation of $\boldsymbol{y}$ for several values of N, are displayed in Tables 1 through 4.

Examining the entries in these four tables, it is evident that our algorithm performs far better than Gauss-Seidel algorithm in terms of CPU time and the number of iterations. In fact, the speed-up factor is greater than 20 for all the values of N and increases as N grows. For problem 4, with the value $N = 1600$, the speed-up factor is about 382. We observe that, despite the residual errors have the same order of magnitude, the numerical behavior of the two algorithms is different. For example, if we look at the components of the vector $\boldsymbol{y}$ provided by the algorithms, we see that the new algorithm generates an *absolute* error, that is about the machine precision, *uniformly* distributed in all the components of $\boldsymbol{y}$. This means that components less than the machine precision may be affected by large relative errors. Whereas the Gauss-Seidel iteration seems to provide components affected by a *relative* error roughly bounded by the machine precision.

If the user is interested in an approximation with a uniform relative error, it is more convenient to apply a suitable number of steps of the Gauss-Seidel iteration to the vector provided by our algorithm in order to refine the smallest components. This kind of refinement does not affect much the computational time of the whole procedure and provides a very accurate solution.

Table 5 reports the CPU time in seconds and the residual error of the computation of the matrices R and U performed by means of the cyclic reduction (CR) and the matrix sign function (MSF) approach implemented in the package TELPACK of [2].

	CR			GS		
N	Time	Iter.	Res.	Time	Iter	Res.
50	0.05	9	2.0e-16	1.02	1275	1.3e-16
100	0.06	9	2.0e-16	3.75	2416	1.5e-16
200	0.10	9	2.0e-16	10.17	3318	1.2e-16
400	0.15	9	2.0e-16	25.75	4201	1.3e-16
800	0.27	9	2.0e-16	73.86	5591	1.4e-16
1600	0.52	9	2.0e-16	199.02	8060	1.4e-16

Table 3. Problem 3

	CR			GS		
N	Time	Iter.	Res.	Time	Iter	Res.
50	0.06	9	3.2e-17	0.94	1180	1.6e-16
100	0.06	9	3.2e-17	3.77	2449	1.6e-16
200	0.10	9	3.2e-17	10.85	3551	1.8e-16
400	0.16	9	3.2e-17	27.81	4525	1.6e-16
800	0.30	9	3.2e-17	73.89	5976	1.6e-16
1600	0.59	9	3.2e-17	210.14	8491	1.7e-16

Table 4. Problem 4

	CR		MSF	
	Time	Res.	Time	Res.
Problem 1	0.06	1.1e-16	0.38	1.4e-14
Problem 2	0.05	1.1e-16	0.42	8.8e-14
Problem 3	0.04	2.2e-16	0.12	1.5e-14
Problem 4	0.04	1.1e-16	0.13	1.6e-14

Table 5. CPU time and residual for the computation of R and U.

Besides being faster than the matrix sign function approach, our method provides a better approximation with a smaller residual error.

Acknowledgments. We wish to thank the anonymous referees for their comments and suggestions that improved the presentation.

References

1. N. Akar and K. Sohraby. Finite and infinite QBD chains: a simple unifying algorithmic approach. In *Proc. IEEE Infocom 97*. 1997.
2. N. Akar and K. Sohraby. An invariant subspace approach in M/G/1 and G/M/1 type Markov chains. *Commun. Statist. Stochastic Models*, 13:381–416, 1997.
3. D. A. Bini and B. Meini. On cyclic reduction applied to a class of Toeplitz-like matrices arising in queueing problems. In *Computations with Markov Chains*, Ed. W. J. Stewart, Kluwer Academic Publisher, 21–38, 1995.
4. D. A. Bini and B. Meini. On the solution of a nonlinear matrix equation arising in queueing problems. *SIAM J. Matrix Anal. Appl.*, 17:906–926, 1996.
5. S. Chakravarthy and M. F. Neuts. Algorithms for the Design of Finite-Capacity Service Units, *Naval Research Logistics*, 36, 147-165, 1989.

6. S. D. Conte and Carl de Boor. *Numerical Analysis: An Algorithmic Approach*, McGraw-Hill, New York, 1972.
7. G. H. Golub and C. F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, 1989.
8. G. Latouche and V. Ramaswami. A logarithmic reduction algorithm for Quasi-Birth-Death processes. *J. Appl. Probability*, 30:650–674, 1993.
9. G. Latouche and V. Ramaswami. The PH/PH/1 queue at epochs of queue size change. *Queueing Syst. Appl.*, 25:97–114, 1997.
10. M. Marcus and H. Minc. *A Survey of Matrix Theory and Matrix Inequalities*, Allyn and Bacon, Boston, MA, 1964.
11. B. Meini. New convergence results on functional iteration techniques for the numerical solution of M/G/1 type Markov chains. *Numer. Math.*, 78:39–58, 1997.
12. V. Naoumov. Matrix-multiplicative approach to quasi-birth-and-death processes analysis. In *Matrix-analytic methods in stochastic models*, Eds. S.R. Chakravarthy and A.S. Alfa, Marcel Dekker, Inc., New York, 87–106, 1996.
13. V. Naoumov, U. R. Krieger, and D. Wagner. Analysis of a multi-server delay-loss system with a general markovian arrival process. In *Matrix-analytic methods in stochastic models*, Eds. S.R. Chakravarthy and A.S. Alfa, Marcel Dekker, Inc., New York, 43–66, 1996.
14. M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models: An algorithmic Approach*, The Johns Hopkins University Press, Baltimore, MD, 1981.
15. M. F. Neuts. *Structured Stochastic Matrices of M/G/1 Type and Their Applications*, Marcel Dekker Inc., New York, 1989.
16. J. Ye and San-Qi Li. Analysis of multi-media traffic queues with finite buffer and overload control–Part I: algorithm. In *Proc. IEEE Infocom 91*, Bal Harbour, FL, pages 1464–1474. 1991.