Journal of the Operations Research
Society of Japan
2007, Vol. 50, No. 1, 55-72

ALGORITHMIC COMPUTATION OF THE TRANSIENT QUEUE LENGTH DISTRIBUTION IN THE BMAP/D/c QUEUE

Kentaro Daikoku Hiroyuki Masuyama *Kyoto University*
Tetsuya Takine *Osaka University*
Yutaka Takahashi *Kyoto University*

(Received May 29, 2006; Revised November 13, 2006)

Abstract This paper proposes a numerically feasible algorithm for the transient queue length distribution in the BMAP/D/c queue. The proposed algorithm ensures the accuracy of the computational result and it is applicable not only to the stable case but also to the unstable case. This paper also discusses a numerical procedure to compute moments of the transient queue length distribution. Finally, some numerical examples are presented to demonstrate the applicability of the proposed algorithm.

Keywords: Queue, transient queue length distribution, batch Markovian arrival process (BMAP), BMAP/D/c queue, deterministic service, multi-server.

1. Introduction

This paper considers the transient queue length distribution in the BMAP/D/c queue. The queueing system has a buffer of infinite capacity and c ($c \geq 1$) identical servers. Customers arrive at the system according to a continuous-time batch Markovian arrival process (BMAP) [5] and their service times are all equal to some constant h ($h > 0$). Each server always serves a customer if any, and services are nonpreemptive.

In BMAP, arrivals are governed by a continuous-time, time-homogeneous Markov chain with finite state space $\mathcal{M} = \{1, 2, \dots, M\}$, which is called the underlying Markov chain hereafter. We assume that the underlying Markov chain has only one recurrent class. The underlying Markov chain stays in state i ($i \in \mathcal{M}$) for an exponential interval of time with mean μ_i^{-1} and then changes its state to state j ($j \in \mathcal{M}$) with probability $p_{i,j}$, where $\sum_{j\in\mathcal{M}} p_{i,j} = 1$ ($\forall i \in \mathcal{M}$). Given a state transition from state i to state j happens, n customer arrive in batch with probability $\zeta_{n,i,j}$, where

$$\sum_{n=0}^{\infty} \zeta_{n,i,j} = 1, \qquad \forall (i,j) \in \mathcal{M} \times \mathcal{M}.$$

Without loss of generality, we assume $\zeta_{0,i,i} = 0$ ($\forall i \in \mathcal{M}$) [5]. Let $\boldsymbol{C}$ and $\boldsymbol{D}_n$ ($n = 1, 2, \dots$) denote $M \times M$ matrices whose (i,j)th ($i, j \in \mathcal{M}$) elements $C_{i,j}$ and $D_{n,i,j}$ are given by

$$C_{i,j} = \begin{cases} -\mu_i, & \text{if } i = j, \\ \mu_i p_{i,j} \zeta_{0,i,j}, & \text{otherwise,} \end{cases}$$
$$D_{n,i,j} = \mu_i p_{i,j} \zeta_{n,i,j},$$

respectively. Thus BMAP can be characterized by the set of $M \times M$ matrices $(\boldsymbol{C}, \boldsymbol{D}_1, \boldsymbol{D}_2, \dots)$. We assume that

$$\boldsymbol{D} = \sum_{n=1}^{\infty} \boldsymbol{D}_n \neq \boldsymbol{O},$$

which excludes the trivial case where customers never arrive at the system.

Roughly speaking, customers arrive in the following way. When a state transition driven by $\boldsymbol{D}_n$ occurs, n customers arrive simultaneously. On the other hand, when a state transition driven by $\boldsymbol{C}$ occurs, no customers arrive (for details, see [5]). Note here that $\boldsymbol{C}+\boldsymbol{D}$ is the infinitesimal generator of the underlying Markov chain. Let $\boldsymbol{\pi}$ denote the stationary probability vector of the underlying Markov chain. $\boldsymbol{\pi}$ then satisfies $\boldsymbol{\pi}(\boldsymbol{C}+\boldsymbol{D}) = \boldsymbol{0}$ and $\boldsymbol{\pi}\boldsymbol{e} = 1$, where $\boldsymbol{e}$ denotes a column vector of ones with an appropriate dimension. Because the underlying Markov chain has only one recurrent class, $\boldsymbol{\pi}$ ($\boldsymbol{\pi} \geq \boldsymbol{0}$) is uniquely determined. Let λ and ρ denote the arrival rate and the traffic intensity, respectively.

$$\lambda = \boldsymbol{\pi} \sum_{n=1}^{\infty} n\boldsymbol{D}_n\boldsymbol{e}, \qquad \rho = \lambda h.$$

There are some papers on the algorithmic analysis of transient solutions of queues fed by BMAP. Lucantoni, Choudhury, and Whitt [7] considered the BMAP/G/1 queue. They applied the two-dimensional transform inversion algorithm [1] to the BMAP/G/1 queue and computed the transient workload and queue length distributions. However, the two-dimensional transform inversion algorithm is not always numerically stable and can not give the accuracy of the numerical results in advance. See also [6].

Le Ny and Sericola [4] considered the BMAP/PH/1 queue. They proposed an algorithm to compute the transient queue length distribution approximately with an accuracy specified in advance, using the uniformization technique [11]. Masuyama and Takine [8] considered spatially inhomogeneous bivariate Markov chains with the skip free to one direction property, which can represent the dynamics of a level-dependent BMAP/PH/c queue as a special case. The algorithm proposed in [8] is based on some ideas (including the uniformization) of achieving a target accuracy and reducing the computational cost. These two algorithms are numerically stable and yield numerical results with the target accuracy. However, they are not applicable to queues with deterministic service times, such as the BMAP/D/c queue.

In this paper, we propose a numerically reliable algorithm to compute the transient queue length distribution for the BMAP/D/c queue. A notable feature of our algorithm is the ability to specify the numerical accuracy in advance. Our algorithm originates from a time-evolution equation for the time-dependent queue length distribution, which is obtained by Crommelin's approach [2, **?**]. In the next section, we provide the time-evolution equation and some preliminary results. In section 3, we explain some key ideas of developing the algorithm to compute the approximation to the transient queue length distribution, and provide its complete description. We also present some theorems related to the feasibility of the algorithm. Section 4 discusses the computation of moments of the transient queue length distribution, and section 5 provides some numerical examples in order to demonstrate the applicability of the proposed algorithm. Finally section 6 contains concluding remarks.

2. Preliminary Results

Let $\boldsymbol{\pi}_n(t)$ ($t \geq 0$, $n = 0, 1, \ldots$) denote a $1 \times M$ vector whose jth ($j \in \mathcal{M}$) element represents $\Pr[L(t) = n,\ S(t) = j]$, where $L(t)$ and $S(t)$ ($t \geq 0$) denote the random variables that represent the number of customers in the system and the state of the underlying Markov chain, respectively, at time t. Because each arriving customer demands the service time of constant length h and he is served without interruption once starting his service, only those customers being served at time $t-h$ leave the system during time interval $(t-h, t]$, and all other customers do not leave the system during this time interval. This implies that customers in the system at time t consist of ones waiting already in the buffer at time $t-h$ and ones arriving during time interval $(t-h, t]$. Thus we have the following result.

Proposition 2.1 *For any $t \geq h$, the $\boldsymbol{\pi}_n(t)$ can be written in terms of the $\boldsymbol{\pi}_n(t-h)$:*

$$\boldsymbol{\pi}_n(t) = \sum_{l=0}^{c-1} \boldsymbol{\pi}_l(t-h)\boldsymbol{N}_n(h) + \sum_{l=c}^{n+c} \boldsymbol{\pi}_l(t-h)\boldsymbol{N}_{n+c-l}(h), \qquad n = 0, 1, \ldots, \tag{2.1}$$

where $\boldsymbol{N}_n(x)$ $(x \geq 0,\ n = 0, 1, \ldots)$ denotes an $M \times M$ matrix whose (i,j)th $(i, j \in \mathcal{M})$ element represents the conditional joint probability that n customers arrive during time interval $(\tau, \tau + x]$ $(\tau \geq 0)$ and $S(\tau + x) = j$ given $S(\tau) = i$.

Note here that $\sum_{n=0}^{\infty} \boldsymbol{N}_n(x)\boldsymbol{e} = \boldsymbol{e}$ and the $\boldsymbol{N}_n(x)$ satisfies

$$\sum_{n=0}^{\infty} z^n \boldsymbol{N}_n(x) = \exp\left[\left(\boldsymbol{C} + \sum_{n=1}^{\infty} z^n \boldsymbol{D}_n\right) x\right]. \tag{2.2}$$

We define $\boldsymbol{F}_{k,n}$ $(k, n = 0, 1, \ldots)$ as a nonnegative $M \times M$ matrix satisfying

$$\sum_{n=0}^{\infty} z^n \boldsymbol{F}_{k,n} = \left[\boldsymbol{I} + \theta^{-1}\left(\boldsymbol{C} + \sum_{n=1}^{\infty} z^n \boldsymbol{D}_n\right)\right]^k, \tag{2.3}$$

where $\theta = \max_{j \in \mathcal{M}} |C_{j,j}|$. From (2.2) and (2.3), we have

$$\begin{aligned}\sum_{n=0}^{\infty} z^n \boldsymbol{N}_n(x) &= \sum_{k=0}^{\infty} e^{-\theta x}\frac{(\theta x)^k}{k!}\left[\boldsymbol{I} + \theta^{-1}\left(\boldsymbol{C} + \sum_{n=1}^{\infty} z^n \boldsymbol{D}_n\right)\right]^k \\ &= \sum_{n=0}^{\infty} z^n \sum_{k=0}^{\infty} e^{-\theta x}\frac{(\theta x)^k}{k!}\boldsymbol{F}_{k,n},\end{aligned} \tag{2.4}$$

and

$$\sum_{n=0}^{\infty} z^n \boldsymbol{F}_{k,n} = \sum_{l=0}^{\infty} z^l \boldsymbol{F}_{1,l} \sum_{n=0}^{\infty} z^n \boldsymbol{F}_{k-1,n}, \qquad k = 1, 2, \ldots. \tag{2.5}$$

From (2.3), (2.4), and (2.5), we readily obtain the following result.

Proposition 2.2 *$\boldsymbol{N}_n(x)$ $(x \geq 0)$ is given by*

$$\boldsymbol{N}_n(x) = \sum_{k=0}^{\infty} e^{-\theta x}\frac{(\theta x)^k}{k!}\boldsymbol{F}_{k,n}, \qquad n = 0, 1, \ldots, \tag{2.6}$$

where the $\boldsymbol{F}_{k,n}$ is determined in the following way:

$$\boldsymbol{F}_{0,n} = \begin{cases} \boldsymbol{I}, & n = 0, \\ \boldsymbol{O}, & n = 1, 2, \ldots, \end{cases} \tag{2.7}$$

$$\boldsymbol{F}_{1,n} = \begin{cases} \boldsymbol{I} + \theta^{-1}\boldsymbol{C}, & n = 0, \\ \theta^{-1}\boldsymbol{D}_n, & n = 1, 2, \ldots, \end{cases} \tag{2.8}$$

and for $k = 2, 3, \ldots,$

$$\boldsymbol{F}_{k,n} = \sum_{l=0}^{n} \boldsymbol{F}_{1,l}\boldsymbol{F}_{k-1,n-l}, \qquad n = 0, 1, \ldots. \tag{2.9}$$

For $t \geq 0$, we define t_m $(m = 0, 1, \dots, \lfloor t/h \rfloor)$ as

$$t_m = \begin{cases} t - \lfloor t/h \rfloor h, & m = 0, \\ t_{m-1} + h = t_0 + mh, & m = 1, 2, \dots, \lfloor t/h \rfloor. \end{cases}$$

Note here that $t_{\lfloor t/h \rfloor} = t$. From Propositions 2.1 and 2.2, we can obtain the $\boldsymbol{\pi}_n(t)$ by computing the $\boldsymbol{\pi}_n(t_m)$ $(m = 1, 2, \dots, \lfloor t/h \rfloor)$ successively, when the $\boldsymbol{\pi}_n(t_0)$ is given. Thus we consider the $\boldsymbol{\pi}_n(t_0)$. We assume that $\Pr[L(0) = l_0] = 1$, where $l_0 \geq 0$. Let $l_0^{(S)}$ denote the number of customers being in service at time zero. Clearly, $l_0^{(S)} = \min(c, l_0)$. We also assume that the remaining service times of the ith $(i = 1, 2, \dots, l_0^{(S)})$ customers in service is equal to h_i, where $h_i \leq h$. Let $d(t_0)$ denote the number of those customers that are in service at time zero and leave the system by time t_0. We then have

$$d(t_0) = \sum_{i=1}^{l_0^{(S)}} \mathbb{I}(t_0 \geq h_i), \tag{2.10}$$

where $\mathbb{I}(\chi)$ denotes an indicator function of event χ. Further, let $A(t_0)$ denote the random variable that represents the number of customers arriving during time interval $(0, t_0]$. It is easy to see that $L(t_0) = l_0 - d(t_0) + A(t_0)$. Thus we obtain the following result, which is a straightforward extension of the result for the M/D/c queue [12].

Proposition 2.3 $\boldsymbol{\pi}_n(t_0)$ *is given by*

$$\boldsymbol{\pi}_n(t_0) = \begin{cases} \boldsymbol{\pi}_{\text{init}} \boldsymbol{N}_{n-l_0+d(t_0)}(t_0), & \textit{if } n \geq l_0 - d(t_0), \\ \mathbf{0}, & \textit{otherwise}, \end{cases} \tag{2.11}$$

where $\boldsymbol{\pi}_{\text{init}}$ *denotes a* $1 \times M$ *vector whose* j*-th* $(j \in \mathcal{M})$ *element represents* $\Pr[S(0) = j]$.

We now consider the computation of $\boldsymbol{\pi}_\nu(t)$'s $(\nu = 0, 1, \dots, n)$ for a fixed n $(n = 0, 1, \dots)$ and t $(t > 0)$. Proposition 2.1 shows that $\boldsymbol{\pi}_\nu(t)$'s $(\nu = 0, 1, \dots, n)$ are obtained in terms of $\boldsymbol{\pi}_\nu(t-h)$'s $(\nu = 0, 1, \dots, n+c)$, and the latter are obtained in terms of $\boldsymbol{\pi}_\nu(t-2h)$'s $(\nu = 0, 1, \dots, n+2c)$ and so on. Following this observation, for each $m = 1, 2, \dots, \lfloor t/h \rfloor$, we need $\boldsymbol{\pi}_\nu(t-mh)$'s $(\nu = 0, 1, \dots, n+mc)$ to obtain $\boldsymbol{\pi}_\nu(t)$'s $(\nu = 0, 1, \dots, n)$. Franx [12] briefly sketched this numerical procedure for the M/D/c queue. However, it cannot produce the *whole* transient queue length distribution $\boldsymbol{\pi}(t) = (\boldsymbol{\pi}_0(t), \boldsymbol{\pi}_1(t), \dots)$ with an overall accuracy specified in advance. First of all, in this procedure, we cannot determine n such that

$$\sum_{\nu=0}^{n} \boldsymbol{\pi}_\nu(t)\boldsymbol{e} \geq 1 - \varepsilon,$$

for target accuracy ε $(0 < \varepsilon \ll 1)$ before starting the computation of the $\boldsymbol{\pi}_n(t)$. Even if n could be determined beforehand, it might be so large as to require huge memory space, especially when the system is unstable.

On the other hand, for any given t $(t > 0)$, our algorithm successively generates the approximations $\widehat{\boldsymbol{\pi}}(t_m) = (\widehat{\boldsymbol{\pi}}_0(t_m), \widehat{\boldsymbol{\pi}}_1(t_m), \dots)$ to $\boldsymbol{\pi}(t_m) = (\boldsymbol{\pi}_0(t_m), \boldsymbol{\pi}_1(t_m), \dots)$ for $m = 0, 1, \dots, \lfloor t/h \rfloor$. Note here that the $\widehat{\boldsymbol{\pi}}(t_{\lfloor t/h \rfloor})$ is an approximation to $\boldsymbol{\pi}(t)$. The generated approximations $\widehat{\boldsymbol{\pi}}(t_m)$ $(m = 0, 1, \dots, \lfloor t/h \rfloor)$ have the following property. The details are described in the next section.

Property 2.1 *For any given* ε $(0 < \varepsilon \ll 1)$,

$$\mathbf{0} \le \widehat{\boldsymbol{\pi}}_n(t_m) \le \boldsymbol{\pi}_n(t_m), \qquad n = 0, 1, \dots, \tag{2.12}$$

$$\sum_{n=\underline{n}_m}^{\overline{n}_m} \widehat{\boldsymbol{\pi}}_n(t_m)\boldsymbol{e} \ge 1 - \frac{m+1}{\lfloor t/h \rfloor + 1}\varepsilon, \tag{2.13}$$

where $\underline{n}_m$ *and* $\overline{n}_m$ *are dynamically determined in the course of the execution of the algorithm, depending on* m, t, h, *and* ε *as well as the initial conditions.*

Remark 2.1 *Property 2.1 implies that the absolute error of* $\widehat{\boldsymbol{\pi}}_n(t_m)$ $(m = 0, 1, \dots, \lfloor t/h \rfloor)$ *is uniformly bounded, i.e.,*

$$\mathbf{0} \le \boldsymbol{\pi}_n(t_m) - \widehat{\boldsymbol{\pi}}_n(t_m) \le \frac{m+1}{\lfloor t/h \rfloor + 1}\varepsilon \boldsymbol{e}^T, \qquad 0 \le \boldsymbol{\pi}_n(t_m)\boldsymbol{e} - \widehat{\boldsymbol{\pi}}_n(t_m)\boldsymbol{e} \le \frac{m+1}{\lfloor t/h \rfloor + 1}\varepsilon,$$

for all $n = 0, 1, \dots$.

3. Algorithm for the Transient Queue Length Distribution

We first explain key ideas behind the proposed algorithm in subsection 3.1. We then give the complete description of the proposed algorithm in subsection 3.2 and show three theorems on its feasibility in subsection 3.3.

3.1. Key ideas behind the proposed algorithm

We introduce an approximation $\widehat{\boldsymbol{N}}_n(x)$ $(n = 0, 1, \dots)$ to the $\boldsymbol{N}_n(x)$, which has the following property.

Property 3.1 *The approximation* $\widehat{\boldsymbol{N}}_n(x)$ $(n = 0, 1, \dots)$ *satisfies*

$$\boldsymbol{O} \le \widehat{\boldsymbol{N}}_n(x) \le \boldsymbol{N}_n(x), \qquad n = 0, 1, \dots, \tag{3.1}$$

$$\sum_{n=\underline{n}(x)}^{\overline{n}(x)} \widehat{\boldsymbol{N}}_n(x)\boldsymbol{e} \ge (1-\delta)\boldsymbol{e}, \tag{3.2}$$

where $\delta = \varepsilon/(\lfloor t/h \rfloor + 1)$, *and where* $\underline{n}(x)$ *and* $\overline{n}(x)$ *are determined, depending on* x *and* δ *(see (3.10) and Remark 3.1).*

For a while, we assume the $\widehat{\boldsymbol{N}}_n(x)$ is available. With $\widehat{\boldsymbol{N}}_n(t_0)$, we define the approximation $\widehat{\boldsymbol{\pi}}(t_0)$ to $\boldsymbol{\pi}(t_0)$ in a way very similar to (2.11):

$$\widehat{\boldsymbol{\pi}}_n(t_0) = \begin{cases} \boldsymbol{\pi}_{\text{init}} \widehat{\boldsymbol{N}}_{n-l_0+d(t_0)}(t_0), & \text{if } \underline{n}_0 \le n \le \overline{n}_0, \\ \mathbf{0}, & \text{otherwise,} \end{cases} \tag{3.3}$$

where

$$\underline{n}_0 = \underline{n}(t_0) + l_0 - d(t_0), \qquad \overline{n}_0 = \overline{n}(t_0) + l_0 - d(t_0). \tag{3.4}$$

It follows from (2.11), (3.1), and (3.3) that

$$\mathbf{0} \le \widehat{\boldsymbol{\pi}}_n(t_0) \le \boldsymbol{\pi}_n(t_0), \qquad n = 0, 1, \dots.$$

Also, from (3.2) and (3.3), we have

$$\sum_{n=\underline{n}_0}^{\overline{n}_0} \widehat{\boldsymbol{\pi}}_n(t_0)\boldsymbol{e} = \boldsymbol{\pi}_{\text{init}} \sum_{n=\underline{n}_0}^{\overline{n}_0} \widehat{\boldsymbol{N}}_{n-l_0+d(t_0)}(t_0)\boldsymbol{e} = \boldsymbol{\pi}_{\text{init}} \sum_{n=\underline{n}(t_0)}^{\overline{n}(t_0)} \widehat{\boldsymbol{N}}_n(t_0)\boldsymbol{e} \ge 1 - \delta, \tag{3.5}$$

where we use $\boldsymbol{\pi}_{\rm init}\boldsymbol{e} = 1$. Therefore the $\widehat{\boldsymbol{\pi}}_n(t_0)$ satisfies Property 2.1.

For $m = 1, 2, \dots, \lfloor t/h \rfloor$, we determine $\widehat{\boldsymbol{\pi}}(t_m) = (\widehat{\boldsymbol{\pi}}_0(t_m), \widehat{\boldsymbol{\pi}}_1(t_m), \dots)$ by the following recursion, which is an analogue of (2.1):

$$\widehat{\boldsymbol{\pi}}_n(t_m) = \Psi_n \left[\sum_{l=\underline{n}_{m-1}}^{\min(c-1,\overline{n}_{m-1})} \widehat{\boldsymbol{\pi}}_l(t_{m-1}) \right] \widehat{\boldsymbol{N}}_n(h) + \sum_{l=\max(c,\underline{n}_{m-1})}^{\min(n+c,\overline{n}_{m-1})} \Psi_{n+c-l} \widehat{\boldsymbol{\pi}}_l(t_{m-1}) \widehat{\boldsymbol{N}}_{n+c-l}(h), \tag{3.6}$$

where

$$\Psi_n = \begin{cases} 1, & \text{if } \underline{n}(h) \le n \le \overline{n}(h), \\ 0, & \text{otherwise.} \end{cases}$$

To put it briefly, (3.6) implies that the recursion (2.1) is executed while discarding right- and left-tail probabilities of the $\boldsymbol{\pi}(t_m)$, which are negligible to guarantee a target accuracy. Clearly the $\widehat{\boldsymbol{\pi}}_n(t_m)$ generated by (3.6) satisfies (2.12). In subsection 3.3, we provide a theorem that ensures that the generated $\widehat{\boldsymbol{\pi}}_n(t_m)$ satisfies (2.13).

In the rest of this subsection, we discuss the approximation $\widehat{\boldsymbol{N}}_n(x)$ satisfying Property 3.1. Because the $\boldsymbol{N}_n(x)$ is given in terms of the doubly-infinite sequence $\boldsymbol{F}_{k,n}$ (see (2.6)), we have to truncate the $\boldsymbol{F}_{k,n}$ in computing the $\boldsymbol{N}_n(x)$. Thus, noting

$$\sum_{k=0}^{\infty} \sum_{n=0}^{\infty} e^{-\theta x} \frac{(\theta x)^k}{k!} \boldsymbol{F}_{k,n} \boldsymbol{e} = \sum_{n=0}^{\infty} \boldsymbol{N}_n(x) \boldsymbol{e} = \boldsymbol{e},$$

we introduce the approximation $\widehat{\boldsymbol{F}}_{k,n}$ $(k, n = 0, 1, \dots)$ to the $\boldsymbol{F}_{k,n}$, which satisfies

$$\boldsymbol{O} \le \widehat{\boldsymbol{F}}_{k,n} \le \boldsymbol{F}_{k,n}, \qquad n = 0, 1, \dots, \tag{3.7}$$

$$\sum_{k=\underline{k}}^{\overline{k}} \sum_{n=\underline{\nu}_k}^{\overline{\nu}_k} e^{-\theta x} \frac{(\theta x)^k}{k!} \widehat{\boldsymbol{F}}_{k,n} \boldsymbol{e} \ge (1-\delta) \boldsymbol{e}, \tag{3.8}$$

where $\underline{k}$, $\overline{k}$, $\underline{\nu}_k$, and $\overline{\nu}_k$ are dynamically determined in the course of the execution of the algorithm, depending on δ and x. We then define an approximation $\widehat{\boldsymbol{N}}_n(x)$ to $\boldsymbol{N}_n(x)$ as follows:

$$\widehat{\boldsymbol{N}}_n(x) = \begin{cases} \displaystyle\sum_{k=\underline{k}}^{\overline{k}} e^{-\theta x} \frac{(\theta x)^k}{k!} \widehat{\boldsymbol{F}}_{k,n}, & \text{if } \underline{n}(x) \le n \le \overline{n}(x), \\ \boldsymbol{O}, & \text{otherwise,} \end{cases} \tag{3.9}$$

where $\underline{n}(x)$ and $\overline{n}(x)$ are given by

$$\underline{n}(x) = \min\{\underline{\nu}_k \mid \underline{k} \le k \le \overline{k}\}, \qquad \overline{n}(x) = \max\{\overline{\nu}_k \mid \underline{k} \le k \le \overline{k}\}, \tag{3.10}$$

respectively. It follows from (3.8) and (3.9) that

$$\sum_{n=\underline{n}(x)}^{\overline{n}(x)} \widehat{\boldsymbol{N}}_n(x) \boldsymbol{e} = \sum_{n=\underline{n}(x)}^{\overline{n}(x)} \sum_{k=\underline{k}}^{\overline{k}} e^{-\theta x} \frac{(\theta x)^k}{k!} \widehat{\boldsymbol{F}}_{k,n} \boldsymbol{e} \ge \sum_{k=\underline{k}}^{\overline{k}} \sum_{n=\underline{\nu}_k}^{\overline{\nu}_k} e^{-\theta x} \frac{(\theta x)^k}{k!} \widehat{\boldsymbol{F}}_{k,n} \boldsymbol{e} \ge (1-\delta) \boldsymbol{e}.$$

(2.6), (3.7), and (3.9) imply that $\boldsymbol{O} \le \widehat{\boldsymbol{N}}_n(x) \le \boldsymbol{N}_n(x)$ for all $n = 0, 1, \dots$. Therefore the $\widehat{\boldsymbol{N}}_n(x)$ defined in (3.9) satisfies Property 3.1.

Remark 3.1 *Clearly, $\underline{k}$, $\overline{k}$, $\underline{\nu}_k$, and $\overline{\nu}_k$ depend on x and δ (see (3.8)). Therefore, so do $\underline{n}(x)$ and $\overline{n}(x)$ given in (3.10).*

Next, we outline how to construct the approximation $\widehat{\boldsymbol{F}}_{k,n}$ satisfying (3.7) and (3.8). The proposed algorithm first chooses $\underline{k}$ and $\overline{k}$ satisfying

$$\sum_{k=\underline{k}}^{\overline{k}} e^{-\theta x}\frac{(\theta x)^k}{k!}(1-\psi)^k \ge 1-\delta, \tag{3.11}$$

for a certain parameter ψ ($0<\psi<1$), depending on x and δ (see (3.18)). Note here that (3.11) is equivalent to

$$\sum_{k=\underline{k}}^{\overline{k}} \exp\left[-\theta x(1-\psi)\right]\frac{\{\theta x(1-\psi)\}^k}{k!} \ge 1-\sigma, \tag{3.12}$$

where

$$\sigma = 1-(1-\delta)e^{\theta x\psi}. \tag{3.13}$$

Because the left hand side of (3.12) is the sum of Poisson probabilities with mean $\theta x(1-\psi)$, we can find $\underline{k}$ and $\overline{k}$ efficiently, with the help of the property of the Poisson distribution. For example, see [3].

With $\underline{k}$ and $\overline{k}$, the algorithm generates $\widehat{\boldsymbol{F}}_{k,n}$ for $k=0,1,\dots,\overline{k}$, while determining $\underline{\nu}_k$ and $\overline{\nu}_k$ such that

$$\sum_{n=\underline{\nu}_k}^{\overline{\nu}_k} \widehat{\boldsymbol{F}}_{k,n}\boldsymbol{e} \ge (1-\psi)^k\boldsymbol{e}. \tag{3.14}$$

Note here that (3.11) and (3.14) yield

$$\sum_{k=\underline{k}}^{\overline{k}} e^{-\theta x}\frac{(\theta x)^k}{k!}\sum_{n=\underline{\nu}_k}^{\overline{\nu}_k}\widehat{\boldsymbol{F}}_{k,n}\boldsymbol{e} \ge \sum_{k=\underline{k}}^{\overline{k}} e^{-\theta x}\frac{(\theta x)^k}{k!}(1-\psi)^k\boldsymbol{e} \ge (1-\delta)\boldsymbol{e},$$

so that (3.8) is satisfied. The construction of $\widehat{\boldsymbol{F}}_{k,n}$ is as follows. According to (2.7), we define $\widehat{\boldsymbol{F}}_{0,n}$ as

$$\widehat{\boldsymbol{F}}_{0,n} = \begin{cases} \boldsymbol{I}, & n=0,\\ \boldsymbol{O}, & n=1,2,\dots, \end{cases}$$

which leads to $\underline{\nu}_0=\overline{\nu}_0=0$. Also, according to (2.8), we define $\widehat{\boldsymbol{F}}_{1,n}$ as

$$\widehat{\boldsymbol{F}}_{1,n} = \begin{cases} \boldsymbol{I}+\theta^{-1}\boldsymbol{C}, & n=0,\\ \theta^{-1}\boldsymbol{D}_n, & n=1,2,\dots. \end{cases} \tag{3.15}$$

Note here that

$$\sum_{n=0}^{\infty}\widehat{\boldsymbol{F}}_{1,n}\boldsymbol{e} = [\boldsymbol{I}+\theta^{-1}(\boldsymbol{C}+\boldsymbol{D})]\boldsymbol{e} = \boldsymbol{e}, \tag{3.16}$$

and therefore we can choose $\underline{\nu}_1$ and $\overline{\nu}_1$ satisfying (3.14) with $k=1$ while computing $\widehat{\boldsymbol{F}}_{1,n}$ for $n=0,1,\dots$. For more details, see subsection 3.2.

For $k = 2, 3, \ldots$, we determine $\widehat{\boldsymbol{F}}_{k,n}$ by

$$\widehat{\boldsymbol{F}}_{k,n} = \sum_{l=\max(\underline{\nu}_1,\, n-\overline{\nu}_{k-1})}^{\min(\overline{\nu}_1,\, n-\underline{\nu}_{k-1})} \widehat{\boldsymbol{F}}_{1,l}\widehat{\boldsymbol{F}}_{k-1,n-l}, \tag{3.17}$$

which is an analogue of (2.9). Suppose (3.14) holds for some $k := k-1 \geq 1$, i.e.,

$$\sum_{n=\underline{\nu}_{k-1}}^{\overline{\nu}_{k-1}} \widehat{\boldsymbol{F}}_{k-1,n}\boldsymbol{e} \geq (1-\psi)^{k-1}\boldsymbol{e}.$$

Summing both sides of (3.17) from $n = \underline{\nu}_{k-1} + \underline{\nu}_1$ to $\overline{\nu}_{k-1} + \overline{\nu}_1$ and post-multiplying them by $\boldsymbol{e}$ yield

$$\sum_{n=\underline{\nu}_{k-1}+\underline{\nu}_1}^{\overline{\nu}_{k-1}+\overline{\nu}_1} \widehat{\boldsymbol{F}}_{k,n}\boldsymbol{e} = \sum_{n=\underline{\nu}_{k-1}}^{\overline{\nu}_{k-1}} \widehat{\boldsymbol{F}}_{k-1,n} \sum_{l=\underline{\nu}_1}^{\overline{\nu}_1} \widehat{\boldsymbol{F}}_{1,l}\boldsymbol{e} \geq (1-\psi)^{k}\boldsymbol{e}.$$

Thus we can choose $\underline{\nu}_k$ $(\geq \underline{\nu}_{k-1} + \underline{\nu}_1)$ and $\overline{\nu}_k$ $(\leq \overline{\nu}_{k-1} + \overline{\nu}_1)$ satisfying (3.14). The details are described in subsection 3.2. Clearly this construction of the $\widehat{\boldsymbol{F}}_{k,n}$ ensures the inequality (3.7).

3.2. Description of the proposed algorithm

This subsection describes the proposed algorithm. We begin with the description of the subroutine **COMP-N** for the approximation $\widehat{\boldsymbol{N}}_n(x)$ to the $\boldsymbol{N}_n(x)$.

[**COMP-N**]

Input: $\boldsymbol{C}$, $\boldsymbol{D}_n$ $(n = 0, 1, \ldots)$, $x > 0$, $0 < \delta \ll 1$, $0 < \alpha < 1$.

Output: $\widehat{\boldsymbol{N}}_n(x)$ $(n = \underline{n}(x), \underline{n}(x) + 1, \ldots, \overline{n}(x))$.

Step I: Determine $\theta = \max_{j\in\mathcal{M}} |C_{j,j}|$ and set ψ to be

$$\psi = \alpha \min\left(1, -\frac{1}{\theta x}\log(1-\delta)\right). \tag{3.18}$$

Step II: Compute σ by (3.13) and find two nonnegative integers $\underline{k}$ and $\overline{k}$ satisfying (3.12).

Step III: Set $\underline{\nu}_0 = \overline{\nu}_0 = 0$ and $\widehat{\boldsymbol{F}}_{0,0} = \boldsymbol{I}$.

Step IV: For $n = 0, 1, \ldots$, compute $\widehat{\boldsymbol{F}}_{1,n}$ by (3.15) until minimum integers $\underline{\nu}_1$ and $\overline{\nu}_1$ satisfying

$$\max_{j\in\mathcal{M}} \left[\sum_{n=0}^{\underline{\nu}_1} \widehat{\boldsymbol{F}}_{1,n}\boldsymbol{e}\right]_j > \frac{\psi}{2}, \tag{3.19}$$

$$\sum_{n=\underline{\nu}_1}^{\overline{\nu}_1} \widehat{\boldsymbol{F}}_{1,n}\boldsymbol{e} \geq (1-\psi)\boldsymbol{e}, \tag{3.20}$$

are found. Further compute f_1 by

$$f_1 = \min_{j\in\mathcal{M}} \left[\sum_{n=\underline{\nu}_1}^{\overline{\nu}_1} \widehat{\boldsymbol{F}}_{1,n}\boldsymbol{e}\right]_j. \tag{3.21}$$

Step V: Repeat the following procedure for $k = 2, 3, \ldots, \overline{k}$. For $n = \underline{\nu}_{k-1} + \underline{\nu}_1, \underline{\nu}_{k-1} + \underline{\nu}_1 + 1, \ldots$, compute $\widehat{\boldsymbol{F}}_{k,n}$ by (3.17) until minimum integers $\underline{\nu}_k$ and $\overline{\nu}_k$ satisfying

$$\max_{j \in \mathcal{M}} \left[\sum_{n=\underline{\nu}_{k-1}+\underline{\nu}_1}^{\underline{\nu}_k} \widehat{\boldsymbol{F}}_{k,n} \boldsymbol{e} \right]_j > \frac{(1-\psi) f_{k-1} - (1-\psi)^k}{2}, \tag{3.22}$$

$$\sum_{n=\underline{\nu}_k}^{\overline{\nu}_k} \widehat{\boldsymbol{F}}_{k,n} \boldsymbol{e} \geq (1-\psi)^k \boldsymbol{e}, \tag{3.23}$$

are found. Further compute f_k by

$$f_k = \min_{j \in \mathcal{M}} \left[\sum_{n=\underline{\nu}_k}^{\overline{\nu}_k} \widehat{\boldsymbol{F}}_{k,n} \boldsymbol{e} \right]_j .$$

Step VI: Determine $\underline{n}(x)$ and $\overline{n}(x)$ by (3.10). For $n = \underline{n}(x), \underline{n}(x)+1, \ldots, \overline{n}(x)$, compute $\widehat{\boldsymbol{N}}_n(x)$ by (3.9).

Remark 3.2 *The input parameter α ($0 < \alpha < 1$) is irrelevant to the target accuracy δ of the approximation $\widehat{\boldsymbol{N}}_n(x)$. We observe the impact of α on the computational cost in section 5.*

Next, we describe the main routine of the proposed algorithm, which computes the transient queue length distribution.

[**Main routine**]

Input: $t > 0$, $0 < \varepsilon \ll 1$, $c \geq 1$, $h > 0$, $l_0 \geq 0$, $1 < h_i \leq h$ ($i = 1, 2, \ldots, l_0^{(S)} = \min(c, l_0)$), $\boldsymbol{\pi}_{\text{init}}$, $\boldsymbol{C}$, $\boldsymbol{D}_n$ ($n = 1, 2, \ldots$).

Output: $\widehat{\boldsymbol{\pi}}_n(t_m)$ ($n = \underline{n}_m, \underline{n}_m + 1, \ldots, \overline{n}_m$) for $m = 0, 1, \ldots, \lfloor t/h \rfloor$.

Step 1: Determine t_0 and δ by

$$t_0 = t - \lfloor t/h \rfloor h, \qquad \delta = \frac{\varepsilon}{\lfloor t/h \rfloor + 1},$$

respectively. Also determine $d(t_0)$ by (2.10).

Step 2: Compute $\widehat{\boldsymbol{N}}_n(t_0)$'s ($n = \underline{n}(t_0), \underline{n}(t_0)+1, \ldots, \overline{n}(t_0)$) by **COMP-N**, with $x := t_0$.

Step 3: Determine $\underline{n}_0$ and $\overline{n}_0$ by (3.4), and compute $\widehat{\boldsymbol{\pi}}_n(t_0)$ ($n = \underline{n}_0, \underline{n}_0 + 1, \ldots, \overline{n}_0$) by (3.3). Further compute ξ_0 by

$$\xi_0 = \sum_{n=\underline{n}_0}^{\overline{n}_0} \widehat{\boldsymbol{\pi}}_n(t_0) \boldsymbol{e}.$$

If $\lfloor t/h \rfloor = 0$, stop computing, and otherwise go to Step 4.

Step 4: Compute $\widehat{\boldsymbol{N}}_n(h)$'s ($n = \underline{n}(h), \underline{n}(h)+1, \ldots, \overline{n}(h)$) by **COMP-N** with $x := h$.

Step 5: Repeat the following procedure for $m = 1, 2, \ldots, \lfloor t/h \rfloor$. Set

$$\underline{n}'_m = \max(\underline{n}_{m-1} - c, 0) + \underline{n}(h), \tag{3.24}$$

$$\overline{n}'_m = \max(\overline{n}_{m-1} - c, 0) + \overline{n}(h). \tag{3.25}$$

For $n = \underline{n}'_m, \underline{n}'_m + 1, \ldots, \overline{n}'_m$, compute $\widehat{\boldsymbol{\pi}}_n(t_m)$ by (3.6) until minimum integers $\underline{n}_m$ and $\overline{n}_m$ satisfying

$$\underline{n}_m \geq \underline{n}'_m, \qquad \overline{n}_m \leq \overline{n}'_m, \tag{3.26}$$

$$\sum_{n=\underline{n}'_m}^{\underline{n}_m} \widehat{\boldsymbol{\pi}}_n(t_m)\boldsymbol{e} > \frac{(1-\delta)\xi_{m-1} - \{1-(m+1)\delta\}}{2}, \tag{3.27}$$

$$\sum_{n=\underline{n}_m}^{\overline{n}_m} \widehat{\boldsymbol{\pi}}_n(t_m)\boldsymbol{e} \geq 1-(m+1)\delta, \tag{3.28}$$

are found. Further compute ξ_m by

$$\xi_m = \sum_{n=\underline{n}_m}^{\overline{n}_m} \widehat{\boldsymbol{\pi}}_n(t_m)\boldsymbol{e}.$$

Remark 3.3 *Owing to $\delta = \varepsilon/(\lfloor t/h \rfloor + 1)$, (3.28) is equivalent to (2.13).*

3.3. Feasibility of the proposed algorithm

In this subsection, we show three theorems that ensure the feasibility of the proposed algorithm. The first two theorems are associated with the subroutine **COMP-N** and the last one with the main routine.

Theorem 3.1 *There exist two nonnegative integers $\underline{k}$ and $\overline{k}$ satisfying (3.12).*

Proof. This theorem holds if and only if $0 < \psi < 1$ and $0 < \sigma < 1$. Clearly ψ in (3.18) satisfies $0 < \psi < 1$ because $0 < \delta < 1$ and $0 < \alpha < 1$. Further, $\sigma < 1$ is clear from definition (3.13). Thus, we prove $\sigma > 0$. From (3.18), we have

$$\psi < -\frac{1}{\theta x}\log(1-\delta),$$

which leads to $(1-\delta)e^{\theta x \psi} < 1$. It then follows from (3.13) that $\sigma = 1-(1-\delta)e^{\theta x \psi} > 0$. □

Theorem 3.2 *There exist two nonnegative integers $\underline{\nu}_k$ and $\overline{\nu}_k$ ($\forall k = 2, 3, \ldots$) satisfying (3.22) and (3.23).*

Proof. We prove this theorem by induction. Note first that there exist $\underline{\nu}_1$ and $\overline{\nu}_1$ satisfying (3.19) and (3.20), which is due to (3.16) and the nonnegativity of the $\widehat{\boldsymbol{F}}_{1,n}$. From (3.17), we have

$$\begin{aligned}\sum_{n=2\underline{\nu}_1}^{2\overline{\nu}_1} \widehat{\boldsymbol{F}}_{2,n}\boldsymbol{e} &= \sum_{n=2\underline{\nu}_1}^{2\overline{\nu}_1} \sum_{l=\max(\underline{\nu}_1, n-\overline{\nu}_1)}^{\min(\overline{\nu}_1, n-\underline{\nu}_1)} \widehat{\boldsymbol{F}}_{1,l}\widehat{\boldsymbol{F}}_{1,n-l}\boldsymbol{e} = \sum_{l=\underline{\nu}_1}^{\overline{\nu}_1} \sum_{n=l+\underline{\nu}_1}^{l+\overline{\nu}_1} \widehat{\boldsymbol{F}}_{1,l}\widehat{\boldsymbol{F}}_{1,n-l}\boldsymbol{e} \\ &= \sum_{l=\underline{\nu}_1}^{\overline{\nu}_1} \widehat{\boldsymbol{F}}_{1,l} \sum_{n=\underline{\nu}_1}^{\overline{\nu}_1} \widehat{\boldsymbol{F}}_{1,n}\boldsymbol{e}.\end{aligned} \tag{3.29}$$

It then follows from (3.20), (3.21), and (3.29) that

$$\sum_{n=2\underline{\nu}_1}^{2\overline{\nu}_1} \widehat{\boldsymbol{F}}_{2,n}\boldsymbol{e} \geq (1-\psi)f_1\boldsymbol{e} \geq (1-\psi)^2\boldsymbol{e}. \tag{3.30}$$

The inequality (3.30) ensures that there exist two nonnegative integers $\underline{\nu}_2$ ($\geq 2\underline{\nu}_1$) and $\overline{\nu}_2$ ($\leq 2\overline{\nu}_1$), which satisfy (3.22) and (3.23) with $k = 2$.

Next, for some $k \geq 2$, we assume that there exist two nonnegative integers $\underline{\nu}_k$ and $\overline{\nu}_k$ satisfying (3.22) and (3.23). We then have

$$\sum_{n=\underline{\nu}_k+\underline{\nu}_1}^{\overline{\nu}_k+\overline{\nu}_1} \widehat{\boldsymbol{F}}_{k+1,n}\boldsymbol{e} \geq (1-\psi)f_k\boldsymbol{e} \geq (1-\psi)^{k+1}\boldsymbol{e},$$

which can be derived in a way very similar to (3.30). The above inequality ensures that there exist nonnegative integers $\underline{\nu}_{k+1}$ ($\geq \underline{\nu}_k + \underline{\nu}_1$) and $\overline{\nu}_{k+1}$ ($\leq \overline{\nu}_k + \overline{\nu}_1$), which satisfy (3.22) and (3.23) with $k := k + 1$. □

Theorem 3.3 *There exist two nonnegative integers $\underline{n}_m$ and $\overline{n}_m$ ($\forall m = 1, 2, \ldots$) satisfying (3.26), (3.27), and (3.28).*

The proof of Theorem 3.3 is given in Appendix A.

4. Recursion for Transient Moments

In this section, we provide the recursion for moments of the transient queue length distribution. We define $\boldsymbol{\pi}^*(t;z)$ as

$$\boldsymbol{\pi}^*(t;z) = \sum_{n=0}^{\infty} z^n \boldsymbol{\pi}_n(t).$$

We then define $\boldsymbol{\pi}^{(r)}(t)$ ($r = 0, 1, \ldots$) as

$$\boldsymbol{\pi}^{(r)}(t) = \lim_{z\to 1-} \frac{1}{r!}\frac{d^r}{dz^r}\boldsymbol{\pi}^*(t;z),$$

where $0! = 1$. Note here that the jth ($j \in \mathcal{M}$) element $\pi_j^{(r)}(t)$ of $\boldsymbol{\pi}^{(r)}(t)$ represents

$$\pi_j^{(r)}(t) = \mathrm{E}\left[\binom{L(t)}{r} \cdot \mathbb{I}(S(t) = j)\right].$$

Multiplying both sides of (2.1) with $t = t_m$ ($m = 1, 2, \ldots, \lfloor t/h \rfloor$) by z^n and summing them over $n = 0, 1, \ldots$, we have

$$z^c\boldsymbol{\pi}^*(t_m;z) = \sum_{k=0}^{c-1}(z^c - z^k)\boldsymbol{\pi}_k(t_{m-1})\boldsymbol{N}^*(h;z) + \boldsymbol{\pi}^*(t_{m-1};z)\boldsymbol{N}^*(h;z), \tag{4.1}$$

where $\boldsymbol{N}^*(x;z)$ ($x \geq 0$) is given by

$$\boldsymbol{N}^*(x;z) = \sum_{n=0}^{\infty} z^n \boldsymbol{N}_n(x) = \exp\left[\left(\boldsymbol{C} + \sum_{n=1}^{\infty} z^n \boldsymbol{D}_n\right)x\right].$$

Differentiating both sides of (4.1) r times with respect to z, multiplying them by $1/r!$, and letting $z \to 1-$, we obtain the following theorem.

Theorem 4.1 *For $m = 1, 2, \ldots, \lfloor t/h \rfloor$, $\boldsymbol{\pi}^{(r)}(t_m)$ $(r = 1, 2, \ldots)$ is given by*

$$\boldsymbol{\pi}^{(r)}(t_m) = -\sum_{l=0}^{r-1} \binom{c}{l} \boldsymbol{\pi}^{(l)}(t_m) + \sum_{l=0}^{r} \boldsymbol{\pi}^{(l)}(t_{m-1}) \boldsymbol{N}^{(r-l)}(h) + \sum_{l=0}^{r} \sum_{k=0}^{c-1} \left[\binom{c}{l} - \binom{k}{l} \right] \boldsymbol{\pi}_k(t_{m-1}) \boldsymbol{N}^{(r-l)}(h), \tag{4.2}$$

where $\boldsymbol{N}^{(0)}(x) = \exp[(\boldsymbol{C} + \boldsymbol{D})x]$ and

$$\boldsymbol{N}^{(r)}(x) = \frac{1}{r!} \frac{d^r}{dz^r} \boldsymbol{N}^*(x; z) \bigg|_{z=1-}, \qquad r = 1, 2, \ldots.$$

Also, multiplying both sides of (2.11) by z^n, and summing them over $n = 0, 1, \ldots$ yield

$$\boldsymbol{\pi}^*(z; t_0) = z^{l_0 - d(t_0)} \boldsymbol{\pi}_{\text{init}} \boldsymbol{N}^*(h; z),$$

from which we obtain the following theorem.

Theorem 4.2 $\boldsymbol{\pi}^{(r)}(t_0)$ *$(r = 0, 1, \ldots, 0 \le t_0 < h)$ is given by*

$$\boldsymbol{\pi}^{(r)}(t_0) = \sum_{l=0}^{r} \binom{l_0 - d(t_0)}{l} \boldsymbol{\pi}_{\text{init}} \boldsymbol{N}^{(r-l)}(t_0). \tag{4.3}$$

Next we consider the $\boldsymbol{N}^{(r)}(x)$ $(x \ge 0,\ r = 0, 1, \ldots)$. We define $\boldsymbol{F}_k^*(z)$ $(k = 0, 1, \ldots)$ and $\boldsymbol{D}^*(z)$ as

$$\boldsymbol{F}_k^*(z) = \sum_{n=0}^{\infty} z^n \boldsymbol{F}_{k,n}, \qquad \boldsymbol{D}^*(z) = \boldsymbol{C} + \sum_{n=1}^{\infty} z^n \boldsymbol{D}_n,$$

respectively. We also define $\boldsymbol{F}_k^{(r)}$ $(k, r = 0, 1, \ldots)$ and $\boldsymbol{D}^{(r)}$ $(r = 0, 1, \ldots)$ as

$$\boldsymbol{F}_k^{(r)} = \lim_{z \to 1-} \frac{1}{r!} \frac{d^r}{dz^r} \boldsymbol{F}_k^*(z), \qquad \boldsymbol{D}^{(r)} = \lim_{z \to 1-} \frac{1}{r!} \frac{d^r}{dz^r} \boldsymbol{D}^*(z),$$

respectively. It then follows from (2.5) and (2.6) that

$$\boldsymbol{F}_k^*(z) = \boldsymbol{F}_1^*(z) \boldsymbol{F}_{k-1}^*(z), \qquad \boldsymbol{N}^*(x; z) = \sum_{k=0}^{\infty} e^{-\theta x} \frac{(\theta x)^k}{k!} \boldsymbol{F}_k^*(z).$$

Theorem 4.3 $\boldsymbol{N}^{(r)}(x)$ *$(x \ge 0,\ r = 0, 1, \ldots)$ is given by*

$$\boldsymbol{N}^{(r)}(x) = \sum_{k=0}^{\infty} e^{-\theta x} \frac{(\theta x)^k}{k!} \boldsymbol{F}_k^{(r)},$$

where the $\boldsymbol{F}_k^{(r)}$ $(k, r = 0, 1, \ldots)$ is determined by the following recursion:

$$\boldsymbol{F}_0^{(r)} = \begin{cases} \boldsymbol{I}, & r = 0, \\ \boldsymbol{O}, & r = 1, 2, \ldots, \end{cases} \qquad \boldsymbol{F}_1^{(r)} = \begin{cases} \boldsymbol{I} + \theta^{-1}(\boldsymbol{C} + \boldsymbol{D}), & r = 0, \\ \theta^{-1} \boldsymbol{D}^{(r)}, & r = 1, 2, \ldots, \end{cases}$$

and for $k = 2, 3, \ldots,$

$$\boldsymbol{F}_k^{(r)} = \sum_{l=0}^{r} \boldsymbol{F}_1^{(l)} \boldsymbol{F}_{k-1}^{(r-l)}, \qquad r = 0, 1, \ldots.$$

Because the proof is straightforward, we omit it.

Theorems 4.1, 4.2, and 4.3 give the recursion for transient moments $\boldsymbol{\pi}^{(r)}(t)$. Note that there are several options on its implementation. Unfortunately, we have not yet constructed an algorithm that can guarantee the accuracy of the resulting approximation to $\boldsymbol{\pi}^{(r)}(t)$ in advance. This seems to be a difficult problem. We therefore close this subsection by showing a basic procedure for computing $\boldsymbol{\pi}^{(r)}(t)$.

[Procedure list of computing $\boldsymbol{\pi}^{(r)}(t)$]

Step A: Compute $\boldsymbol{N}^{(l)}(x)$ $(l = 0, 1, \dots, r)$ for $x = h$ and t_0 by the recursion in Theorem 4.3.

Step B: Compute $\boldsymbol{\pi}^{(l)}(t_0)$ $(l = 0, 1, \dots, r)$ by (4.3). If $\lfloor t/h \rfloor = 0$, stop computing, and otherwise compute $\boldsymbol{\pi}_k(t_0)$ $(k = 0, 1, \dots, c-1)$ by (2.11) and go to Step C.

Step C: Repeat the following procedure for $m = 1, 2, \dots, \lfloor t/h \rfloor$. Compute $\boldsymbol{\pi}^{(l)}(t_m)$ $(l = 0, 1, \dots, r)$ by (4.2). If $m \le \lfloor t/h \rfloor - 1$, compute $\boldsymbol{\pi}_k(t_m)$ $(k = 0, 1, \dots, c-1)$ by (2.1) with $t := t_m$.

5. Numerical Examples

This section presents some numerical examples. For this purpose, we assume the followings. We consider BMAP/D/2 queues, where the length of constant service times is chosen as the unit time, i.e., $h = 1$. The arrival process is characterized by

$$\boldsymbol{C} = \begin{pmatrix} -0.4 & 0.1 \\ 0.1 & -0.8 \end{pmatrix}, \qquad \boldsymbol{D}_n = \frac{1}{4r}\left(1 - \frac{1}{4r}\right)^{n-1} \begin{pmatrix} 0.3 & 0 \\ 0 & 0.7 \end{pmatrix}, \qquad n = 1, 2, \dots,$$

where $r \ge 0.25$. Thus the arrival rate of batches is equal to 0.5 and the traffic intensity per server is given by $\rho/c = 0.5 \times 4rh/2 = r$. We also assume that there exist two customers in service at time zero ($l_0^{(S)} = 2$) and their remaining service times h_1 and h_2 are equal to 0.25 and 0.75, respectively. Further, the initial state distribution $\boldsymbol{\pi}_{\text{init}}$ of the underlying Markov chain is given by (0.5, 0.5). In addition, we set the target accuracy $\varepsilon = 10^{-11}$.

Before discussing the time-dependent behavior of the queue length distribution, we examine the impact of α in (3.18) on the computational cost of the $\widehat{\boldsymbol{N}}_n(h)$. To do so, we consider the total number $n_F(\alpha)$ of $\widehat{\boldsymbol{F}}_{k,n}$'s generated in the course of the computation of the $\widehat{\boldsymbol{N}}_n(h)$, i.e.,

$$n_F(\alpha) = 1 + (\overline{\nu}_1 + 1) + \sum_{k=2}^{\overline{k}} \left[\overline{\nu}_k - (\underline{\nu}_{k-1} + \underline{\nu}_1) + 1\right].$$

Figure 1 plots $n_F(\alpha)$ for α $(0 < \alpha < 1)$ in three cases, $\rho/c = 0.7, 1$, and 2. We observe that α does not have a great impact on the computational cost. Because parameter α is irrelevant to the accuracy of the approximation $\widehat{\boldsymbol{\pi}}(t_m)$ (see subsection 3.1), we set $\alpha = 0.5$ in all numerical experiments.

We now observe the time-dependent behavior of the queue length distribution. We first consider the case of $\rho/c = r = 0.7 < 1$ (i.e., the positive-recurrent case) with the initial queue length $l_0 = 30$. Figure 2 plots the time-dependent queue length mass function $\boldsymbol{\pi}_n(t)\boldsymbol{e}$ for $t = 0$, 3, 10, 30, and 100, along with the stationary queue length distribution computed by M/G/1 paradigm [9, **?**]. On the other hand, Figure 3 plots the time-dependent mean queue length $\boldsymbol{\pi}^{(1)}(t)\boldsymbol{e}$ as a function of time t. From these two figures, we can observe how the $\boldsymbol{\pi}_n(t)\boldsymbol{e}$ and $\boldsymbol{\pi}^{(1)}(t)\boldsymbol{e}$ converge to the steady-state ones as t increases.

Next, we consider the case of $\rho/c = r = 1$ (i.e., the null-recurrent case) with $l_0 = 300$. Figure 4 plots the time-dependent queue length mass function $\boldsymbol{\pi}_n(t)\boldsymbol{e}$ for $t = 0$, 30, 100,

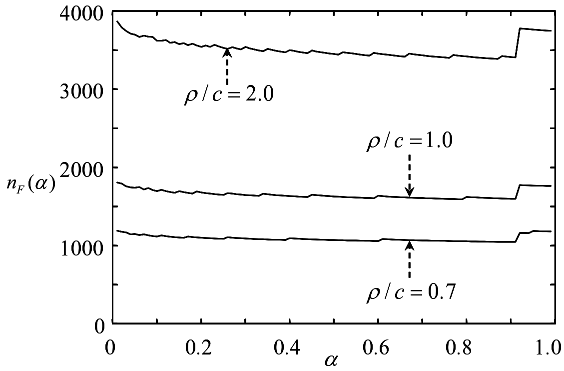

Figure 1: Total number $n_F(\alpha)$ of computed $\widehat{\boldsymbol{F}}_{k,n}$'s

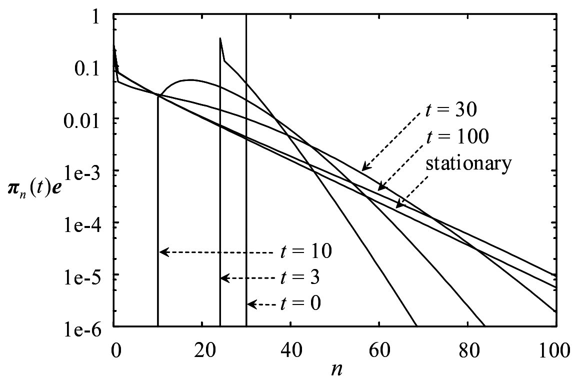

Figure 2: Queue length distribution ($\rho/c = 0.7$)

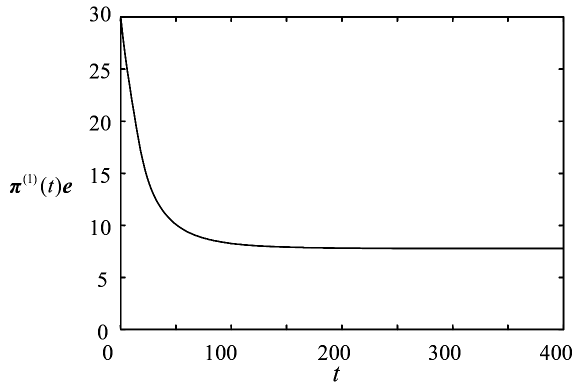

Figure 3: Mean queue length ($\rho/c = 0.7$)

300, 1000, and 3000. Because the queue length process is null recurrent, every $\boldsymbol{\pi}_n(t)\boldsymbol{e}$ ($n = 0, 1, \ldots$) goes to zero as $t \to \infty$. We observe that, as t increases, the time-dependent mass function is spread over a wider range of the queue length, while the mode of the queue length distribution remains around the initial queue length $l_0 = 300$. Figure 5 plots the time-dependent mean queue length $\boldsymbol{\pi}^{(1)}(t)\boldsymbol{e}$ as a function of time t. It is interesting that $\boldsymbol{\pi}^{(1)}(t)\boldsymbol{e}$ remains almost constant for a while and then turns to increase slowly with t. The reason of this phenomenon is that the drift of the queue length process is equal to zero and the state of the empty system works as a reflecting barrier.

Finally, we consider the case of $\rho/c = r = 2 > 1$ (i.e., the transient case), where we set $l_0 = 300$ again. Figure 6 plots the time-dependent queue length mass function $\boldsymbol{\pi}_n(t)\boldsymbol{e}$ for $t = 0$, 200, 400, 600, 800, and 1000. We observe that with time t, the mode of the queue length distribution moves toward the right at constant speed $\rho - c$ $(= 2)$. Figure 7 also plots the time-dependent mean queue length $\boldsymbol{\pi}^{(1)}(t)\boldsymbol{e}$ as a function of time t. We observe that $\boldsymbol{\pi}^{(1)}(t)\boldsymbol{e}$ increases linearly with t. Table 1 confirms these observations, where $\mathrm{mode}(t) = \arg\max_{n=0,1,\ldots}\{\boldsymbol{\pi}_n(t)\boldsymbol{e}\}$ and the mean queue length $\boldsymbol{\pi}^{(1)}(t)\boldsymbol{e}$ are shown. We observe that the mode of the queue length distribution increases at constant speed $\rho - c = 2$ and the slope of the mean queue length is equal to $\rho - c = 2$.

From the above observation, we conclude that the time-dependent behavior in transient case is totally different from that in the null-recurrent case, even though every $\boldsymbol{\pi}_n(t)\boldsymbol{e}$ ($n = 0, 1, \ldots$) in both cases goes to zero with time t.

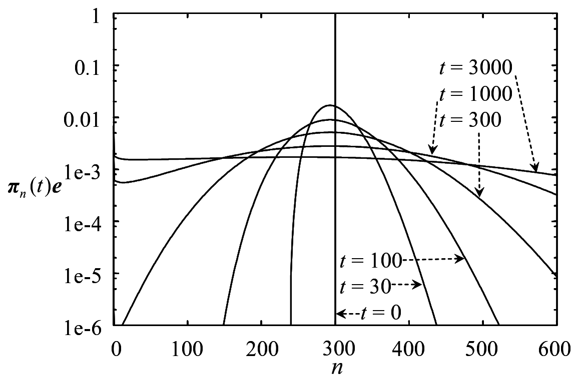

Figure 4: Queue length distribution ($\rho/c = 1$)

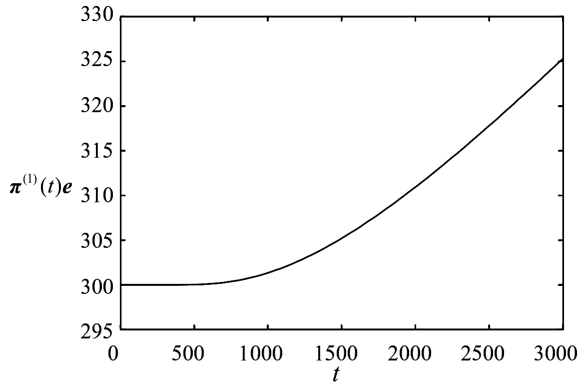

Figure 5: Mean queue length ($\rho/c = 1$)

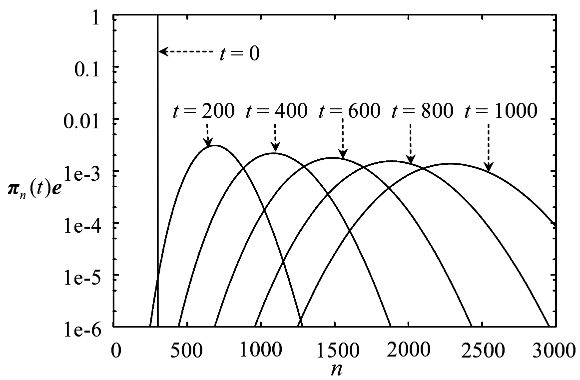

Figure 6: Queue length distribution ($\rho/c = 2$)

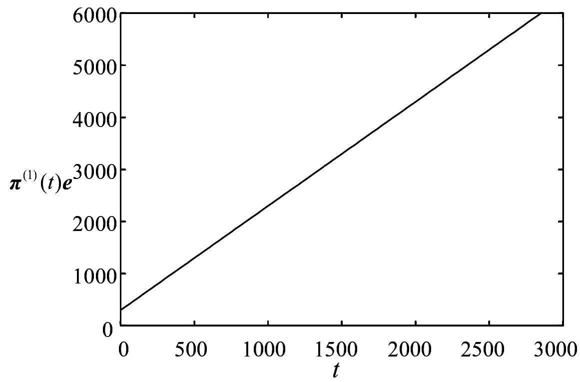

Figure 7: Mean queue length ($\rho/c = 2$)

Table 1: Mode and mean of the queue length distribution ($\rho/c = 2$)

t	0	200	400	600	800	1000
mode(t)	300	679	1079	1479	1879	2279
$\boldsymbol{\pi}^{(1)}(t)\boldsymbol{e}$	300.0	693.6	1093.6	1493.6	1893.6	2293.6

6. Concluding Remarks

We proposed the numerically feasible algorithm for the transient queue length distribution in the BMAP/D/c queue. The algorithm enables us to control the accuracy of the numerical result. We also presented numerical examples for positive-recurrent, null-recurrent, and transient cases.

We make a brief comment on the computational complexity of our algorithm. Through computational experiments, we observed the followings: (i) The number c of servers does not have a great impact on the total computational cost, (ii) the computational time increases almost quadratically with the target time t, and (iii) the computational time grows rapidly with the traffic intensity ρ/c when ρ/c is close to or over one.

A. Proof of Theorem 3.3

Note first that (see Step 3 and (3.5))

$$\xi_0 = \sum_{n=\underline{n}_0}^{\overline{n}_0} \widehat{\boldsymbol{\pi}}_n(t_0)\boldsymbol{e} \geq 1-\delta. \tag{A.1}$$

Summing both sides of (3.6) with $m=1$ from $n=\underline{n}'_1$ to $\overline{n}'_1$, we obtain

$$\begin{aligned}\sum_{n=\underline{n}'_1}^{\overline{n}'_1} \widehat{\boldsymbol{\pi}}_n(t_1) &= \sum_{l=\underline{n}_0}^{\min(c-1,\overline{n}_0)} \widehat{\boldsymbol{\pi}}_l(t_0) \sum_{n=\underline{n}'_1}^{\overline{n}'_1} \Psi_n \widehat{\boldsymbol{N}}_n(h) \\ &\quad + \sum_{l=\max(c,\,\underline{n}_0)}^{\overline{n}_0} \widehat{\boldsymbol{\pi}}_l(t_0) \sum_{n=\max(\underline{n}'_1,l-c)}^{\overline{n}'_1} \Psi_{n+c-l} \widehat{\boldsymbol{N}}_{n+c-l}(h). \end{aligned} \tag{A.2}$$

Note here that

$$\Psi_{n+c-l} = \begin{cases} 1, & \text{if } l-c+\underline{n}(h) \leq n \leq l-c+\overline{n}(h), \\ 0, & \text{otherwise.} \end{cases}$$

From (3.24) and (3.25), we obtain

$$\begin{aligned} l-c+\underline{n}(h) &\geq \max(\underline{n}_0-c,0)+\underline{n}(h) = \underline{n}'_1, \\ l-c+\overline{n}(h) &\leq \max(\overline{n}_0-c,0)+\overline{n}(h) = \overline{n}'_1, \end{aligned}$$

for l such that $\max(c,\,\underline{n}_0) \leq l \leq \overline{n}_0$. Thus we have

$$\sum_{l=\max(c,\,\underline{n}_0)}^{\overline{n}_0} \widehat{\boldsymbol{\pi}}_l(t_0) \sum_{n=\max(\underline{n}'_1,l-c)}^{\overline{n}'_1} \Psi_{n+c-l} \widehat{\boldsymbol{N}}_{n+c-l}(h) = \sum_{l=\max(c,\,\underline{n}_0)}^{\overline{n}_0} \widehat{\boldsymbol{\pi}}_l(t_0) \sum_{n=\underline{n}(h)}^{\overline{n}(h)} \widehat{\boldsymbol{N}}_n(h). \tag{A.3}$$

Substituting (A.3) into the second term on the right hand side of (A.2) yields

$$\sum_{n=\underline{n}'_1}^{\overline{n}'_1} \widehat{\boldsymbol{\pi}}_n(t_1) = \sum_{l=\underline{n}_0}^{\min(c-1,\overline{n}_0)} \widehat{\boldsymbol{\pi}}_l(t_0) \sum_{n=\underline{n}'_1}^{\overline{n}'_1} \Psi_n \widehat{\boldsymbol{N}}_n(h) + \sum_{l=\max(c,\,\underline{n}_0)}^{\overline{n}_0} \widehat{\boldsymbol{\pi}}_l(t_0) \sum_{n=\underline{n}(h)}^{\overline{n}(h)} \widehat{\boldsymbol{N}}_n(h). \tag{A.4}$$

If $\underline{n}_0 \geq c$, the first term on the right hand side of (A.4) is zero sum and hence (A.4) is reduced to

$$\sum_{n=\underline{n}'_1}^{\overline{n}'_1} \widehat{\boldsymbol{\pi}}_n(t_1) = \sum_{l=\underline{n}_0}^{\overline{n}_0} \widehat{\boldsymbol{\pi}}_l(t_0) \sum_{n=\underline{n}(h)}^{\overline{n}(h)} \widehat{\boldsymbol{N}}_n(h). \tag{A.5}$$

We now suppose $\underline{n}_0 \leq c-1$, so that $\underline{n}'_1 = \underline{n}(h)$ from (3.24). It then follows from (3.25) that $\overline{n}'_1 \geq \overline{n}(h)$ always holds. Thus we obtain

$$\sum_{n=\underline{n}'_1}^{\overline{n}'_1} \Psi_n \widehat{\boldsymbol{N}}_n(h) = \sum_{n=\underline{n}(h)}^{\overline{n}(h)} \widehat{\boldsymbol{N}}_n(h), \tag{A.6}$$

because $\Psi_n = 0$ unless $\underline{n}(h) \leq n \leq \overline{n}(h)$. It then follows from (A.4) and (A.6) that

$$\sum_{n=\underline{n}'_1}^{\overline{n}'_1} \widehat{\boldsymbol{\pi}}_n(t_1) = \sum_{l=\underline{n}_0}^{\min(c-1,\overline{n}_0)} \widehat{\boldsymbol{\pi}}_l(t_0) \sum_{n=\underline{n}(h)}^{\overline{n}(h)} \widehat{\boldsymbol{N}}_n(h) + \sum_{l=c}^{\overline{n}_0} \widehat{\boldsymbol{\pi}}_l(t_0) \sum_{n=\underline{n}(h)}^{\overline{n}(h)} \widehat{\boldsymbol{N}}_n(h). \tag{A.7}$$

If $\overline{n}_0 \le c-1$, the second term on the right hand side of (A.7) is zero sum and hence

$$\sum_{n=\underline{n}'_1}^{\overline{n}'_1} \widehat{\boldsymbol{\pi}}_n(t_1) = \sum_{l=\underline{n}_0}^{\min(c-1,\overline{n}_0)} \widehat{\boldsymbol{\pi}}_l(t_0) \sum_{n=\underline{n}(h)}^{\overline{n}(h)} \widehat{\boldsymbol{N}}_n(h) = \sum_{l=\underline{n}_0}^{\overline{n}_0} \widehat{\boldsymbol{\pi}}_l(t_0) \sum_{n=\underline{n}(h)}^{\overline{n}(h)} \widehat{\boldsymbol{N}}_n(h).$$

On the other hand, if $\overline{n}_0 \ge c$, we have

$$\begin{aligned}\sum_{n=\underline{n}'_1}^{\overline{n}'_1} \widehat{\boldsymbol{\pi}}_n(t_1) &= \sum_{l=\underline{n}_0}^{c-1} \widehat{\boldsymbol{\pi}}_l(t_0) \sum_{n=\underline{n}(h)}^{\overline{n}(h)} \widehat{\boldsymbol{N}}_n(h) + \sum_{l=c}^{\overline{n}_0} \widehat{\boldsymbol{\pi}}_l(t_0) \sum_{n=\underline{n}(h)}^{\overline{n}(h)} \widehat{\boldsymbol{N}}_n(h) \\ &= \sum_{l=\underline{n}_0}^{\overline{n}_0} \widehat{\boldsymbol{\pi}}_l(t_0) \sum_{n=\underline{n}(h)}^{\overline{n}(h)} \widehat{\boldsymbol{N}}_n(h).\end{aligned}$$

Therefore in both cases $\overline{n}_0 \le c-1$ and $\overline{n}_0 \ge c$, (A.7) is reduced to (A.5).

The above discussion shows that (A.5) holds for any couple of $\underline{n}_0$ and $\overline{n}_0$ $(0 \le \underline{n}_0 \le \overline{n}_0)$. Post-multiplying both sides of (A.5) by $\boldsymbol{e}$ and using (3.2) and (A.1) yield

$$\sum_{n=\underline{n}'_1}^{\overline{n}'_1} \widehat{\boldsymbol{\pi}}_n(t_1)\boldsymbol{e} \ge \xi_0(1-\delta) \ge (1-\delta)(1-\delta) > 1-2\delta. \tag{A.8}$$

The inequality (A.8) ensures that there exist two nonnegative integers $\underline{n}_1$ and $\overline{n}_1$ satisfying (3.26), (3.27), and (3.28) with $m=1$.

Next, for some $m \ge 1$, we assume that there exist two nonnegative integers $\underline{n}_m$ and $\overline{n}_m$ satisfying (3.26), (3.27), and (3.28). We can show

$$\sum_{n=\underline{n}'_{m+1}}^{\overline{n}'_{m+1}} \widehat{\boldsymbol{\pi}}_n(t_{m+1})\boldsymbol{e} \ge \xi_m(1-\delta) \ge \{1-(m+1)\delta\}(1-\delta) > 1-(m+2)\delta,$$

in a way very similar to (A.8). This inequality shows that there exist two nonnegative integers $\underline{n}_{m+1}$ and $\overline{n}_{m+1}$ satisfying (3.26), (3.27), and (3.28) with $m := m+1$. □

Acknowledgment

Research of the second author was supported in part by Grant-in-Aid for Young Scientists (B) of Japan Society for the Promotion of Science under Grant No. 18710131. Research of the third author was supported in part by Grant-in-Aid for Scientific Research (C) of Japan Society for the Promotion of Science under Grant No. 18560377. Research of the fourth author was supported in part by Grant-in-Aid for Scientific Research (B) of Japan Society for the Promotion of Science under Grant No. 18360181.

References

[1] G. L. Choudhury, D. M. Lucantoni and W. Whitt: Multidimensional transform inversion with applications to the transient M/G/1 queue. *The Annals of Applied Probability*, **4** (1994), 719–740.

[2] C. D. Crommelin: Delay probability formulae when the holding times are constant. *Post Office Electrical Engineer's Journal*, **25** (1932), 41–50.

[3] B. L. Fox and P. W. Glynn: Computing Poisson probabilities. *Communications of the ACM*, **31** (1988), 440–445.

[4] L.-M. Le Ny and B. Sericola: Transient analysis of the BMAP/PH/1 queue. *International Journal of Simulation: Systems, Science & Technology. Special Issue on Analytical & Stochastic Modeling Techniques*, **3** (2002), 4–14.

[5] D. M. Lucantoni: New results on the single server queue with a batch Markovian arrival process. *Stochastic Models*, **7** (1991), 1–46.

[6] D. M. Lucantoni: Further transient analysis of the BMAP/G/1 queue. *Stochastic Models*, **14** (1998), 461–478.

[7] D. M. Lucantoni, G. L. Choudhury and W. Whitt: The transient BMAP/G/1 queue. *Stochastic Models*, **10** (1994), 145–182.

[8] H. Masuyama and T. Takine: Algorithmic computation of the time-dependent solution of structured Markov chains and its application to queues. *Stochastic Models*, **21** (2005), 885-912.

[9] M. F. Neuts: The c-server queue with constant service times and a versatile Markovian arrival process. R. L. Disney, T. J. Ott, eds.: *Proceedings of The Conference on Applied Probability–Computer Science: The Interface* (Boca Raton, FL, 1982), Vol. I, 31–70.

[10] M. F. Neuts: *Structured Stochastic Matrices of M/G/1 Type and Their Applications* (Marcel Dekker, New York, 1989).

[11] H. C. Tijms: *Stochastic Models: An Algorithmic Approach* (John Wiley & Sons, Chichester, 1994).

[12] G. J. Franx: The transient M/D/c queueing system. available: http://www.math.vu.nl/~franx/ (2002).

Kentaro Daikoku
Graduate School of Informatics
Kyoto University
Yoshida-honmachi, Sakyo-ku
Kyoto 606-8501, JAPAN
E-mail: daikoku@sys.i.kyoto-u.ac.jp