Q-MAM: A Tool for Solving Infinite Queues using Matrix-Analytic Methods

J. F. Pérez, J. Van Velthoven and B. Van Houdt
Department of Mathematics and Computer Science
University of Antwerp
Middelheimlaan 1, B-2020 Antwerpen, Belgium
{juanfernando.perez,jeroen.vanvelthoven,benny.vanhoudt}@ua.ac.be

ABSTRACT

In this paper we propose a novel MATLAB tool, called Q-MAM, to compute queue length, waiting time and sojourn time distributions of various discrete and continuous time queuing systems with an underlying structured Markov chain/process. The underlying paradigms include M/G/1- and GI/M/1-type, quasi-birth-death and non-skip-free Markov chains (implemented by the SMCSolver tool), as well as Markov processes with a matrix exponential distribution. We consider various single server queueing systems with phase-type, matrix exponential, Markovian, rational and semi-Markovian arrival and service processes; queues with multiple customer types, where the service depends on the customer type and where consecutive customer types may be correlated; and queues with multiple servers for which the typical dimensionality problem can be avoided. Apart from implementing various classical and more advanced solution techniques, the tool also extends and improves some of the existing solution techniques in a number of cases.

1. INTRODUCTION

Over the last three decades, broad classes of frequently encountered queueing models have been analyzed by *matrix-analytic methods* [7, 24, 26, 27]. The embedded Markov chains and processes include quasi-birth-and-death (QBD), M/G/1- and GI/M/1-type and non-skip-free (NSF) Markov chains, as well as Markov processes with a matrix exponential distribution [28]. Matrix-analytic models include notions such as the Markovian, rational and semi-Markovian arrival process (MAP, RAP and SM), as well as the phase-type (PH) and matrix exponential (ME) distribution (both in discrete and continuous time). Considerable efforts have been put into the development of efficient and numerically stable methods for their analysis [7].

More recently, the SMCSolver tool [8, 9] implementing various state-of-the-art solution techniques (e.g., cyclic reduction, invariant subspace approach, the shift technique, Ramaswami reduction, etc.) for such Markov chains, was introduced. Also, both Fortran and MATLAB implementations are available online. In this paper, we propose a novel MATLAB tool, called Q-MAM, that builds on the SMCSolver tool to compute queue length, waiting time and sojourn time distributions of various classical and more advanced queueing systems both in discrete and continuous time. For instance, we consider various single server queueing systems with PH, ME, MAP, RAP and SM arrival and service processes. We analyze queues with multiple customer types, where the service depends on the customer type and where consecutive customer types may be correlated, meaning the arrival and service processes are PH[K], MMAP[K] or SM[K]. We further support queues with multiple servers for which the typical dimensionality problem can be avoided.

We also like to stress that apart from a few well known solution techniques (e.g., the queue length of a MAP/MAP/1 queue), this tool mostly implements various advanced and often lesser known techniques (e.g., Markov processes with a matrix exponential distribution). A lot of attention was also paid to optimizing the code in order to limit the computational resources when computing the performance measures. For instance, for the PH/PH/1 queue, we relied on the QBD introduced in [23] as its block size is only the *sum* of the number of phases of both distributions, as opposed to the product when utilizing the standard QBD. Furthermore, in a number of cases the functions also generalize some of the existing results in the literature and/or propose novel, more efficient numerical solution techniques (this is mostly the case whenever we encounter Sylvester matrix equations).

The paper is structured as follows. We start by giving a brief overview of the underlying structured Markov chains and processes involved. Afterward, we discuss the most significant issues of the building blocks of the queueing systems under consideration. In Section 4 we indicate how to use the MATLAB tool, while Section 5 lists some of the queues implemented by the Q-MAM tool and discusses some of the main issues of their implementation. This list is not exhaustive and a complete online listing will be maintained when the tool is made available online.

2. MATRIX-ANALYTIC METHODS OVERVIEW

When solving queues using matrix-analytic methods, one usually constructs a discrete or continuous time Markov chain/process having a highly structured transition or rate matrix, respectively. The queues implemented by our tool make use of the five (well-known) paradigms introduced be-

SMCTools 2008, October 20, 2008, Athens, GREECE
Copyright 2008 ICST ISBN # 978-963-9799-31-8.

low. For the first four, we limit ourselves to discussing the discrete time Markov chains only, their continuous time variants have a similar form. Furthermore, each of these continuous time Markov chains can be reduced easily to their discrete time counterparts via a uniformization argument. For more detailed information about these MCs we refer to [7, 14, 24, 26, 27, 28].

2.1 QBD Markov chains

QBD Markov chains are characterized by a transition matrix P of the form

$$P = \begin{bmatrix} B_0 & A_1 & & & 0 \\ B_{-1} & A_0 & A_1 & & \\ & A_{-1} & A_0 & A_1 & \\ & & A_{-1} & A_0 & \ddots \\ 0 & & & \ddots & \ddots \end{bmatrix},$$

where $A_{-1}, A_0, A_1 \in \mathbb{R}^{b\times b}$ and B_0, $B_{-1} \in \mathbb{R}^{b\times b}$, are nonnegative matrices such that $A_{-1} + A_0 + A_1$, $B_{-1} + A_0 + A_1$ and $B_0 + A_1$ are stochastic. We will refer to b as the block size and to the set of states $\{ib+1, \ldots, (i+1)b\}$ as level i of the QBD, for $i \geq 0$, e.g., the matrix A_x holds the transition probabilities from level i to $i + x$ for $i > 1$ and $x = -1, 0, 1$. The memory and time complexity to obtain the stationary vector of a QBD is $O(b^2)$ and $O(b^3)$ (per iteration), respectively. Any of the quadratically converging algorithms (like Logarithmic Reduction (LR), Cyclic Reduction (CR), Newton Iteration (NI), Invariant Subspace (IS)) typically requires less than 15 iterations. Moreover, the Schur decomposition variant of the IS algorithm only requires one iteration[1] (but is in general not faster than the other variants).

The form of the transitions toward and from level 0 may be relaxed, for instance, allowing a different number of states for level 0. This is convenient as level 0 often corresponds to an empty queue where there is no need to keep track of the progress of the service process.

2.2 M/G/1-type Markov chains

M/G/1-type Markov chains are defined by a transition matrix P of the form

$$P = \begin{bmatrix} B_0 & B_1 & B_2 & B_3 & \ldots \\ A_{-1} & A_0 & A_1 & A_2 & \ldots \\ & A_{-1} & A_0 & A_1 & \ddots \\ & & A_{-1} & A_0 & \ddots \\ 0 & & & \ddots & \ddots \end{bmatrix},$$

where A_i, for $i \geq -1$, and B_i, for $i \geq 0$, are nonnegative matrices in $\mathbb{R}^{b\times b}$ such that $\sum_{i=-1}^{+\infty} A_i$, $\sum_{i=0}^{+\infty} B_i$, are stochastic. The class of M/G/1-type Markov chains is clearly a generalization of the QBD paradigm. As with the QBDs, A_x holds the transition probabilities from level i to $i + x$ for $i \geq 1$, but with $x \geq -1$.

2.3 GI/M/1-type Markov chains

GI/M/1-type Markov chains are characterized by a transition matrix P of the form

$$P = \begin{bmatrix} B_0 & A_1 & & & 0 \\ B_{-1} & A_0 & A_1 & & \\ B_{-2} & A_{-1} & A_0 & A_1 & \\ B_{-3} & A_{-2} & A_{-1} & A_0 & \ddots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{bmatrix},$$

where A_{-i}, $i \geq -1$, and B_{-i}, $i \geq 0$ are nonnegative matrices in $\mathbb{R}^{b\times b}$ such that $\sum_{i=-1}^{n-1} A_{-i} + B_{-n}$ is stochastic for all $n \geq 0$. The class of GI/M/1-type Markov chains is also a generalization of the QBD paradigm. As with the QBDs, A_x holds the transition probabilities from level i to $i + x$ for $i \geq 0$, but with $-i + 1 \leq x \leq 1$. In the computation of the steady state probability vector of this type of MCs the matrix R plays a central role. This matrix is the minimal nonnegative solution to the non-linear matrix equation

$$R = \sum_{i\geq -1} R^{i+1} A_{-i}.$$

2.4 The case of Non-skip-free processes

Markov chains which are non-skip-free to *lower* levels are defined by the generalized block upper Hessenberg matrix

$$P = \begin{bmatrix} B_0 & B_1 & B_2 & B_3 & \ldots \\ B_{-1} & A_0 & A_1 & A_2 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ B_{-N+1} & A_{-N+2} & A_{-N+3} & A_{-N+4} & \ddots \\ A_{-N} & A_{-N+1} & A_{-N+2} & A_{-N+3} & \ddots \\ & A_{-N} & A_{-N+1} & A_{-N+2} & \ddots \\ & & A_{-N} & A_{-N+1} & \ddots \\ & & & A_{-N} & \ddots \\ 0 & & & & \ddots \end{bmatrix}. \qquad (1)$$

for $b \times b$ blocks A_i, $i \geq -N$ and B_i, $i \geq -N + 1$, where $N \geq 1$ is an integer. Markov chains which are non-skip-free to *upper* levels can be similarly defined in terms of a generalized block lower Hessenberg matrix. Even though the matrix P can be reblocked into blocks $\mathcal{B}_i$, $i \geq 0$ and $\mathcal{A}_i$, $i \geq -1$ of size bN as

$$P = \begin{bmatrix} \mathcal{B}_0 & \mathcal{B}_1 & \mathcal{B}_2 & \mathcal{B}_3 & \ldots \\ \mathcal{A}_{-1} & \mathcal{A}_0 & \mathcal{A}_1 & \mathcal{A}_2 & \ldots \\ & \mathcal{A}_{-1} & \mathcal{A}_0 & \mathcal{A}_1 & \ddots \\ & & \mathcal{A}_{-1} & \mathcal{A}_0 & \ddots \\ & & & \ddots & \ddots \end{bmatrix}. \qquad (2)$$

and solved like a standard M/G/1-type Markov chain, more efficient solutions that exploit the internal structure of these blocks have been developed. Markov chains of this type typically surface when bulk services are involved.

2.5 Markov processes with a matrix-exponential distribution

The class of bivariate Markov processes $\{(X_t, N_t)|t \geq 0\}$ with a matrix-exponential distribution are defined as follows:

[1] However, the Schur decomposition is itself computed with a number of iterations of the QR algorithm.

- The stochastic process $\{X_t | t \geq 0\}$ is skip-free to the right and takes values in $[0, \infty)$. It increases at a linear rate of 1, provided there are no downward transitions.

- N_t takes a finite number of values $\{1, \ldots, b\}$.

- If $(X_t, N_t) = (x, i)$, it can change its state to somewhere between $(x - u, j)$ and $(x - u + du, j)$ at a rate of $dA_{ij}(u)$, where $0 \leq u < x$ and $1 \leq i, j \leq b$.

- If $(X_t, N_t) = (x, i)$, it can change its state to $(0, j)$ at a rate of $B_{ij}(x)$, where $0 < x$ and $1 \leq i, j \leq b$.

- The matrices $A(x) = \int_0^x dA(u)$ and $B(x)$ satisfy the equation $\sum_{j=1}^{b}(A_{ij}(x) + B_{ij}(x)) = -d_i$, where $-d_i$ is the rate at which the next state change can take place from (x, i). Notice, this implies that the probability that X_t takes a downward jump of u from x is independent of the level x, given that a downward jump has taken place.

Under the suitable stability conditions, there exists a $b \times b$ matrix T such that

$$\pi(x) = \pi(0) \exp(Tx),$$

where $\pi(x)$ is a $1 \times b$ vector holding the steady state densities of being at level x. This type of Markov process may be regarded as a GI/M/1-type MC with a continuous level. As opposed to R, the matrix T is, in general, a solution to a non-linear integral equation

$$T = D + \int_0^\infty \exp(Tu) dA(u), \tag{3}$$

where D is a diagonal matrix holding the negative scalars d_i. An iterative algorithm to compute T is presented in [28] by repeatedly evaluating the right hand side of (3) starting with $T = D$ (the implementation of which is not included in the SMCSolver tool).

3. THE BUILDING BLOCKS: ARRIVAL AND SERVICE PROCESSES

3.1 Renewal processes

In this section we discuss three classes of distributions: Phase-Type (PH), Matrix Exponential (ME) and General distributions. These can be used to model the service process by assuming that the service times of consecutive customers are independent and identically distributed. They can also be used to characterize an arrival process by considering their corresponding renewal process.

3.1.1 The Phase-Type distribution and renewal process

A Phase-Type (PH) distribution is characterized by its matrix representation (α, S), where α is a (sub)stochastic $1 \times m$ vector and S is an $m \times m$ matrix. In the discrete time case, S is a nonnegative substochastic matrix and the PH cumulative distribution is given by $F(n) = 1 - \alpha S^n e$, where e is a column vector with all its entries equal to one. For the continuous time setting, the entries of S are such that $-S_{j,j} \geq \sum_{j' \neq j} S_{j,j'}$, with all the entries of S nonnegative, except for its diagonal entries. The density function of the continuous PH distribution is given by $\alpha \exp(Sx)s$, with $s = -Se$. In other words, a PH distribution may be regarded as the time until absorption in a discrete/continuous time MC characterized by S where the initial phase is determined by α. Phase-Type distributions are well suited for representing most of the types of services encountered in communication systems [20, 21]. There is also an active search for accurate matching algorithms to fit data to a PH distribution [4, 11, 29].

3.1.2 The Matrix Exponential distribution and renewal process

The class of Matrix Exponential (ME) distributions can be seen as a generalization of the continuous PH class. The representation of an ME distribution is also given by the tuple (α, S), where α is a $1 \times m$ vector, S is an $m \times m$ matrix and m is a positive integer referred to as the order of the representation [10]. In this case, however, the entries of (α, S) can be complex numbers without any restriction on their sign. It is actually possible to restrict these entries to be real numbers since this does not limit the ME class [5]. Additionally, the vector α can always be chosen such that $\alpha e = 1$ [6]. Defining $s = -Se$, the ME density function is given by $f(x) = \alpha \exp(Sx)s$, for $x > 0$, and with a point mass α_0 at 0 (here we assume that $\alpha_0 = 0$). For the tuple (α, S) to be a representation of an ME distribution, the matrix S must be invertible, and the density must be positive and integrate to one [5]. Given this conditions it is in general not easy to determine whether a given vector and matrix represent an ME distribution or not [1, 13]. For the functions involving ME distributions as input, we check the validity of the representation by evaluating the density function at several points along the x axis. These points are chosen from a range that comprises a total probability close to one. Naturally, we verify that the matrix S is nonsingular and that $\alpha e = 1$. This condition is particularly relevant for the solution of the QBD with RAP components as the one that arises when analyzing the ME/ME/1 queue [6].

From the definition above, it is clear that any PH distribution is an ME distribution but the PH class imposes some additional restrictions, as explained in the previous section. In particular, the PH and ME classes are equivalent when their order is two. This means that the ME class is broader than the PH class only for order greater than or equal to three. A specific feature of an ME density is that it can be zero for some $x > 0$, while a PH density can not show this behavior. The use of ME distributions in queuing theory arises from the possibility of extending many of the results for PH distributions to this broader class. Even though this has not been completely accomplished, several results have been obtained in that direction [6, 10, 25]. Additionally, the fact that the set of ME distributions imposes less restrictions than the PH class makes it attractive to fit data to this type of distribution. Some results about the characterization and fitting of ME distributions can be found in [12, 13, 19, 29, 30].

In the same way as for PH distributions, it is possible to define a discrete counterpart for ME distributions. These distributions are called matrix-geometric (MG) and are characterized by a row vector α, a square matrix P and a column vector s. The probability mass function is given by $p_n = \alpha P^n s$, for $n \in \{0, 1, 2, \ldots\}$ and $s = e - Pe$ [5].

3.1.3 GI - the General distribution and renewal process (with marked arrivals (GI[K]))

A GI renewal process is fully characterized by any cumulative distribution function $F(t)$, where $F(t)$ denotes the probability that the interarrival time is smaller than or equal to t. $F(t)$ can have a discrete, continuous or mixed nature. The GI[K] renewal process generates arrivals of K different types and is characterized by a set of K cumulative distribution functions $F_k(t)$ and probabilities p_k, for $k = 1, \ldots, K$, such that $\sum_{k=1}^{K} p_k = 1$. The probability p_k gives the probability that the next customer is of type k, while $F_k(t)$ represents the interarrival time, given that a type k arrival occurs.

3.2 Non-renewal processes

Even though some of the processes discussed below are called *arrival* processes, they can be used just as easily to model the service process of a queueing system, e.g., as in the MAP/MAP/1 queue.

3.2.1 MAP-the Markovian Arrival Process (with batches (BMAP) or markings (MMAP[K]))

In discrete time, the BMAP (or MMAP[K]) is characterized by a set of $K+1$ nonnegative substochastic $m \times m$ matrices $\{D_k | k = 0, \ldots, K\}$, with $\sum_{k=0}^{K} D_k$ stochastic (and irreducible). The (j, j')-th entry $(D_k)_{j,j'}$ of D_k holds the probability

$$(D_k)_{j,j'} = P[N_{t+1} = k, J_{t+1} = j' | J_t = j],$$

where J_t is the phase of the underlying MC, characterized by $\sum_{k=0}^{K} D_k$, at time t and N_t denotes the *number* of arrivals (for the BMAP) or the *type* of the arriving customer (for the MMAP[K]) occurring at time t (where type 0 corresponds to no arrival). MAPs are BMAPs (or MMAP[K]s) with $K = 1$.

In continuous time, the set of $m \times m$ matrices $\{D_k | k = 0, \ldots, K\}$ characterizing the BMAP (or MMAP[K]), are such that $-(D_0)_{j,j} = \sum_{j' \neq j} (D_0)_{j,j'} + \sum_{k=1}^{K} \sum_{j'=1}^{m} (D_k)_{j,j'}$, with all the entries of D_k, for $k = 0, \ldots, K$, nonnegative, except for the diagonal entries of D_0. The (j, j')-th entry $(D_k)_{j,j'}$ of D_k holds the rate at which the underlying continuous time MC, characterized by $\sum_{k=0}^{K} D_k$, changes its phase from j to j', while generating k arrivals for the BMAP or a type k arrival for the MMAP[K] (for $k \neq 0$ or $j \neq j'$).

Notice, whenever the MAP process is used to model the service process, its phase gets frozen during the time intervals where the server becomes idle.

3.2.2 RAP - the Rational Arrival Process

The RAP is a generalization of the continuous-time MAP in the same way as ME distributions generalize continuous-time PH distributions [2]. The RAP is characterized by two $m \times m$ matrices D_0 and D_1. For these matrices to represent a RAP the dominant eigenvalue of D_0 must have negative real part, the dominant eigenvalue of $D_0 + D_1$ must be zero and the row sum of $D_0 + D_1$ must also be zero. Additionally, the joint density function of the inter-event times must be non-negative; it is given by $f(x_1, x_2, \ldots, x_n) = \alpha \exp(D_0 x_1) D_1 \exp(D_0 x_2) D_1 \ldots \exp(D_0 x_n) D_1 e$, where α is a row vector specifying the initial conditions of the process. To verify if a tuple (D_0, D_1) is a representation of a RAP we first consider the direct conditions on the matrices. If the matrices comply with these conditions, we consider the event-stationary version of the RAP to test the non-negativity of the inter-arrival density function. In the event-stationary case, the vector α is chosen such that $\alpha(-D_0^{-1} D_1) = \alpha$, i.e., α is a stationary vector of the matrix $-D_0^{-1} D_1$. In the MAP case, the entries of this matrix are the phase transition probabilities at event epochs. Using this vector we evaluate the density $f(x_1)$ and the joint density $f(x_1, x_2)$ at several points such that the covered range accounts for a probability close to 1. As with ME distributions, some fitting methods have been developed to capture the moments of the inter-event distribution and the joint moments of successive events with RAPs [12, 29]. Additionally, some methods have been developed to find a MAP representation from a RAP in case it exists [29]. With this result it is possible to analyze a queue using the simpler MAP process rather than the RAP. Nevertheless, these procedures may require a considerable amount of time and in some cases a MAP representation may be impossible to get since this class is a proper subset of the RAP class. Using the implemented methods to solve the RAP/RAP/1 queue directly is useful to avoid the MAP translation and to analyze a broader set of queues.

3.2.3 SM - the semi-Markovian process (with marked arrivals (SM[K])

In discrete time the SM[K] *arrival* process is characterized by a set of nonnegative $m \times m$ matrices D_k^i, for $k = 1, \ldots, K$ and $i \geq 1$, such that $\sum_{i,k} D_k^i$ is stochastic. The (j, j')-th entry $(D_k^i)_{j,j'}$ of D_k^i holds the probability

$$(D_k^i)_{j,j'} = P[I_n = i, \tau_n = k, J_n = j' | J_{n-1} = j],$$

where J_n is the phase of the underlying MC, characterized by $\sum_{i,k} D_k^i$, after the n-th arrival, τ_n is the type of the n-th arrival and I_n denotes the interarrival time between customer $n-1$ and n. The SM processes correspond to the subclass of the SM[K] processes with $K = 1$.

When the discrete time SM[K] process is used to model the *service* times, these matrices have a slightly different meaning:

$$(D_k^i)_{j,j'} = P[S_n = i, J_{n+1} = j' | \tau_n = k, J_n = j],$$

where S_n is the service time of the n-th customer. Here, the D_k^i matrices have the requirement that $D = \sum_i D_k^i$ has to be identical and stochastic for every $k = 1, \ldots, K$. Thus, the type k of the customer does not influence the phase of the underlying process (but only influences the service time). Notice, for $K = 1$, there is no difference between the SM arrival or service process.

4. BASIC OPERATION OF THE MATLAB TOOL

The MATLAB implementation of the tool consists of a collection of MATLAB functions which can be executed from the command line (or called from within other functions or scripts). Each function takes at least one *required* parameter as input and may support several *optional* parameters. A call to a function, called fname, uses the following syntax:

$$\textit{output\_para} = \text{fname}(\textit{required\_para}, \textit{optional\_para})$$

If the function produces a single output parameter O1, one simply replaces *output_para* by O1. In case of multiple output parameters, O1, O2, ..., Ok, one sets *output_para* equal to [O1,O2,...,Ok]. If the user is only interested in the first

$l < k$ output variables, it suffices to shorten the list to Ol. This may potentially shorten the computation time as the values of the remaining $k - l$ output variables are typically computed last. As with any other MATLAB function, the *required_para* field holds the list of required parameters separated by commas: R1,R2,. . . ,Rr.

The *optional_para* field contains the list of optional parameters. When a function call uses no options, one simply passes the *required_para* field to the function. Otherwise, a pair of inputs must be given for each optional parameter used: the parameter name (pname) and the parameter value (pvalue). The name is always placed between single quotes, the value is placed between quotes if it holds a string (and not a numeric value or cell object). The parameter name has to be followed by its value, the order of the optional parameters on the other hand is arbitrary. Hence, in case of t optional parameters we have

optional_para = 'pname1',pvalue1,. . .,'pname$t$',pvalue$t$

The tool parses all optional parameters using the support function *Q_ParseOptPara.m*. As with any inbuilt MATLAB function, help can be requested for a function part of this tool by typing 'help fname' on the command line. The names of the functions start with the prefix Q_DT (resp. Q_CT) if the queueing system under analysis works in discrete (resp. continuous) time. The rest of the name includes the arrival process, the service process and the number of servers, separated by underscores. Some of these functions are included in Figure 1 as rectangular boxes. This figure shows the *default* dependency of the main Q-MAM functions on the SMCSolver routines. It also shows that when the underlying Markov process has a matrix exponential distribution, the Q_Sylvest function is the default option to solve Equation (3). The functions of the SMCSolver marked with an asterisk were modified to support the continuous-time version of the implemented algorithms. Additional dependency relations not shown in this figure are related to the validity check of the input arguments. Depending on its input parameters, each function calls different subroutines to analyze the validity of the parameters of the arrival and service processes. From Figure 1 it is clear that in most cases the underlying MC is solved using the CR algorithm as a default choice. We now show how this option can be modified and introduce other optional parameters supported by the Q-MAM functions.

4.1 General Optional Parameters

In this subsection we discuss three optional parameters that are shared by many of the supported functions.

4.1.1 Mode

Many functions compute the queue length, waiting and sojourn time distribution of queueing systems with an underlying QBD, M/G/1- or GI/M/1-type Markov chain. To obtain the steady state of these underlying Markov chains, one first needs to compute either the well-known R or G matrix. The SMCSolver tool supports various functions for doing so, e.g., for M/G/1-type MCs one can use functional iterations (FI), cyclic reduction (CR), the invariant subspace (IS) approach or the Ramaswami reduction (RR). By default, R or G are always computed using the CR algorithm. However, a different underlying algorithm can be selected using the optional parameter with pname='Mode'. For instance, to select the invariant subspace approach one sets pvalue='IS'. Whenever the underlying structured Markov process has a matrix exponential distribution, two modes of operation are supported: pvalue='Direct' and pvalue='Sylves'. Comments on the distinction between these two modes of operation are given in the discussion on the continuous time MMAP[K]/PH[K]/1 queue.

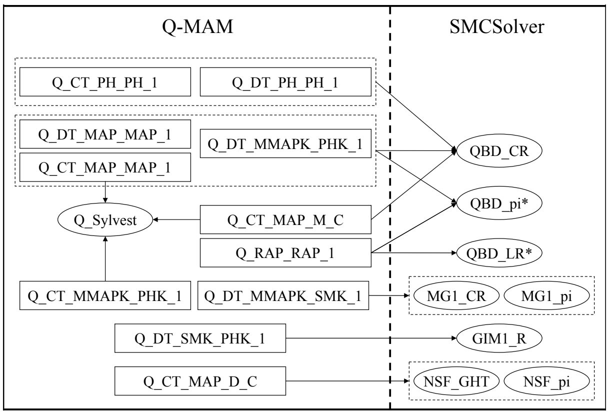

Figure 1: Dependency Graph of the Q-MAM Tool

4.1.2 Optfname

Our tool also offers the possibility to pass optional parameters to any of the underlying SMCSolver tool functions when called by a Q-MAM function. This is realized through the option 'Optfname', where fname is the name of the underlying function, e.g., fname=QBD_CR. The pvalue of this option is a cell object holding the option name and value for each of the underlying options. For instance, to activate the 'Verbose' option and set the 'Mode' equal to 'Basic', for the underlying function QBD_CR, one sets pname equal to 'OptQBD_CR' and pvalue equal to varname, where varname is a MATLAB cell object with varname{1}='Verbose', varname{2}=1, varname{3}='Mode', varname{4}='Basic'. A single call may contain multiple Optfname options and their order with respect to the other optional parameters is irrelevant.

4.1.3 Verbose

Similar to the SMCSolver tool, the 'Verbose' option will inform the user about the progress of the computation when set to 1. By default, its option value equals 0, indicating that no feedback is provided. Notice, activating the 'Verbose' option of a Q-MAM function does *not* automatically activate the 'Verbose' options of the underlying SMCSolver functions, if needed, this must be done via the Optfname option.

5. DESCRIPTION OF IMPLEMENTED QUEUEING SYSTEMS

In this section we present a listing of the various queueing systems that can be solved with our tool. We like to emphasize that this overview presents only a limited selection of the functionality offered by the tool. A complete listing with details for all the functions available will be maintained (and updated) online. Some of these functions also implement additional performance measures or variants of queues discussed in the existing literature. Each of these functions checks the validity of the input parameters and generates an error if the load of the queue exceeds one.

5.1 Single-type queueing systems

5.1.1 The MAP/MAP/1 queue

The queue length, waiting time and sojourn time distributions of a single server queue where both the arrival process and the service process are described by a continuous and discrete time MAP can be obtained using the functions Q_CT_MAP_MAP_1 and Q_DT_MAP_MAP_1, respectively. For such queues, consecutive interarrival times as well as consecutive service times can be correlated. Assume the MAP arrival process is characterized by the $m_a \times m_a$ matrices C_0 and C_1, whereas the MAP characterizing the service process is given by the matrices D_0 and D_1 of dimension m_s.

To retrieve the queue length distribution of the continuous time MAP/MAP/1 queue, we use the standard QBD MC with $A_{-1} = I \otimes D_1$, $A_0 = C_0 \oplus D_0$, $A_1 = C_1 \otimes I$, $B_{-1} = A_{-1}$ and $B_0 = C_0 \otimes I$. The Q_CT_MAP_MAP_1 function relies on the QBD_CR and QBD_pi functions with a block size $b = m_a m_s$ to solve this QBD. In an analogue manner, the Q_DT_MAP_MAP_1 function computes the queue length of the corresponding discrete time queue.

The waiting and sojourn time distributions of a continuous time MAP/MAP/1 queue are Phase-Type with b phases and their PH representations are obtained by constructing a Markov process $\{(X_t, J_t)|t \geq 0\}$ with a matrix exponential distribution. X_t represents the age of the customer in service at time t, while J_t stores the current phase of the service process and the phase of the arrival process at time $t - X_t$. This Markov process is found as a simple generalization of the technique presented in [16], for the special case of the continuous time MAP/PH/1 queue. For a brief discussion on the two modes of operations for solving (3), we refer to the continuous time MMAP[K]/PH[K]/1 queue.

The waiting and sojourn time distributions in the discrete time case are computed using the invariant vector of the standard QBD for the queue length, by first determining the probability that the system holds i customers immediately after an arrival, while the phase of the service process is j, which we denote as the j-th entry of the row vector π_i^a. Subsequently, the probability that the sojourn time equals n is given by $\sum_{i=1}^n \pi_i^a \mu_i^n$ (an analogue formula can be given for the waiting time), where the j-th entry of the column vector μ_i^n holds the probability that the i-th arrival of a MAP characterized by (D_0, D_1) arrives at time n given that the phase equals j at time 0. The vectors μ_i^n are easily obtained in a recursive manner, starting with $\mu_1^1 = D_1 e$.

5.1.2 The PH/PH/1 queue

In this queueing system the respective representation for the inter-arrival and the service distributions are (α, T) of order m_a and (β, S) of order m_s. To obtain the queue length distribution, one could use an approach similar to the one discussed in the previous section by constructing a QBD MC with a generator matrix in which the block size equals $m_a m_s$.

However, in [23] it was shown that by considering the process at epochs of queue size change only, one obtains a QBD MC where the size of the blocks in the generator matrix equals $m_a + m_s$. That is, if the most recent event was an arrival, one has to keep track of the phase of the service process only, while after a service completion it is sufficient to remember the phase of the arrival process. The computation of the order m_s PH distribution (wt_α, wt_T) of the waiting time distribution from this smaller QBD MC, is also discussed in this paper and implemented by the function Q_CT_PH_PH_1.

The function Q_DT_PH_PH_1 is also available and can be used to solve the discrete-time variant of this queue using a QBD with block size $b = m_a + m_s$ (note, [23] was limited to the continuous time case only), where the waiting time distribution is obtained via a series of convolutions after having found the queue length distribution and service phase at arrival time.

5.1.3 Combinations of MAP and PH

Our tool also includes functions for the MAP/PH/1 and the PH/MAP/1 queue. As these are special cases of the MAP/MAP/1 queue, analogue QBD MCs and Markov processes are used to solve them. These queues are also considered in both continuous and discrete time.

5.1.4 The RAP/RAP/1 queue

The function Q_RAP_RAP_1 computes the queue length distribution in a single server queue where the arrivals are described by a size m_a RAP(C_0,C_1) and the services by a size m_s RAP(D_0,D_1). First, the arrival and service processes are tested to assure that the matrices comply with the necessary conditions and the non-negativity of the density functions is tested for several points, as described in Section 3. The analysis is then carried out by specifying a Markov process $X(t)$ on the state space $\mathbb{N}_0 \times \mathcal{A}$, where the first component of $X(t)$ is the queue length and the second is the phase vector. $\mathcal{A}$ is the set made by the product of the compact convex sets on which the arrival and service RAPs are defined [6]. In a similar way as for the MAP/MAP/1 queue, the matrices $A_1 = C_1 \oplus I$ and $A_{-1} = I \oplus D_1$ describe the upward and downward jumps, respectively, when the level is greater than or equal to one. Additionally, the evolution of the phase vector between jumps is determined by a differential equation expressed in terms of $A_1 = C_0 \oplus D_0$. The downward jumps from level one to level zero are also ruled by the matrix A_{-1}, while the behavior of the phase vector between jumps in level zero depends on the matrix $B_0 = C_0 \otimes I$.

The well-known matrices G, R and U that arise in the analysis of traditional QBDs also appear in this more general setting. Furthermore, the nonlinear equation $A_{-1} + A_0 G + A_1 G^2 = 0$ holds [6, Corollary 6]. Thus, it is possible to determine the matrix G using algorithms that rely on this equation, e.g. Functional Iterations or LR. In the tool, the equation is solved using the LR algorithm with the function QBD_LR of the SMCSolver. This function was extended with the RAPComp option, which is set to 1 when calling QBD_LR. To compute the stationary queue length distribution we extended the function QBD_pi with a similar option to determine the expected state of the stationary phase vector at level zero π_0.

5.2 Multitype queueing systems

5.2.1 The discrete-time MMAP[K]/PH[K]/1 queue

This function solves a single server queue with customers of K different types, the service time of a type k customer has a size m_k PH distribution characterized by (α_k, S_k), for $k = 1, \ldots, K$. The arrival process is a size m MMAP[K], meaning consecutive interarrival times and customer types can be correlated. The function Q_DT_MMAPK_PHK_1 computes the overall and per type queue length, waiting and sojourn time distribution (whereas [31] was limited to waiting and sojourn times only).

Its implementation relies on [31], where a QBD MC is derived from a GI/M/1-type MC $\{(X_n, J_n)|n \geq 0\}$, where X_n represents the age of the customer in service and J_n stores the type of the customer in service, the current phase of the service process and the phase of the MMAPK when the customer in service arrived. The block size b of the resulting QBD MC is $b = m + m\sum_k m_k$. The function relies on the QBD_CR and QBD_pi functions and therefore requires $O(b^2)$ memory and $O(b^3)$ time per iteration.

To determine the queue length distribution of the type k customers, we start by determining the probability that the age of the customer in service equals i, its type is k, while the phase of the arrival process is j, which we denote as the j-th entry of the row vector π_i^k. Subsequently, the probability that the type k queue length equals n, with $n > 0$, is given by $\sum_{i \geq n-1} \pi_i^k \mu_{n-1}^i + \sum_{i \geq n}(\sum_{k' \neq k} \pi_i^{k'})\mu_n^i$ (an analogue formula can be given for the overall queue length), where the j-th entry of the column vector μ_n^i holds the probability that n type k arrivals of the MMAPK arrive in a length i interval that starts in phase j. The vectors μ_n^i are easily obtained in a recursive manner, starting with $\mu_0^0 = e$.

5.2.2 The continuous-time MMAP[K]/PH[K]/1 queue

The Q_CT_MMAPK_PHK_1 function computes the overall and per type queue length, sojourn and waiting time distribution for the MMAP[K]/PH[K]/1 queue in continuous time. Its implementation relies on [16], where a Markov process $\{(X_t, J_t)|t \geq 0\}$ is constructed, where X_t represents the age of the customer in service at time t and J_t stores the type of the customer in service, the current phase of the service process and the phase of the MMAP[K] when the customer in service arrived. The age X_t of a customer is a continuous variable and the resulting Markov process turns out to be a Markov process with a matrix exponential distribution. The block size b of the resulting Markov process is $b = m\sum_k m_k$.

In general, each iteration of the algorithm used to solve the non-linear integral matrix equation (3) requires a numerical integration. However, as demonstrated in [16] for this specific queueing system, numerical integration can be avoided by solving a (large) system of b^2 linear equations in b^2 unknowns, at a cost of $O(b^6)$ time and $O(b^4)$ memory per iteration. By recognizing that this linear system corresponds to a Sylvester matrix equation of the form $T_n X + X(D_0 \otimes I) = -I$, we have reduced this cost to $O(b^3)$ and $O(b^2)$, respectively, using the Q_Sylvest function. More specifically, during the n-th iteration we need to perform a Hessenberg decomposition of T_n, taking $14b^3/3$ time and solve a set of b Hessenberg systems, each requiring $O(b^2)$ time. The required Schur decomposition of $(D_0 \otimes I)$ is obtained from the decomposition of D_0 and needs to be performed just once.

The Q_CT_MMAPK_PHK_1 function offers two modes of operation: (i) the Direct mode which solves the (large) linear system during each iteration and (ii) the Sylves mode which relies on the Q_Sylvest function. The latter of these two modes of operation is the default mode. The (continuous, per type and overall) waiting time and sojourn time distribution of such a queueing system is of Phase-Type (with at most b phases), as such the return values of the function correspond to their PH-representations.

To retrieve the distribution of the number of type k customers present in the queue, we use a modified version of the formulas presented in [16, Section 5.2] (such that the customer in service is also taken into account). The key step consists in determining a set of $b \times b$ matrices $L_k(n)$ for $n \geq 0$, where $L_k(0)$ is a solution of $TL_k(0) + L_k(0)(D_{0,k} \otimes I) = -T$ and $TL_k(n) + L_k(n)(D_{0,k} \otimes I) = -L_k(n-1)(D_k \otimes I)$, for $n > 0$, where $D_{0,k} = D_0 + \sum_{i \neq k} D_i$. A direct-sum approach is proposed in [16] to compute these matrices. However, by recognizing that $L_k(n)$ is the solution of a Sylvester matrix equation of the form $AX + XB = C$, where A and B are identical for any n, it suffices to perform a single Hessenberg decomposition of $A = T$ and a Schur decomposition of $B = (D_{0,k} \otimes I)$, while $L_k(n)$ can be obtained from $L_k(n-1)$ by solving a set of b Hessenberg linear systems. An analogue approach can be used for the overall queue length by replacing $D_{0,k}$ by D_0 and D_k by $\sum_{i=1}^K D_i$.

5.2.3 The discrete-time SM[K]/PH[K]/1 queue

This function solves a single server queue with customers of K different types, the service time of a type k customer has a size m_k PH distribution characterized by (α_k, S_k), for $k = 1, \ldots, K$. The arrival process is a size m SM[K], meaning consecutive interarrival times and customer types can be correlated. The function Q_DT_SMK_PHK_1 computes the overall and per type sojourn time distribution and waiting time distribution. Optionally, one can also request a PH-representation of these distributions.

Its implementation relies on [18], where a GI/M/1-type MC $\{(X_n, J_n)|n \geq 0\}$ is constructed, with X_n the age of the customer in service and J_n holds the type of the customer in service, the current phase of the service process and the phase of the SM[K] arrival process when the customer in service was generated. The block size b of the resulting GI/M/1-type MC is $b = m\sum_k m_k$. As opposed to the MMAP[K]/PH[K]/1 queue, there is no efficient QBD reduction possible. The function relies on the GIM1_R function of the SMCSolver.

Notice, queueing systems like the SM/PH/1, GI/PH/1 and GI[K]/PH[K]/1 queue (in discrete time) are special cases of the SM[K]/PH[K]/1 queue and can therefore be solved using the Q_DT_SMK_PHK_1 function.

5.2.4 The discrete-time MMAP[K]/SM[K]/1 queue

This function solves a single server queue with customers of K different types, the service times are determined by a size m_s SM[K] process, thus, consecutive service times can be correlated and may depend on the customer type. The arrival process is a size m MMAP[K], meaning consecutive interarrival times and customer types can be correlated. The function Q_DT_MMAPK_SMK_1 computes the overall and per type sojourn time distribution and waiting time distribution.

Its implementation relies on [17], where an M/G/1-type MC $\{(X_n, J_n)|n \geq 0\}$ is constructed, where X_n represents the workload in the system and J_n stores the current phase of the service and arrival process. The block size b of the resulting M/G/1-type MC is $b = mm_s$. The function relies on the MG1_CR and MG1_pi functions of the SMCSolver.

Notice, queueing systems like the MAP/G/1, MAP/SM/1 and MMAP[K]/G[K]/1 queue (in discrete time) are special cases of the MMAP[K]/SM[K]/1 queue and can therefore be solved using the Q_DT_MMAPK_SMK_1 function.

5.3 Multiserver queues

Although the solution methods for most of the queueing systems considered above can be generalized to multiserver queueing systems, this typically requires the storage of the (current) phase of all the servers, making the block size b of the corresponding structured Markov chain impractical, unless the number of servers is (very) small. As we aim at developing efficient, state-of-the-art implementations of queueing systems with an underlying structured Markov chain, such implementations are *not* supported by our tool. The tools does support a number of multiserver queueing systems where such dimensionality problems can be avoided in an elegant manner.

5.3.1 The continuous time MAP/D/c queue

The analysis of the queue with c servers, deterministic service times and MAP arrivals is based on [26]. The queue is observed every s time units, where s is the length of the service time. Let X_n be the number of customers in the system and J_n be the phase of the arrival process. Then the Markov chain $\{(X_n, J_n)|n \geq 0\}$ has a non-skip-free structure to lower levels. The function Q_CT_MAP_D_C uses this Markov chain to determine the stationary distribution of the queue length at an arbitrary point in time. The parameters of the function are the $m \times m$ matrices $\{D_0, D_1\}$ of the MAP process, the length s of the service time and the number of servers c.

The transition matrix of the chain can be expressed in terms of the $m \times m$ matrices $P(k, s) = [P_{ij}(k, s)]$. The elements $P_{ij}(k, s)$ are equal to the probability of having k arrivals in an interval of length s and the phase of the arrival process at the end of this interval is j, given that it was in phase i at the beginning of the interval. These matrices are computed using the uniformization method [24]. The transition matrix has the structure of matrix P in Equation (1) with $A_{-c+i} = P(i, s)$, for $i \geq 0$. These matrices are used to compute the matrix G using the function NSF_GHT of the SMCSolver. Furthermore, the first c rows of the transition matrix are all equal to row $c + 1$: $[A_{-c}\ A_{-c+1}\ A_{-c+2} \ldots]$. This structure can be exploited by using the function NSF_pi to compute the joint stationary probability distribution of the queue length and the phase of the arrival process.

Additionally, the waiting time distribution can be computed for a specific set of points on $[0, \infty)$. This set is determined by the option NumSteps which specifies the number of equally separated points to evaluate in the range $[0, s)$. The default number of steps is equal to 1, meaning that the waiting time distribution is only evaluated at multiples of s. If this number is set equal to some integer K, it is necessary to evaluate the matrices $P(k, x)$ for $x = \{\frac{s}{K}, \frac{2s}{K}, \ldots, \frac{(K-1)s}{K}\}$ and $k \geq 0$.

5.3.2 The MAP/M/c queue

The function Q_CT_MAP_M_C computes the queue length and waiting time distributions of the queue with continuous-time MAP arrivals and c exponential servers. The MAP is characterized by the $m \times m$ matrices D_0 and D_1, and the rate of the exponential service times is μ. To determine the queue length a level-dependent QBD MC is set up. Let $X(t)$ be the number of customers in the system and $J(t)$ be the phase of the arrival process at time t, then $\{(X(t), J(t))|t \geq 0\}$ is MC with a QBD structure. Let the set of states $\{(k, j), 0 \leq j \leq m\}$ be the level k of the MC, in which k customers are in the system. From level c onwards the transitions are independent of the level with the QBD matrices given by $A_{-1} = c\mu I$, $A_0 = D_0 - c\mu I$ and $A_1 = D_1$. The QBD_CR function of the SMCSolver is used to compute the matrix R of this QBD which allows the determination of the components of the stationary probability vector for the levels greater than or equal to c. Let the stationary probability vector π of the MC be partitioned as $\pi = [\pi_0, \pi_1, \ldots]$, where the $1 \times m$ vector π_k corresponds to level k. The QBD structure implies that $\pi_{c-1+i} = \pi_{c-1} R^i$ for $i \geq 1$. To determine $[\pi_0, \pi_1, \ldots, \pi_{c-1}]$ we use the algorithm in [15] for finite level-dependent QBDs. The MC between levels 0 and $c - 1$ is a level-dependent QBD as the transitions from level i to level $i - 1$ are given by the matrix $i\mu I$. To analyze the first c levels as a finite QBD we consider the generator of the process restricted to these levels. This implies that the transient generator at level $c - 1$ is modified to be $D_0 - (c - 1)\mu I + RA_0$ and the only allowed transitions outside this level are to level $c-2$ given by the matrix $(c - 1)\mu I$. Once the vector $[\pi_0, \pi_1, \ldots, \pi_{c-1}]$ is computed it must be rescaled to comply with the condition $\sum_{k=0}^{c-2} \pi_k e + \pi_{c-1}(I - R)^{-1} e = 1$. The quantities $\pi_k e$, for $k \geq 0$, give the probability that there are k customer in the system at an arbitrary point in time.

For the computation of the waiting time distribution we rely on [3], where it is shown that the waiting time is Phase-Type with some representation (ρ, S). The matrix S is determined by first computing the T matrix of the Markov process $\{(X_t, N_t), t \geq 0\}$ with a matrix exponential distribution obtained by observing the queueing system only when all the servers are busy, where X_t is the age of the youngest customer being served and N_t denotes the phase of the arrival process when this youngest customer arrived. This T matrix is identical to the one of the Markov process used to compute the waiting time distribution in a MAP/M/1 queue with service rate $c\mu$. In [3, Section 6], the vector ρ and matrix S are computed from T using another vector α. Instead of using relation (7.2) in Section 7 to compute α, it is worth noting that α_i equals the probability that the phase of the arrival process equals i immediately after an arrival who found at least $c - 1$ busy servers upon arrival. Hence, we can easily recover α from the QBD constructed to compute the queue length distribution as $\alpha = \pi_{c-1}(I - R)^{-1} D_1 / (\pi_{c-1}(I - R)^{-1} D_1 e)$. As a final step, the vector ρ needs to be normalized correctly by the probability that a customer has a nonzero waiting time. This probability is again more easily obtained from the queue length QBD using $\pi_c(I-R)^{-1} D_1 e / (\sum_{i=0}^{\infty} \pi_i D_1 e)$.

The queue length and waiting time distributions for the discrete time version of this queue are computed by the function Q_DT_MAP_Geo_C. In this case the services are geometrically distributed with parameter p and service completions may occur simultaneously. By setting up an MC similar to the one just described one obtains a GI/M/1-type structure from level c onward. The process is characterized by the matrices $A_{-c+j} = D_0 b(c, c-j) + D_1 b(c, c-j+1)$ for $j = 0, \ldots, c+1$. The term $b(c,j)$ is the binomial probability of having j service completions among c busy servers, i.e., $b(c,j) = \binom{c}{j} p^j (1-p)^{c-j}$ for $j = 0, \ldots, c$, and it is equal to zero for $j < 0$ or $j > c$. The R matrix of the process is computed using the GIM1_R function of the SMCSolver and it is used to censor the MC between levels 0 and c. This finite MC has a skip-free-to-the-right structure and therefore the algorithm in [22] can be applied to compute the steady-state probability vector for the first $c+1$ levels. This vector is similarly renormalized and the probability vector for higher levels can be computed from π_c and R. The waiting time distribution is then recursively computed from the queue length distribution at arrival epochs.

6. REFERENCES

[1] S. Asmussen and M. Bladt. Renewal theory and queueing algorithms for matrix-exponential distributions. In S. Chakravarthy and A. S. Alfa, editors, *Matrix-Analytic Methods in Stochastic Models*, pages 313–341. Marcel Dekker, New York, 1996.

[2] S. Asmussen and M. Bladt. Point processes with finite-dimensional conditional probabilities. *Stochastic Processes and their Applications*, 82:127–142, 1999.

[3] S. Asmussen and J. Moller. Calculation of the steady state waiting time distribution in GI/PH/c and MAP/PH/c queues. *Queueing systems*, 37(1-3):9–29, 2001.

[4] S. Asmussen, O. Nerman, and M. Olsson. Fitting phase-type distributions via the EM algorithm. *Scandinavian Journal of Statistics*, 23:419–441, 1996.

[5] S. Asmussen and C. A. O'Cinneide. Matrix-exponential distributions. In S. Kotz, C. B. Read, and D. L. Banks, editors, *Encyclopedia of Statistical Sciences*, volume 2, pages 435–440. Wiley, New York, 1998.

[6] N. G. Bean and B. F. Nielsen. Quasi-birth-and-death processes with rational arrival process components. Technical Report 2007-20, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2007.

[7] D. A. Bini, G. Latouche, and B. Meini. *Numerical methods for structured Markov chains*. Oxford University Press, 2005.

[8] D. A. Bini, B. Meini, S. Steffé, and B. Van Houdt. Structured Markov chains solver: algorithms. In *SMCtools Workshop*, Pisa, Italy, 2006. ACM Press.

[9] D. A. Bini, B. Meini, S. Steffé, and B. Van Houdt. Structured Markov chains solver: software tools. In *SMCtools Workshop*, Pisa, Italy, 2006. ACM Press.

[10] M. Bladt and M. F. Neuts. Matrix-exponential distributions: calculus and interpretations via flows. *Stochastic Models*, 19:113–124, 2003.

[11] A. Bobbio, A. Horváth, and M. Telek. Matching three moments with minimal acyclic phase type distributions. *Stochastic Models*, 21:303–326, 2005.

[12] L. Bodrog, A. Horváth, and M. Telek. Moment characterization of matrix exponential and Markovian arrival processes. *Annals of Operations Research*, 160:51–68, 2007.

[13] M. Fackrell. Fitting with matrix-exponential distributions. *Stochastic Models*, 21:377–400, 2005.

[14] H. R. Gail, S. L. Hantler, and B. A. Taylor. Non-skip-free $M/G/1$ and $G/M/1$ type Markov chains. *Adv. in Appl. Probab.*, 29(3):733–758, 1997.

[15] D. Gaver, P. Jacobs, and G. Latouche. Finite Birth-and-Death models in randomly changing environments. *Adv. in Appl. Probab.*, 16:715–731, 1984.

[16] Q. HE. Analysis of a continuous time SM[K]/PH[K]/1/FCFS queue: Age process, sojourn times, and queue lengths. Working paper 04-01, Department of Industrial Engineering, Dalhousie University, 2004.

[17] Q. HE. Workload process, waiting times, and sojourn times in a discrete time MMAP[K]/SM[K]/1/FCFS queue. *Stochastic Models*, 20(4):415–437, 2004.

[18] Q. HE. Age process, workload process, sojourn times, and waiting times in a discrete-time SM[K]/PH[K]/1/FCFS queue. *Queueing Systems*, 49:363–403, 2005.

[19] Q. HE and H. Zhang. On matrix exponential distributions. *Advances in Applied Probability*, 39:271–292, 2007.

[20] R. Khayari, R. Sadre, and B. Haverkort. Fitting world-wide web request traces with the EM-algorithm. *Performance Evaluation*, 52:175–191, 2003.

[21] A. Lang and J. L. Arthur. Parameter approximation for phase-type distributions. In *Matrix-Analytic Methods in Stochastic Models, (S. R. Chakravarthy and A. S. Alfa (Editors))*, pages 151–206, New York, 1996. Marcel-Dekker, Inc.

[22] G. Latouche, P. Jacobs, and D. Gaver. Finite Markov chain models skip-free in one direction. *Naval Research Logistics Quarterly*, 31:571–588, 1984.

[23] G. Latouche and V. Ramaswami. The PH/PH/1 queue at epochs of queue size change. *Queueing Systems*, 25:97–114, 1997.

[24] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods and stochastic modeling.* SIAM, Philadelphia, 1999.

[25] L. Lipsky. *Queueing Theory: a Linear Algebraic Approach.* Macmillan, New York, 1992.

[26] M. Neuts. *Matrix-Geometric Solutions in Stochastic Models, An Algorithmic Approach.* John Hopkins University Press, 1981.

[27] M. Neuts. *Structured Stochastic Matrices of M/G/1 type and their applications.* Marcel Dekker, Inc., New York and Basel, 1989.

[28] B. Sengupta. Markov processes whose steady state distribution is matrix-exponential with an application to the GI/PH/1 queue. *Adv. in Appl. Probab.*, 21:159–180, 1989.

[29] M. Telek and G. Horváth. A minimal representation of Markov arrival processes and a moment matching method. *Performance Evaluation*, 64:1153–1168, 2007.

[30] A. van de Liefvoort. The moment problem for continuous distributions. Technical report, University of Missouri, 1990.

[31] B. Van Houdt and C. Blondia. The waiting time distribution of a type k customer in a MMAP[K]/PH[K]/c (c=1,2) queue using QBDs. *Stochastic Models*, 20(1):55–69, 2004.