A fast Newton's iteration for M/G/1-type and GI/M/1-type Markov chains

Juan F. Pérez[*], Miklós Telek[†] and Benny Van Houdt[‡]

Abstract

In this paper we revisit Newton's iteration as a method to find the G or R matrix in M/G/1-type and GI/M/1-type Markov chains. We start by reconsidering the method proposed in [15] which required $O(m^6 + Nm^4)$ time per iteration, and show that it can be reduced to $O(Nm^4)$, where m is the block size and N the number of blocks. Moreover, we show how this method is able to further reduce this time complexity to $O(Nr^3 + Nm^2r^2 + m^3r)$ when A_0 has rank $r < m$. In addition, we consider the case where $[A_1 A_2 \dots A_N]$ is of rank $r < m$ and propose a new Newton's iteration method which is proven to converge quadratically and that has a time complexity of $O(Nm^3 + Nm^2r^2 + mr^3)$ per iteration. The computational gains in all the cases are illustrated through numerical examples.

1 Introduction

Discrete-time M/G/1-type and GI/M/1-type Markov chains (MCs) are generalizations of the well-known Quasi-Birth-Death Markov chains and have been studied and used extensively by various authors [17, 13, 2]. An M/G/1-type MC is defined by a transition probability matrix of the form

$$P = \begin{bmatrix} B_0 & B_1 & B_2 & B_3 & \dots \\ C & A_1 & A_2 & A_3 & \dots \\ & A_0 & A_1 & A_2 & \ddots \\ & & A_0 & A_1 & \ddots \\ 0 & & & \ddots & \ddots \end{bmatrix}, \tag{1}$$

[*]Department of Electrical and Electronics Engineering, Universidad de los Andes, Bogotá, Colombia. Email: jf.perez33@uniandes.edu.co

[†]Department of Telecommunications, Budapest University of Technology and Economics, Budapest, Hungary. Email: telek@hit.bme.hu

[‡]Performance Analysis of Telecommunication Systems Research Group, Department of Mathematics and Computer Science, University of Antwerp - IBBT, Antwerp, Belgium. Email: benny.vanhoudt@ua.ac.be

while the transition probability matrix of a GI/M/1-type MC can be expressed as

$$P = \begin{bmatrix} B_0 & C & & & 0 \\ B_1 & A_1 & A_0 & & \\ B_2 & A_2 & A_1 & A_0 & \\ B_3 & A_3 & A_2 & A_1 & \ddots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{bmatrix}, \tag{2}$$

where B_0 and A_1 are square matrices of size m_0 and m, respectively. Denote N as the smallest index i such that A_i, for $i > N$, is (numerically) zero. The key step in determining the steady state probability vector of an M/G/1-type MC, if it exists, is computing the matrix G that is the smallest non-negative solution to the matrix equation [17]

$$G = \sum_{i=0}^{N} A_i G^i. \tag{3}$$

Similarly, for the GI/M/1-type the matrix R is the smallest non-negative solution to the matrix equation [16]

$$R = \sum_{i=0}^{N} R^i A_i. \tag{4}$$

Furthermore, any M/G/1-type MC can be transformed into a GI/M/1-type MC and vice versa by means of either the Ramaswami [23] or Bright [24] dual, and the G (R) matrix can be expressed directly in terms of the R (G) matrix of the dual chain. For instance, in the SMCSolver tool [5], the R matrix of a GI/M/1-type MC is computed via the G matrix of its dual.

Various algorithms have been developed to compute G iteratively. These include functional iterations (FI) [17], point-wise cyclic reduction (CR) [3], the invariant subspace (IS) approach [1], the Ramaswami reduction (RR) [4] and the Newton iteration (NI) [15, 22, 12]. The FI algorithm requires only $O(Nm^3)$ time per iteration, but converges only linearly, meaning thousands of iterations might be required. The other algorithms have at least quadratic convergence (meaning about 15 iterations suffice in most cases), where the CR algorithm typically performs best, although CR does require substantially more memory than FI (the exact amount depends on the number of roots required by the point-wise evaluation). The IS algorithm performs well for very small N values, but is often numerically unstable for N values of 10 or above, while the RR may outperform the CR algorithm on occasion, but typically requires substantially more time (and memory). Finally, the NI as introduced in [15] and [12] requires $O(m^6 + Nm^4)$ time per iteration, making it impractical even for small values of m and N, as most of the algorithms have a cubic time complexity in m. To mitigate the high computation time per iteration, a sub-Newton iteration was defined in [22], at the cost of many more iterations.

The computational complexity of the algorithms listed above can sometimes be further reduced by exploiting the internal structure of the matrices A_0 to A_N. A particularly useful example is the class of M/G/1-type MCs for which the matrix A_0 has only r non-zero columns, a property referred to as restricted downward transitions. In this case the matrix G contains (at most) r non-zero columns as well [17]. Restricted downward transitions were studied in [9] for the linearly convergent FI algorithms only and in the special case of $N = 2$. An algorithm with quadratic convergence that makes use of the CR algorithm for $N \geq 2$ was introduced in [21]. The operation and computational complexity of this algorithm, termed the RT-CR algorithm, is briefly discussed in Appendix C as it is compared in Section 6 with the Newton iteration. In this paper we consider both general M/G/1-type MCs as well as those with restricted downward transitions. In fact, we consider the more general low-rank downward transitions that only demand that the matrix A_0 has rank $r < m$, as opposed to having only r non-zero columns. In fact, in some cases the low-rank downward transitions can be reduced to restricted downward transitions as explained in [19] (at the expense of a slight increase in m) . Additionally, we also consider M/G/1-type MCs with low-rank local and upward transitions, which means that $[A_1 A_2 \ldots A_N]$ is of rank $r < m$.

In this paper the following contributions are made regarding the computation of G (and R):

1. In Section 2 we revisit the Newton iteration as defined in [15] and indicate that the time complexity per iteration can be reduced from $O(m^6 + Nm^4)$ to $O(Nm^4)$ using a Schur-decomposition method recently introduced in [20, 21].

2. When the matrix A_0 is of rank r, meaning it can be decomposed as $A_0 = \hat{A}_0 \Gamma$ with $\hat{A}_0$ an $m \times r$ and Γ an $r \times m$ matrix, the time complexity per iteration can be further reduced to $O(Nr^3 + Nm^2r^2 + m^3r)$, as indicated in Section 3. An M/G/1-type MC for which A_0 has this property is termed an M/G/1-type MC with low-rank downward transitions.

3. We introduce a new Newton iteration (in Section 4) based on the U matrix [13], and prove that it converges quadratically to the matrix U, from which G (or R) can be computed directly, e.g, $G = (I - U)^{-1} A_0$ [13].

4. When the matrices A_i for the M/G/1-type MC can be decomposed as $\Gamma \hat{A}_i$, for $i > 0$, where Γ an $m \times r$ and $\hat{A}_i$ is an $r \times m$ matrix, we refer to the MC as having low-rank local and upward transitions. Note, this is equivalent to stating that the matrix $[A_1 A_2 \ldots A_N]$ has rank r. Similarly, for GI/M/1-type MCs we state that they have low-rank local and downward transitions if A_i can be decomposed as $\hat{A}_i \Gamma$, for $i > 0$, where $\hat{A}_i$ is an $m \times r$ and Γ an $r \times m$ matrix. Using the novel Newton iteration we show in Section 5 that the time per iteration to compute G (or R) can be reduced from $O(Nm^4)$ to $O(Nm^3 + Nm^2r^2 + mr^3)$ in case of local upward (or downward) low-rank transitions.

5. We have integrated these new solution methods into the SMCSolver tool and used it to exemplify the substantial gains that the above algorithms offer to the existing methods (see Section 6), especially for limited m values and large N values. Further, the novel methods require only as much memory as the FI algorithm (that is, $O(Nm^2)$).

In the paper we will limit ourselves to M/G/1-type MCs (except for Section 6) as the results for the GI/M/1-type follow immediately by duality [23].

2 Accelerating Newton's iteration to compute G

In this section we revisit Newton's iteration as introduced in [15], in order to show how the time per iteration can be reduced significantly. As mentioned in the introduction, the matrix G is the minimal nonnegative solution of $X = \sum_{v=0}^{N} A_v X^v$. Denote by $\langle X, Y\rangle$ the set of matrices Z such that $X \leq Z \leq Y$, where we use the natural partial order, i.e., $X \leq Y$ if $X_{ij} \leq Y_{ij}$ for all i, j. We can restate the problem of computing G as that of finding the matrix X such that $\mathcal{G}X = 0$, where the operator $\mathcal{G}$ is defined as $\mathcal{G}X = \sum_{v=0}^{N} A_v X^v - X$, for every matrix X in $\langle 0, G\rangle$.

The Gateaux derivative [18] of this operator is given by

$$\mathcal{G}'(X)H = \sum_{v=1}^{N}\sum_{j=0}^{v-1} A_v X^j H X^{v-1-j} - H.$$

Using this operator we can define a Newton's iteration that starts with $G_0 = 0$, and iteratively computes $G_{k+1} = G_k - \mathcal{G}'(G_k)^{-1}\mathcal{G}(G_k)$, for $k \geq 0$. It has been shown in [15] that $\mathcal{G}$ is order-convex and $\mathcal{G}'$ is isotone on $\langle 0, G\rangle$, and it is easy to see that $\mathcal{G}(0) \geq 0$ and $\mathcal{G}(G) = 0$. Define the quantity ρ, referred to as the drift of the chain, as

$$\rho = \pi\beta, \tag{5}$$

where π is the stationary probability vector of the irreducible stochastic matrix $A = \sum_{i=0}^{N} A_i$, β is given by $\beta = \sum_{i=1}^{N} iA_i e$, and e is a column vector of ones. The drift is equal to one when the MC is null recurrent, while $\rho < 1$ and $\rho > 1$ correspond to the positive recurrent and transient cases, respectively [13]. It was also shown in [15] that, if $\rho \neq 1$, the inverse $\mathcal{G}'(X)^{-1}$ exists and is a non-positive matrix for every $X \in \langle 0, G\rangle$. Moreover, from [12] we know that $\mathcal{G}'$ is Lipschitz-continuous. Based on these results we can restate (a slightly stronger version of) Theorem 9 in [15], which is a direct application of the Monotone Newton Theorem in [18, 13.3.4].

Theorem 1 (Theorem 9 in [15])**.** *Let $G_0 = 0$, and let the Newton iterates G_{k+1} be given by $G_{k+1} = G_k - \mathcal{G}'(G_k)^{-1}\mathcal{G}G_k$, for $k \geq 0$. Then, if $\rho \neq 1$, the iterates satisfy $\lim_{k\to\infty} G_k = G$, and $G_{k+1} \geq G_k$, where G is the minimal nonnegative solution of Eq.* (3)*. Furthermore, the convergence toward G is quadratic, i.e., there is a positive constant c such that*

$$||G_{k+1} - G|| \leq c||G_k - G||^2.$$

The Newton's iterates G_k can be restated as $G_{k+1} = G_k + X_k$, where X_k is the solution to $\mathcal{G}'(G_k)X_k = -\mathcal{G}G_k$, for $k \geq 1$. Thus, X_k is the solution to

$$X_k - \sum_{v=1}^{N}\sum_{j=0}^{v-1} A_v G_k^j X_k G_k^{v-1-j} = \sum_{v=0}^{N} A_v G_k^v - G_k,$$

which, after rearranging the terms and interchanging the sums can be rewritten as

$$X_k - \sum_{j=0}^{N-1}\sum_{v=j+1}^{N} A_v G_k^{v-1-j} X_k G_k^j = \sum_{v=0}^{N} A_v G_k^v - G_k. \tag{6}$$

If we define $S_{k,i} = \sum_{j=i}^{N} A_j G_k^{j-i}$, the previous equation is equivalent to

$$(S_{k,1} - I)\, X_k + \sum_{j=1}^{N-1} S_{k,j+1} X_k G_k^j = G_k - S_{k,0}, \tag{7}$$

which clearly is a linear equation of the form $\sum_{j=0}^{N-1} B_j X C^j = E$, where all the matrices involved are of size m. By applying general methods, we would need $O(m^6)$ to solve this system since it is a linear system with m^2 unknowns and equations. However, as explained in [20, 21], a linear system of this form can be solved faster by applying a Schur decomposition on the matrix C, which in our case is the $m \times m$ matrix G_k, and then solving m linear systems with m unknows and equations. For completeness, we have included a description of this solution method in Appendix B. We would like to stress that the equation of the form $\sum_{j=0}^{N-1} B_j X C^j = E$, which appeared in [20, 21], was used for a very different purpose (see Appendix C for details) and the matrices B_j, for $j = 0, \ldots, N-1$, and E were not square either. Further, the Newton iteration solves such a linear system during each iteration, while the algorithm in [21] is also in part iterative, but the linear system appears only after the iterative part has ended.

To summarize, to compute the matrix X_k for the iterate G_{k+1}, we need to determine the matrices that premultiply X_k in Eq. (7), at a cost of $O(Nm^3)$ time, perform the Schur decomposition of G_k, requiring $O(m^3)$ time, and compute N powers of the upper quasi-triangular matrix obtained from the decomposition (see Appendix B), at a cost of $O(Nm^3)$ time. Afterward, to find all the columns of X_k, apart from solving m linear systems at a cost of $O(m^3)$ each, we also need to build the matrices that define these linear systems, which requires $O(Nm^4)$ time in total. As a result, the total cost of this method is $O(Nm^4)$, showing a significant reduction in computation time compared to the general approach, and turning this method into a feasible alternative for moderate block size m. We will illustrate this in Section 6 with some numerical examples. Also, the space complexity of this method is $O(Nm^2)$ only. In the next section we will show that this method has another advantage over other quadratically-convergent methods: if the matrix A_0 has low rank, a case that we refer to as low-rank downward transitions, the Newton's method can be adapted to reduce its computational cost per iteration.

3 The case of low-rank downward transitions

We now assume that the matrix A_0 has rank $r < m$, such that it can be written as $A_0 = \hat{A}_0\Gamma$, with $\hat{A}_0$ and Γ of size $m \times r$ and $r \times m$, respectively. If we apply the Newton method to this case, we find that all the matrices X_k can be written as $\hat{X}_k\Gamma$. This is obviously true for $k = 0$, and assume that it is true for $j = 0, \ldots, k-1$. Therefore, G_k can be written as $G_k = \hat{G}_k\Gamma$, since

$G_k = \sum_{j=0}^{k-1} X_j = (\sum_{j=0}^{k-1} \hat{X}_j)\Gamma$. Now we can rewrite (6) as

$$\begin{aligned} X_k &= \hat{A}_0\Gamma + \sum_{j=1}^{N} A_j G_k^{j-1}\hat{G}_k\Gamma - \hat{G}_k\Gamma + \sum_{v=1}^{N} A_v G_k^{v-1} X_k + \sum_{j=1}^{N-1}\sum_{v=j+1}^{N} A_v G_k^{v-1-j} X_k G_k^{j-1}\hat{G}_k\Gamma, \\ &= \left(I - \sum_{v=1}^{N} A_v G_k^{v-1}\right)^{-1}\left(\hat{A}_0 + \sum_{j=1}^{N} A_j G_k^{j-1}\hat{G}_k - \hat{G}_k + \sum_{j=1}^{N-1}\sum_{v=j+1}^{N} A_v G_k^{v-1-j} X_k G_k^{j-1}\hat{G}_k\right)\Gamma, \end{aligned}$$

and therefore X_k can be written as the product of an $m \times r$ matrix $\hat{X}_k$ and Γ. The inverse on the right-hand-side exists since $0 \leq \sum_{v=1}^{N} A_v G_k^{v-1} \leq \sum_{v=1}^{N} A_v G^{v-1}$, for every $k \geq 0$, and the spectral radius of $\sum_{v=1}^{N} A_v G^{v-1}$ is strictly less than one [2, Theorem 4.15]. Therefore, $I - \sum_{v=1}^{N} A_v G_k^{v-1}$ is a nonsingular M-matrix.

As a result, we can focus on finding $\hat{X}_k$ as the solution of

$$\begin{aligned} \hat{X}_k &= \hat{A}_0 + \sum_{j=1}^{N} A_j G_k^{j-1}\hat{G}_k - \hat{G}_k + \sum_{v=1}^{N} A_v G_k^{v-1}\hat{X}_k + \sum_{j=1}^{N-1}\sum_{v=j+1}^{N} A_v G_k^{v-1-j}\hat{X}_k\Gamma G_k^{j-1}\hat{G}_k, \\ &= \hat{A}_0 + \left(\sum_{j=1}^{N} A_j G_k^{j-1} - I\right)\hat{G}_k + \sum_{j=0}^{N-1} S_{k,j+1}\hat{X}_k(\Gamma\hat{G}_k)^j, \end{aligned}$$

or, equivalently,

$$(S_{k,1} - I)\hat{X}_k + \sum_{j=1}^{N-1} S_{k,j+1}\hat{X}_k(\Gamma\hat{G}_k)^j = \left(I - \sum_{j=1}^{N} A_j G_k^{j-1}\right)\hat{G}_k - \hat{A}_0.$$

This equation can be solved with the Schur decomposition method in [20, 21], as it was shown in the previous section with Eq. (7). In this case, the matrix that post-multiplies $\hat{X}_k$ is of size r, and therefore its Schur decomposition and the N powers of the associated upper quasi-triangular matrix take $O(Nr^3)$ time. Also, computing the matrices that pre-multiply $\hat{X}_k$ require $O(N(m^2 + mr^2))$ time. Finally, each of the r systems to solve takes $O(m^3)$, and setting up the matrices associated to all these systems takes $O(Nm^2r^2)$. The total cost is $O(Nr^3 + Nm^2r^2 + m^3r)$ time, implying an important gain compared to the general case shown in the previous section, especially when $r \ll m$, a case that often arises when low-rank downward transitions are present.

Remark 1. *As noted in the introduction, low-rank downward transitions include the so-called* restricted *downward transitions as a special case [21]. In the latter case the matrix A_0 has r nonzero columns only, and therefore its rank is at most r. This type of transitions has been analyzed for the case of M/G/1-type MCs in [21] and in Section 6 we will compare the method of [21] with the above Newton iteration.*

Remark 2. *In general the matrix $\hat{A}_0$ may be of mixed sign and therefore so are the $\hat{G}_k$ matrices, which may potentially cause numerical issues. However, as in the examples of Section 6, the matrices $\hat{A}_0$ and Γ are in fact often nonnegative as these low-rank properties are typically caused by a renewal in one (or more) of the dimensions of the multi-dimensional MC.*

4 A new Newton's iteration

In this section we introduce a new Newton's iteration that has the same complexity per iteration as the one discussed in Section 2. Although it does not provide a computational gain, it features the ability to exploit low-rank local and upward transitions, as will be discussed in Section 5. The new iteration is based on the matrix U [13], which is the generator of the censored Markov chain on level i, starting from level i, before the first transition to level $i-1$. Based on a level crossing argument we can write

$$U = \sum_{i=1}^{N} A_i G^{i-1} = \sum_{i=1}^{N} A_i \Big((I-U)^{-1} A_0 \Big)^{i-1}. \tag{8}$$

To find U we therefore need to solve the equation $X - \sum_{i=1}^{N} A_i \left((I-X)^{-1} A_0 \right)^{i-1} = 0$, and G can be obtained from $G = (I-U)^{-1}A_0$ [13]. To this end we define the operators $\mathcal{L}$ on $\langle 0, U \rangle$, and $\mathcal{H}$ on $\langle 0, G \rangle$ as

$$\mathcal{L}X = (I-X)^{-1}A_0 \quad \text{and} \quad \mathcal{H}W = \sum_{i=1}^{N} A_i W^{i-1}.$$

From the definition of U and the irreducibility of the Markov chain, we know that U is a sub-stochastic matrix with spectral radius $sp(U) < 1$ [17, page 98], and thus the matrix $(I-U)^{-1}$ exists and is equal to $\sum_{k \geq 0} U^k$. Since, for $0 \leq X \leq Y$, $sp(X) \leq sp(Y)$, the matrix $(I-X)^{-1}$, and the operator $\mathcal{L}$, are well defined for all X in $\langle 0, U \rangle$. Clearly,

$$\mathcal{L}X \leq \mathcal{L}Y, \text{ for } X \leq Y \text{ in } \langle 0, U \rangle, \tag{9}$$

$$\mathcal{H}V \leq \mathcal{H}W, \text{ for } V \leq W \text{ in } \langle 0, G \rangle. \tag{10}$$

Also, $\mathcal{L}U = (I-U)^{-1}A_0 = G$, and $\mathcal{H}G = \sum_{i=1}^{N} A_i G^{i-1} = U$. Hence, the operator $\mathcal{L}$ maps $\langle 0, U \rangle$ on $\langle 0, G \rangle$, while $\mathcal{H}$ maps $\langle 0, G \rangle$ on $\langle 0, U \rangle$.

We now define the operator $\mathcal{F}$ on $\langle 0, U \rangle$ as $\mathcal{F}X = (\mathcal{I} - \mathcal{H}\mathcal{L})X$, such that to find U we need to solve $\mathcal{F}X = 0$. This can be done iteratively by means of Newton's method, starting with $U_0 = 0$ and computing $U_{k+1} = U_k - \mathcal{F}'(U_k)^{-1}\mathcal{F}U_k$, for $k \geq 0$, where $\mathcal{F}'(U_k)^{-1}$ denotes the inverse of the derivative of $\mathcal{F}$ evaluated at U_k. The correctness and quadratic convergence of this iterative scheme is stated in the following result, which, relying on Lemmas A.1 and A.2 in Appendix A, follows directly from the Monotone Newton Theorem in [18, 13.3.4].

Theorem 2. *Let U_0 be an $m \times m$ matrix in $\langle 0, U \rangle$ such that $\mathcal{F}(U_0) \leq 0$, and let the Newton iterates U_{k+1} be given by $U_{k+1} = U_k - \mathcal{F}'(U_k)^{-1}\mathcal{F}U_k$, for $k \geq 0$. Then, if $\rho \neq 1$, the iterates satisfy $\lim_{k\to\infty} U_k = U$, and $U_{k+1} \geq U_k$, where U is the minimal nonnegative solution of Eq. (8). Furthermore, the convergence toward U is quadratic, i.e., there is a positive constant c such that*

$$||U_{k+1} - U|| \leq c||U_k - U||^2.$$

To comply with the assumptions of Theorem 2 we choose $U_0 = 0$, since $\mathcal{F}(0) = -\sum_{i=1}^{N} A_i A_0^{i-1} \le 0$. Moreover, we can re-state the iterates as $U_{k+1} = U_k + Y_k$, where Y_k solves $\mathcal{F}'(U_k)Y_k = -\mathcal{F}U_k$, for $k \ge 0$. Since $\mathcal{F}'(U_k)Y_k$ is equal to (see Appendix A)

$$\begin{aligned}
\mathcal{F}'(U_k)Y_k &= \left(\mathcal{I} - \mathcal{H}'(\mathcal{L}U_k)\mathcal{L}'(U_k)\right) Y_k,\\
&= Y_k - \mathcal{H}'(\mathcal{L}U_k)(I-U_k)^{-1}Y_k(I-U_k)^{-1}A_0,\\
&= Y_k - \sum_{i=2}^{N} A_i \sum_{j=1}^{i-1}((I-U_k)^{-1}A_0)^{j-1}(I-U_k)^{-1}Y_k((I-U_k)^{-1}A_0)^{i-j},
\end{aligned}$$

Y_k is found as the solution to

$$Y_k - \sum_{i=2}^{N} A_i \sum_{j=1}^{i-1}((I-U_k)^{-1}A_0)^{j-1}(I-U_k)^{-1}Y_k((I-U_k)^{-1}A_0)^{i-j} = \sum_{i=1}^{N} A_i \left((I-U_k)^{-1}A_0\right)^{i-1} - U_k. \quad (11)$$

By defining $R_{k,j} = \sum_{i=j+1}^{N} A_i \left((I-U_k)^{-1}A_0\right)^{i-j-1}(I-U_k)^{-1}$, for $j = 1, \dots, N-1$, interchanging the sums, and rearranging the terms, we can rewrite (11) as

$$Y_k - \sum_{j=1}^{N-1} R_{k,j} Y_k ((I-U_k)^{-1}A_0)^j = \sum_{i=1}^{N} A_i \left((I-U_k)^{-1}A_0\right)^{i-1} - U_k, \quad (12)$$

which is a linear equation of the form $\sum_{j=0}^{N-1} B_j X C^j = E$. As mentioned before, this system can be solved with the method introduced in [20, 21], a description of which is outlined in Appendix B. Moreover, since the size of the matrices involved is identical to that of the matrices in the Newton's iteration described in Section 2, the time and memory complexity of this method, per iteration, are $O(Nm^4)$ and $O(Nm^2)$ respectively. In addition to being well-suited for numerical computation, the iteration introduced in this section is also able to exploit low-rank local and upward transitions. This is the topic of the next section.

5 Exploiting low-rank local and upward transitions

In this section we consider the case where the $m \times m$ blocks $\{A_i, 1 \le i \le N\}$ can be decomposed as $A_i = \Gamma \hat{A}_i$, where Γ and $\hat{A}_i$ are of size $m \times r$ and $r \times m$, respectively. This property is referred to as low-rank local and upward transitions. From Eq. (8), we can write U as

$$U = \sum_{i=1}^{N} A_i \left((I-U)^{-1}A_0\right)^{i-1} = \Gamma \left(\sum_{i=1}^{N} \hat{A}_i \left((I-U)^{-1}A_0\right)^{i-1}\right) = \Gamma \hat{U},$$

which shows that in this case U is of rank r, while $G = (I-U)^{-1}A_0$ is of rank m in general. As a result we can first compute $\hat{U}$ as the solution of

$$\hat{U} = \sum_{i=1}^{N} \hat{A}_i \left((I-\Gamma\hat{U})^{-1}A_0\right)^{i-1},$$

and then obtain G from $G = (I - \Gamma\hat{U})^{-1}A_0$. For this purpose we specialize the Newton's iteration introduced in the previous section, since the iterates $U_{k+1} = U_k + Y_k$, where Y_k solves (11), can be written in the form $U_k = \Gamma\hat{U}_k$. Again, this statement can be proven by induction, and it obviously holds for $k = 0$. Assume that $U_k = \Gamma\hat{U}_k$, then from (11) we can write Y_k as

$$Y_k = \Gamma \left[\sum_{i=2}^{N} \hat{A}_i \sum_{j=1}^{i-1} \left((I - U_k)^{-1}A_0\right)^{j-1} (I - U_k)^{-1} Y_k \left((I - U_k)^{-1}A_0\right)^{i-j} \right.$$
$$\left. + \sum_{i=1}^{N} \hat{A}_i \left((I - U_k)^{-1}A_0\right)^{i-1} - \hat{U}_k \right],$$

which means that Y_k has the form $\Gamma\hat{Y}_k$, and the same holds for $U_{k+1} = U_k + Y_k$.

Using this result and Eq. (11) we obtain that $\hat{Y}_k$ is the solution to

$$\hat{Y}_k - \sum_{i=2}^{N} \hat{A}_i \sum_{j=1}^{i-1} \left((I - U_k)^{-1}A_0\right)^{j-1} (I - U_k)^{-1}\Gamma\hat{Y}_k \left((I - U_k)^{-1}A_0\right)^{i-j} =$$
$$\sum_{i=1}^{N} \hat{A}_i \left((I - U_k)^{-1}A_0\right)^{i-1} - \hat{U}_k,$$

which, by defining $\hat{R}_{k,j} = \sum_{i=j+1}^{N} \hat{A}_i \left((I - U_k)^{-1}A_0\right)^{i-j-1} (I - U_k)^{-1}\Gamma$, and rearranging the terms, can be rewritten as

$$\hat{Y}_k - \sum_{j=1}^{N-1} \hat{R}_{k,j}\hat{Y}_k((I - U_k)^{-1}A_0)^j = \sum_{i=1}^{N} \hat{A}_i \left((I - U_k)^{-1}A_0\right)^{i-1} - \hat{U}_k. \tag{13}$$

This again is a linear equation of the form $\sum_{j=0}^{N-1} B_j X C^j = E$, but in this case the unknown matrix $\hat{Y}_k$ is of size $r \times m$, and it is pre-multiplied by square matrices of size r, the computation of which takes $O(N(m^2r + mr^2))$ time. As the matrix $((I - U_k)^{-1}A_0)$ is of size m, computing its Schur decomposition and the powers of the associated upper quasi-triangular matrix requires $O(Nm^3)$ time. To obtain $\hat{Y}_k$ it is also necessary to solve m linear systems, the solution of each demands $O(r^3)$ time, while setting-up the matrices involved in the solution of these systems takes $O(Nm^2r^2)$ time. The total cost per iteration in this method is therefore $O(Nm^3 + Nm^2r^2 + mr^3)$ time.

Remark 3. *In general the matrices $\hat{A}_i$, for $i > 0$, may be of mixed sign and therefore so are the $\hat{U}_k$ matrices, which may potentially cause numerical issues. However, as in the examples of Section 6, the matrices $\hat{A}_i$ and Γ are in fact often nonnegative as these low-rank properties are typically caused by a renewal in one (or more) of the dimensions of the multi-dimensional MC.*

Remark 4. *A special case of interest arises when, among $\{A_1, \ldots, A_N\}$, the only nonzero matrix is A_N, and this matrix is of rank r. An example of an M/G/1-type Markov chain with this structure appeared in the study of a production/inventory system with periodic review [6]. As a matter of fact, the MC analyzed in [6] is of the GI/M/1-type, but, as mentioned before, the methods here can* *be applied on the dual of the original MC, or directly on the GI/M/1-type MC after some minor adaptations. In this case we can rewrite Eq.* (13) *as*

$$\hat{Y}_k - \sum_{j=1}^{N-1} \hat{A}_D \left((I-U_k)^{-1}A_0\right)^{N-j-1} (I-X)^{-1}\Gamma\ \hat{Y}_k \left((I-U_k)^{-1}A_0\right)^{j} = \hat{A}_D \left((I-U_k)^{-1}A_0\right)^{N-1} - \hat{U}_k,$$

which is again a linear equation of the type $\sum_{j=0}^{N-1} B_j X C^j = E$ *where the unknown matrix is of size* $r \times m$*. Therefore, this system can be solved at the same cost than* (13)*, but the cost of computing the matrices that pre-multiply* $\hat{Y}_k$ *is significantly lower.*

6 Numerical Results

In this section we illustrate the computational behavior of the Newton algorithm by means of three examples: a Markovian (BMAP/PH/1) queue, a semi-Markovian (SM/PH/1) queue, and a production/inventory system. The MC to describe the first system is of the M/G/1 type (1), and has the low-rank downward transitions property, i.e., $\text{rank}(A_0) = r < m$. In the second and third cases the MCs are of the GI/M/1 type (2), and have low-rank local and downward transitions, i.e., $\text{rank}([A_1 A_2 \ldots A_N]) = r < m$. Consequently, their duals are M/G/1-type MCs and have low-rank local and upward transitions. The Newton's iteration will be compared with the functional iterations and cyclic reduction, as well as with the restricted transitions approach introduced in [21] for the case of the Markovian queue.

6.1 The BMAP/PH/1 queue

Our first example is the continuous-time BMAP/PH/1 queue, which can be analyzed with an M/G/1-type MC that has low-rank downward transitions. We start by describing the arrival process and service-time distribution at this queue, and then introduce the blocks of the M/G/1-type MC, and the low-rank property of the block A_0.

The arrival process at this queue is a Batch Markovian Arrival process (BMAP) [14], characterized by the $m_a \times m_a$ matrices $\{D_0, D_1, \ldots, D_L\}$. This point process is driven by an underlying continuous-time MC (CTMC) with $m_a \times m_a$ generator matrix $D = \sum_{j=0}^{L} D_j$, where L is the maximum batch size. The (i,j)-th entry of D_k holds the rate at which, when the underlying chain is in state i, a batch of size k arrives and the chain makes a transition to state j, for $1 \leq i,j \leq m_a$, and $1 \leq k \leq L$. The off-diagonal entries of the matrix D_0 hold the rates related to transitions without arrivals, and its (negative) diagonal entries are such that $De = 0$, where 0 is a column vector with all its entries equal to zero. In general, this process is able to model correlation between the inter-arrival times (IATs) and the batch size distribution.

On the other hand, the service times follow a Phase-Type (PH) distribution [13, 16] characterized by (m_s, α, T). A PH distribution describes the time until absorption in a CTMC with m_s transient states and one absorbing state. The initial probability distribution on the transient states

Table 1: BMAP/PH/1 queue - Time (sec) to compute G - $m = 40$, $\gamma = 0.5$ - The effect of L

L	10					20				
load	FI	CR	NI	RT-CR	NI-LR	FI	CR	NI	RT-CR	NI-LR
0.1	0.02	1.63	0.27	0.06	0.02	0.03	3.16	0.64	0.06	0.02
0.3	0.06	2.92	0.31	0.08	0.00	0.09	4.38	0.75	0.11	0.02
0.5	0.16	3.70	0.31	0.08	0.02	0.25	5.06	0.89	0.13	0.02
0.7	0.45	4.48	0.38	0.11	0.02	0.70	10.1	1.13	0.14	0.02
0.9	1.70	6.39	0.53	0.16	0.02	2.77	11.2	1.27	0.19	0.02
0.99	21.0	7.38	1.05	0.16	0.09	24.4	12.0	12.2	0.20	0.06
L	40					80				
load	FI	CR	NI	RT-CR	NI-LR	FI	CR	NI	RT-CR	NI-LR
0.1	0.06	8.45	0.84	0.23	0.02	0.13	12.7	1.25	0.78	0.05
0.3	0.16	9.13	0.98	0.33	0.03	0.28	25.5	1.45	0.95	0.06
0.5	0.41	13.3	1.14	0.36	0.03	0.73	29.5	1.67	1.06	0.06
0.7	1.27	14.8	1.31	0.41	0.03	2.25	31.2	1.92	1.19	0.08
0.9	4.75	16.0	1.77	0.45	0.05	8.56	32.2	4.58	1.27	0.14
0.99	42.1	17.5	15.2	0.50	0.16	75.4	34.6	22.1	1.33	0.25

is given by the $1 \times m_s$ vector α, and the transitions between the transient states are ruled by the $m_s \times m_s$ transient generator T. The (i, j)-th entry of this matrix holds the non-negative rate at which a transition from state i to state j occurs, with $i \neq j$. The diagonal entries are negative and such that the matrix T has non-positive row sums, with at least one row having a strictly negative row-sum. The absorption rate at state i is therefore given by the i-th entry of the vector $t = -Te$, for $1 \leq i \leq m_s$. The cumulative distribution function of the time until absorption is given by $F(x) = 1 - \alpha \exp(Tx)e$, for $x \geq 0$. In the remainder of the paper the states of the MCs that underlie the arrival process and the service-time distribution are also referred to as phases.

To describe this queue we setup a MC with three variables: the number of customers in the queue, the state of the arrival process, and the state of the current service, if any. By organizing the state space lexicographically, the resulting MC is of the M/G/1 type, where the level keeps track of the number of customers, and the phase holds the information regarding the arrival and service processes. It is easy to see that the blocks $\{A_i, i \geq 0\}$ can be defined as

$$A_0 = I_{m_a} \otimes t\alpha, \quad A_1 = D_0 \oplus T, \quad A_i = D_{i-1} \otimes I_{m_s}, \; i = 2, \ldots, L+1, \tag{14}$$

where $\otimes$ and $\oplus$ stand for Kronecker product and sum [8], respectively. Clearly, the block A_0 can be written as $A_0 = \hat{A}_0\Gamma$, with $\hat{A}_0 = I_{m_a} \otimes t$, and $\Gamma = I_{m_a} \otimes \alpha$, and therefore this MC has low-rank downward transitions, a property that the Newton method can exploit as shown in Section 3. We now consider various instances of the BMAP/PH/1 queue, for which we compare the times to compute G using the general Newton algorithm (NI), with the U-based functional iterations

Table 2: BMAP/PH/1 queue - Time (sec) to compute G - $L = 10$, $\gamma = 0.5$ - The effect of m

m	40					80				
load	FI	CR	NI	RT-CR	NI-LR	FI	CR	NI	RT-CR	NI-LR
0.1	0.02	1.63	0.27	0.06	0.02	0.16	9.05	3.45	0.06	0.03
0.3	0.06	2.92	0.31	0.08	0.00	0.36	15.6	4.03	0.08	0.03
0.5	0.16	3.70	0.31	0.08	0.02	0.94	19.5	4.61	0.09	0.03
0.7	0.45	4.48	0.38	0.11	0.02	2.83	23.8	5.78	0.13	0.05
0.9	1.70	6.39	0.53	0.16	0.02	11.0	33.2	6.36	0.16	0.06
0.99	21.0	7.38	1.05	0.16	0.09	98.4	38.5	57.7	0.19	0.23
m	160					320				
load	FI	CR	NI	RT-CR	NI-LR	FI	CR	NI	RT-CR	NI-LR
0.1	1.23	57.5	52.9	0.19	0.14	9.66	302	823	0.70	0.86
0.3	2.97	66.7	61.8	0.22	0.19	22.2	437	962	0.84	1.02
0.5	7.91	119	70.8	0.25	0.19	58.2	-	1098	0.92	1.16
0.7	24.5	141	88.8	0.30	0.22	178	-	1373	1.03	1.59
0.9	96.4	199	133	0.34	0.42	690	-	1787	1.13	2.19
0.99	837	231	684	0.36	3.13	6220	-	13642	1.16	4.83

(FI) [13], and point-wise cyclic reduction (CR) [3]. We also show the computation times when the low-rank structure of A_0 is exploited, using the Newton algorithm (NI-LR), and the method introduced in [21] (RT-CR).

For this queue we consider three sets of experiments, in all of which we let the service time be the sum of n_s PH-distributed stages, each of them with mean one and squared coefficient of variation (SCV) equal to two. Each of the PH-distributed stages is of order two and matches these first two moments [25, 26]. Actually, the distribution of each stage is hyper-exponential with two phases, which is a subclass of order-2 PH distributions. Thus, the size of the PH representation of the service-time distribution is $m_s = 2n_s$. Also, the arrival process is a BMAP with arrival rate λ, IATs' SCV equal to 5, and decay rate of the autocorrelation function equal to γ. These characteristics are matched with a MAP of order two using the method in [7], which for our experiments (SCV>1 and $\gamma > 0$) results in a sequence of positively correlated hyper-exponential random variables. The arrival rate is determined by the load, defined as λ/μ, where μ is the service rate and is equal to $1/n_s$. The block size is $m = m_a m_s = 4n_s$, and the number of blocks N is equal to $L + 1$, where L is the maximum batch size. The batch sizes are assumed to be i.i.d. and uniformly distributed between one and L. As a result, the system is fully determined by n_s, L, γ and the load. In the first set of experiments we set $n_s = 10$, $\gamma = 0.5$ and vary ρ and L. These results are presented in Table 1.

The first clear observation is that for low loads FI beats the other general-purpose methods, but this behavior is completely reversed for large loads. In the latter case, which is the most

Table 3: BMAP/PH/1 queue - Time (sec) to compute G - $L = 5$, $m = 160$ - The effect of γ

γ	0.9					0.99				
load	FI	CR	NI	RT-CR	NI-LR	FI	CR	NI	RT-CR	NI-LR
0.1	0.78	25.1	42.3	0.06	0.09	0.80	24.9	42.3	0.08	0.09
0.3	1.83	46.9	49.4	0.09	0.11	1.97	47.0	49.5	0.09	0.09
0.5	7.39	67.9	63.7	0.11	0.13	10.1	72.8	71.0	0.11	0.14
0.7	52.8	106	84.9	0.14	0.16	435	137	350	0.17	1.33
0.9	225	150	284	0.17	0.30	1978	216	709	0.22	1.56
0.99	2087	184	707	0.22	1.14	4094	244	711	0.25	1.59

time-demanding for all the methods, FI performs very poorly and NI is the best among the three methods. In fact, in this case, where m is not large, NI is almost always better than CR, even for loads of 0.99, where NI's performance worsens significantly. This behavior of NI is clearly connected to ρ, as defined in Eq. (5), being very close to one, a case in which the linear system that must be solved in each iteration becomes ill-conditioned. We will discuss this further later on. Also, the gains in computation time obtained with NI hold despite the moderately large values of L. On top of these gains, the results clearly show that exploiting the low-rank transitions provides very important reductions, and NI-LR clearly outperforms all the other methods, including RT-CR. Recall that in this case the rank of A_0 is equal to $r = m_a = 2$, and therefore the ratio m/r is 20.

In the second set of experiments for this queue, the results of which are included in Table 2, we set the value of L equal to 10, γ equal to 0.5, and let the load and n_s, and therefore m, vary. The first scenario ($m = 40$, $L = 10$) coincides with the first scenario in the previous table, but now we observe the effect of the block size in the computation times. Here again, FI outperforms the other general-purpose algorithms for low to medium loads, but it falls behind under high loads. For block sizes up to 160, and loads up to 0.95 (not shown in the table), NI performs better than CR, although the gains diminish as m increases. Actually, for $m = 320$, CR outperforms NI, but this advantage cannot be fully exploited as CR results in an out-of-memory error for high loads (on a machine with 2GB RAM). For this case, FI also outperforms NI for every load value, showing how NI suffers from having an m^4 term in its time complexity. Regarding the methods that exploit the low-rank downward transitions, we observe a very important gain compared to general purpose methods, that becomes more apparent as m increases. This is to be expected since the rank of A_0 remains equal to $r = m_a = 2$, and thus the ratio m/r increases with the block size. NI-LR and RT-CR show a competitive behavior, with NI-LR performing better under small block sizes and low to medium loads, and RT-CR taking over for large block sizes and high loads.

In the third set of experiments we consider higher values for the decay rate of the autocorrelation function γ, namely 0.9 and 0.99. We fix $L = 5$, $n_S = 40$, thus $m = 160$, and consider a broad set of load values. The results, summarized in Table 3, show how badly the load affects the behavior of FI in this case. While for a load of 0.1 it takes less than second to compute G, this method requires more than an hour when both the load and γ are equal to 0.99. Also, CR shows a significantly

Table 4: BMAP/PH/1 queue - Condition number - $L = 5$, $m = 20$

load	Drift	Original	Shifted
0.5	0.90069	6.01E+01	3.74E+01
0.9	0.98105	9.77E+02	2.66E+01
0.99	0.99812	1.14E+04	2.49E+01
0.999	0.99981	1.15E+05	2.48E+01
0.9999	0.99998	1.15E+06	2.48E+01

better performance than NI for both low and high loads. Although for mid loads both methods show similar computation times, under high loads CR is able to run in a third of the time required by NI, and these are the most time-consuming scenarios. Hence, we observe that CR outperforms NI when the block size is large and the correlation is high, although CR cannot exploit this advantage fully due to its large memory requirements, as illustrated in the previous set of experiments. This example also reveals how large the gains of exploiting the low-rank of A_0 can be, using either RT-CR or NI-LR. In this case, partially due to the large block size of the original chain, NI-LR is outperformed by RT-CR, and the difference becomes larger as the load increases.

As mentioned above, the performance of NI worsens significantly as the load approaches one. This behavior can be observed in all the experiments performed in this section. Actually, in Table 3 this behavior appears even at a load of 0.9, as in this case the correlation is very high. These results are connected with the drift ρ of the MC, as defined in Eq. (5), being very close to one. In each iteration of NI applied to the matrix U (Theorem 2) we need to solve the equation $\mathcal{F}'(U_k)Y_k = -\mathcal{F}U_k$ for Y_k, since $U_{k+1} = U_k + Y_k$. However, Theorem 2 relies on Lemma A.2, that ensures that the inverse of $\mathcal{F}'(X)$ exists, for all $X \in \langle 0, U \rangle$, if $\rho \neq 1$. Moreover, from [15] we know that if the drift ρ is equal to one, $sp(\mathcal{T}(G)) = 1$, implying (see proof of Lemma A.2) that $\mathcal{F}'(U)$ is singular. This means that if ρ is close to one (the chain is almost null-recurrent), and the iterate U_k is close to U, the linear system that must be solved in an iteration of NI is close to being singular. This is illustrated in Table 4, where we show the condition number of the matrix associated with the linear system solved in the last iteration of NI (under the label Original). To be able to compute this number, we have chosen a small example where the block size m is equal to 20, so the matrix associated with the linear system is of size 400. Clearly, as the load increases the drift approaches one and the condition number of the matrix worsens. The effect of this ill-conditioning is that the precision of the solution Y_k is not enough to meet the termination criteria, which is that the infinity norm of the difference between two consecutive iterates be below 10^{-14}. The algorithm therefore terminates because it reaches the maximum number of iterations allowed (50), causing long computation times. An alternative to this problem is to use the shift technique [10, 2], which improves the conditioning of the linear system. This can be seen in the last column of Table 4, where we show the condition number of the linear system solved in the last iteration of NI when the shifting is applied. Clearly, the condition number remains stable even when the drift

Table 5: SM/PH/1 queue - Time (sec) to compute R - $L = 10$

q		0.1				0.01			
load	N	FI	CR	NI	NI-LR	FI	CR	NI	NI-LR
0.1	6254	0.78	90.8	4.78	1.66	2.50	110	4.94	1.74
0.3	1956	0.41	16.1	0.81	0.39	2.14	17.9	0.91	0.45
0.5	1096	0.36	4.13	0.28	0.17	2.30	4.55	0.28	0.19
0.7	728	0.39	3.72	0.19	0.14	2.75	3.89	0.20	0.14
0.9	523	0.75	1.00	0.14	0.09	5.17	1.20	0.16	0.09
0.99	458	5.13	0.92	0.16	0.11	33.4	1.09	0.33	0.11

is very close to one, avoiding the ill-conditioning found in the original system. Therefore, provided that convergence occurs (which is not guaranteed in general when applying the shift technique), this technique can be used to improve the numerical behavior of NI when the drift is close to one. We must point out that, despite the numerical issues just described, the accuracy of the solutions obtained with NI is still very good, in many cases better than CR and FI. The accuracy is measured with the residual error, defined as

$$\left\| \tilde{G} - \sum_{i=0}^{N} A_i \tilde{G}^i \right\|_\infty ,$$

where $\tilde{G}$ is the solution found by the iterative method. In all the experiments performed in this section, even when the drift was very close to one, the residual error for NI was always below 10^{-14}.

6.2 The discrete-time SM/PH/1 queue

We now consider a different queueing example, where the arrival process is described as a semi-Markovian process with two states. In the first state the inter-arrival times (IATs) follow a geometric distribution with parameter p, and whenever there is an arrival the process switches to the second state with probability q. While in the second state, the IATs follow a uniform distribution between 1 and L, and whenever there is an arrival the process jumps back to the first state with probability q. Also, the service times follow a Negative Binomial distribution with parameters (k, γ), where k is a positive integer and γ a real number between 0 and 1. This distribution falls within the class of PH distributions of order k, hence this is an SM/PH/1 queue that can be modeled as a GI/M/1-type MC using the age process described in [11]. To that end we define the matrices $D(n)$, for $n \geq 1$, the (i, j)-th entry of which holds the probability that, given that the arrival process is in state i, the next arrival will occur in n time units and after the arrival the underlying process will make a transition to state j. In our case these matrices are given by

$$D(n) = \begin{bmatrix} (1-p)^{n-1}p(1-q) & (1-p)^{n-1}pq \\ \frac{q}{L}\mathbf{1}\{1 \leq n \leq L\} & \frac{1-q}{L}\mathbf{1}\{1 \leq n \leq L\} \end{bmatrix}, \ n \geq 1,$$

Table 6: SM/PH/1 queue - Time (sec) to compute R - $L = 50$

q		0.1				0.01			
load	N	FI	CR	NI	NI-LR	FI	CR	NI	NI-LR
0.1	63635	8.48	3600	92.7	26.4	34.5	3629	113	33.1
0.3	20653	5.30	917	15.4	6.00	33.8	919	15.5	5.81
0.5	12057	5.27	231	4.86	2.38	38.6	232	4.92	2.39
0.7	8373	6.50	230	2.69	1.45	52.3	232	3.02	1.58
0.9	6326	13.9	75.3	2.41	1.28	108	75.5	2.89	1.39
0.99	5675	100	75.2	3.13	1.41	755	75.2	3.13	2.56

where **1** is the indicator function. Now, letting (α, T) be the parameters of the PH distribution of the service times, the blocks of the GI/M/1-type MC that describe the age process of this queue are given by

$$A_0 = I_{m_a} \otimes T, \; A_i = D(i) \otimes t\alpha, \; i \geq 1,$$

where $m_a = 2$ as the arrival process has only two states. The block size is thus $m = m_a m_s = 2k$, where k is the number of stages in the Negative Binomial distribution. The service rate in this queue is $\mu = \gamma/k$, and the arrival rate can be found to be $\lambda = \frac{4p}{2+(L+1)p}$. As the load is defined as λ/μ, for a given k and L we can match a predefined load by adjusting p adequately. To find the matrix R of this chain we actually solve the dual M/G/1-type MC, relying on Ramaswami's duality result [23].

One of the most salient characteristics of this MC is that many blocks are required to make $a = \sum_{i \geq 0} A_i e$ numerically stochastic. In this particular case, if we want a to differ from e by less than ϵ, we need $N = \left\lceil \frac{\ln(\epsilon)}{\ln(1-p)} \right\rceil$ blocks. This is typically a very large number, especially under low loads, where p must be small enough to match the load. For this queue we consider two instances. In the first one, we set $L = 10$, q can be either 0.1 or 0.01, and we let the load take a broad range of values. The results are included in Table 5, where we also show the number of blocks N required for a to be numerically stochastic, with $\epsilon = 10^{-14}$. As expected, under low loads the number of blocks is very large, which makes CR behave very poorly, while FI outperforms the other general-purpose methods. However, as the load increases, NI becomes the best of these algorithms, and the performance of CR improves significantly. Also, a smaller q, which implies longer sojourn times in each state of the arrival process and the spectral radius of R to be closer to 1, worsens the performance of all the methods, but especially that of FI under high loads. While CR and NI are both less sensitive to this change, NI still outperforms CR. In addition, the blocks $\{A_i, i \geq 1\}$ can be written as $A_i = (D(i) \otimes t)(I_{m_a} \otimes \alpha) = \hat{A}_i \Gamma$, a feature that the Newton method can exploit, as explained in Section 5. The results for this method are shown under the label NI-LR, and illustrate the gains that can be obtained by exploiting this property. In this case the rank r of the blocks is $r = m_a = 2$ and the ratio m/r is only $k = 5$. In spite of this small ratio, NI-LR provides additional reductions compared to NI, and actually outperforms all the other methods in all but a

Table 7: Time (sec) to compute R - Production-Inventory system

Case	0				1		2		3	
SCV	FI	CR	NI	NI-LR	FI	NI-LR	FI	NI-LR	FI	NI-LR
1	0.33	5.77	2.27	0.11	338	280	448	281	539	328
5	0.75	7.91	2.56	0.13	590	327	660	326	731	326
10	1.28	10.2	2.59	0.14	888	326	933	324	993	326
20	2.33	13.5	2.91	0.13	1463	326	1487	324	1527	372

single instance.

The second case only differs from the previous one in that L is set to 50, instead of 10. This change however has a large effect in the total number of blocks, as can be seen in the third column of Table 6. The larger number of blocks affects all the methods, but CR is particularly sensitive to this increase, and even its accuracy becomes compromised as the residual errors become as large as 10^{-6}. The residual error in this case is measured as

$$\left\| \tilde{R} - \sum_{i=0}^{N} \tilde{R}^i A_i \right\|_\infty ,$$

where $\tilde{R}$ is the solution found by the iterative method. On the other hand, FI and NI perform best under low and high loads, respectively, and their accuracy does not worsen with the increase in the number of blocks, with their residual errors being always less than 10^{-14}. Also in this case, in spite of the small m/r ratio, NI-LR is able to provide a significant reduction in computation time.

6.3 A production/inventory system with periodic review

Our last example is an integrated production and inventory model, first introduced in [6]. In this model, a retailer faces a demand and uses a smoothing replenishment policy to manage its inventory. Therefore, depending on the demand and a smoothing parameter β, the retailer places orders to the production facility, which works under a make-to-order policy. This system can be described with a GI/M/1-type MC that keeps track of the age of the order currently under production. This MC has only two nonzero blocks: A_0 and A_N, where N is the deterministic order inter-arrival time (due to the periodic review policy). For a detailed description of the model we refer the reader to [6], and here we simply focus on analyzing four cases, three of which were originally considered in [6]. These cases differ in the distribution of the demand, which, among others, affects the block size. We first consider a simple example, called Case 0, where the demand takes values in the set $\{5, 6, 7, 8\}$ with probability $\{0.2, 0.3, 0.3, 0.2\}$. We set N equal to 16, obtaining a load of 0.8125, and a block size of 94. Also, in this and all the other cases the smoothing parameter β is equal to 0.4. Since the load is fixed, we consider various values for the SCV of the production time per unit, a parameter that has a similar effect as the load: a larger SCV pushes the largest eigenvalue of the matrix R closer to one.

The results are shown in Table 7, where we observe that FI outperforms the other methods, a result of the not-too-high load, and the large block size, which affects the performance of NI. In this example, the block A_N can be written as $\hat{A}_N\Gamma$, and for this case Γ has $r = 7$ rows, and therefore the ratio m/r is equal to 9.42. As a result, an important gain is obtained with the method from Section 5, here labeled NI-LR. In fact, this method performs not only better than NI, but also better than FI for every value of the SCV. Table 7 also includes the results for Cases 1 to 3, as defined in [6], where the demand takes values between 1 and 20, in each case with a different distribution. In these three cases the block size is 838, and Γ has $r = 39$ rows, thus the ratio m/r is equal to 21.49. Due to the large block size, both CR and NI are unable to run on a 2GB RAM machine. Also, FI was modified to take advantage of all the blocks $\{A_i, 1 \leq i \leq N-1\}$ being equal to zero. None of the other methods was modified for this purpose, although both NI and NI-LR can be specialized to this end. However, that due to the large block size we expect NI to feature very long computation times. In spite of this asymmetry, we observe that NI-LR is able to outperform FI in all the instances considered, and while FI's performance worsens significantly when the SCV increases, NI-LR's times remain comparatively stable. In the most variable cases (SCV = 20), FI takes about 25 minutes to compute R, while NI-LR requires around five minutes only, a very important gain, especially if many scenarios need to be analyzed. Although the results presented in Table 7 were obtained with a specific value of the smoothing parameter, namely $\beta = 0.4$, we have considered several values for β and found that the behavior shown here holds in most cases. However, if the value of β is very small, e.g. $\beta = 0.01$, the computation times with FI can easily exceed the hour, even when the SCV is equal to one. On the other hand, even in these extreme instances the performance of NI remains very stable, with computation times between 5 and 6 minutes.

A Proof of Theorem 2

For Theorem 2 to follow from the Monotone Newton Theorem in [18, 13.3.4], we must show that the operator $\mathcal{F}$ and its derivative $\mathcal{F}'$ comply with certain requirements. That is the purpose of the lemmas to be introduced in this section, which themselves rely on several results in [12], [15] and [18]. We start by introducing the Gateaux derivatives of $\mathcal{L}$ and $\mathcal{H}$ as

$$\mathcal{L}'(X)H = (I-X)^{-1}H(I-X)^{-1}A_0,$$
$$\mathcal{H}'(W)H = \sum_{i=2}^{N} A_i \sum_{j=1}^{i-1} W^{j-1}HW^{i-1-j}.$$

Clearly,

$$\mathcal{L}'(X) \leq \mathcal{L}'(Y), \text{ for } X \leq Y \text{ in } \langle 0, U\rangle, \tag{15}$$
$$\mathcal{H}'(V) \leq \mathcal{H}'(W), \text{ for } V \leq W \text{ in } \langle 0, G\rangle. \tag{16}$$

From [12, Lemma 4.2] we know that $\mathcal{L}'$ is Lipschitz-continuous and uniformly bounded in $\langle 0, U\rangle$. Therefore, by [18, 3.2.8], $\mathcal{L}'$ is the Fréchet derivative of $\mathcal{L}$, and, by [18, 3.1.6], $\mathcal{L}$ is also Lipschitz-continuous. Also, the Lipschitz-continuity and uniform boundedness of $\mathcal{H}'$ has been established in [12, Lemma 4.1], and therefore $\mathcal{H}'$ is the Fréchet derivative of $\mathcal{H}$. Since both $\mathcal{L}'$ and $\mathcal{H}'$ are Fréchet derivatives, from [18, 3.1.7] the chain rule applies, and we can define the Fréchet derivative of $\mathcal{F}$ as $\mathcal{F}'(X) = \mathcal{I} - \mathcal{H}'(\mathcal{L}X)\mathcal{L}'(X)$. We now establish the following result regarding $\mathcal{F}$ and $\mathcal{F}'$.

Lemma A.1. *The operator $\mathcal{F}$ is order-concave on $\langle 0, U\rangle$, and the operator $\mathcal{F}'$ is antitone and Lipschitz-continuous on $\langle 0, U\rangle$.*

Proof. This proof follows the lines of that of Lemma 4.3 in [12]. Let $X \leq Y$ be two $m \times m$ matrices in $\langle 0, U\rangle$, then

$$\mathcal{F}'(X) - \mathcal{F}'(Y) = \mathcal{H}'(\mathcal{L}Y)\mathcal{L}'Y - \mathcal{H}'(\mathcal{L}X)\mathcal{L}'X \geq 0,$$

which follows from (10), (15) and (16), since both $\mathcal{L}Y$ and $\mathcal{L}X$ are in $\langle 0, G\rangle$. This shows that $\mathcal{F}'$ is antitone. From this, and [18, 13.3.2], it is easy to see that $\mathcal{F}$ is order-concave since

$$\left(\mathcal{F}'(X) - \mathcal{F}'(Y)\right)(X - Y) \leq 0.$$

The Lipschitz-continuity of $\mathcal{F}'$ follows from the same property of $\mathcal{L}$, $\mathcal{L}'$ and $\mathcal{H}'$, and the uniform boundedness of $\mathcal{L}'$ and $\mathcal{H}'$. To see this, let $X \leq Y$ be two matrices in $\langle 0, U\rangle$, and write, using the l_∞ norm,

$$\begin{aligned} ||\mathcal{F}'(X) - \mathcal{F}'(Y)|| &= ||\mathcal{H}'(\mathcal{L}X)\mathcal{L}'(X) - \mathcal{H}'(\mathcal{L}Y)\mathcal{L}'(Y)||, \\ &= ||\mathcal{H}'(\mathcal{L}X)\mathcal{L}'(X) - \mathcal{H}'(\mathcal{L}Y)\mathcal{L}'(X) + \mathcal{H}'(\mathcal{L}Y)\mathcal{L}'(X) - \mathcal{H}'(\mathcal{L}Y)\mathcal{L}'(Y)||, \\ &\leq ||\mathcal{H}'(\mathcal{L}X) - \mathcal{H}'(\mathcal{L}Y)||\ ||\mathcal{L}'(X)|| + ||\mathcal{H}'(\mathcal{L}Y)||\ ||\mathcal{L}'(X) - \mathcal{L}'(Y)||, \\ &\leq \gamma_1 ||\mathcal{H}'(\mathcal{L}X) - \mathcal{H}'(\mathcal{L}Y)|| + \gamma_2 ||\mathcal{H}'(\mathcal{L}Y)||\ ||X - Y||, \\ &\leq \gamma_1\gamma_3 ||\mathcal{L}X - \mathcal{L}Y|| + \gamma_2\gamma_4 ||X - Y|| \leq \gamma ||X - Y||, \end{aligned}$$

where the second inequality follows from the Lipschitz-continuity and uniform boundedness of $\mathcal{L}'$, the third inequality from the same properties of $\mathcal{H}'$, and the last one from $\mathcal{L}$ being Lipschitz-continuous. □

The next step is to prove that the inverse of $\mathcal{F}'(X)$ exists and is nonnegative for all $X \in \langle 0, U\rangle$. To this end we introduce the operator $\mathcal{T}$ on $\mathbb{R}^{m^2}$, defined as $\mathcal{T}(V)H = \sum_{i=1}^{N} \sum_{j=0}^{i-1} A_i V^j H V^{i-1-j}$. In [15] it was shown that $sp(\mathcal{T}(G)) < 1$ whenever $\rho \neq 1$, where ρ is defined as in Eq. (5). Using this result we can state the following lemma.

Lemma A.2. *The inverse of $\mathcal{F}'(X)$ exists and is nonnegative for all $X \in \langle 0, U\rangle$, if $\rho \neq 1$.*

Proof. Since $\mathcal{F}'(X) = \mathcal{I} - \mathcal{H}'(\mathcal{L}X)\mathcal{L}'(X)$, if we show that $sp(\mathcal{H}'(\mathcal{L}X)\mathcal{L}'(X)) < 1$ for all $X \in \langle 0, U\rangle$, this implies that the inverse of $\mathcal{F}'(X)$ exists, it is equal to $\sum_{k\geq 0}\left(\mathcal{H}'(\mathcal{L}X)\mathcal{L}'(X)\right)^k$ and is therefore nonnegative. And, as for $0 \leq X \leq Y$ we know that $sp(X) \leq sp(Y)$, it is enough to show that

$sp(\mathcal{H}'(\mathcal{L}U)\mathcal{L}'(U)) < 1$. Thus, we write

$$\begin{aligned}
\mathcal{H}'(\mathcal{L}U)\mathcal{L}'(U)H &= \mathcal{H}'(\mathcal{L}U)(I-U)^{-1}H(I-U)^{-1}A_0, \\
&= \sum_{i=2}^{N} A_i \sum_{j=1}^{i-1} (\mathcal{L}U)^{j-1}(I-U)^{-1}H(I-U)^{-1}A_0(\mathcal{L}U)^{i-1-j}, \\
&= \sum_{i=2}^{N} A_i \sum_{j=1}^{i-1} G^{j-1}(I-U)^{-1}HG^{i-j}, \\
&= \sum_{i=1}^{N} A_i \sum_{j=0}^{i-1} G^{j}(I-U)^{-1}HG^{i-j-1} - \sum_{i=1}^{N} A_i G^{i-1}(I-U)^{-1}H, \\
&= \mathcal{T}(G)(I-U)^{-1}H - U(I-U)^{-1}H = (\mathcal{T}(G) - \mathcal{I}U)(I-U)^{-1}H.
\end{aligned}$$

This means that we can write

$$\mathcal{F}'(U) = \mathcal{I} - (\mathcal{T}(G) - \mathcal{I}U)(I-U)^{-1} = (I - \mathcal{T}(G))(I-U)^{-1},$$

which is invertible if and only if $(I - \mathcal{T}(G))$ is invertible, which is itself invertible if and only if $sp(\mathcal{T}(G)) < 1$, and, as mentioned above, from [15] this is true if $\rho \neq 1$. Thus, if $\rho \neq 1$, $(I - \mathcal{H}'(\mathcal{L}U)\mathcal{L}'(U))$ is invertible, and therefore $sp(\mathcal{H}'(\mathcal{L}U)\mathcal{L}'(U)) < 1$, which suffices to prove the lemma. □

B Solving the general linear system

The linear system we are interested in is

$$\sum_{i=0}^{N} B_i X C^i = E, \tag{17}$$

where X is an $m \times r$ unknown matrix, B_i, for $0 \le i \le N$, is a square matrix of size m, C is a square matrix of size r, and E is an $m \times r$ matrix. This is a linear system with mr unknowns and equations, that can be solved in $O(m^3r^3)$ time with general procedures. Thus, such procedures are only feasible for small values of both m and r. We now show the procedure introduced in [20, 21] to reduce the time complexity to $O(m^3r + Nr^3 + Nm^2r^2)$. The key observation to solve this system is that all the matrices that post-multiply the unknown matrix X are powers of the same matrix C. This feature can be exploited by applying a real Schur decomposition [8] to C, i.e., to find an orthogonal matrix $\Theta \in \mathbb{R}^{r\times r}$ such that $\Theta' C \Theta = T$, where $'$ denotes the transpose operator. Recall that a matrix Θ is called orthogonal if $\Theta'\Theta = \Theta\Theta' = I$. The matrix $T \in \mathbb{R}^{r\times r}$ is upper quasi-triangular, meaning it is block upper triangular and the diagonal blocks are of size one or two [8]. By post-multiplying (17) by Θ, and since $\Theta' C^i \Theta = T^i$ for any nonnegative integer i, we obtain

$$-\sum_{i=0}^{N} B_i X \Theta T^i = E\Theta.$$

Now let $F = E\Theta$, which is a known matrix, and let $Y = X\Theta$, to obtain

$$-\sum_{i=0}^{N} B_i Y T^i = F. \tag{18}$$

This system can be equivalently written column-wise as

$$-\sum_{i=0}^{N} B_i \sum_{j=1}^{r} [T^i]_{jk} Y_j = F_k, \tag{19}$$

for $k = 1, \ldots, r$, where M_k and $[M]_{i,j}$ are the k-th column and the (i,j)-th entry of a matrix M, respectively.

In Eq. (18) the matrices that post-multiply Y are all upper quasi-triangular matrices, and all they have the same block structure as they are powers of T. Therefore it is possible to iteratively compute the columns Y_k, starting with Y_1. Let us assume that we have already found $\{Y_1, \ldots, Y_{k-1}\}$ and we want to compute Y_k, for some $1 \leq k \leq r$. Given the upper quasi-triangular nature of T, there are two possibilities. The first is that the entry $[T]_{k+1,k}$ is zero, meaning that Equation (19) can be rewritten as

$$-\sum_{i=0}^{N} B_i [T^i]_{kk} Y_k = F_k + \sum_{i=0}^{N} B_i \sum_{j=1}^{k-1} [T^i]_{jk} Y_j.$$

Therefore we can find the column Y_k by solving a linear system of size m, which requires $O(m^3)$ time. The second case is when $[T]_{k+1,k} \neq 0$, which, due to the upper quasi-triangular structure (the diagonal blocks are at most of size two) implies that $[T]_{k+2,k+1} = 0$. Hence we can find the columns Y_k and Y_{k+1} simultaneously by solving the system

$$-\begin{bmatrix} \sum_{i=0}^{N} B_i [T^i]_{kk} & \sum_{i=0}^{N} B_i [T^i]_{k+1,k} \\ \sum_{i=0}^{N} B_i [T^i]_{k,k+1} & \sum_{i=0}^{N} B_i [T^i]_{k+1,k+1} \end{bmatrix} \begin{bmatrix} Y_k \\ Y_{k+1} \end{bmatrix} = \begin{bmatrix} \hat{F}_k^{k-1} \\ \hat{F}_{k+1}^{k-1} \end{bmatrix},$$

where $\hat{F}_k^l = F_k + \sum_{i=0}^{N} B_i \sum_{j=1}^{l} [T^i]_{jk} Y_j$, for $1 \leq l \leq k-1$ and $1 \leq k \leq r$. This is a linear system with $2m$ unknowns that requires $O(m^3)$ time to be solved. In summary, we can start by finding the first (two) column(s) of Y and iteratively compute the others. After we have computed Y, X is obtained from $X = Y\Theta'$. The Schur decomposition of C and the powers of T require $O(Nr^3)$ time. In addition, we need to compute the matrices that define the r linear systems, at a cost of $O(Nm^2r^2)$ time, while the solution of each of these systems takes $O(m^3)$. The total cost of this approach is therefore $O(rm^3 + Nr^3 + Nm^2r^2)$.

C The RT-CR algorithm

In this section we briefly discuss the RT-CR algorithm presented in [21] to compute the G matrix of an M/G/1-type MC with restricted downward transitions, i.e., for which the $m \times m$ matrix A_0 has only $r < m$ non-zero columns. Without loss of generality we may assume that the first r columns of A_0 are non-zero. State $k \geq 0$ of an M/G/1-type Markov chain is typically denoted as (i, j) with

$i = \lfloor k/m \rfloor$ and $j = (k \mod m) + 1$, where i is called the level and j the phase of the Markov chain. In our setting, this implies that the phase must be part of $\{1, \ldots, r\}$ whenever the level decreases by one. In this case, the G matrix can be written as

$$G = \begin{bmatrix} G_+ & 0 \\ G_0 & 0 \end{bmatrix},$$

where G_+ (resp. G_0) is an $r \times r$ (resp. $(m-r) \times r$) matrix [17]. The RT-CR algorithm first computes the matrix G_+ in an iterative manner and afterwards computes G_0 by solving a linear system of the form $\sum_{i=0}^{N-1} B_j G_0 (G_+)^i = E$, for some $(m-r) \times (m-r)$ matrices B_j and $m \times r$ matrix E that are easy to obtain from A_i and G_+.

The RT-CR algorithm determines the matrix G_+ by first constructing a new MC that observes the original M/G/1-type MC, characterized by the A_i matrices for $i \geq 0$, when its phase is part of the set $\{1, \ldots, r\}$ only. This new MC is also an M/G/1-type MC characterized by the $r \times r$ blocks $\bar{A}_i$, for $i = 1, \ldots, M$, where M is determined numerically such that $\sum_{i=0}^{M} \bar{A}_i$ is numerically stochastic (for details on how to compute the $\bar{A}_i$ matrices we refer to [21]). The $r \times r$ matrix G of this new MC was also shown to be identical to G_+. Hence, to compute G_+ the RT-CR algorithm first constructs the matrices $\bar{A}_i$, for $i = 0, \ldots, M$, and then uses the cyclic reduction (CR) algorithm to compute G_+.

The time complexity to compute G_+ depends to a large extend on M. More precisely, the construction of the $\bar{A}_i$ matrices requires about $O(MNrm^2)$ time, while the CR algorithm uses only $O(d'r^3 + r^2 d' \log d')$, where d' is the numerical degree of the power series evaluated at each step. As indicated in [21], the computation of the matrix G_0 from G_+ can be performed in $O((m-r)^3 r + Nr^3 + N(m-r)2r^2)$ time. Hence, the Newton iteration of Section 3 is very different from the LR-CR algorithm and whether it outperforms the LR-CR algorithm depends to a large extend on the value of M which is often considerably larger than N.

References

[1] N. Akar and K. Sohraby. An invariant subspace approach in M/G/1 and G/M/1 type Markov chains. *Communications in Statistics: Stochastic Models*, 13:381–416, 1997.

[2] D. Bini, G. Latouche, and B. Meini. *Numerical Methods for Structured Markov Chains*. Oxford University Press, 2005.

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

[4] D. Bini, B. Meini, and V. Ramaswami. Analyzing M/G/1 paradigms through QBDs: the role of the block structure in computing the matrix G. In G. Latouche and P. Taylor, editors, *Advances in Algorithmic Methods for Stochastic Models*, pages 73–86. Notable Publications, New Jersey, 2000.

[5] 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.

[6] R.N. Boute, M.R. Lambrecht, and B. Van Houdt. Performance evaluation of a production/inventory system with periodic review and endogenous lead times. *Naval Research Logistics*, 54:462–473, 2007.

[7] J. E. Diamond and A. S. Alfa. On approximating higher order MAPs with MAPs of order two. *Queueing Systems*, 34:269–288, 2000.

[8] G. H. Golub and C. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 1996.

[9] W. K. Grassmann and J. Tavakoli. Comparing some algorithms for solving QBD processes exhibiting special structure. *Information systems and operations research*, 48:133–141, 2010.

[10] C. He, B. Meini, and N.H. Rhee. A shifted cyclic reduction algorithm for quasi-birth-and-death problems. *SIAM Journal of Matrix Analysis and Applications*, 23:673–691, 2001.

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

[12] G. Latouche. Newton's iteration for non-linear equations in Markov chains. *IMA Journal of Numerical Analysis*, 14:583–598, 1994.

[13] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling.* ASA-SIAM Series on Statistics and Applied Probability. SIAM, Philadelphia, PA, 1999.

[14] D. Lucantoni. New results on the single server queue with a batch markovian arrival process. *Stochastic Models*, 7:1–46, 1991.

[15] M.F. Neuts. Moment formulas for the Markov renewal branching process. *Advances in Applied Probability*, 8:690–711, 1976.

[16] M.F. Neuts. *Matrix-Geometric Solutions in Stochastic Models.* The John Hopkins University Press, Baltimore, 1981.

[17] M.F. Neuts. *Structured stochastic matrices of M/G/1 type and their applications.* Marcel Dekker Inc., 1989.

[18] J.M. Ortega and W.C. Rheinbloldt. *Iterative solution of nonlinear equations in several variables.* Academic Press, 1970.

[19] J. F. Pérez and B. Van Houdt. Analyzing M/G/1-type Markov chains with low-rank downward transitions. In *6th International Workshop on the Numerical Solution of Markov Chains (NSMC)*, pages 75–78, Williamsburg (USA), 2010.

[20] J.F. Pérez. *Performance Modeling: Structured Markov chains, optical grids and switches*. PhD thesis, University of Antwerp, 2010.

[21] J.F. Pérez and B. Van Houdt. The M/G/1-type Markov chain with restricted transitions and its application to queues with batch arrivals. *Probability in the Engineering and Informational Sciences (PEIS)*, 25(4):487–517, 2011.

[22] V. Ramaswami. Nonlinear matrix equations in applied probability - solution techniques and open problems. *SIAM Review*, 30:256–263, 1988.

[23] V. Ramaswami. A duality theorem for the matrix paradigms in queueing theory. *Communications in Statistics Stochastic Models*, 6:151–161, 1990.

[24] P. G. Taylor and B. Van Houdt. On the dual relationship between Markov chains of GI/M/1 and M/G/1 type. *Advances in Applied Probability*, 42:210–225, 2010.

[25] M. Telek and A. Heindl. Matching moments for acyclic discrete and continuous phase-type distributions of second order. *International Journal of Simulation Systems, Science & Technology*, 3:47–57, 2002.

[26] W. Whitt. Approximating a point process by a renewal process, I: Two basic methods. *Operations Research*, 30:125–147, 1982.